A Computational Bow-spring Model

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JUN 30 2010

LIBRARIES

by

Blake A. Sessions

B.S. Mechanical Engineering
Massachusetts Institute of Technology, 2011

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE IN MECHANICAL ENGINEERING
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

ARCHIVES

MAY 2010
[February 2011]

©2010 Blake A. Sessions. All rights reserved.

The author hereby grants to MIT permission to reproduce and to distribute publicly paper and electronic copies of this thesis document in whole or in part in any medium now known or hereafter created.

Signature of Author: [illegible]
Department of Mechanical Engineering
May 3, 2010

Certified by: [illegible]
Hugh Herr
Professor, Media Arts and Sciences
Thesis Supervisor

Accepted by: [illegible]
John H. Lienhard V
Collins Professor of Mechanical Engineering
Chairman, Undergraduate Thesis Committee

A Computational Bow Spring Model

by

Blake A. Sessions

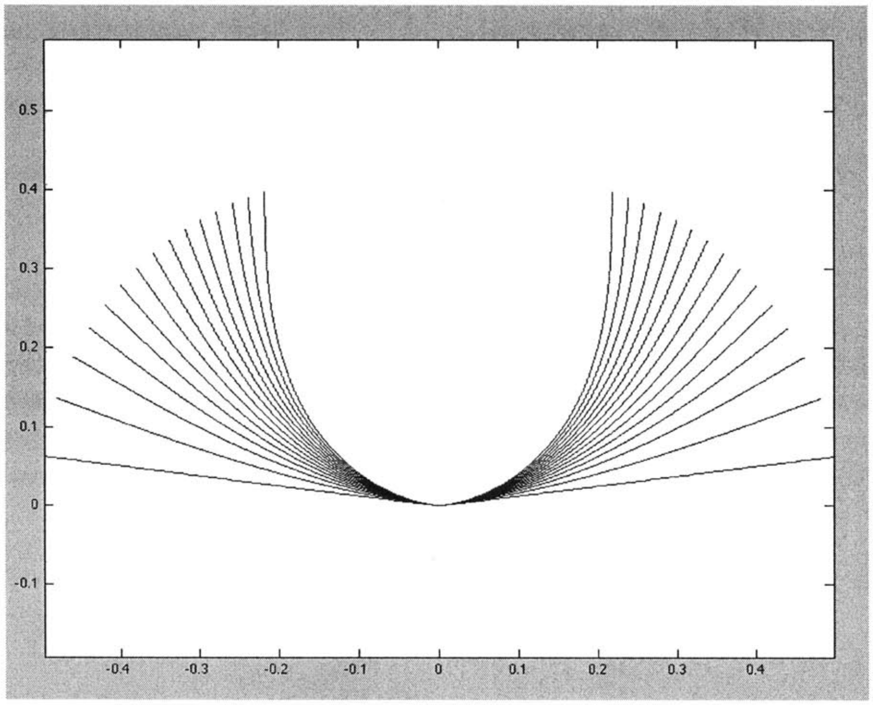
Submitted to the Department of Mechanical Engineering on May 7, 2010 in Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Mechanical Engineering

ABSTRACT

Bow-springs find few applications in industry. Principally, they are used in archery. In addition, they have found some use in a compression-spring mode in the field of biomechatronics, to emulate elastic human legs. The mechanical behavior (characterized by deflected shape and deformation force) is difficult to model, because internal forces and moments and the geometry are both unknown. The only closed-form solutions to such systems are relatively useless to a mechanical engineer.

This work comprises an iterative model developed in MATLAB that computes the mechanical behavior of buckled beam (or bow-spring) sections, over a range of parameters and geometries, to be used in the development and testing of compression bow-springs as parallel loading systems to the human leg.

Thesis Supervisor: Hugh Herr
Title: Professor, Media Arts and Sciences



The set of spatial curves that a thin bow-spring will assume in two-force-member compression, iterated to 60% deformation

Introduction

Bowed members are difficult to analyze mechanically, because the compatibility arguments (and overall geometry) are intricately convolved with the internal forces and moments. Standard mechanical analyses (for example - a beam undergoing a torsional moment, or a cantilevered and loaded beam) of internal forces and moments are typically very straight-forward, and require no more technical aptitude than second-order single variable calculus.

Bowed springs, however, are a different story: The geometry is such that it cannot be simplified into a single differential equation that can be integrated. There is very little established in current literature that correctly defines the shape and force-deformation behavior of a bowed spring (given the geometry, elastic modulus, and other material properties.)

Using standard mechanical analysis techniques, we could start with a compatibility argument. Subjecting our bow of length L to an end-to-end deformation D would cause the bow to curve in order to fit compatibly. Unfortunately, there are an infinite number of spatial curves that the bow could assume to satisfy this compatibility. Only one satisfies *static* compatibility, but this does not give us enough information to solve the problem. The shape is not parabolic in nature, nor is it an arc segment. Thus, we do not know how far the center point of the bow lies from the loading axis (if we knew this, we could then simply find the internal moment and divide by the distance to the loading axis to find the force output.)

