A Multiple Stack Triangular Thin Shell Element

Stephanie Golmon
Noah Ledford
Liao Liu
May 3, 2009

Objectives:

1. Combine the triangular plate bending element and the triangular membrane element to create a shell element.

The plate bending element has two rotational degrees of freedom and one translational degree of freedom. The membrane element has two translational degrees of freedom and one rotational degree of freedom. Merging these will create an element with full rotation and translational degrees of freedom, which is useful in modeling heterogeneous wall constructions.

2. Create a triangular stack thin shell element to model composite materials.

N layers of shell element will be stacked, applying Multi-Freedom Constraints (MFCs) on each node, to form a stack element with 18 DOFs in total. This requires a transformation of single element node DOFs to a reference surface. By using a stack element formulation, the total number of degrees of freedom is reduced from 18*N to 18 DOF per element. This element can be used in composite wall fabrications.

3. Improve the static performance of stack shell element by preventing interlayer slip.

The Multi-Freedom constraints are only applied at the nodes, at the corners of the element. This allows for interlayer slip, which weakens the element's performance in static cases. In order to minimize with the goal of eliminating this interlayer slip, parameters are added to higher and lower order terms in the stiffness matrices. Two tests were carried out for finding best parameters of a homogeneous stack shell structure, and results are compared to the analytical solution for a cantilever beam.

Formulations:

Objective 1:

A *Mathematica* code[1]was provided on course web site helping setting up a single layer shell element. This code was converted to matlab code for ease of implemtation for the authors. The matlab m-files are provided in Appendix A. The order of DOF of the element in global coordinate system is designated as:

$$u_{x1}\ u_{y1}, u_{z1}, \theta_{x1}, \theta_{y1}, \theta_{z1}\ u_{x2}, u_{y2}, u_{z2}, \theta_{x2}, \theta_{y2}, \theta_{z2}, u_{x3}, u_{y3}, u_{z3}, \theta_{x3}, \theta_{y3}, \theta_{z3}.$$

Assembling procedure was shown in the picture below. The stiffness matrix of a shell was obtained by putting each DOF of these two component element into corresponding place of the 18 by 18 total stiffness matrix.

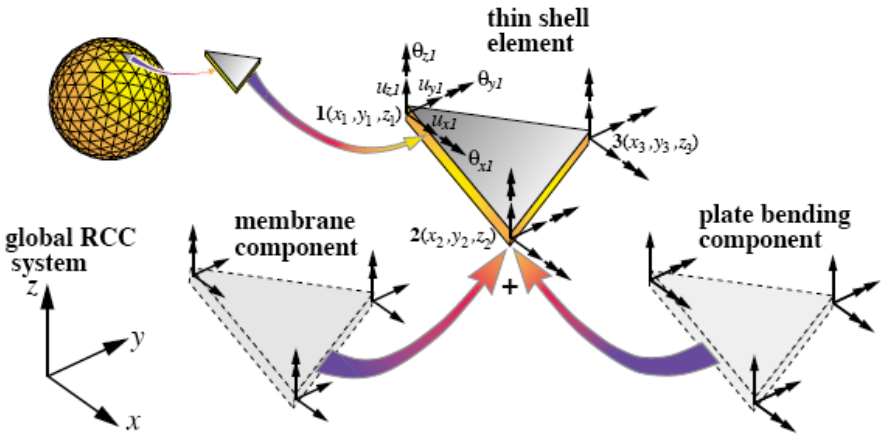


Figure 1.A triangular shell element constructed by combining a membrane (in plane bending) and a Kirchhoff plate bending component.

Objective 2:

In the local coordinate system of an element, choose the mid-surface of the stack as the reference surface, and let the DOFs of the stacked shell element lie on this reference surface. For an element with certain number of layers, the transformation relationship between lamina DOFs and stacked shell DOFs is obtained by imposing MFCs on them, choosing stacked element DOFs as masters, and the lamina DOFs as slaves.

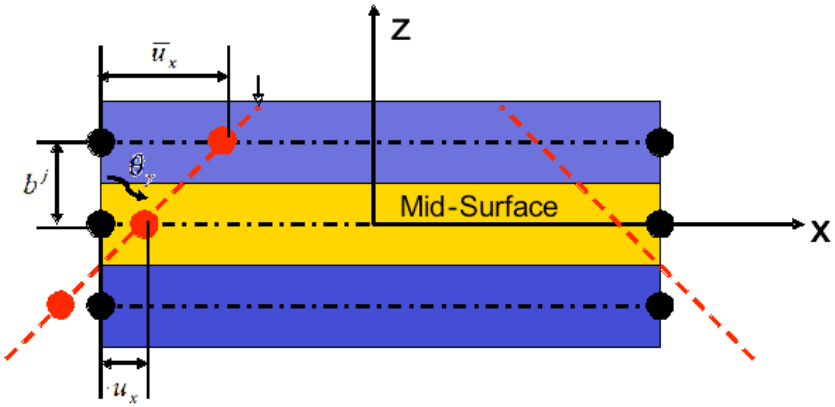


Figure 2: A 3-layer stacked shell element in local X-Z plane. Deformed shape shown to illustrate the relationship between lamina DOFs and the stacked shell DOFs in term of b_i^j .

In Figure 2, b_i^j denotes the distance between node *j* on a mid-surface of layer *i*, and that of the stack element. The transformation relation is:

$$[\bar{u}_{x1},\bar{u}_{y1},\bar{u}_{z1},\bar{\theta}_{x1},\bar{\theta}_{y1},\bar{\theta}_{z1},\cdots\cdots\bar{\theta}_{z3}]^T = T_i\,[u_{x1},u_{y1},u_{z1},\theta_{x1},\theta_{y1},\theta_{z1},\cdots\cdots\theta_{z3}]^T \quad (1)$$

$$(18\times 1) \qquad (18\times 18) \qquad (18\times 1)$$

where the local coordinates for an element are on the left and the coordinates for the stack element are on the right.

The transformation matrix T_i for a lamina, *i*, is given by:

$$T_i = \begin{bmatrix} [T^1] & [0] & [0] \\ [0] & [T^2] & [0] \\ [0] & [0] & [T^3] \end{bmatrix} \qquad where\ T^j = \begin{bmatrix} 1 & 0 & 0 & 0 & b_i^j & 0 \\ 0 & 1 & 0 & -b_i^j & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

where $j=1,2,3$ corresponding to each node of the triangle, and i refers to the lamina

Then the 18 by 18 local stiffness matrix of each layer is obtained by doing a transformation as follows:

$$K_i = T_i^T K_{SH} T_i \tag{3}$$

The total local stiffness matrix of the stacked shell element is the sum of stiffness matrix of all the layers, expressed as:

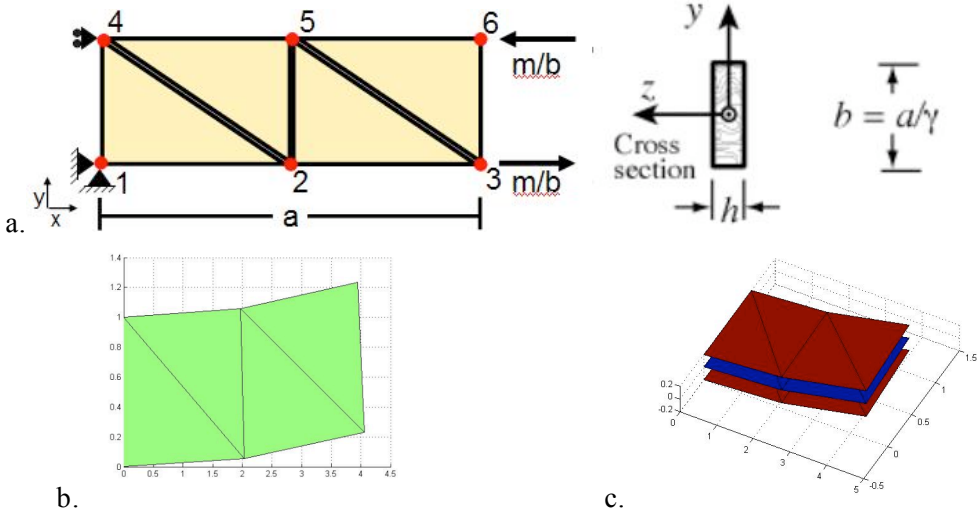
$$K_{ST} = \sum_i^n K_i \tag{4}$$

A local-to-global transformation was carried out after this to have the stiffness matrix in a global system, which is necessary when dealing with structures made of this stacked element. Two tests were carried out: first to check the stacked element formulation and then to help finding out the best form of the stiffness matrix to prevent inter-layer slip of this element.

