LSTC Report 1018 Rev. 2

LS-DYNA3D

THEORETICAL MANUAL

June 1991
October 1992, Rev. 1
July 1993, Rev. 2

JOHN O. HALLQUIST

LIVERMORE SOFTWARE
TECHNOLOGY CORPORATION
2876 Waverley Way
Livermore, CA 94550
USA

TABLE OF CONTENTS

Theoretical Manual for DYNA3D

ABSTRACT

This report provides a theoretical manual for LS-DYNA3D, a vectorized explicit three-dimensional finite element code for analyzing the large deformation dynamic response of inelastic solids. A contact-impact algorithm that permits gaps and sliding along material interfaces is described. By a specialization of this algorithm, such interfaces can be rigidly tied to admit variable zoning without the need of transition regions. Spatial discretization is achieved by the use of eight node solid elements, two node beam elements, three and four node shell elements, eight node solid shell elements, truss elements, membrane elements, discrete elements, and rigid bodies. The equations of motion are integrated in time by the central difference method. LS-DYNA3D currently contains thirty-nine material models and ten equations of state to cover a wide range of material behavior. LS-DYNA3D is operational on a large number of mainframes and workstations.

1. INTRODUCTION

LS-DYNA3D is seeing extensive use by many of the world's top automobile, aerospace, and ordnance companies, by government and corporate laboratories and research organizations, and in universities. This theoretical manual has been written to provide users and potential users with insight into the mathematical and physical basis of the code. It replaces the original manual, published in 1983 [Hallquist 1983] and now obsolete.

The code has gone through a great deal of development since its inception in 1976, as it has been improved and speeded up and new features have been added to meet the needs of users. Below we summarize the principal features of each of the nine versions from 1976 to 1993.

1.1 History of DYNA3D

The first version of DYNA3D [Hallquist 1976a] was released in 1976 with constant stress 4- or 8-node solid elements, 16- and 20-node solid elements with $2 \times 2 \times 2$ Gaussian quadrature, 3-, 4-, and 8-node membrane elements, and a 2-node cable element.

A nodal constraint contact-impact interface algorithm [Hallquist 1977] was available. On the Control Data CDC-7600, a supercomputer in 1976, the speed of the code varied from 36 minutes per 10^6 mesh cycles with 4-8 node solids to 180 minutes per 10^6 mesh cycles with 16 and 20 node solids. Without hourglass control to prevent formation of non-physical zero energy deformation modes, constant stress solids were processed at 12 minutes per 10^6 mesh cycles. A moderate number of very costly solutions were obtained with this version of DYNA3D using 16- and 20-node solids. Hourglass modes prevented us from obtaining solutions with constant stress elements.

In this early development, several things became apparent. Hourglass deformation modes of the constant stress elements were invariably excited by the contact-impact algorithm, showing that a new sliding interface algorithm was needed. Higher order elements seemed to be impractical for shock wave propagation because of numerical noise resulting from the ad hoc way mass is lumped to generate a diagonal mass matrix. Although lower frequency structural response was accurately computed with these elements, their high computer cost made analysis so expensive as to be impractical. We were eventually convinced that realistic three-dimensional structural calculations were possible, if and only if the under-integrated eight node constant stress solid element could be made to function. This implied a need for a much better sliding interface algorithm, a more cost-effective hourglass control, more optimal programming, and a machine much faster than the CDC-7600. This latter need was fulfilled several years later when LLNL took deliver of its first CRAY-1. At this time, DYNA3D was completely rewritten.

The new version, released in 1979, achieved the aforementioned goals. On the CRAY the vectorized speed was 50 times faster, 0.67 minutes per million mesh cycles. A symmetric, penalty-based, contact-impact algorithm was considerably faster in execution speed and exceedingly reliable. Due to lack of use, the membrane and cable elements were stripped and all higher order elements were eliminated as well. Wilkins' finite difference equations [Wilkins et al. 1974] were implemented in unvectorized form in an overlay to compare their performance with the finite element method. The finite difference algorithm proved to be nearly two times more expensive than the finite element approach (apart from vectorization) with no compensating increase in accuracy, and was removed in the next code update.

The 1981 version [Hallquist 1981a] evolved from the 1979 version. Nine additional material models were added to allow a much broader range of problems to be modeled including explosive-structure and soil-structure interactions. Body force loads were implemented for angular velocities and base accelerations. A link was also

established from the 3D Eulerian code JOY [Couch, in prep.] for studying the structural response to impacts by penetrating projectiles. An option was provided for storing element data on disk thereby doubling the capacity of DYNA3D.

The 1982 version of DYNA3D [Hallquist 1982] accepted DYNA2D [Hallquist 1980] material input directly. The new organization was such that equations of state and constitutive models of any complexity could be easily added. Complete vectorization of the material models had been nearly achieved with about a 10 percent increase in execution speed over the 1981 version.

In the 1986 version of DYNA3D [Hallquist and Benson 1986], many new features were added, including beams, shells, rigid bodies, single surface contact, interface friction, discrete springs and dampers, optional hourglass treatments, optional exact volume integration, and VAX/VMS, IBM, UNIX, COS operating systems compatibility, that greatly expanded its range of applications. DYNA3D thus became the first code to have a general single surface contact algorithm.

In the 1987 version of DYNA3D [Hallquist and Benson 1987] metal forming simulations and composite analysis became a reality. This version included shell thickness changes ,the Belytschko-Tsay shell element [Belytschko and Tsay, 1981], and dynamic relaxation. Also included were non-reflecting boundaries, user specified integration rules for shell and beam elements, a layered composite damage model, and single point constraints.

New capabilities added in the 1988 DYNA3D [Hallquist 1988] version included a cost effective resultant beam element, a truss element, a C^0 triangular shell, the BCIZ triangular shell [Bazeley et al. 1965], mixing of element formulations in calculations, composite failure modeling for solids, noniterative plane stress plasticity, contact surfaces with spotwelds, tiebreak sliding surfaces, beam surface contact, finite stonewalls, stonewall reaction forces, energy calculations for all elements, a crushable foam constitutive model, comment cards in the input, and one-dimensional slidelines.

In the middle of 1988 the author began working halftime at LLNL to devote more time to the development and support of LS-DYNA3D for automotive applications. By the end of 1988 it was obvious that a much more concentrated effort would be required in the development of LS-DYNA3D if problems in crashworthiness were to be properly solved; therefore, at the start of 1989 the author resigned from LLNL to continue code development full time at Livermore Software Technology Corporation. The 1989 version of LS-DYNA3D introduced many enhanced capabilities including a one-way treatment of slide surfaces with voids and friction; cross-sectional forces for structural elements; an optional user specified minimum time step size for shell elements using elastic and elastoplastic material models; nodal accelerations in the time history database; a

compressible Mooney-Rivlin material model; a closed-form update shell plasticity model; a general rubber material model; unique penalty specifications for each slide surface; external work tracking; optional time step criterion for 4-node shell elements; and internal element sorting to allow full vectorization of right-hand-side force assembly.

The 1990 version of LS-DYNA3D [Hallquist 1990] included: arbitrary node and element numbers; a fabric model for seat belts and airbags; a composite glass model; vectorized type 3 contact and single surface contact; many more I/O options; all shell materials available for the 8-node brick shell; strain rate dependent plasticity for beams; fully vectorized iterative plasticity; interactive graphics on some computers; nodal damping; shell thickness taken into account in shell type 3 contact; shell thinning accounted for in type 3 and type 4 contact; soft stonewalls; print suppression option for node and element data; rivets; spotwelds; expanded TAURUS database; force limited resultant beam; rotational spring and dampers; local coordinate systems for discrete elements; resultant plasticity for C^0 triangular element; energy dissipation calculations for stonewalls; hourglass energy calculations for solid and shell elements; viscous and Coulomb friction with arbitrary variation over surface; distributed loads on beam elements; Cowper and Symonds strain rate model; segmented stonewalls; stonewall Coulomb friction; stonewall energy dissipation; airbags; nodal rigid bodies; and automatic sorting of triangular shells into C^0 groups.

In the 1991 version of LS-DYNA3D, new features included: mass scaling for quasistatic analyses; warpage checks on shell elements; thickness consideration in all contact types; automatic orientation of contact segments; sliding interface energy dissipation calculations; nodal force and energy database for applied boundary conditions; defined stonewall velocity with input energy calculations; user defined subroutines; rigid/deformable material switching; rigid bodies impacting rigid walls; strain rate effects in crushable foam model 26; shells and beams interfaces included for subsequent component analyses; external work computed for prescribed displacement/velocity/accelerations; linear constraint equations; the MPGS database; and the MOVIE database.

Many new features were added in the 1992 production version. Among these features are: automated contact input for all input types, automatic single surface contact without element orientation, constraint technique for contact, cut planes for resultant forces, crushable cellular foams, urethane foam model with hystersis, subcycling, friction in the contact entities, strains computed and written for the 8 node thick shells, “good” 4 node tetrahedron solid element with nodal rotations, 8 node solid element with nodal rotations, 2×2 integration for the membrane element, Belytschko-Schwer integrated beam,

thin-walled Belytschko-Schwer integrated beam, improved TAURUS database control, null material for beams to display springs and seatbelts in TAURUS, parallel implementation on Crays and SGI computers, coupling to rigid body codes, and seat belt capability.

1.2 Acknowledgments

The numerical techniques described below, especially those for treating shock waves, plasticity, and time integration, are based on work by Noh [1976], Wilkins [1964], and Maenchen and Sack [1964] in the finite difference literature. The finite element techniques are described in detail in numerous textbooks (see, for example, [Cook 1974], [Bathe 1982], [Hughes 1987]). DYNA3D is unique in its efficient, fully vectorized, and parallelized implementation of these well-known equations. The code organization of LS-DYNA3D, which has evolved considerably since 1976, had its origins in finite difference codes and its organization was influenced by the early finite element codes developed by Wilson (e.g., [Farhoomand and Wilson 1970]). As will be seen, many of the later developments are based on the pioneering work of Belytschko and his co-workers as well as Hughes and his co-workers.

Some of the most original work in LS-DYNA3D is the interface treatment for handling contact, sliding, and impact which has proven reliable, relatively low cost, and useful in a great many types of calculations. The rigid body capability and single surface contact was developed during the mid-eighties with Prof. David J. Benson, U.C. San Diego while working with David at the Lawrence Livermore National Laboratory.

Of course, much credit goes to the Lawrence Livermore National Laboratory where the author did the original development in an environment completely free of restraints and to the willingness and generosity of LLNL to release the code into the public domain thereby introducing finite element users everywhere in the world to the power of explicit software.

The author would like to thank those who read through this manual and made many excellent suggestions some of which will be added in future releases. In particular the author would like to thank Drs. Wayne V. Nack of Chrysler and now at General Motors, Klaus Weimar of CAD-FEM, Prof. Karl Schweizerhof of Karlsruhe University, and T.L. Lin of LSTC. Dr. L.E. Schwer worked closely with the author in assembling this theoretical manual especially in documenting the elements developed by Belytschko. The section on ALE was written with Prof. Benson and includes excerpts from a report on equipotential smoothing by Dr. Allan Winslow for LSTC.

In closing, the author would like to thank our secretary Valli A. James for typing this manuscript.

2. PRELIMINARIES

Consider the body shown in Figure 2.1. We are interested in time-dependent deformation in which a point in b initially at X_α (α = 1, 2, 3) in a fixed rectangular Cartesian coordinate system moves to a point x_i (i = 1, 2, 3) in the same coordinate system. Since a Lagrangian formulation is considered, the deformation can be expressed in terms of the convected coordinates X_α, and time t

$$x_i = x_i (X_\alpha, t) \tag{2.1}$$

At time t = 0 we have the initial conditions

$$x_i (X_\alpha, 0) = X_\alpha \tag{2.2a}$$

$$\dot{x}_i (X_\alpha, 0) = V_i (X_\alpha) \tag{2.2b}$$

where V_i defines the initial velocities.

2.1 Governing Equations

We seek a solution to the momentum equation

$$\sigma_{ij,j} + \rho f_i = \rho \ddot{x}_i \tag{2.3}$$

satisfying the traction boundary conditions

$$\sigma_{ij} n_i = t_i (t) \tag{2.4}$$

on boundary ∂b_1, the displacement boundary conditions

$$x_i (X_\alpha, t) = D_i (t) \tag{2.5}$$

on boundary ∂b_2, the contact discontinuity

$$\left(\sigma_{ij}^{+} - \sigma_{ij}^{-}\right) n_i = 0 \tag{2.6}$$

along an interior boundary ∂b_3 when $x_i^{+} = x_i^{-}$. Here σ_{ij} is the Cauchy stress, ρ is the current density, f_i is the body force density, $\ddot{x}$ is acceleration, the comma denotes covariant differentiation, and n_j is a unit outward normal to a boundary element of ∂b.

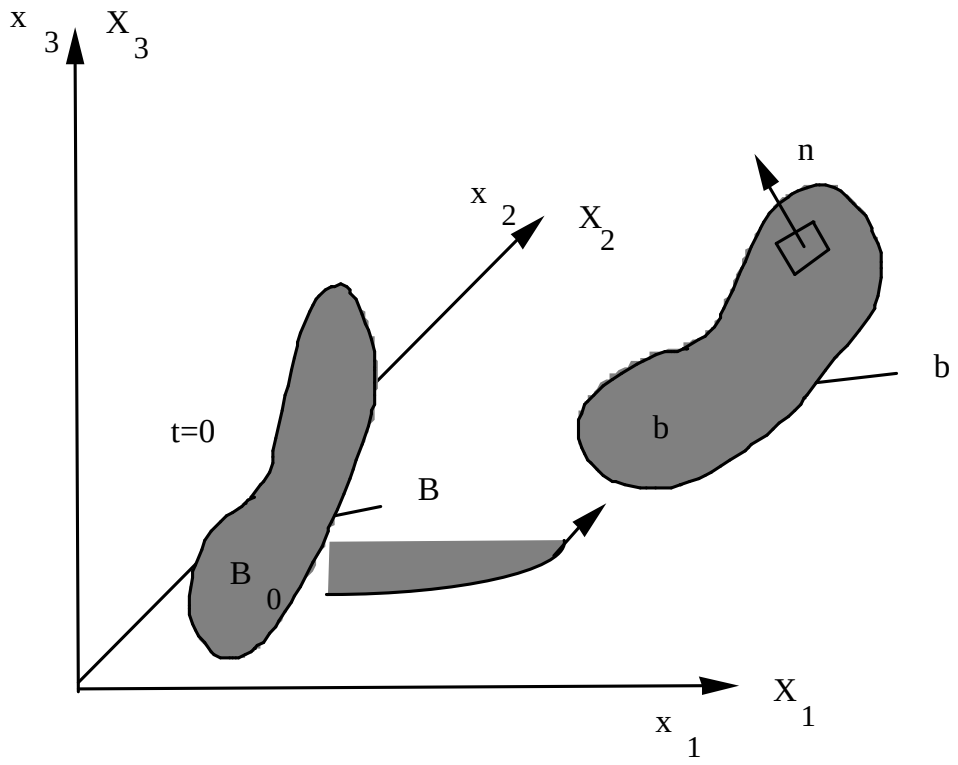


Figure 2.1. Notation.

Mass conservation is trivially stated

$$\rho V = \rho_0 \tag{2.7}$$

where V is the relative volume, i.e., the determinant of the deformation gradient matrix, F_{ij},

$$F_{ij} = \frac{\partial x_i}{\partial X_j} \tag{2.8}$$

and ρ_0 is the reference density. The energy equation

$$\dot{E} = V s_{ij} \dot{\varepsilon}_{ij} - (p+q)\dot{V} \tag{2.9}$$

is integrated in time and is used for equation of state evaluations and a global energy balance. In Equation (2.9), s_{ij} and p represent the deviatoric stresses and pressure,

$$s_{ij} = \sigma_{ij} + (p + q)\,\delta_{ij} \tag{2.10}$$

$$p = -\frac{1}{3}\sigma_{ij}\delta_{ij} - q = -\frac{1}{3}\sigma_{kk} - q \tag{2.11}$$

respectively, q is the bulk viscosity, δ_{ij} is the Kronecker delta ($\delta_{ij} = 1$ if i = j; otherwise $\delta_{ij} = 0$) and $\dot{\varepsilon}_{ij}$ is the strain rate tensor. The strain rates and bulk viscosity are discussed later.

We can write:

$$\int_v \left(\rho\ddot{x}_i - \sigma_{ij,j} - \rho f\right)\delta x_i d\upsilon + \int_{\partial b_1}\left(\sigma_{ij}n_j - t_i\right)\delta x_i ds + \int_{\partial b_3}\left(\sigma_{ij}^{+} - \sigma_{ij}^{-}\right)n_j\delta x_i ds = 0 \tag{2.12}$$

where δx_i satisfies all boundary conditions on ∂b_2, and the integrations are over the current geometry. Application of the divergence theorem gives

$$\int_\upsilon \left(\sigma_{ij}\delta x_i\right)_{,j} d\upsilon = \int_{\partial b_1}\sigma_{ij}n_j\delta x_i ds + \int_{\partial b_3}\left(\sigma_{ij}^{+} - \sigma_{ij}^{-}\right)n_j\delta x_i ds \tag{2.13}$$

and noting that

$$(\sigma_{ij}\,\delta x_i)_{,j} - \sigma_{ij,j}\,\delta x_i = \sigma_{ij}\,\delta x_{i,j} \tag{2.14}$$

leads to the weak form of the equilibrium equations:

$$\delta\pi = \int_\upsilon \rho\ddot{x}_i\,\delta x_i d\upsilon + \int_\upsilon \sigma_{ij}\delta x_{i,j} d\upsilon - \int_\upsilon \rho f_i\delta x_i d\upsilon - \int_{\partial b_1} t_i\delta x_i ds = 0 \tag{2.15}$$

a statement of the principle of virtual work.

We superimpose a mesh of finite elements interconnected at nodal points on a reference configuration and track particles through time, i.e.,

$$x_i\left(X_\alpha, t\right) = x_i\left(X_\alpha(\xi,\eta,\zeta), t\right) = \sum_{j=1}^{k}\phi_j(\xi,\eta,\zeta)x_i^j(t) \tag{2.16}$$

where ϕ_j are shape (interpolation) functions of the parametric coordinates (ξ, η, ζ), k is the number of nodal points defining the element, and is x_i^j the nodal coordinate of the jth node in the ith direction.

Summing over the n elements we may approximate $\delta\pi$ with

$$\delta\pi = \sum_{m=1}^{n} \delta\pi_m = 0 \tag{2.17}$$

and write

$$\sum_{m=1}^{n} \left\{ \int_{\upsilon_m} \rho \ddot{x}_i \Phi_i^m d\upsilon + \int_{\upsilon_m} \sigma_{ij}^m \Phi_{i,j}^m d\upsilon - \int_{\upsilon_m} \rho f_i \Phi_i^m d\upsilon - \int_{\partial b_1} t_i \Phi_i^m ds \right\} = 0 \tag{2.18}$$

where

$$\Phi_i^m = \left(\phi_1, \phi_2, ..., \phi_k\right)_i^m \tag{2.19}$$

In matrix notation Equation (2.18) becomes

$$\sum_{m=1}^{n} \left\{ \int_{\upsilon_m} \rho \mathrm{N}^t \mathrm{N} a \, d\upsilon + \int_{\upsilon_m} \mathrm{B}^t \sigma \, d\upsilon - \int_{\upsilon_m} \rho \mathrm{N}^t \mathrm{b} \, d\upsilon - \int_{\partial b_1} \mathrm{N}^t \mathrm{t} \, ds \right\}^m = 0 \tag{2.20}$$

where N is an interpolation matrix, σ is the stress vector

$$\sigma^t = (\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{yz}, \sigma_{zx}) \tag{2.21}$$

B is the strain-displacement matrix, a is the nodal acceleration vector

$$\begin{bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \ddot{x}_3 \end{bmatrix} = \mathrm{N} \begin{bmatrix} a_{x_1} \\ a_{y_1} \\ \vdots \\ a_{y_k} \\ a_{z_k} \end{bmatrix} = \mathrm{N} a \tag{2.22}$$

b is the body force load vector, and t are applied traction loads.

$$b = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}, \quad t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{2.23}$$

3. SOLID ELEMENTS

For a mesh of 8-node hexahedron solid elements, Equation (2.16) becomes:

$$x_i(X_\alpha, t) = x_i\left(X_\alpha(\xi,\eta,\zeta), t\right) = \sum_{j=1}^{8} \phi_j(\xi,\eta,\zeta) x_i^j(t) \tag{3.1}$$

The shape function ϕ_j is defined for the 8-node hexahedron as

$$\phi_j = \frac{1}{8}(1+\xi\xi_j)(1+\eta\eta_j)(1+\zeta\zeta_j) \tag{3.2}$$

where ξ_j, η_j, ζ_j take on their nodal values of (±1, ±1, ±1) and x_i^j is the nodal coordinate of the *j*th node in the *i*th direction (see Figure 3.1).

For a solid element, N is the 3×24 rectangular interpolation matrix give by

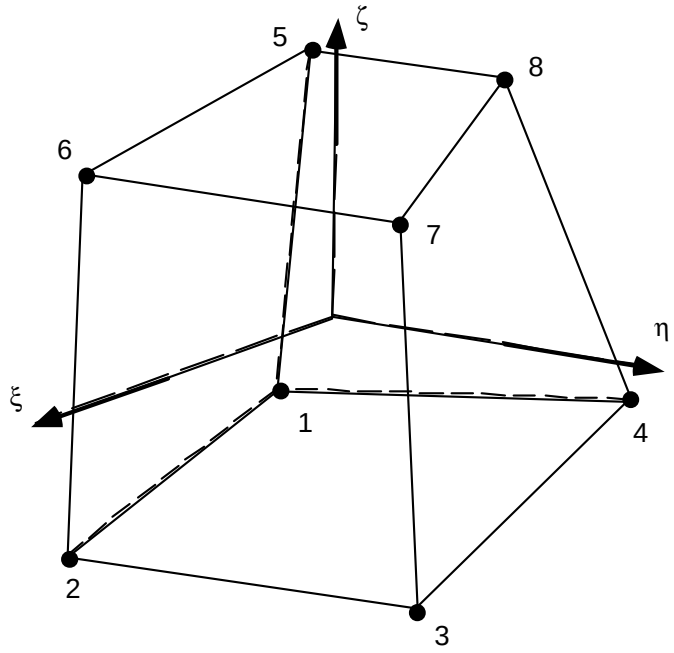
$$\mathrm{N}(\xi,\eta,\zeta) = \begin{bmatrix} \phi_1 & 0 & 0 & \phi_2 & 0 & \cdots & 0 & 0 \\ 0 & \phi_1 & 0 & 0 & \phi_2 & \cdots & \phi_8 & 0 \\ 0 & 0 & \phi_1 & 0 & 0 & \cdots & 0 & \phi_8 \end{bmatrix} \tag{3.3}$$

σ is the stress vector

$$\sigma^t = (\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{yz}, \sigma_{zx}) \tag{3.4}$$

B is the 6×24 strain-displacement matrix

$$\mathrm{B} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \end{bmatrix} \mathrm{N} \tag{3.5}$$



Node	ξ	η	ζ
1	-1	-1	-1
2	1	-1	-1
3	1	1	-1
4	-1	1	-1
5	-1	-1	1
6	1	-1	1
7	1	1	1
8	-1	1	1

Figure 3.1. Eight-node solid hexahedron element.

In order to achieve a diagonal mass matrix the rows are summed giving the kth diagonal term as

$$m_{kk} = \int_{\upsilon} \rho \phi_k \sum_{i=1}^{8} \phi_i \, d\upsilon = \int_{\upsilon} \rho \phi_k \, d\upsilon \tag{3.6}$$

since the basis functions sum to unity.

Terms in the strain-displacement matrix are readily calculated. Note that

$$\begin{aligned}
\frac{\partial \phi_i}{\partial \xi} &= \frac{\partial \phi_i}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial \phi_i}{\partial y}\frac{\partial y}{\partial \xi} + \frac{\partial \phi_i}{\partial z}\frac{\partial z}{\partial \xi} \\
\frac{\partial \phi_i}{\partial \eta} &= \frac{\partial \phi_i}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial \phi_i}{\partial y}\frac{\partial y}{\partial \eta} + \frac{\partial \phi_i}{\partial z}\frac{\partial z}{\partial \eta} \\
\frac{\partial \phi_i}{\partial \zeta} &= \frac{\partial \phi_i}{\partial x}\frac{\partial x}{\partial \zeta} + \frac{\partial \phi_i}{\partial y}\frac{\partial y}{\partial \zeta} + \frac{\partial \phi_i}{\partial z}\frac{\partial z}{\partial \zeta}
\end{aligned} \tag{3.7}$$

which can be rewritten as

$$\begin{bmatrix} \frac{\partial \phi_i}{\partial \xi} \\ \frac{\partial \phi_i}{\partial \eta} \\ \frac{\partial \phi_i}{\partial \zeta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} \begin{bmatrix} \frac{\partial \phi_i}{\partial x} \\ \frac{\partial \phi_i}{\partial y} \\ \frac{\partial \phi_i}{\partial z} \end{bmatrix} = J \begin{bmatrix} \frac{\partial \phi_i}{\partial x} \\ \frac{\partial \phi_i}{\partial y} \\ \frac{\partial \phi_i}{\partial z} \end{bmatrix} \tag{3.8}$$

Inverting the Jacobian matrix, J, we can solve for the desired terms

$$\begin{bmatrix} \frac{\partial \phi_i}{\partial x} \\ \frac{\partial \phi_i}{\partial y} \\ \frac{\partial \phi_i}{\partial z} \end{bmatrix} = J^{-1} \begin{bmatrix} \frac{\partial \phi_i}{\partial \xi} \\ \frac{\partial \phi_i}{\partial \eta} \\ \frac{\partial \phi_i}{\partial \zeta} \end{bmatrix} \tag{3.9}$$

3.1 Volume Integration

Volume integration is carried out with Gaussian quadrature. If g is some function defined over the volume, and n is the number of integration points, then

$$\int_{\upsilon} g \, d\upsilon = \int_{-1}^{1} \int_{-1}^{1} \int_{-1}^{1} g \left| J \right| d\xi \, d\eta \, d\zeta \tag{3.10}$$

is approximated by

$$\sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} g_{jkl} \left| J_{jkl} \right| w_j w_k w_l \tag{3.11}$$

where w_j, w_k, w_l are the weighting factors,

$$g_{jkl} = g\,(\xi_j, \eta_k, \zeta_l) \tag{3.12}$$

and $|\mathbf{J}|$ is the determinant of the Jacobian matrix. For one-point quadrature

$$\begin{gathered} n = 1 \\ w_i = w_j = w_k = 2 \\ \xi_1 = \eta_1 = \zeta_1 = 0 \end{gathered} \tag{3.13}$$

and we can write

$$\int g d\upsilon = 8g\,(0,0,0)\,|\,J\,(0,0,0)\,| \tag{3.14}$$

Note that $8\,|\,J\,(0,0,0)\,|$ approximates the element volume.

Perhaps the biggest advantage to one-point integration is a substantial savings in computer time. An anti-symmetry property of the strain matrix

$$\begin{aligned} \frac{\partial \phi_1}{\partial x_i} &= -\frac{\partial \phi_7}{\partial x_i} \quad \frac{\partial \phi_3}{\partial x_i} = -\frac{\partial \phi_5}{\partial x_i} \\ \frac{\partial \phi_2}{\partial x_i} &= -\frac{\partial \phi_8}{\partial x_i} \quad \frac{\partial \phi_4}{\partial x_i} = -\frac{\partial \phi_6}{\partial x_i} \end{aligned} \tag{3.15}$$

at $\xi = \eta = \zeta = 0$ reduces the amount of effort required to compute this matrix by more than 25 times over an 8-point integration. This cost savings extends to strain and element nodal force calculations where the number of multiplies is reduced by a factor of 16. Because only one constitutive evaluation is needed, the time spent determining stresses is reduced by a factor of 8. Operation counts for the constant stress hexahedron are given in Table 3.2. Included are counts for the Flanagan and Belytschko [1981] hexahedron and the hexahedron used by Wilkins [1974] in his integral finite difference method which was also implemented [Hallquist 1979].

It may be noted that 8-point integration has another disadvantage in addition to cost. Fully integrated elements used in the solution of plasticity problems and other problems where Poisson's ratio approaches .5 lock up in the constant volume bending modes. To preclude locking, an average pressure must be used over the elements; consequently, the zero energy modes are resisted by the deviatoric stresses. If the deviatoric stresses are insignificant relative to the pressure or, even worse, if material failure causes loss of this stress state component, hourglassing will still occur, but with no means of resisting it. Sometimes, however, the cost of the fully integrated element may be justified by increased reliability and if used sparingly may actually increase the overall speed.

3.2 Hourglass Control

The biggest disadvantage to one-point integration is the need to control the zero energy modes, called hourglassing modes, which arise. Undesirable hourglass modes tend to have periods that are typically much shorter than the periods of the structural

response, and they are often observed to be oscillatory. However, hourglass modes that have periods that are comparable to the structural response periods are a stable kinematic component of the global deformation modes and must be admissible. One way of resisting undesirable hourglassing is with a viscous damping or small elastic stiffness capable of stopping the formation of the anomalous modes but having a negligible affect on the stable global modes. Two of the early three-dimensional algorithms for controlling the hourglass modes were developed by Kosloff and Frazier [1974] and Wilkins et al. [1974]. Since the hourglass modes are made orthogonal to the real deformation, work done by the hourglass resistance is neglected in the energy equation. This may lead to a slight loss of energy; however, hourglass control is always recommended for the underintegrated solid elements. The energy dissipated in the hourglass modes is tracked and reported in the output files MATSUM and GLSTAT.

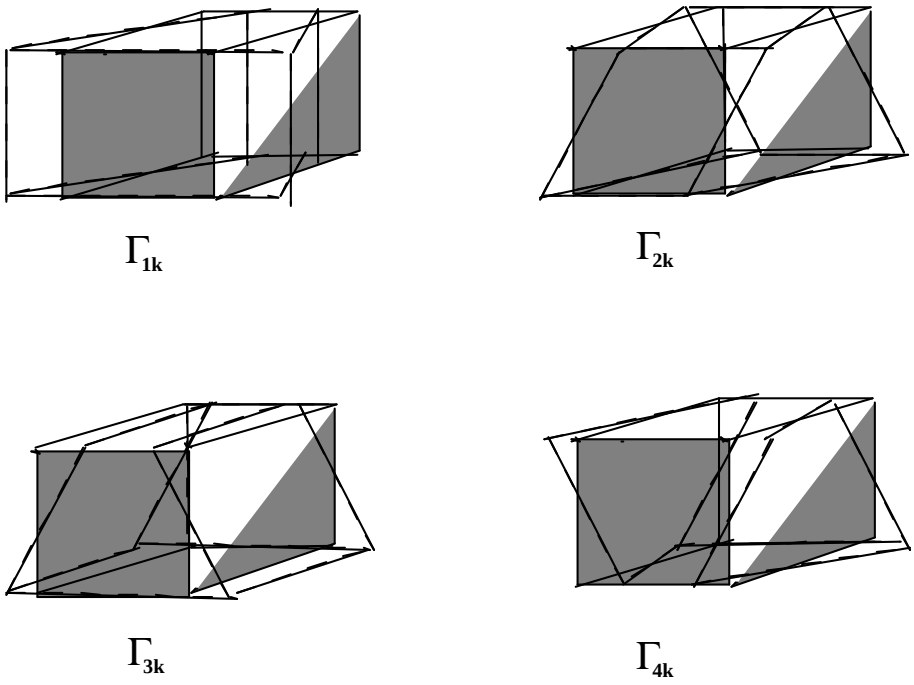


Figure 3.2. Hourglass modes of an eight node element with one integration point [Flanagan and Belytschko 1981].

