Tension-Actuated, Multi-Jointed Compliant Finger Analysis

Analysis by Joshua Gafford

1 Objective

The motivation is to create a static analytical model of a cable-actuated, multi-jointed finger, with alternating structural and flexural sections, to obtain quantitative values for output force and deflection vs. input tensioning force. The model will be coded in MATLAB such that parameters and material properties can be altered to observe the theoretical performance of the finger pre-manufacturing. This model will assist in the design of underactuated finger mechanisms via 3D-printing or shape deposition manufacturing (SDM). The model presented herein is linear, and as such, the scope is limited and quantitative results should be interpreted with caution for large deformations. Regardless it can be used to compare different finger geometries, materials and configurations in terms of relative performance.

1.1 Model Limitations

The model derived herein assumes a linear-elastic relationship between the stress and strain of the compliant joint material based on an effective elastic modulus. However the materials used in flexible joints are typically non-linear and viscoelastic, and as such, are more accurately modeled by Mooney-Rivlin or Neo-Hookean material models, which introduce more complexity in terms of defining material parameters. A comparison between a linear-elastic approximation and a Mooney-Rivlin model for Urethane is shown in Figure 1, and we can see how the two approximations deviate at larger strains. As such, this model is intended to be a qualitative evaluation tool that can be used to quickly compare different finger geometries.

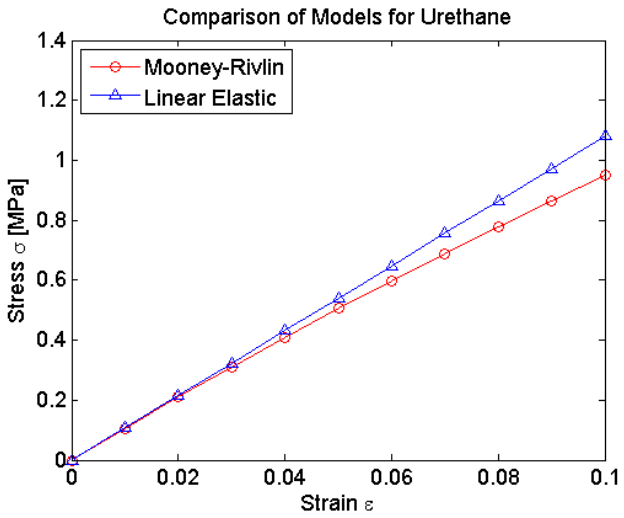

Figure 1. Comparison of linear-elastic and Mooney-Rivlin models for urethane

2 Theory

2.1 Deflection

An illustration of a tendon-driven finger with a single compliant joint is shown in Figure 2. The finger consists of a rigid section (subscript *s*) and a flexible section (subscript *f*). The finger is actuated by a tension force *T* which establishes a moment about the compliant joint.

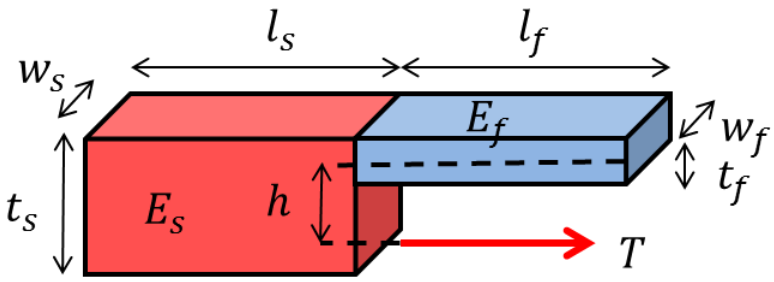

Figure 2. Illustration of simplified cable-driven finger model under cable tension

