**Computational Lab 2:**

**Simulations of the Stress-Strain Properties of Wavy Parallel Springs**

**Matthew Mirek**

**Due: 10/14/2016**

**Extracellular Matrix: BE549**

**Professor Suki**

**Introduction & Theory:**

In tissue, springs usually are modeled as nonlinear to be characterized correctly. In the specific case of collagen, it is necessary to model them as varying length springs in parallel so they demonstrate recoupment. In this lab a simplified model of N nonlinear springs in parallel will be used as shown in *Figure 1.* By developing a Matlab code to calculate the displacement, energy, force, and stress strain curve the behavior of the spring system can be analyzed, and an intuitive sense of how springs in parallel of varying lengths behave can be achieved. ­

Figure 1: Three springs in parallel­

For this lab, the energy of each spring can be determine from *Equation 1*, where the variable is the displacement of the individual spring. The Matlab code uses this equation by taking the derivative of it, and summing all of the individual forces that each spring adds to the entire system.

(i=1,2,…N)

Equation 1: Energy of Non-linear Spring

**Matlab Code Setup:**

Instead of creating individual functions, a parallel spring class was created. By using classes in Matlab, an object can be created that has all of the properties and variables of a parallel spring system. Then by referencing the object in Matlab, integrated functions can be called to display Force, Energy, and the Stress-Strain Curve. In *Figure 2*, an example of how a spring system can be initialized is shown. *Figure 3* shows how the parallel spring class can be used. The class code is attached the appendix of this lab.

W

A1

B1

C1

N

Type of length distribution

µ 



Figure 2: Initialization of the Parallel Spring Class

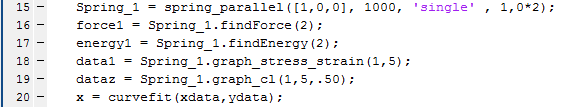
­­ ­­

Figure 3: Utilization of Parallel Spring Class

**Questions:**

**1)**

In this scenario, a simple spring of a multiple N = 1000 was plotted from the stress strain curve. It can be seen that slope is equal to 1 in this case. Therefore from *Equation 2-5*, it can be deduced that since the slope was 1, the initial length was 1, and the area equaled a 1000, the effective stiffness was 1000.

(2)

(3)

(4)

(5)

This can also be stated in a more general sense, where in the linear spring case. This problem can also be solved analytically, by summing up all of the contributing forces as seen by *Equation 5*. This yields the same result, assuming that the case is linear springs, constant length.

(6)



Figure 4: Input for Simple Case of Constant Length, Linear Spring

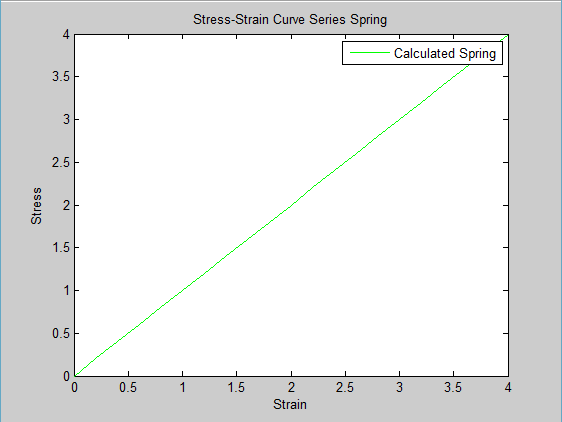


Figure 5: Linear System Initial Calculation and Result

**2)**

This portion of the lab utilized linear springs with a uniform random distribution. *Figure 6* displays all of the inputs for generating the spring systems. In the case that the mean of the distribution is 1.5 and the half width is .1, *Figure 7* shows that the distribution of values stay within the range of 1.4 and 1.6. For *Figure 8-9*, the stress – strain curves are affected by the distribution. As it can be seen, with larger distributions contributions to the total force with earlier than smaller averages when the mean of distribution stays constant. When shifted to a mean of 3, and a narrow distribution of lengths forces are a significant amount of strain is required before a force is generated. However it should be noted that as with larger strains, the curves collapse into a single straight line. Every spring system model with a mean of 1.5 collapses into one curve, while the spring systems with a mean of 3 collapse into a different curve. This is probably due to the distributions having the most effect in initial stages of stretching, once the upper bounds of the distribution is passed, each spring is contributing force to the system.