It is easy to understand the reasons for the formation of the hourglass modes. Consider the following strain rate calculations for the 8-node solid element

$$\dot{\varepsilon}_{ij} = \frac{1}{2}\left(\sum_{k=1}^{8} \frac{\partial \phi_k}{\partial x_i} \dot{x}_j^k + \frac{\partial \phi_k}{\partial x_j} \dot{x}_i^k \right) \tag{3.16}$$

Whenever diagonally opposite nodes have identical velocities, i.e.,

$$\dot{x}_i^1 = \dot{x}_i^7 \ , \dot{x}_i^2 = \dot{x}_i^8 \ , \dot{x}_i^3 = \dot{x}_i^5 \ , \dot{x}_i^4 = \dot{x}_i^6 \tag{3.17}$$

the strain rates are identically zero:

$$\dot{\varepsilon}_{ij} = 0 \tag{3.18}$$

due to the asymmetries in Equations (3.15). It is easy to prove the orthogonality of the hourglass shape vectors which are listed in Table 3.1 and shown in Figure 3.2 with the derivatives of the shape functions:

$$\sum_{k=1}^{8} \frac{\partial \phi_k}{\partial x_i} \Gamma_{\alpha k} = 0 \qquad i = 1,2,3 \qquad \alpha = 1,2,3,4 \tag{3.19}$$

The product of the base vectors with the nodal velocities

$$h_{i\alpha} = \sum_{k=1}^{8} \dot{x}_i^k \Gamma_{\alpha k} = 0 \tag{3.20}$$

are nonzero if hourglass modes are present. The 12 hourglass-resisting force vectors, $f_{i\alpha}^k$ are

$$f_{i\alpha}^{\ k} = a_h h_{i\alpha} \Gamma_{\alpha k} \tag{3.21}$$

where

$$a_h = Q_{hg} \rho v_e^{2/3} \frac{c}{4} \tag{3.21}$$

in which v_e is the element volume, c is the material sound speed, and Q_{hg} is a user-defined constant usually set to a value between .05 and .15.

	$\alpha = 1$	$\alpha = 2$	$\alpha = 3$	$\alpha = 4$
Γ_{j1}	1	1	1	1
Γ_{j2}	-1	1	-1	-1
Γ_{j3}	1	-1	-1	1
Γ_{j4}	-1	-1	1	-1
Γ_{j5}	1	-1	-1	-1
Γ_{j6}	-1	-1	1	1
Γ_{j7}	1	1	1	-1
Γ_{j8}	-1	1	-1	1

Table 3.1. Hourglass base vectors.

	DYNA3D	Flanagan-Belytschko [1981]	Wilkins FDM
Strain displacement matrix	94	357	843
Strain rates	87	156	
Force	117	195	270
Subtotal	298	708	1,113
Hourglass control	130	620	680
Total	428	1,328	1,793

Table 3.2: Operation count for constant stress hexahedron (includes adds, subtracts, multiplies, and divides in major subroutines, and is independent of vectorization). Material subroutines will add as little as 60 operations for the bilinear elastic-plastic routine to ten times as much for multi-surface plasticity and reactive flow models. Unvectorized material models will increase that share of the cost a factor of four or more.

Instead of resisting components of the bilinear velocity field that are orthogonal to the strain calculation, Flanagan and Belytschko resist components of the velocity field that are not part of a fully linear field. They call this field, defined below, the hourglass velocity field

$$\dot{x}_i^{k\,HG} = \dot{x}_i - \dot{x}_i^{k\,LIN} \tag{3.23}$$

where

$$\dot{x}_i^{k\,LIN} = \dot{\bar{x}} + \dot{\bar{x}}_{i,j}\left(x_j^k - \bar{x}_j\right) \tag{3.24}$$

$$\bar{x}_i = \frac{1}{8}\sum_{k=1}^{8} x_i^k \qquad \dot{\bar{x}}_i = \frac{1}{8}\sum_{k=1}^{8} \dot{x}_i^k \tag{3.25}$$

Flanagan and Belytschko construct geometry-dependent hourglass shape vectors that are orthogonal to the fully linear velocity field and the rigid body field. With these vectors they resist the hourglass velocity deformations. Defining hourglass shape vectors in terms of the base vectors as

$$\gamma_{\alpha k} = \Gamma_{\alpha k} - \phi_{k,i}\sum_{n=1}^{8} x_i^n \Gamma_{\alpha n} \tag{3.26}$$

and setting

$$g_{i\alpha} = \sum_{k=1}^{8} \dot{x}_i^{\,k}\gamma_{\alpha k} = 0$$

the 12 resisting force vectors become

$$f_{i\alpha}^{\,k} = a_h g_{i\alpha}\gamma_{\alpha k} \tag{3.27}$$

where a_h is a constant given in Equation (3.21).

These methods are identical if the hexahedron element is a parallelepiped. The default hourglass control method for solid element is given by Equation (3.21); however, we recommend the Flanagan-Belytschko approach for problems that have large rigid body rotations since the default approach is not orthogonal to rigid body rotations.

A cost comparison in Table 3.2 shows that the default hourglass viscosity requires approximately 130 adds or multiplies per hexahedron, compared to 620 and 680 for the algorithms of Flanagan-Belytschko and Wilkins.

3.3 Fully Integrated Brick Elements

To avoid locking in the fully integrated brick elements strain increments at a point in a constant pressure, solid element are defined by [see Nagtegaal, Parks, and Rice 1974]

$$\Delta\varepsilon_{xx} = \frac{\partial \Delta u}{\partial x}^{n+\frac{1}{2}} + \phi \qquad \Delta\varepsilon_{xy} = \frac{\frac{\partial \Delta v}{\partial x}^{n+\frac{1}{2}} + \frac{\partial \Delta u}{\partial y}^{n+\frac{1}{2}}}{2}$$

$$\Delta\varepsilon_{yy} = \frac{\partial \Delta v}{\partial y}^{n+\frac{1}{2}} + \phi \qquad \Delta\varepsilon_{yz} = \frac{\frac{\partial \Delta w}{\partial y}^{n+\frac{1}{2}} + \frac{\partial \Delta v}{\partial z}^{n+\frac{1}{2}}}{2} \tag{3.28}$$

$$\Delta\varepsilon_{zz} = \frac{\partial \Delta w}{\partial z}^{n+\frac{1}{2}} + \phi \qquad \Delta\varepsilon_{zx} = \frac{\frac{\partial \Delta u}{\partial z}^{n+\frac{1}{2}} + \frac{\partial \Delta w}{\partial x}^{n+\frac{1}{2}}}{2}$$

where ϕ modifies the normal strains to ensure that the total volumetric strain increment at each integration point is identical

$$\phi = \Delta\varepsilon_v^{n+\frac{1}{2}} - \frac{\frac{\partial \Delta u}{\partial x}^{n+\frac{1}{2}} + \frac{\partial \Delta v}{\partial y}^{n+\frac{1}{2}} + \frac{\partial \Delta w}{\partial z}^{n+\frac{1}{2}}}{3}$$

and $\Delta\varepsilon_v$ is the average volumetric strain increment

$$\frac{\frac{1}{3}\int_{v^{n+\frac{1}{2}}} \left(\frac{\partial \Delta u}{\partial x}^{n+\frac{1}{2}} + \frac{\partial \Delta v}{\partial y}^{n+\frac{1}{2}} + \frac{\partial \Delta w}{\partial z}^{n+\frac{1}{2}} \right) dv^{n+\frac{1}{2}}}{\int_{v^{n+\frac{1}{2}}} dv^{n+\frac{1}{2}}}, \tag{3.29}$$

Δu, Δv and Δw are displacement increments in the x, y and z directions, respectively, and

$$x^{n+\frac{1}{2}} = \frac{\left(x^n + x^{n+1}\right)}{2}, \tag{3.30a}$$

$$y^{n+\frac{1}{2}} = \frac{\left(y^n + y^{n+1}\right)}{2}, \tag{3.30b}$$

$$z^{n+\frac{1}{2}} = \frac{\left(z^n + z^{n+1}\right)}{2}, \tag{3.30c}$$

To satisfy the condition that rigid body rotations cause zero straining, it is necessary to use the geometry at the midstep. We are currently using the geometry at step n+1 to save operations. Problems should not occur unless the motion in a single step becomes too large, which is not normally expected in explicit computations.

Since the bulk modulus is constant in the plastic and viscoelastic material models, constant pressure solid elements result. In the thermoelasticplastic material, a constant temperature is assumed over the element. In the soil and crushable foam material, an average relative volume is computed for the element at time step n+1, and the pressure and bulk modulus associated with this relative volume is used at each integration point. For equations of state just one pressure evaluation is done per element.

The foregoing procedure requires that the strain-displacement matrix corresponding to Equations (3.28) and consistent with a constant volumetric strain, $\overline{B}$, be used in the nodal force calculations [Hughes 1980]. It is easy to show that:

$$F = \int_{V^{n+1}} \overline{B}^{n+1^t} \sigma^{n+1} dv^{n+1} = \int_{V^{n+1}} B^{n+1^t} \sigma^{n+1} dv^{n+1} \tag{3.31}$$

and avoid the needless complexities of computing $\overline{B}$.

3.4 Fully Integrated Brick Element with 48 degrees-of-freedom

The forty-eight degree of freedom brick element is derived from the twenty node solid element, see Figure 3.3, through a transformation of the nodal displacements and rotations of the mid-side nodes [Yunus, Pawlak, and Cook, 1989]. This element has the advantage that shell nodes can be shared with brick nodes and that the faces have just four

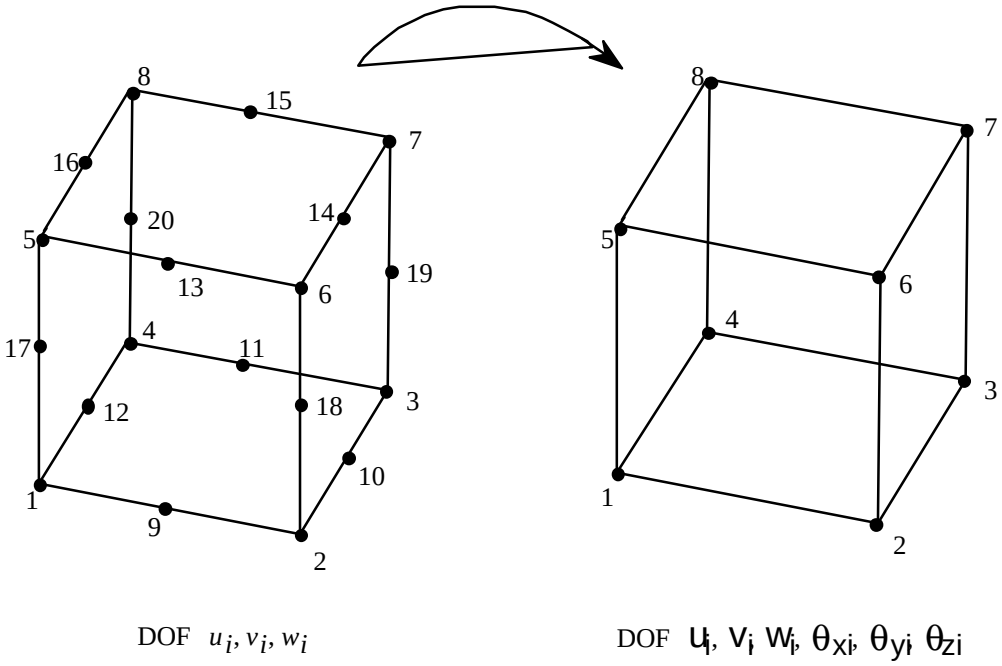


Figure 3.3. The 20-node solid element is transformed to an 8-node solid with 6 degrees-of-freedom (DOF) per node.

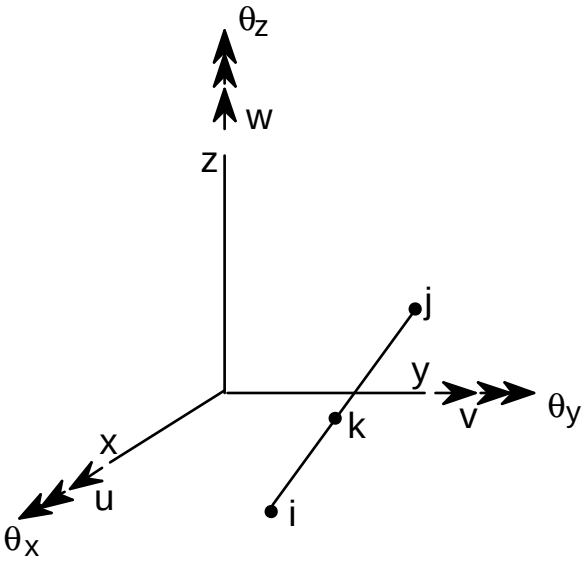


Figure 3.4. A typical element edge from [Yunus, Pawlak, and Cook, 1989] .

nodes-a real advantage for the contact-impact logic. The accuracy of this element is relatively good for problems in linear elasticity but degrades as Poisson's ratio approaches the incompressible limit. This can be remedied by using incompatible modes in the element formulation but such an approach seems impractical for explicit computations.

The instantaneous velocity for a midside node k is given as a function of the corner node velocities as (See Figure 3.4),

$$\dot{u}_k = \frac{1}{2}\left(\dot{u}_i + \dot{u}_j\right) + \frac{y_j - y_i}{8}\left(\dot{\theta}_{zj} - \dot{\theta}_{zi}\right) + \frac{z_j - z_i}{8}\left(\dot{\theta}_{yi} - \dot{\theta}_{yj}\right)$$

$$\dot{v}_k = \frac{1}{2}\left(\dot{v}_i + \dot{v}_j\right) + \frac{z_j - z_i}{8}\left(\dot{\theta}_{xj} - \dot{\theta}_{xi}\right) + \frac{x_j - x_i}{8}\left(\dot{\theta}_{zi} - \dot{\theta}_{zj}\right) \tag{3.32}$$

$$\dot{w}_k = \frac{1}{2}\left(\dot{w}_i + \dot{w}_j\right) + \frac{x_j - x_i}{8}\left(\dot{\theta}_{yj} - \dot{\theta}_{yi}\right) + \frac{y_j - y_i}{8}\left(\dot{\theta}_{xi} - \dot{\theta}_{xj}\right)$$

where u, v, w, θ_x, θ_y, and θ_z are the translational and rotational displacements in the global x, y and z directions. The velocity field for the twenty node hexahedron element in terms of the nodal velocities is:

$$\begin{Bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{Bmatrix} = \begin{bmatrix} \phi_1 & \phi_2 \ldots \phi_{20} & 0 & 0 \ldots 0 & 0 & 0 \ldots 0 \\ 0 & 0 \ldots 0 & \phi_1 & \phi_2 \ldots \phi_{20} & 0 & 0 \ldots 0 \\ 0 & 0 \ldots 0 & 0 & 0 \ldots 0 & \phi_1 & \phi_2 \ldots \phi_{20} \end{bmatrix} \begin{Bmatrix} \dot{u}_1 \\ \vdots \\ \dot{u}_{20} \\ \dot{v}_1 \\ \vdots \\ \dot{v}_{20} \\ \dot{w}_1 \\ \vdots \\ \dot{w}_{20} \end{Bmatrix} \tag{3.33}$$

where ϕ_i are given by [Bathe and Wilson 1976] as,

$$\phi_1 = g_1 - \frac{(g_9 + g_{12} + g_{17})}{2} \qquad \phi_6 = g_6 - \frac{(g_{13} + g_{14} + g_{18})}{2}$$
$$\phi_2 = g_2 - \frac{(g_9 + g_{10} + g_{18})}{2} \qquad \phi_7 = g_7 - \frac{(g_{14} + g_{15} + g_{19})}{2}$$
$$\phi_3 = g_3 - \frac{(g_{10} + g_{11} + g_{19})}{2} \qquad \phi_8 = g_8 - \frac{(g_{15} + g_{16} + g_{20})}{2} \qquad (3.34)$$
$$\phi_4 = g_4 - \frac{(g_{11} + g_{12} + g_{20})}{2} \qquad \phi_i = g_i \text{ for } j = 9,...20$$
$$\phi_5 = g_5 - \frac{(g_{13} + g_{16} + g_{17})}{2}$$

$$g_i = G(\xi, \xi_i)\, G(\eta, \eta_i)\, G(\zeta, \zeta_i)$$

$$G(\beta, \beta_i) = \frac{1}{2}(1 + \beta\beta i) \quad \text{for } \beta_i = \pm 1 \;\; ; \;\; \beta = \xi, \eta, \zeta$$

$$G(\beta, \beta_i) = 1 - \beta^2 \quad \text{for } \beta_i = 0$$

The standard formulation for the twenty node solid element is used with the above transformations. The element is integrated with a fourteen point integration rule [Cook 1974]:

$$\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} f(\xi,\eta,\zeta)\,d\xi\,d\eta\,d\zeta =$$

$$B_6\left[f(-b,0,0) + f(b,0,0) + f(0,-b,0) + ...(6\,terms)\right] + \qquad (3.35)$$

$$C_8\left[f(-c,-c,-c) + f(c,-c,-c) + f(c,c,-c) + ...(8\,terms)\right]$$

where

$B_6 = 0.8864265927977938$ $\qquad$ $b = 0.7958224257542215$

$C_8 = 0.3351800554016621$ $\qquad$ $c = 0.7587869106393281$

Cook reports that this rule has nearly the same accuracy as the twenty-seven point Gauss rule, which is very costly. The difference in cost between eight point and fourteen point integration, though significant, is necessary to eliminate the zero energy modes.

3.5 Fully Integrated Tetrahedron Element with 24 degrees-of-freedom

Automatic mesh generators often use tetrahedron element extensively and have therefore motivated the addition of this element. The twenty-four degree of freedom tetrahedron element is derived from the ten node tetrahedron element, see Figure 3.5, following the same procedure used above for the forty-eight degree of freedom brick element [Yunus, Pawlak, and Cook, 1989]. This element has the advantage that shell nodes can be shared with its nodes and it is compatible with the brick element discussed above. The accuracy of this element is relatively good-at least when compared to the constant strain tetrahedron element.

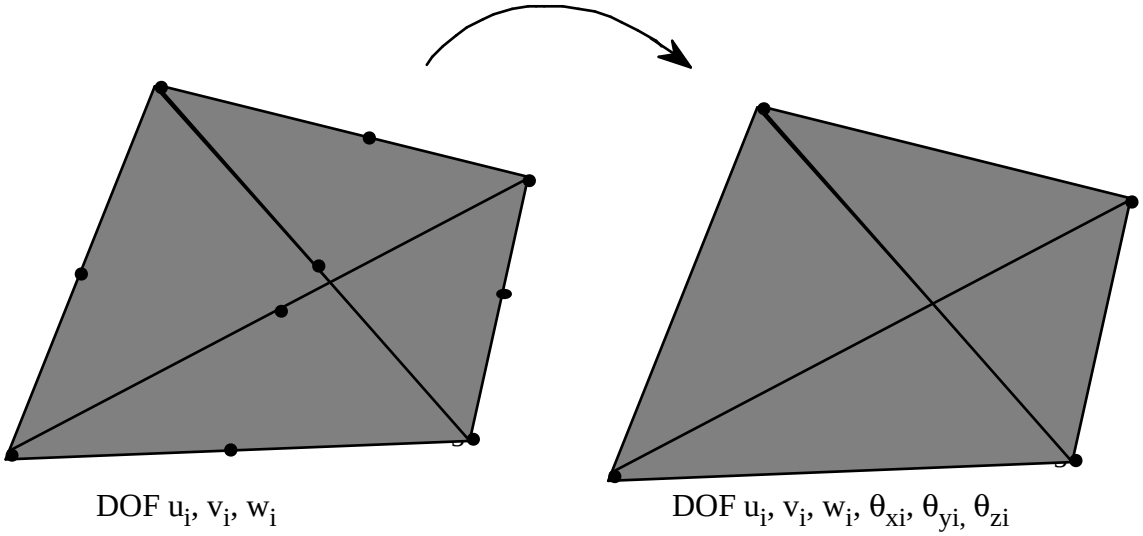


Figure 3.5. Twenty-four degree of freedom tetrahedron element [Yunus, Pawlak, and Cook, 1989].

In our implementation we have not strictly followed the reference. In order to prevent locking in applications that involve incompressible behavior, selective reduced integration is used with a total of 5 integration points. Although this is rather expensive, no zero energy modes exists. We use the same approach in determining the rotary mass that is used in the implementation of the shell elements.

Figures 3.6 and 3.7 show the construction of a hexahedron element from five and six tetrahedron elements, respectively. When two sides of the adjacent bricks made from five tetrahedron are together, it is likely that four unique triangular segments exists. This

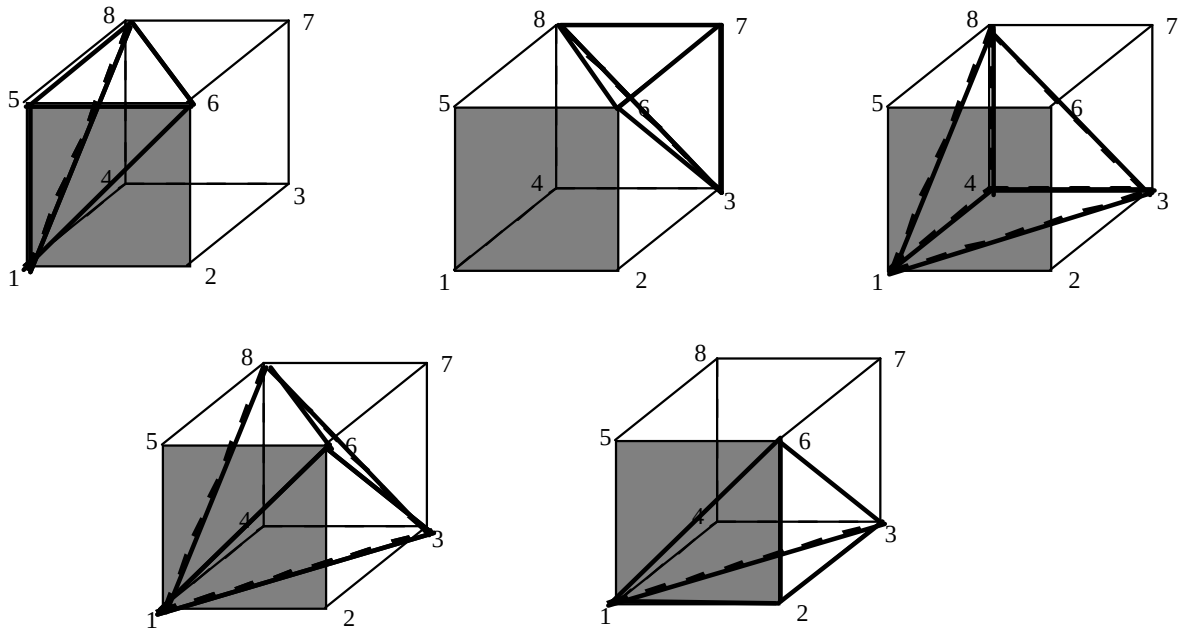


Figure 3.6. Construction of a hexahedron element with five tetrahedrons.

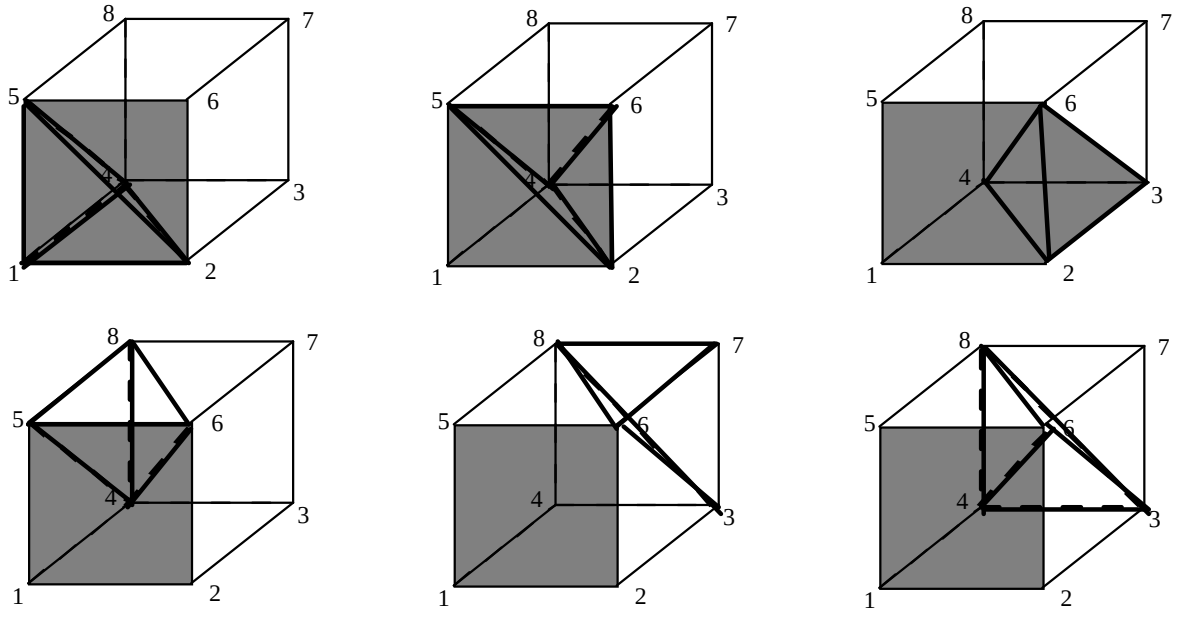


Figure 3.7. Construction of a hexahedron element with six tetrahedrons.

creates a problem in TAURUS which uses the numbering as a basis for eliminating interior polygons prior to display. Consequently, the graphics in the post-processing phase can be considerably slower with the degeneration in Figure 3.6. However, marginally better results may be obtained with five tetrahedrons per hexahedron due to a better constraint count.

4. BELYTSCHKO BEAM

The Belytschko beam element formulation [Belytschko et al.1977] is part of a family of structural finite elements, by Belytschko and other researchers, that employ a 'co-rotational technique' in the element formulation for treating large rotation. This section discusses the co-rotational formulation, since the formulation is most easily described for a beam element, and then describes the beam theory used to formulate the co-rotational beam element.

4.1 Co-rotational Technique

In any large displacement formulation, the goal is to separate the deformation displacements from the rigid body displacements, as only the deformation displacements give rise to strains and the associated generation of strain energy. This separation is usually accomplished by comparing the current configuration with a reference configuration.

The current configuration is a complete description of the deformed body in its current spatial location and orientation, giving locations of all points (nodes) comprising the body. The reference configuration can be either the initial configuration of the body, i.e., nodal locations at time zero, or the configuration of the body at some other state (time). Often the reference configuration is chosen to be the previous configuration, say at time $t^n = t^{n+1} - \Delta t$.

The choice of the reference configuration determines the type of deformations that will be computed: total deformations result from comparing the current configuration with the initial configuration, while incremental deformations result from comparing with the previous configuration. In most time stepping (numerical) Lagrangian formulations, incremental deformations are used because they result in significant simplifications of other algorithms, chiefly constitutive models.

A direct comparison of the current configuration with the reference configuration does not result in a determination of the deformation, but rather provides the total (or incremental) displacements. We will use the unqualified term displacements to mean either the total displacements or the incremental displacements, depending on the choice of the reference configuration as the initial or the last state. This is perhaps most obvious if the reference configuration is the initial configuration. The direct comparison of the current configuration with the reference configuration yields displacements which contain components due to deformations and rigid body motions. The task remains of separating the deformation and rigid body displacements. The deformations are usually found by

subtracting from the displacements an estimate of the rigid body displacements. Exact rigid body displacements are usually only known for trivial cases where they are prescribed a priori as part of a displacement field. The co-rotational formulations provides one such estimate of the rigid body displacements.

The co-rotational formulation uses two types of coordinate systems: one system associated with each element, i.e., element coordinates which deform with the element, and another associated with each node, i.e., body coordinates embedded in the nodes. (The term 'body' is used to avoid possible confusion from referring to these coordinates as 'nodal' coordinates. Also, in the more general formulation presented in [Belytschko et al. 1977], the nodes could optionally be attached to rigid bodies. Thus the term 'body coordinates' refers to a system of coordinates in a rigid body, of which a node is a special case.) These two coordinate systems are shown in the upper portion of Figure 4.1(a).

The element coordinate system is defined to have the local x-axis $\hat{x}$ originating at node *I* and terminating at node *J*; the local y-axis $\hat{y}$ and, in three dimension, the local z-axis $\hat{z}$, are constructed normal to $\hat{x}$. The element coordinate system $(\hat{x}, \hat{y}, \hat{z})$ and associated unit vector triad ($\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$) are updated at every time step by the same technique used to construct the initial system; thus the unit vector $\mathbf{e}_1$ deforms with the element since it always points from node I to node J.

The embedded body coordinate system is initially oriented along the principal inertial axes; either the assembled nodal mass or associated rigid body inertial tensor are used in determining the inertial principal values and directions. Although the initial orientation of the body axes is arbitrary, the selection of a principal inertia coordinate system simplifies the rotational equations of motion, i.e., no inertial cross product terms are present in the rotational equations of motion. Because the body coordinates are fixed in the node, or rigid body, they rotate and translate with the node and are updated by integrating the rotational equations of motion, as will be described subsequently.