The simplified model assumes that the deformation behavior is dominated by an applied moment *Th* at the distal end of the finger and contributions from gravity and reaction forces in the x-direction are negligible. We can use superposition to determine the overall deflection at the distal end of the finger by propagating internal moments back to the proximal end. Let δ_i denote the deflection between section *i* and section *i-1*, and let θ_i denote the angle between section *i* and section *i-1*. Given a moment at the distal end of the finger resulting from cable tensioning:

$$\delta_i = \frac{-Thl_i^2}{2E_iI_i} \tag{1}$$

$$\theta_i = \frac{-Thl_i}{E_iI_i} \tag{2}$$

To find the global coordinates of each section (which we'll denote Δ_i for deflection in the y-direction and L_i for displacement in the x-direction), we need to add contributions from previous relative deflections and account for rotations due to the angle of the previous member. We can use homogenous transformation matrices for this task. Assuming we have a point described by $\vec{p}_i$ that we want to rotate about another point $\vec{p}_r$ by an angle α to obtain the point's coordinates in a global frame, we must first translate the point $\vec{p}_r$ back to the origin, perform the rotation, and translate back to $\vec{p}_r$. The sequence looks like the following

$$\vec{p}_f = \boldsymbol{T_1}(\vec{p}_r)\boldsymbol{C}(\alpha)\boldsymbol{T_2}(\vec{p}_r)\cdot\vec{p}_i \tag{3}$$

where $\vec{p}_f$ is the point's coordinates in the global frame, $\boldsymbol{T_1}(\vec{p}_r)$ and $\boldsymbol{T_2}(\vec{p}_r)$ are transformation matrices dependent on the intermediate point $\vec{p}_r$, and $\boldsymbol{C}(\alpha)$ is a rotation matrix dependent on angle α. Rewriting in terms of previously-defined nomenclature:

$$\begin{bmatrix} L_i \\ \Delta_i \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & L_{i-1} \\ 0 & 1 & \Delta_{i-1} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_{i-1}) & -\sin(\theta_{i-1}) & 0 \\ \sin(\theta_{i-1}) & \cos(\theta_{i-1}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -L_{i-1} \\ 0 & 1 & -\Delta_{i-1} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} L_{i-1} + l_i \\ \Delta_{i-1} + \delta_i \\ 1 \end{bmatrix} \tag{4}$$

This model can be extended to an arbitrary number of joints *N*.

2.2 Tip Force

The reaction force at the distal end of the finger is dependent on the cable mounting scheme. We consider two cases as below. A direct-through mounting scheme (left) results in a pure moment about the flexural joints, as the tension force is in-line with the axis of the finger. A top mounted scheme (right) results in a capstan-like effect dependent on contact angle φ.

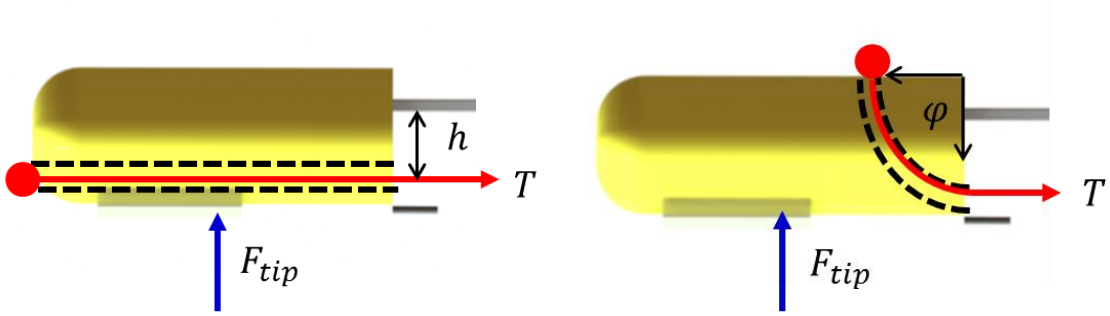

Figure 3. Cable mounting schemes, *(left)* direct-through resulting in pure moment, and *(right)* top-mount that results in a capstan effect

2.2.1 Direct-Through

For the first case, consider a simplified cantilever model shown below in Figure 4. The reaction force at the distal end of the beam is denoted $R_{o,y}$ and is generated by an applied moment M_o. The model is statically indeterminate, meaning the reaction forces can't be solved by equilibrium equations alone.

