

Finite Element Analysis of Membrane Structures

R.L. Taylor*

Department of Civil and Environmental Engineering
University of California at Berkeley, USA

May 27, 2001

Abstract

This report summarizes the formulation for a large displacement formulation of a membrane composed of three-node triangular elements. A formulation in terms of the deformation gradient is first constructed in terms of nodal variables. In particular, the use of the right Cauchy-Green deformation tensor is shown to lead to a particularly simple representation in terms of nodal quantities. This may then be used to construct general models for use in static and transient analyses.

1 Introduction

The behavior of curved, thin bodies can be modeled by a membrane theory of shells. In such a theory only the in-plane stress resultants are included. The deformation state for a membrane may be represented by the position of points on the two-dimensional surface. General theories for shells may be specialized to those for a membrane by ignoring the resultant couples and associated changes in curvature deformations as well as any transverse shearing effects. A numerical approximation to the shell may then be constructed using a finite element approach. Examples for general shell theory and finite element solution may be found for small deformations in standard books.^[1] Theory for large deformation can proceed following the presentations of Simo *et al.*^[2, 3, 4, 5] or Ramm *et al.*^[6, 7, 8, 9, 10, 11, 12, 13]

*Visiting Professor, CIMNE, UPC, Barcelona, Spain.
e-mail: rlt@ce.vulture.berkeley.edu

For the simplest shape finite element composed of a 3-node triangular form with displacement parameters at each vertex (a 9-degree of freedom element) it is far simpler to formulate the membrane behavior directly. This is especially true for large displacement response. Here the initially flat form of a simple triangular response is maintained throughout all deformation states. Consequently, one may proceed directly with the construction of the kinematic behavior, even in the presence of large strains. This approach is followed in the present work.

The loading of a membrane is often by pressures which remain normal to the surface throughout all deformations. Such *follower* loading generally leads to a form in which yields an unsymmetric tangent matrix. Such formulation has been presented in works by Schweizerhof and Ramm^[14] and by Simo *et al.*^[15] The general approach presented in the last work is used for the special case of the flat triangular element used in this work.

The formulation included in the present study includes inertial and damping terms based on second and first time derivatives of the motion. These are discretized in space using standard techniques (e.g., the Newmark method^[16, 17]). Both explicit and implicit schemes are presented together with all linearizations needed to implement a full Newton type solution. The inclusion of the damping term permits solution of the first order form in order to obtain a final static solution. Generally, the first order form is used until the final state is reached at which point the rate terms are deleted and the full static solution achieved using a standard Newton iterative method.

The work presented is implemented in the general purpose finite element solution system *FEAP*^[18] and used to solve example problems. The solution to some basic example problems are included to show the behavior of the element and solution strategies developed.

2 Governing Equations

Reference configuration coordinates in the global Cartesian frame are indicated in upper case by \mathbf{X} and current configuration in lower case by \mathbf{x} . The difference between the coordinates defines a displacement \mathbf{u} .

Using standard interpolation for a linear triangle positions in the element may be specified as

$$\mathbf{X} = \xi_\alpha \tilde{\mathbf{X}}^\alpha \tag{1}$$

in the reference configuration and

$$\mathbf{x} = \xi_\alpha \tilde{\mathbf{x}}^\alpha \tag{2}$$

for the current configurations. If necessary, the displacement vector may be deduced as

$$\mathbf{u} = \xi_\alpha \tilde{\mathbf{u}}^\alpha \quad (3)$$

In the above $\tilde{\mathbf{X}}^\alpha$, $\tilde{\mathbf{x}}^\alpha$, $\tilde{\mathbf{u}}^\alpha$ denote *nodal* values of the reference coordinates, current coordinates and displacement vector, respectively. Furthermore, the natural (*area*) coordinates satisfy the constraint

$$\xi_1 + \xi_2 + \xi_3 = 1 \quad (4)$$

It is convenient to introduce a surface coordinate system denoted by Y_1 , Y_2 with normal direction N in the reference state and y_1 , y_2 with normal direction n in the current state (see Fig. 1).

Placing the origin of the Y_I and y_i coordinates at nodal location $\tilde{\mathbf{X}}^1$ and $\tilde{\mathbf{x}}^1$, respectively, the unit base vectors may be constructed from the linear displacement triangle by aligning the first vector along the 1 – 2 side. Accordingly, we define the first unit vector as

$$\mathbf{v}_1 = \frac{\tilde{\mathbf{x}}^2 - \tilde{\mathbf{x}}^1}{\|\tilde{\mathbf{x}}^2 - \tilde{\mathbf{x}}^1\|} \quad (5)$$

where

$$\|\mathbf{a}\| = (\mathbf{a}^T \mathbf{a})^{1/2}$$

Next a vector *normal* to the triangle is constructed as

$$\mathbf{v}_3 = (\tilde{\mathbf{x}}^2 - \tilde{\mathbf{x}}^1) \times (\tilde{\mathbf{x}}^3 - \tilde{\mathbf{x}}^1) \quad (6)$$

and normalized to a unit vector as

$$\mathbf{n} = \frac{\mathbf{v}_3}{\|\mathbf{v}_3\|} \quad (7)$$

Finally, the second unit vector for the plane of the triangle may be computed as

$$\mathbf{v}_2 = \mathbf{n} \times \mathbf{v}_1 \quad (8)$$

The vector \mathbf{v}_3 plays a special role in later development of nodal forces for *follower pressure loading* as it is twice the area of the triangle times the unit normal vector \mathbf{n} .

The above developments have been performed based on the current configuration. However, reference quantities may be deduced by replacing lower case letters by upper case ones.

With the above base vectors defined for the plane of the triangle, positions may be given directly as

$$y_i = (\mathbf{x} - \tilde{\mathbf{x}}^1) \cdot \mathbf{v}_i \quad (9)$$

In general, an interpolation may be given as

$$\mathbf{y} = \xi_\alpha \tilde{\mathbf{y}}^\alpha \quad (10)$$

We note from Eq. (9) that $\tilde{\mathbf{y}}^1$ is identically zero hence Eq. (10) reduces to

$$\mathbf{y} = \xi_2 \tilde{\mathbf{y}}^2 + \xi_3 \tilde{\mathbf{y}}^3 \quad (11)$$

and it is no longer necessary to use the constraint (4).

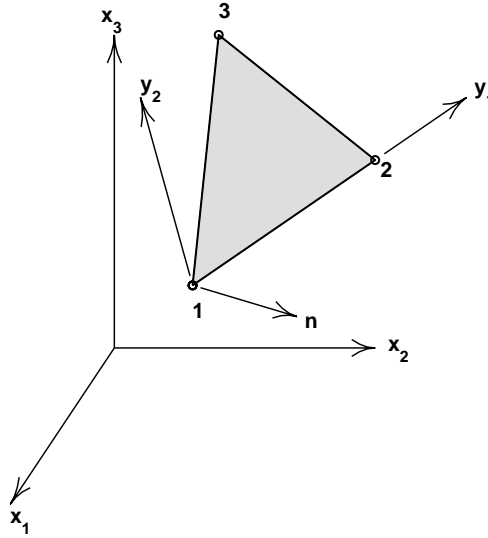


Figure 1: Description of coordinates for triangular element

2.1 Deformation gradient

From the above description of the motion of the triangle it is now possible to deduce the deformation gradient as

$$\mathbf{F} = \frac{\partial \mathbf{y}}{\partial \mathbf{Y}} = \mathbf{1} + \frac{\partial \mathbf{u}}{\partial \mathbf{Y}} \quad (12)$$

Using the parametric representations (11) we can express the deformation gradient as

$$\frac{\partial \mathbf{Y}}{\partial \xi} \mathbf{F} = \frac{\partial \mathbf{Y}}{\partial \xi} \frac{\partial \mathbf{y}}{\partial \mathbf{Y}} = \frac{\partial \mathbf{y}}{\partial \xi} \quad (13)$$

If we define the arrays \mathbf{J} and \mathbf{j} as

$$\mathbf{J} = \frac{\partial \mathbf{Y}}{\partial \xi} \quad \text{and} \quad \mathbf{j} = \frac{\partial \mathbf{y}}{\partial \xi} \quad (14)$$

then the deformation gradient is given by

$$\mathbf{F} = \mathbf{j} \mathbf{J}^{-1} \quad (15)$$

In the above \mathbf{J} is the Jacobian transformation for the reference frame and \mathbf{j} that for the current frame. Expanding the relations for each of the jacobians we obtain

$$\mathbf{J} = \begin{bmatrix} \left(\tilde{\mathbf{X}}^2 - \tilde{\mathbf{X}}^1 \right)^T \mathbf{V}_1 & \left(\tilde{\mathbf{X}}^3 - \tilde{\mathbf{X}}^1 \right)^T \mathbf{V}_1 \\ \left(\tilde{\mathbf{X}}^2 - \tilde{\mathbf{X}}^1 \right)^T \mathbf{V}_2 & \left(\tilde{\mathbf{X}}^3 - \tilde{\mathbf{X}}^1 \right)^T \mathbf{V}_2 \end{bmatrix} \quad (16)$$

and

$$\mathbf{j} = \begin{bmatrix} \left(\tilde{\mathbf{x}}^2 - \tilde{\mathbf{x}}^1 \right)^T \mathbf{v}_1 & \left(\tilde{\mathbf{x}}^3 - \tilde{\mathbf{x}}^1 \right)^T \mathbf{v}_1 \\ \left(\tilde{\mathbf{x}}^2 - \tilde{\mathbf{x}}^1 \right)^T \mathbf{v}_2 & \left(\tilde{\mathbf{x}}^3 - \tilde{\mathbf{x}}^1 \right)^T \mathbf{v}_2 \end{bmatrix} \quad (17)$$

By noting that $\tilde{\mathbf{X}}^2 - \tilde{\mathbf{X}}^1$ is orthogonal to \mathbf{V}_2 and similarly for the current configuration that $\tilde{\mathbf{x}}^2 - \tilde{\mathbf{x}}^1$ is orthogonal to \mathbf{v}_2 and in addition using the definition for \mathbf{V}_i and \mathbf{v}_i the above simplify to

$$\mathbf{J} = \begin{bmatrix} \|\tilde{\mathbf{X}}^2 - \tilde{\mathbf{X}}^1\|, & \left[\left(\tilde{\mathbf{X}}^2 - \tilde{\mathbf{X}}^1 \right)^T \left(\tilde{\mathbf{X}}^3 - \tilde{\mathbf{X}}^1 \right) \right] / \|\tilde{\mathbf{X}}^2 - \tilde{\mathbf{X}}^1\| \\ 0, & \|\mathbf{N}\| / \|\tilde{\mathbf{X}}^2 - \tilde{\mathbf{X}}^1\| \end{bmatrix} \quad (18)$$

and

$$\mathbf{j} = \begin{bmatrix} \|\tilde{\mathbf{x}}^2 - \tilde{\mathbf{x}}^1\|, & \left[\left(\tilde{\mathbf{x}}^2 - \tilde{\mathbf{x}}^1 \right)^T \left(\tilde{\mathbf{x}}^3 - \tilde{\mathbf{x}}^1 \right) \right] / \|\tilde{\mathbf{x}}^2 - \tilde{\mathbf{x}}^1\| \\ 0, & \|\mathbf{n}\| / \|\tilde{\mathbf{x}}^2 - \tilde{\mathbf{x}}^1\| \end{bmatrix} \quad (19)$$