The first test is an in plane bending test, shown in Figure 3. The structure was made of four triangular stacked shell elements, and under the force applied in the figure, it will behave like a cantilever beam. The analytical solution of this problem can be easily calculated by:

$$u_{y6} = u_{y3} = \frac{Ma^2}{2EI_{zz}} \tag{5}$$

where $M = 1\times10^8\ Nm$, $I_{zz} = hb^3/12$, $h = 0.6\,m$, and $b = 1\,m$. By comparing the tip displacement of the FEM model to the analytical solution, it was verified that the element formulation is correct. The results for testing different numbers of layer and composite structure were shown in Table 1.



a.

b.

c.

Figure 3: (a) Test 1 for stacked shell element, (b) Deformed homogenous structure from FEM model, (c) Deformed structure of a composite.

Table 1: Tip displacements of test 1

	1 layer homogenous	3 layer homogenous	20 layer homogenous	3 layer composite
Model	0.4637	0.4637	0.4637	0.2782
Analytical	0.4571	0.4571	0.4571	N/A

The second test is a plate bending test. The same structure was used, with a different applied load, as shown in Figure 4. The analytical solution of this problem is:

$$u_{z6} = u_{z3} = \frac{Ma^2}{2EI_{yy}} \tag{6}$$

where $I_{yy} = h^3 b / 12$ and $M = 1 \times 10^6 \ Nm$. Also, an auxiliary test was carried out by replacing the loading condition from a moment to an equivalent pair of forces. The results of both loading condition were shown in Table 2.

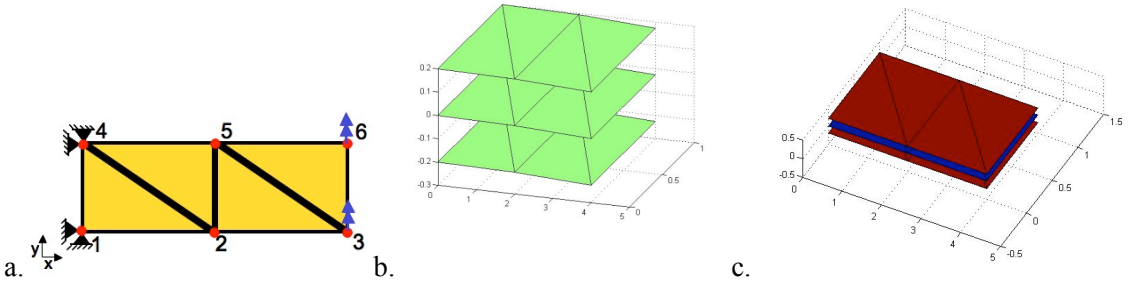


Figure 4: (a) Test 2 for stacked shell element, (b) Deformed homogeneous structure from FEM model, (c) Deformed structure of a composite.

Table 2: Tip displacements for test 2

	1 layer homogenous	3 layer homogenous	20 layer homogenous	3layer composite
Model Moment	-0.0254	-0.0254	-0.0254	-0.0129
Model Force	-0.0168	-0.0236	-0.3546	-0.0127
Analytical	-0.0254	-0.0254	-0.0254	-0.0254

Objective 3: Finding a beta coefficient

The issue of slipping can be resolved by adding in a scaling factor to different aspects of the stiffness matrix; as was shown for the stacked beam example in chapter 20 of the IFEM notes. The formulation of the shell stiffness matrix is constructed of four different parts, a higher and lower term from both of the bending and membrane plate elements. In exploring which terms have a greater effect on the results, of the two different types of bending tests preformed described earlier, four different scaling factors were added called beta. This new construction of the shell stiffness matrix is demonstrated in the Figure 5.

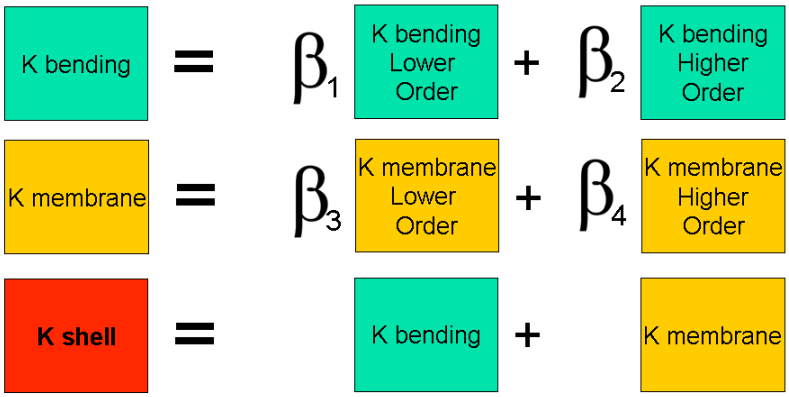


Figure 5: Construction of the shell stiffness with the beta scaling factors

Next, the task of determining which components contributed to the bending tests was undertaken by varying all of the beta factors. Done in a series of five nested for loops starting with number of layers and working through the beta factors the bending tests were solved and their results saved and compared to the analytical solution. The results of this are shown in Figure 6.

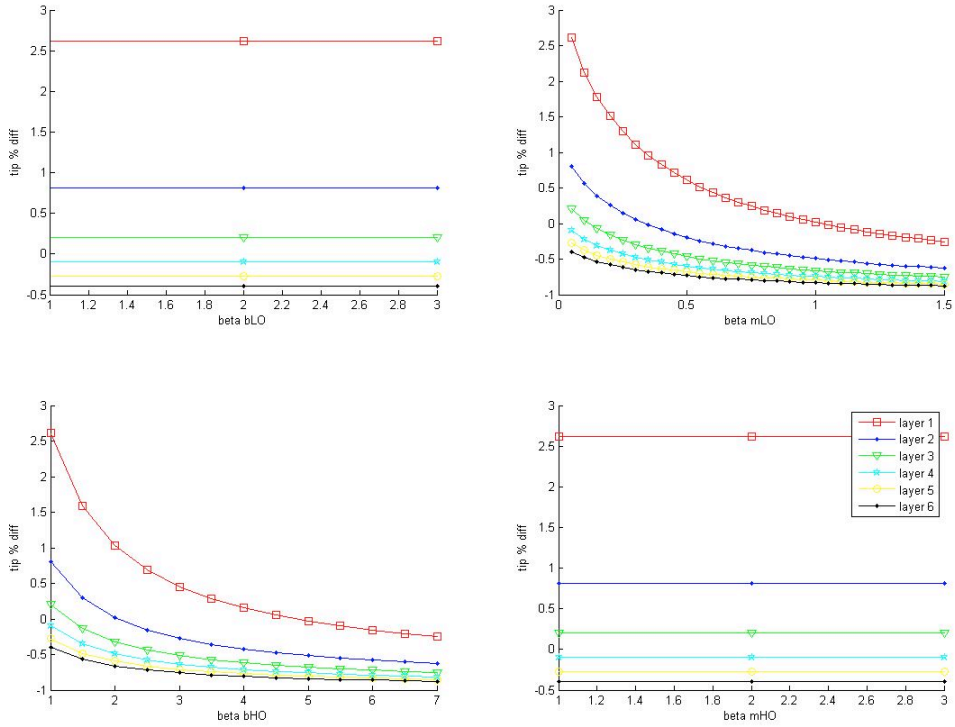


Figure 6: Effect of the beta factors on the in plane bending test.

Figure 6 shows how the different beta factors influence the in plane bending results of the shell element. The four figures display the percent error in the computed solution and the analytical solution. The two on the top are the lower order term for both bending and membrane plate stiffness matrices, designated ‘bLO’ and ‘mLO’ respectively , and the higher order terms below them, ‘bHO’ and ‘mHO’. The most notable feature in the above figure is that the membrane lower order term and the higher order term for bending are the only two that have impact on the

accuracy of the solution. The second major feature is the dependence of the accuracy of the solution on the number of layers. This is seen by the different colored lines and is detailed in the legend. The dependence of the solution on three different parameters makes it challenging to visualize, the next figure shows only the results for one layer however the trade off between the two beta terms that matter can be seen more clearly.