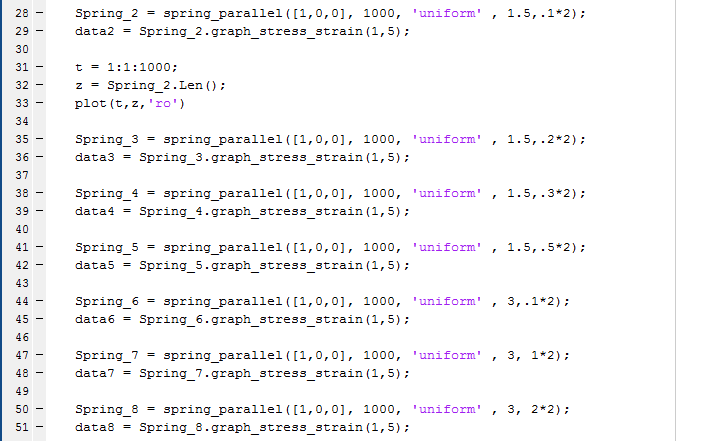


Figure 6: Initialization of springs for Problem 2

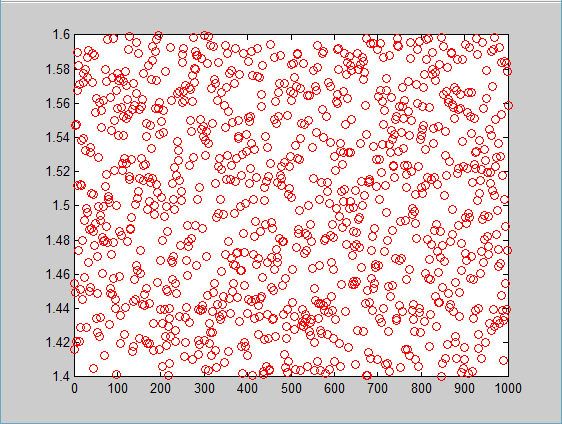


Figure 7: Verification that Distribution of Lengths are between 1.4-1.6

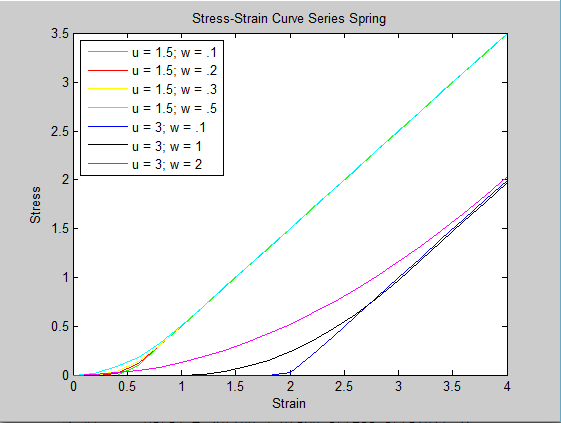


Figure 8: Stress Strain Curve for Distributed Lengths/Linear Springs

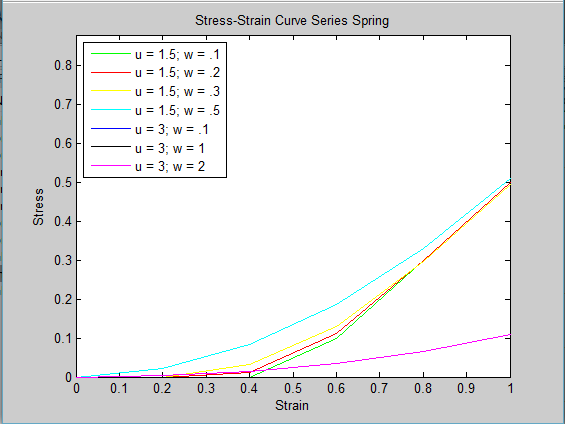


Figure 9: Stress Strain Curve for Distributed Lengths/Linear springs (close up)

**3)**

*Figure 10 and 11*, demonstrate the implementation of a nonlinear spring system with consistent lengths. Since the lengths are the same case, the stress-strain shows a generic nonlinear spring. *Figure 12 and 13*, show the same distribution values used in problem 2 with nonlinear coefficients. The nonlinear coefficients seem to stop the curves from converging for the same mean. This can be clearly seen in the mean = 3 case, and a possible explanation for this phenomena is that the with large strain the stiffness of the springs drastically increase so any initial offset in the stress strain curve is held through later increments of strain.



Figure 10: Implementation of Constant Length, Nonlinear Spring Case

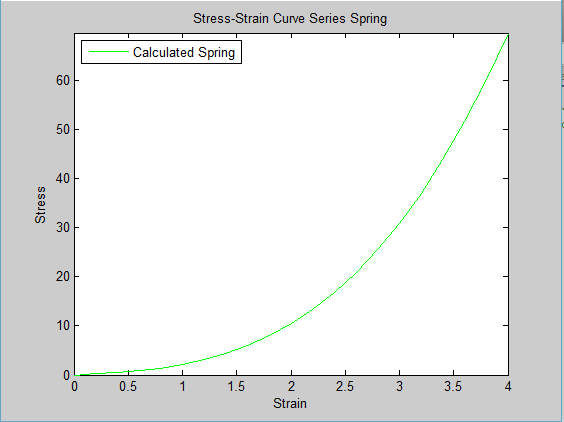


Figure 11: Single Length Case with Nonlinear Springs

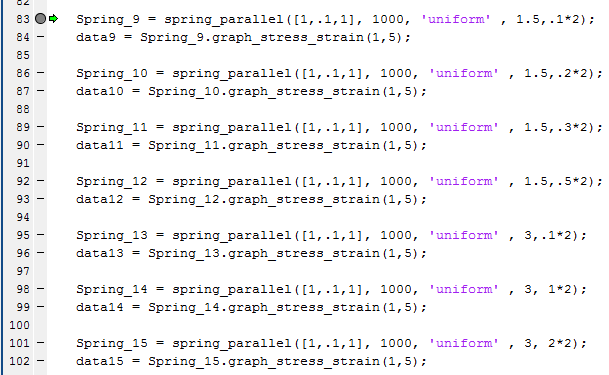


Figure 12: Distributed Length Case with Nonlinear Springs

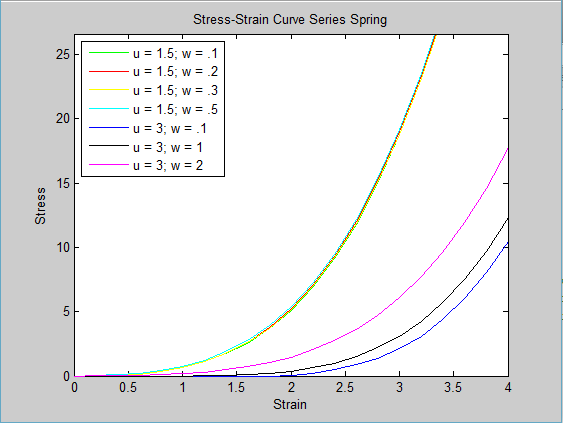


Figure 13: Distributed Length Case with Nonlinear Springs Curve

**4)**

To implement this problem, a different parallel\_spring class was implemented where the distribution ci was included. *Figure 14*, shows this implementation, while *Figure 15* graphs the stress strain curves of the distributed Ci. The first thing to be noted is that there seems to be very little discrepancy between the stress strain curves as a function of distribution. Even from other trials of alternating the size of the window, the points seems to line up very close onto each other. In terms of answer the prompt, a stress strain curve cannot be achieved close the one found in problem 2. All of the curves in problem 2 converges on a linear path, but in *Figure 15,* alternating the Ci distribution does not change the fact that with increases strained the curve will still follow a nonlinear path.