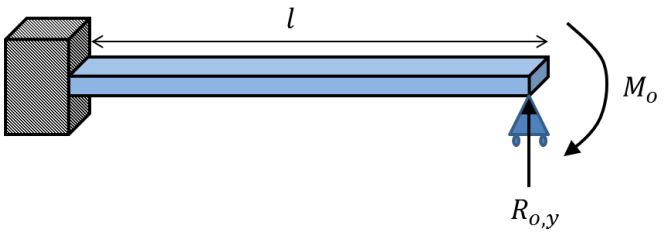

Figure 4. Simplified model of statically-indeterminate cantilever beam

To solve for $R_{o,y}$ we use the superposition method where we (1) remove the reaction force and consider deflection v_o due to the moment, and (2) remove the moment and consider deflection v'_o due to the reaction force, and (3) sum the two deflections and equate to zero to solve for $R_{o,y}$.

$$v_o = \frac{M_o l^2}{2EI} \tag{5}$$

$$v'_o = \frac{R_{o,y} l^3}{3EI} \tag{6}$$

$$v_o - v'_o = \frac{M_o l^2}{2EI} - \frac{R_{o,y} l^3}{3EI} = 0 \tag{7}$$

$$R_{o,y} = \frac{3M_o}{2l} \tag{8}$$

Since the *E* and *I* drop out of the equation, we don't need to consider interactions between the alternating stiff and flexible sections to obtain the overall load at the distal end of the finger. Re-writing in terms of previously-defined terminology:

$$R_{o,y} = F_{tip} = \frac{3Th}{2\sum_{i=1}^{2N} l_i} \tag{9}$$

2.2.2 Top-Mounting

The reaction force is modeled by the capstan equation, which is reproduced below:

$$F_{tip} = Te^{-\mu\varphi} \tag{10}$$

where φ is the contact angle (see Figure 3) and μ is the coefficient of static friction between the cable and the sheath.

2.3 Parasitic Capstan Effects

For higher fidelity, we can consider 'parasitic capstan' effects. This describes the phenomenon that, when the finger is actuated and begins to curve, the frictional force between the cable and the stiff segments increases, which detracts from the transmission ratio of the system.

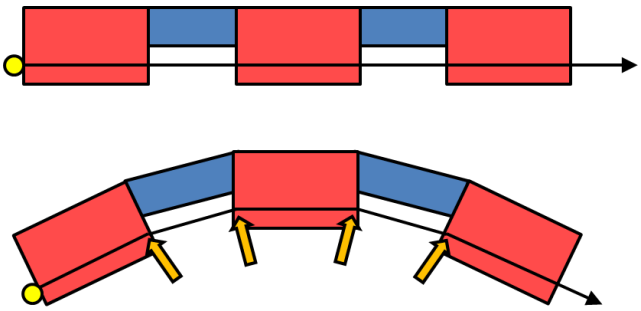

Figure 5. Parasitic capstan effects as the finger assumes a curved profile

The implication of this phenomenon is that the reaction force experiences diminishing returns given greater end deflections due to this parasitic effect. As this effect increases with increasing curvatures, we can model it as a parasitic capstan effect dependent on relative angle between subsequent segments in the kinematic chain. Thus we can refine the distal forces from Equations (9) as the following:

$$F_{tip} = \frac{3Th}{2\sum_{i=1}^{2N} l_i} \prod_{\substack{i=2 \\ i=even}}^{2n-2} e^{-\mu(\theta_{i-1}-\theta_i)} \tag{11}$$

Similarly, for the top-mounted scheme, the distal force becomes:

$$F_{tip} = \frac{T}{e^{\mu\varphi}} \prod_{\substack{i=2 \\ i=even}}^{2n-2} e^{-\mu(\theta_{i-1}-\theta_i)} \tag{12}$$

3 Results

Results of the analysis, for arbitrary materials and dimensions, are given below. Figure 6 is a screencap of the deformed shape for three different loading conditions for *N*=3 joints. Figure 7 shows the results of the analytical model overlaid on top of SDM empirical models with identical characteristics, showing good correlation between results.