This is a classic problem of Calculus of Variations: We are picking the particular spatial curve **r(s)** that results in the minimum elastic energy storage in the beam. Any dynamics or oscillations will be about this equilibrium point. The analytic solution, however, results in an elliptical integral: the only closed form solution to the deformed shape of bow-spring sections. However, the degrees of freedom used to express the state of the elastically deformed section are such that they may not be translated into Cartesian geometry. Although the closed form may be of help intellectually, it is of very little use practically.

This work comprises the development of a numerical system that describes the bow-spring behavior, which can be articulated in two manners. The end-to-end force as it relates to the deformation is incredibly useful to the engineer, because from it we can model the spring as a simple variable impedance spring (as a function of deformation or load.)

In addition, the physical shape that the bow-spring assumes under load is equally useful, because from it we can find the maximum internal moment and curvature in the material (which resides at the center of the spring, if statically loaded.) From these parameters and our simple linear stress-strain techniques, we may find the maximum internal stress in the material and define safety factors to ensure that the material will not fail in application.

Structure of the solution

The initial assumption, from which the rest of the derivation is based, resides in the fact that the internal moment of the bow will be proportional to its displacement from the two-force axis. This is immediately apparent if an internal slice of the beam is taken; at any section, there is an internal moment M which is equal to F*y, where F is the end-to-end spring force, and y is the perpendicular distance to the load axis.

Because the spring force does not change with the location of our cut, I find that $M_{int}(y) = F^*y$, a direct proportionality. The internal moment and the curvature are linearly related, viz.

$$M_{int} = EI^*K$$

So the internal curvature is also proportional to the distance from the load axis,

$$K(y) = (F/EI)^*y$$

When making the compatibility argument, we exploit this proportionality. Because we don't know the end-to-end force F, we settle for an unknown constant:

$$K(y) = B^*y$$

Thus, we find that the curvature at each particular point along the length of the section is proportional to its distance from the load axis.

Iteration can then be accomplished, by subjecting the beam to a curvature which is proportional to y. We must know the initial angle that the beam makes with the load axis (at the load point) in order to iterate from the end of the beam. Because we don't yet know this, we must choose to iterate from the center. By symmetry, we know the initial angle that the beam makes with the load axis: It is the angle defined in the rigid knee, divided by two.

However, we have yet another problem. Starting from the center of the beam, the code must iterate outwards with a curvature proportional to the distance to the load axis. At which y value does the load axis reside? We don't yet know, so we define another arbitrary constant Y_{mo}, the distance from the center of the beam to the load axis.

If guesses of "B" and "Y_{mo}" are made, we can find a particular bow spring curve; one that could possibly exist, *somewhere.* The geometry varies as a function of B and Y_{mo} , which are the initial parameters for the iteration. Lowercase x,y represent the point locations of the bow in cartesian space, with the origin set at the midpoint of the bow. Lowercase "l" is the length of the bow from the center-point as the code iterates, otherwise often called "s" for path length. Theta is the angle that is made with the horizontal axis. The capital versions of these variables are storing the instantaneous variables as vectors for later use. The interior-most loop continues to iterate until the curve reaches Y_{mo}, the loading axis.

There exists a major problem with this iteration, however. Although it has produced a bow-spring curve, it is not *the* curve that is desired. The length and the deformation of the bow-spring vary as a function of B, Y_{mo}. I prefer to use non-dimensional deformation, defined as e = d/L; the spring deformation divided by the length of the spring. Then:

$$(e, L) = F(B, Y_{mo})$$

Where each variable e,L has a dependence upon the other two variables B, Y_{mo}. A highly non-linear system, the dependent variables e,L cannot be put into a linear matrix form. Thus, we resort to another method: we assume that the non-dimensional shape of bow-springs remains constant. If we have two bow springs, one of length L with a deformation d and one of length 2*L and a deformation 2*d, the spatial curve that the latter assumes will be a linear scale factor of 2 of the initial geometry. This can be seen from an energetics perspective; if we proceed to attack the problem with a calculus of variations approach (as stated in the introduction), dimensionless parameters pop out of the system that are linear scales of the contained elastic energy, and are not dependent upon the non-dimensional shape.

We then define "ed", the desired non-dimensional deformation. The code runs through an array of Y_{mo} until the curve satisfies the non-dimensional shape requirement. Each time, it resets the state variables x, y, l, and theta. Once the outer iteration loop hits the correct ed value, the bow-spring curve has been non-dimensionally defined. It then take the vectors X and Y (which have been storing the instantaneous values), and scale them accordingly so that the curve assumes its proper size.