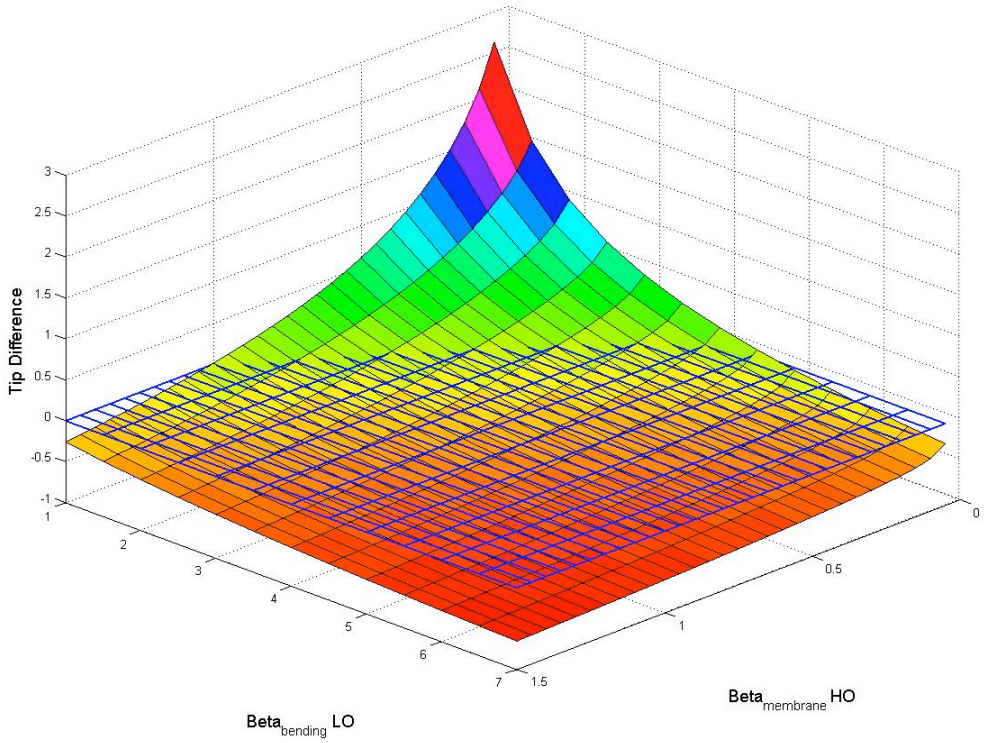


Figure 7: Beta terms effect on the percent error for two layers of in plane bending.

Figure 7 shows how the two beta terms can work together to converge onto the correct solution. The intersection of the two surfaces, the colored one being the percent difference of the solution and the blue lined mesh being the zero error surface, create a curve that trades off one beta value for another to get the correct solution. This line is where one would want to be when using the beta scaling factors. The challenge for this comes when you start adding n number of layers; as is discussed below with the second test.

The second test, the plate bending test, was subjected to the same analysis and the results for the different beta terms are shown in Figure 8.

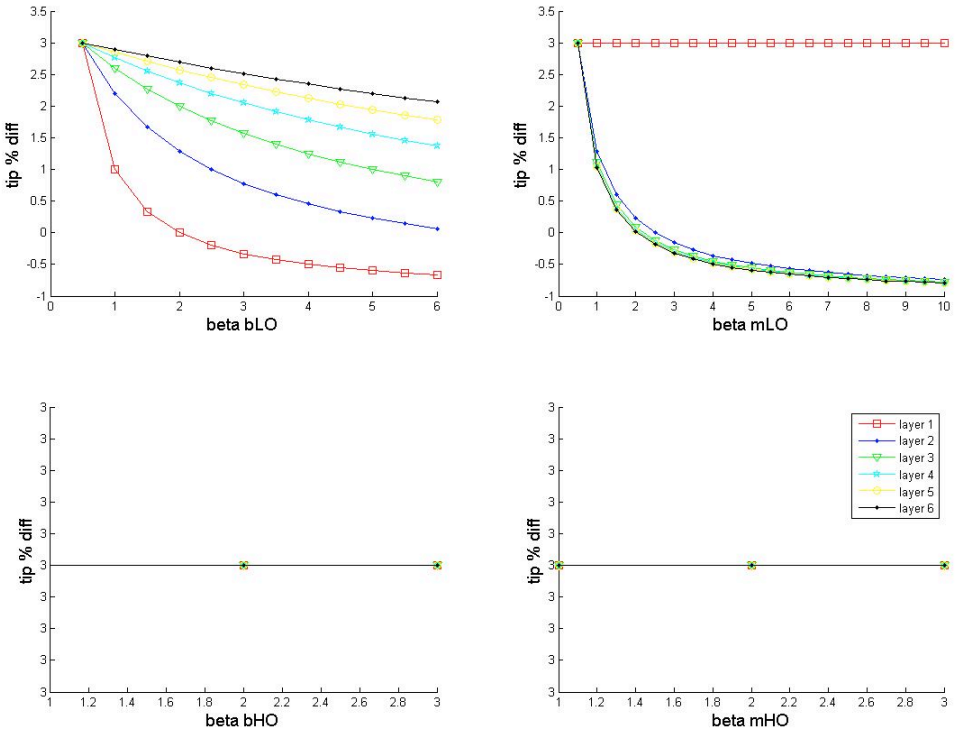


Figure 8: Plate bending test for different beta values.

In Figure 8, the solution is clearly seen to be a function of the bending and membrane lower order terms. Especially interesting is that the membrane term is only a factor when the number of layers is greater than one.

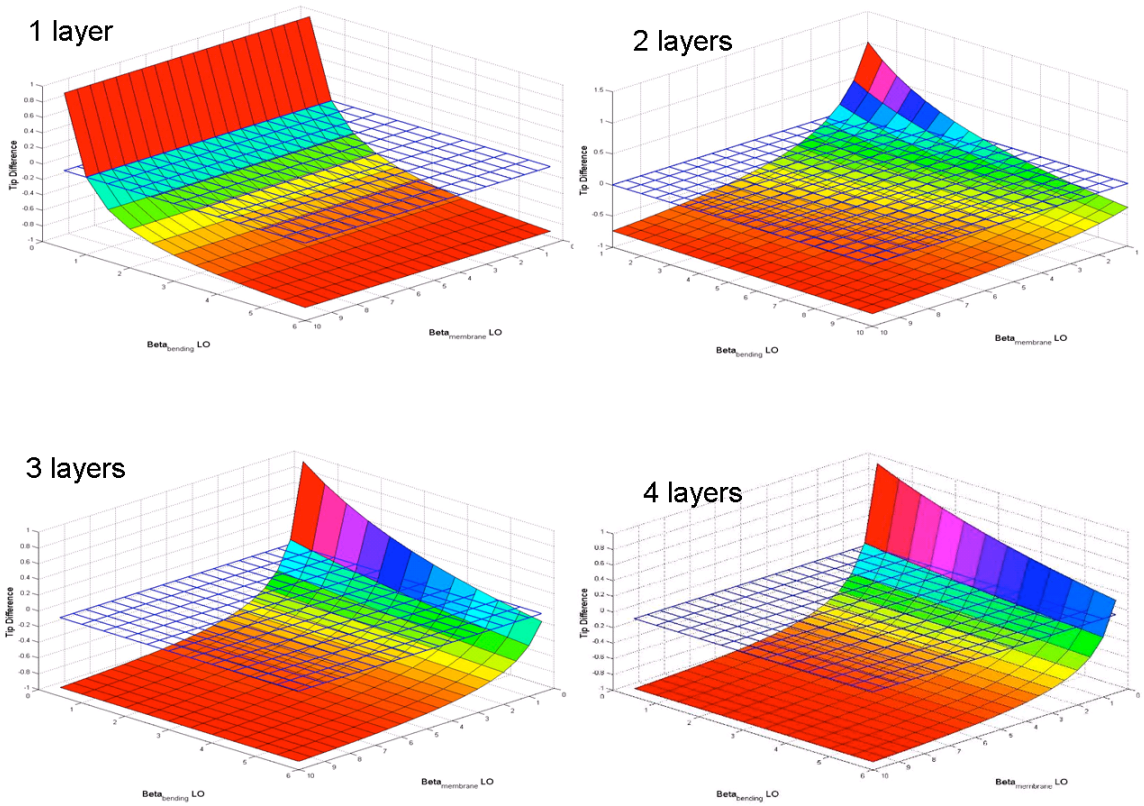


Figure 9: The relation between the bending and membrane lower order terms.