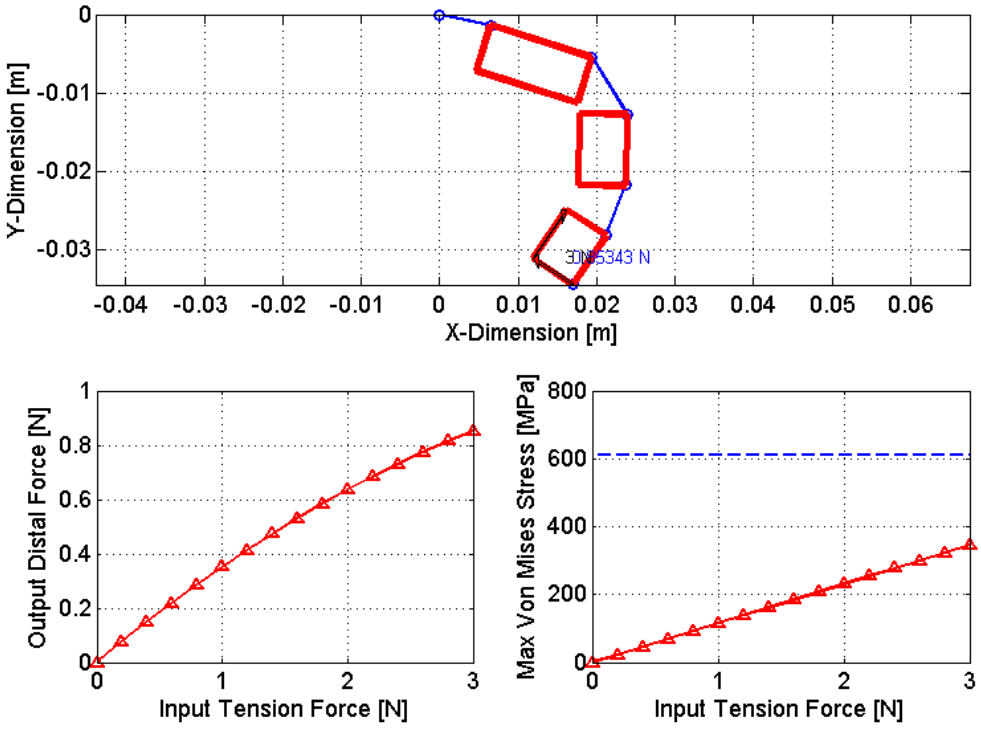

Figure 6. Simulation results for a 3-jointed finger with variable joint stiffness. *(clockwise from top)* deflection visualization of finger, von-mises stress of joint reinforcement flexures, and transmission ratio.

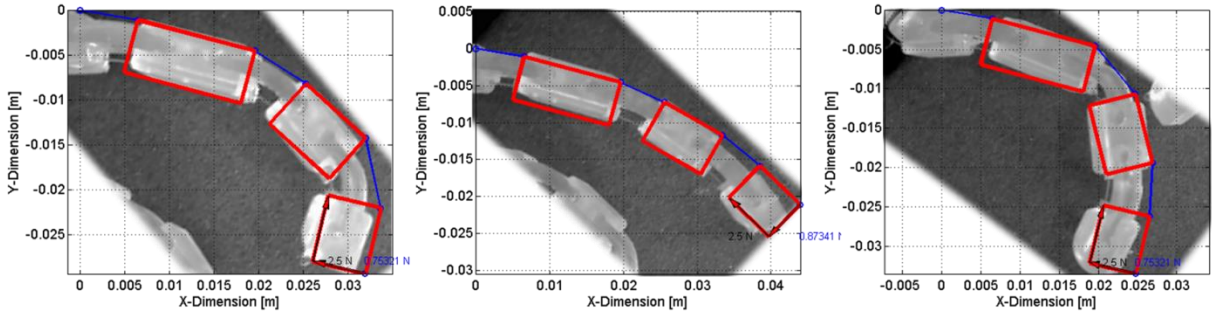

Figure 7. Comparison of analytical model with SDM-fabricated finger prototypes, showing good agreement for a variety of different finger configurations.

4 References

Gere and Timoshenko, Mechanics of Materials

Young and Budynas, Roark's Formulas for Stress and Strain

5 Source Code

Note that this code also uses arrow.m which can be downloaded from the MathWorks website.