The next lines of code are geometric arguments to find the internal curvature of the beam, at the center point (for every deformation and length, one value of maximum curvature is defined.) Internal moments and end-to-end forces are easily derived from this set of data. The outermost loop runs through a deformation vector and stores the information from each loop iteration. Finally, some offsetting code is implemented at the end to make the F-d relation more apparent and to cut off the "tail."

As further proof of the non-dimensional assumption and scaling factor argument, we find that the force-deformation curve is *not dependent* upon the initial scaling factor, B. B can be defined from 0.1 to 5 with little effect; small errors of order 1% are apparent due to the larger steps of iteration. Thus, <u>the initially defined parameters B, Y_{mo} do not</u>

affect the output curve, which is what we expect. The "inputs" have been iterated away to become intermediate variables to the force output.

Using the code

Inputs are:

EI	bending stiffness (N^*m^2)
Ltot	end-to-end length of bow-spring (m)
dmax	the maximum deformation (m)
KAngle	the knee angle, or intersection angle of unstressed springs (radians)
n	number of iterations for the force-deformation curve.

Outputs are:

def	end-to-end deformation (m) (vector)
Fout	end-to-end force (N) (vector)

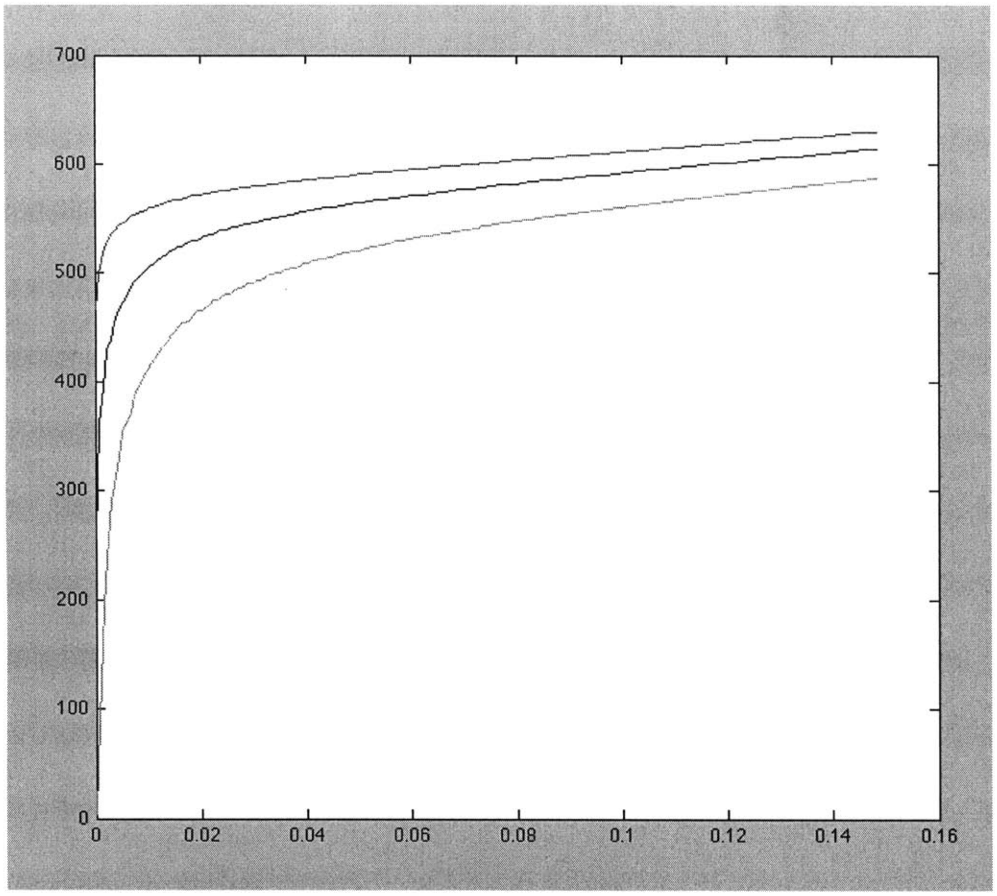
(d,F) plotted by default.

Remarks

Intuitively, we expect a few things from bow-spring behavior. We expect that the output force will be linearly related to the bending stiffness. For a defined deformation "d", we expect the output force to decrease as we choose larger and larger spring sections, because the non-dimensional shape gets progressively "less bowed" as we choose larger springs. We expect the curve to be independent of the number of deformation iterations. All of the above can be seen in the model.