Using these definitions, the right Cauchy-Green deformation tensor may be expanded as

$$\mathbf{C} = \mathbf{F}^T \mathbf{F} = \mathbf{J}^{-T} \mathbf{j}^T \mathbf{j} \mathbf{J}^{-1} = \mathbf{G}^T \mathbf{g} \mathbf{G} \quad (20)$$

where \mathbf{G} is used to denote the inverse of \mathbf{J} . In component form we have

$$\mathbf{C} = \frac{1}{J_{11}^2 J_{22}^2} \begin{bmatrix} J_{22} & -J_{12} \\ 0 & J_{11} \end{bmatrix} \begin{bmatrix} g_{11} & g_{12} \\ g_{12} & g_{22} \end{bmatrix} \begin{bmatrix} J_{22} & 0 \\ -J_{12} & J_{11} \end{bmatrix} \quad (21)$$

in which the terms in the kernel array involving \mathbf{j} may be expressed in the particularly simple form

$$\begin{aligned} g_{11} &= j_{11}^2 = \left(\tilde{\mathbf{x}}^2 - \tilde{\mathbf{x}}^1 \right)^T \left(\tilde{\mathbf{x}}^2 - \tilde{\mathbf{x}}^1 \right) \\ g_{12} &= j_{12} j_{11} = \left(\tilde{\mathbf{x}}^2 - \tilde{\mathbf{x}}^1 \right)^T \left(\tilde{\mathbf{x}}^3 - \tilde{\mathbf{x}}^1 \right) \\ g_{22} &= j_{12}^2 + j_{22}^2 = \left(\tilde{\mathbf{x}}^3 - \tilde{\mathbf{x}}^1 \right)^T \left(\tilde{\mathbf{x}}^3 - \tilde{\mathbf{x}}^1 \right) \end{aligned} \quad (22)$$

2.2 Material constitution - elastic behavior

In the present work we assume that a simple St.Venant-Kirchhoff material model may be used to express the stresses from the deformations. Stresses are thus given by

$$\mathbf{S} = \mathcal{D} \mathbf{E} \quad (23)$$

where the Green-Lagrange strains \mathbf{E} are given in terms of the deformation tensor as

$$\mathbf{E} = \frac{1}{2} (\mathbf{C} - \mathbf{I}) \quad (24)$$

In each triangular element the deformation may be computed from (19) to (22), thus giving directly the stress.

3 Weak Form for Equations of Motion

A weak form for the membrane may be written using a virtual work expression given by

$$\delta \Pi = \int_{\Omega} \delta x_i \rho_0 h \ddot{x}_i \, d\Omega + \int_{\Omega} \delta x_i c_0 \dot{x}_i \, d\Omega - \int_{\Omega} \frac{h}{2} \delta C_{IJ} S_{IJ} \, d\Omega - \int_{\Omega} \delta x_i b_i \, d\omega - \int_{\gamma_t} \delta x_i \bar{t}_i \, d\gamma \quad (25)$$

in which ρ_0 is mass density in the reference configuration, c_0 is a linear damping coefficient in the reference configuration, h is membrane thickness, S_{IJ} are components of the second Piola-Kirchhoff stress, b_i are components of loads in global coordinate directions (e.g., gravity), and \bar{t}_i are components of specified membrane force per unit length. Upper case letters refer to components expressed on the reference configuration, whereas, lower case letters refer to current configuration quantities. Likewise, Ω and ω are surface area for the reference and current configurations, respectively. Finally, γ_t is a part of the current surface contour on which traction values are specified.

The linear damping term is included only for purposes in getting initially stable solutions. That is, by ignoring the inertial loading based on $\ddot{\mathbf{x}}$ only first derivatives of time will occur. This results in a transient form which is critically damped - thus permitting the reaching of a static loading state in a more monotonic manner.

We note that components for a normal pressure loading may be expressed as

$$b_i = p n_i \quad (26)$$

where p is a specified pressure and n_i are components of the normal to the surface.

Writing Eq. (20) in component form we have

$$C_{IJ} = G_{iI} g_{ij} G_{jJ} \quad \text{for } i, j = 1, 2 \quad I, J = 1, 2 \quad (27)$$

where

$$G_{11} = \frac{1}{J_{11}} \quad ; \quad G_{22} = \frac{1}{J_{22}} \quad ; \quad G_{12} = \frac{J_{12}}{J_{11} J_{22}} \quad ; \quad G_{21} = 0 \quad (28)$$

The integrand of the first term in (25) may be written as

$$\delta C_{IJ} S_{IJ} = G_{iI} \delta g_{ij} G_{jJ} S_{IJ} = \delta g_{ij} s_{ij} \quad (29)$$

where the stress like variable s_{ij} is defined by

$$s_{ij} = G_{iI} G_{jJ} S_{IJ} \quad (30)$$

The transformation of stress given by (30) may be written in matrix form as

$$\mathbf{s} = \mathbf{Q}^T \mathbf{S} \quad (31)$$

in which

$$Q_{ab} \leftarrow G_{iI} G_{jJ}$$

where the index map is performed according to Table 1, yielding the result

$$\mathbf{Q} = \begin{bmatrix} G_{11}^2 & 0 & 0 \\ G_{12}^2 & G_{22}^2 & G_{12} G_{22} \\ G_{11} G_{12} & 0 & G_{11} G_{22} \end{bmatrix} \quad (32)$$

Table 1: Index map for \mathbf{Q} array

Indices	Values		
a	1	2	3
I,J	1,1	2,2	1,2 & 2,1
b	1	2	3
i,j	1,1	2,2	1,2 & 2,1

Since the deformation tensor is constant over each element, the results for the stresses are constant when h is taken constant over each element and, thus, the surface integral for the first term leads to the simple expression

$$\int_{\Omega} \frac{h}{2} \delta C_{IJ} S_{IJ} d\Omega = \int_{\Omega} \frac{h}{2} \delta g_{ij} s_{ij} d\Omega = \frac{h}{2} \delta g_{ij} s_{ij} A \quad (33)$$

where A is the reference area for the triangular element.

The variation of g_{ij} results in the values

$$\begin{aligned}\delta g_{11} &= 2 (\delta \tilde{\mathbf{x}}^2 - \delta \tilde{\mathbf{x}}^1)^T (\tilde{\mathbf{x}}^2 - \tilde{\mathbf{x}}^1) \\ \delta g_{12} &= (\delta \tilde{\mathbf{x}}^2 - \delta \tilde{\mathbf{x}}^1)^T (\tilde{\mathbf{x}}^3 - \tilde{\mathbf{x}}^1) + (\delta \tilde{\mathbf{x}}^3 - \delta \tilde{\mathbf{x}}^1)^T (\tilde{\mathbf{x}}^2 - \tilde{\mathbf{x}}^1) \\ \delta g_{22} &= 2 (\delta \tilde{\mathbf{x}}^3 - \delta \tilde{\mathbf{x}}^1)^T (\tilde{\mathbf{x}}^3 - \tilde{\mathbf{x}}^1)\end{aligned}\quad (34)$$

At this stage it is convenient to transform the second order tensors to matrix form and write

$$\frac{1}{2} \delta C_{IJ} S_{IJ} = \delta E_{IJ} S_{IJ} = \begin{bmatrix} \delta E_{11} & \delta E_{22} & 2 \delta E_{12} \end{bmatrix} \begin{bmatrix} S_{11} \\ S_{22} \\ S_{12} \end{bmatrix} = \delta \mathbf{E}^T \mathbf{S} \quad (35)$$

or for the alternative form

$$\delta g_{ij} s_{ij} = \begin{bmatrix} \delta g_{11} & \delta g_{22} & 2 \delta g_{12} \end{bmatrix} \begin{bmatrix} s_{11} \\ s_{22} \\ s_{12} \end{bmatrix} = \delta \mathbf{g}^T \mathbf{s} \quad (36)$$

Using (34) we obtain the result directly in terms of global cartesian components as

$$\begin{aligned}\frac{1}{2} \delta \mathbf{g}^T \mathbf{s} &= \begin{bmatrix} \delta(\tilde{\mathbf{x}}^1)^T & \delta(\tilde{\mathbf{x}}^2)^T & \delta(\tilde{\mathbf{x}}^3)^T \end{bmatrix} [\mathbf{b}]^T \mathbf{s} \\ &= \begin{bmatrix} \delta(\tilde{\mathbf{x}}^1)^T & \delta(\tilde{\mathbf{x}}^2)^T & \delta(\tilde{\mathbf{x}}^3)^T \end{bmatrix} [\mathbf{b}]^T \mathbf{Q}^T \mathbf{S} = \delta \mathbf{E}^T \mathbf{S}\end{aligned}\quad (37)$$

where $\Delta \tilde{\mathbf{x}}^{ij} = \tilde{\mathbf{x}}^i - \tilde{\mathbf{x}}^j$ and the *strain-displacement matrix* \mathbf{b} is given by

$$[\mathbf{b}] = \underbrace{\begin{bmatrix} -(\Delta \tilde{\mathbf{x}}^{21})^T & (\Delta \tilde{\mathbf{x}}^{21})^T & \mathbf{0} \\ -(\Delta \tilde{\mathbf{x}}^{31})^T & \mathbf{0} & (\Delta \tilde{\mathbf{x}}^{31})^T \\ -(\Delta \tilde{\mathbf{x}}^{21} + \Delta \tilde{\mathbf{x}}^{31})^T & (\Delta \tilde{\mathbf{x}}^{31})^T & (\Delta \tilde{\mathbf{x}}^{21})^T \end{bmatrix}}_{3 \times 9} \quad (38)$$