The unit vectors of the two coordinate systems define rotational transformations between the global coordinate system and each respective coordinate system. These transformations operate on vectors with global components $\mathbf{A} = (A_x, A_y, A_z)$, body coordinate components $\overline{A} = \left(\overline{A}_x, \overline{A}_y, \overline{A}_z\right)$, and element coordinate components $\hat{A} = \left(\hat{A}_x, \hat{A}_y, \hat{A}_z\right)$ which are defined as:

$$A = \begin{Bmatrix} A_x \\ A_y \\ A_z \end{Bmatrix} = \begin{bmatrix} b_{1x} & b_{2x} & b_{3x} \\ b_{1y} & b_{2y} & b_{3y} \\ b_{1z} & b_{2z} & b_{3z} \end{bmatrix} \begin{Bmatrix} \overline{A}_x \\ \overline{A}_y \\ \overline{A}_z \end{Bmatrix} = [\lambda]\{\overline{A}\} \tag{4.1}$$

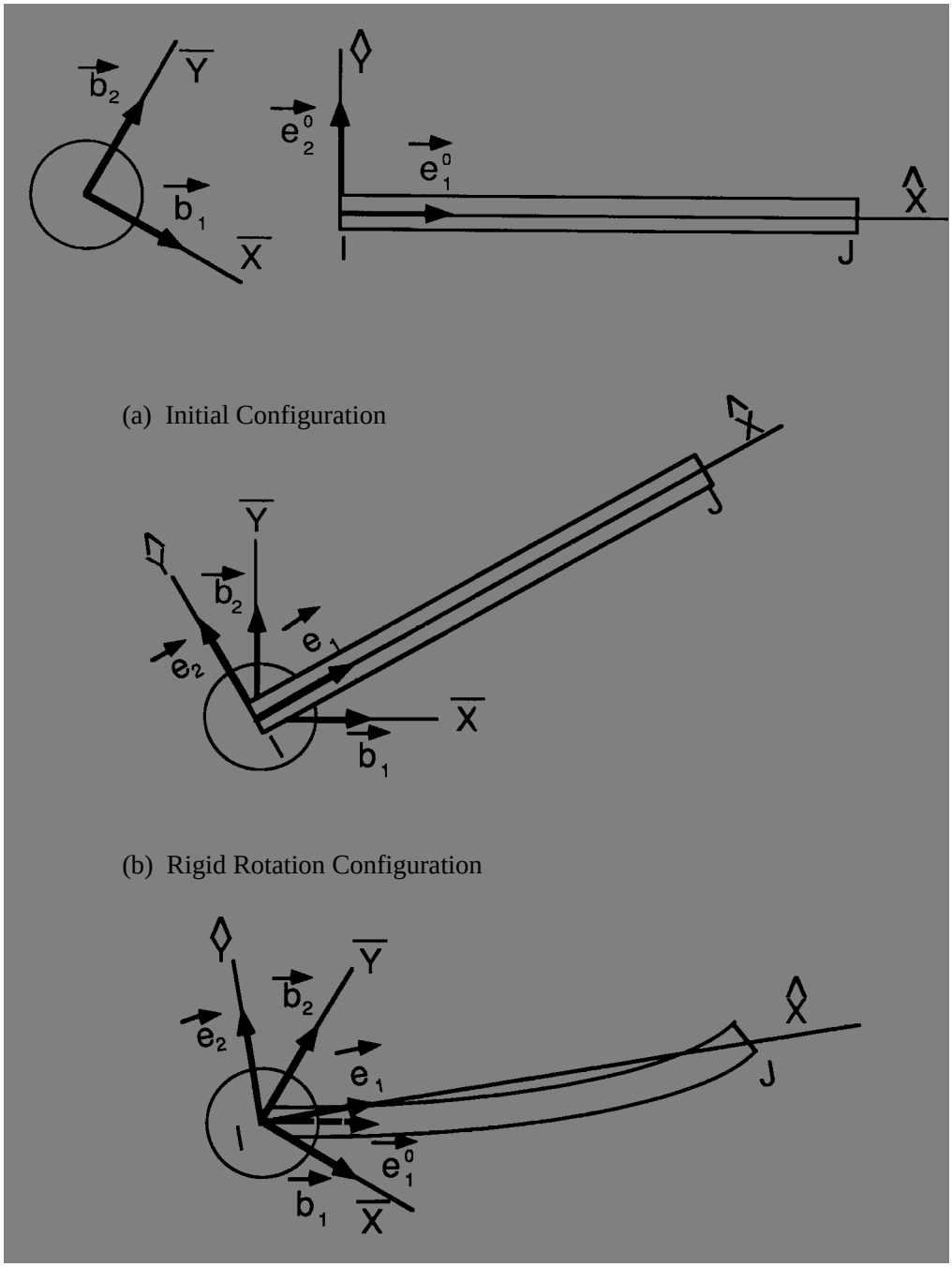


Figure 4.1. Co-rotational coordinate system: (a) initial configuration, (b) rigid rotational configuration and (c) deformed configuration.

where b_{ix}, b_{iy}, b_{iz} are the global components of the body coordinate unit vectors. Similarly for the element coordinate system:

$$A = \begin{Bmatrix} A_x \\ A_y \\ A_z \end{Bmatrix} = \begin{bmatrix} e_{1x} & e_{2x} & e_{3x} \\ e_{1y} & e_{2y} & e_{3y} \\ e_{1z} & e_{2z} & e_{3z} \end{bmatrix} \begin{Bmatrix} \hat{A}_x \\ \hat{A}_y \\ \hat{A}_z \end{Bmatrix} = [\mu]\{\hat{A}\} \tag{4.2}$$

where e_{ix}, e_{iy}, e_{iz} are the global components of the element coordinate unit vectors. The inverse transformations are defined by the matrix transpose, i.e.:

$$\{\bar{A}\} = [\lambda]^T \{A\} \tag{4.3}$$

$$\{\hat{A}\} = [\mu]^T \{A\} \tag{4.4}$$

since these are proper rotational transformations.

The following two examples illustrate how the element and body coordinate system are used to separate the deformations and rigid body displacements from the displacements:

Rigid Rotation. First consider a rigid body rotation of the beam element about node I, as shown in the center of Figure 4.1(b), i.e., consider node I to be a pinned connection. Because the beam does not deform during the rigid rotation, the orientation of the unit vector $\mathbf{e}_1$ in the initial and rotated configuration will be the same with respect to the body coordinates. If the body coordinate components of the initial element unit vector e_1^0 were stored, they would be identical to the body coordinate components of the current element unit vector $\mathbf{e}_1$.

Deformation Rotation. Next consider node I to be constrained against rotation, i.e., a clamped connection. Now node J is moved, as shown in the lower portion of Figure 4.1(c), causing the beam element to deform. The updated element unit vector $\mathbf{e}_1$ is constructed and its body coordinate components are compared to the body coordinate components of the original element unit vector e_1^0. Because the body coordinate system did not rotate, as node I was constrained, the original element unit vector and the current element unit vector are not collinear. Indeed, the angle between these two unit vectors is the amount of rotational deformation at node I i.e.

$$e_1 \times e_1^0 = \theta_\ell e_3 \tag{4.5}$$

Thus the co-rotational formulation separates the deformation and rigid body deformations by using:

- a coordinate system that deforms with the element, i.e., the element coordinates; or
- a coordinate system that rigidly rotates with the nodes, i.e., the body coordinates;

Then it compares the current orientation of the element coordinate system with the initial element coordinate system, using the rigidly rotated body coordinate system, to determine the deformations.

4.2 Belytschko Beam Element Formulation

The deformation displacements used in the Belytschko beam element formulation are:

$$\hat{d}^T = \left\{ \delta_{IJ}, \hat{\theta}_{xJI}, \hat{\theta}_{yI}, \hat{\theta}_{yJ}, \hat{\theta}_{zI}, \hat{\theta}_{zJ} \right\} \tag{4.6}$$

where

$$\begin{aligned} \delta_{IJ} &= \text{length change} \\ \hat{\theta}_{xJI} &= \text{torsional deformation} \\ \hat{\theta}_{yI}, \hat{\theta}_{yJ}, \hat{\theta}_{zI}, \hat{\theta}_{zJ} &= \text{bending rotational deformations} \end{aligned}$$

The superscript ^ emphasizes that these quantities are defined in the local element coordinate system, and I and J are the nodes at the ends of the beam.

The beam deformations, defined above in Equation (4.6), are the usual small displacement beam deformations (see, for example, [Przemieniecki 1986]). Indeed, one advantage of the co-rotational formulation is the ease with which existing small displacement element formulations can be adapted to a large displacement formulation having small deformations in the element system. Small deformation theories can be easily accommodated because the definition of the local element coordinate system is independent of rigid body rotations and hence deformation displacement can be defined directly.

4.2.1 Calculation of Deformations

The elongation of the beam is calculated directly from the original nodal coordinates (X_I, Y_I, Z_I) and the total displacements (u_{xI}, u_{yI}, u_{zI}):

$$\delta_{IJ} = \frac{1}{l + l^o}\left[2\left(X_{JI}u_{xJI} + Y_{JI}u_{yJI} + Z_{JI}u_{zJI}\right) + u_{xJI}^2 + u_{yJI}^2 + u_{zJI}^2\right] \tag{4.7}$$

where

$$X_{JI} = X_J - X_I \tag{4.8}$$

$$u_{xJI} = u_{xJ} - u_{xI} \text{ etc.} \tag{4.9}$$

The deformation rotations are calculated using the body coordinate components of the original element coordinate unit vector along the beam axis, i.e., e_1^0 , as outlined in the previous section. Because the body coordinate components of initial unit vector e_1^0 rotate with the node, in the deformed configuration it indicates the direction of the beam's axis if no deformations had occurred. Thus comparing the initial unit vector e_1^0 with its current orientation $\mathbf{e}_1$ indicates the magnitude of deformation rotations. Forming the vector cross product between e_1^0 and $\mathbf{e}_1$:

$$e_1 \times e_1^0 = \hat{\theta}_y e_2 + \hat{\theta}_z e_3 \tag{4.10}$$

where

$\hat{\theta}_y$ is the incremental deformation about the local $\hat{y}$ axis

$\hat{\theta}_z$ is the incremental deformation about the local $\hat{z}$ axis

The calculation is most conveniently performed by transforming the body components of the initial element vector into the current element coordinate system:

$$\begin{Bmatrix} \hat{e}_{1x}^0 \\ \hat{e}_{1y}^0 \\ \hat{e}_{1z}^0 \end{Bmatrix} = [\mu]^T [\lambda] \begin{Bmatrix} \bar{e}_{1x}^0 \\ \bar{e}_{1y}^0 \\ \bar{e}_{1z}^0 \end{Bmatrix} \tag{4.11}$$

Substituting the above into Equation (4.10)

$$e_1 \times e_1^0 = \det\begin{bmatrix} e_1 & e_2 & e_3 \\ 1 & 0 & 0 \\ \hat{e}_{1x}^0 & \hat{e}_{1y}^0 & \hat{e}_{1z}^0 \end{bmatrix} = -\hat{e}_{1z}^0 e_2 + \hat{e}_{1y}^0 e_3 = \hat{\theta}_y e_2 + \hat{\theta}_z e_3 \tag{4.12}$$

Thus

$$\hat{\theta}_y = -\hat{e}_{1z}^0 \tag{4.13}$$

$$\hat{\theta}_z = \hat{e}_{1y}^0 \tag{4.14}$$

The torsional deformation rotation is calculated from the vector cross product of initial unit vectors, from each node of the beam, that were normal to the axis of the beam, i.e., $\hat{e}_{2I}^0$ and $\hat{e}_{2J}^0$; note that $\hat{e}_{3I}^0$ and $\hat{e}_{3J}^0$ could also be used. The result from this vector cross product is then projected onto the current axis of the beam, i.e.,

$$\hat{\theta}_{xJI} = e_1 \cdot \left(\hat{e}_{2I}^0 \times \hat{e}_{2J}^0\right) = e_1 \det\begin{bmatrix} e_1 & e_2 & e_3 \\ \hat{e}_{x2I}^0 & \hat{e}_{y2I}^0 & \hat{e}_{z2I}^0 \\ \hat{e}_{x2J}^0 & \hat{e}_{y2J}^0 & \hat{e}_{z2J}^0 \end{bmatrix} = \hat{e}_{y2I}^0 \hat{e}_{z2J}^0 - \hat{e}_{y2J}^0 \hat{e}_{z2I}^0 \tag{4.15}$$

Note that the body components of $\bar{e}_{2I}^0$ and $\bar{e}_{2J}^0$ are transformed into the current element coordinate system before performing the indicated vector products.

4.2.2 Calculation of Internal Forces

There are two methods for computing the internal forces for the Belytschko beam element formulation:

1. functional forms relating the overall response of the beam, e.g., moment-curvature relations,
2. direct through-the-thickness integration of the stress.

Currently only the former method, as explained subsequently, is implemented; the direct integration method is detailed in [Belytschko et al.1977].

Axial Force. The internal axial force is calculated from the elongation of the beam δ, as given by Equation (4.7), and an axial stiffness:

$$\hat{f}_{xJ} = K^a \delta \tag{4.16}$$

where

$K^a = AE/\ell^0$ is the axial stiffness
A is the cross sectional area of the beam
E is Young's Modulus
ℓ^0 is the original length of the beam

Bending Moments. The bending moments are related to the deformation rotations by

$$\begin{Bmatrix} \hat{m}_{yI} \\ \hat{m}_{yJ} \end{Bmatrix} = \frac{K_y^b}{1+\phi_y} \begin{bmatrix} 4+\phi_y & 2-\phi_y \\ 2-\phi_y & 4+\phi_y \end{bmatrix} \begin{Bmatrix} \hat{\theta}_{yI} \\ \hat{\theta}_{yJ} \end{Bmatrix} \tag{4.17}$$

$$\begin{Bmatrix} \hat{m}_{zI} \\ \hat{m}_{zJ} \end{Bmatrix} = \frac{K_z^b}{1+\phi_z} \begin{bmatrix} 4+\phi_z & 2-\phi_z \\ 2-\phi_z & 4+\phi_z \end{bmatrix} \begin{Bmatrix} \hat{\theta}_{zI} \\ \hat{\theta}_{zJ} \end{Bmatrix} \tag{4.18}$$

where Equation (4.17) is for bending in the $\hat{x}$–$\hat{z}$ plane and Equation (4.18) is for bending in the $\hat{x}$–$\hat{y}$ plane. The bending constants are given by

$$K_y^b = \frac{EI_{yy}}{l^0} \tag{4.19}$$

$$K_z^b = \frac{EI_{zz}}{l^0} \tag{4.20}$$

$$I_{yy} = \int\int \hat{z}^2 \, d\hat{y} d\hat{z} \tag{4.21}$$

$$I_{zz} = \int\int \hat{y}^2 \, d\hat{y} d\hat{z} \tag{4.22}$$

$$\phi_y = \frac{12EI_{yy}}{GA_s l^2} \qquad \phi_z = \frac{12EI_{zz}}{GA_s l^2} \tag{4.23}$$

Hence ϕ is the shear factor, G the shear modulus, and A_s is the effective area in shear.

Torsional Moment. The torsional moment is calculated from the torsional deformation rotation as

$$\hat{m}_{xJ} = K^t \hat{\theta}_{xJI} \tag{4.24}$$

where

$$K^t = \frac{GJ}{l^0} \tag{4.25}$$

$$J = \int\int \hat{y}\hat{z}\ d\hat{y}d\hat{z} \tag{4.26}$$

The above forces are conjugate to the deformation displacements given previously in Equation (4.6), i.e.

$$\hat{d}^T = \left\{\delta_{IJ}, \hat{\theta}_{xJI}, \hat{\theta}_{yI}, \hat{\theta}_{yJ}, \hat{\theta}_{zI}, \hat{\theta}_{zJ}\right\} \tag{4.6}$$

where

$$\left\{\hat{d}\right\}^T \left\{\hat{f}\right\} = W^{\text{int}} \tag{4.27}$$

$$\hat{f}^T = \left\{\hat{f}_{xJ}, \hat{m}_{xJ}, \hat{m}_{yI}, \hat{m}_{yJ}, \hat{m}_{zI}, \hat{m}_{zJ}\right\} \tag{4.28}$$

The remaining internal force components are found from equilibrium:

$$\begin{aligned} \hat{f}_{xI} &= -\hat{f}_{xJ} & \hat{m}_{xI} &= -\hat{m}_{xJ} \\ \hat{f}_{zI} &= -\frac{\hat{m}_{yI} + \hat{m}_{yJ}}{l^0} & \hat{f}_{zI} &= -\hat{f}_{zJ} \\ \hat{f}_{yJ} &= -\frac{\hat{m}_{zI} + \hat{m}_{zJ}}{l^0} & \hat{f}_{yI} &= -\hat{f}_{yJ} \end{aligned} \tag{4.29}$$

4.2.3 Updating the Body Coordinate Unit Vectors

The body coordinate unit vectors are updated using the Newmark β-Method [Newmark 1959] with β = 0, which is almost identical to the central difference method [Belytschko 1974]. In particular, the body component unit vectors are updated using the formula

$$b_i^{j+1} = b_i^j + \Delta t \frac{db_i^j}{dt} + \frac{\Delta t^2}{2} \frac{d^2 b_i^j}{dt^2} \tag{4.30}$$

where the superscripts refer to the time step and the subscripts refer to the three unit vectors comprising the body coordinate triad. The time derivatives in the above equation are replaced by their equivalent forms from vector analysis:

$$\frac{db_i^j}{dt} = \omega \times b_i \tag{4.31}$$

$$\frac{d^2 b_i^j}{dt^2} = \omega \times (\omega \times b_i) + (\alpha_i \times b_i) \tag{4.32}$$

where ω and α are vectors of angular velocity and acceleration, respectively, obtained from the rotational equations of motion. With the above relations substituted into Equation (4.30), the update formula for the unit vectors becomes

$$b_i^{j+1} = b_i^j + \Delta t(\omega \times b_i) + \frac{\Delta t^2}{2}\{[\omega \times (\omega \times b_i) + (\alpha_i \times b_i)]\} \tag{4.33}$$

To obtain the formulation for the updated components of the unit vectors, the body coordinate system is temporarily considered to be fixed and then the dot product of Equation (4.33) is formed with the unit vector to be updated. For example, to update the $\bar{x}$ component of $\mathbf{b}_3$, the dot product of Equation (4.33), with i = 3, is formed with $\mathbf{b}_1$, which can be simplified to the relation

$$\bar{b}_{x3}^{j+1} = b_1^j \cdot b_{x3}^{j+1} = \Delta t \omega_y^j + \frac{\Delta t^2}{2}\left(\omega_x^j \omega_z^j + \alpha_y^j\right) \tag{4.34}$$

Similarly

$$\bar{b}_{y3}^{j+1} = b_2^j \cdot b_3^{j+1} = \Delta t \omega_x^j + \frac{\Delta t^2}{2}\left(\omega_y^j \omega_z^j + \alpha_x^j\right) \tag{4.35}$$

$$\bar{b}_{y1}^{j+1} = b_1^j \cdot b_2^{j+1} = \Delta t \omega_z^j + \frac{\Delta t^2}{2}\left(\omega_x^j \omega_y^j + \alpha_z^j\right) \tag{4.36}$$

The remaining components b_3^{j+1} and b_1^{j+1} are found by using normality and orthogonality, where it is assumed that the angular velocities w are small during a time step so that the

quadratic terms in the update relations can be ignored. Since b_3^{j+1} is a unit vector, normality provides the relation

$$\bar{b}_{z3}^{j+1} = \sqrt{1-\left(\bar{b}_{x3}^{j+1}\right)^2 - \left(\bar{b}_{y3}^{j+1}\right)^2} \tag{4.37}$$

Next, if it is assumed that $\bar{b}_{x1}^{j+1} \approx 1$, orthogonality yields

$$\bar{b}_{z1}^{j+1} = -\frac{\bar{b}_{x3}^{j+1} + \bar{b}_{y1}^{j+1}\bar{b}_{y3}^{j+1}}{\bar{b}_{z3}^{j+1}} \tag{4.38}$$

The component $\bar{b}_{x1}^{j+1}$ is then found by enforcing normality:

$$\bar{b}_{x1}^{j+1} = \sqrt{1-\left(\bar{b}_{y1}^{j+1}\right)^2 - \left(\bar{b}_{z1}^{j+1}\right)^2} \tag{4.39}$$

The updated components of $\mathbf{b}_1$ and $\mathbf{b}_3$ are defined relative to the body coordinates at time step j. To complete the update and define the transformation matrix, Equation (4.1), at time step j+1, the updated unit vectors $\mathbf{b}_1$ and $\mathbf{b}_3$ are transformed to the global coordinate system, using Equation (4.1) with $[\lambda]$ defined at step j, and their vector cross product is used to form $\mathbf{b}_2$.

5. HUGHES-LIU BEAM

The Hughes-Liu beam element formulation, based on the shell [Hughes and Liu 1981a, 1981b] discussed later, was the first beam element we implemented. It has several desirable qualities:

- it is incrementally objective (rigid body rotations do not generate strains), allowing for the treatment of finite strains that occur in many practical applications;
- it is simple, which usually translates into computational efficiency and robustness;
- it is compatible with the brick elements, because the element is based on a degenerated brick element formulation;
- it includes finite transverse shear strains. The added computations needed to retain this strain component, compare to those for the assumption of no transverse shear strain, are insignificant.

5.1 Geometry

The Hughes-Liu beam element is based on a degeneration of the isoparametric 8-node solid element, an approach originated by Ahmad et al.[1970]. Recall the solid element isoparametric mapping of the biunit cube

$$x(\xi,\eta,\zeta) = N_a(\xi,\eta,\zeta)x_a \tag{5.1}$$

$$N_a(\xi,\eta,\zeta) = \frac{(1+\xi_a\xi)(1+\eta_a\eta)(1+\zeta_a\zeta)}{8} \tag{5.2}$$

where $\mathbf{x}$ is an arbitrary point in the element, (ξ, η, ζ) are the parametric coordinates, $\mathbf{x}_a$ are the global nodal coordinates of node a, and N_a are the element shape functions evaluated at node a, i.e., (ξ_a, η_a, ζ_a) are (ξ, η, ζ) evaluated at node a.

In the beam geometry, ξ determines the location along the axis of the beam and the coordinate pair (η,ζ) defines a point on the cross section. To degenerate the 8-node brick geometry into the 2-node beam geometry, the four nodes at $\xi = -1$ and at $\xi = 1$ are combined into a single node with three translational and three rotational degrees of freedom. Orthogonal, inextensible nodal fibers are defined at each node for treating the rotational degrees of freedom. Figure 5.1 shows a schematic of the biunit cube and the beam element. The mapping of the biunit cube into the beam element is separated into three parts:

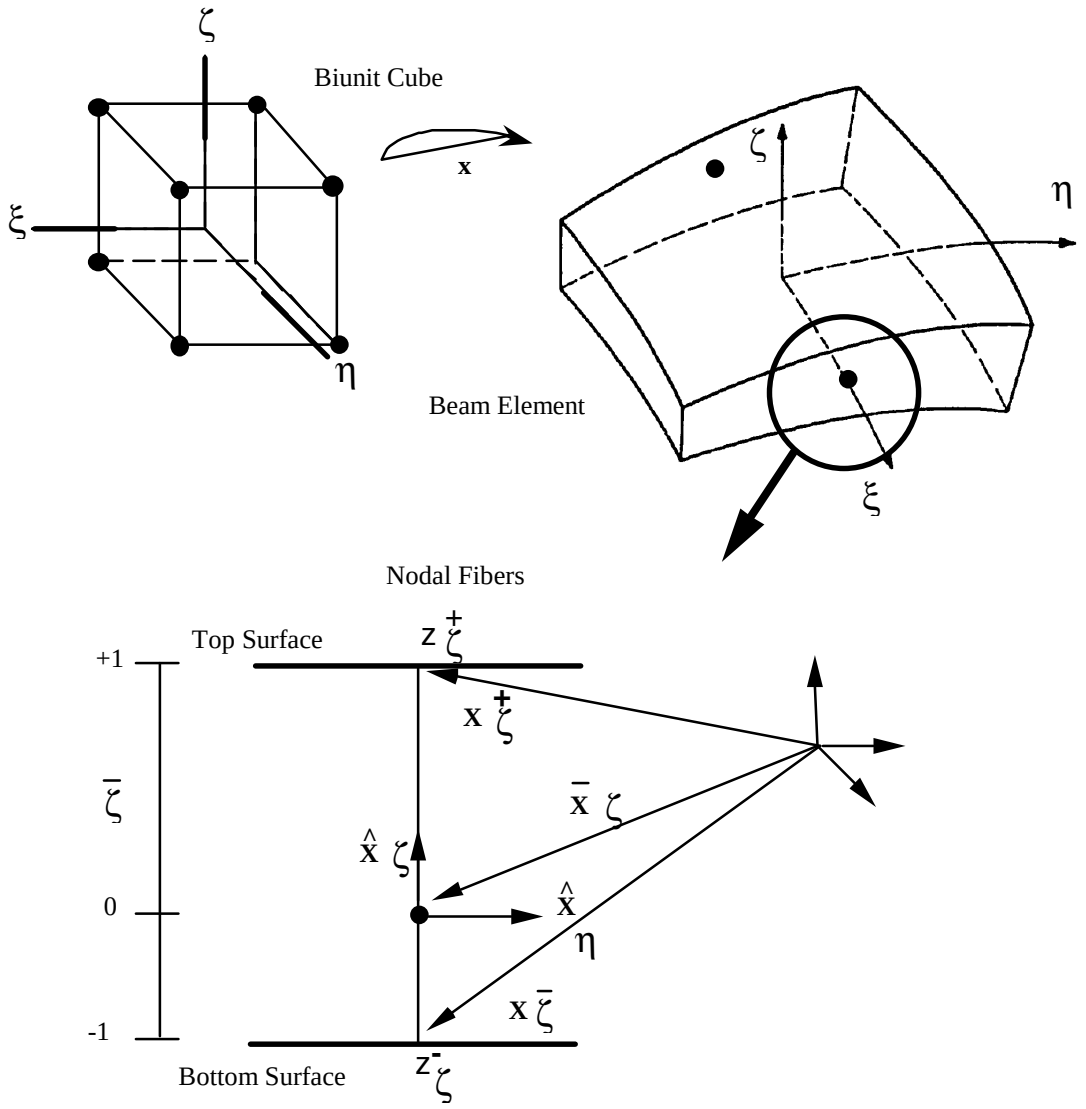


Figure 5.1 Hughes-Liu beam element.

$$x(\xi,\eta,\zeta) = \bar{x}(\xi) + X(\xi,\eta,\zeta) = \bar{x}(\xi) + X_\zeta(\xi,\zeta) + X_\eta(\xi,\eta) \tag{5.3}$$

where $\bar{x}$ denotes a position vector to a point on the reference axis of the beam, and X_ζ and X_η are position vectors at point $\bar{x}$ on the axis that define the fiber directions through that point. In particular,

$$\bar{x}(\xi) = N_a(\xi)\ \bar{x}_a \tag{5.4}$$

$$X_\eta(\xi,\eta) = N_a(\xi)\, X_{\eta a}(\eta) \tag{5.5}$$

$$X_\zeta(\xi,\zeta) = N_a(\xi)\, X_{\zeta a}(\zeta) \tag{5.6}$$

With this description, arbitrary points on the reference line $\bar{x}$ are interpolated by the one-dimensional shape function N(ξ) operating on the global position of the two beam nodes that define the reference axis i.e., $\bar{x}_a$. Points off the reference axis are further interpolated by using a one-dimensional shape function along the fiber directions, i.e., $X_{\eta a}(\eta)$ and $X_{\zeta a}(\zeta)$ where

$$X_{\eta a}(\eta) = z_\eta(\eta)\hat{X}_{\eta a} \tag{5.7a}$$

$$z_\eta(\eta) = N_+(\eta) z_{\eta a}^+ + N_-(\eta) z_{\eta a}^- \tag{5.7b}$$

$$N_+(\eta) = \frac{(1+\eta)}{2} \tag{5.7c}$$

$$N_-(\eta) = \frac{(1-\eta)}{2} \tag{5.7d}$$

$$X_{\zeta a}(\zeta) = z_\zeta(\zeta)\hat{X}_{\zeta a} \tag{5.8a}$$

$$z_\zeta(\zeta) = N_+(\zeta) z_{\zeta a}^+ + N_-(\zeta) z_{\zeta a}^- \tag{5.8b}$$

$$N_+(\zeta) = \frac{(1+\zeta)}{2} \tag{5.8c}$$

$$N_-(\zeta) = \frac{(1-\zeta)}{2} \tag{5.8d}$$

where $z_\zeta(\zeta)$ and $z_\eta(\eta)$ are "thickness functions".

The Hughes-Liu beam formulation uses four position vectors, in addition to ξ, to locate the reference axis and define the initial fiber directions. Consider the two position vectors $x_{\zeta a}^\top$ and $x_{\zeta a}$ located on the top and bottom surfaces, respectively, at node a. Then

$$\bar{x}_{\zeta a} = \frac{1}{2}\left(1-\bar{\zeta}\right)x_{\zeta a}^- + \left(1+\bar{\zeta}\right)x_{\zeta a}^+ \tag{5.9a}$$

$$\hat{X}_{\zeta a} = \frac{\left(x_{\zeta a}^+ - x_{\zeta a}^-\right)}{\left\| x_{\zeta a}^+ - x_{\zeta a}^- \right\|} \tag{5.9b}$$

$$z_{\zeta a}^+ = \frac{1}{2}\left(1-\bar{\zeta}\right)\cdot\left\| x_{\zeta a}^+ - x_{\zeta a}^- \right\| \tag{5.9c}$$

$$z_{\zeta a}^{-} = -\tfrac{1}{2}\left(1+\bar{\zeta}\right)\cdot\left\|x_{\zeta a}^{+} - x_{\zeta a}^{-}\right\| \tag{5.9d}$$

$$\bar{x}_{\eta a} = \frac{1}{2}\left(1-\bar{\zeta}\right)x_{\eta a}^{-} + \left(1+\bar{\zeta}\right)x_{\eta a}^{+} \tag{5.10a}$$

$$\hat{X}_{\eta a} = \frac{\left(x_{\eta a}^{+} - x_{\eta a}^{-}\right)}{\left\|x_{\eta a}^{+} - x_{\eta a}^{-}\right\|} \tag{5.10b}$$

$$z_{\eta a}^{+} = \frac{1}{2}\left(1-\bar{\eta}\right)\cdot\left\|x_{\eta a}^{+} - x_{\eta a}^{-}\right\| \tag{5.10c}$$

$$z_{\eta a}^{-} = -\frac{1}{2}\left(1+\bar{\eta}\right)\cdot\left\|x_{\eta a}^{+} - x_{\eta a}^{-}\right\| \tag{5.10d}$$

where $\|\cdot\|$ is the Euclidean norm. The reference surface may be located at the midsurface of the beam or offset at the outer surfaces. This capability is useful in several practical situations involving contact surfaces, connection of beam elements to solid elements, and offsetting elements such as for beam stiffeners in stiffened shells. The reference surfaces are located within the beam element by specifying the value of the parameters $\bar{\eta}$ and $\bar{\zeta}$, (see lower portion of Figure 5.1). When these parameters take on the values –1 or +1, the reference axis is located on the outer surfaces of the beam. If they are set to zero, the reference axis is at the center.