5.1 Analysis Initialization Function

```
function null = Analysis_Init_Custom()
% Written by: Joshua Gafford
% Written on: March 3, 2013
%
% Description
%This function initializes the compliant, multi-jointed finger bending
%analysis. Geometric and material parameters are defined herein, and the
%analysis iterates through each joint to compute the overall bend profile
%and distal reaction force. The user may select to run through a ramp or
%sinusoidal loading profile, and save the results to an .avi movie.
%
% Notes
%This model includes a stress calculation for any steel reinforcement
%flexures embedded into the elastomer based on a simple von mises analysis.
%If no reinforcement flexures are being built into the finger, disregard
%this information

close all;clear all;clc;clf;

% User Parameters
Moment=0;   % Cable mounting case, 1=direct through, 0=top mount
angle=0;    % Cable angle for moment=0

% Flexible Joint Parameters
%From proximal to distal
lf=[6.5E-3;6.5E-3;6.5E-3];          % segment length [m]
tf=[.150E-3;.050E-3;.050E-3;];      % steel reinforcement flexure thickness [m]
tp=[2.65E-3;1.65E-3;2.15E-3];       % polymer flexure thickness
wf=3E-3;                            % width of steel flexible section [m]
wp=6E-3;                            % width of polymer flexible section [m]
If=wf.*(tf.^3)/12;  % Second moment of area of steel [m^4]
Ip=wp.*(tp.^3)/12;  % Second moment of area of polymer
Ef=0E9;             % Young's Modulus of steel flexible section [Pa]
Ep=13.8E6;          % Young's Modulus of polymer flexible section [Pa]
mu=0.3;

% Structural Segment Parameters
%From proximal to distal
ls=[13.5E-3;9E-3;7.5E-3];    % length of stiff section [m]
ts=[6.0E-3;6.0E-3;6.0E-3];     % thickness of stiff section [m]
ws=6E-3;            % width of stiff section [m]
Is=ws.*(ts.^3)./12; % Second moment of area [m^4]
Es=2E9;             % Young's Modulus of stiff section [Pa]
h=2E-3;           % Distance between cable and attachment [m]

prevDim=[];

% Select loading profile (ramp or sine)
disp('Please define all custom properties in the .m file');
str=input('Loading Profile? Ramp [r] Sine [s]','s');
switch str
    case 's'
        [amplitude]=input('Input Force Amplitude? [N]');
        [frequency]=input('Resolution? [N]');
        [Ncycles]=input('Number of Cycles?');
         L=(amplitude/2)-(amplitude/2)*cos([0:frequency:2*pi*Ncycles]);
    case 'r'
        [amplitude]=input('Input Force Amplitude? [N]');
        [frequency]=input('Resolution? [N]');
        L=0:frequency:amplitude;
    otherwise
end

% Ask user if movie should be saved
```

```
% movstr=input('Save Movie? [y/n]','s');
movstr='n';
switch movstr
    case 'y'
        filestr=input('Filename?','s');
        moviesave=1;
    case 'n'
        moviesave=0;
    otherwise
end
N=length(lf);   % Number of Joints

% allocating space for local deflection and slope vector
theta=zeros(length(N*2),1);
delta=zeros(length(N*2),1);

deflection=zeros(length(L),1);  % global tip deflection vector

% Initialize Figure
figure(1)
fig = figure('position',[100 100 850 600]);
for ii=1:length(L);
    clf;
    shrinkVec=[];
    if Moment==1;
        for i=N:-1:1
            keffd=2*Ef*If(i)/(lf(i)^2)+2*Ep*Ip(i)/(lf(i)^2);
            delta(2*i)=-L(ii)*h*(ls(i)^2)/(2*Es*Is(i));
            delta(2*i-1)=-L(ii)*h/keffd;

            kefft=Ef*If(i)/(lf(i))+Ep*Ip(i)/(lf(i));
            theta(2*i)=L(ii)*h*ls(i)/(Es*Is(i));
            theta(2*i-1)=L(ii)*h/kefft;
        end
    else
        for i=N:-1:1
            L_adj=L(ii)/(exp(mu*pi/2));
            keffd=3*Ef*If(i)/(lf(i)^3)+3*Ep*Ip(i)/(lf(i)^3);
            delta(2*i)=-L_adj*cos(angle)*(ls(i)^3)/(3*Es*Is(i));
            delta(2*i-1)=-L_adj*cos(angle)/keffd;

            kefft=2*Ef*If(i)/(lf(i)^2)+2*Ep*Ip(i)/(lf(i)^2);
            theta(2*i)=L_adj*cos(angle)*(ls(i)^2)/(2*Es*Is(i));
            theta(2*i-1)=L_adj*cos(angle)/kefft;
        end
    end
    % create vector for global section deflection and x-displacement
    delta_sum=zeros(length(delta),1);
    delta_sum(1)=delta(1);
    lengthvec=zeros(length(delta),1);
    lengthvec(1)=lf(1);

    % add contributions from local deflections, slopes
    for j=2:1:length(delta)
        if ~mod(j,2)
%             compression=L(ii)*ls(ceil(j/2))/(Es*ws*ts(ceil(j/2)));
            compression=0;
            % find coordinates for next point by HTM
            [lengthvec(j) delta_sum(j)]=transform(lengthvec(j-1),...