Thus, directly we have in each element

$$\delta \mathbf{E} = \mathbf{Q} \mathbf{b} \delta \tilde{\mathbf{x}} = \frac{1}{2} \delta \mathbf{C} \quad (39)$$

where $\tilde{\mathbf{x}}$ denotes the three nodal values on the element. It is immediately obvious that we can describe a strain-displacement matrix for the variation of \mathbf{E} as

$$\mathbf{B} = \mathbf{Q} \mathbf{b} \quad (40)$$

A residual form for each element may be written as

$$\begin{Bmatrix} \mathbf{R}^1 \\ \mathbf{R}^2 \\ \mathbf{R}^3 \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}^1 \\ \mathbf{f}^2 \\ \mathbf{f}^3 \end{Bmatrix} - [\mathbf{M}_e] \begin{Bmatrix} \ddot{\tilde{\mathbf{x}}}^1 \\ \ddot{\tilde{\mathbf{x}}}^2 \\ \ddot{\tilde{\mathbf{x}}}^3 \end{Bmatrix} - [\mathbf{C}_e] \begin{Bmatrix} \dot{\tilde{\mathbf{x}}}^1 \\ \dot{\tilde{\mathbf{x}}}^2 \\ \dot{\tilde{\mathbf{x}}}^3 \end{Bmatrix} - h A [\mathbf{B}]^T \begin{Bmatrix} S_{11} \\ S_{22} \\ S_{12} \end{Bmatrix} \quad (41)$$

where $[\mathbf{M}_e]$ and where $[\mathbf{C}_e]$ are the element mass and damping matrices given by

$$[\mathbf{M}_e] = \begin{bmatrix} \mathbf{M}^{11} & \mathbf{M}^{12} & \mathbf{M}^{13} \\ \mathbf{M}^{21} & \mathbf{M}^{22} & \mathbf{M}^{23} \\ \mathbf{M}^{31} & \mathbf{M}^{32} & \mathbf{M}^{33} \end{bmatrix} \quad \text{and} \quad [\mathbf{C}_e] = \begin{bmatrix} \mathbf{C}^{11} & \mathbf{C}^{12} & \mathbf{C}^{13} \\ \mathbf{C}^{21} & \mathbf{C}^{22} & \mathbf{C}^{23} \\ \mathbf{C}^{31} & \mathbf{C}^{32} & \mathbf{C}^{33} \end{bmatrix} \quad (42)$$

with

$$\mathbf{M}^{\alpha\beta} = \int_{\Omega} \rho_0 h \xi_{\alpha} \xi_{\beta} d\Omega \mathbf{I} \quad \text{and} \quad \mathbf{C}^{\alpha\beta} = \int_{\Omega} c_0 h \xi_{\alpha} \xi_{\beta} d\Omega \mathbf{I} \quad (43)$$

3.1 Pressure follower loading

For membranes subjected to internal pressure loading, the finite element nodal forces must be computed based on the *deformed* current configuration. Thus, for each triangle we need to compute the nodal forces from the relation

$$\delta \tilde{\mathbf{x}}^{\alpha,T} \mathbf{f}^{\alpha} = \delta \tilde{\mathbf{x}}^{\alpha,T} \int_{\omega} \xi_{\alpha} (p \mathbf{n}) d\omega \quad (44)$$

For the constant triangular element and constant pressure over the element, denoted by p_e , the normal vector \mathbf{n} is also constant and thus the integral yields the nodal forces

$$\mathbf{f}^{\alpha} = \frac{1}{3} p_e \mathbf{n} A_e \quad (45)$$

We noted previously from Eq. (6) that the cross product of the incremental vectors $\Delta \tilde{\mathbf{x}}^{21}$ with $\Delta \tilde{\mathbf{x}}^{31}$ resulted in a vector normal to the triangle with magnitude of twice the area. Thus, the nodal forces for the pressure are given by the simple relation

$$\mathbf{f}^{\alpha} = \frac{1}{6} p_e \Delta \tilde{\mathbf{x}}^{21} \times \Delta \tilde{\mathbf{x}}^{31} \quad (46)$$

Instead of the cross products it is convenient to introduce a matrix form denoted by

$$\Delta \tilde{\mathbf{x}}^{21} \times \Delta \tilde{\mathbf{x}}^{31} = \left[\widehat{\Delta \tilde{\mathbf{x}}^{21}} \right] \Delta \tilde{\mathbf{x}}^{31} \quad (47)$$

where

$$\left[\widehat{\Delta \tilde{\mathbf{x}}^{21}} \right] = \begin{bmatrix} 0 & -\Delta \tilde{x}_3^{21} & \Delta \tilde{x}_2^{21} \\ \Delta \tilde{x}_3^{21} & 0 & -\Delta \tilde{x}_1^{21} \\ -\Delta \tilde{x}_2^{21} & \Delta \tilde{x}_1^{21} & 0 \end{bmatrix} \quad (48)$$

4 Reinforcements

It is common to place reinforcing cables in membranes to provide added strength or shape control. The cables are generally very strong in axial load capacity (generally tension) and weak in bending. Accordingly, they may be modeled by a *truss* type member. In the form admitted here it is not necessary for the reinforcement to be placed at the edges of membrane elements - they may pass through an element as shown in Fig 2.

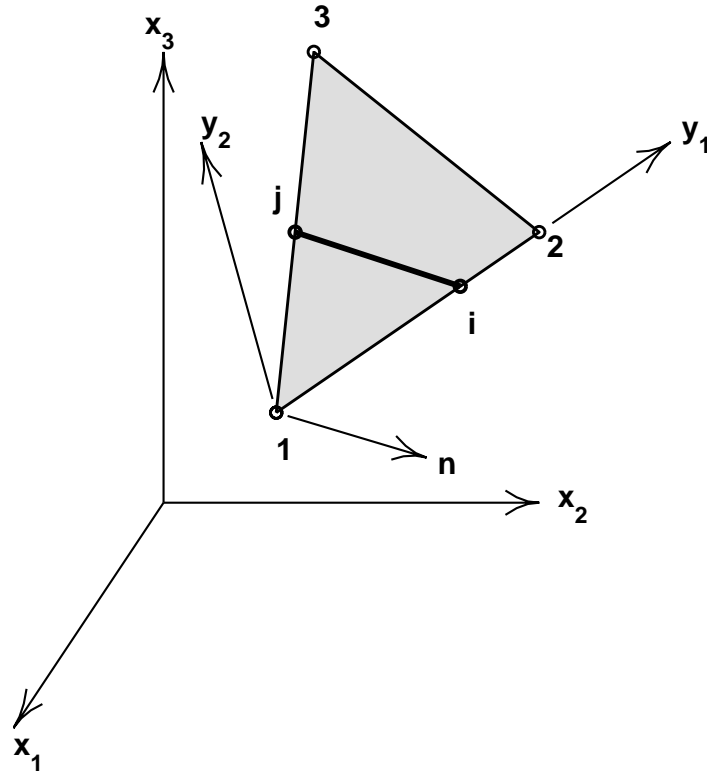


Figure 2: Cable reinforced membrane element at(i-j)

The ends of a typical reinforcement are denoted as i and j in the figure and have reference coordinates \mathbf{X}^i and \mathbf{X}^j , respectively. These points may be referred to the nodal values of the membrane by computing the values of the natural coordinates so that

$$\mathbf{X}^k = \xi_\alpha^k \mathbf{X}^\alpha \quad \text{for } k = i, j \quad (49)$$

The solution for the two points may be trivially constructed from from linear interpolation on the edges. The results are (for the points intersecting the

edges shown in the figure)

$$\begin{aligned}\xi_2^i &= \frac{\|\mathbf{X}^i - \mathbf{X}^1\|}{\|\mathbf{X}^2 - \mathbf{X}^1\|} ; & \xi_1^i &= 1 - \xi_2^i ; & \xi_3^i &= 0 \\ \xi_3^j &= \frac{\|\mathbf{X}^j - \mathbf{X}^1\|}{\|\mathbf{X}^3 - \mathbf{X}^1\|} ; & \xi_1^j &= 1 - \xi_3^j ; & \xi_2^j &= 0\end{aligned}\quad (50)$$

Using these values the deformed position of the reinforcement cable may be written as

$$\begin{Bmatrix} \mathbf{x}^i \\ \mathbf{x}^j \end{Bmatrix} = \begin{Bmatrix} \xi_\alpha^i \mathbf{I} \\ \xi_\alpha^j \mathbf{I} \end{Bmatrix} \tilde{\mathbf{x}}^\alpha \quad (51)$$

4.1 Deformation of cable

The deformation of the reinforcement cable may be expressed in terms of the Green-Lagrange strain given by

$$E_{ij} = \frac{1}{2} \left[\frac{(\mathbf{x}^j - \mathbf{x}^i)^T (\mathbf{x}^j - \mathbf{x}^i)}{(\mathbf{X}^j - \mathbf{X}^i)^T (\mathbf{X}^j - \mathbf{X}^i)} - 1 \right] = \frac{1}{2} \left[\frac{\|\Delta \mathbf{x}^{ji}\|^2}{\|\Delta \mathbf{X}^{ji}\|^2} - 1 \right] \quad (52)$$

where $\Delta \mathbf{X}^{ji} = \mathbf{X}^j - \mathbf{X}^i$. The variation of the strain is then expressed as

$$\delta E_{ij} = \frac{(\delta \mathbf{x}^j - \delta \mathbf{x}^i)^T (\Delta \mathbf{x}^{ji})}{\|\Delta \mathbf{X}^{ji}\|^2} \quad (53)$$

4.2 Material constitution

For simplicity we again assume that the material is elastic and may be represented by a one-dimensional form of a St.Venant-Kirchhoff model expressed as

$$S_{ij} = \mathbf{E} E_{ij} \quad (54)$$

where S_{ij} is the constant second Piola-Kirchhoff stress in the cable and \mathbf{E} is an elastic modulus.

4.3 Weak form for reinforcement

A weak form for an individual reinforcement cable in an element may be written as

$$\delta \Pi_r = \delta \mathbf{x}^k \left(\mathbf{M}_{kl} \ddot{\mathbf{x}}^l + \mathbf{C}_{kl} \dot{\mathbf{x}}^l \right) - \delta E_{ij} S_{ij} A_{ij} L_{ij} \quad ; \quad k, l = i, j \quad (55)$$

where A_{ij} is the cross sectional area of the reinforcement; L_{ij} the length of the cable (i.e., $\|\Delta \mathbf{X}^{ji}\|$); \mathbf{M}_{kl} is the mass matrix; and \mathbf{C}_{kl} is the damping matrix.