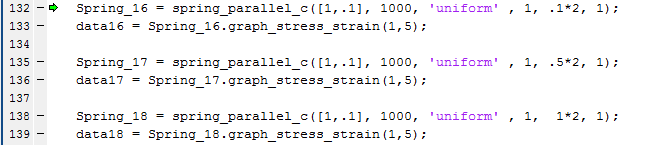


Figure 14: Nonlinear Spring System with Varying Ci

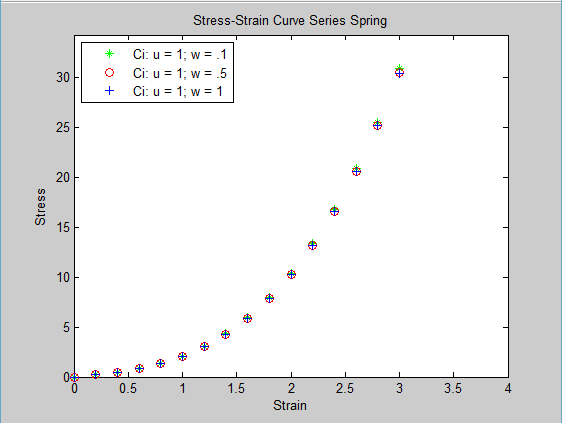


Figure 15: Nonlinear Spring System with Varying Ci

**5)**

Through iterative curve fitting, a single nonlinear spring was able to get close to the curve of a linear spring system in parallel with a uniform distribution of length L. This can be seen in *Figure 16 and 17.* Nonlinear model number 9, became the closest to modeling the curve.

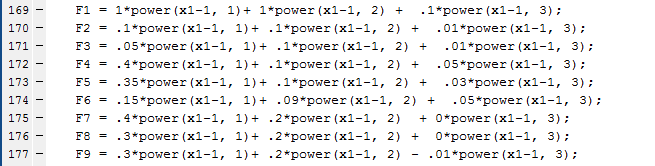


Figure 16: Manual Curve Fitting Implementation

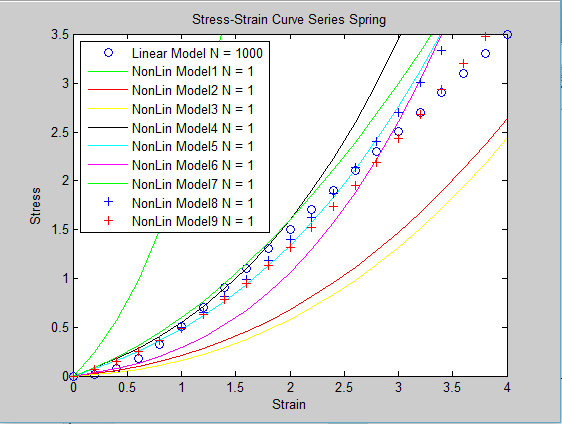


Figure 17: Manual Curve Fitting Stress Strain Curve

**6)**

As seen from the results from *Figure 20-23*, the accuracy of the graph seems to be influenced by the distribution of the lengths and the average value. The graphs were all of a relatively good fit besides *Figure 22*, which dipped down to a negative force in the initial increments of strain. The conjecture could be made that the fit is only good when the distribution of lengths start at the initial length of the fiber bundle. If this is not the case, the the first part of the stress strain curve will be flat. This makes it difficult for a fit to be sized since the force equation used in the matlab script does not accommadate for a flat region of the curve.

You cannot use the Ai of the single compartment model to characterize the stiffness of the parallel spring system. As it was seen in *Figure 19,* in some instances the A value became negative and theoretically stiffness can not have a negative value. Because of this evidence stiffness can be characterized by the A constance in this parallel spring system.

*Figure 18* shows the function used for the lsqcurvefit.

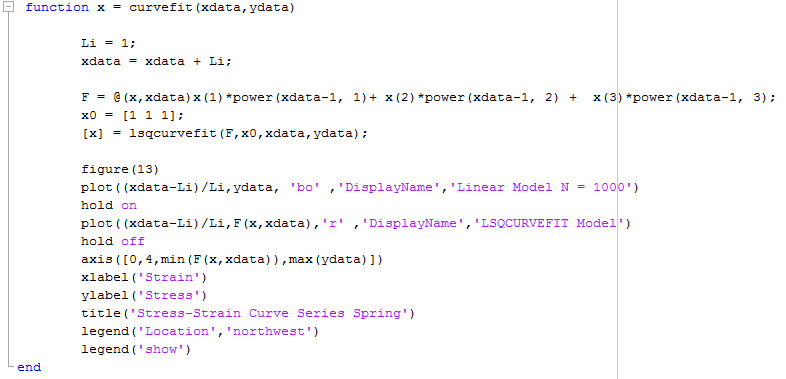


Figure 18: Matlab Script for Curve Fit Function

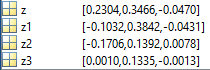


Figure 19: [a, b, c] Curve Fit Results from [1.5, 0.5] = Z; [2, 1] = Z1; [3, 1] = Z2; [3, 2] = Z3;