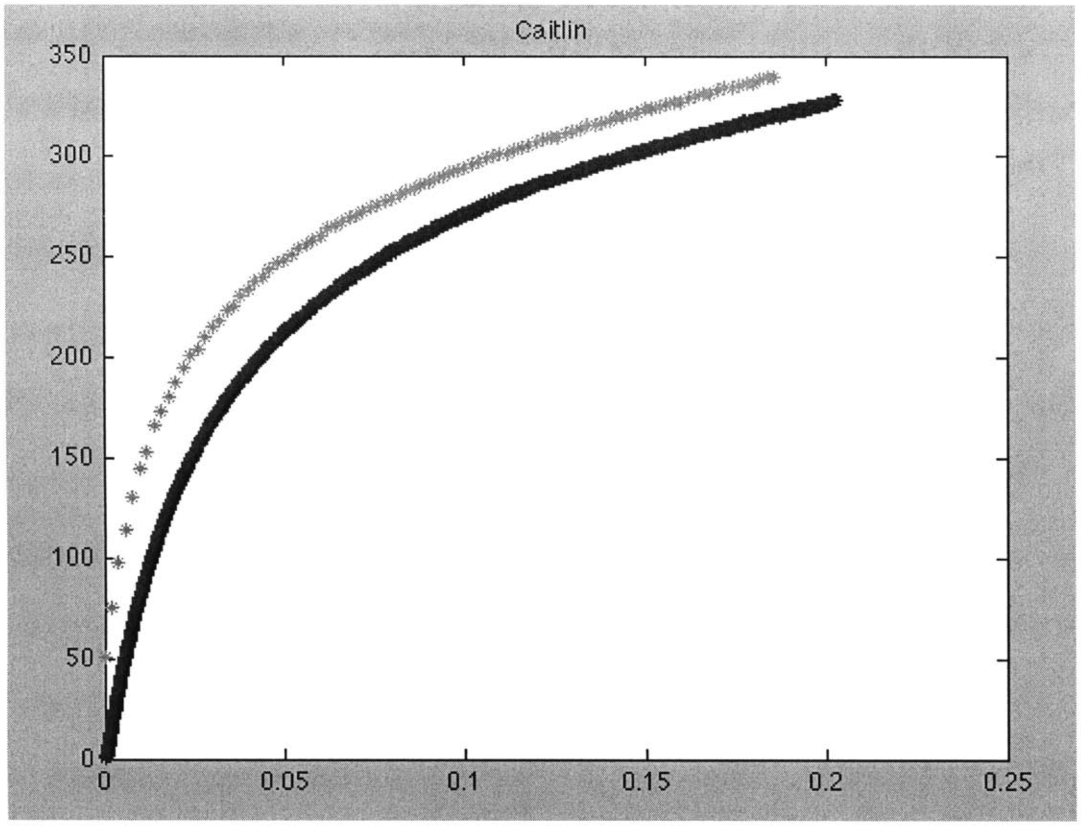
The effect of the knee angle upon the force-deformation relation is more interesting. With a shallower (closer to parallel) knee angle, we expect a spring that is initially stiffer. Furthermore, we expect that the behavior of a set bow-springs with the length held constant and the knee angle varied to converge. At low deformations, a 10-degree knee angle will result in a very stiff leg, because our initial force input will be resisted at a small radius, which yields a small internal moment and a small curvature. A 40-degree knee angle will result in a substantially increased compliance, for a similar reason. However, at large deformations, the compliances of the 10-degree and 40-degree systems should converge; Both springs have been displaced substantially from the neutral axis, and the "memory" of the system has lowered at large deformations. In other words, the compliance has a large functional dependence upon knee angle at low deformations, and a negligible dependence upon knee angle at high deformations.

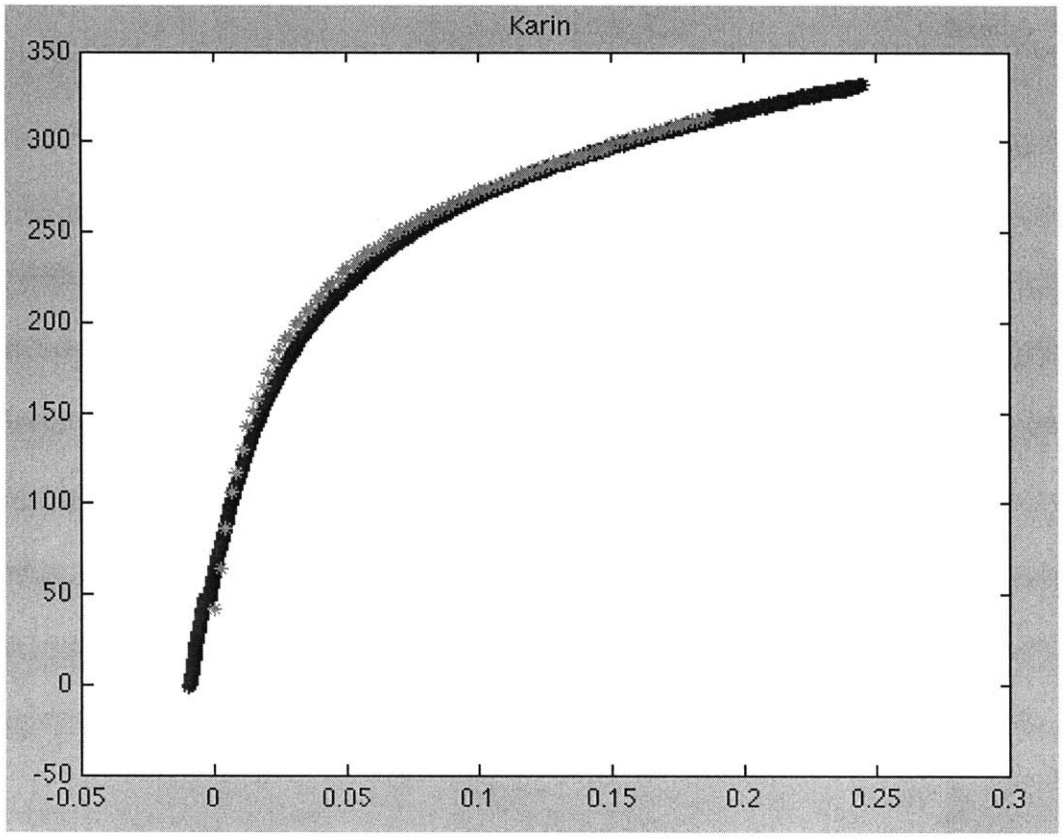
Below are three force-deformation plots, with initial parameters EI = 120, L = 1, dmax = 0.15, and three different knee angles KAngle = 0.02,0.05,0.1 radians. Notice that the compliances (values represented by the initial slope) are drastically different at small deformation, and converge to the same value at large deformation. (units: X - meters, Y - Newtons.)