                  delta_sum(j-1),lengthvec(j-1)+ls(ceil(j/2))-compression,delta_sum(j-1)...
                  +delta(j),-sum(theta(1:j-1)));
        else
%             compression=L(ii)*lf(ceil(j/2))/(Ef*wf*tf(ceil(j/2)));
            compression=0;
            % find coordinates for next point by HTM
            [lengthvec(j) delta_sum(j)]=transform(lengthvec(j-1),...
                  delta_sum(j-1),lengthvec(j-1)+lf(ceil(j/2))-compression,delta_sum(j-1)...
                  +delta(j),-sum(theta(1:j-1)));
        end
    end
```

```
    % insert zero at boundary
    lengthvec=[0;lengthvec];
    delta_sum=[0;delta_sum];

    % Finger deflection visualization
    subplot(2,1,1);
    plot(lengthvec,delta_sum,'-o','LineWidth',2);grid on;hold on;
    % plot rectangles for stiff sections, vectors for load values
    for k=1:length(lengthvec)-1;
        % only plot rectangles for stiff sections
        if ~mod(k,2)
            % only plot vectors on last section
            if k==length(lengthvec)-1
                term=1;
            else
                term=0;
            end
            if Moment==1
                tipload(ii)=3*L(ii)*h./(2*lengthvec(end)*(exp(0.3*(pi/2))+exp(0.3*sum(theta))));
% tip load
                stress(ii)=L(ii)*h*(tf(1)/2)/(If(1));
            else
                tipload(ii)=cos(angle)*L(ii)/(exp(0.3*(pi/2))+exp(0.3*sum(theta)));
                sig11=-3*delta_sum(2)*Ef*(tf(1)/2)/(lf(1)^2);
                sig22=L(ii)*sin(angle)/(tf(1)*wf);
                Q=(tf(1)/4)*(wf)*tf(1)/2;
                tau12=(L(ii)*cos(angle)*Q)/(If(1)*tf(1));
                stress(ii)=sqrt(sig11^2-sig11*sig22+sig22^2+3*(tau12^2));
            end
            [x_bl x_br]=drawRect(0,0,lengthvec(k),delta_sum(k),ts(ceil(k/2)),...
                ls(ceil(k/2)),sum(-theta(1:k)),term,L(ii),tipload(ii));hold on;
            shrinkVec=[shrinkVec;x_bl x_br];
            set(gca,'FontSize',13);
            xlabel('X-Dimension [m]');ylabel('Y-Dimension [m]');
        else
            continue;
        end
    end
    deltaVec(ii)=ChangeInLength(shrinkVec,lengthvec);


    axis equal;
    deflection(ii)=delta_sum(end); %extract global deflection of last section

    % Transmission Ratio Plot
    subplot(2,2,3)
    plot(L(1:ii),tipload(1:ii),'r-^','LineWidth',2);hold on;grid on;
     set(gca,'FontSize',13);
    xlabel('Input Tension Force [N]');ylabel('Output Distal Force [N]');

    % Reinforcement Flexure Stress Plot
    subplot(2,2,4)
    plot(L(1:ii),stress(1:ii)./(10^6),'r-^','LineWidth',2);hold on;grid on;
    line([min(L) max(L)],[610 610],'LineWidth',2,'LineStyle','--');hold on;
    set(gca,'FontSize',13);
    xlabel('Input Tension Force [N]');ylabel('Max Von Mises Stress [MPa]');
    F(ii)=getframe(fig);
end
if moviesave==1
    movie2avi(F, strcat(filestr,'.avi'),'compression','None');
else
end
% Uncomment the following for individual plots
%plot results
% figure(2)
% plot(L,tipload,'-r^','LineWidth',2);grid on;hold on;
% set(gca,'FontSize',13);
% xlabel('Input Tension Force [N]');ylabel('Output Distal Force [N]');
% figure(3)
% plot(L,deflection,'-bo','LineWidth',2);grid on;
% set(gca,'FontSize',13);
```

```
% xlabel('Input Tension Force [N]');ylabel('Output Deflection [m]');
% figure(3)
% plot(-deflection,tipload,'-r^','LineWidth',2);grid on;hold on;
% set(gca,'FontSize',13);
% xlabel('Deflection [m]');ylabel('Distal Force [N]');
% figure(4)
% plot(tipload,stress./(621*10^6),'-r^','LineWidth',2);grid on;hold on;
% set(gca,'FontSize',13);
% xlabel('Tension Force [N]');ylabel('Factor-of-Safety');
end

function delta=ChangeInLength(shrinkVec,lengthVec)
delta=0;
prevDim=[0 0];

for i=2:length(lengthVec)
    if mod(i,2)
        coord=shrinkVec(ceil(1/2),:);
        delta=delta+abs(lengthVec(i)-lengthVec(i-1)-
distance(prevDim(1),coord(1),prevDim(2),coord(2)));
        prevDim=[coord(3) coord(4)];
    else
    end
end
end

function c=distance(x0,xf,y0,yf)
    c=sqrt((xf-x0)^2+(yf-y0)^2);
end


function [x_trans y_trans]=transform(x_o,y_o,x_f,y_f,theta)
%   Description
% This function takes rotates the point [x_f, y_f] about the point
% [x_o,y_o] by angle theta using homogeneous transformations

% Rotation Matrix
C=[cos(theta) -sin(theta) 0;sin(theta) cos(theta) 0;0 0 1];
% Translation Matrix
T1=[1 0 -x_o;0 1 -y_o;0 0 1];
T2=[1 0 x_o;0 1 y_o;0 0 1];
% Translate [x_o,y_o] to origin, rotate by theta, and translate back to
% [x_o,y_o]
XY_trans=T2*C*T1*[x_f; y_f; 1];
x_trans=XY_trans(1);
y_trans=XY_trans(2);
end
```