The variation of the strain is rewritten from Eq. (53) as

$$\delta E_{ij} = \begin{bmatrix} \delta \mathbf{x}^{i,T} & \delta \mathbf{x}^{j,T} \end{bmatrix} \left\{ \begin{array}{c} -\frac{\Delta \mathbf{x}^{ji}}{L_{ij}^2} \\ \frac{\Delta \mathbf{x}^{ji}}{L_{ij}^2} \end{array} \right\} \quad (56)$$

Equation (55) is appended to the other terms from the membrane by replacing variations of end displacements and the rate terms by their representation in terms of the membrane nodal parameters as given by Eq. (51). The result is:

$$\delta \Pi_r = \delta \tilde{\mathbf{x}}^{\alpha,T} \left\{ \mathbf{M}_r^{\alpha\beta} \ddot{\tilde{\mathbf{x}}}^\beta + \mathbf{C}_r^{\alpha\beta} \dot{\tilde{\mathbf{x}}}^\beta - \mathbf{P}_r^\alpha \right\} \quad (57)$$

where

$$\mathbf{P}_r^\alpha = \Delta \xi_\alpha^{ji} \Delta \xi_\beta^{ji} \tilde{\mathbf{x}}^\beta \frac{S_{ij} A_{ij}}{L_{ij}} ; \quad (58)$$

$$\mathbf{M}_r^{\alpha\beta} = \xi_\alpha^i \mathbf{M}_{ii} \xi_\beta^i + \xi_\alpha^i \mathbf{M}_{ij} \xi_\beta^j + \xi_\alpha^j \mathbf{M}_{ji} \xi_\beta^i + \xi_\alpha^j \mathbf{M}_{jj} \xi_\beta^j \quad (59)$$

and

$$\mathbf{C}_r^{\alpha\beta} = \xi_\alpha^i \mathbf{C}_{ii} \xi_\beta^i + \xi_\alpha^i \mathbf{C}_{ij} \xi_\beta^j + \xi_\alpha^j \mathbf{C}_{ji} \xi_\beta^i + \xi_\alpha^j \mathbf{C}_{jj} \xi_\beta^j \quad (60)$$

In the definition of \mathbf{P}_r^α the incremental ξ denote

$$\Delta \xi_\alpha^{ji} = \xi_\alpha^j - \xi_\alpha^i \quad (61)$$

Multiple reinforcement strands within any element may be simply considered by summing over all the ij -pairs of intersection points.

5 Solution Methods

5.1 Explicit solution

In an explicit time integration the lumped mass is commonly used. Furthermore, we shall assume that the damping is negligible and thus may be ignored. A diagonal (lumped) mass is usually also constructed where

$$\mathbf{M}^{\alpha\beta} = \begin{cases} \frac{1}{3} \int_\Omega \rho_0 h \, d\Omega \mathbf{I} & ; \text{ for } \alpha = \beta \\ \mathbf{0} & ; \text{ for } \alpha \neq \beta \end{cases} \quad (62)$$

Diagonalization of the mass matrix in the presence of reinforcement is more difficult and, if performed, must be based on purely physical lumping arguments as no clear mathematical justification is available.^[17]

A solution is then computed for each discrete time t_n from

$$\ddot{\mathbf{x}}_n = \mathbf{M}^{-1} \left[\mathbf{f}_n - \sum_e (\mathbf{B}_{n,e}^T \mathbf{S}_{n,e} A_e h_e + \mathbf{P}_{r,e}) \right] \quad (63)$$

where subscripts e refer to individual element quantities and subscript n to the time step. The inverse of the mass is trivial due to its diagonal form, hence the method is directly proportional to the number of nodes in the mesh.

The solution state may now be advanced in time using any time integration procedure. For example using a Newmark method

$$\begin{aligned} \dot{\mathbf{x}}_n &= \dot{\mathbf{x}}_{n-1} + \frac{1}{2} \Delta t_{n-1} (\ddot{\mathbf{x}}_{n-1} + \ddot{\mathbf{x}}_n) \\ \mathbf{x}_{n+1} &= \mathbf{x}_n + \Delta t_n \dot{\mathbf{x}}_n + \frac{1}{2} \Delta t_n^2 \ddot{\mathbf{x}}_n \end{aligned} \quad (64)$$

in which $\Delta t_n = t_n - t_{n-1}$.

An explicit solution is conditionally stable and requires

$$\Delta t_n \leq \Delta t_{cr} \quad (65)$$

for all time steps. The critical time step depends on element size and the maximum wave speed for the element material. The resulting time increment is often much too small for practical considerations in computer effort and for the response necessary to model slowly varying loads. In these situations it is more expedient to use an implicit time integration procedure in which inertial, stress, and loading matrices also depend on position and velocity at the current time.

5.2 Implicit solution

In an implicit solution case it is necessary to use an iterative solution scheme at each time step which solves a sequence of linear, coupled algebraic problems. Here we only present results for the St. Venant-Kirchhoff material model and the normal *follower* pressure loading. We assume that the transient problem will be integrated using the Newmark method, however, other schemes may also be utilized with minor modifications.

The Newmark method may be written for implicit solutions as

$$\begin{aligned} \mathbf{x}_n &= \mathbf{x}_{n-1} + \Delta t_n \dot{\mathbf{x}}_{n-1} + \left(\frac{1}{2} - \beta \right) \Delta t_n^2 \ddot{\mathbf{x}}_{n-1} + \beta \Delta t_n^2 \ddot{\mathbf{x}}_n \\ \dot{\mathbf{x}}_n &= \dot{\mathbf{x}}_{n-1} + (1 - \gamma) \Delta t_n \ddot{\mathbf{x}}_{n-1} + \gamma \Delta t_n \ddot{\mathbf{x}}_n \end{aligned} \quad (66)$$

The equations to be solved at each time step may be expressed as

$$\begin{aligned} \mathbf{R}_n^\alpha &= \mathbf{f}_n^\alpha - \sum_e (\mathbf{M}_e^{\alpha\beta} + \mathbf{M}_r^{\alpha\beta}) \ddot{\mathbf{x}}_n^\beta - \sum_e (\mathbf{C}_e^{\alpha\beta} + \mathbf{C}_r^{\alpha\beta}) \dot{\mathbf{x}}_n^\beta \\ &\quad - \sum_e (h_e A_e \mathbf{B}_e^{\alpha,T} \mathbf{S}_e + \mathbf{P}_{r,e}^\alpha)_n \end{aligned} \quad (67)$$

In an implicit method Eq. (67) may be solved iteratively using a Newton method. In this process the nonlinear residual equations are linearized about a given set of nodal positions $\tilde{\mathbf{x}}_n^k$ corresponding to known values at some iteration stage k . The result is written as

$$\mathbf{R}_n^{k+1} \approx \mathbf{R}_n^k + \left. \frac{\partial \mathbf{R}_n}{\partial \tilde{\mathbf{x}}} \right|^k d\tilde{\mathbf{x}}_n^k = \mathbf{0} \quad (68)$$

If we define the tangent (jacobian) matrix \mathbf{A} as

$$\mathbf{A} = - \frac{\partial \mathbf{R}}{\partial \tilde{\mathbf{x}}} \quad (69)$$

the set of linear algebraic equations to be solved at each iteration may be expressed as

$$\mathbf{A}_n^k d\tilde{\mathbf{x}}_n^k = \mathbf{R}_n^k \quad (70)$$

The solution may then be updated using

$$\mathbf{x}_n^{k+1} = \mathbf{x}_n^k + d\mathbf{x}_n^k \quad (71)$$

and iteration continued until convergence is achieved.

The Newton scheme will have a quadratic asymptotic rate of convergence provided a careful derivation of the tangent matrix \mathbf{A} is constructed. Typically such jacobians are referred to as the *consistent tangent matrix*. For transient applications the use of the specific time stepping algorithm is required to compute the tangent matrix. The computation for the transient term is performed as

$$\mathbf{A} = - \frac{\partial \mathbf{R}}{\partial \tilde{\mathbf{x}}} - \frac{\partial \mathbf{R}}{\partial \dot{\tilde{\mathbf{x}}}} \frac{\partial \dot{\tilde{\mathbf{x}}}}{\partial \tilde{\mathbf{x}}} - \frac{\partial \mathbf{R}}{\partial \ddot{\tilde{\mathbf{x}}}} \frac{\partial \ddot{\tilde{\mathbf{x}}}}{\partial \tilde{\mathbf{x}}} \quad (72)$$

The result may be written as

$$\mathbf{A} = c_1 \mathbf{K} + c_2 \mathbf{C} + c_3 \mathbf{M} \quad (73)$$

where the c_i result from any differentiation of the nodal vectors with respect to the solution vector. For the Newmark method the result from (67) gives $c_1 = 1$ and from (66) we obtain

$$\frac{\partial \tilde{\mathbf{x}}}{\partial \ddot{\tilde{\mathbf{x}}}_n} = \beta \Delta t_n^2 \mathbf{I} \quad \text{and} \quad \frac{\partial \dot{\tilde{\mathbf{x}}}}{\partial \ddot{\tilde{\mathbf{x}}}_n} = \gamma \Delta t_n \mathbf{I}$$

thus giving

$$c_2 = \frac{\gamma}{\beta \Delta t_n} \quad \text{and} \quad c_3 = \frac{1}{\beta \Delta t_n^2}$$

From (67) we find that

$$\begin{aligned} \mathbf{M}_n &= \sum_e (\mathbf{M}_e + \mathbf{M}_r) \\ \mathbf{C}_n &= \sum_e (\mathbf{C}_e + \mathbf{C}_r) \\ \mathbf{K}_n &= \sum_e (\mathbf{K}_e + \mathbf{K}_r) \end{aligned} \quad (74)$$

5.2.1 Membrane tangent matrix

To compute the element stiffness matrix it is necessary to determine the change in stress due to an incremental change in the motion. Accordingly, for the St.Venant-Kirchhoff model we obtain

$$d\mathbf{S}_e = \mathbf{D} d\mathbf{E}_e \quad (75)$$

where

$$d\mathbf{E}_e = \mathbf{Q}_e \mathbf{b}_e d\tilde{\mathbf{x}}_e = \mathbf{B}_e d\tilde{\mathbf{x}}_e \quad (76)$$

The element stiffness matrix is given by

$$\mathbf{K}_e = (h A \mathbf{B}_n^T \mathbf{D}_n \mathbf{B}_n + \mathbf{K}_g)_e \quad (77)$$

where \mathbf{K}_g is a geometric stiffness computed from the term

$$\frac{h_e}{2} A_e d(\delta C_{IJ}) S_{IJ} = \frac{h_e}{2} A_e d(\delta g_{ij}) s_{ij} \quad (78)$$