Figure 9 shows how the surface of the solution changes with the increase of number of layers. The first subfigure, for 1 layer, shows that the membrane stiffness doesn't factor into the accuracy of the solution at all. With 2 layers there is a more even trade off between the two beta factors. Yet by 4 layers the membrane stiffness dominates the bending stiffness in importance. This effect is also seen in Figure 8, where the slope of the bending terms past 2 layers is much softer than that of the membrane terms. This agrees with a physically intuitive understanding: as more layers are added to the stack, the membrane terms will become more important.

To generate a method for finding the correct beta term that will allow for confidence in the element, a complex surface arises. This surface is dependent on the two beta factors for each test as well as the number of layers in the stack. In order to develop a method that will allow for the beta factors to be found for various configurations, an optimizer was employed to find the best combination of beta values for a given problem and number of layers. Using the MATLAB fmincon.m function in the optimization toolbox, with the difference of the model solution to the analytical solution as the cost function, subject only to box constraints ensuring positive beta values, feasible values for the different beta terms were determined for the first 10 layers.

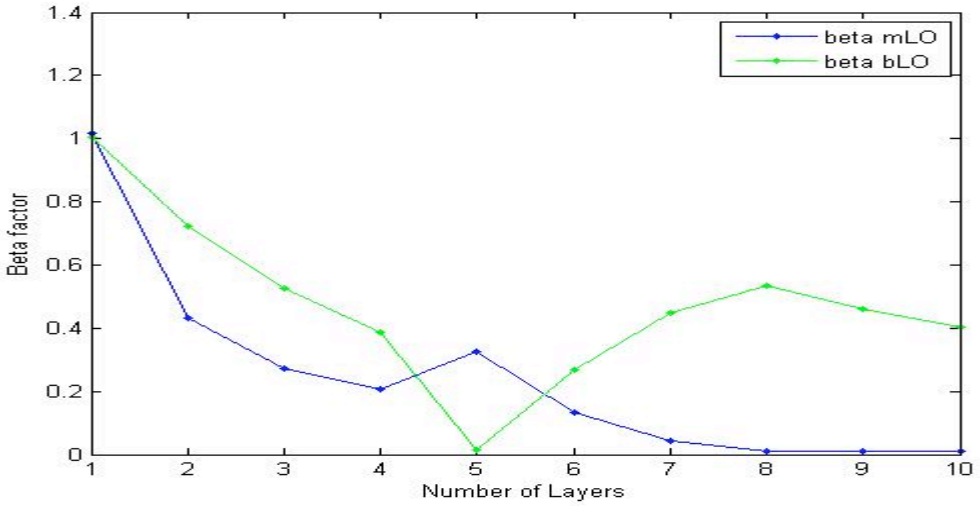


Figure 10: Beta values for in plane bending for layers 1-10.

Figure 10 shows two values that are on the intersection of the two planes showed in Figure 7. These are just two points on the curve that contains scaling factors that will match the analytical solution, but the goal was not to determine all points just to develop a method that will ensure that the element is on that curve. An analytical solution to this would be preferable and much more robust but was beyond the scope of this project.

The beta factors were shown to be a reasonable way to adjust the stacked shell element to avoid the complications that arise with multiple layers. The use of the beta factors does have some key draw backs. That different stiffness matrices are important for different types of loading is a major downfall; this means that a different relation for beta and number of layers has to be developed for each type of loading, and more complex loading would be an even greater challenge. There is also a lack of mathematical rigor about the solution found; the beta factor doesn't have any physical meaning and is serving as a patch or 'band aid' to help the shell element find the correct solution. If this term was able to be related back to physical properties, the understanding of how the element works would be greatly improved. Despite these drawbacks, this method does work well, yielding a solution that agrees to the seventh decimal place with the analytical solution and is easy to implement.

Reference:

1. Carlos A. Felippa, Ch.36, Advanced Finite Element Method

Appendix A: Matlab m-files

```
function [xl,yl,zl,dcm,area]=SM3SHLOCALSYS(xg,yg,zg)
% function [xl,yl,zl,dcm,area]=SM3SHLOCALSYS(xg,yg,zg)
% PURPOSE Define local system of 3-node triangle in 3D space
% AUTHOR C. A. Felippa, September 1996
%       Converted to Matlab from Mathematica, by S. Golmon, N. Ledford,
%          L. Liu, April 2009
% KEYWORDS finite element shell triangle local system direction cosines
%
% SM3SHLOCALSYS computes the local corner coordinates of a flat triangular
% element in 3D space and the direction cosines of the local system
%
% INPUTS:
%      XG,YG,ZG  Corner coordinates of triangle in global system
%
% OUTPUTS:
%      XL,YL,ZL   Corner coordinates of triangle in local system
%      DCM        Matrix of direction cosines of local system
%      AREA       Signed area of triangle
%
%    The local system is defined as follows:
%       x' is directed parallel to the 2-1 side
%       z' is the external normal (counterclockwise).
%       y' computed as z' x x'

x21=xg(2)-xg(1); y21=yg(2)-yg(1);
z21=zg(2)-zg(1); x32=xg(3)-xg(2);
y32=yg(3)-yg(2); z32=zg(3)-zg(2);
xlr=sqrt( x21^2+y21^2+z21^2 );
if xlr<=0
  error('Nodes 1-2 coincide');
end

dx(1)=x21/xlr; dx(2)=y21/xlr; dx(3)=z21/xlr;
dz(1)=y21*z32-z21*y32; dz(2)=z21*x32-x21*z32;
dz(3)=x21*y32-y21*x32;
zlr = sqrt( dz(1)^2+dz(2)^2+dz(3)^2 );
if zlr<=0
  error('Nodes 1-2-3 are colinear');
end

dz(1)=dz(1)/zlr; dz(2)=dz(2)/zlr; dz(3)=dz(3)/zlr;
dy(1)=dz(2)*dx(3)-dz(3)*dx(2);
dy(2)=dz(3)*dx(1)-dz(1)*dx(3);
dy(3)=dz(1)*dx(2)-dz(2)*dx(1);
ylr=sqrt(dy(1)^2+dy(2)^2+dy(3)^2);

dy(1)=dy(1)/ylr; dy(2)=dy(2)/ylr; dy(3)=dy(3)/ylr;
x0=(xg(1)+xg(2)+xg(3))/3;
y0=(yg(1)+yg(2)+yg(3))/3;
z0=(zg(1)+zg(2)+zg(3))/3;

for i=1:3
   xl(i)=dx(1)*(xg(i)-x0)+dx(2)*(yg(i)-y0)+dx(3)*(zg(i)-z0);
   yl(i)=dy(1)*(xg(i)-x0)+dy(2)*(yg(i)-y0)+dy(3)*(zg(i)-z0);
```

```
  zl(i)=dz(1)*(xg(i)-x0)+dz(2)*(yg(i)-y0)+dz(3)*(zg(i)-z0);
  dcm(1,i)=dx(i);
  dcm(2,i)=dy(i);
  dcm(3,i)=dz(i);
end

area=(1/2)*((xl(2)-xl(1))*(yl(3)-yl(1)) - (xl(3)-xl(1))*(yl(1)-yl(2)) );

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function db = SM3SHISODB(em,nu,h)
% PURPOSE Form isotropic bending constitutive matrix DB
% AUTHOR C. A. Felippa
%       Converted to Matlab from Mathematica, by S. Golmon, N. Ledford,
%         L. Liu, April 2009

c=em*h^3/(1-nu^2)/12;

db=[c,nu*c,0;
   nu*c,c,0;
   0,0,c*(1-nu)/2];

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function dm = SM3SHISODM(em, nu, h)
% DECK SM3SHISODM SM3SHISODM FORTRAN
% PURPOSE Form isotropic membrane constitutive matrix dm
% AUTHOR C. A. Felippa
%       Converted to Matlab from Mathematica, by S. Golmon, N. Ledford,
%         L. Liu, April 2009

c=em*h/(1-nu^2);
dm=[c,nu*c,0;
   nu*c,c,0;
   0,0,c*(1-nu)/2];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function sm = SM3SHTRANSFORM(esm,t1,t2,t3,t4,t5,t6)
% PURPOSE Transform stiffness of 3-node shell element to global axes
% AUTHOR C. A. Felippa, September 1996
%       Converted to Matlab from Mathematica, by S. Golmon, N. Ledford,
%         L. Liu, April 2009
% KEYWORDS finite element shell triangle local global transform
% BLOCK ABSTRACT
%
%   SM3SHTRANSFORM transforms the local element stiffness to
%   global coordinates.
%
%   The stiffness transformation is assumed to have the block
%   diagonal form
%
%   [T1' 0  0  0  0  0 ] [S11 S12 S13 S14 S15 S16] [T1  0  0  0  0  0]
%   [ 0 T2' 0  0  0  0 ] [    S22 S23 S24 S25 S26] [ 0 T2  0  0  0  0]
%   [ 0  0 T3' 0  0  0 ] [        S33 S34 S35 S36] [ 0  0 T3  0  0  0]
%   [ 0  0  0 T4' 0  0 ] [            S44 S45 S46] [ 0  0  0 T4  0  0]
```

```
%   [ 0  0  0  0 T5' 0 ] [              S55 S56] [ 0  0  0  0 T5  0]
%   [ 0  0  0  0  0 T6'] [   symm          S66] [ 0  0  0  0  0 T6]
%
%   where Ti are direction cosines of local directions wrt global.
%   Each block is 3 x 3. Indices 1, 3 and 5 pertains to the three
%   translations at corners 1, 2 and 3, respectively. Indices 2, 4
%   and 6 pertain to the three rotations at corners 1, 2 and 3.
%
%   The subroutine accounts for the possibility that T1, T2, ... T6
%   may be different, although they will often be the same.
%
%   The transformed block Sij is evidently Ti'.Sij.Tj, which can be
%   computed in 54 multiply-add operations.  For all blocks this
%   amounts to about 1000 operations. The implementation below uses
%   50% more operations: 1539, to streamline loops.  A brute force
%   version ignoring the block-diagonal form would require 2 x18^3/2
%   ~ 6000  operations.  Tests on the Sun show that the present
%   implementation is 8 times faster than the brute force, because
%   of the careful use of local variables to reduce indexing overhead.
%
% INPUTS:
%     SM            Incoming 18 x 18 matrix in local system
%     T1 through T6    Direction cosine matrices
%
% OUTPUT:
%     SM            Output 18 x 18 matrix in global system

sm = esm;

t=zeros(18,3);
st1=zeros(1,18);
st2=zeros(1,18);
st3=zeros(1,18);

%%
for i=1:3
   for j=1:3
      t(i,j,1)=t1(i,j);
      t(i,j,2)=t2(i,j);
      t(i,j,3)=t3(i,j);
      t(i,j,4)=t4(i,j);
      t(i,j,5)=t5(i,j);
      t(i,j,6)=t6(i,j);
   end
end

%%
for jb=1:6
   j=3*(jb-1);
   t11=t(1,1,jb);
   t21=t(2,1,jb);
   t31=t(3,1,jb);
   t12=t(1,2,jb);
   t22=t(2,2,jb);
   t32=t(3,2,jb);
   t13=t(1,3,jb);
```

```
    t23=t(2,3,jb);
    t33=t(3,3,jb);
    for i=1:18
        st1(i)=sm(i,j+1)*t11+sm(i,j+2)*t21+sm(i,j+3)*t31;
        st2(i)=sm(i,j+1)*t12+sm(i,j+2)*t22+sm(i,j+3)*t32;
        st3(i)=sm(i,j+1)*t13+sm(i,j+2)*t23+sm(i,j+3)*t33;
    end

    for ib=1:jb,
        i=3*(ib-1);
        t11=t(1,1,ib); t21=t(2,1,ib); t31=t(3,1,ib);
        t12=t(1,2,ib); t22=t(2,2,ib); t32=t(3,2,ib);
        t13=t(1,3,ib); t23=t(2,3,ib); t33=t(3,3,ib);
        tst11=t11*st1(i+1)+t21*st1(i+2)+t31*st1(i+3);
        tst21=t12*st1(i+1)+t22*st1(i+2)+t32*st1(i+3);
        tst31=t13*st1(i+1)+t23*st1(i+2)+t33*st1(i+3);
        tst12=t11*st2(i+1)+t21*st2(i+2)+t31*st2(i+3);
        tst22=t12*st2(i+1)+t22*st2(i+2)+t32*st2(i+3);