The same parametric representation used to describe the geometry of the beam elements is used to interpolate the beam element displacements, i.e., an isoparametric representation. Again the displacements are separated into the reference axis displacements and rotations associated with the fiber directions:

$$u(\xi,\eta,\zeta) = \bar{u}(\xi) + U(\xi,\eta,\zeta) = \bar{u}(\xi) + U_{\zeta}(\xi,\zeta) + U_{\eta}(\xi,\eta) \tag{5.11a}$$

$$\bar{u}(\xi) = N_a(\xi)\bar{u}_a \tag{5.11b}$$

$$U_{\eta}(\xi,\eta) = N_a(\xi)U_{\eta a}(\eta) \tag{5.11c}$$

$$U_{\zeta}(\xi,\zeta) = N_a(\xi)U_{\zeta a}(\zeta) \tag{5.11d}$$

$$U_{\eta a}(\eta) = z_{\eta a}(\eta)\hat{U}_{\eta a} \tag{5.11e}$$

$$U_{\zeta a}(\zeta) = z_{\zeta a}(\zeta)\hat{U}_{\zeta a} \tag{5.11f}$$

where **u** is the displacement of a generic point, $\bar{u}$ is the displacement of a point on the reference surface, and **U** is the 'fiber displacement' rotations. The motion of the fibers can be interpreted as either displacements or rotations as will be discussed.

Hughes and Liu introduced the notation that follows, and the associated schematic shown in Figure 5.2, to describe the current deformed configuration with respect to the reference configuration:

$$y = \bar{y} + Y \tag{5.12a}$$

$$\bar{y} = \bar{x} + \bar{u} \tag{5.12b}$$

$$\bar{y}_a = \bar{x}_a + \bar{u}_a \tag{5.12c}$$

$$\mathrm{Y} = \mathrm{X} + \mathrm{U} \tag{5.12d}$$

$$\mathrm{Y_a} = \mathrm{X_a} + \mathrm{U_a} \tag{5.12e}$$

$$\hat{Y}_{\eta a} = \hat{X}_{\eta a} + \hat{U}_{\eta a} \tag{5.12f}$$

$$\hat{Y}_{\zeta a} = \hat{X}_{\zeta a} + \hat{U}_{\zeta a} \tag{5.12g}$$

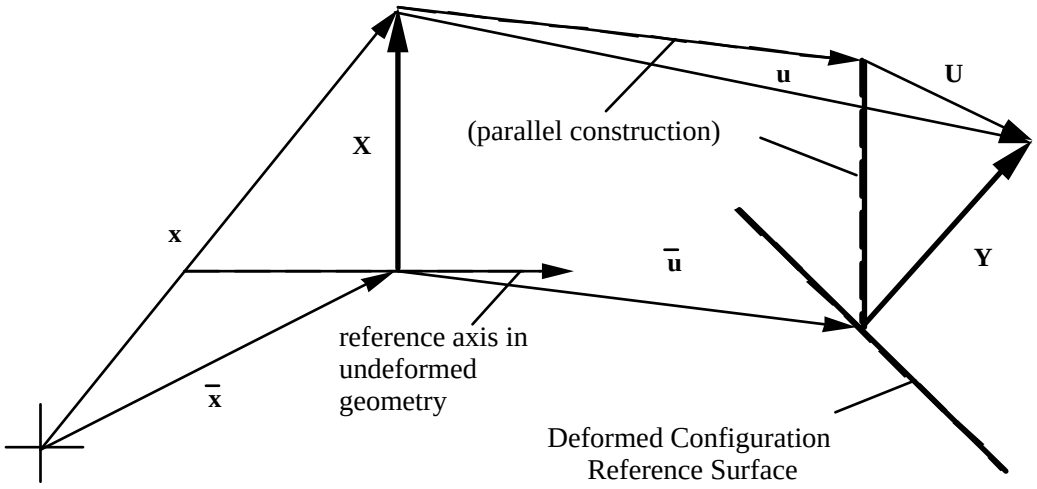


Figure 5.2. Schematic of deformed configuration displacements and position vectors.

In the above relations, and in Figure 5.2, the **x** quantities refer to the reference configuration, the **y** quantities refer to the updated (deformed) configuration and the **u** quantities are the displacements. The notation consistently uses a superscript bar $(\bar{\cdot})$ to indicate reference surface quantities, a superscript caret $(\hat{\cdot})$ to indicate unit vector

quantities, lower case letter for translational displacements, and upper case letters for fiber displacements. Thus to update to the deformed configuration, two vector quantities are needed: the reference surface displacement $\bar{u}$ and the associated nodal fiber displacement **U**. The nodal fiber displacements are defined in the fiber coordinate system, described in the next subsection.

5.2 Fiber Coordinate System

For a beam element, the known quantities will be the displacements of the reference surface $\bar{u}$ obtained from the translational equations of motion and the rotational quantities at each node obtained from the rotational equations of motion. What remains to complete the kinematics is a relation between nodal rotations and fiber displacements **U**. The linearized relationships between the incremental components $\Delta\hat{U}$ and the incremental rotations are given by

$$\begin{Bmatrix} \Delta\hat{U}_{\eta 1} \\ \Delta\hat{U}_{\eta 2} \\ \Delta\hat{U}_{\eta 3} \end{Bmatrix} = \begin{bmatrix} 0 & \hat{Y}_{\eta 3} & -\hat{Y}_{\eta 2} \\ -\hat{Y}_{\eta 3} & 0 & \hat{Y}_{\eta 1} \\ \hat{Y}_{\eta 2} & -\hat{Y}_{\eta 1} & 0 \end{bmatrix} \begin{Bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \Delta\theta_3 \end{Bmatrix} = h_\eta \Delta\theta \tag{5.13a}$$

$$\begin{Bmatrix} \Delta\hat{U}_{\zeta 1} \\ \Delta\hat{U}_{\zeta 2} \\ \Delta\hat{U}_{\zeta 3} \end{Bmatrix} = \begin{bmatrix} 0 & \hat{Y}_{\zeta 3} & -\hat{Y}_{\zeta 2} \\ -\hat{Y}_{\zeta 3} & 0 & \hat{Y}_{\zeta 1} \\ \hat{Y}_{\zeta 2} & -\hat{Y}_{\zeta 1} & 0 \end{bmatrix} \begin{Bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \Delta\theta_3 \end{Bmatrix} = h_\zeta \Delta\theta \tag{5.13b}$$

Equations (5.13) are used to transform the incremental fiber tip displacements to rotational increments in the equations of motion. The second-order accurate rotational update formulation due to Hughes and Winget [1980] is used to update the fiber vectors:

$$\hat{Y}_{\eta i}^{\,n+1} = R_{ij}(\Delta\theta)\hat{Y}_{\eta i}^{\,n} \tag{5.14a}$$

$$\hat{Y}_{\zeta i}^{\,n+1} = R_{ij}(\Delta\theta)\hat{Y}_{\zeta i}^{\,n} \tag{5.14b}$$

then

$$\Delta\hat{U}_{\eta a} = \hat{Y}_{\eta a}^{n+1} - \hat{Y}_{\eta a}^{n} \tag{5.15a}$$

$$\Delta\hat{U}_{\zeta a} = \hat{Y}_{\zeta a}^{n+1} - \hat{Y}_{\zeta a}^{n} \tag{5.15b}$$

where

$$R_{ij}(\Delta\theta) = \delta_{ij} + \frac{\left(2\delta_{ij} + \Delta S_{ik}\right)\Delta S_{ik}}{2D} \tag{5.16a}$$

$$\Delta S_{ij} = e_{ikj}\,\Delta\theta_k \tag{5.16b}$$

$$2D = 2 + \frac{1}{2}\left(\Delta\theta_1^2 + \Delta\theta_2^2 + \Delta\theta_3^2\right) \tag{5.16c}$$

Here δ_{ij} is the Kronecker delta and e_{ikj} is the permutation tensor.

5.2.1 Local Coordinate System

In addition to the above described fiber coordinate system, a local coordinate system is needed to enforce the zero normal stress conditions transverse to the axis. The orthonormal basis with two directions $\hat{e}_2$ and $\hat{e}_3$ normal to the axis of the beam is constructed as follows:

$$\hat{e}_1 = \frac{\bar{y}_2 - \bar{y}_1}{\|\bar{y}_2 - \bar{y}_1\|} \tag{5.17}$$

$$e_2' = \frac{\hat{Y}_{\eta 1} + \hat{Y}_{\eta 2}}{\left\|\hat{Y}_{\eta 1} + \hat{Y}_{\eta 2}\right\|} \tag{5.18}$$

From the vector cross product of these local tangents.

$$\hat{e}_3 = \hat{e}_1 \times e_2' \tag{5.19}$$

and to complete this orthonormal basis, the vector

$$\hat{e}_2 = \hat{e}_3 \times \hat{e}_1 \tag{5.20}$$

is defined. This coordinate system rigidly rotates with the deformations of the element.

The transformation of vectors from the global to the local coordinate system can now be defined in terms of the basis vectors as

$$\hat{A} = \begin{Bmatrix} \hat{A}_x \\ \hat{A}_y \\ \hat{A}_z \end{Bmatrix} = \begin{bmatrix} e_{1x} & e_{2x} & e_{3x} \\ e_{1y} & e_{2y} & e_{3y} \\ e_{1z} & e_{2z} & e_{3z} \end{bmatrix}^T \begin{Bmatrix} A_x \\ A_y \\ A_z \end{Bmatrix} = [q]\{A\} \tag{5.21}$$

where e_{ix}, e_{iy}, e_{iz} are the global components of the local coordinate unit vectors, $\hat{A}$ is a vector in the local coordinates, and **A** is the same vector in the global coordinate system.

5.3 Strains and Stress Update

5.3.1 Incremental Strain and Spin Tensors

The strain and spin increments are calculated from the incremental displacement gradient

$$G_{ij} = \frac{\partial \Delta u_i}{\partial y_j} \tag{5.22}$$

where Δu_i are the incremental displacements and y_j are the deformed coordinates. The incremental strain and spin tensors are defined as the symmetric and skew-symmetric parts, respectively, of G_{ij}:

$$\Delta \varepsilon_{ij} = \frac{1}{2}\left(G_{ij} + G_{ji}\right) \tag{5.23}$$

$$\Delta \omega_{ij} = \frac{1}{2}\left(G_{ij} - G_{ji}\right) \tag{5.24}$$

The incremental spin tensor $\Delta\omega_{ij}$ is used as an approximation to the rotational contribution of the Jaumann rate of the stress tensor; in an implicit implementation [Hallquist 1981b] the more accurate Hughes-Winget [1980] transformation matrix is used, Equation (5.16), with the incremental spin tensor for the rotational update. Here the Jaumann rate update is approximated as

$$\underline{\sigma}_{ij} = \sigma_{ij}^{n} + \sigma_{ip}^{n}\Delta\omega_{pj} + \sigma_{jp}^{n}\Delta\omega_{pi} \tag{5.25}$$

where the superscripts on the stress tensor refer to the updated (n+1) and reference (n) configurations. This update of the stress tensor is applied before the constitutive evaluation, and the stress and strain are stored in the global coordinate system.

5.3.2 Stress Update

To evaluate the constitutive relation, the stresses and strain increments are rotated from the global to the local coordinate system using the transformation defined previously in Equation (5.21), viz.

$$\sigma_{ij}^{l\,n} = q_{ik}\sigma_{kn}^{}q_{jn} \tag{5.26a}$$

$$\Delta\varepsilon_{ij}^{l} = q_{ik}\Delta\varepsilon_{kn}q_{jn} \tag{5.26b}$$

where the superscript l indicates components in the local coordinate system. The stress is updated incrementally:

$$\sigma_{ij}^{l\,n+1} = \sigma_{ij}^{l\,n} + \Delta\sigma_{ij}^{l\,n+\frac{1}{2}} \tag{5.27}$$

and rotated back to the global system:

$$\sigma_{ij}^{n+1} = q_{ki}\sigma_{kn}^{l\,n+1}q_{nj} \tag{5.28}$$

before computing the internal force vector.

5.3.3 Incremental Strain-Displacement Relations

After the constitutive evaluation is completed, the fully updated stresses are rotated back to the global coordinate system. These global stresses are then used to update the internal force vector

$$f_a^{\,\text{int}} = \int B_a^T \sigma d\upsilon \tag{5.29}$$

where $f_a^{\,\text{int}}$ are the internal forces at node a and B_a is the strain-displacement matrix in the global coordinate system associated with the displacements at node a. The B matrix relates six global strain components to eighteen incremental displacements [three translational displacements per node and the six incremental fiber tip displacements of Equation (5.15)]. It is convenient to partition the B matrix:

$$B = [B_1, B_2] \tag{5.30}$$

Each B_a submatrix is further partitioned into a portion due to strain and spin with the following submatrix definitions:

$$B_a = \begin{bmatrix} B_1 & 0 & 0 & B_4 & 0 & 0 & B_7 & 0 & 0 \\ 0 & B_2 & 0 & 0 & B_5 & 0 & 0 & B_8 & 0 \\ 0 & 0 & B_3 & 0 & 0 & B_6 0 & 0 & 0 & B_9 \\ B_2 & B_1 & 0 & B_5 & B_4 & 0 & B_8 & B_7 & 0 \\ 0 & B_3 & B_2 & 0 & B_6 & B_5 & 0 & B_9 & B_8 \\ B_3 & 0 & B_1 & B_6 & 0 & B_4 & B_9 & 0 & B_7 \end{bmatrix} \tag{5.31}$$

where

$$B_i = \begin{cases} N_{a,i} = \dfrac{\partial N_a}{\partial y_i} & \text{for } i = 1,2,3 \\ \left(N_a z_{\eta a}\right)_{,i-3} = \dfrac{\partial\left(N_a z_{\eta a}\right)}{\partial y_{i-3}} & \text{for } i = 4,5,6 \\ \left(N_a z_{\zeta a}\right)_{,i-6} = \dfrac{\partial\left(N_a z_{\zeta a}\right)}{\partial y_{i-6}} & \text{for } i = 7,8,9 \end{cases} \tag{5.32}$$

With respect to the strain-displacement relations, note that:

- the derivative of the shape functions are taken with respect to the global coordinates;
- the B matrix is computed on the cross-section located at the mid-point of the axis;
- the resulting B matrix is a 6×18 matrix.

The internal force, f, given by

$$f' = T^t f_a^{\text{int}} \tag{5.33}$$

is assembled into the global righthand side internal force vector. T is defined as (also see Equation (5.13)):

$$T = \begin{bmatrix} I & 0 \\ 0 & h_\eta \\ 0 & h_\zeta \end{bmatrix} \tag{5.34}$$

where I is a 3×3 identity matrix.

5.3.4 Spatial Integration

The integration of Equation (5.29) for the beam element is performed with one-point integration along the axis and multiple points in the cross section. For rectangular cross sections a variety of choices are available as is shown in Figure 5.3. The beam has no zero energy or locking modes.

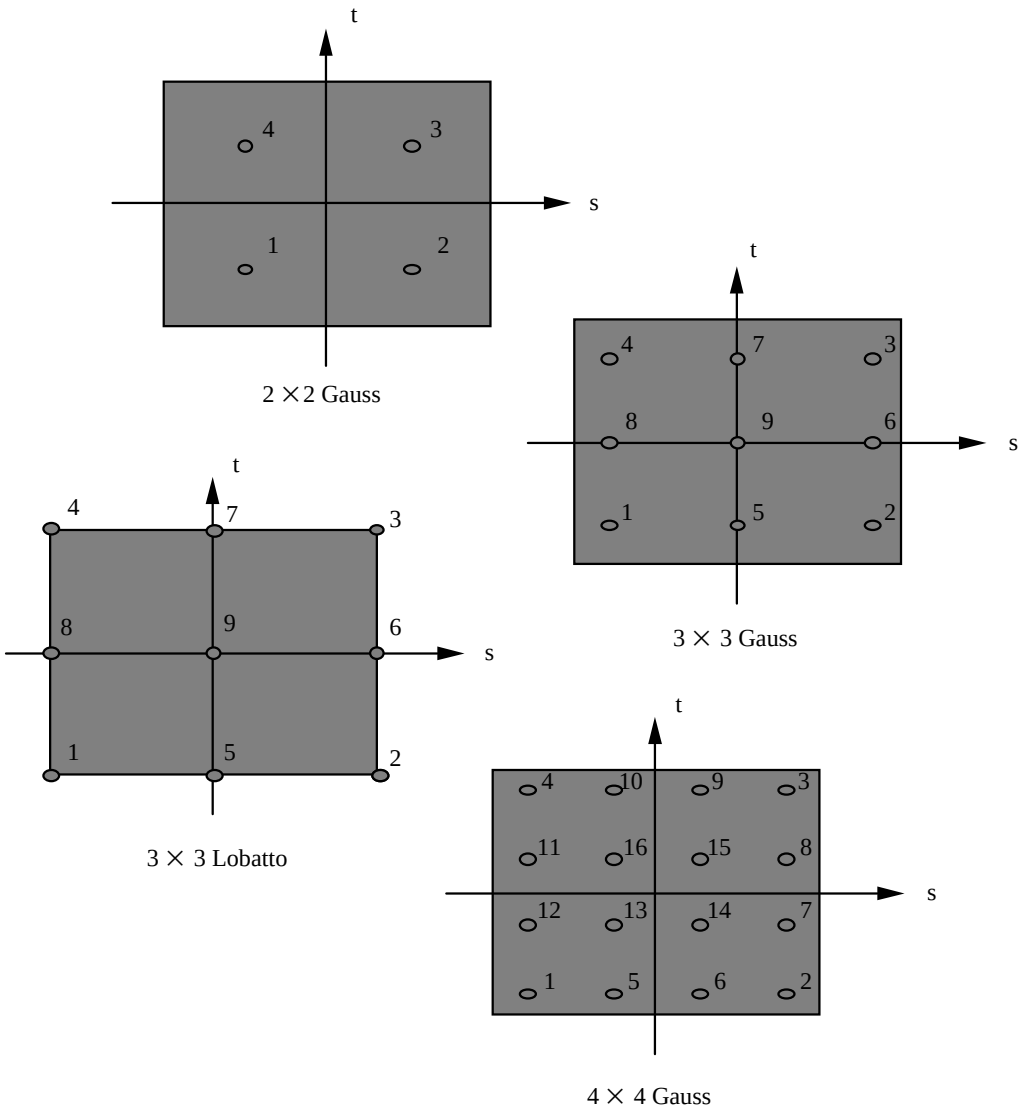


Figure 5.3 Integration possibilities for rectangular cross sections in the Hughes-Liu beam element.

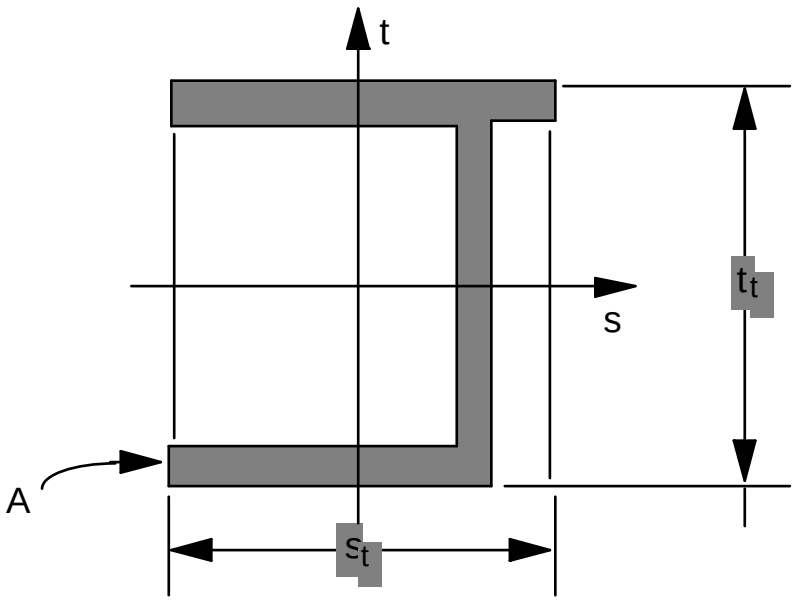


Figure 5.4. Specification of the nodal thicknesses, s_t and t_t, for a beam with an arbitrary cross-section.

For the user defined rule it is necessary to specify the number of integration points and the relative area for the total cross section:

$$A_r = \frac{A}{s_t \cdot t_t}$$

where s_t and t_t are the beam thicknesses specified on either the cross section or beam element cards. The rectangular cross-section which contains s_t and t_t should completely contain the cross-sectional geometry. Figure 5.4 illustrates this for a typical cross-section. In Figure 5.5 the area is broken into twelve integration points. For each integration point it is necessary to define the s and t parametric coordinates, (s_i,t_i), of the centroid of the i th integration point and the relative area associated with the point

$$A_{ri} = \frac{A_i}{A}$$

where A_i is the ratio of the area of the integration point and the actual area of the cross-section, A.

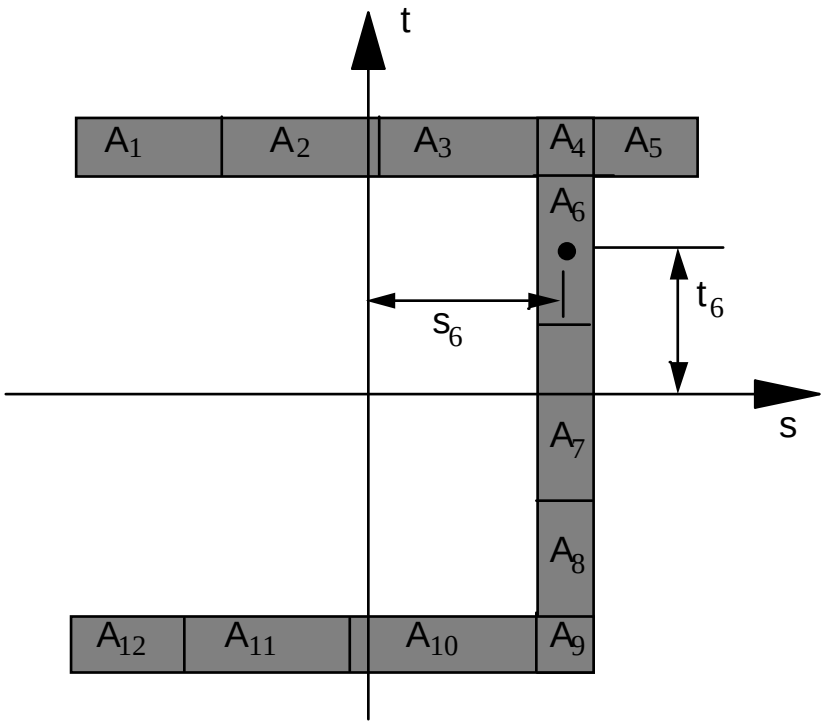


Figure 5.5. A break down of the cross section geometry in Figure 5.4 into twelve integration points.

6. BELYTSCHKO-LIN-TSAY SHELL

The Belytschko-Lin-Tsay shell element ([Belytschko and Tsay 1981], [Belytschko et al. 1984a])was implemented in as a computationally efficient alternative to the Hughes-Liu shell element. For a shell element with five through-the-thickness integration points, the Belytschko-Lin-Tsay shell elements requires 725 mathematical operations compared to 4066 operations for the under integrated Hughes-Liu element. The selectively reduced integration formulation of the Hughes-Liu element requires 35,367 mathematical operations. Because of its computational efficiency, the Belytschko-Lin-Tsay shell element is usually the shell element formulation of choice. For this reason, it has become the default shell element formulation.

The Belytschko-Lin-Tsay shell element is based on a combined co-rotational and velocity-strain formulation. The efficiency of the element is obtained from the mathematical simplifications that result from these two kinematical assumptions. The co-rotational portion of the formulation avoids the complexities of nonlinear mechanics by embedding a coordinate system in the element. The choice of velocity strain, or rate of deformation, in the formulation facilitates the constitutive evaluation, since the conjugate stress is the more familiar Cauchy stress. We closely follow the notation of Belytschko, Lin, and Tsay in the following development.

6.1 Co-rotational Coordinates

The midsurface of the quadrilateral shell element, or reference surface, is defined by the location of the element's four corner nodes. An embedded element coordinate system (see Figure 6.1) that deforms with the element is defined in terms of these nodal coordinates. Then the procedure for constructing the co-rotational coordinate system begins by calculating a unit vector normal to the main diagonal of the element:

$$\hat{e}_3 = \frac{s_3}{\|s_3\|} \tag{6.1a}$$

$$\|s_3\| = \sqrt{s_{31}^2 + s_{32}^2 + s_{33}^2} \tag{6.1b}$$

$$s_3 = r_{31} \times r_{42} \tag{6.1c}$$

where the superscript caret $(\hat{\cdot})$ is used to indicate the local (element) coordinate system.

It is desired to establish the local x axis $\hat{x}$ approximately along the element edge between nodes 1 and 2. This definition is convenient for interpreting the element stresses

which are defined in the local $\hat{x}-\hat{y}$ coordinate system. The procedure for constructing this unit vector is to define a vector s_1 that is nearly parallel to the vector r_{21}, viz.

$$s_1 = r_{21} - (r_{21}\cdot\hat{e}_3)\hat{e}_3 \tag{6.2a}$$

$$\hat{e}_1 = \frac{s_1}{\|s_1\|} \tag{6.2b}$$

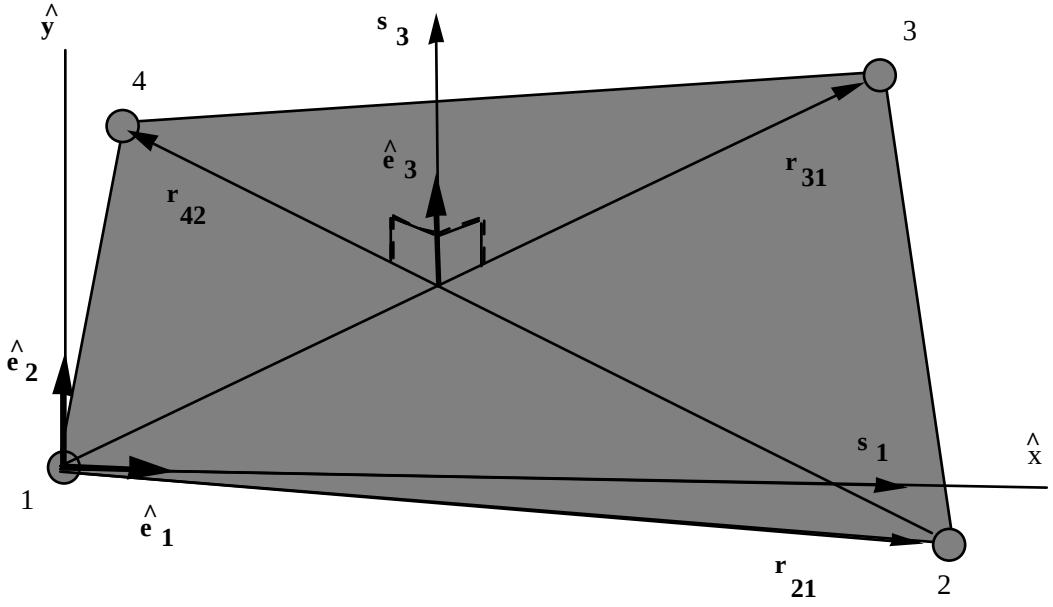


Figure 6.1. Construction of element coordinate system.

The remaining unit vector is obtained from the vector cross product

$$\hat{e}_2 = \hat{e}_3 \times \hat{e}_1 \tag{6.3}$$

If the four nodes of the element are coplanar, then the unit vectors $\hat{e}_1$ and $\hat{e}_2$ are tangent to the midplane of the shell and $\hat{e}_3$ is in the fiber direction. As the element deforms, an angle may develop between the actual fiber direction and the unit normal $\hat{e}_3$. The magnitude of this angle may be characterized as

$$|\hat{e}_3 \cdot f - 1| < \delta \tag{6.4}$$

where f is the unit vector in the fiber direction and the magnitude of δ depends on the magnitude of the strains. According to Belytschko et al., for most engineering

applications acceptable values of δ are on the order of 10^{-2} and if the condition presented in Equation (6.4) is met, then the difference between the rotation of the co-rotational coordinates $\hat{e}$ and the material rotation should be small.

The global components of this co-rotational triad define a transformation matrix between the global and local element coordinate systems. This transformation operates on vectors with global components **A** = (A_x, A_y, A_z) and element coordinate components $\hat{A} = \left(\hat{A}_x, \hat{A}_y, \hat{A}_z\right)$, and is defined as;

$$\{A\} = \begin{Bmatrix} A_x \\ A_y \\ A_z \end{Bmatrix} = \begin{bmatrix} e_{1x} & e_{2x} & e_{3x} \\ e_{1y} & e_{2y} & e_{3y} \\ e_{1z} & e_{2z} & e_{3z} \end{bmatrix} \begin{Bmatrix} \hat{A}_x \\ \hat{A}_y \\ \hat{A}_z \end{Bmatrix} = [\mu]\left\{\hat{A}\right\} = [q]^T \left\{\hat{A}\right\} \tag{6.5a}$$

where e_{ix}, e_{iy}, e_{iz} are the global components of the element coordinate unit vectors. The inverse transformation is defined by the matrix transpose, i.e.;

$$\left\{\hat{A}\right\} = [\mu]^T \{A\} \tag{6.5b}$$

6.2 Velocity-Strain Displacement Relations

The above small rotation condition, Equation (6.4), does not restrict the magnitude of the element's rigid body rotations. Rather the restriction is placed on the out-of-plane deformations, and thus on the element strain. Consistent with this restriction on the magnitude of the strains, the velocity-strain displacement relations used in the Belytschko-Lin-Tsay shell are also restricted to small strains.