From (34) we obtain

$$\begin{aligned} d(\delta g_{11}) &= 2 (\delta \tilde{\mathbf{x}}^2 - \delta \tilde{\mathbf{x}}^1)^T (d\tilde{\mathbf{x}}^2 - d\tilde{\mathbf{x}}^1) \\ d(\delta g_{12}) &= (\delta \tilde{\mathbf{x}}^2 - \delta \tilde{\mathbf{x}}^1)^T (d\tilde{\mathbf{x}}^3 - d\tilde{\mathbf{x}}^1) + (\delta \tilde{\mathbf{x}}^3 - \delta \tilde{\mathbf{x}}^1)^T (d\tilde{\mathbf{x}}^2 - d\tilde{\mathbf{x}}^1) \\ d(\delta g_{22}) &= 2 (\delta \tilde{\mathbf{x}}^3 - \delta \tilde{\mathbf{x}}^1)^T (d\tilde{\mathbf{x}}^3 - d\tilde{\mathbf{x}}^1) \end{aligned} \quad (79)$$

Using these expressions the geometric matrix may be written as

$$\mathbf{K}_g = h_e A_e \begin{bmatrix} (s_{11} + 2s_{12} + s_{22})\mathbf{I} & -(s_{11} + s_{12})\mathbf{I} & -(s_{22} + s_{12})\mathbf{I} \\ -(s_{11} + s_{12})\mathbf{I} & s_{11}\mathbf{I} & s_{12}\mathbf{I} \\ -(s_{22} + s_{12})\mathbf{I} & s_{12}\mathbf{I} & s_{22}\mathbf{I} \end{bmatrix} \quad (80)$$

5.2.2 Reinforcement tangent matrix

In a similar manner to that for the membrane, the tangent matrix for the reinforcement may be computed from

$$\begin{aligned}\mathbf{K}_r^{\alpha\beta} &= \frac{\partial \mathbf{P}_r^\alpha}{\partial \tilde{\mathbf{x}}^\beta} \\ &= \Delta \xi_\alpha^{ji} \left[\frac{\mathbf{E}A_{ij}}{L_{ij}} (\Delta \xi_\gamma^{ji} \tilde{\mathbf{x}}^\gamma \tilde{\mathbf{x}}^{\delta,T} \Delta \xi_\delta^{ji}) + \frac{S_{ij}A_{ij}}{L_{ij}} \mathbf{I} \right] \Delta \xi_\beta^{ji}\end{aligned}\quad (81)$$

5.3 Quasi-static Solutions

Membrane structures typically have no stiffness during the initial phase of loading. Thus, it is necessary to perform some form of a transient analysis until an equilibrium position is neared, at which time it is possible to switch to a *static* loading in which no rate terms are included. To avoid oscillations during the solution process the equations of motion are treated here in a first order form as

$$\mathbf{R} = \mathbf{f} - \mathbf{C} \dot{\tilde{\mathbf{x}}} - \sum_e (\mathbf{h}_e \mathbf{A}_e \mathbf{B}^T \mathbf{S} + \mathbf{P}_{e,r}) = \mathbf{0} \quad (82)$$

and solved using an implicit *backward Euler* solution process in which discrete rates are approximated by

$$\dot{\tilde{\mathbf{x}}}_n = \frac{1}{\Delta t_n} (\tilde{\mathbf{x}}_n - \tilde{\mathbf{x}}_{n-1}) \quad (83)$$

A solution is computed until the rate terms are small at which time they are dropped and the solution is computed from the static form

$$\mathbf{R} = \mathbf{f} - \sum_e (\mathbf{h}_e \mathbf{A}_e \mathbf{B}^T \mathbf{S} + \mathbf{P}_{e,r}) = \mathbf{0} \quad (84)$$

In results reported in numerical examples, a diagonal (lumped) form of \mathbf{C} is used. The diagonalization is performed identically to that for the lumped mass form.

6 Numerical Examples

6.1 Sphere subjected to internal follower pressure

The first problem presented is a sphere of initial (unstressed) radius of 10 units which is subjected to an internal follower pressure loading of 5. The

material properties are

$$E = 1000 \ ; \ \nu = 0.25 \ ; \ \rho_0 = 10 \ ; \ h_e = 0.1$$

A mesh for one quadrant of the sphere (1/8 of the total) is constructed as shown in Fig. 3 Both the undeformed and deformed configurations are

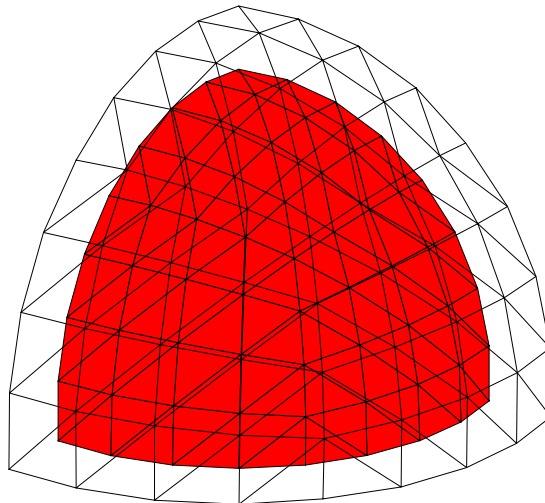


Figure 3: Mesh for sphere problem symmetrically supported at edges

included to indicate the amount of deformation occurring.

The problem is solved using a backward Euler time integrator for the case where \mathbf{M} is zero and a diagonal damping matrix \mathbf{C} with $c_0 = \rho_0$ is used. The full internal pressure is applied during the first time step and held constant during subsequent steps. The problem is solved using 100 steps with a constant Δt of 0.001. Subsequently, the rate terms are ignored and a final static state determined during one additional step. Convergence to full machine precision is achieved in three iterations for all steps - indicating a correct implementation for the Newton strategy described in this study. A plot of the contours for the u_3 displacement is shown in Fig 4.

6.2 Corner supported sphere quadrant subjected to internal follower pressure

The boundary conditions for the first problem are changed to ones in which the vertices of the quadrant are fully restrained (fixed in all directions). Loading is again by a follower loading with magnitude 5. The contours for the

vertical displacement (u_3) are shown on the deformed mesh in Fig. 5. Solution is obtained in the same way as for the previous example. Convergence was achieved in all steps to full machine accuracy in 3-4 iterations each step.

6.3 Square supported at 4-corners

The third problem is presented to test the performance of the triangular membrane formulation described above under gravity loading. Here a square region of 20 units on a side is supported only at its four corners. Due to symmetry only one quadrant is analyzed with the origin of coordinates at the lower left corner and a uniform mesh of 20×20 spaces as shown in Fig 6. The flat configuration lies in the $x_1 - x_2$ coordinate plane at $x_3 = 0$. A uniform body loading is applied in the x_3 direction. To permit a sag to occur the upper right corner node (which is fully restrained in all three directions) is displaced equally in the x_1 and x_2 by a negative 2 units (that is the deformed position of the node at this point has final coordinates $(8, 8, 0)$).

For the membrane problem considered here, the properties are taken as

$$E = 10000 \ ; \ \nu = 0.25 \ ; \ \rho_0 = 10 \ ; \ h_e = 0.1 \ ; \ b_3 = 1$$

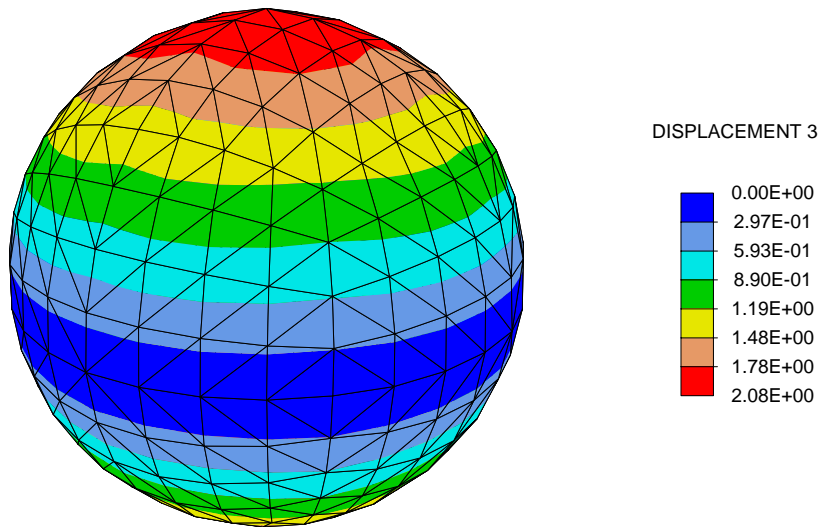


Figure 4: Contours for u_3 . Full sphere shown by reflections.

The solution is computed using a backward Euler solution process with $\mathbf{M} = \mathbf{0}$ and a diagonal \mathbf{C} matrix with $c_0 = \rho_0$. The time behavior is constructed using 1000 time steps of $\Delta t = 0.0001$ followed by an additional 1000 steps at $\Delta t = 0.001$ and a final 1000 steps at $\Delta t = 0.002$. The corner displacements are moved 2 units at a uniform rate until $t = 1$ after which they are held constant. At the final time of 3.1 the solution is switch to the static state and a converged solution achieved in 4 iterations. In general each time step converges at the quadratic rate in 3 to 4 iterations per step. Use of larger step sizes, however, resulted in divergence of the solution after a few steps. The final shape of the membrane is shown in Fig. 7 where the full problem is shown by reflecting results about the symmetry axes.