        tst32=t13*st2(i+1)+t23*st2(i+2)+t33*st2(i+3);
        tst13=t11*st3(i+1)+t21*st3(i+2)+t31*st3(i+3);
        tst23=t12*st3(i+1)+t22*st3(i+2)+t32*st3(i+3);
        tst33=t13*st3(i+1)+t23*st3(i+2)+t33*st3(i+3);
        sm(i+1,j+1)=tst11; sm(j+1,i+1)=tst11;
        sm(i+1,j+2)=tst12; sm(j+2,i+1)=tst12;
        sm(i+1,j+3)=tst13; sm(j+3,i+1)=tst13;
        sm(i+2,j+1)=tst21; sm(j+1,i+2)=tst21;
        sm(i+2,j+2)=tst22; sm(j+2,i+2)=tst22;
        sm(i+2,j+3)=tst23; sm(j+3,i+2)=tst23;
        sm(i+3,j+1)=tst31; sm(j+1,i+3)=tst31;
        sm(i+3,j+2)=tst32; sm(j+2,i+3)=tst32;
        sm(i+3,j+3)=tst33; sm(j+3,i+3)=tst33;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function sm = SM3SHBENDB(x,y,db,f,clr,cqr,ls,esm)
% DECK SM3SHBENDB SM3SHBENDB FORTRAN
% PURPOSE Form SM3SHBENDB bending stiffness of 3-node, 9 dof Kirchhoff triangle
% AUTHOR C. A. Felippa, May 1984
% VERSION September 1989
%       Converted to Matlab from Mathematica, by S. Golmon, N. Ledford,
%         L. Liu, April 2009
% KEYWORDS Kirchhoff thin plate bending, finite element triangle basic stiffness matrix
%
% SM3SHBENDB forms the basic material element stiffness matrix
%    of a Kirchhoff thin plate bending triangle constructed with the
%    generalized FF/ANS formulation
%
% INPUTS:
%      X        (3 x 1) array of x coordinates of triangle nodes
%      Y        (3 x 1) array of y coordinates of triangle nodes
%      DB       (3 x 3) moment-curvature constitutive matrix
%      F        Factor by which stiffness entries will be multiplied.
%      CLR,CQR   Use CLR*LLR+CQR*LQR for lumping matrix L
%               thus CLR+CQR  must add up to 1.
```

```
%               LLR=linear rotation, LLR=quadratic rotation (Kirchhoff)
%       LS        (9 x 1) array of stiffness location pointers
%               (see output SM).
%       esm       Incoming material stiffness array.
% OUTPUTS:
%       SM        Output stiffness array with basic stiffness
%               coefficients added in.  the (I,J)-th entry of the
%               (9 x 9) element bending stiffness is added to
%               SM(K,L), where K=LS(I) and L=LS(J).
%       STATUS    Status character variable.  blank if no error detected.

Llr=zeros(9,3);
Lqr=zeros(9,3);
sm=esm;

x21 = x(2) - x(1);
x12 =-x21;
x32 = x(3) - x(2);
x23 =-x32;
x13 = x(1) - x(3);
x31 =-x13;
y21 = y(2) - y(1);
y12 =-y21;
y32 = y(3) - y(2);
y23 =-y32;
y13 = y(1) - y(3);
y31 =-y13;
area2 = y21*x13 - x21*y13;

if area2<=0
   error('Negative or zero area');
end

x0=(x(1)+x(2)+x(3))/3;
y0=(y(1)+y(2)+y(3))/3;
cab=3/area2;
a1=-cab*(y(3)-y0);
a2=-cab*(y(1)-y0);
a3=-cab*(y(2)-y0);
b1= cab*(x(3)-x0);
b2= cab*(x(1)-x0);
b3= cab*(x(2)-x0);
xl12=sqrt(x12^2+y12^2);
xl23=sqrt(x23^2+y23^2);
xl31=sqrt(x31^2+y31^2);
Llr(3,1)=y32/2;  Llr(6,1)=y13/2;  Llr(9,1)=y21/2;
Llr(2,2)=x32/2;  Llr(5,2)=x13/2;  Llr(8,2)=x21/2;
Llr(2,3)=-y32/2; Llr(3,3)=-x32/2; Llr(5,3)=-y13/2;
Llr(6,3)=-x13/2; Llr(8,3)=-y21/2; Llr(9,3)=-x21/2;

c12 =y21/xl12; s12 =x12/xl12; c23 =y32/xl23;
s23 =x23/xl23; c31 =y13/xl31; s31 =x31/xl31;
cc12 =c12*c12; cc23 =c23*c23; cc31 =c31*c31;
ss12 =s12*s12; ss23 =s23*s23; ss31 =s31*s31;
cs12 =c12*s12; cs23 =c23*s23; cs31 =c31*s31;
Lqr(1,1)=cs12-cs31;
```

```
Lqr(1,2)=-Lqr(1,1);
Lqr(1,3)=(cc31-ss31)-(cc12-ss12);
Lqr(2,1)=(cc12*x12+cc31*x31)/2;
Lqr(2,2)=(ss12*x12+ss31*x31)/2;
Lqr(2,3)=ss12*y21+ss31*y13;
Lqr(3,1)=-(cc12*y21+cc31*y13)/2;
Lqr(3,2)=-(1/2)*Lqr(2,3);
Lqr(3,3)=-2*Lqr(2,1);
Lqr(4,1)=cs23-cs12;
Lqr(4,2)=-Lqr(4,1);
Lqr(4,3)=(cc12-ss12)-(cc23-ss23);
Lqr(5,1)=(cc12*x12+cc23*x23)/2;
Lqr(5,2)=(ss12*x12+ss23*x23)/2;
Lqr(5,3)= ss12*y21+ss23*y32;
Lqr(6,1)=-(cc12*y21+cc23*y32)/2;
Lqr(6,2)=-(1/2)*Lqr(5,3);
Lqr(6,3)=-2*Lqr(5,1);
Lqr(7,1)= cs31-cs23;
Lqr(7,2)=-Lqr(7,1);
Lqr(7,3)=(cc23-ss23)-(cc31-ss31);
Lqr(8,1)=(cc23*x23+cc31*x31)/2;
Lqr(8,2)=(ss23*x23+ss31*x31)/2;
Lqr(8,3)=ss23*y32+ss31*y13;
Lqr(9,1)=-(cc23*y32+cc31*y13)/2;
Lqr(9,2)=-(1/2)*Lqr(8,3);
Lqr(9,3)=-2*Lqr(8,1);

L=clr*Llr+cqr*Lqr;

tmp = db*2*f/area2;
db11 = tmp(1,1);
db12 = tmp(1,2);
db13 = tmp(1,3);
db21 = tmp(2,1);
db22 = tmp(2,2);
db23 = tmp(2,3);
db31 = tmp(3,1);
db32 = tmp(3,2);
db33 = tmp(3,3);

for j=1:9
   jj=ls(j);
   s1=db11*L(j,1)+db12*L(j,2)+db13*L(j,3);
   s2=db12*L(j,1)+db22*L(j,2)+db23*L(j,3);
   s3=db13*L(j,1)+db23*L(j,2)+db33*L(j,3);
   for i=1:j
       ii=ls(i);
       sm(jj,ii)=sm(jj,ii)+s1*L(i,1)+s2*L(i,2)+s3*L(i,3);
       sm(ii,jj)=sm(jj,ii);
   end
end

sm=1*sm;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function sm = SM3SHMEMBB(x,y,dm,alpha,f,ls,esm)
% DECK SM3SHMEMBB
% PURPOSE Form basic membrane stiffness of 9-dof triangle
% AUTHOR C. A. Felippa, June 1984
%       Converted to Matlab from Mathematica, by S. Golmon, N. Ledford,
%         L. Liu, April 2009
% VERSION June 1984
% KEYWORDS finite element membrane plane stress
% KEYWORDS basic material stiffness matrix
%
%    SM3SHMEMBB forms the material element stiffness matrix associated with
%    the basic displacement modes (rigid modes + constant strain
%    modes) of a 9-dof plane-stress triangle based on the
%    free formulation of Bergan and Nygard.
%
% INPUTS:
%      X        (3 x 1) array of x coordinates of triangle nodes.
%      Y        (3 x 1) array of y coordinates of triangle nodes.
%      DM       (3 x 3) matrix relating in-plane forces to strains.
%      ALPHA    Rotational lumping factor; if zero form CST.
%      F        Factor by which stiffness entries will be multiplied.
%      LS       (9 x 1) array of stiffness location pointers, see output SM
%      ESM      Incoming material stiffness array.
%
% OUTPUTS
%      SM       Output stiffness array with basic stiffness
%               coefficients added in.  The (i,j)-th entry of the
%               basic element stiffness is added to SM(K,L),
%               where K=LS(I) and L=LS(J).
%
%    Note: the internal ordering of degrees of freedom is
%        ux1,uy1, ux2,uy2, ux3,uy3, thetaz1,thetaz2,thetaz3
%    This simplifies the stiffness formation loop if alpha=0.

sm=esm;

x21 = x(2)-x(1);
x32 = x(3)-x(2);
x13 = x(1)-x(3);
x12 = -x21;