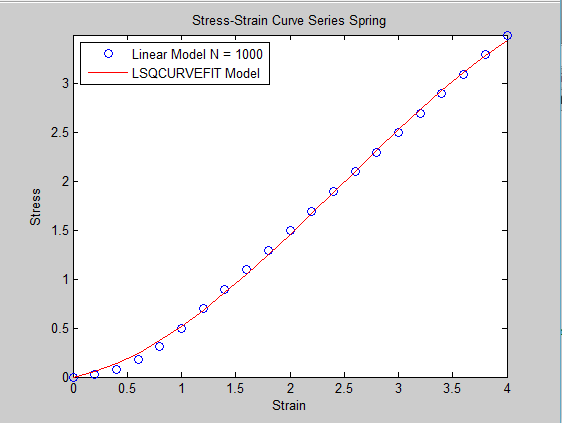


Figure 20: LSQFIT [1.5, 0.5]

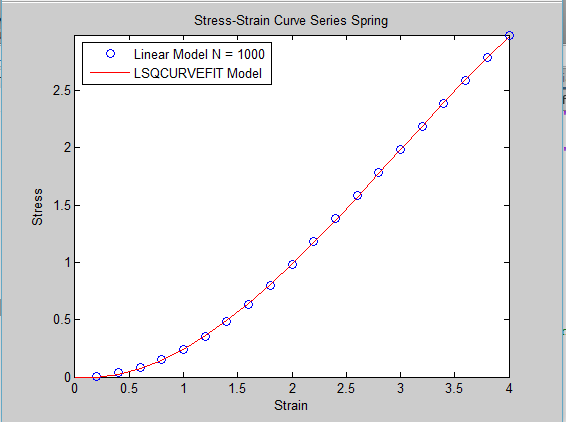


Figure 21: LSQFIT [2, 1]

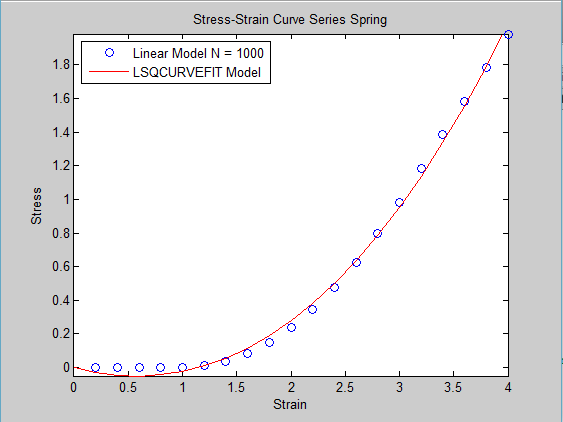


Figure 22: LSQFIT [3, 1]

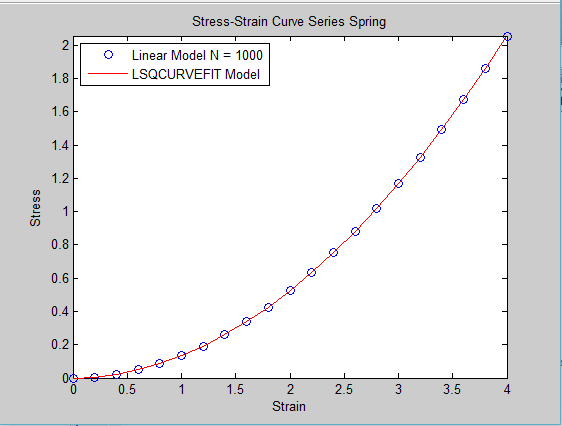


Figure 23: LSQFIT [3, 2]

**7)**

Recruitement generates nonlinearity by the distribution of different fibers. From the matlab modeling it can be seen that as with increasing strain, more fibers become engaged and contribute force. This explains the knee, since in the initial increments of displacement only a small fraction of the fibers contribute the force leaving the slope small.

The spring constant of a single fibril can be best represented by the later part of the stress strain curve past the distribution of initial lengths of the spring. The reason for this is in the initial increments of the displacement, the force exerted by the spring parallel system is a function of both the parameters of the length distribution and the constants of the individual spring. Once the fibers becomes extended past the distribution of lengths, the only variables becomes the constants of the individual spring.

From *Problem 6*, it can be seen that linear wavy fibers can only be respresented by a single compartment model when the distribution of the intial lengths of the fibers starts at the initial displacement of the spring system. If there is a flat 0 Force region in the stress strain curve, the solver will not be able to fit the single compartment model.

In a hierchical multiscale structure, we neeed to first consider how a fibril acts on its own, and not in a bundle of other fibrils. From literature on indivdual fibril stress strain curves, the fibril acts in a nonliner fashion when hydrated with water, which would be the expected the case. The stress strain curve is seen in *Figure 24* demonstrates that relationship. With this data the best way to model the hierchial structure would be to have a nonlinear set of springs acting in a parallel bundle. However, the draw back from this model is that this only can model small strain and it doesn’t incorporate fracturing of the collagen fibrils.