As in the Hughes-Liu shell element, the displacement of any point in the shell is partitioned into a midsurface displacement (nodal translations) and a displacement associated with rotations of the element's fibers (nodal rotations). The Belytschko-Lin-Tsay shell element uses the Mindlin [1951] theory of plates and shells to partition the velocity of any point in the shell as

$$v = v^m - \hat{z}\, e_3 \times \theta \tag{6.6}$$

where v^m is the velocity of the mid-surface, θ is the angular velocity vector, and $\hat{z}$ is the distance along the fiber direction (thickness) of the shell element. The corresponding co-rotational components of the velocity strain (rate of deformation) are given by

$$\hat{d}_{ij} = \frac{1}{2}\left(\frac{\partial \hat{\upsilon}_i}{\partial \hat{x}_j} + \frac{\partial \hat{\upsilon}_j}{\partial \hat{x}_i} \right) \tag{6.7}$$

Substitution of Equation (6.6) into the above yields the following velocity-strain relations:

$$\hat{d}_x = \frac{\partial \hat{d}_x^m}{\partial \hat{x}} + \hat{z}\frac{\partial \hat{\theta}_y}{\partial \hat{x}} \tag{6.8a}$$

$$\hat{d}_y = \frac{\partial \hat{\upsilon}_y^m}{\partial \hat{y}} - \hat{z}\frac{\partial \hat{\theta}_x}{\partial \hat{y}} \tag{6.8b}$$

$$2\hat{d}_{xy} = \frac{\partial \hat{\upsilon}_x^m}{\partial \hat{y}} + \frac{\partial \hat{\upsilon}_y^m}{\partial \hat{x}} + \hat{z}\left(\frac{\partial \hat{\theta}_y}{\partial \hat{y}} - \frac{\partial \hat{\theta}_x}{\partial \hat{x}} \right) \tag{6.8c}$$

$$2\hat{d}_{yz} = \frac{\partial \hat{\upsilon}_z^m}{\partial \hat{y}} - \hat{\theta}_x \tag{6.8d}$$

$$2\hat{d}_{xz} = \frac{\partial \hat{\upsilon}_z^m}{\partial \hat{x}} + \hat{\theta}_y \tag{6.8e}$$

The above velocity-strain relations need to be evaluated at the quadrature points within the shell. Standard bilinear nodal interpolation is used to define the midsurface velocity, angular velocity, and the element's coordinates (isoparametric representation). These interpolations relations are given by

$$v^m = N_I(\xi,\eta)\, v_I \tag{6.9a}$$

$$\theta^m = N_I(\xi,\eta)\, \theta_I \tag{6.9b}$$

$$x^m = N_I(\xi,\eta)\, x_I \tag{6.9c}$$

where the subscript I is summed over all the element's nodes and the nodal velocities are obtained by differentiating the nodal coordinates with respect to time, i.e., $\upsilon_I = \dot{x}_I$. The bilinear shape functions are

$$N_1 = \frac{1}{4}(1-\xi)(1-\eta) \tag{6.10a}$$

$$N_2 = \frac{1}{4}(1+\xi)(1-\eta) \tag{6.10b}$$

$$N_3 = \frac{1}{4}(1 + \xi)(1 + \eta) \tag{6.10c}$$

$$N_4 = \frac{1}{4}(1 - \xi)(1 + \eta) \tag{6.10d}$$

The velocity strains at the center of the element, i.e., at $\xi = 0$, and $\eta = 0$, are obtained by substitution of the above relations into the previously defined velocity-strain displacement relations, Equations (6.8a) and (6.8e). After some algebra, this yields

$$\hat{d}_x = B_{1I}\hat{\upsilon}_{xI} + \hat{z}B_{1I}\hat{\theta}_{yI} \tag{6.11a}$$

$$\hat{d}_y = B_{2I}\hat{\upsilon}_{yI} - \hat{z}B_{2I}\hat{\theta}_{xI} \tag{6.11b}$$

$$2\hat{d}_{xy} = B_{2I}\hat{\upsilon}_{xI} + B_{1I}\hat{\upsilon}_{yI} + \hat{z}\left(B_{2I}\hat{\theta}_{yI} - B_{1I}\hat{\theta}_{xI}\right) \tag{6.11c}$$

$$2\hat{d}_{xz} = B_{1I}\hat{\upsilon}_{zI} + N_I\hat{\theta}_{yI} \tag{6.11d}$$

$$2\hat{d}_{yz} = B_{2I}\hat{\upsilon}_{zI} - N_I\hat{\theta}_{xI} \tag{6.11e}$$

where

$$B_{1I} = \frac{\partial N_I}{\partial \hat{x}} \tag{6.12a}$$

$$B_{2I} = \frac{\partial N_I}{\partial \hat{y}} \tag{6.12b}$$

The shape function derivatives B_{aI} are also evaluated at the center of the element, i.e., at $\xi = 0$, and $\eta = 0$.

6.3 Stress Resultants and Nodal Forces

After suitable constitutive evaluations using the above velocity strains, the resulting stresses are integrated through the thickness of the shell to obtain local resultant forces and moments. The integration formula for the resultants are

$$\hat{f}_{\alpha\beta}^{R} = \int \hat{\sigma}_{\alpha\beta}\, d\hat{z} \tag{6.13a}$$

$$\hat{m}_{\alpha\beta}^{R} = -\int \hat{z}\hat{\sigma}_{\alpha\beta}\, d\hat{z} \tag{6.13b}$$

where the superscript R indicates a resultant force or moment and the Greek subscripts emphasize the limited range of the indices for plane stress plasticity.

The above element-centered force and moment resultants are related to the local nodal forces and moments by invoking the principle of virtual power and performing a one- point quadrature. The relations obtained in this manner are

$$\hat{f}_{xI} = A\left(B_{1I}\,\hat{f}_{xx}^{R} + B_{2I}\,\hat{f}_{xy}^{R}\right) \tag{6.14a}$$

$$\hat{f}_{yI} = A\left(B_{2I}\,\hat{f}_{yy}^{R} + B_{1I}\,\hat{f}_{xy}^{R}\right) \tag{6.14b}$$

$$\hat{f}_{zI} = A\kappa\left(B_{1I}\,\hat{f}_{xz}^{R} + B_{2I}\,\hat{f}_{yz}^{R}\right) \tag{6.14c}$$

$$\hat{m}_{xI} = A\left(B_{2I}\hat{m}_{yy}^{R} + B_{1I}\hat{m}_{xy}^{R} - \frac{\kappa}{4}\,\hat{f}_{yz}^{R}\right) \tag{6.14d}$$

$$\hat{m}_{yI} = -A\left(B_{1I}\hat{m}_{xx}^{R} + B_{2I}\hat{m}_{xy}^{R} - \frac{\kappa}{4}\,\hat{f}_{xz}^{R}\right) \tag{6.14e}$$

$$\hat{m}_{zI} = 0 \tag{6.14f}$$

where A is the area of the element and κ is the shear factor from the Mindlin theory. In the Belytschko-Lin-Tsay formulation, κ is used as a penalty parameter to enforce the Kirchhoff normality condition as the shell becomes thin.

The above local nodal forces and moments are then transformed to the global coordinate system using the transformation relations given previously as Equation (6.5a). The global nodal forces and moments are then appropriately summed over all the nodes and the global equations of motion are solved for the next increment in nodal accelerations.

6.4 Hourglass Control (Belytschko-Lin-Tsay)

In part, the computational efficiency of the Belytschko-Lin-Tsay and the under integrated Hughes-Liu shell elements are derived from their use of one-point quadrature in the plane of the element. To suppress the hourglass deformation modes that accompany one-point quadrature, hourglass viscosity stresses are added to the physical

stresses at the local element level. The discussion of the hourglass control that follows pertains to the Hughes-Liu and the membrane elements as well.

The hourglass control used by Belytschko et al. extends an earlier derivation by Flanagan and Belytschko [1981], (see also Kosloff and Frazier [1978], Belytschko and Tsay [1983]). The hourglass shape vector τ_I is defined as

$$\tau_I = h_I - \left(h_J \hat{x}_{aJ}\right) B_{aI} \tag{6.15}$$

where

$$h = \begin{bmatrix} +1 \\ -1 \\ +1 \\ -1 \end{bmatrix} \tag{6.16}$$

is the basis vector that generates the deformation mode that is neglected by one-point quadrature. In Equation (6.15) and the reminder of this subsection, the Greek subscripts have a range of 2, e.g., $\hat{x}_{aI} = \left(\hat{x}_{1I}, \hat{x}_{2I}\right) = \left(\hat{x}_I, \hat{y}_I\right)$.

The hourglass shape vector then operates on the generalized displacements, in a manner similar to Equations (6.11a - e), to produce the generalized hourglass strain rates

$$\dot{q}_\alpha^B = \tau_I \hat{\theta}_{\alpha I} \tag{6.17a}$$

$$\dot{q}_3^B = \tau_I \hat{\upsilon}_{zI} \tag{6.17b}$$

$$\dot{q}_\alpha^M = \tau_I \hat{\upsilon}_{\alpha I} \tag{6.17c}$$

where the superscripts B and M denote bending and membrane modes, respectively. The corresponding hourglass stress rates are then given by

$$\dot{Q}_\alpha^B = \frac{r_\theta E t^3 A}{192} B_{\beta I} B_{\beta I} \dot{q}_\alpha^B \tag{6.18a}$$

$$\dot{Q}_3^B = \frac{r_w \kappa G t^3 A}{12} B_{\beta I} B_{\beta I} \dot{q}_3^B \tag{6.18b}$$

$$\dot{Q}_\alpha^M = \frac{r_m E t A}{8} B_{\beta I} B_{\beta I} \dot{q}_\alpha^M \tag{6.18c}$$

where t is the shell thickness and the parameters, r_θ, r_w and r_m are generally assigned values between 0.01 and 0.05.

Finally, the hourglass stresses which are updated from the stress rates in the usual way, i.e.,

$$Q^{n+1} = Q^n + \Delta t \dot{Q} \tag{6.19}$$

and the hourglass resultant forces are then

$$\hat{m}_{\alpha I}^H = \tau_I Q_\alpha^B \tag{6.20a}$$

$$\hat{f}_{3I}^H = \tau_I Q_3^B \tag{6.20b}$$

$$\hat{f}_{\alpha I}^H = \tau_I Q_\alpha^M$$

where the superscript H emphasizes that these are internal force contributions from the hourglass deformations. These hourglass forces are added directly to the previously determined local internal forces due to deformations Equations (6.14a - f).

6.5 Hourglass Control (Englemann and Whirley)

Englemann and Whirley [1991] have developed an alternative hourglass control which they recently implemented in the framework of the Belytschko, Lin, and Tsay shell element. Although only the hourglass control has changed they refer to the shell using this formulation as the YASE (Yet Another Shell Element) element. The implicit implementation of YASE [Englemann, Whirley, and Goudreau 1989] is a fully integrated shell and provides the basis for the hourglass stabilization. We will briefly highlight their procedure here which has proven to be cost effective-only twenty percent more expensive than the default control.

In the new hourglass procedure, the in-plane strain field (subscript p) is decomposed into the one point strain field plus the stabilization strain field:

$$\dot{\bar{\varepsilon}}_p = \dot{\bar{\varepsilon}}_p^0 + \dot{\bar{\varepsilon}}_p^s \tag{6.21}$$

where the stabilization strain field, which is obtained from the assumed strain fields of Pian and Sumihara [1984] is given in terms of the hourglass velocity field as

$$\dot{\varepsilon}_p^s = W_m \dot{q}_m + z W_b \dot{q}_b \tag{6.22}$$

Here, W_m and W_b play the role of stabilization strain velocity operators for membrane and bending:

$$W_m = \begin{bmatrix} f_1^p(\xi,\eta) & f_4^p(\xi,\eta) \\ f_2^p(\xi,\eta) & f_5^p(\xi,\eta) \\ f_3^p(\xi,\eta) & f_6^p(\xi,\eta) \end{bmatrix} \tag{6.23}$$

$$W_b = \begin{bmatrix} -f_4^p(\xi,\eta) & f_1^p(\xi,\eta) \\ -f_5^p(\xi,\eta) & f_2^p(\xi,\eta) \\ -f_6^p(\xi,\eta) & f_3^p(\xi,\eta) \end{bmatrix} \tag{6.24}$$

where the terms $f_i^p(\xi,\eta)$ i=1,2,...,6, are rather complicated and the reader is referred to the reference [Englemann and Whirley 1991].

To obtain the transverse shear assumed strain field, the procedure given in [Bathe and Dvorkin,1984] is used. The transverse shear strain field can again be decomposed into the one point strain field plus the stabilization field:

$$\dot{\varepsilon}_s^s = W_s \dot{q}_s \tag{6.25}$$

that is related to the hourglass velocities by

$$\dot{\varepsilon}_s^s = W_s \dot{q}_s \tag{6.26}$$

where the transverse shear stabilization strain-velocity operator W_S is given by

$$W_s = \begin{bmatrix} f_1^s(\xi,\eta) & -g_1^s\xi & g_2^s\eta & g_3^s\xi & g_3^s\eta \\ f_2^s(\xi,\eta) & g_4^s\xi & g_4^s\eta & -g_2^s\xi & g_1^s\eta \end{bmatrix} \tag{6.27}$$

Again, the coefficients $f_1^s(\xi,\eta)$ and g_1^s are defined in the reference..

In their formulation the hourglass forces are related to the hourglass velocity field through an incremental hourglass constitutive equation derived from an additive decomposition of the stress into a “one-point stress,” plus a “stabilization stress.” The

integration of the stabilization stress gives a resultant constitutive equation relating hourglass forces to hourglass velocities.

The in-plane and transverse stabilization stresses are updated according to:

$$\begin{aligned} \tau_s^{s,n+1} &= \tau_s^{s,n} + \Delta t c_s C_s \dot{\bar{\varepsilon}}_s^s \\ \tau_s^{s,n+1} &= \tau_s^{s,n} + \Delta t c_s C_s \dot{\bar{\varepsilon}}_s^s \end{aligned} \tag{6.28}$$

where the tangent matrix is the product of a matrix C, which is constant within the shell domain, and a scalar c which is constant in the plane but may vary through the thickness.

The stabilization stresses can now be used to obtain the hourglass forces:

$$\begin{aligned} Q_m &= \int_{-\frac{h}{2}}^{\frac{h}{2}} \int_A W_m^T \tau_p^s \, dA dz \\ Q_b &= \int_{-\frac{h}{2}}^{\frac{h}{2}} \int_A W_b^T \tau_p^s \, dA dz \\ Q_s &= \int_{-\frac{h}{2}}^{\frac{h}{2}} \int_A W_s^T \tau_s^s \, dA dz \end{aligned} \tag{6.29}$$

6.6 Belytschko-Wong-Chiang Improvement

Since the Belytschko-Tsay element (and therefore the YASE shell above) is based on a perfectly flat geometry, warpage is not considered. Although this generally poses no major difficulties and provides for an efficient element, incorrect results in the twisted beam problem are obtained where the nodal points of the elements used in the discretization are not coplanar. The Hughes-Liu shell element considers non-planar geometry and gives good results on the twisted beam, but is relatively expensive. The effect of neglecting warpage in typical a application cannot be predicted beforehand and may lead to less than accurate results, but the latter is only speculation and is difficult to verify in practice. Obviously, it would be better to use shells that consider warpage if the added costs are reasonable and if this unknown effect is eliminated. In this section we briefly describe the simple and computationally inexpensive modifications necessary in the Belytschko-Tsay shell to include the warping stiffness. The improved transverse shear treatment is also described which is necessary for the element to pass the Kirchhoff

patch test. Readers are directed to the references [Belytschko, Wong, and Chang 1989, 1992] for an in depth theoretical background.

In order to include warpage in the formulation it is convenient to define nodal fiber vectors as shown in Figure 6.2. The geometry is interpolated over the surface of the shell from:

$$x = x^m + \bar{\zeta} p = \left(x_I + \bar{\zeta} p_I \right) N_I(\xi, \eta) \tag{6.30}$$

where

$$\bar{\zeta} = \frac{\zeta h}{2}$$

and ζ is a parametric coordinate which varies between -1 to +1. The in plane strain

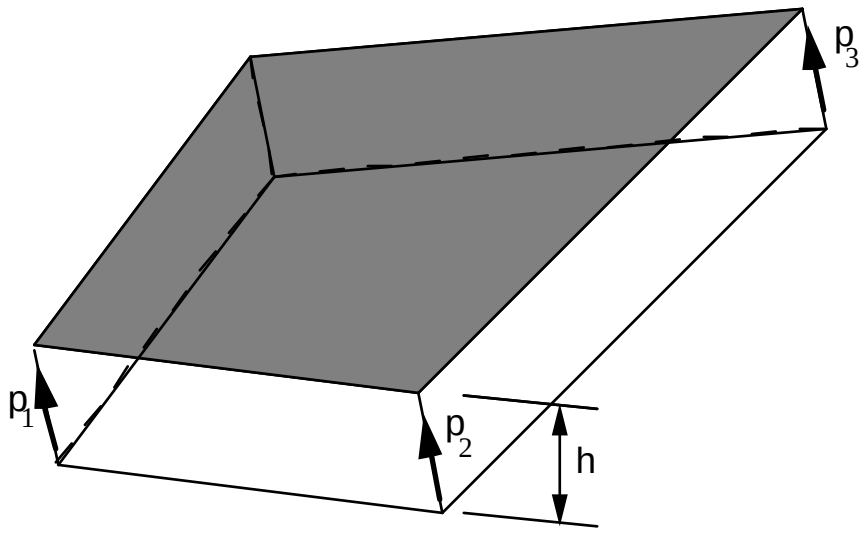


Figure 6.2. Nodal fiber vectors p_1, p_2, and p_3 where h is the thickness.

components are given by:

$$d_{xx} = b_{xI} \hat{v}_{xI} + \bar{\zeta} \left(b^c_{xI} \hat{v}_{xI} + b_{xI} \dot{p}_{xI} \right) \tag{6.31a}$$

$$d_{yy} = b_{yI} \hat{v}_{yI} + \bar{\zeta} \left(b^c_{yI} \hat{v}_{yI} + b_{yI} \dot{p}_{yI} \right) \tag{6.31b}$$

$$d_{xy} = \frac{1}{2} b_{xI} \hat{v}_{yI} + b_{yI} \hat{v}_{xI} + \overline{\zeta} \left(b^c_{xI} \hat{v}_{yI} + b_{xI} \dot{p}_{yI} + b^c_{yI} \hat{v}_{xI} + b_{yI} \dot{p}_{xI} \right) \qquad (6.31c)$$

The coupling terms are come in through b^c_{iI} : which is defined in terms of the components of the fiber vectors as:

$$\begin{Bmatrix} b^c_{xI} \\ b^c_{yI} \end{Bmatrix} = \begin{bmatrix} p_{\hat{y}2} - p_{\hat{y}4} & p_{\hat{y}3} - p_{\hat{y}1} & p_{\hat{y}4} - p_{\hat{y}2} & p_{\hat{y}1} - p_{\hat{y}3} \\ p_{\hat{x}2} - p_{\hat{x}4} & p_{\hat{x}3} - p_{\hat{x}1} & p_{\hat{x}4} - p_{\hat{x}2} & p_{\hat{x}1} - p_{\hat{x}3} \end{bmatrix} \qquad (6.32)$$

For a flat geometry the normal vectors are identical and no coupling can occur. Two methods are used by Belytschko for computing b^c_{iI} and the reader is referred to his papers for the details. Both methods have been tested in LS-DYNA3D and comparable results were obtained.

The transverse shear strain components are given as

$$\hat{\gamma}_{xz} = -N_I(\xi, \eta) \overline{\theta}_{\hat{y}I} \qquad (6.33a)$$

$$\hat{\gamma}_{yz} = -N_I(\xi, \eta) \overline{\theta}_{\hat{x}I} \qquad (6.33b)$$

where the nodal rotational components are defined as:

$$\overline{\theta}_{\hat{x}I} = \left(e^I_n \cdot e_{\hat{x}} \right) \overline{\theta}^I_n + \left(e^K_n \cdot e_{\hat{x}} \right) \overline{\theta}^K_n \qquad (6.34a)$$

$$\overline{\theta}_{\hat{y}I} = \left(e^I_n \cdot e_{\hat{y}} \right) \overline{\theta}^I_n + \left(e^K_n \cdot e_{\hat{y}} \right) \overline{\theta}^K_n \qquad (6.34b)$$

The rotation $\overline{\theta}^I_n$ comes from the nodal projection

$$\overline{\theta}^I_n = \frac{1}{2} \left(\theta^I_{nI} + \theta^I_{nJ} \right) + \frac{1}{L^{IJ}} \left(\hat{\upsilon}_{zJ} - \hat{\upsilon}_{zJ} \right) \qquad (6.35)$$

where the subscript n refers to the normal component of side I as seen in Figure 6.3 and L^{IJ} is the length of side *IJ*.

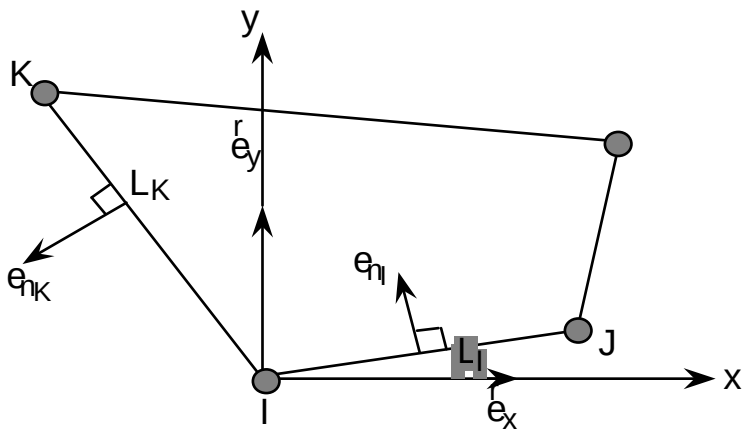


Figure 6.3 Vector and edge definitions for computing the transverse shear strain components.

7. C^0 TRIANGULAR SHELL

The C^0 shell element due to Kennedy, Belytschko, and Lin [1986] has been implemented as a computationally efficient triangular element complement to the Belytschko-Lin-Tsay quadrilateral shell element ([Belytschko and Tsay 1981], [Belytschko et al. 1984a]). For a shell element with five through-the-thickness integration points, the element requires 649 mathematical operations (the Belytschko-Lin-Tsay quadrilateral shell element requires 725 mathematical operations) compared to 1417 operations for the Marchertas-Belytschko triangular shell [Marchertas and Belytschko 1974] (referred to as the BCIZ [Bazeley, Cheung, Irons, and Zienkiewicz 1965] triangular shell element in the DYNA3D user's manual).

Triangular shell elements are offered as optional elements primarily for compatibility with local user grid generation and refinement software. Many computer aided design (CAD) and computer aided manufacturing (CAM) packages include finite element mesh generators, and most of these mesh generators use triangular elements in the discretization. Similarly, automatic mesh refinement algorithms are typically based on triangular element discretization. Also, triangular shell element formulations are not subject to zero energy modes inherent in quadrilateral element formulations.

The triangular shell element's origin are based on the work of Belytschko et al. [Belytschko, Stolarski, and Carpenter 1984b] where the linear performance of the shell was demonstrated. Because the triangular shell element formulations parallels closely the formulation of the Belytschko-Lin-Tsay quadrilateral shell element presented in the previous section (Section 6), the following discussion is limited to items related specifically to the triangular shell element.

7.1 Co-rotational Coordinates

The midsurface of the triangular shell element, or reference surface, is defined by the location of the element's three nodes. An embedded element coordinate system (see Figure 7.1) that deforms with the element is defined in terms of these nodal coordinates. The procedure for constructing the co-rotational coordinate system is simpler than the corresponding procedure for the quadrilateral, because the three nodes of the triangular element are guaranteed coplanar.

The local x-axis $\hat{x}$ is directed from node 1 to 2. The element's normal axis $\hat{z}$ is defined by the vector cross product of a vector along $\hat{x}$ with a vector constructed from node 1 to node 3. The local y-axis $\hat{y}$ is defined by a unit vector cross product of $\hat{e}_3$ with $\hat{e}_1$, which are the unit vectors in the $\hat{z}$ directions, respectively. As in the case of the

quadrilateral element, this triad of co-rotational unit vectors defines a transformation between the global and local element coordinate systems (see Equations (6.5a,b)).

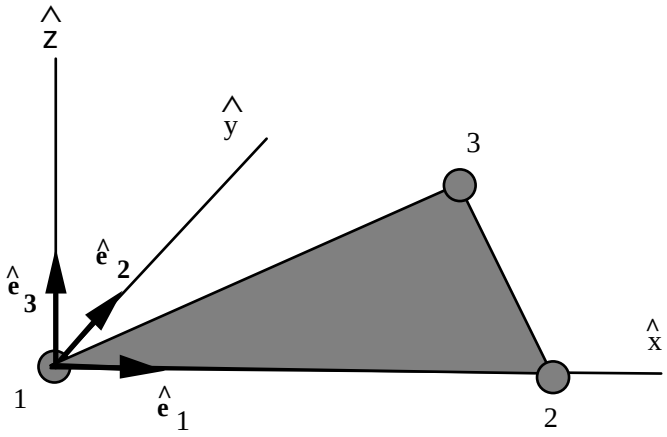


Figure 7.1 Local element coordinate system for C^0 shell element.

7.2 Velocity-Strain Relations

As in the Belytschko-Lin-Tsay quadrilateral shell element, the displacement of any point in the shell is partitioned into a midsurface displacement (nodal translations) and a displacement associated with rotations of the element's fibers (nodal rotations). The Kennedy-Belytschko-Lin triangular shell element also uses the Mindlin [Mindlin 1951] theory of plates and shells to partition the velocity of any point in the shell (recall Equation (6.6)):

$$v = v^m - \hat{z}\, e_3 \times \theta \tag{7.1}$$

where v^m is the velocity of the mid-surface, θ is the angular velocity vector, and z is the distance along the fiber direction (thickness) of the shell element. The corresponding co-rotational components of the velocity strain (rate of deformation) were given previously in Equation (6.11a - e).

Standard linear nodal interpolation is used to define the midsurface velocity, angular velocity, and the element's coordinates (isoparametric representation). These interpolation functions are the area coordinates used in triangular element formulations. Substitution of the nodally interpolated velocity fields into the velocity-strain relations (see Belytschko et al. for details), leads to strain rate-velocity relations of the form

$$\hat{d} = B\hat{v} \tag{7.2}$$

where $\hat{d}$ are the velocity strains (strain rates), the elements of **B** are derivatives of the nodal interpolation functions, and the $\hat{v}$ are the nodal velocities and angular velocities.

It is convenient to partition the velocity strains and the **B** matrix into membrane and bending contributions. The membrane relations are given by

$$\begin{Bmatrix} \hat{d}_x \\ \hat{d}_y \\ 2\hat{d}_{xy} \end{Bmatrix}^M = \frac{1}{\hat{x}_2\hat{y}_3} \begin{bmatrix} \hat{y}_3 & 0 & \hat{y}_3 & 0 & 0 & 0 \\ 0 & \hat{x}_3 - \hat{x}_2 & 0 & -\hat{x}_3 & 0 & \hat{x}_2 \\ \hat{x}_3 - \hat{x}_2 & -\hat{y}_3 & -\hat{x}_3 & \hat{y}_3 & \hat{x}_2 & 0 \end{bmatrix} \begin{Bmatrix} \hat{\upsilon}_{x1} \\ \hat{\upsilon}_{y1} \\ \hat{\upsilon}_{x2} \\ \hat{\upsilon}_{y2} \\ \hat{\upsilon}_{x3} \\ \hat{\upsilon}_{y3} \end{Bmatrix} \tag{7.3}$$

or

$$\hat{d}^{\,M} = B_M\,\hat{v} \tag{7.4}$$

The bending relations are given by

$$\begin{Bmatrix} \hat{\kappa}_x \\ \hat{\kappa}_y \\ 2\hat{\kappa}_{xy} \end{Bmatrix} = \frac{-1}{\hat{x}_2\hat{y}_3} \begin{bmatrix} 0 & -\hat{y}_3 & 0 & \hat{y}_3 & 0 & 0 \\ \hat{x}_3 - \hat{x}_2 & 0 & \hat{x}_3 & 0 & -\hat{x}_2 & 0 \\ \hat{y}_3 & \hat{x}_3 - \hat{x}_2 & -\hat{y}_3 & -\hat{x}_3 & 0 & \hat{x}_2 \end{bmatrix} \begin{Bmatrix} \hat{\theta}_{x1} \\ \hat{\theta}_{y1} \\ \hat{\theta}_{x2} \\ \hat{\theta}_{y2} \\ \hat{\theta}_{x3} \\ \hat{\theta}_{y3} \end{Bmatrix} \tag{7.5}$$

or

$$\hat{\kappa}^{\,M} = B_M\hat{\theta}^{\,\mathrm{def}} \tag{7.6}$$

The local element velocity strains are then obtained by combining the above two relations:

$$\begin{Bmatrix} \hat{d}_x \\ \hat{d}_y \\ 2\hat{d}_{xy} \end{Bmatrix} = \begin{Bmatrix} \hat{d}_x \\ \hat{d}_y \\ 2\hat{d}_{xy} \end{Bmatrix}^M - \hat{z} \begin{Bmatrix} \hat{\kappa}_x \\ \hat{\kappa}_y \\ 2\hat{\kappa}_{xy} \end{Bmatrix} = \hat{d}^{\,M} - \hat{z}\hat{\kappa} \tag{7.7}$$

The remaining two transverse shear strain rates are given by

$$\begin{Bmatrix} 2\hat{d}_{xz} \\ 2\hat{d}_{yz} \end{Bmatrix} = \frac{1}{6\hat{x}_2\,\hat{y}_3}$$
$$\begin{bmatrix} -\hat{y}_3^{\,2} & \hat{y}_3(2\hat{x}_2+\hat{x}_3) & \hat{y}_3^{\,2} & \hat{y}_3(3\hat{x}_2-\hat{x}_3) & 0 & \hat{x}_2\hat{y}_3 \\ \hat{y}_3(\hat{x}_2-2\hat{x}_2) & \hat{x}_2^2-\hat{x}_3^2 & -\hat{y}_3^{\,2}(\hat{x}_2+\hat{x}_3) & \hat{x}_3(\hat{x}_3-2\hat{x}_2) & -3\hat{x}_2\hat{y}_3 & \hat{x}_2(2\hat{x}_3-\hat{x}_2) \end{bmatrix}$$
$$\begin{Bmatrix} \hat{\theta}_{x1} \\ \hat{\theta}_{y1} \\ \hat{\theta}_{x2} \\ \hat{\theta}_{y2} \\ \hat{\theta}_{x3} \\ \hat{\theta}_{y3} \end{Bmatrix}^{\text{def}} \tag{7.8}$$

or

$$\hat{d}^{\,S} = B_S \hat{\theta}^{\,\text{def}} \tag{7.9}$$

All of the above velocity-strain relations have been simplified by using one-point quadrature.

In the above relations, the angular velocities $\hat{\theta}^{\,\text{def}}$ are the deformation component of the angular velocity $\hat{\theta}$ obtained by subtracting the portion of the angular velocity due to rigid body rotation, i.e.

$$\hat{\theta}^{\,\text{def}} = \hat{\theta} - \hat{\theta}^{\,\text{rig}} \tag{7.10}$$

The two components of the rigid body angular velocity are given by

$$\hat{\theta}_y^{\text{rig}} = \frac{\hat{\upsilon}_{z1} - \hat{\upsilon}_{z2}}{\hat{x}_2} \tag{7.11a}$$

$$\hat{\theta}_x^{\text{rig}} = \frac{(\hat{\upsilon}_{z3} - \hat{\upsilon}_{z1})\hat{x}_2 - (\hat{\upsilon}_{z2} - \hat{\upsilon}_{z1})\hat{x}_3}{\hat{x}_2 \hat{y}_3} \tag{7.11b}$$

The first of the above two relations is obtained by considering the angular velocity of the local x-axis about the local y-axis. Referring to Figure 7.1, by construction nodes 1 and 2 lie on the local x-axis and the distance between the nodes is $\hat{x}_2$ i.e., the $\hat{x}$ distance from node 2 to the local coordinate origin at node 1. Thus the difference in the nodal $\hat{z}$ velocities divided by the distance between the nodes is an average measure of the rigid body rotation rate about the local y-axis.