x23 = -x32;
x31 = -x13;
y21 = y(2)-y(1);
y32 = y(3)-y(2);
y13 = y(1)-y(3);
y12 = -y21;
y23 = -y32;
y31 = -y13;
area2=y21*x13-x21*y13;
if area2<=0
   error('Negative or zero area');
end

p= [y23,0,x32;
   0,x32,y23;
   y31,0,x13;
```

```
   0,x13,y31;
   y12,0,x21;
   0,x21,y12;
   y23*(y13-y21)*alpha/6,x32*(x31-x12)*alpha/6,(x31*y13-x12*y21)*alpha/3;
   y31*(y21-y32)*alpha/6,x13*(x12-x23)*alpha/6,(x12*y21-x23*y32)*alpha/3;
   y12*(y32-y13)*alpha/6,x21*(x23-x31)*alpha/6,(x23*y32-x31*y13)*alpha/3];

tmp = dm*(f/(2*area2));
dm11 = tmp(1,1);
dm12 = tmp(1,2);
dm13 = tmp(1,3);
dm21 = tmp(2,1);
dm22 = tmp(2,2);
dm23 = tmp(2,3);
dm31 = tmp(3,1);
dm32 = tmp(3,2);
dm33 = tmp(3,3);

for j=1:9
   l=ls(j);
   s1=dm11*p(j,1)+dm12*p(j,2)+dm13*p(j,3);
   s2=dm12*p(j,1)+dm22*p(j,2)+dm23*p(j,3);
   s3=dm13*p(j,1)+dm23*p(j,2)+dm33*p(j,3);
   for i=1:j,
      k=ls(i);
      sm(k,l)=sm(k,l)+s1*p(i,1)+s2*p(i,2)+s3*p(i,3);
      sm(l,k)=sm(k,l);
   end
end

sm = 1*sm;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function sm = SM3SHMEMBH(x,y,dm,f,ls,esm)
% PURPOSE Form H.O. material stiffness of 9-dof ANDES membrane triangle
% AUTHOR C. A. Felippa, June 1991
%       Converted to Matlab from Mathematica, by S. Golmon, N. Ledford,
%         L. Liu, April 2009
% VERSION July 1991
%       Converted to Matlab from Mathematica, by S. Golmon, N. Ledford,
%         L. Liu, April 2009
% KEYWORDS finite element
%         material stiffness matrix high-order
%         triangle membrane assumed natural deviatoric strain
% SM3MANDES forms the higher order element stiffness matrix
%         of a 9-dof membrane triangle based on the ANDES formulation.
%         Implementation moderately optimized for speed.
%
%    INPUTS:
%    X        (3 x 1) array of x coordinates of triangle nodes
%    Y        (3 x 1) array of y coordinates of triangle nodes
%    DM       (3 x 3) membrane constitutive matrix already
%             integrated through the thickness
%    F        Factor by which all stiffness entries will be multiplied.
%    SM       Incoming material stiffness array.
```

```
%    LS       (9 x 1) array of stiffness location pointers
%             (see examples in SM3SHMEMBB)
%    M        First dimension of SM in calling program.
%
%    OUTPUTS:
%    SM       Output stiffness array with higher order stiffness
%             coefficients added in.
%             The (i,j)-th entry of the basic element stiffness is added
%             to SM(K,L), where K=LS(I) and L=LS(J).

d=[0,0,0];s=[0,0,0];w=[0,0,0];
e=zeros(3);c=zeros(3);t=zeros(3);kth=zeros(3);
qm=zeros(3,3,3);
sm=esm;

x21=x(2)-x(1);x31=x(3)-x(1);x32=x(3)-x(2);
y21=y(2)-y(1);y31=y(3)-y(1);y32=y(3)-y(2);
x12=-x21;x13=-x31;x23=-x32;
y12=-y21;y13=-y31;y23=-y32;

area2=y21*x13-x21*y13;
if area2<=0
  error('Negative or zero area');
end

area=area2/2;

ll21=x21^2+y21^2; ll32=x32^2+y32^2; ll13=x13^2+y13^2;
tfac=1/(4*area^2);
t=[y23*y13*ll21,y31*y21*ll32,y12*y32*ll13;
  x23*x13*ll21,x31*x21*ll32,x12*x32*ll13;
  (y23*x31+x32*y13)*ll21,(y31*x12+x13*y21)*ll32,(y12*x23+x21*y32)*ll13]*tfac;
wfac=(3/4)*f*area;
e=dm*wfac;

for j=1:3
   et1=e(1,1)*t(1,j)+e(1,2)*t(2,j)+e(1,3)*t(3,j);
   et2=e(2,1)*t(1,j)+e(2,2)*t(2,j)+e(2,3)*t(3,j);
   et3=e(3,1)*t(1,j)+e(3,2)*t(2,j)+e(3,3)*t(3,j);
   for i=1:3
      c(i,j)=t(1,i)*et1+t(2,i)*et2+t(3,i)*et3;
   end
end

area43=(2/3)*area2;
chi213=area43/ll21;chi321=area43/ll32;chi132=area43/ll13;
qm(1,1,1)=-(1/4)*chi213; qm(1,2,1)=-qm(1,1,1);
qm(2,1,1)= (1/4)*chi321; qm(2,2,1)= (1/2)*chi321;
qm(2,3,1)= qm(2,1,1);  qm(3,1,1)=-(1/2)*chi132;
qm(3,2,1)=-(1/4)*chi132; qm(3,3,1)= qm(3,2,1);
qm(1,1,2)=-(1/4)*chi213; qm(1,2,2)=-(1/2)*chi213;
qm(1,3,2)= qm(1,1,2);  qm(2,2,2)=-(1/4)*chi321;
qm(2,3,2)=-qm(2,2,2);  qm(3,1,2)= (1/4)*chi132;
qm(3,2,2)= qm(3,1,2);  qm(3,3,2)= (1/2)*chi132;
qm(1,1,3)= (1/2)*chi213; qm(1,2,3)= (1/4)*chi213;
qm(1,3,3)= qm(1,2,3);  qm(2,1,3)=-(1/4)*chi321;
```

```
qm(2,2,3)= qm(2,1,3);  qm(2,3,3)=-(1/2)*chi321;
qm(3,1,3)= (1/4)*chi132; qm(3,3,3)=-qm(3,1,3);
c11=c(1,1);c12=c(1,2);c13=c(1,3);
c21=c(2,1);c22=c(2,2);c23=c(2,3);
c31=c(3,1);c32=c(3,2);c33=c(3,3);

    for k=1:3
       for j=1:3
          d=[c11*qm(1,j,k)+c12*qm(2,j,k)+c13*qm(3,j,k),...
             c21*qm(1,j,k)+c22*qm(2,j,k)+c23*qm(3,j,k),...
             c31*qm(1,j,k)+c32*qm(2,j,k)+c33*qm(3,j,k)];
          for i=1:j
              kth(i,j)=kth(i,j)+qm(1,i,k)*d(1)+qm(2,i,k)*d(2)+qm(3,i,k)*d(3);
              kth(j,i)=kth(i,j);
          end
       end
    end

 s=[kth(1,1)+kth(1,2)+kth(1,3),...
    kth(2,1)+kth(2,2)+kth(2,3),...
    kth(3,1)+kth(3,2)+kth(3,3)];
 xyij=[x32,y32,x13,y13,x21,y21]/(4*area);
 for j=1:9
    l=ls(j);
       for i=1:3
          if j<=6
             w(i)=s(i)*xyij(j);
          else w(i)=kth(i,j-6);
          end
       end
    sum=w(1)+w(2)+w(3);
    for i=1:j
       k=ls(i);
       if  i <=6
          sm(k,l)=sm(k,l)+sum*xyij(i);
       else sm(k,l)=sm(k,l)+w(i-6);
       end
       sm(l,k)=sm(k,l);
    end
 end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function sm = SM3SHBENDH(x,y,db,f,ls,esm)
% DECK SM3SHBENDH
% PURPOSE Form Kh for 3-node Kirchhoff ANDES plate bending elem
% AUTHOR C. Militello and C. Felippa, May 1989
% VERSION July 1989
%       Converted to Matlab from Mathematica, by S. Golmon, N. Ledford,
%         L. Liu, April 2009
% KEYWORDS thin plate bending, finite element triangle higher order stiffness matrix
%
% SM3SHBENDH forms the higher order material stiffness matrix of a
%     9-dof thin-plate-bending triangle obtained by using linear
%     curvatures over the sides.  Implementation optimized for speed
%
```

```
% INPUTS:
%       X        (3 x 1) array of x coordinates of triangle nodes
%       Y        (3 x 1) array of y coordinates of triangle nodes
%       DB       (3 x 3) moment-curvature matrix.
%       F        Factor by which stiffness entries will be multiplied.
%       LS       (9 x 1) array of stiffness location pointers
%                (see Output SM).
%       SM       Incoming material stiffness array.
%       M        First dimension of SM in calling program.