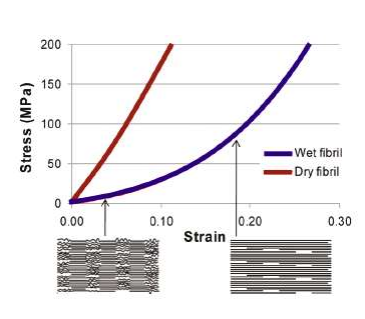


Figure 24: Mechanism of molecular stretching within the fibril (lecture notes)

**Bonus)**

The interpration of the assignment was to introduce increased force where there was a theoretical cross link between two fibers. The way this was done in matlab can be seen by *Figure 25 and Figure 26.* When .graph\_cl was called, instead of looping through the array of varying length and calculating forces, it first initializes a uniform random distribution between [0:1], which became a probability of crosslink. An additional parameter is introduced “density” which symbolized the percentage of cross linking between fibers. In this model, the hypothetical scenario is that all of the fibers are linearly distrubuted and cross link random distribution symbolizes the chance of a connection between two adjacent fibers. This model is unrealistic in real world applications, but simplifies the matlab code while still demonstrating the effects of cross linking. When the force is calculated, a set of logical if statements are used to evaluate to see if there is crosslinking with the fiber before and after it. In the case the statement returns positive a 50% addition of force is added on to the fiber symbolizing an apparent stiffness. From *Figure 27*, the stress strain curve demosntrates that increasing crosslinking increases the stiffness of the fiber bundle. *Figure 28* shows the input variables that were used for used for the spring properties.

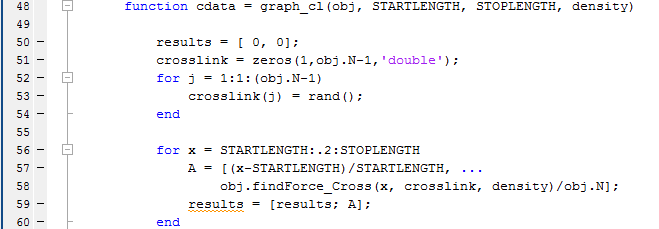


Figure 25: Modified graphing function for Cross Linking

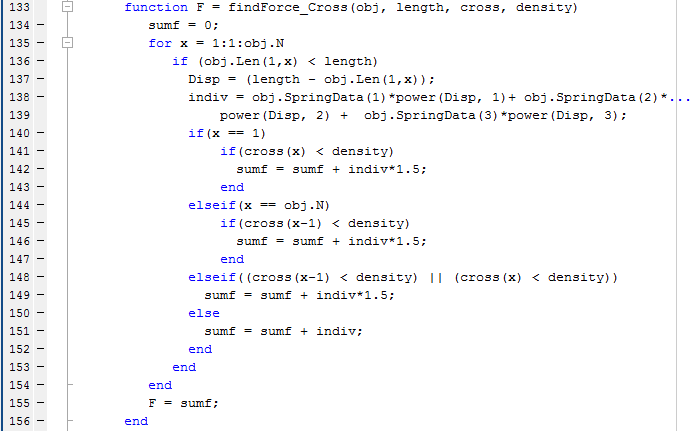


Figure 26: Modified Force Calculation Function for Crosslinking

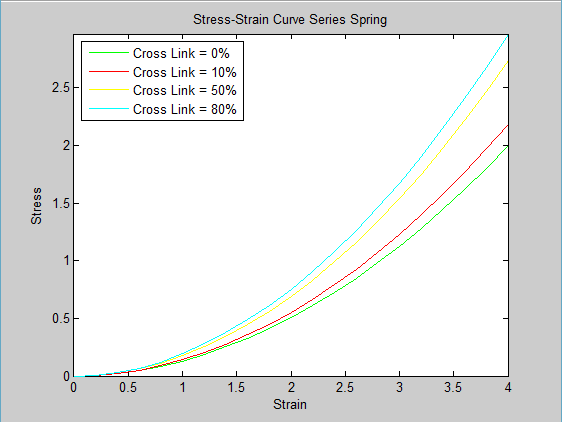


Figure 27: Crosslinking Stress Strain Curve Results

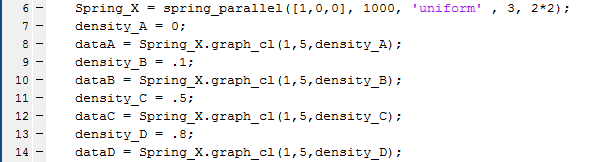


Figure 28: Input Variables for Crosslinking Analysis

**Appendix:**

***Class parallel\_spring***

%================Lab 2,Simulation of the stress-strain====================%

%=================properties of wavy parallel fibers======================%

%QUESTION 1

%1) Write a Matlab code that simulates the stress-strain curve of the

%fiber. Assume first that the system includes N=1000 linear identical

%springs each with ai=1, bi=0 and ci=0. Also, assume that L0,i=1 for each

%spring. Calculate the stress-strain curve of the model while stretching

%the system from length L=1 to L=5. Plot the result! What is the effective

%stiffness of the fiber as a function of the spring constant of the

%fibrils?

Spring\_1 = spring\_parallel([1,0,0], 1000, 'single' , 1,0\*2);

force1 = Spring\_1.findForce(2);

data1 = Spring\_1.graph\_stress\_strain(1,5);

%QUESTION 2

%Transform this distribution such that ?=1.5 and w=0.1. Plot and check

% that all values are between 1.4 and 1.6 and assign them to L0,i. Calculate

% the corresponding stress-strain curve. Repeat the stress-strain curve

% calculations for w=0.2, 0.3 and 0.5. Next, increase ? to 3 and repeat

% the calculations for w=0.1, 1, and 2. Plot the stress-strain curves on

% the same graph. How do the mean and the width of the initial length

%distribution affect the shape of the stress-strain curve?