7 Closure

This report has summarized the steps required to develop and implant a 3-node triangular membrane finite element which can undergo arbitrarily large finite motions. The material behavior included here is restricted to a St.Venant-Kirchhoff material, however, extension to other types of constitu-

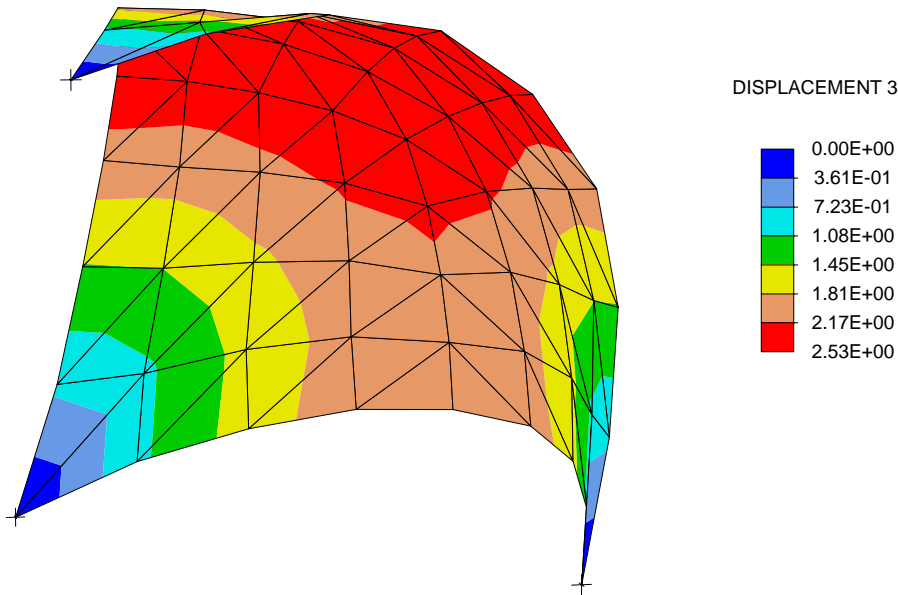


Figure 5: Contours for u_3 . Corner supported sphere shown in deformed configuration

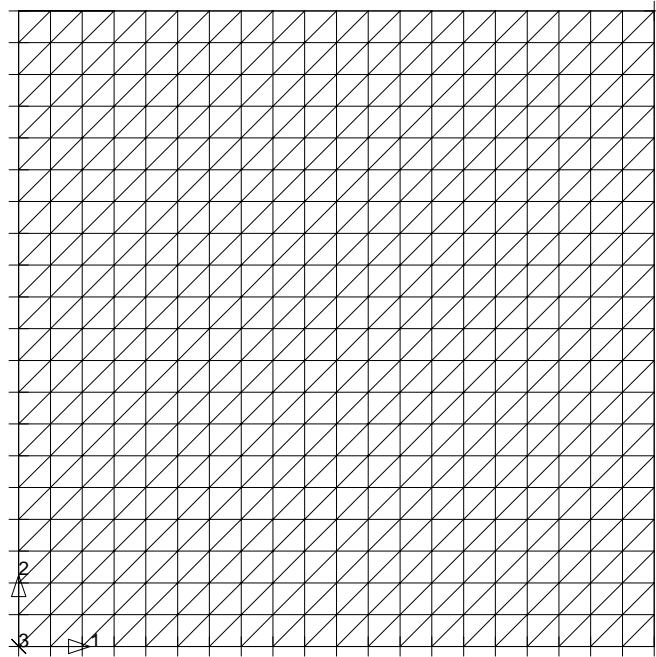


Figure 6: Mesh for membrane problem supported at corners

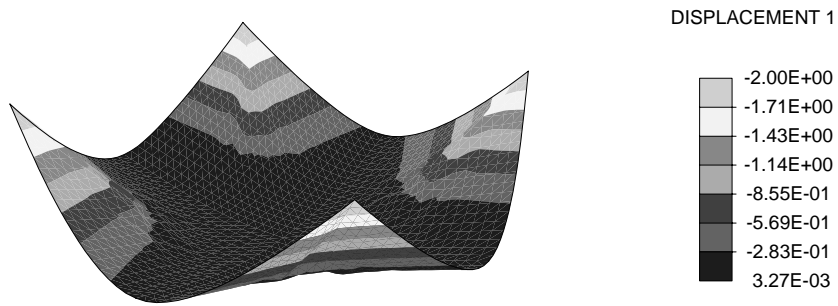


Figure 7: Deformed configuration of membrane

tive models for isotropic behavior is straight forward.

The formulation presented includes both inertia and damping effects leading to a general second-order semi-discrete form. Solution to these ordinary differential equations may be achieved using explicit or implicit transient forms or using a quasi-static form in which all rate effects are omitted. Use of a solution algorithm employing only first-order form is shown to lead to a means of solving the final static membrane state starting from the unstressed reference configuration.

A listing for the subroutine necessary to link the formulation to the general finite element solution system *FEAP*^[18] is included in an appendix.

References

- [1] O.C. Zienkiewicz and R.L. Taylor. *The Finite Element Method: Solid Mechanics*, volume 2. Butterworth-Heinemann, Oxford, 5th edition, 2000.
- [2] J.C. Simo and D.D. Fox. On a stress resultant geometrically exact shell model. Part I: Formulation and optimal parametrization. *Computer Methods in Applied Mechanics and Engineering*, 72:267–304, 1989.
- [3] J.C. Simo, D.D. Fox, and M.S. Rifai. On a stress resultant geometrically exact shell model. Part II: The linear theory; computational aspects. *Computer Methods in Applied Mechanics and Engineering*, 73:53–92, 1989.
- [4] J.C. Simo, M.S. Rifai, and D.D. Fox. On a stress resultant geometrically exact shell model. Part IV Variable thickness shells with through-the-thickness stretching. *Computer Methods in Applied Mechanics and Engineering*, 81:91–126, 1990.
- [5] J.C. Simo and N. Tarnow. A new energy and momentum conserving algorithm for the non-linear dynamics of shells. *International Journal for Numerical Methods in Engineering*, 37:2527–2549, 1994.
- [6] N. Büchter and E. Ramm. Shell theory versus degeneration – a comparison in large rotation finite element analysis. *International Journal for Numerical Methods in Engineering*, 34:39–59, 1992.
- [7] N. Büchter, E. Ramm, and D. Roehl. Three-dimensional extension of non-linear shell formulations based on the enhanced assumed strain concept. *International Journal for Numerical Methods in Engineering*, 37:2551–2568, 1994.

- [8] M. Braun, M. Bischoff, and E. Ramm. Nonlinear shell formulations for complete three-dimensional constitutive laws include composites and laminates. *Computational Mechanics*, 15:1–18, 1994.
- [9] M. Bischoff and E. Ramm. Shear deformable shell elements for large strains and rotations. *International Journal for Numerical Methods in Engineering*, 40:4427–49, 1997.
- [10] M. Bischoff and E. Ramm. Solid-like shell or shell-like solid formulation? A personal view. In W. Wunderlich, editor, *Proc. Eur Conf on Comp. Mech (ECCM'99 on CD-ROM)*, Munich, September 1999.
- [11] E. Ramm. From Reissner plate theory to three dimensions in large deformation shell analysis. *Z. Angew. Math. Mech.*, 79:1–8, 1999.
- [12] M. Bischoff and E. Ramm. On the physical significance of higher order kinematic and static variables in a three-dimensional shell formulation. *International Journal of Solids and Structures*, 37:6933–60, 2000.
- [13] K.-U. Bletzinger, M. Bischoff, and E. Ramm. A unified approach for shear-locking-free triangular and rectangular shell finite elements. *Computers and Structures*, 75:321–34, 2000.
- [14] K. Schweizerhof and E. Ramm. Displacement dependent pressure loads in non-linear finite element analysis. *Computers and Structures*, 18(6):1099–1114, 1984.
- [15] J.C. Simo, R.L. Taylor, and P. Wriggers. A note on finite element implementation of pressure boundary loading. *Communications in Applied Numerical Methods*, 7:513–525, 1991.
- [16] N. Newmark. A method of computation for structural dynamics. *Journal of the Engineering Mechanics Division*, 85:67–94, 1959.
- [17] O.C. Zienkiewicz and R.L. Taylor. *The Finite Element Method: The Basis*, volume 1. Butterworth-Heinemann, Oxford, 5th edition, 2000.
- [18] R.L. Taylor. *FEAP - A Finite Element Analysis Program, Programmer Manual*. University of California, Berkeley. <http://www.ce.berkeley.edu/~rlt>.

A Listing for Membrane Element

```
subroutine elmt01(d,ul,xl,ix,tl,s,r,ndf,ndm,nst,isw)
c      * * F E A P * * A Finite Element Analysis Program
c.... Copyright (c) 1984-2001: Robert L. Taylor
c      Triangular finite displacement membrane element

implicit none

include 'bdata.h'
include 'cdata.h'
include 'eldata.h'
include 'elplot.h'
include 'eltran.h'
include 'fdata.h'
include 'iofile.h'
include 'prstrs.h'
include 'strnum.h'

integer ndf,ndm,nst,isw, i,j

real*8 pr,ro,he,dv,xx,yy,lfac,cfac,mass,rnormn

integer ix(*),is(9)
real*8 d(*),ul(ndf,nen,*),xl(ndm,*),tl(*),s(nst,nst),r(ndf,*)
real*8 shp(3,9),sig(9),eps(6),dd(3,3),ss(3)
real*8 q(3,3),bb(3,9),bmat(3,9),bdi(3)
real*8 xc(3,3),rg(2,2),gg(2,2),rb(3),cn(3),rn(3)
real*8 cdx21(3),cdx31(3),rdx21(3),rdx31(3)

save

c      Output element type

if(isw.eq.0 .and. ior.lt.0) then
  write(*,*) ' Elmt 1: Finite displacement membrane'

c      Input material properties

elseif(isw.eq.1) then

  call inpt01(d)

c      Deactivate dof in element for dof > ndm

  do i = ndm+1,ndf
    ix(i) = 0
  end do ! i

c      Set plot for 3-node triangles

  call pltri3(iel)

c      Set number of projected stresses

  istv = 3

c      Set assembly vector
```

```

do i = 1,3
  is(i ) = i
  is(i+3) = i + ndf
  is(i+6) = i + ndf*2
end do ! i

c Check element for errors in input data

elseif(isw.eq.2) then

  call cktris(ix,xl,shp,ndm)

c Compute stress-divergence vector (r) and stiffness matrix (s)

elseif(isw.eq.3 .or. isw.eq.4 .or. isw.eq.5 .or. isw.eq.6) then

c Set current configuration coordinates

do i = 1,3
  do j = 1,ndm
    xc(j,i) = xl(j,i) + ul(j,i,1)
  end do ! j
end do ! i

c Compute incremental vectors: 'r' = reference; 'c' = current

do i = 1,ndm
  rdx21(i) = xl(i,2) - xl(i,1)
  rdx31(i) = xl(i,3) - xl(i,1)
  cdx21(i) = xc(i,2) - xc(i,1)
  cdx31(i) = xc(i,3) - xc(i,1)
end do ! i

c Normal vector times area

rn(1) = rdx21(2)*rdx31(3) - rdx21(3)*rdx31(2)
rn(2) = rdx21(3)*rdx31(1) - rdx21(1)*rdx31(3)
rn(3) = rdx21(1)*rdx31(2) - rdx21(2)*rdx31(1)

cn(1) = cdx21(2)*cdx31(3) - cdx21(3)*cdx31(2)
cn(2) = cdx21(3)*cdx31(1) - cdx21(1)*cdx31(3)
cn(3) = cdx21(1)*cdx31(2) - cdx21(2)*cdx31(1)

c Norm of reference normal vector (twice area of triangle)

rnormn = sqrt(rn(1)*rn(1) +rn(2)*rn(2) +rn(3)*rn(3))

c Set G_iI transformation

rg(1,1) = 1.d0/sqrt(rdx21(1)*rdx21(1)
& + rdx21(2)*rdx21(2)
& + rdx21(3)*rdx21(3))
rg(2,2) = 1.d0/(rnormn*rg(1,1))
rg(1,2) = -rg(1,1)**2*rg(2,2)*(rdx21(1)*rdx31(1)
& + rdx21(2)*rdx31(2)
& + rdx21(3)*rdx31(3))

c Form Q-array

q(1,1) = rg(1,1)*rg(1,1)
q(2,1) = rg(1,2)*rg(1,2)