%
% OUTPUTS:
%       SM       Output stiffness array with higher order stiffness
%                coefficients added in.  The (i,j)-th entry of the
%                (9 by 9) element bending stiffness is added to
%                SM(K,L), where K=LS(I) and L=LS(J).

t = zeros(3);
sh = zeros(9);
sm = esm;

x21 =  x(2) - x(1);
x12 = -x21;
x32 =  x(3) - x(2);
x23 = -x32;
x13 =  x(1) - x(3);
x31 = -x13;
y21 =  y(2) - y(1);
y12 = -y21;
y32 =  y(3) - y(2);
y23 = -y32;
y13 =  y(1) - y(3);
y31 = -y13;
area2=y21*x13-x21*y13;

if area2<=0
   error('Negative or zero area');
end

lam12=(x12*x13+y12*y13)/(x21*x21+y21*y21);
lam23=(x23*x21+y23*y21)/(x32*x32+y32*y32);
lam31=(x31*x32+y31*y32)/(x13*x13+y13*y13);
mu11=2*(lam12^2+1-lam12);
mu22=2*(lam23^2+1-lam23);
mu33=2*(lam31^2+1-lam31);
mu12=2*lam12-1-(1+lam12)*lam23;
mu23=2*lam23-1-(1+lam23)*lam31;
mu13=2*lam31-1-(1+lam31)*lam12;

t=[y23*y13,y31*y21,y12*y32;
 x23*x13,x31*x21,x12*x32;
 x31*y23+x32*y13,x12*y31+x13*y21,x23*y12+x21*y32]/area2^2;
dbt11 = db(1,1) *t(1,1)+db(1,2) *t(2,1)+db(1,3) *t(3,1);
dbt12 = db(1,1) *t(1,2)+db(1,2) *t(2,2)+db(1,3) *t(3,2);
dbt13 = db(1,1) *t(1,3)+db(1,2) *t(2,3)+db(1,3) *t(3,3);
dbt21 = db(2,1) *t(1,1)+db(2,2) *t(2,1)+db(2,3) *t(3,1);
dbt22 = db(2,1) *t(1,2)+db(2,2) *t(2,2)+db(2,3) *t(3,2);
```

```
dbt23 = db(2,1) *t(1,3)+db(2,2) *t(2,3)+db(2,3) *t(3,3);
dbt31 = db(3,1) *t(1,1)+db(3,2) *t(2,1)+db(3,3) *t(3,1);
dbt32 = db(3,1) *t(1,2)+db(3,2) *t(2,2)+db(3,3) *t(3,2);
dbt33 = db(3,1) *t(1,3)+db(3,2) *t(2,3)+db(3,3) *t(3,3);

area=area2/2; cfac=area*f;
c11 = cfac*mu11* (t(1,1) *dbt11+t(2,1) *dbt21+t(3,1) *dbt31);
c12 = cfac*mu12* (t(1,2) *dbt11+t(2,2) *dbt21+t(3,2) *dbt31);
c13 = cfac*mu13* (t(1,3) *dbt11+t(2,3) *dbt21+t(3,3) *dbt31);
c22 = cfac*mu22* (t(1,2) *dbt12+t(2,2) *dbt22+t(3,2) *dbt32);
c23 = cfac*mu23* (t(1,3) *dbt12+t(2,3) *dbt22+t(3,3) *dbt32);
c33 = cfac*mu33* (t(1,3) *dbt13+t(2,3) *dbt23+t(3,3) *dbt33);

sh(1,1) =  4*(c33-c13-c13+c11);
sh(1,2) =  2*((c11-c13)*y21+(c13-c33)*y13);
sh(1,3) =  2*((c13-c11)*x21+(c33-c13)*x13);
sh(1,4) =  4*(-c23+c13+c12-c11);
sh(1,5) =  2*((c12-c23)*y32+(c11-c13)*y21);
sh(1,6) =  2*((c23-c12)*x32+(c13-c11)*x21);
sh(1,7) =  4*(-c33+c23+c13-c12);
sh(1,8) =  2*((c12-c23)*y32+(c13-c33)*y13);
sh(1,9) =  2*((c23-c12)*x32+(c33-c13)*x13);
sh(2,2) =  c11*y21^2+2*c13*y13*y21+c33*y13^2;
sh(2,3) =  (-c11*x21-c13*x13)*y21+(-c13*x21-c33*x13)*y13;
sh(2,4) =  2*((c12-c11)*y21+(c23-c13)*y13);
sh(2,5) =  (c12*y21+c23*y13)*y32+c11*y21^2+c13*y13*y21;
sh(2,6) =  (-c12*x32-c11*x21)*y21+(-c23*x32-c13*x21)*y13;
sh(2,7) =  2*((c13-c12)*y21+(c33-c23)*y13);
sh(2,8) =  (c12*y21+c23*y13)*y32+c13*y13*y21+c33*y13^2;
sh(2,9) =  (-c12*x32-c13*x13)*y21+(-c23*x32-c33*x13)*y13;
sh(3,3) =  c11*x21^2+2*c13*x13*x21+c33*x13^2;
sh(3,4) =  2*((c11-c12)*x21+(c13-c23)*x13);
sh(3,5) =  (-c12*x21-c23*x13)*y32+(-c11*x21-c13*x13)*y21;
sh(3,6) =  (c12*x21+c23*x13)*x32+c11*x21^2+c13*x13*x21;
sh(3,7) =  2*((c12-c13)*x21+(c23-c33)*x13);
sh(3,8) =  (-c12*x21-c23*x13)*y32+(-c13*x21-c33*x13)*y13;
sh(3,9) =  (c12*x21+c23*x13)*x32+c13*x13*x21+c33*x13^2;
sh(4,4) =  4*(c22-c12-c12+c11);
sh(4,5) =  2*((c22-c12)*y32+(c12-c11)*y21);
sh(4,6) =  2*((c12-c22)*x32+(c11-c12)*x21);
sh(4,7) =  4*(c23-c22-c13+c12);
sh(4,8) =  2*((c22-c12)*y32+(c23-c13)*y13);
sh(4,9) =  2*((c12-c22)*x32+(c13-c23)*x13);
sh(5,5) =  c22*y32^2+2.*c12*y21*y32+c11*y21^2;
sh(5,6) =  (-c22*x32-c12*x21)*y32+(-c12*x32-c11*x21)*y21;
sh(5,7) =  2*((c23-c22)*y32+(c13-c12)*y21);
sh(5,8) =  c22*y32^2+(c12*y21+c23*y13)*y32+c13*y13*y21;
sh(5,9) =  (-c22*x32-c23*x13)*y32+(-c12*x32-c13*x13)*y21;
sh(6,6) =  c22*x32^2+2.*c12*x21*x32+c11*x21^2;
sh(6,7) =  2*((c22-c23)*x32+(c12-c13)*x21);
sh(6,8) =  (-c22*x32-c12*x21)*y32+(-c23*x32-c13*x21)*y13;
sh(6,9) =  c22*x32^2+(c12*x21+c23*x13)*x32+c13*x13*x21;
sh(7,7) =  4*(c33-c23-c23+c22);
sh(7,8) =  2*((c23-c22)*y32+(c33-c23)*y13);
sh(7,9) =  2*((c22-c23)*x32+(c23-c33)*x13);
sh(8,8) =  c22*y32^2+2.*c23*y13*y32+c33*y13^2;
```

```
sh(8,9) =  (-c22*x32-c23*x13)*y32+(-c23*x32-c33*x13)*y13;
sh(9,9) =  c22*x32^2+2.*c23*x13*x32+c33*x13^2;

for i=1:9
   k=ls(i);
   for j=i:9
      l=ls(j);
      sm(k,l)=sm(k,l)+sh(i,j);
      sm(l,k)=sm(k,l);
   end
end
sm=1*sm;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [esm,T]=SM3SHELL(xg ,yg ,zg ,em ,nu ,h)
% PURPOSE Form  stiffness of 3-node, 18-dof flat triangle shell
% AUTHOR C. A. Felippa, October 1996
%       Converted to Matlab from Mathematica, by S. Golmon, N. Ledford,
%         L. Liu, April 2009
% KEYWORDS Kirchhoff thin shell isotropic homogeneous finite element triangle stiffness matrix
%
% SM3SHBENDB forms the material element stiffness matrix
%    of a 3-node, 18-dof triangular shell constructed with the
%    generalized FF/ANS formulation.  Isotropic material,
%
%    The input arguments are:
%
%      xg       (3 x 1) array of x coordinates of triangle nodes
%      yg       (3 x 1) array of y coordinates of triangle nodes
%      zg       (3 x 1) array of z coordinates of triangle nodes
%      em       Elastic modulus
%      nu       Poisson's ratio
%      h        Corner thicknesses
%
%    The outputs are:
%      esm       Output stiffness array with basic stiffness
%                coefficients added in.  the (I,J)-th entry of the
%                (9 x 9) element bending stiffness is added to
%                SM(K,L), where K=LS(I) and L=LS(J).
%      T        Transformation matrix

lb=[3,4,5, 9,10,11, 15,16,17];
le=[1,2, 7,8, 13,14, 6,12,18];
[xlp,ylp,zlp,dcm,area]=SM3SHLOCALSYS(xg,yg,zg);

if ~isempty(h)
    h0=(h(1)+h(2)+h(3))/3;
else
    h0=h;
end

dm=SM3SHISODM(em,nu,h0);
db=SM3SHISODB(em,nu,h0);
esm=zeros(18,18);
betab=max(1/100,(1-4*nu^2)/2);
```

```
alphab=3/2;
esm=SM3SHMEMBB(xlp,ylp, dm, alphab, 1, le, esm);
esm=SM3SHMEMBH(xlp,ylp, dm, betab,  le, esm);
esm=SM3SHBENDB(xlp,ylp, db, 1, 0, 1, lb,esm);
esm=SM3SHBENDH(xlp,ylp, db, 1, lb, esm);

T=eye(18);

for i=1:3
    T((i - 1)*6 + 1, (i - 1)*6 + 5) = zg(1);
   T((i - 1)*6 + 2, (i - 1)*6 + 4) = -zg(1);
end
esm=T'*esm*T;

esm=SM3SHTRANSFORM(esm, dcm,dcm,dcm,dcm,dcm,dcm);
```