Spring\_2 = spring\_parallel([1,0,0], 1000, 'uniform' , 1.5,.1\*2);

data2 = Spring\_2.graph\_stress\_strain(1,5);

t = 1:1:1000;

z = Spring\_2.Len();

plot(t,z,'ro')

Spring\_3 = spring\_parallel([1,0,0], 1000, 'uniform' , 1.5,.2\*2);

data3 = Spring\_3.graph\_stress\_strain(1,5);

Spring\_4 = spring\_parallel([1,0,0], 1000, 'uniform' , 1.5,.3\*2);

data4 = Spring\_4.graph\_stress\_strain(1,5);

Spring\_5 = spring\_parallel([1,0,0], 1000, 'uniform' , 1.5,.5\*2);

data5 = Spring\_5.graph\_stress\_strain(1,5);

Spring\_6 = spring\_parallel([1,0,0], 1000, 'uniform' , 3,.1\*2);

data6 = Spring\_6.graph\_stress\_strain(1,5);

Spring\_7 = spring\_parallel([1,0,0], 1000, 'uniform' , 3, 1\*2);

data7 = Spring\_7.graph\_stress\_strain(1,5);

Spring\_8 = spring\_parallel([1,0,0], 1000, 'uniform' , 3, 2\*2);

data8 = Spring\_8.graph\_stress\_strain(1,5);

figure(2)

plot(data2(:,1),data2(:,2), 'g' ,'DisplayName','u = 1.5; w = .1')

hold on

plot(data3(:,1),data3(:,2), 'r' ,'DisplayName','u = 1.5; w = .2')

hold on

plot(data4(:,1),data4(:,2), 'y' ,'DisplayName','u = 1.5; w = .3')

hold on

plot(data5(:,1),data5(:,2), 'c' ,'DisplayName','u = 1.5; w = .5')

hold on

plot(data6(:,1),data6(:,2), 'b' ,'DisplayName','u = 3; w = .1')

hold on

plot(data7(:,1),data7(:,2), 'k' ,'DisplayName','u = 3; w = 1')

hold on

plot(data8(:,1),data8(:,2), 'm' ,'DisplayName','u = 3; w = 2')

axis([0,4,0,max(data5(:,2))])

xlabel('Strain')

ylabel('Stress')

title('Stress-Strain Curve Series Spring')

legend('Location','northwest')

legend('show')

%QUESTION 3

%Repeat 1) and 2) with nonlinear individual springs using bi=0.1 and ci=1.

%How does microscopic nonlinearity influence macroscopic mechanics in the

%presence of heterogeneous waviness?

Spring\_8b = spring\_parallel([1,.1,1], 1000, 'single' , 1,0\*2);

data8b = Spring\_8b.graph\_stress\_strain(1,5);

Spring\_9 = spring\_parallel([1,.1,1], 1000, 'uniform' , 1.5,.1\*2);

data9 = Spring\_9.graph\_stress\_strain(1,5);

Spring\_10 = spring\_parallel([1,.1,1], 1000, 'uniform' , 1.5,.2\*2);

data10 = Spring\_10.graph\_stress\_strain(1,5);

Spring\_11 = spring\_parallel([1,.1,1], 1000, 'uniform' , 1.5,.3\*2);

data11 = Spring\_11.graph\_stress\_strain(1,5);

Spring\_12 = spring\_parallel([1,.1,1], 1000, 'uniform' , 1.5,.5\*2);

data12 = Spring\_12.graph\_stress\_strain(1,5);

Spring\_13 = spring\_parallel([1,.1,1], 1000, 'uniform' , 3,.1\*2);

data13 = Spring\_13.graph\_stress\_strain(1,5);

Spring\_14 = spring\_parallel([1,.1,1], 1000, 'uniform' , 3, 1\*2);

data14 = Spring\_14.graph\_stress\_strain(1,5);

Spring\_15 = spring\_parallel([1,.1,1], 1000, 'uniform' , 3, 2\*2);

data15 = Spring\_15.graph\_stress\_strain(1,5);

figure(3)

plot(data9(:,1),data9(:,2), 'g' ,'DisplayName','u = 1.5; w = .1')

hold on

plot(data10(:,1),data10(:,2), 'r' ,'DisplayName','u = 1.5; w = .2')

hold on

plot(data11(:,1),data11(:,2), 'y' ,'DisplayName','u = 1.5; w = .3')

hold on

plot(data12(:,1),data12(:,2), 'c' ,'DisplayName','u = 1.5; w = .5')

hold on

plot(data13(:,1),data13(:,2), 'b' ,'DisplayName','u = 3; w = .1')

hold on

plot(data14(:,1),data14(:,2), 'k' ,'DisplayName','u = 3; w = 1')

hold on

plot(data15(:,1),data15(:,2), 'm' ,'DisplayName','u = 3; w = 2')

axis([0,4,0,max(data15(:,2))\*1.5])

xlabel('Strain')

ylabel('Stress')

title('Stress-Strain Curve Series Spring')

legend('Location','northwest')

legend('show')

%QUESTION 4

%Next, set all L0,i=1 and distribute ci uniformly around its mean of 1

%with w=0.1, 0.5 and 1. Compute the stress-strain curves and compare them

%to those in 2). Can you get a stress-strain curve similar to one of those

%in 2) where L0 was distributed?

Spring\_16 = spring\_parallel\_c([1,.1], 1000, 'uniform' , 1, .1\*2, 1);

data16 = Spring\_16.graph\_stress\_strain(1,5);

Spring\_17 = spring\_parallel\_c([1,.1], 1000, 'uniform' , 1, .5\*2, 1);

data17 = Spring\_17.graph\_stress\_strain(1,5);

Spring\_18 = spring\_parallel\_c([1,.1], 1000, 'uniform' , 1, 1\*2, 1);

data18 = Spring\_18.graph\_stress\_strain(1,5);

figure(11)

plot(data16(:,1),data16(:,2), 'g\*' ,'DisplayName','Ci: u = 1; w = .1')

hold on

plot(data17(:,1),data17(:,2), 'ro' ,'DisplayName','Ci: u = 1; w = .5')

hold on

plot(data18(:,1),data18(:,2), 'b+' ,'DisplayName','Ci: u = 1; w = 1')

hold on

axis([0,4,0,max(data18(:,2)/2)])

xlabel('Strain')

ylabel('Stress')

title('Stress-Strain Curve Series Spring')

legend('Location','northwest')

legend('show')