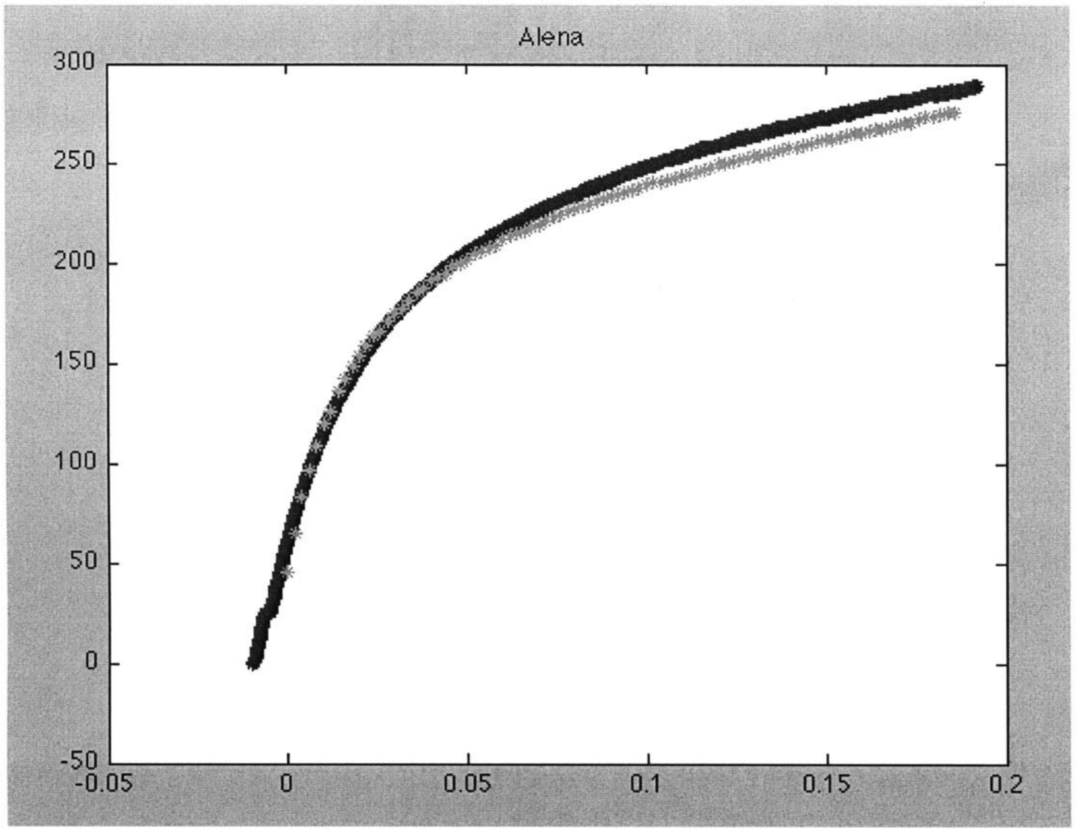


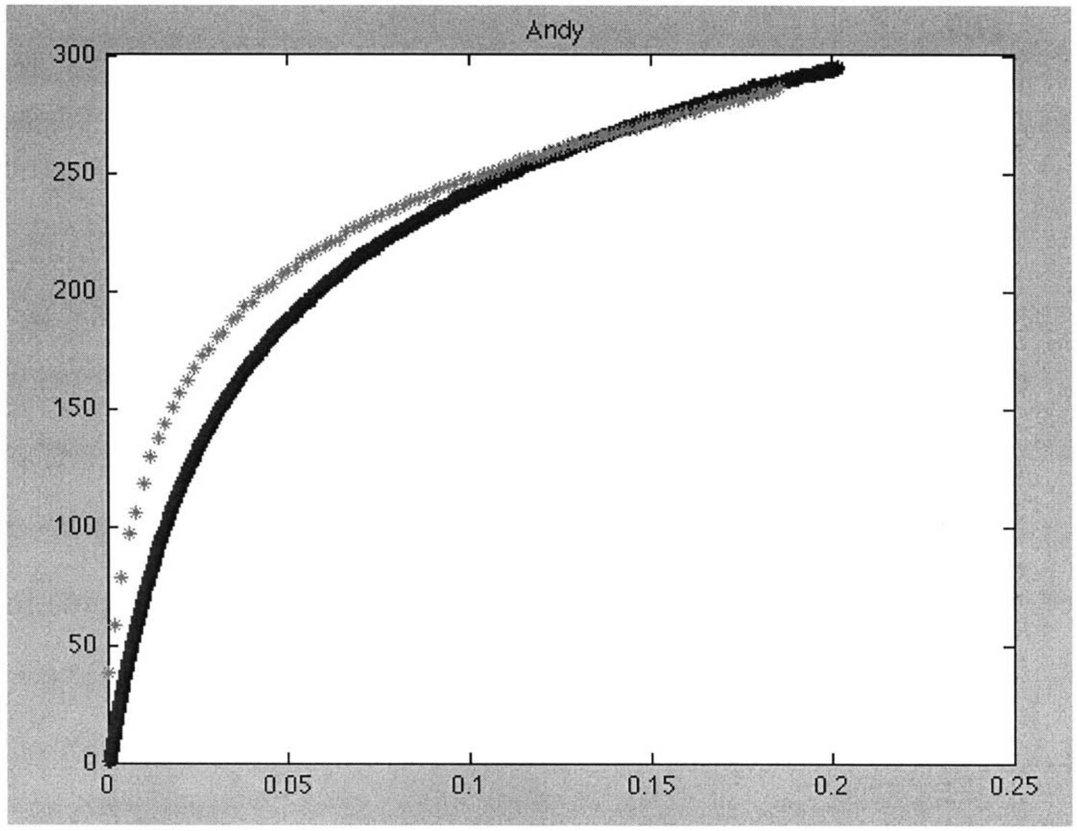
Comparisons

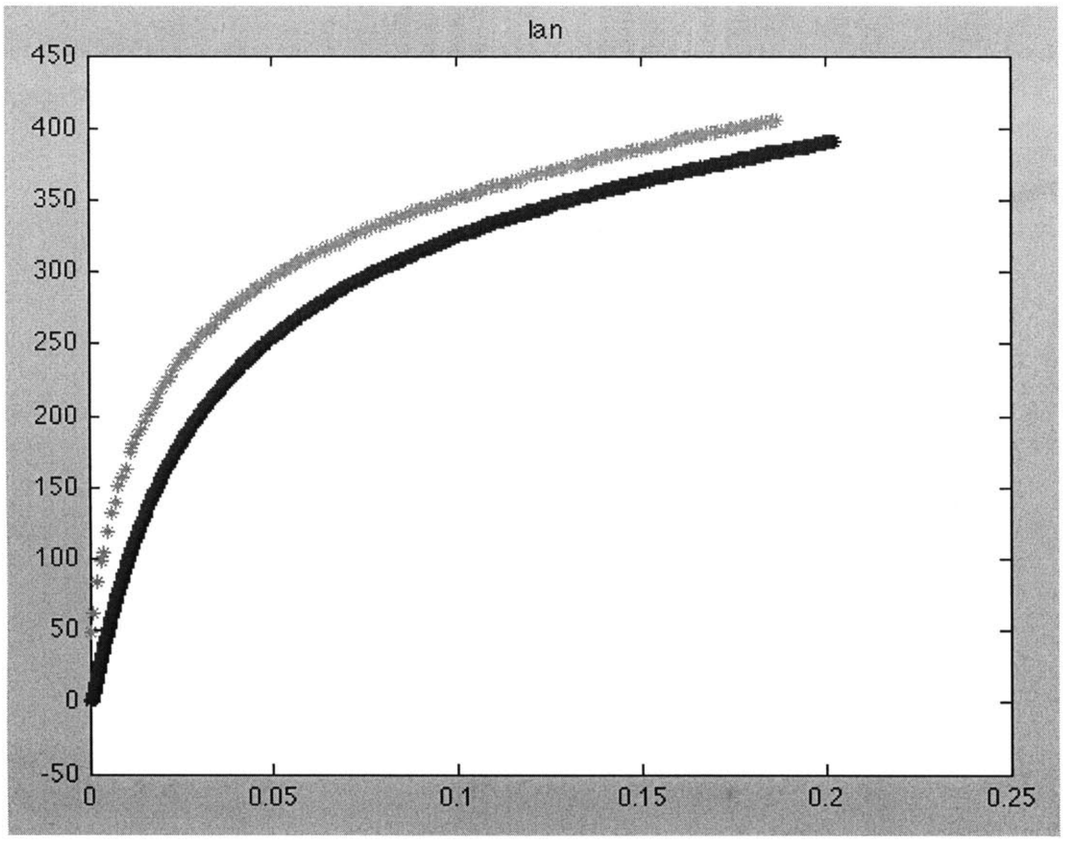
Alena and Beth tested the force-deformation behavior of various sizes of bow-spring. Using the model, the inputs required for a force-deformation curve are bending stiffness, length, deformation range, and knee angle. With the proper parameters implemented, the following curves result for the bow-springs used in the hopping tests of the following people: Alena, Caitlin, Hazel, Andy, Ian, and Karin. Data represents a variety of spring thicknesses, from 0.25" to 0.31", and a range of spring lengths from 33" to 37." All plots have units X - meters, Y - Newtons. Green: model, Blue: measured.