5.2 drawrect()

```
function null = drawRect(x,y,xoffset,yoffset,h,w,alpha,term,L,tipL)
%Description
% This function draws a rectangle with height h, width w, rotated about
% angle alpha. It also draws tensioning and loading vectors and outputs text
% at the loading location with their respective value

% xv=[x-w/2 x+w/2 x+w/2 x-w/2 x-w/2]; % x-coordinate vector
xv=[x x+w x+w x x];
yv=[y y y-h y-h y];                 % y-coordinate vector
alpha=alpha;                       % reverse angle
% Rotation matrix
C=[cos(alpha) -sin(alpha) 0;sin(alpha) cos(alpha) 0;0 0 1];
% Translation Matrix
T=[1 0 xoffset;0 1 yoffset;0 0 1];

%rotate angle alpha
R(1,:)=xv;R(2,:)=yv;R(3,:)=ones(1,5);
alpha=-alpha;
XY=T*C*R;
```

```
plot(XY(1,:),XY(2,:),'r','LineWidth',4);hold on;
if term==1
    arrow([XY(1,2), XY(2,2)],[XY(1,3), XY(2,3)],'Width',2);hold on;
    text(XY(1,2), XY(2,3),strcat(num2str(tipL),' N'),'Color',[0 0 1],'LineWidth',2);
    arrow([XY(1,3), XY(2,3)],[XY(1,4), XY(2,4)],'Width',1);hold on;
    text(XY(1,4), XY(2,3),strcat(num2str(L),' N'));
end
```

arrow() function is left out for brevity but can be downloaded from MathWorks by following this link: http://www.mathworks.com/matlabcentral/fileexchange/278