%QUESTION 5

%Set therefore N=1 and allow for nonlinearity. Then go back to your

%linear model in 2) and compute the stress-strain curve with ?=1.5 and

%w=0.5. Imagine that this stress-strain curve comes from the measurement

%of the stress-strain curve of a fiber and you now want to fit your single

%compartment model to these data. Try to tune the parameters a, b and c

%of the single compartment model by hand to best match the stress-strain

%curve from the parallel model. Try no more than 10-15 different combinations.

linear\_model = data5;

x1 = 1:0.2:5;

F1 = 1\*power(x1-1, 1)+ 1\*power(x1-1, 2) + .1\*power(x1-1, 3);

F2 = .1\*power(x1-1, 1)+ .1\*power(x1-1, 2) + .01\*power(x1-1, 3);

F3 = .05\*power(x1-1, 1)+ .1\*power(x1-1, 2) + .01\*power(x1-1, 3);

F4 = .4\*power(x1-1, 1)+ .1\*power(x1-1, 2) + .05\*power(x1-1, 3);

F5 = .35\*power(x1-1, 1)+ .1\*power(x1-1, 2) + .03\*power(x1-1, 3);

F6 = .15\*power(x1-1, 1)+ .09\*power(x1-1, 2) + .05\*power(x1-1, 3);

F7 = .4\*power(x1-1, 1)+ .2\*power(x1-1, 2) + 0\*power(x1-1, 3);

F8 = .3\*power(x1-1, 1)+ .2\*power(x1-1, 2) + 0\*power(x1-1, 3);

F9 = .3\*power(x1-1, 1)+ .2\*power(x1-1, 2) - .01\*power(x1-1, 3);

figure(4)

plot(data5(:,1),data5(:,2), 'bo' ,'DisplayName','Linear Model N = 1000')

hold on

plot((x1-1)/1,F1, 'g' ,'DisplayName','NonLin Model1 N = 1')

hold on

plot((x1-1)/1,F2, 'r' ,'DisplayName','NonLin Model2 N = 1')

hold on

plot((x1-1)/1,F3, 'y' ,'DisplayName','NonLin Model3 N = 1')

hold on

plot((x1-1)/1,F4, 'k' ,'DisplayName','NonLin Model4 N = 1')

hold on

plot((x1-1)/1,F5, 'c' ,'DisplayName','NonLin Model5 N = 1')

hold on

plot((x1-1)/1,F6, 'm' ,'DisplayName','NonLin Model6 N = 1')

hold on

plot((x1-1)/1,F7, 'g' ,'DisplayName','NonLin Model7 N = 1')

hold on

plot((x1-1)/1,F8, 'b+' ,'DisplayName','NonLin Model8 N = 1')

hold on

plot((x1-1)/1,F9, 'r+' ,'DisplayName','NonLin Model9 N = 1')

hold on

axis([0,4,0,max(data5(:,2))])

xlabel('Strain')

ylabel('Stress')

title('Stress-Strain Curve Series Spring')

legend('Location','northwest')

legend('show')

%QUESTION 6

%LSQFIT

xdata = data5(:,1);

ydata = data5(:,2);

Spring\_30 = spring\_parallel([1,0,0], 1000, 'uniform' , 2,1\*2);

data30 = Spring\_30.graph\_stress\_strain(1,5);

Spring\_31 = spring\_parallel([1,0,0], 1000, 'uniform' , 3,1\*2);

data31 = Spring\_31.graph\_stress\_strain(1,5);

Spring\_32 = spring\_parallel([1,0,0], 1000, 'uniform' , 3,2\*2);

data32 = Spring\_32.graph\_stress\_strain(1,5);

z = curvefit(xdata,ydata);

z1 = curvefit(xdata,data30(:,2));

z2 = curvefit(xdata,data31(:,2));

z3 = curvefit(xdata,data32(:,2));

%BONUS QUESTION

%CrossLinking

%Show that increasing the cross-linking density significantly influences

%the stress-strain curve of the fiber.

Spring\_X = spring\_parallel([1,0,0], 1000, 'uniform' , 3, 2\*2);

density\_A = 0;

dataA = Spring\_X.graph\_cl(1,5,density\_A);

density\_B = .1;

dataB = Spring\_X.graph\_cl(1,5,density\_B);

density\_C = .5;

dataC = Spring\_X.graph\_cl(1,5,density\_C);

density\_D = .8;

dataD = Spring\_X.graph\_cl(1,5,density\_D);

figure(10)

plot(dataA(:,1),dataA(:,2), 'g' ,'DisplayName','Cross Link = 0%')

hold on

plot(dataB(:,1),dataB(:,2), 'r' ,'DisplayName','Cross Link = 10%')

hold on

plot(dataC(:,1),dataC(:,2), 'y' ,'DisplayName','Cross Link = 50%')

hold on

plot(dataD(:,1),dataD(:,2), 'c' ,'DisplayName','Cross Link = 80%')

axis([0,4,0,max(dataD(:,2))])

xlabel('Strain')

ylabel('Stress')

title('Stress-Strain Curve Series Spring')

legend('Location','northwest')

legend('show')

**Class of Parallel Spring Bundle:**

%Computaional Lab 2

%Matthew Mirek

classdef spring\_parallel

properties

SpringData

Xo = 0;