```



```

q(3,1) = rg(1,2)*rg(1,1)*2.d0
q(2,2) = rg(2,2)*rg(2,2)
q(2,3) = rg(1,2)*rg(2,2)
q(3,3) = rg(1,1)*rg(2,2)

c      Local Green-Lagrange deformation tensor

      gg(1,1) = cdx21(1)*cdx21(1) + cdx21(2)*cdx21(2)
&      + cdx21(3)*cdx21(3)
      gg(1,2) = cdx21(1)*cdx31(1) + cdx21(2)*cdx31(2)
&      + cdx21(3)*cdx31(3)
      gg(2,1) = gg(1,2)
      gg(2,2) = cdx31(1)*cdx31(1) + cdx31(2)*cdx31(2)
&      + cdx31(3)*cdx31(3)

c      Compute global Green-Lagrange strain tensor

      eps(1) = 0.5d0* gg(1,1)*q(1,1) - 0.5d0
      eps(2) = 0.5d0*(gg(1,1)*q(2,1) + gg(2,2)*q(2,2))
&      + gg(1,2)*q(2,3) - 0.5d0
      eps(3) = 0.5d0* gg(1,1)*q(3,1) + gg(1,2)*q(3,3)

c      Form Stiffness and Residual arrays

      if(isw.eq.3 .or. isw.eq.6) then

c          Compute stresses and moduli

          call strso1(d,eps,sig,dd)

c          Store time history plot data for element

          do j = 1,6
            tt(j) = sig(j)
          end do ! j

c          Form strain displacement matrix

          do i = 1,ndm
            bb(1,i ) = -cdx21(i)
            bb(1,i+3) = cdx21(i)
            bb(1,i+6) = 0.0d0
            bb(2,i ) = -cdx31(i)
            bb(2,i+3) = 0.0d0
            bb(2,i+6) = cdx31(i)
            bb(3,i ) = -cdx21(i) - cdx31(i)
            bb(3,i+3) = cdx31(i)
            bb(3,i+6) = cdx21(i)
          end do ! i

          do i = 1,3
            do j = 1,9
              bmat(i,j) = q(i,1)*bb(1,j)
&              + q(i,2)*bb(2,j)
&              + q(i,3)*bb(3,j)
            end do ! j
          end do ! i

c          Set loading factors

          he = d(15)
          dv = 0.5d0*he*rnormn

```

```

ro    = d( 4)*he
rb(1) = d(11)*he*rnormn/3.d0
rb(2) = d(12)*he*rnormn/3.d0
rb(3) = d(13)*he*rnormn/3.d0
pr    = d(14)/6.d0
cfac  = d(18)
lfac  = 1.d0 - cfac

c      Lumped mass term

mass = ro*dv/3.d0

c      Scale stress by volume

do i = 1,3
  sig(i) = sig(i)*dv
  do j = 1,3
    dd(j,i) = dd(j,i)*dv*ctan(1)
  end do ! j
end do ! i

c      Form residual

do i = 1,3
  r(i,1) = rb(i) + pr*cn(i) - mass*ul(i,1,5)
&        - bmat(1,i  )*sig(1)
&        - bmat(2,i  )*sig(2)
&        - bmat(3,i  )*sig(3)

  r(i,2) = rb(i) + pr*cn(i) - mass*ul(i,2,5)
&        - bmat(1,i+3)*sig(1)
&        - bmat(2,i+3)*sig(2)
&        - bmat(3,i+3)*sig(3)

  r(i,3) = rb(i) + pr*cn(i) - mass*ul(i,3,5)
&        - bmat(1,i+6)*sig(1)
&        - bmat(2,i+6)*sig(2)
&        - bmat(3,i+6)*sig(3)
end do ! i

c      Form material tangent

if(isw.eq.3) then
  do i = 1,9
    do j = 1,3
      bdi(j) = bmat(1,i)*dd(1,j)
&           + bmat(2,i)*dd(2,j)
&           + bmat(3,i)*dd(3,j)
    end do ! J
    do j = 1,9
      s(is(i),is(j)) = bdi(1)*bmat(1,j)
&                   + bdi(2)*bmat(2,j)
&                   + bdi(3)*bmat(3,j)
    end do ! j
  end do ! i

c      Add inertial tangent

  s(is(i),is(i)) = s(is(i),is(i)) + mass*ctan(3)
end do ! i

c      Form geometric tangent

```

```

      ss(1) = ctan(1)*(q(1,1)*sig(1) + q(2,1)*sig(2)
&          + q(3,1)*sig(3))
      ss(2) = ctan(1)*(q(2,2)*sig(2))
      ss(3) = ctan(1)*(q(2,3)*sig(2) + q(3,3)*sig(3))

      do i = 1,3
&        s(is(i ),is(i )) = s(is(i ),is(i )) + ss(1) + ss(2)
          + ss(3)*2.d0
        s(is(i+3),is(i )) = s(is(i+3),is(i )) - ss(1) - ss(3)
        s(is(i+6),is(i )) = s(is(i+6),is(i )) - ss(2) - ss(3)
        s(is(i ),is(i+3)) = s(is(i ),is(i+3)) - ss(1) - ss(3)
        s(is(i+3),is(i+3)) = s(is(i+3),is(i+3)) + ss(1)
        s(is(i+6),is(i+3)) = s(is(i+6),is(i+3)) + ss(3)
        s(is(i ),is(i+6)) = s(is(i ),is(i+6)) - ss(2) - ss(3)
        s(is(i+3),is(i+6)) = s(is(i+3),is(i+6)) + ss(3)
        s(is(i+6),is(i+6)) = s(is(i+6),is(i+6)) + ss(2)
      end do ! i

c      Follower pressure tangent

      do i = 1,3
        cdx21(i) = cdx21(i)*pr*ctan(1)
        cdx31(i) = cdx31(i)*pr*ctan(1)
      end do ! i

      do i = 1,9,3
        s(is(i ),is(2)) = s(is(i ),is(2)) - cdx21(3) + cdx31(3)
        s(is(i ),is(3)) = s(is(i ),is(3)) + cdx21(2) - cdx31(2)
        s(is(i ),is(5)) = s(is(i ),is(5)) - cdx31(3)
        s(is(i ),is(6)) = s(is(i ),is(6)) + cdx31(2)
        s(is(i ),is(8)) = s(is(i ),is(8)) + cdx21(3)
        s(is(i ),is(9)) = s(is(i ),is(9)) - cdx21(2)

        s(is(i+1),is(1)) = s(is(i+1),is(1)) + cdx21(3) - cdx31(3)
        s(is(i+1),is(3)) = s(is(i+1),is(3)) - cdx21(1) + cdx31(1)
        s(is(i+1),is(4)) = s(is(i+1),is(4)) + cdx31(3)
        s(is(i+1),is(6)) = s(is(i+1),is(6)) - cdx31(1)
        s(is(i+1),is(7)) = s(is(i+1),is(7)) - cdx21(3)
        s(is(i+1),is(9)) = s(is(i+1),is(9)) + cdx21(1)

        s(is(i+2),is(1)) = s(is(i+2),is(1)) - cdx21(2) + cdx31(2)
        s(is(i+2),is(2)) = s(is(i+2),is(2)) + cdx21(1) - cdx31(1)
        s(is(i+2),is(4)) = s(is(i+2),is(4)) - cdx31(2)
        s(is(i+2),is(5)) = s(is(i+2),is(5)) + cdx31(1)
        s(is(i+2),is(7)) = s(is(i+2),is(7)) + cdx21(2)
        s(is(i+2),is(8)) = s(is(i+2),is(8)) - cdx21(1)
      end do ! i

      endif ! stiffness form

c      Output of element quantities

      elseif(isw.eq.4) then

c      Compute stresses and moduli

      call strso1(d,eps,sig,dd)
      call pstr2d(sig,sig(7))

      xx = (xl(1,1) + xl(1,2) + xl(1,3))/3.d0
      yy = (xl(2,1) + xl(2,2) + xl(2,3))/3.d0

```

```

c      Output stresses and strains

      mct = mct - 2
      if(mct.le.0) then
        write(iow,2001) head
        if(ior.lt.0 .and. pfr) then
          write(*,2001) head
        endif
        mct = 50
      endif
      write(iow,2002) n,ma,sig(9),(sig(i),i=1,4),sig(7),
&                  xx,yy,(eps(i),i=1,4),sig(8)
      if(ior.lt.0 .and. pfr) then
&        write(*,2002) n,ma,sig(9),(sig(i),i=1,4),sig(7),
&                  xx,yy,(eps(i),i=1,4),sig(8)
      endif

c      Compute lumped mass matrix

      elseif(isw.eq.5) then

        dv   = 0.5d0*he*rnormn
        ro   = d( 4)*he
        cfac = d(18)
        lfac = 1.d0 - cfac
        mass = ro*dv/3.d0

        do i = 1,ndm
          r(i,1) = mass
          r(i,2) = mass
          r(i,3) = mass
        end do ! i

        do i = 1,9
          s(is(i),is(i)) = mass
        end do ! i

      endif

    endif

c      Formats for output

2001 format(20a4//5x,'Element Stresses'//'   Elmt Mat Angle',
& ' 11-stress 22-stress 33-stress 12-stress',
& ' 1-stress'//' 1-coord 2-coord 11-strain',
& ' 22-strain 33-strain 12-strain 2-stress')

2002 format(i8,i4,0p,f6.1,1p,5e12.3/0p,2f9.3,1p,5e12.3/1x)

end

subroutine inpt01(d)

implicit none

include 'hdata.h'
include 'iofile.h'

character text(2)*15
logical pcomp
real*8 e1,e2,nu12,g12, det, ev(7),d(*)

```

```

c   Default thickness value
    d(15) = 1.d0
c   Input record
10  call tinput(text,2,ev,7)
c   Input thickness
    if (pcomp(text(1),'thic',4)) then
        d(15) = ev(1)
c   Input density
    elseif(pcomp(text(1),'dens',4)) then
        d(4) = ev(1)
c   Input body forces
    elseif(pcomp(text(1),'body',4)) then
        d(11) = ev(1)
        d(12) = ev(2)
        d(13) = ev(3)
c   Input pressure load
    elseif(pcomp(text(1),'pres',4)) then
        d(14) = ev(1)
c   Input principal direction angle
    elseif(pcomp(text(1),'angl',4)) then
        d(31) = ev(1)
c   Input elastic properties
    elseif(pcomp(text(1),'elas',4)) then
c   Orthotropic inputs
        if(pcomp(text(2),'orth',4)) then
            e1 = ev(1)
            e2 = ev(2)
            nu12 = ev(3)
            g12 = ev(4)
c   Isotropic inputs
        else
            e1 = ev(1)
            e2 = e1
            nu12 = ev(2)
            g12 = 0.5d0*e1/(1.d0 + nu12)
            d(1) = e1
            d(2) = nu12
        endif
endif