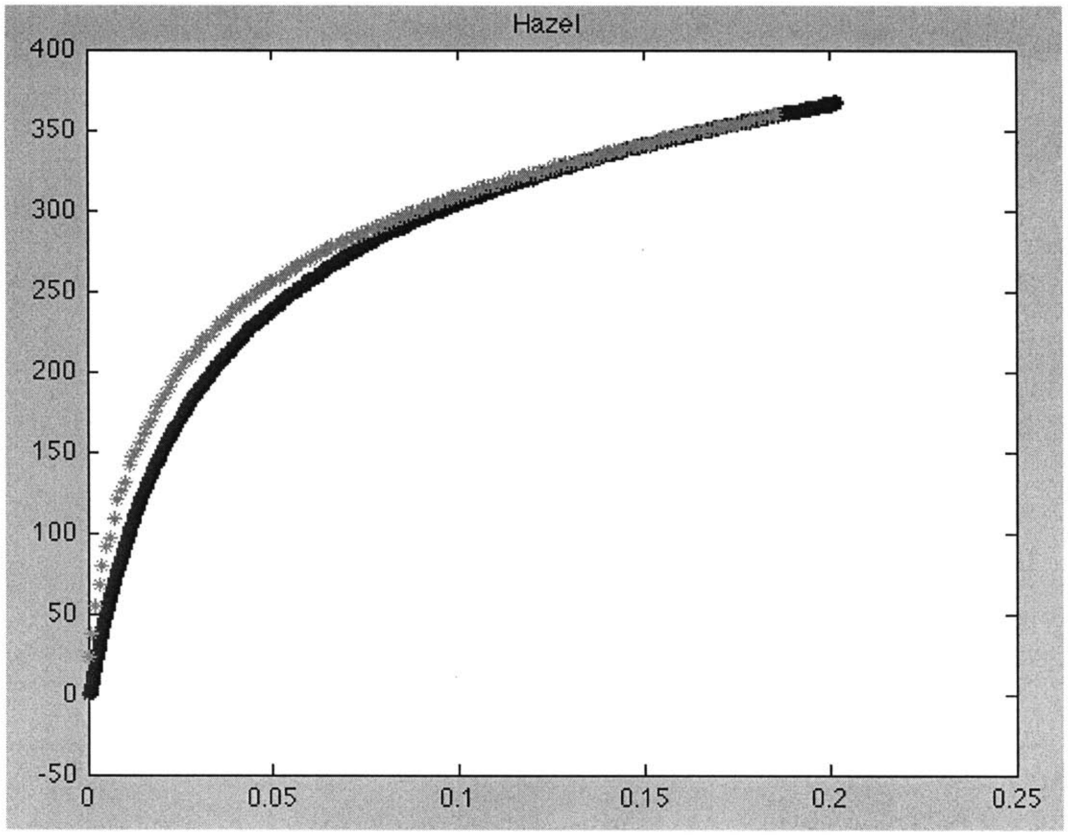

Caitlin
350
300
250
200
150
100
50
0
0
0.05
0.1
0.15
0.2
0.25


Karin
350
300
250
200
150
100
50
0
-50
-0.05
0
0.05
0.1
0.15
0.2
0.25
0.3


Alena
300
250
200
150
100
50
0
-50
-0.05
0
0.05
0.1
0.15
0.2


Andy
300
250
200
150
100
50
0
0
0.05
0.1
0.15
0.2
0.25


Ian
450
400
350
300
250
200
150
100
50
0
-50
0
0.05
0.1
0.15
0.2
0.25


Hazel
400
350
300
250
200
150
100
50
0
-50
0
0.05
0.1
0.15
0.2
0.25

Error Analysis

There exist three principal sources of error. The first is a non-linearity in the bending stiffness of the beams; for all of these loadings, we assume an elastic loading with a stress proportional to the distance to the neutral axis. In reality, this isn't quite the case, and the force-deformation curves don't scale linearly with the bending stiffness, EI. Generally, the linearity dies away as we get farther from the neutral axis of the bent beam; thus, the internal stress distribution will not produce as great an internal moment as the model predicts for any particular defined curvature. This will result in a spring which, in reality, will exhibit a lower force output. These errors are particularly apparent in the cases with thicker springs (Caitlin, Ian, Andy.)