N

Distr\_Type

Distr\_Width

Distr\_Average

Len

end

methods

%Constructor class, initiliazes variables

function obj = spring\_parallel(A, size, type, avg, width)

if nargin > 0

obj.SpringData = A;

obj.N = size;

obj.Distr\_Type = type;

obj.Distr\_Width = width;

obj.Distr\_Average = avg;

obj.Len = zeros(1,size,'double');

if strcmpi('single',obj.Distr\_Type)

for x = 1:1:obj.N

obj.Len(1,x) = obj.Distr\_Average;

end

end

if strcmpi('uniform',obj.Distr\_Type)

for x = 1:1:obj.N

obj.Len(1,x) = obj.Distr\_Average - obj.Distr\_Width/2 + rand\*obj.Distr\_Width;

end

end

end

end

function cdata = graph\_cl(obj, STARTLENGTH, STOPLENGTH, density)

results = [ 0, 0];

crosslink = zeros(1,obj.N-1,'double');

for j = 1:1:(obj.N-1)

crosslink(j) = rand();

end

for x = STARTLENGTH:.2:STOPLENGTH

A = [(x-STARTLENGTH)/STARTLENGTH, ...

obj.findForce\_Cross(x, crosslink, density)/obj.N];

results = [results; A];

end

results(1,:) = [];

figure(1)

plot(results(:,1),results(:,2), 'g' ,'DisplayName','Calculated Spring')

axis([0,(STOPLENGTH-STARTLENGTH)/STARTLENGTH,0,max(results(:,2))])

xlabel('Strain')

ylabel('Stress')

title('Stress-Strain Curve Series Spring')

legend('Location','northwest')

legend('show')

cdata = results;

end

function E = findEnergy(obj, length)

sum = 0;

for x = 1:1:obj.N

if (obj.Len(1,x) < length)

Disp = (obj.Len(1,x) - length);

sum = sum + 1/2\*obj.SpringData(1)\*power(Disp, 2)+ 1/3\*obj.SpringData(2)\*power(Disp, 3) + 1/4\*obj.SpringData(3)\*power(Disp, 4);

end

end

E = sum;

end

function F = findForce(obj, length)

sumf = 0;

for x = 1:1:obj.N

if (obj.Len(1,x) < length)

Disp = (length - obj.Len(1,x));

indiv = obj.SpringData(1)\*power(Disp, 1)+ obj.SpringData(2)\*power(Disp, 2) + obj.SpringData(3)\*power(Disp, 3);

sumf = sumf + indiv;

end

end

F = sumf;

end

function data = graph\_stress\_strain(obj, STARTLENGTH, STOPLENGTH)

results = [ 0, 0];

for x = STARTLENGTH:.2:STOPLENGTH

A = [(x-STARTLENGTH)/STARTLENGTH, obj.findForce(x)/obj.N];

results = [results; A];

end

results(1,:) = [];

figure(1)

plot(results(:,1),results(:,2), 'g' ,'DisplayName','Calculated Spring')

axis([0,(STOPLENGTH-STARTLENGTH)/STARTLENGTH,0,max(results(:,2))])

xlabel('Strain')

ylabel('Stress')

title('Stress-Strain Curve Series Spring')

legend('Location','northwest')

legend('show')

data = results;

end

function F = findForce\_Cross(obj, length, cross, density)

sumf = 0;

for x = 1:1:obj.N

if (obj.Len(1,x) < length)

Disp = (length - obj.Len(1,x));

indiv = obj.SpringData(1)\*power(Disp, 1)+ obj.SpringData(2)\*...

power(Disp, 2) + obj.SpringData(3)\*power(Disp, 3);

if(x == 1)

if(cross(x) < density)

sumf = sumf + indiv\*1.5;

end

elseif(x == obj.N)

if(cross(x-1) < density)

sumf = sumf + indiv\*1.5;

end

elseif((cross(x-1) < density) || (cross(x) < density))

sumf = sumf + indiv\*1.5;

else

sumf = sumf + indiv;

end

end

end

F = sumf;

end

function L = get.Len(obj)

L = obj.Len;

end

end

end

**Class of Parallel Spring Bundle (modified for distributed Ci case)**

%Computaional Lab 2

%Matthew Mirek

%Modified Class to change distribution of ci

classdef spring\_parallel\_c

properties

SpringData

Xo = 0;

N

Distr\_Type

Distr\_Width

Distr\_Average

Initial\_Length

C

end

methods

%Constructor class, initiliazes variables

function obj = spring\_parallel\_c(A, size, type, avg, width, initial)

if nargin > 0

obj.SpringData = A;

obj.N = size;

obj.Distr\_Type = type;

obj.Distr\_Width = width;

obj.Distr\_Average = avg;

obj.Initial\_Length = initial;

obj.C = zeros(1,size,'double');

if strcmpi('uniform',obj.Distr\_Type)

for x = 1:1:obj.N

obj.C(1,x) = obj.Distr\_Average - obj.Distr\_Width/2 + rand\*obj.Distr\_Width;

end

end

end

end

function F = findForce(obj, length)

sumf = 0;

for x = 1:1:obj.N

Disp = (length - obj.Initial\_Length);

indiv = obj.SpringData(1)\*power(Disp, 1)+ obj.SpringData(2)\*power(Disp, 2) + obj.C(1,x)\*power(Disp, 3);

sumf = sumf + indiv;

end

F = sumf;

end

function data = graph\_stress\_strain(obj, STARTLENGTH, STOPLENGTH)

results = [ 0, 0];

for x = STARTLENGTH:.2:STOPLENGTH

A = [(x-STARTLENGTH)/STARTLENGTH, obj.findForce(x)/obj.N];

results = [results; A];

end

results(1,:) = [];

figure(1)

plot(results(:,1),results(:,2), 'g' ,'DisplayName','Calculated Spring')

axis([0,(STOPLENGTH-STARTLENGTH)/STARTLENGTH,0,max(results(:,2))])

xlabel('Strain')

ylabel('Stress')

title('Stress-Strain Curve Series Spring')

legend('Location','northwest')

legend('show')

data = results;

end

end

end