The second relation is conceptually identical, but is implemented in a slightly different manner due to the arbitrary location of node 3 in the local coordinate system. Consider the two local element configurations shown in Figure 7.2. For the leftmost configuration, where node 3 is the local y-axis, the rigid body rotation rate about the local x-axis is given by

$$\hat{\theta}_{x-left}^{\text{rig}} = \frac{\hat{\upsilon}_{z3} - \hat{\upsilon}_{z1}}{\hat{y}_3} \tag{7.12}$$

and for the rightmost configuration the same rotation rate is given by

$$\hat{\theta}_{x-right}^{\text{rig}} = \frac{\hat{\upsilon}_{z3} - \hat{\upsilon}_{z2}}{\hat{y}_3} \tag{7.13}$$

Although both of these relations yield the average rigid body rotation rate, the selection of the correct relation depends on the configuration of the element, i.e., on the location of node 3. Since every element in the mesh could have a configuration that is different in general from either of the two configurations shown in Figure 7.2, a more robust relation is needed to determine the average rigid body rotation rate about the local x-axis. In most typical grids, node 3 will be located somewhere between the two configurations shown in Figure 7.2. Thus a linear interpolation between these two rigid body rotation rates was devised using the distance $\hat{x}_3$ as the interpolant:

$$\hat{\theta}_x^{\text{rig}} = \hat{\theta}_{x-left}^{\text{rig}} \left(1 - \frac{\hat{x}_3}{\hat{x}_2}\right) + \hat{\theta}_{x-right}^{\text{rig}} \left(\frac{\hat{x}_3}{\hat{x}_2}\right) \tag{7.14}$$

Substitution of Equations (7.12) and (7.13) into (7.14) and simplifying produces the relations given previously as Equation (7.11b).

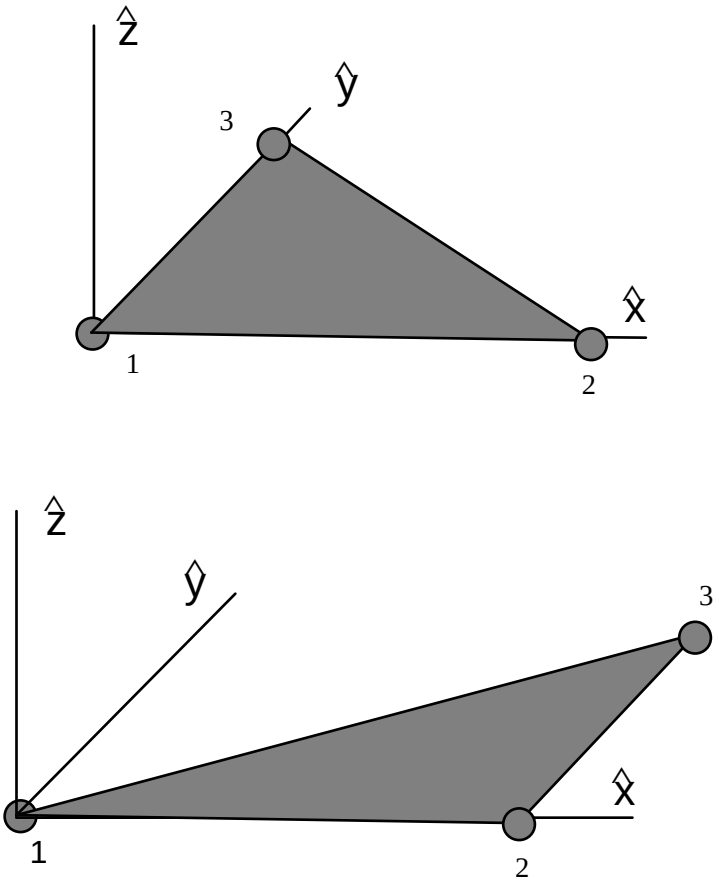


Figure 7.2 Element configurations with node 3 aligned with node 1(left) and node 3 aligned with node 2 (right).

7.3 Stress Resultants and Nodal Forces

After suitable constitutive evaluation using the above velocity strains, the resulting local stresses are integrated through the thickness of the shell to obtain local resultant forces and moments. The integration formula for the resultants are

$$\hat{f}^R_{\alpha\beta} = \int \hat{\sigma}_{\alpha\beta} d\,\hat{z} \tag{7.15a}$$

$$\hat{m}^R_{\alpha\beta} = -\int \hat{z}\,\hat{\sigma}_{\alpha\beta} d\,\hat{z} \tag{7.15b}$$

where the superscript R indicates a resultant force or moment and the Greek subscripts emphasize the limited range of the indices for plane stress plasticity.

The above element-centered force and moment resultant are related to the local nodal forces and moments by invoking the principle of virtual power and performing a one-point quadrature. The relations obtained in this manner are

$$\begin{Bmatrix} \hat{f}_{x1} \\ \hat{f}_{y1} \\ \hat{f}_{x2} \\ \hat{f}_{y2} \\ \hat{f}_{x3} \\ \hat{f}_{y3} \end{Bmatrix} = AB_M^T \begin{Bmatrix} \hat{f}_{xx}^R \\ \hat{f}_{yy}^R \\ \hat{f}_{xy}^R \end{Bmatrix} \tag{7.16a}$$

$$\begin{Bmatrix} \hat{m}_{x1} \\ \hat{m}_{y1} \\ \hat{m}_{x2} \\ \hat{m}_{y2} \\ \hat{m}_{x3} \\ \hat{m}_{y3} \end{Bmatrix} = AB_M^T \begin{Bmatrix} \hat{m}_{xx}^R \\ \hat{m}_{yy}^R \\ \hat{m}_{xy}^R \end{Bmatrix} + AB_S^T \begin{Bmatrix} \hat{f}_{xz}^R \\ \hat{f}_{yz}^R \end{Bmatrix} \tag{7.16b}$$

where A is the area of the element $\left(2A = \hat{x}_2 \hat{y}_3\right)$.

The remaining nodal forces, the $\hat{z}$ component of the force $\left(\hat{f}_{z3}, \hat{f}_{z2}, \hat{f}_{z1}\right)$, are determined by successively solving the following equilibration equations

$$\hat{m}_{x1} + \hat{m}_{x2} + \hat{m}_{x3} + \hat{y}_3 \hat{f}_{z3} = 0 \tag{7.17a}$$

$$\hat{m}_{y1} + \hat{m}_{y2} + \hat{m}_{y3} - \hat{x}_3 \hat{f}_{z3} - \hat{x}_2 \hat{f}_{z2} = 0 \tag{7.17b}$$

$$\hat{f}_{z1} + \hat{f}_{z2} + \hat{f}_{z3} = 0 \tag{7.17c}$$

which represent moment equilibrium about the local x-axis, moment equilibrium about the local y-axis, and force equilibrium in the local z-direction, respectively.

8. MARCHERTAS-BELYTSCHKO TRIANGULAR SHELL

The Marchertas-Belytschko [1974] triangular shell element, or the BCIZ triangular shell element as it is referred to in the LS-DYNA3D user's manual, was developed in the same time period as the Belytschko beam element [Belytschko, Schwer, and Klein, 1977], see Section 4, forming the first generation of co-rotational structural elements developed by Belytschko and co-workers. This triangular shell element became the first triangular shell implemented in DYNA3D. Although the Marchertas-Belytschko shell element is relatively expensive, i.e. the C^0 triangular shell element with five through-the-thickness integration points requires 649 mathematical operations compared to 1417 operations for the Marchertas-Belytschko triangular shell, it is maintained in LS-DYNA3D for compatibility with earlier user models. However, as the LS-DYNA3D user community moves to application of the more efficient shell element formulations, the use of the Marchertas-Belytschko triangular shell element will decrease.

As mentioned above, the Marchertas-Belytschko triangular shell has a common co-rotational formulation origin with the Belytschko beam element. The interested reader is referred to the beam element description, see Section, 4 for details on the co-rotational formulation. In the next subsection a discussion of how the local element coordinate system is identical for the triangular shell and beam elements. The remaining subsections discuss the triangular element's displacement interpolants, the strain displacement relations, and calculations of the element nodal forces. In the report [1974], much greater detail is provided.

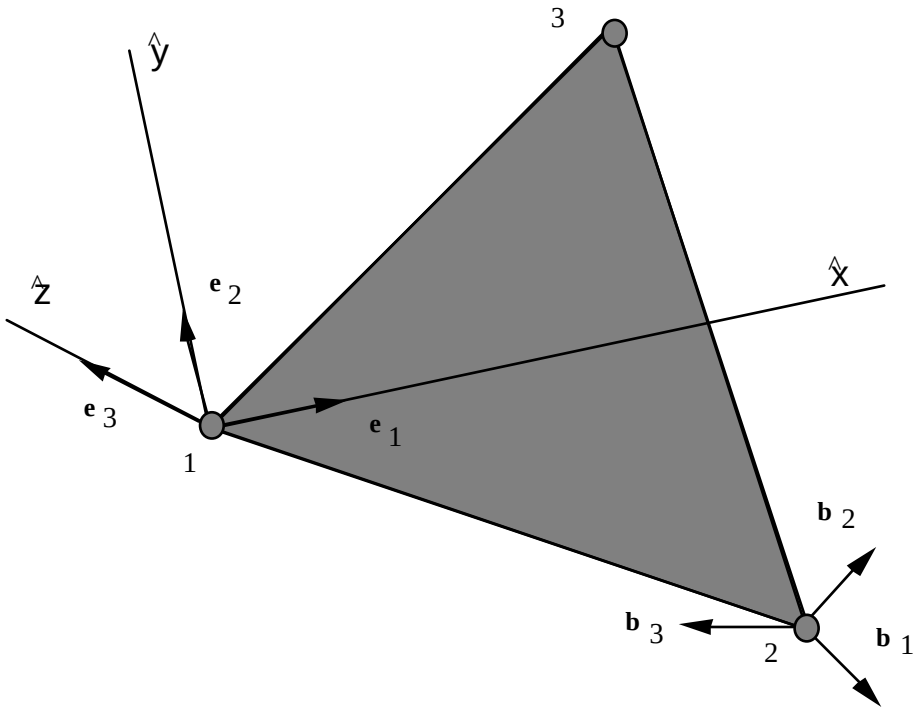
8.1 Element Coordinates

Figure 8.1(a) shows the element coordinate system, $(\hat{x}, \hat{y}, \hat{z})$ originating at Node 1, for the Marchertas-Belytschko triangular shell. The element coordinate system is associated with a triad of unit vectors ($\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$) the components of which form a trans-formation matrix between the global and local coordinate systems for vector quantities. The nodal or body coordinate system unit vectors ($\mathbf{b}_1$, $\mathbf{b}_2$, $\mathbf{b}_3$) are defined at each node and are used to define the rotational deformations in the element, see Section 4.

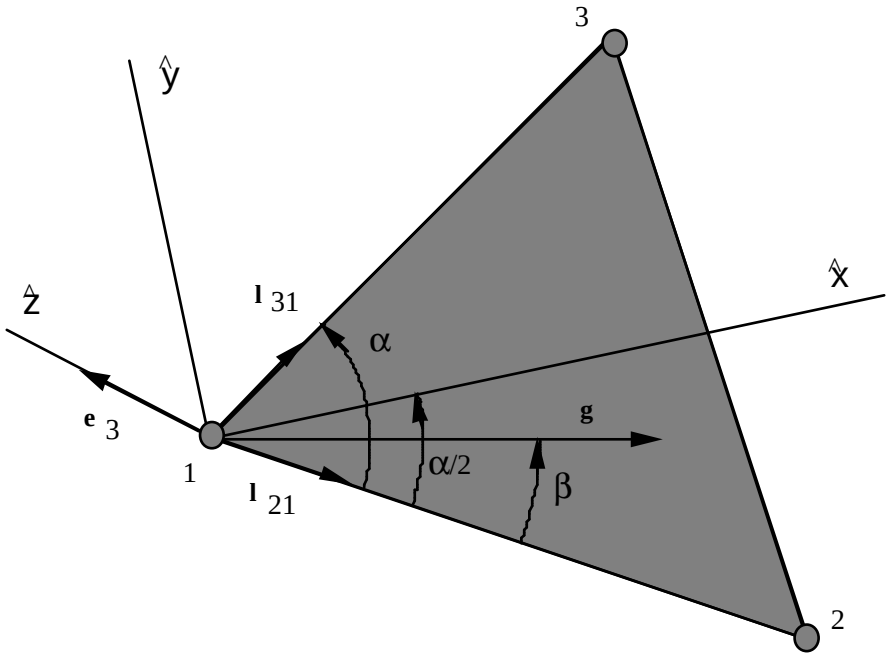
The unit normal to the shell element e_3 is formed from the vector cross product

$$\mathbf{e}_3 = l_{21} \times l_{31} \tag{8.1}$$

where l_{21} and l_{31} are unit vectors originating at Node 1 and pointing towards Nodes 2 and 3 respectively, see Figure 8.1(b).



(a) Element and Body Coordinates



(b) Construction of Element Coordinates

Figure 8.1 Construction of local element coordinate system.

Next a unit vector **g**, see Figure 8.1(b), is assumed to be in the plane of the triangular element with its origin at Node 1 and forming an angle β with the element side between Nodes 1 and 2 i.e. the vector l_{21}. The direction cosines of this unit vector are represented by the symbols (g_x g_y, g_z). Since **g** is the unit vector, its direction cosines will satisfy the equation

$$g_x^2 + g_y^2 + g_z^2 = 1 \tag{8.2}$$

Also, since **g** and $\mathbf{e}_3$ are orthogonal unit vectors, their vector dot product must satisfy the equation

$$e_{3x}g_x + e_{3y}g_y + e_{3z}g_z = 0 \tag{8.3}$$

In addition, the vector dot product of the co-planar unit vectors **g** and l_{21} satisfies the equation

$$I_{21x}g_x + I_{21y}g_y + I_{21z}g_z = \cos\beta \tag{8.4}$$

where (l_{21x}, l_{21y}, l_{21z}) are the direction cosines of $\mathbf{l}_{21}$.

Solving this system of three simultaneous equation, i.e. Equation (8.2), (8.3), and (8.4), for the direction cosines of the unit vector g yields

$$\begin{aligned} g_x &= l_{21x}\cos\beta + (e_{3y}l_{21z} - e_{3z}l_{21y})\sin\beta \\ g_y &= l_{21y}\cos\beta + (e_{3z}l_{21x} - e_{3x}l_{21z})\sin\beta \\ g_z &= l_{21z}\cos\beta + (e_{3x}l_{21y} - e_{3y}l_{21x})\sin\beta \end{aligned} \tag{8.5}$$

These equations provide the direction cosines for any vector in the plane of the triangular element that is oriented at an angle β from the element side between Nodes 1 and 2. Thus the unit vector components of $\mathbf{e}_1$, and $\mathbf{e}_2$ are obtained by setting $\beta = \alpha/2$ and $\beta = (\pi + \alpha)/2$ in Equation (8.5), respectively. The angle α is obtained from the vector dot product of the unit vectors l_{21} and l_{31},

$$\cos\alpha = l_{21} \_ l_{31} \tag{8.6}$$

8.2 Displacement Interpolation

As with the other large displacement and small deformation co-rotational element formulations, the nodal displacements are separated into rigid body and deformation displacements,

$$\mathbf{u} = \mathbf{u}^{\text{rigid}} + \mathbf{u}^{\text{def}} \tag{8.7}$$

where the rigid body displacements are defined by the motion of the local element coordinate system, i.e. the co-rotational coordinates, and the deformation displacement are defined with respect to the co-rotational coordinates. The deformation displacement are defined by

$$\begin{Bmatrix} \hat{u}_x \\ \hat{u}_y \\ --- \\ \hat{u}_z \end{Bmatrix}^{def} = \begin{bmatrix} \phi_x^m \\ \phi_y^m \\ --- \\ \phi_z^f \end{bmatrix} \begin{Bmatrix} \delta \\ --- \\ \hat{\theta} \end{Bmatrix} \tag{8.8}$$

where

$$\{\delta\}^T = \{\delta_{12}\ \delta_{23}\ \delta_{31}\} \tag{8.9}$$

are the edge elongations and

$$\left\{\hat{\theta}\right\} = \left\{\hat{\theta}_{1x}\hat{\theta}_{1y}\hat{\theta}_{2x}\hat{\theta}_{2y}\hat{\theta}_{3x}\hat{\theta}_{3y}\right\} \tag{8.10}$$

are the local nodal rotation with respect to the co-rotational coordinates.

The matrices ϕ_x^m, ϕ_y^m and ϕ_z^f are the membrane and flexural interpolation functions, respectively. The element's membrane deformation is defined in terms of the edge elongations. Marchertas and Belytschko adapted this idea from Argyris et.al. [1964], where incremental displacements are used, by modifying the relations for total displacements,

$$\delta_{ij} = \frac{2\left(x_{ji}u_{jix} + y_{ji}u_{jiy} + z_{ji}u_{jiz}\right) + u_{jix}^2 + u_{jiy}^2 + u_{jiz}^2}{l_{ij}^0 + l_{ij}} \tag{8.11}$$

where $x_{ji} = x_j - x_i$, etc.

The non conforming shape functions used for interpolating the flexural deformations, ϕ_z^f were originally derived by Bazeley, Cheung, Irons, and Zienkiewicz [1965]; hence the LS-DYNA3D reference to the BCIZ element. Explicit expressions for ϕ_z^f are quite tedious and are not given here. The interested reader is referred to Appendix G in the original work of Marchertas and Belytschko [1974].

The local nodal rotations, which are interpolated by these flexural shape functions, are defined in a manner similar to those used in the Belytschko beam element. The current components of the original element normal are obtained from the relation

$$e_3^0 = \mu^T \lambda \bar{e}_3^0 \tag{8.12}$$

where μ and λ are the current transformations between the global coordinate system and the element (local) and body coordinate system, respectively. The vector $\bar{e}_3^0$ is the original element unit normal expressed in the body coordinate system. The vector cross product between this current-original unit normal and the current unit normal,

$$e_3 \times e_3^0 = \hat{\theta}_x e_1 + \hat{\theta}_y e_2 \tag{8.13}$$

define the local nodal rotations as

$$\hat{\theta}_x = -\hat{e}_{3y}^0 \tag{8.14}$$

$$\hat{\theta}_y = \hat{e}_{3x}^0 \tag{8.15}$$

Note that at each node the corresponding λ transformation matrix is used in Equation (8.12).

8.3 Strain-Displacement Relations

Marchertas-Belytschko impose the usual Kirchhoff assumptions that normals to the midplane of the element remain straight and normal, to obtain

$$e_{xx} = \frac{\partial u_x}{\partial x} - z\frac{\partial^2 u_z}{\partial x^2} \tag{8.16a}$$

$$e_{yy} = \frac{\partial u_y}{\partial y} - z\frac{\partial^2 u_z}{\partial y^2} \tag{8.16b}$$

$$2e_{xy} = \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} - 2z\frac{\partial^2 u_z}{\partial x \partial y} \tag{8.16c}$$

where it is understood that all quantities refer to the local element coordinate system.

Substitution of Equations (8.8) into the above strain-displacement relations yields

$$\{\varepsilon\} = \left[E^m\right]\{\delta\} - z\left[E^f\right]\{\hat{\theta}\} \tag{8.17}$$

where

$$\{\varepsilon\}^T = \{\varepsilon_{xx}\varepsilon_{yy}2\varepsilon_{xy}\} \tag{8.18}$$

with

$$\left[E^m\right] = \begin{bmatrix} \dfrac{\partial \phi_{xi}^m}{\partial x} \\ \dfrac{\partial \phi_{yi}^m}{\partial y} \\ \dfrac{\partial \phi_{xi}^m}{\partial y} + \dfrac{\partial \phi_{yi}^m}{\partial x} \end{bmatrix} \tag{8.19a}$$

and

$$\left[E^f\right] = \begin{bmatrix} \dfrac{\partial^2 \phi_{zi}^f}{\partial x^2} \\ \dfrac{\partial^2 \phi_{zi}^f}{\partial y^2} \\ 2\dfrac{\partial^2 \phi_{zi}^f}{\partial x \partial y} \end{bmatrix} \tag{8.19b}$$

Again, the interested reader is referred to Appendices F and G in the original work of Marchertas and Belytschko [1974] for explicit expressions of the above two matrices.

8.4 Nodal Force Calculations

The local element forces and moments are found by integrating the local element stresses through the thickness of the element. The local nodal forces are given by

$$\hat{f} = \int \left[E^m\right]^t \hat{\sigma}\, dV \tag{8.20}$$

where

$$\hat{f}^{\ T} = \left\{f_{12}\, f_{23}\, f_{31}\right\} \tag{8.21}$$

$$\hat{\sigma}^{\ T} = \left\{\sigma_{xx}\sigma_{yy}\sigma_{xy}\right\} \tag{8.22}$$

where the side forces and stresses are understood to all be in the local convected coordinate system.

Similarly, the local moments are given by

$$\hat{m} = -\int z\left[E^f\right]^t \hat{\sigma}\, dV \tag{8.23}$$

where

$$\hat{m}^T = \left\{\hat{m}_{1x}\hat{m}_{1y}\hat{m}_{2x}\hat{m}_{2y}\hat{m}_{3x}\hat{m}_{3y}\right\} \tag{8.24}$$

The through-the-thickness integration portions of the above local force and moment integrals are usually performed with a 3 or 5 point trapezoidal integration. A three-point inplane integration is also used; it is inpart this three-point inplane integration that increases the operation count for this element over the C^0 shell, which used one-point inplane integration with hourglass stabilization.

The remaining transverse nodal forces are obtained from element equilibrium considerations. Moment equilibrium requires

$$\begin{Bmatrix}\hat{f}_{2z}\\ \hat{f}_{3z}\end{Bmatrix} = \frac{1}{2A}\begin{bmatrix}-\hat{x}_3 & \hat{y}_3\\ \hat{x}_2 & -\hat{y}_2\end{bmatrix}\begin{Bmatrix}\hat{m}_{1x}+\hat{m}_{2x}+\hat{m}_{3x}\\ \hat{m}_{1y}+\hat{m}_{2y}+\hat{m}_{3y}\end{Bmatrix} \tag{8.25}$$

where A is the area of the element. Next transverse force equilibrium provides

$$\hat{f}_{1z} = -\hat{f}_{2z} - \hat{f}_{3z} \tag{8.26}$$

The corresponding global components of the nodal forces are obtained from the following transformation

$$\begin{Bmatrix} f_{ix} \\ f_{iy} \\ f_{iz} \end{Bmatrix} = \frac{f_{ij}}{l_{ij}} \begin{Bmatrix} x_{ij} + u_{ijx} \\ y_{ij} + u_{ijy} \\ z_{ij} + u_{ijz} \end{Bmatrix} + \frac{f_{ik}}{l_{ik}} \begin{Bmatrix} x_{ik} + u_{ikx} \\ y_{ik} + u_{iky} \\ z_{ik} + u_{ikz} \end{Bmatrix} + \hat{f}_{iz} \begin{Bmatrix} e_{3x} \\ e_{3y} \\ e_{3z} \end{Bmatrix} \tag{8.27}$$

Finally, the local moments are transformed to the body coordinates using the relation

$$\begin{Bmatrix} \overline{m}_{ix} \\ \overline{m}_{iy} \\ \overline{m}_{iz} \end{Bmatrix} = \lambda^T \mu \begin{Bmatrix} \hat{m}_{ix} \\ \hat{m}_{iy} \\ \hat{m}_{iz} \end{Bmatrix} \tag{8.28}$$

9. HUGHES-LIU SHELL

The Hughes-Liu shell element formulation ([Hughes and Liu 1981a, b], [Hughes et al. 1981], [Hallquist et al. 1985]) was the first shell element implemented in LS-DYNA3D. It was selected from among a substantial body of shell element literature because the element formulation has several desirable qualities:

- it is incrementally objective (rigid body rotations do not generate strains), allowing for the treatment of finite strains that occur in many practical applications;
- it is simple, which usually translates into computational efficiency and robustness;
- it is compatible with brick elements, because the element is based on a degenerated brick element formulation. This compatibility allows many of the efficient and effective techniques developed for the DYNA3D brick elements to be used with this shell element;
- it includes finite transverse shear strains;
- a through-the-thickness thinning option (see [Hughes and Carnoy 1981]) is also available when needed in some shell element applications.

The remainder of this section reviews the Hughes-Liu shell element (referred to by Hughes and Liu as the U1 element) which is a four-node shell with uniformly reduced integration, and summarizes the modifications to their theory as it is implemented in LS-DYNA3D. A detailed discussion of these modifications, as well as those associated with the implementation of the Hughes-Liu shell element in NIKE3D, are presented in an article by Hallquist and Benson [1986].

9.1 Geometry

The Hughes-Liu shell element is based on a degeneration of the standard 8-node brick element formulation, an approach originated by Ahmad et al. [1970]. Recall from the discussion of the solid elements the isoparametric mapping of the biunit cube:

$$x(\xi, \eta, \zeta) = N_a (\xi, \eta, \zeta)\, x_a \tag{9.1}$$

$$N_a (\xi,\eta,\zeta) = \frac{(1+\xi_a\xi)(1+\eta_a\eta)(1+\zeta_a\zeta)}{8} \tag{9.2}$$

where **x** is an arbitrary point in the element, (ξ, η, ζ) are the parametric coordinates, $\mathbf{x}_a$ are the global nodal coordinates of node a, and N_a are the element shape functions evaluated at node a, i.e., (ξ_a, η_a, ζ_a) are (ξ, η, ζ) evaluated at node a.

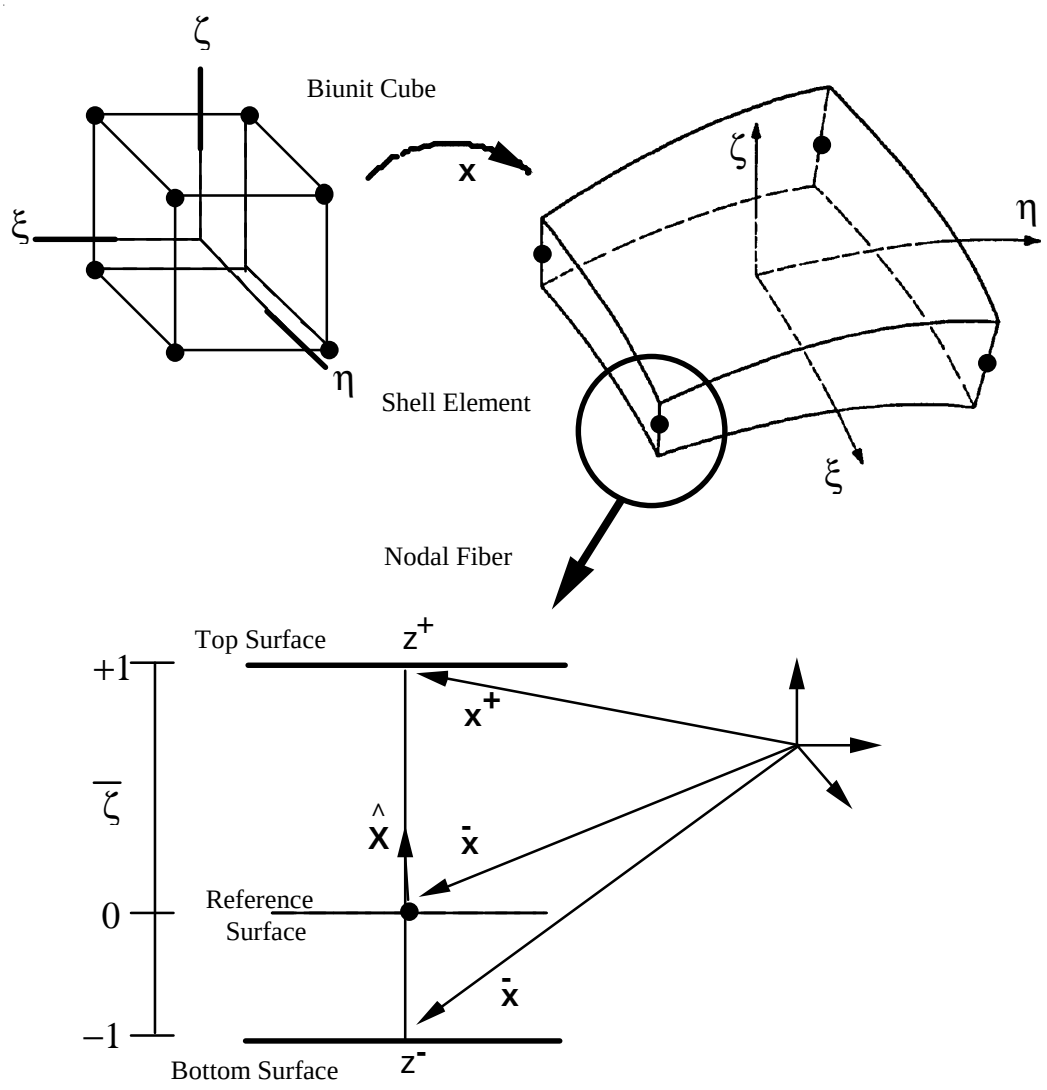


Figure 9.1. Mapping of the biunit cube into the Hughes-Liu shell element and nodal fiber nomenclature

In the shell geometry, planes of constant ζ will define the lamina or layers of the shell and fibers are defined by through-the-thickness lines when both ξ and η are constant (usually only defined at the nodes and thus referred to as 'nodal fibers'). To degenerate the 8-node brick geometry into the 4-node shell geometry, the nodal pairs in the ζ direction (through the shell thickness) are combined into a single node, for the translation degrees of freedom, and an inextensible nodal fiber for the rotational degrees of freedom. Figure 9.1 shows a schematic of the biunit cube and the shell element.

The mapping of the biunit cube into the shell element is separated into two parts

$$\mathrm{x}(\xi,\eta,\zeta) = \bar{\mathrm{x}}(\xi,\eta) + \mathrm{X}(\xi,\eta,\zeta) \tag{9.3}$$

where $\bar{\mathrm{x}}$ denotes a position vector to a point on the reference surface of the shell and X is a position vector, based at point $\bar{\mathrm{x}}$ on the reference, that defines the fiber direction through that point. In particular, if we consider one of the four nodes which define the reference surface, then

$$\bar{\mathrm{x}}(\xi,\eta) = N_a(\xi,\eta)\bar{\mathrm{x}}_a \tag{9.4}$$

$$X(\xi,\eta,\zeta) = N_a(\xi,\eta)\mathrm{X}_a(\zeta) \tag{9.5}$$

With this description, arbitrary points on the reference surface $\bar{\mathrm{x}}$ are interpolated by the two-dimensional shape function N(ξ, η) operating on the global position of the four shell nodes that define the reference surfaces, i.e., $\bar{\mathrm{x}}_a$. Points off the reference surface are further interpolated by using a one-dimensional shape function along the fiber direction, i.e., $\mathbf{X}_a(\zeta)$, where

$$X_a(\zeta) = z_a(\zeta)\hat{X}_a \tag{9.6a}$$

$$z_a(\zeta) = N_+(\zeta)z_a^+ + N_-(\zeta)z_a^- \tag{9.6b}$$

$$N_+(\zeta) = \frac{(1+\zeta)}{2} \tag{9.6c}$$

$$N_-(\zeta) = \frac{(1-\zeta)}{2} \tag{9.6d}$$

As shown in the lower portion of Figure 9.1, $\hat{X}_a$ is a unit vector in the fiber direction and z(ζ) is a thickness function. (Thickness changes (see [Hughes and Carnoy 1981]) are

accounted for by explicitly adjusting the fiber lengths at the completion of a time step based on the amount of straining in the fiber direction. Updates of the fiber lengths always lag one time step behind other kinematical quantities.)