Secondly, the model assumes elasticity up to the center-point of the bow. Every part of the bow produces a curvature when loaded. In reality, we have a rigid knee locking the two carbon fiber sections together, which has a mechanical impedance that is much greater than the springs themselves. When we load the beam, this rigid section will not deform as the model predicts, and will have a greater stiffness in reality than in the predicted model. Particularly for the clutched system (whose knee is large,) the leg will exhibit a much higher F-d curve in reality.

Finally, error can be attributed to the asymmetry of the leg. The model predicts a perfect symmetry of the bowed section, and in reality the knee is off-center. We may derive some insight into this effect by looking at the end condition, as the knee nears one of the pivots. In this case, the material distribution (with the knee angle held constant) will be much closer to the loading axis, and it will be harder to deform outwards. Thus, the spring in reality will exhibit a higher impedance when the knee is off-center. With these considerations in mind, the model still yields a result that is generally accurate to ~10% or less, and may prove to be useful in the future.

Appendix A - MATLAB code

```
%BOW SPRING MODEL. UNITS IN SI UNLESS NOTED.

EI = 66; %Bending stiffness. N*m^2
Ltot = 1; %BS length, end to end.
dmax = 0.3;
KAngle = 0.25; %From horizontal to one side length.
n = 10; %number of linear deformation iterations




thetao = KAngle/2;
i = 1;
Ld = Ltot;
def=0;
Fout=0;
B = 0.5; %Arbitrary initial constant.

for ed = 0.0001/Ld:dmax/(n*Ld):dmax/Ld; %desired non-dimensional
deformation.
e = 0; %for the first iteration

Ymo = 0;
 X = 0;
    Y = 0;
    L = 0;
    Theta = 0;
while e <= ed
    Ymo = Ymo + 0.01;
    x = 0;
    y = 0;
    l = 0;
    theta = thetao;
    dL = 0.001;
    j = 1;

        while y <= Ymo;
        %DIFFERENTIALS
        dtheta = B*(Ymo-y)*dL;
        dx = dL*cos(theta);
        dy = dL*sin(theta);

        %UPDATING STATE VARIABLES
        theta = theta + dtheta;
        x = x + dx;
        y = y + dy;
        l = l + dL;
        X(j) = x; %FOR plotting.
        Y(j) = y;
        Theta(j) = theta;
        L(j) = l;
        j = j+1; %FOR NEXT ITERATION w/ unknown length

        end



    e = 1-x/l;



end

%SCALING
X = X*Ld/(2*l);
Y = Y*Ld/(2*l);
L = L*Ld/(2*l);

%CURVE HAS BEEN DEFINED COMPLETELY AS A FUNCTION OF DEFORMATION.
%Finding the internal curvature. using points 2,3,4.

dx1=X(1,26)-X(1,2);
dx2=X(1,50)-X(1,26);
dy1=Y(1,26)-Y(1,2);
dy2=Y(1,50)-Y(1,26);
phi1 = atan(dy1/dx1);
phi2 = atan(dy2/dx2);
dphi=phi2-phi1;
diffle1 = sqrt(dx1^2+dy1^2);
diffle2 = sqrt(dx2^2+dy2^2);
diff=(diffle1 + diffle2);
curvature = dphi/diff;



%plotting the curve of the bowspring, scaled properly.
    plot(X,Y,'b')
    hold on
    plot(-X,Y,'b')
    hold on
    axis equal

    %for calculating internal moment.
    Mint(i)=EI*curvature;
    Yend(i)=Y(end);
    i = i+1;

end

deformation = 0.0001:dmax/n:dmax; %Deformation output vector.
F = Mint./Yend; %Force output vector.

%Compensating for the tail. find the number of pts before def.
force, then
%define new vectors.
j = 1;
while F(j) <= 1 %Discard force.
    Fout(1) = F(j);
    j = j+1;
end

%Reassignment.
for k = 1:1:(n-j)
    Fout(k)=F(k+j);
    def(k) = deformation(k+j);
    internalmoment(k) = Mint(k+j);
end


%Shifting curve to zero point.
offset=def(1);
def = def - offset;

%F-d plot.
%plot(def,Fout,'r*')
%hold on
```