```

```

elseif(pcomp(text(1),' ',4)) then
  go to 11
endif

go to 10

c   Set the moduli

11  det  = 1.d0/(e2 - e1*nu12*nu12)
    d(21) = e1*e2*det
    d(22) = e2*e2*det
    d(23) = d(21)*nu12
    d(24) = g12

c   Output parameters for element

write(iow,2001) e1,e2,nu12,g12,d(31),
&              d(15),d(4),d(11),d(12),d(13),d(14)

if(ior.lt.0) then
  write(*,2001) e1,e2,nu12,g12,d(31),
&              d(15),d(4),d(11),d(12),d(13),d(14)
endif

c.   Formats

2001 format(/5x,'E l a s t i c   S o l i d   E l e m e n t'//
&          10x,'M e c h a n i c a l   P r o p e r t i e s'//
&          10x,'Plane Stree Analysis'//
&          10x,'Modulus E-1 ', 1p,e16.5/
&          10x,'Modulus E-2 ', 1p,e16.5/
&          10x,'Poisson ratio 12',0p,f8.5 /
&          10x,'Modulus G-12', 1p,e16.5/
&          10x,'Angle (psi) ',0p,f8.5 /
&          10x,'Thickness ', 1p,e16.5/
&          10x,'Density ', 1p,e16.5/
&          10x,'1-gravity ', 1p,e16.5/
&          10x,'2-gravity ', 1p,e16.5/
&          10x,'3-gravity ', 1p,e16.5/
&          10x,'Pressure ', 1p,e16.5)

end

subroutine str01(d,eps,sig,dd)

implicit none

include 'eldata.h'
include 'sdata.h'

integer i
real*8 d(*), eps(3), sig(3), dd(3,3)

c   Get moduli

call dmat01(d,d(31),dd)

c   Compute stresses

do i = 1,3
  sig(i) = dd(i,1)*eps(1) + dd(i,2)*eps(2) + dd(i,3)*eps(3)
end do ! i

```

```

end

subroutine dmat01(d,psi,dmg)

c      Rotation of material arrays from principal to local element directions

c      Inputs:
c      d      - Array with material properties
c      psi    - Angle from y1-axis (local) to 1-axis (principal)

c      Output:
c      dmg(3,3) - Plane modulus matrix

c      Variables used in subroutine

c      qm(3,3) - Transformation matrix for plane problems

c      dml(3,3) - Local (orthotropic ) plane modulus matrix
c      dmlqj(3) - intermediate matrix for triple product
-----
c      implicit none

c      integer i, j
c      real*8 psi, si, co, s2, c2, cs
c      real*8 d(*), dml(3,3), dmg(3,3), qm(3,3), dmlqj(3)

c      Assign material properties

c      No rotation

c      if(psi.eq.0.0d0) then

c          dmg(1,1) = d(21)
c          dmg(2,2) = d(22)
c          dmg(3,3) = d(24)

c          dmg(1,2) = d(23)
c          dmg(2,1) = dmg(1,2)

c          dmg(1,3) = 0.0d0
c          dmg(3,1) = 0.0d0

c          dmg(2,3) = 0.0d0
c          dmg(3,2) = 0.0d0

c      Orthotropic material (rotations)

c      else

c      Set constants for transformation

c          si = sin(psi)
c          co = cos(psi)
c          s2 = si*si
c          c2 = co*co
c          cs = co*si

c      Set transformation matrix

c          qm(1,1) = c2
c          qm(1,2) = s2

```

```

qm(1,3) = cs
qm(2,1) = s2
qm(2,2) = c2
qm(2,3) = -cs
qm(3,1) = -2.d0 * cs
qm(3,2) = 2.d0 * cs
qm(3,3) = c2 - s2

```

c Set local (orthotropic) plane matrix

```

dml(1,1) = d(21)
dml(2,2) = d(22)
dml(3,3) = d(24)

dml(1,2) = d(23)
dml(2,1) = dml(1,2)

dml(2,3) = 0.0d0
dml(3,2) = 0.0d0

dml(3,1) = 0.0d0
dml(1,3) = 0.0d0

```

c Convert plane local to global matrix

```

do j = 1,3 ! {
    dmlqj(1) = dml(1,1)*qm(1,j) + dml(1,2)*qm(2,j)
    dmlqj(2) = dml(2,1)*qm(1,j) + dml(2,2)*qm(2,j)
    dmlqj(3) = dml(3,3)*qm(3,j)

    do i = 1,3 ! {
        & dmg(i,j) = qm(1,i)*dmlqj(1) + qm(2,i)*dmlqj(2)
        + qm(3,i)*dmlqj(3)
    end do ! i }

end do ! j }

endif

end

```


B Reinforcement calculation

```

subroutine rein01(d,xi,xj,xl,xg,ul,r,s,ndm,ndf,nst)

implicit none

include 'cdata.h'
include 'eltran.h'

integer ndm,ndf,nst, i,j, a1,a2,a1, b1,b2,be
real*8 d(*),xi(3),xj(3),xl(ndm,3),xg(3,3),ul(ndf,ndm,5)
real*8 r(ndf,3),s(nst,nst), mm(9,9),cc(9,9)
real*8 dxji(3),dl(3),dg(3),sr(3,3),cr(3),mr(3)
real*8 lij,eij,sij, ra,ca,ea,sa

c Compute differences in natural coordinates

do i = 1,3
dxji(i) = xj(i) - xi(3)
end do ! i

c Compute reinforcement length and cross section properties

do i = 1,ndm
dl(i) = dxji(1)*xl(i,1) + dxji(2)*xl(i,2) + dxji(3)*xl(i,3)
dg(i) = dxji(1)*xg(i,1) + dxji(2)*xg(i,2) + dxji(3)*xg(i,3)
end do ! i

lij = sqrt(dl(1)**2 +dl(2)**2 +dl(3)**2)
eij = 0.5d0*sqrt(dg(1)**2 +dg(2)**2 +dg(3)**2)/lij - 0.5d0
sij = d(1)*eij
ra = d(1)*d(2)*lij*0.5d0
ca = d(1)*d(2)*lij*0.5d0
ea = d(1)*d(2)/lij
sa = sij*d(2)/lij

c Lumped properties of reinforcement

do i = 1,ndm
cr(i) = ca
mr(i) = ra
end do ! i

c Compute contribution to membrane mass and damping

a1 = 1
do a1 = 1,3
b1 = 1
do be = 1,3
cc(a1 ,b1 ) = cc(a1 ,b1 ) + xi(1)*cr(1)*xi(1)
& + xi(2)*cr(2)*xi(2)
& + xi(3)*cr(3)*xi(3)
& + xj(1)*cr(1)*xj(1)
& + xj(2)*cr(2)*xj(2)
& + xj(3)*cr(3)*xj(3)
mm(a1 ,b1 ) = mm(a1 ,b1 ) + xi(1)*mr(1)*xi(1)
& + xi(2)*mr(2)*xi(2)
& + xi(3)*mr(3)*xi(3)
& + xj(1)*mr(1)*xj(1)
& + xj(2)*mr(2)*xj(2)
& + xj(3)*mr(3)*xj(3)
cc(a1+1,b1+1) = cc(a1 ,b1 )

```

```

        mm(a1+1,b1+1) = mm(a1 ,b1 )
        cc(a1+2,b1+2) = cc(a1 ,b1 )
        mm(a1+2,b1+2) = mm(a1 ,b1 )
        b1 = b1 + 3
    end do ! be
    a1 = a1 + 3
end do ! a1

c   Residual computation

do a1 = 1,3
    a1 = 1
    do i = 1,ndm
        r(i,a1) = r(i,a1) - dxji(a1)*ea*dg(i)
&         - cc(a1,a1 )*ul(i,1,4)
&         - cc(a1,a1+3)*ul(i,2,4)
&         - cc(a1,a1+6)*ul(i,3,4)
&         - mm(a1,a1 )*ul(i,1,5)
&         - mm(a1,a1+3)*ul(i,2,5)
&         - mm(a1,a1+6)*ul(i,3,5)
    end do ! i
    a1 = a1 + 3
end do ! j

c   Final tangent stiffness

do i = 1,ndm
    do j = 1,ndm
        sr(i,j) = ea*dg(i)*dg(j)*ctan(1)
    end do ! j
    sr(i,i) = sr(i,i) + sa*ctan(1)
end do ! i

a1 = 1
a2 = 1
do a1 = 1,3
    b1 = 1
    b2 = 1
    do be = 1,3
        s(a1 ,b1 ) = s(a1 ,b1 ) + dxji(a1)*sr(1,1)*dxji(be)
&         + cc(a2 ,b2 )*ctan(2)
&         + mm(a2 ,b2 )*ctan(3)
        s(a1 ,b1+1) = s(a1 ,b1+1) + dxji(a1)*sr(1,2)*dxji(be)
        s(a1 ,b1+2) = s(a1 ,b1+2) + dxji(a1)*sr(1,3)*dxji(be)
        s(a1+1,b1 ) = s(a1+1,b1 ) + dxji(a1)*sr(2,1)*dxji(be)
        s(a1+1,b1+1) = s(a1+1,b1+1) + dxji(a1)*sr(2,2)*dxji(be)
&         + cc(a2+1,b2+1)*ctan(2)
&         + mm(a2+1,b2+1)*ctan(3)
        s(a1+1,b1+2) = s(a1+1,b1+2) + dxji(a1)*sr(2,3)*dxji(be)
        s(a1+2,b1 ) = s(a1+2,b1 ) + dxji(a1)*sr(3,1)*dxji(be)
        s(a1+2,b1+1) = s(a1+2,b1+1) + dxji(a1)*sr(3,2)*dxji(be)
        s(a1+2,b1+2) = s(a1+2,b1+2) + dxji(a1)*sr(3,3)*dxji(be)
&         + cc(a2+2,b2+2)*ctan(2)
&         + mm(a2+2,b2+2)*ctan(3)
    end do ! be
    b1 = b1 + ndf
    b2 = b2 + 3
end do ! a1
a1 = a1 + ndf
a2 = a2 + 3
end do ! a1

end

```