The reference surface may be located at the midsurface of the shell or at either of the shell's outer surfaces. This capability is useful in several practical situations involving contact surfaces, connection of shell elements to solid elements, and offsetting elements such as stiffeners in stiffened shells. The reference surface is located within the shell element by specifying the value of the parameter $\bar{\zeta}$ (see lower portion of Figure 9.1). When $\bar{\zeta}$ = –1, 0, +1, the reference surface is located at the bottom, middle, and top surface of the shell, respectively.

The Hughes-Liu formulation uses two position vectors, in addition to x_a^+, to locate the reference surface and define the initial fiber direction. The two position vectors x_a^+ and x_a^- are located on the top and bottom surfaces, respectively, at node a. From these data the following are obtained:

$$\bar{x}_a = \frac{1}{2}\left(1-\bar{\zeta}\right)x_a^- + \left(1+\bar{\zeta}\right)x_a^+ \tag{9.7a}$$

$$\hat{X}_a = \frac{\left(x_a^+ - x_a^-\right)}{h_a} \tag{9.7b}$$

$$z_a^+ = \frac{1}{2}\left(1-\bar{\zeta}\right)h_a \tag{9.7c}$$

$$z_a^- = -\frac{1}{2}\left(1+\bar{\zeta}\right)h_a \tag{9.7d}$$

$$h_a = \left\| x_a^+ - x_a^- \right\| \tag{9.7e}$$

where $\|\cdot\|$ is the Euclidean norm.

9.2 Kinematics

The same parametric representation used to describe the geometry of the shell element, i.e. reference surface and fiber vector interpolation, are used to interpolate the shell element displacement, i.e. an isoparametric representation. Again the displacements are separated into the reference surface displacements and rotations associated with the fiber direction:

$$u(\xi,\eta,\zeta) = \bar{u}(\xi,\eta) + U(\xi,\eta,\zeta) \tag{9.8a}$$

$$\bar{u}(\xi,\eta) = N_a(\xi,\eta)\bar{u}_a \tag{9.8b}$$

$$U(\xi,\eta,\zeta) = N_a(\xi,\eta)U_a(\zeta) \tag{9.8c}$$

$$U_a(\zeta) = z_a(\zeta)\hat{U}_a \tag{9.8d}$$

where **u** is the displacement of a generic point; $\bar{u}$ is the displacement of a point on the reference surface, and **U** is the 'fiber displacement' rotations; the motion of the fibers can be interpreted as either displacements or rotations as will be discussed.

Hughes and Liu introduce the notation that follows, and the associated schematic shown in Figure 9.2, to describe the current deformed configuration with respect to the reference configuration:

$$y = \bar{y} + Y \tag{9.9a}$$

$$\bar{y} = \bar{x} + \bar{u} \tag{9.9b}$$

$$\bar{y}_a = \bar{x}_a + \bar{u}_a \tag{9.9c}$$

$$\mathrm{Y} = \mathrm{X} + \mathrm{U} \tag{9.9d}$$

$$\mathrm{Y_a} = \mathrm{X_a} + \mathrm{U_a} \tag{9.9e}$$

$$\hat{Y}_a = \hat{X}_a + \hat{U}_a \tag{9.9f}$$

In the above relations, and in Figure 9.2, the **x** quantities refer to the reference configuration, the **y** quantities refer to the updated (deformed) configuration and the **u** quantities are the displacements. The notation consistently uses a superscript bar $(\bar{\cdot})$ to indicate reference surface quantities, a superscript caret $(\hat{\cdot})$ to indicate unit vector quantities, lower case letters for translational displacements, and upper case letters indicating fiber displacements. To update to the deformed configuration, two vector quantities are needed: the reference surface displacement $\bar{u}$ and the associated nodal fiber displacement **U**. The nodal fiber displacements are defined in the fiber coordinate system, described in the next subsection.

9.2.1 Fiber Coordinate System

For a shell element with four nodes, the known quantities will be the displacements of the reference surface $\bar{u}$ obtained from the translational equations of motion and some rotational quantities at each node obtained from the rotational equations of motion. To complete the kinematics, we now need a relation between nodal rotations and fiber displacements **U**.

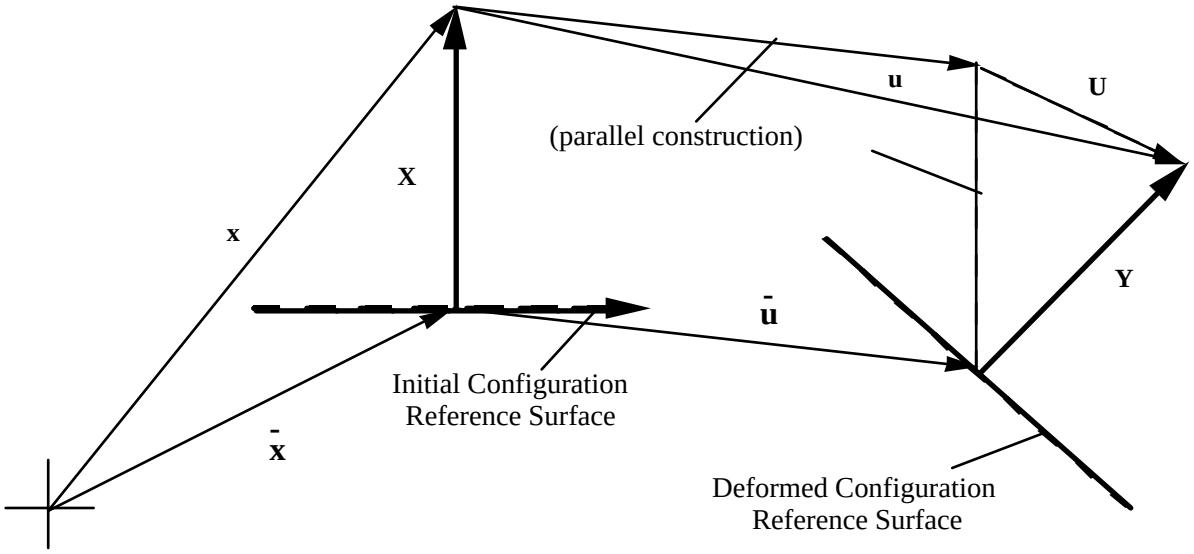


Figure 9.2. Schematic of deformed configuration displacements and position vectors.

At each node a unique local Cartesian coordinate system is constructed that is used as the reference frame for the rotation increments. The relation presented by Hughes and Liu for the nodal fiber displacements (rotations) is an incremental relation, i.e. it relates the current configuration to the last state, not to the initial configuration. Figure 9.3 shows two triads of unit vectors: $\left(b_1^f, b_2^f, b_3^f\right)$ comprising the orthonormal fiber basis in the reference configuration (where the fiber unit vector is now $\hat{Y} = b_3^f$) and (b_1, b_2, b_3) indicating the incrementally updated current configuration of the fiber vectors. The reference triad is updated by applying the incremental rotations, $\Delta\theta_1$ and $\Delta\theta_2$, obtained from the rotational equations of motion, to the fiber vectors $\left(b_1^f \text{ and } b_2^f\right)$ as shown in Figure 9.3. The linearized relationship between the components of ΔU in the fiber system viz, $\Delta\hat{U}_1^f, \Delta\hat{U}_2^f, \Delta\hat{U}_3^f$, and the incremental rotations is given by

$$\begin{Bmatrix} \Delta\hat{U}_1^f \\ \Delta\hat{U}_2^f \\ \Delta\hat{U}_3^f \end{Bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \end{Bmatrix} \tag{9.10}$$

Although the above Hughes-Liu relation for updating the fiber vector enables a reduction in the number of nodal degrees of freedom from six to five, it is not implemented in LS-DYNA3D because it is not applicable to beam elements.

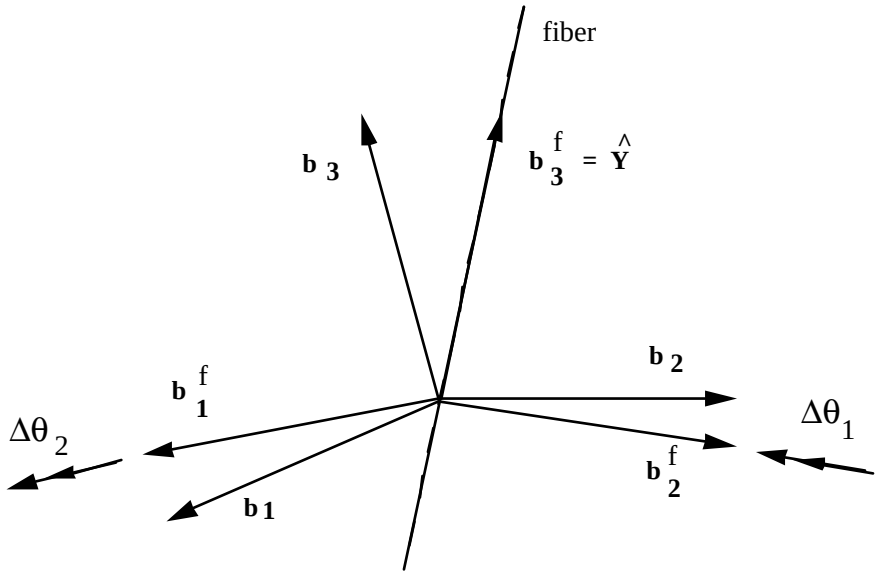


Figure 9.3. Incremental update of fiber vectors using Hughes-Liu incremental rotations.

In LS-DYNA3D and LS-NIKE3D, three rotational increments are used, defined with reference to the global coordinate axes:

$$\begin{Bmatrix} \Delta\hat{U}_1 \\ \Delta\hat{U}_2 \\ \Delta\hat{U}_3 \end{Bmatrix} = \begin{bmatrix} 0 & \hat{Y}_3 & -\hat{Y}_2 \\ -\hat{Y}_3 & 0 & \hat{Y}_1 \\ \hat{Y}_2 & -\hat{Y}_1 & 0 \end{bmatrix} \begin{Bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \Delta\theta_3 \end{Bmatrix} \tag{9.11}$$

Equation (9.11) is adequate for updating the stiffness matrix in NIKE3D, but for finite rotations the error is significant. A more accurate second-order technique is used in both DYNA3D and NIKE3D for updating the unit fiber vectors:

$$\hat{Y}_i^{\,n+1} = R_{ij}(\Delta\theta)\hat{Y}_i^{\,n} \tag{9.12}$$

where

$$R_{ij}(\Delta\theta) = \delta_{ij} + \frac{1}{2}\frac{\left(2\delta_{ij} + \Delta S_{ik}\right)\Delta S_{ik}}{D} \tag{9.13a}$$

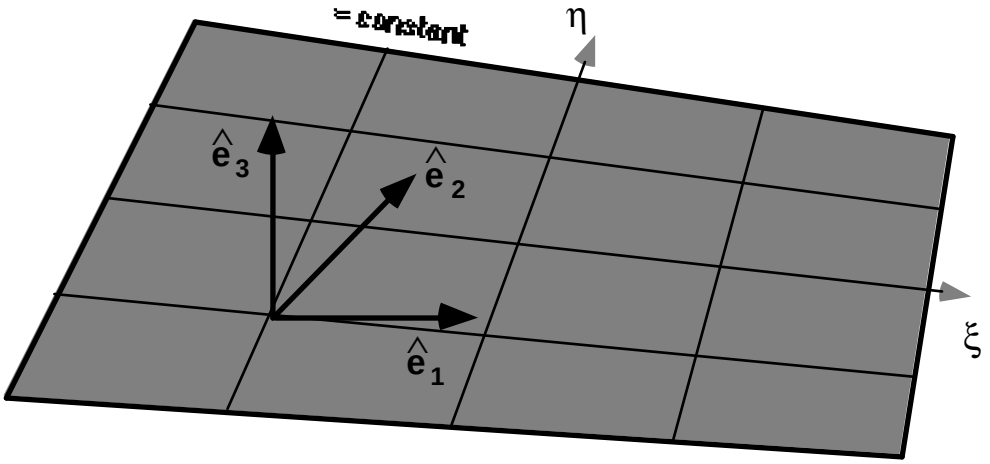
$$\Delta S_{ij} = e_{ikj}\Delta\theta_k \tag{9.13b}$$

$$2D = 2 + \frac{1}{2}\left(\Delta\theta_1^2 + \Delta\theta_2^2 + \Delta\theta_3^2\right) \tag{9.13c}$$

Here δ_{ij} is the Kronecker delta and e_{ikj} is the permutation tensor. This rotational update is often referred to as the Hughes-Winget formula [Hughes and Winget 1980]. An exact rotational update using Euler angles or Euler parameters could easily be substituted in Equation (9.12), but it is doubtful that the extra effort would be justified.

9.2.2 Lamina Coordinate System

In addition to the above described fiber coordinate system, a local lamina coordinate system is needed to enforce the zero normal stress condition, i.e. plane stress. Lamina are layers through the thickness of the shell that correspond to the locations and associated thicknesses of the through-the-thickness shell integration points; the analogy is that of lamina in a fibrous composite material. The orthonormal lamina basis (Figure 9.4), with one direction $\hat{e}_3$ normal to the lamina of the shell, is constructed at every integration point in the shell.



The lamina basis is constructed by forming two unit vectors locally tangent to the lamina:

$$\hat{e}_1 = \frac{y,\xi}{\left\| y,\xi \right\|} \tag{9.14}$$

$$e_2' = \frac{y,\eta}{\left\| y,\eta \right\|} \tag{9.15}$$

where as before y is the position vector in the current configuration. The normal to the lamina at the integration point is constructed from the vector cross product of these local tangents:

$$\hat{e}_3 = \hat{e}_1 \times e_2' \tag{9.16}$$

To complete this orthonormal lamina basis, the vector

$$\hat{e}_2 = \hat{e}_3 \times \hat{e}_1 \tag{9.17}$$

is defined, because $\hat{e}_2$, although tangent to both the lamina and lines of constant ξ, may not be normal to $\hat{e}_1$ and $\hat{e}_3$. The lamina coordinate system rotates rigidly with the element.

The transformation of vectors from the global to lamina coordinate system can now be defined in terms of the lamina basis vectors as

$$\hat{A} \tag{9.18}$$

where e_{ix}, e_{iy}, e_{iz} are the global components of the lamina coordinate unit vectors; $\hat{A}$ is a vector in the lamina coordinates, and **A** is the same vector in the global coordinate system.

9.3 Strains and Stress Update

9.3.1 Incremental Strain and Spin Tensors

The strain and spin increments are calculated from the incremental displacement gradient

$$G_{ij} = \frac{\partial \Delta u_i}{\partial y_j} \tag{9.19}$$

where Δu_i are the incremental displacements and y_j are the deformed coordinates. The incremental strain and spin tensors are defined as the symmetric and skew-symmetric parts, respectively, of G_{ij}:

$$\Delta\varepsilon_{ij} = \frac{1}{2}\left(G_{ij} + G_{ji}\right) \tag{9.20}$$

$$\Delta\omega_{ij} = \frac{1}{2}\left(G_{ij} - G_{ji}\right) \tag{9.21}$$

The incremental spin tensor $\Delta\omega_{ij}$ is used as an approximation to the rotational contribution of the Jaumann rate of the stress tensor; NIKE3D uses the more accurate Hughes-Winget transformation matrix (Equation (9.12)) with the incremental spin tensor for the rotational update. The Jaumann rate update is approximated as:

$$\underline{\sigma}_{ij}^{n+1} = \sigma_{ij}^{n} + \sigma_{ip}^{n}\Delta\omega_{pj} + \sigma_{jp}^{n}\Delta\omega_{pi} \tag{9.22}$$

where the superscripts on the stress refer to the updated (n+1) and reference (n) configurations. The Jaumann rate update of the stress tensor is applied in the global configuration before the constitutive evaluation is performed. In the Hughes-Liu shell the stresses and history variables are stored in the global coordinate system.

9.3.2 Stress Update

To evaluate the constitutive relation, the stresses and strain increments are rotated from the global to the lamina coordinate system using the transformation defined previously in Equation (9.18), viz.

$$\underline{\sigma}_{ij}^{l\,n+1} = q_{ik}\,\underline{\sigma}_{kn}^{n+1} q_{jn} \tag{9.23}$$

$$\Delta\varepsilon_{ij}^{l\,n+1/2} = q_{ik}\Delta\varepsilon_{kn}^{n+1/2} q_{jn} \tag{9.24}$$

where the superscript l indicates components in the lamina (local) coordinate system.

The stress is updated incrementally:

$$\sigma_{ij}^{l\,n+1} = \underline{\sigma}_{ij}^{l\,n+1} + \Delta\sigma_{ij}^{l\,n+1/2} \tag{9.25}$$

and rotated back to the global system:

$$\sigma_{ij}^{n+1} = q_{ki}\sigma_{kn}^{l^{n+1}} q_{nj} \tag{9.26}$$

before computing the internal force vector.

9.3.3 Incremental Strain-Displacement Relations

The global stresses are now used to update the internal force vector

$$f_a^{\text{int}} = \int T_a^T B_a^T \sigma d\upsilon \tag{9.27}$$

where f_a^{int} are the internal forces at node a, $\mathbf{B}_a$ is the strain-displacement matrix in the lamina coordinate system associated with the displacements at node a, and $\mathbf{T}_a$ is the transformation matrix relating the global and lamina components of the strain-displacement matrix. Because the **B** matrix relates six strain components to twenty-four displacements (six degrees of freedom at four nodes), it is convenient to partition the **B** matrix into four groups of six:

$$\mathrm{B} = [\mathrm{B}_1,\ \mathrm{B}_2,\ \mathrm{B}_3,\ \mathrm{B}_4] \tag{9.28a}$$

Each $\mathbf{B}_a$ submatrix is further partitioned into a portion due to strain and spin:

$$B_a = \begin{bmatrix} B_a^{\varepsilon} \\ B_a^{\omega} \end{bmatrix} \tag{9.28b}$$

with the following submatrix definitions:

$$B_a^{\varepsilon} = \begin{bmatrix} B_1 & 0 & 0 & B_4 & 0 & 0 \\ 0 & B_2 & 0 & 0 & B_5 & 0 \\ \overline{B}_2 & \overline{B}_1 & 0 & \overline{B}_5 & \overline{B}_4 & 0 \\ 0 & \overline{B}_3 & \overline{B}_2 & 0 & \overline{B}_6 & \overline{B}_5 \\ \overline{B}_3 & 0 & \overline{B}_1 & \overline{B}_6 & 0 & \overline{B}_4 \end{bmatrix} \tag{9.28c}$$

$$B_a^{\omega} = \begin{bmatrix} \overline{B}_2 & -\overline{B}_1 & 0 & \overline{B}_5 & -\overline{B}_4 & 0 \\ 0 & \overline{B}_3 & -\overline{B}_2 & 0 & \overline{B}_6 & -\overline{B}_5 \\ -\overline{B}_3 & 0 & \overline{B}_1 & -\overline{B}_6 & 0 & \overline{B}_4 \end{bmatrix} \tag{9.28d}$$

where

$$B_i = \begin{cases} N_{a,i} = \dfrac{\partial N_a}{\partial y_i^l} & \text{for } i = 1,2,3 \\ \left(N_a z_a\right)_{,i-3} = \dfrac{\partial \left(N_a z_a\right)}{\partial y_{i-3}^l} & \text{for } i = 4,5,6 \end{cases}$$

Notes on strain-displacement relations:

- The derivatives of the shape functions are taken with respect to the lamina coordinate system, e.g. $y^l = qy$.

- The superscript bar indicates the *B*'s are evaluated at the center of the lamina (0, 0, ζ), The strain-displacement matrix uses the 'B-Bar' $\left(\overline{B}\right)$ approach advocated by Hughes [1980]. In the NIKE3D and DYNA3D implementations, this entails replacing certain rows of the **B** matrix and the strain increments with their counterparts evaluated at the center of the element. In particular, the strain-displacement matrix is modified to produce constant shear and spin increments throughout the lamina.
- The resulting B-matrix is a 8×24 matrix. Although there are six strain and three rotations increments, the **B** matrix has been modified to account for the fact that σ_{33} will be zero in the integration of Equation (9.27).

9.4 Element Mass Matrix

Hughes, Liu, and Levit [Hughes et al. 1981] describe the procedure used to form the shell element mass matrix in problems involving explicit transient dynamics. Their procedure, which scales the rotary mass terms, is used for all shell elements in LS--DYNA3D including those formulated by Belytschko and his co-workers. This scaling permits large critical time step sizes without loss of stability.

The consistent mass matrix is defined by

$$M = \int_{\upsilon_m} \rho \mathrm{N}^t \mathrm{N} \, d\upsilon_m \tag{9.29}$$

but cannot be used effectively in explicit calculations where matrix inversions are not feasible. In LS-DYNA3D only three- and four-node shell elements are used with linear

interpolation functions; consequently, we compute the translational masses from the consistent mass matrix by row summing, leading to the following mass at element node a:

$$M_{disp_a} = \int_v \rho \phi_a \, d\upsilon \tag{9.30}$$

The rotational masses are computed by scaling the translational mass at the node by the factor α:

$$M_{rot_a} = \propto M_{disp_a} \tag{9.31}$$

where

$$\propto = \max\{\propto_1, \propto_2\} \tag{9.32}$$

$$\propto_1 = \langle z_a \rangle^2 + \frac{1}{12}[z_a]^2 \tag{9.33}$$

$$\propto_2 = \frac{V}{8h} \tag{9.34}$$

$$\langle z_a \rangle = \frac{\left(z_a^+ + z_a^-\right)}{2} \tag{9.35}$$

$$[z_a] = z_a^+ - z_a^- \tag{9.36}$$

and V and h are the volume and the thickness of the element, respectively.

9.5 Accounting for Thickness Changes

Hughes and Carnoy [1981] describe the procedure used to update the shell thickness due to large membrane stretching. Their procedure with any necessary modifications is used across all shell element types in LS-DYNA3D. One key to updating the thickness is an accurate calculation of the normal strain component $\Delta\varepsilon_{33}$. This strain component is easily obtained for elastic materials but can require an iterative algorithm for nonlinear material behavior. In LS-DYNA3D we therefore default to an iterative plasticity update to accurately determine $\Delta\varepsilon_{33}$.

Hughes and Carnoy integrate the strain tensor through the thickness of the shell in order to determine a mean value $\Delta\bar{\varepsilon}_{ij}$:

$$\Delta\bar{\varepsilon}_{ij} = \frac{1}{2}\int_{-1}^{1} \Delta\varepsilon_{ij}\, d\zeta \tag{9.37}$$

and then project it to determine the straining in the fiber direction:

$$\bar{\varepsilon}^{\,f} = \hat{Y}^T \Delta\bar{\varepsilon}_{ij} \hat{Y} \tag{9.38}$$

Using the interpolation functions through the integration points the strains in the fiber directions are extrapolated to the nodal points if 2×2 selectively reduced integration is employed. The nodal fiber lengths can now be updated:

$$h_a^{n+1} = h_a^n \left(1 + \bar{\varepsilon}_a^f\right) \tag{9.39}$$

9.6 Fully Integrated Hughes-Liu Shells

It is well known that one-point integration results in zero energy modes that must be resisted. The four-node under integrated shell with six degrees of freedom per node has nine zero energy modes, six rigid body modes, and four unconstrained drilling degrees of freedom . Deformations in the zero energy modes are always troublesome but usually not a serious problem except in regions where boundary conditions such as point loads are active. In areas where the zero energy modes are a problem, it is highly desirable to provide the option of using the original formulation of Hughes-Liu with selectively reduced integration as shown in Figure 9.5.

The major disadvantages of full integration are twofold:

- nearly four times as much data must be stored;
- the operation count increases three- to fourfold. The level 3 loop is added as shown in Figure 9.6.

However these disadvantages can be more than offset by the increased reliability and accuracy.

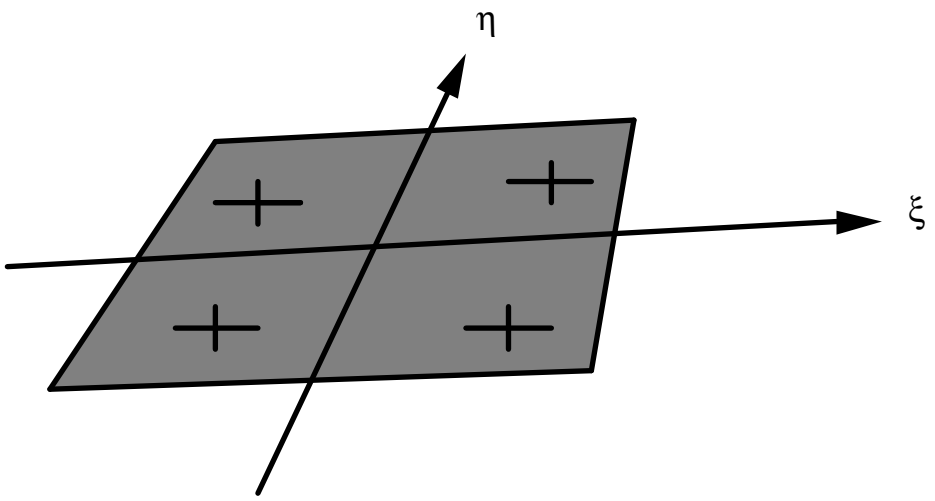


Figure 9.5. Selectively reduced integration rule results in four inplane points being used.

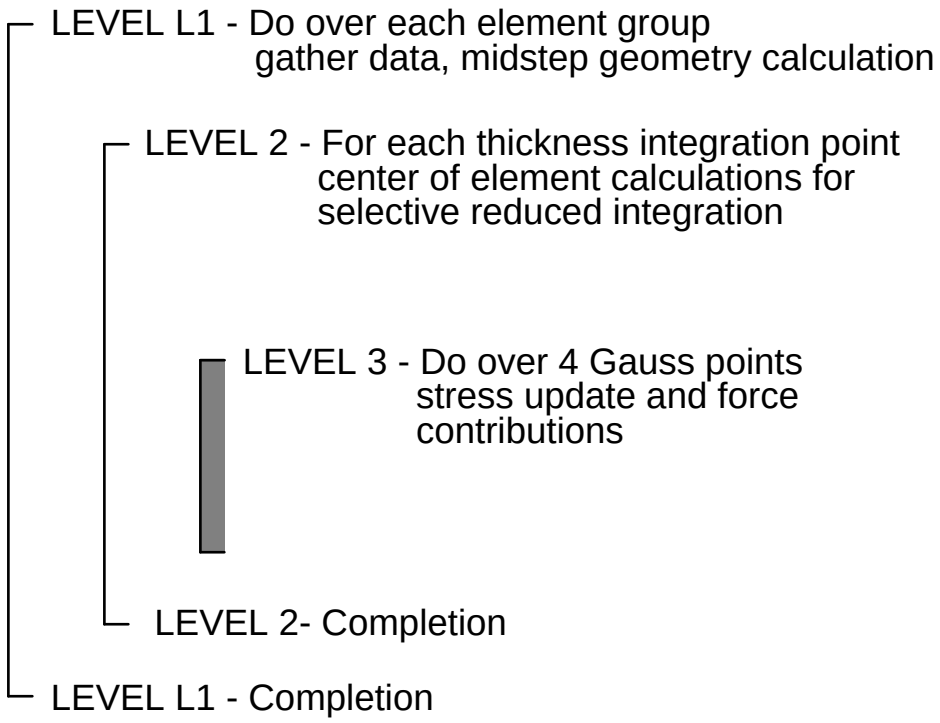


Figure 9.6. An inner loop, LEVEL 3, is added for the Hughes-Liu shell with selectively reduced integration.

LEVEL L1 - Once per element	
Midstep translation geometry, etc.	204
Midstep calculation of Y	318
LEVEL L2 - For each integration point through thickness (NT points)	
Strain increment at $(0, 0, \zeta)$	316
Hughes-Winget rotation matrix	33
Square root of Hughes-Winget matrix	47
Rotate strain increments into lamina coordinates	66
Calculate rows 3-8 of B matrix	919
LEVEL L3 - For each integration point in lamina	
Rotate stress to n+1/2 configuration	75
Incremental displacement gradient matrix	370
Rotate stress to lamina system	75
Rotate strain increments to lamina system	55
Constitutive model	model dependent
Rotate stress back to global system	69
Rotate stress to n+1 configuration	75
Calculate rows 1 and 2 of B matrix	358
Stresses in n+1 lamina system	75
Stress divergence	245
TOTAL	**522 +NT {1381 +4 * 1397}**

Table 9.1. Operation counts for the Hughes-Liu shell with selectively reduced integration.

We have implemented two version of the Hughes-Liu shell with selectively reduced integration. The first closely follows the intent of the original paper, and therefore no assumptions are made to reduce costs which are outlined in operation counts in Table 9.1. These operation counts can be compared with those in Table 9.2 for the Hughes-Liu shell with uniformly reduced integration. The second formulation, which reduces the number of operation by more than a factor of two, is referred to as the co-

LEVEL L1 - Once per element	
Calculate displacement increments	24
Element areas for time step	53
Calculate Y	238
LEVEL L2 and L3 - Integration point through thickness (NT points)	
Incremental displacement gradient matrix	284
Jaumann rotation for stress	33
Rotate stress into lamina coordinates	75
Rotate stain increments into lamina coordinates	81
Constitutive model	model dependent
Rotate stress to n+1 global coordinates	69
Stress divergence	125
LEVEL L1 - Cleanup	
Finish stress divergence	60
Hourglass control	356
TOTAL	**731 +NT * 667**

Table 9.2. Operation counts for DYNA3D implementation of the uniformly reduced Hughes-Liu shell.

rotational Hughes-Liu shell in the LS-DYNA3D user's manual. This shell is considerably cheaper due to the following simplifications:

- Strains rates are not centered. The strain displacement matrix is only computed at time n+1 and not at time $n+^1/_2$.
- The stresses are stored in the local shell system following the Belytschko-Tsay shell. The transformations of the stresses between the local and global coordinate systems are thus avoided.
- The Jaumann rate rotation is not performed, thereby avoiding even more computations. This does not necessarily preclude the use of the shell in large deformations.

To study the effects of these simplifying assumptions, we can compare results with those obtained with the full Hughes-Liu shell. Thus far, we have been able to get comparable results.

10. EIGHT-NODE SOLID SHELL ELEMENT

The isoparametric eight-node brick element discussed in Section 3 forms the basis for this shell element with enhancements based on the Hughes-Liu and the Belytschko-Lin-Tsay shells. Like the eight-node brick, the geometry is interpolated from:

$$x_i(X_\alpha, t) = x_i\left(X_\alpha(\xi,\eta,\zeta), t\right) = \sum_{j=1}^{8} \phi_j(\xi,\eta,\zeta) x_i^j(t) \tag{10.1}$$

where the shape function ϕ_j are defined as

$$\phi_j = \frac{1}{8}(1+\xi\xi_j)(1+\eta\eta_j)(1+\zeta\zeta_j) \tag{10.2}$$

where ξ_j, η_j, ζ_j take on their nodal values of (±1, ±1, ±1) and x_i^j is the nodal coordinate of the jth node in the ith direction (Figure 3.1).

As with solid elements, **N** is the 3×24 rectangular interpolation matrix:

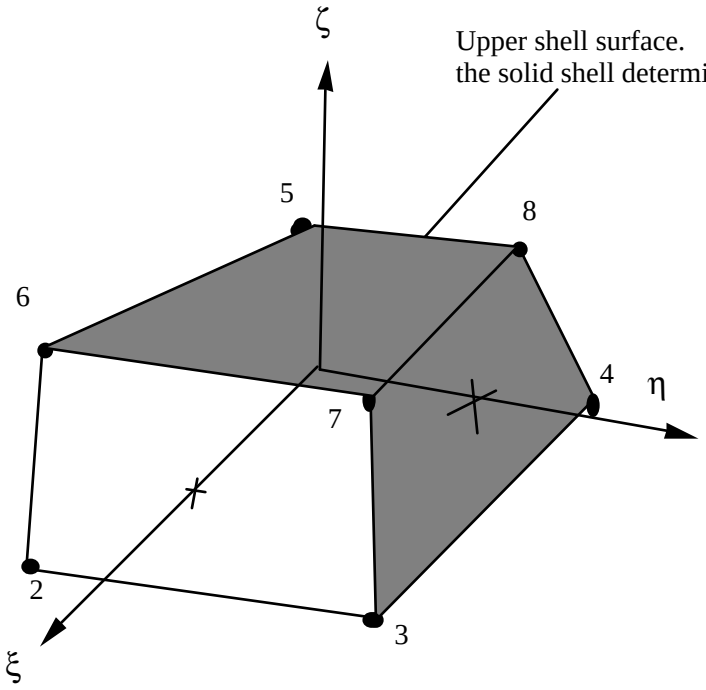
$$\mathrm{N}(\xi,\eta,\zeta) = \begin{bmatrix} \phi_1 & 0 & 0 & \phi_2 & 0 & \dots & 0 & 0 \\ 0 & \phi_1 & 0 & 0 & \phi_2 & \dots & \phi_8 & 0 \\ 0 & 0 & \phi_1 & 0 & 0 & \dots & 0 & \phi_8 \end{bmatrix} \tag{10.3}$$

σ is the stress vector:

$$\sigma^t = (\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{yz}, \sigma z_x) \tag{10.4}$$

and **B** is the 6×24 strain-displacement matrix:

$$\mathrm{B} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \end{bmatrix} \mathrm{N} \tag{10.5}$$



Node	ξ	η	ζ
1	-1	-1	-1
2	1	-1	-1
3	1	1	-1
4	-1	1	-1
5	-1	-1	1
6	1	-1	1
7	1	1	1
8	-1	1	1

Figure 10.1. Eight node solid shell element.

Terms in the strain-displacement matrix are readily calculated. Note that

$$\begin{aligned}
\frac{\partial \phi_i}{\partial \xi} &= \frac{\partial \phi_i}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial \phi_i}{\partial y}\frac{\partial y}{\partial \xi} + \frac{\partial \phi_i}{\partial z}\frac{\partial z}{\partial \xi} \\
\frac{\partial \phi_i}{\partial \eta} &= \frac{\partial \phi_i}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial \phi_i}{\partial y}\frac{\partial y}{\partial \eta} + \frac{\partial \phi_i}{\partial z}\frac{\partial z}{\partial \eta} \\
\frac{\partial \phi_i}{\partial \zeta} &= \frac{\partial \phi_i}{\partial x}\frac{\partial x}{\partial \zeta} + \frac{\partial \phi_i}{\partial y}\frac{\partial y}{\partial \zeta} + \frac{\partial \phi_i}{\partial z}\frac{\partial z}{\partial \zeta}
\end{aligned} \tag{10.6}$$

which can be rewritten as

$$\begin{bmatrix} \frac{\partial \phi_i}{\partial \xi} \\ \frac{\partial \phi_i}{\partial \eta} \\ \frac{\partial \phi_i}{\partial \zeta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} \begin{bmatrix} \frac{\partial \phi_i}{\partial x} \\ \frac{\partial \phi_i}{\partial y} \\ \frac{\partial \phi_i}{\partial z} \end{bmatrix} = J \begin{bmatrix} \frac{\partial \phi_i}{\partial x} \\ \frac{\partial \phi_i}{\partial y} \\ \frac{\partial \phi_i}{\partial z} \end{bmatrix} \tag{10.7}$$

Inverting the Jacobian matrix, **J**, we can solve for the desired terms

$$\begin{bmatrix} \frac{\partial \phi_i}{\partial x} \\ \frac{\partial \phi_i}{\partial y} \\ \frac{\partial \phi_i}{\partial z} \end{bmatrix} = J^{-1} \begin{bmatrix} \frac{\partial \phi_i}{\partial \xi} \\ \frac{\partial \phi_i}{\partial \eta} \\ \frac{\partial \phi_i}{\partial \zeta} \end{bmatrix} \tag{10.8}$$

To obtain shell-like behavior from the solid element, it is necessary to use multiple integration points through the shell thickness along the ζ axis while employing a plane stress constitutive subroutine. Consequently, it is necessary to construct a reference surface within the brick shell. We locate the reference surface midway between the upper and lower surfaces and construct a local coordinate system exactly as was done for the Belytschko-Lin-Tsay shell element. Following the procedure outlined in Section 6, Equations (6.1) – (6.3), the local coordinate system can be constructed as depicted in Figure 10.2. Equation (6.5a) gives the transformation matrix in terms of the local basis:

$$\{A\} = \begin{Bmatrix} A_x \\ A_y \\ A_z \end{Bmatrix} = \begin{bmatrix} e_{1x} & e_{2x} & e_{3x} \\ e_{1y} & e_{2y} & e_{3y} \\ e_{1z} & e_{2z} & e_{3z} \end{bmatrix} \begin{Bmatrix} \hat{A}_x \\ \hat{A}_y \\ \hat{A}_z \end{Bmatrix} = [\mu]\{\hat{A}\} = [q]^T \{\hat{A}\} \tag{10.9}$$

As with the Hughes-Liu shell, the next step is to perform the Jaumann rate update:

$$\underline{\sigma}_{ij}^{n+1} = \sigma_{ij}^{n} + \sigma_{ip}^{n} \Delta\omega_{pj} + \sigma_{jp}^{n} \Delta\omega_{pi} \tag{9.22}$$

to account for the material rotation between time steps n and n+1. The Jaumann rate update of the stress tensor is applied in the global configuration before the constitutive evaluation is performed. In the solid shell, as in the Hughes-Liu shell, the stresses and history variables are stored in the global coordinate system. To evaluate the constitutive relation, the stresses and the strain increments are rotated from the global to the lamina coordinate system using the transformation defined previously:

$$\underline{\sigma}_{ij}^{l\,n+1} = q_{ik} \underline{\sigma}_{kn}^{n+1} q_{jn} \tag{9.23}$$

$$\Delta\varepsilon_{ij}^{l\,n+1/2} = q_{ik} \Delta\varepsilon_{kn}^{n+1/2} q_{jn} \tag{9.24}$$

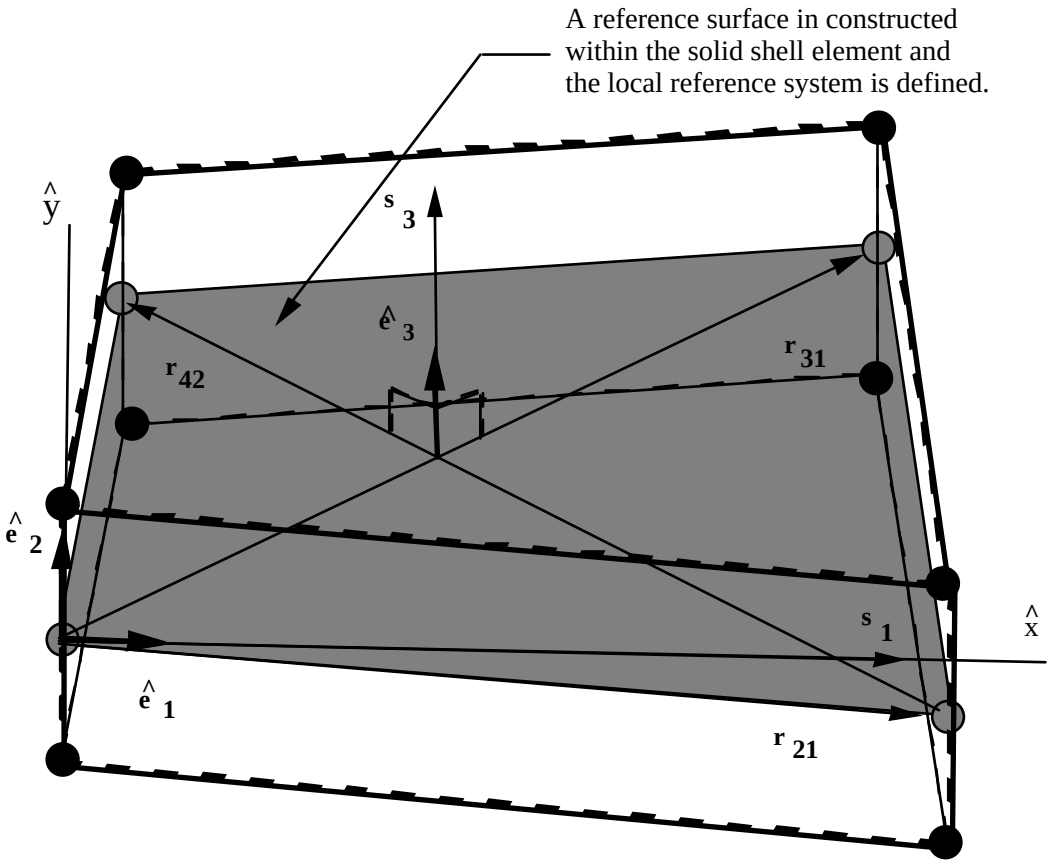


Figure 10.2 Construction of the reference surface in the solid shell element.

where the superscript l indicates components in the lamina (local) coordinate system. The stress is updated incrementally:

$$\sigma_{ij}^{l\,n+1} = \underline{\sigma}_{ij}^{l\,n+1} + \Delta\sigma_{ij}^{l\,n+1/2} \tag{9.25}$$

Independently from the constitutive evaluation

$$\sigma_{33}^{l} = 0 \tag{10.10}$$

which ensures that the plane stress condition is satisfied, we update the normal stress which is used as a penalty to maintain the thickness of the shell:

$$\sigma_{33}^{penalty\,n+1} = \sigma_{33}^{penalty\,n} + E\Delta\varepsilon_{33}^{l} \tag{10.11}$$

where E is the elastic Young's modulus for the material. The stress tensor of Equation (9.25) is rotated back to the global system:

$$\sigma_{ij}^{n+1} = q_{ki}\sigma_{kn}^{l^{n+1}} q_{nj} \tag{9.26}$$

A penalty stress tensor is then formed by transforming the normal penalty stress tensor (a null tensor except for the 33 term) back to the global system:

$$\sigma_{ij}^{penalty^{n+1}} = q_{ki}\sigma_{kn}^{penalty^{l^{n+1}}} q_{nj} \tag{10.12}$$

before computing the internal force vector. The internal force vector can now be computed:

$$f^{\text{int}} = \int B^{t^{n+1}} \left[\sigma^{n+1} + \sigma^{penalty^{n+1}} \right] d\upsilon \tag{10.13}$$

The brick shell exhibits no discernible locking problems with this approach.

The treatment of the hourglass modes is identical to that described for the solid elements in Section 3.

11. TRUSS ELEMENT

One of the most simple elements is the pin-jointed truss element shown in Figure 11.1. This element has three degrees of freedom at each node and carries an axial force. The displacements and velocities measured in the local system are interpolated along the axis according to

$$u = u_1 + \frac{x}{L}(u_2 - u_1) \tag{11.1}$$

$$\dot{u} = \dot{u}_1 + \frac{x}{L}(\dot{u}_2 - \dot{u}_1) \tag{11.2}$$

where at $x = 0$, $u = u_1$ and at $x = L$, $u = u_2$. Incremental strains are found from

$$\Delta\varepsilon = \frac{(\dot{u}_2 - \dot{u}_1)}{L}\Delta t \tag{11.3}$$

and are computed in LS-DYNA3D using

$$\Delta\varepsilon^{n+\frac{1}{2}} = \frac{2\left(\dot{u}_2^{n+\frac{1}{2}} - \dot{u}_1^{n+\frac{1}{2}}\right)}{L^n + L^{n+1}}\Delta t^{n+\frac{1}{2}} \tag{11.4}$$

The normal force N is then incrementally updated using a tangent modulus E^t according to

$$N^{n+1} = N^n + AE^t\,\Delta\varepsilon^{\,n+1/2} \tag{11.5}$$

Two constitutive models are implemented for the truss element: elastic and elastic-plastic with kinematic hardening.

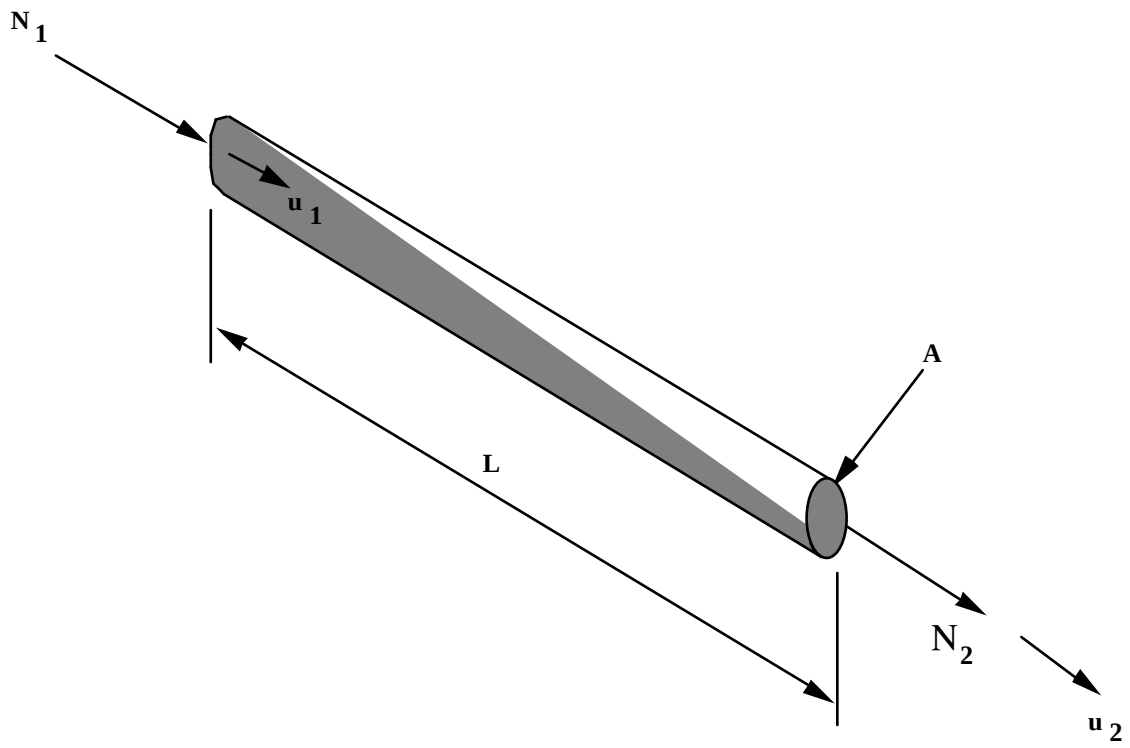


Figure 11.1 Truss element.

12. MEMBRANE ELEMENT

The Belytschko-Lin-Tsay shell element {Belytschko and Tsay [1981], Belytschko et al. [1984a]} is the basis for this very efficient membrane element. In this section we briefly outline the theory employed which, like the shell on which it is based, uses a combined co-rotational and velocity-strain formulation. The efficiency of the element is obtained from the mathematical simplifications that result from these two kinematical assumptions. The co-rotational portion of the formulation avoids the complexities of nonlinear mechanics by embedding a coordinate system in the element. The choice of velocity strain or rate of deformation in the formulation facilitates the constitutive evaluation, since the conjugate stress is the more familiar Cauchy stress.

In membrane elements the rotational degrees of freedom at the nodal points may be constrained, so that only the translational degress of freedom contribute to the straining of the membrane. A triangular membrane element may be obtained by collapsing adjacent nodes of the quadrilateral.

12.1 Co-rotational Coordinates

The midsurface of the quadrilateral membrane element is defined by the location of the element's four corner nodes. An embedded element coordinate system (Figure 6.1) that deforms with the element is defined in terms of these nodal coordinates. The co-rotational coordinate system follows the development in Section 6, Equations (6.1)—(6.3).

12.2 Velocity-Strain Displacement Relations

The co-rotational components of the velocity strain (rate of deformation) are given by:

$$\hat{d}_{ij} = \frac{1}{2}\left(\frac{\partial \hat{\upsilon}_i}{\partial \hat{x}_j} + \frac{\partial \hat{\upsilon}_j}{\partial \hat{x}_i} \right) \tag{6.7}$$

The above velocity-strain relations are evaluated only at the center of the shell. Standard bilinear nodal interpolation is used to define the midsurface velocity, angular velocity, and the element's coordinates (isoparametric representation). These interpolation relations are given by

$$v^m = N_I(\xi,\eta)\, v_I \tag{6.9a}$$

$$x^m = N_I(\xi,\eta)\, x_I \tag{6.9c}$$

where the subscript I is summed over all the element's nodes and the nodal velocities are obtained by differentiating the nodal coordinates with respect to time, i.e. $\upsilon_I = \dot{x}_I$. The bilinear shape functions are defined in Equations (6.10).

The velocity strains at the center of the element, i.e. at $\xi = 0$, and $\eta = 0$, are obtained as in Section 6 giving:

$$\hat{d}_x = B_{1I}\hat{\upsilon}_{xI} \tag{6.11a}$$

$$\hat{d}_y = B_{2I}\hat{\upsilon}_{yI} \tag{6.11b}$$

$$2\hat{d}_{xy} = B_{2I}\hat{v}_{xI} + B_{1I}\hat{v}_{yI} \tag{6.11c}$$

where

$$B_{1I} = \frac{\partial N_I}{\partial \hat{x}} \tag{6.12a}$$

$$B_{2I} = \frac{\partial N_I}{\partial \hat{y}} \tag{6.12b}$$

12.3 Stress Resultants and Nodal Forces

After suitable constitutive evaluations using the above velocity strains, the resulting stresses are multiplied by the thickness of the membrane, h, to obtain local resultant forces. Therefore,

$$\hat{f}^R_{\alpha\beta} = h\hat{\sigma}_{\alpha\beta} \tag{12.1}$$

where the superscript *R* indicates a resultant force and the Greek subscripts emphasize the limited range of the indices for plane stress plasticity.

The above element centered force resultants are related to the local nodal forces by

$$\hat{f}_{xI} = A\left(B_{1I}\,\hat{f}^R_{xx} + B_{2I}\,\hat{f}^R_{xy}\right) \tag{6.14a}$$

$$\hat{f}_{yI} = A\left(B_{2I}\,\hat{f}_{yy}^{R} + B_{1I}\,\hat{f}_{xy}^{R}\right) \tag{6.14b}$$

where A is the area of the element.

The above local nodal forces are then transformed to the global coordinate system using the transformation relations given in Equation (6.5a).

12.4 Membrane Hourglass Control

Hourglass deformations need to be resisted for the membrane element. The hourglass control for this element is discussed in Section 6.4.

13. DISCRETE ELEMENTS AND MASSES

The discrete elements and masses in DYNA3D provide a capability for modeling simple spring-mass systems as well as the response of more complicated mechanisms. Occasionally, the response of complicated mechanisms or materials need to be included in DYNA3D models, e.g. energy absorbers used in passenger vehicle bumpers. These mechanisms are often experimentally characterized in terms of force-displacement curves. DYNA3D provides a selection of discrete elements that can be used individually or in combination to model complex force-displacement relations.

The discrete elements are assumed to be massless. However, to solve the equations of motion at unconstrained discrete element nodes or nodes joining multiple discrete elements, nodal masses must be specified at these nodes. DYNA3D provides a direct method for specifying these nodal masses in the model input.

All of the discrete elements are two-node elements, i.e. three-dimensional springs or trusses. A discrete element may be attached to any of the other DYNA3D continuum, structural, or rigid body element. The force update for the discrete elements may be written as

$$\hat{f}^{\,i+1} = \hat{f}^{\,i} + \Delta\hat{f} \tag{13.1}$$

where the superscript i + 1 indicates the time increment and the superposed caret $(\hat{})$ indicates the force in the local element coordinates, i.e. along the axis of the element. In the default case, i.e., no orientation vector is used, the global components of the discrete element force are obtained by using the element's direction cosines:

$$\begin{Bmatrix} F_x \\ F_y \\ F_z \end{Bmatrix} = \frac{\hat{f}}{l} \begin{Bmatrix} \Delta l_x \\ \Delta l_y \\ \Delta l_z \end{Bmatrix} = \hat{f} \begin{Bmatrix} n_x \\ n_y \\ n_z \end{Bmatrix} = \hat{f}\, \underset{\sim}{n} \tag{13.2}$$

where

$$\Delta \underset{\sim}{l} = \begin{Bmatrix} \Delta l_x \\ \Delta l_y \\ \Delta l_z \end{Bmatrix} = \begin{Bmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{Bmatrix} \tag{13.3}$$

l is the length

$$l = \sqrt{\Delta l_x^2 + \Delta l_y^2 + \Delta l_z^2} \tag{13.4}$$

and (x_i, y_i, z_i) are the global coordinates of the nodes of the spring element. The forces in Equation (13.2) are added to the first node and subtracted from the second node.

For a node tied to ground we use the same approach but for the (x_2,y_2,z_2) coordinates in Equation (13.2) the initial coordinates of node 1, i.e., (x_0,y_0,z_0) are used instead; therefore,

$$\begin{Bmatrix} F_x \\ F_y \\ F_z \end{Bmatrix} = \frac{\hat{f}}{l} \begin{Bmatrix} x_0 - x_1 \\ y_0 - y_1 \\ z_0 - z_1 \end{Bmatrix} = \hat{f} \begin{Bmatrix} n_x \\ n_y \\ n_z \end{Bmatrix} \tag{13.5}$$

The increment in the element force is determined from the user specified force-displacement relation. Currently eight types of force-displacement relations may be specified:

1. linear elastic;
2. linear viscous;
3. nonlinear elastic;
4. nonlinear viscous;
5. elasto-plastic with isotropic hardening;
6. general nonlinear;
7. linear viscoelastic.
8. inelastic tension and compression only.

The force-displacement relations for these models are discussed in the following later.

13.1 Orientation Vectors

An orientation vector,

$$\underset{\sim}{m} = \begin{Bmatrix} m_1 \\ m_2 \\ m_3 \end{Bmatrix} \tag{13.6}$$

can be defined to control the direction the spring acts. We will first consider the portion of the displacement that lies in the direction of the vector. The displacement of the spring is updated based on the change of length given by

$$\Delta I = I - I_0 \tag{13.7}$$

where I_0 is the initial length in the direction of the vector and l is the current length given for a node to node spring by

$$I = m_1 (x_2 - x_1) + m_2 (y_2 - y_1) + m_3 (z_2 - z_1) \tag{13.8}$$

and for a node to ground spring by

$$I = m_1 (x_0 - x_1) + m_2 (y_0 - y_1) + m_3 (z_0 - z_1) \tag{13.9}$$

The latter case is not intutively obvious and can affect the sign of the force in unexpected ways if the user is not familiar with the relevant equations. The nodal forces are then given by

$$\begin{Bmatrix} F_x \\ F_y \\ F_z \end{Bmatrix} = \hat{f} \begin{Bmatrix} m_1 \\ m_2 \\ m_3 \end{Bmatrix} \tag{13.10}$$

The orientation vector can be either permanently fixed in space as defined in the input or acting in a direction determined by two moving nodes which must not be coincident but may be independent of the nodes of the spring. In the latter case we recompute the direction every cycle according to:

$$\begin{Bmatrix} m_1 \\ m_2 \\ m_3 \end{Bmatrix} = \frac{1}{l^n} \begin{Bmatrix} x_2^n - x_1^n \\ y_2^n - y_1^n \\ z_2^n - z_1^n \end{Bmatrix} \tag{13.11}$$

In Equation (13.9) the superscript, n, refers to the orientation nodes.

For the case where we consider motion in the plane perpendicular to the orientation vector we consider only the displacements in the plane, $\Delta \underset{\sim}{l}^p$, given by,

$$\Delta \underset{\sim}{l}^p = \Delta \underset{\sim}{l} - \underset{\sim}{m} \left(\underset{\sim}{m} \cdot \Delta \underset{\sim}{l} \right) \tag{13.12}$$

We update the displacement of the spring based on the change of length in the plane given by

$$\Delta l^p = l^p - l_0^p \tag{13.13}$$

where l_0^p is the initial length in the direction of the vector and l is the current length given for a node to node spring by

$$l^p = m_1^p(x_2 - x_1) + m_2^p(y_2 - y_1) + m_3^p(z_2 - z_1) \tag{13.14}$$

and for a node to ground spring by

$$l^p = m_1^p(x_0 - x_1) + m_2^p(y_0 - y_1) + m_3^p(z_0 - z_1) \tag{13.15}$$

where

$$\begin{Bmatrix} m_1^p \\ m_2^p \\ m_3^p \end{Bmatrix} = \frac{1}{\sqrt{\Delta l_x^{p\,2} + \Delta l_y^{p\,2} + \Delta l_z^{p\,2}}} \begin{Bmatrix} \Delta l_x^p \\ \Delta l_y^p \\ \Delta l_z^p \end{Bmatrix} \tag{13.16}$$

After computing the displacements the nodal forces are then given by

$$\begin{Bmatrix} F_x \\ F_y \\ F_z \end{Bmatrix} = \hat{f} \begin{Bmatrix} m_1^p \\ m_2^p \\ m_3^p \end{Bmatrix} \tag{13.17}$$

13.2 Dynamic Magnification "Strain Rate" Effects

To account for "strain rate" effects, we have a simple method of scaling the forces based to the relative velocities that applies to all springs. The forces computed from the spring elements are assumed to be the static values and are scaled by an amplification factor to obtain the dynamic value:

$$F_{dynamic} = \left(1. + k_d \frac{V}{V_0}\right) F_{static} \tag{13.18}$$

where

k_d = is a user defined input value
V = absolute relative velocity
V_0 = dynamic test velocity

For example, if it is known that a component shows a dynamic crush force at 15m/s equal to 2.5 times the static crush force, use k_d=1.5 and V_0=15.

13.3 Deflection Limits in Tension and Compression

The deflection limit in compression and tension is restricted in its application to no more than one spring per node subject to this limit, and to deformable bodies only. For example in the former case, if three spring are in series either the center spring or the two end springs may be subject to a limit but not all three. When the limiting deflection is reached momentum conservation calculations are performed and a common acceleration is computed:

$$\hat{a}_{common} = \frac{\hat{f}_1 + \hat{f}_2}{m_1 + m_2} \tag{13.19}$$

An error termination will occur if a rigid body node is used in a spring definition where compression is limited.

13.4 Linear Elastic or Linear Viscous

These discrete elements have the simplest force-displacement relations. The linear elastic discrete element has a force-displacement relation of the form

$$\hat{f} = K\Delta l \tag{13.20}$$

where K is the element's stiffness and Δl is the change in length of the element. The linear viscous element has a similar force-velocity (rate of displacement) relation:

$$\hat{f} = C\frac{\Delta l}{\Delta t} \tag{13.21}$$

where C is a viscous damping parameter and Δt is the time step increment.

13.5 Nonlinear Elastic or Nonlinear Viscous

These discrete elements use piecewise force-displacement or force-relative velocity relations. The nonlinear elastic discrete element has a tabulated force-displacement relation of the form

$$\hat{f} = K\Delta l \tag{13.22}$$

where K(Δl) is the tabulated force that depends on the total change in the length of the element (Figure 13.1) The nonlinear viscous element has a similar tabulated force-relative velocity relation:

$$\hat{f} = C\frac{\Delta l}{\Delta t} \tag{13.23}$$

where $C\left(\frac{\Delta l}{\Delta t}\right)$ is the viscous damping force that depends on the rate of change of the element's length. Nonlinear discrete elements unload along the loading paths.

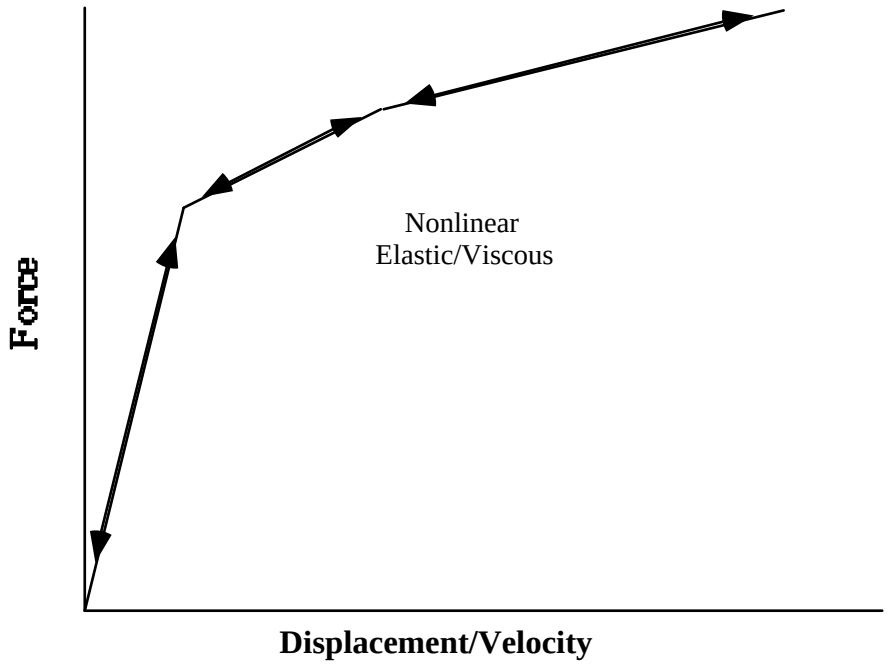


Figure 13.1 Piecewise linear force-displacement curve for nonlinear elastic discrete element.

If the spring element is initially of zero length and if no orientation vectors are used then only the tensile part of the stress strain curve needs to be defined. However, if the spring element is initially of finite length then the curve must be defined in both the positive and negative quadrants.

13.6 Elasto-Plastic with Isotropic Hardening

The elasto-plastic discrete element has a bilinear force-displacement relationship that is specified by the elastic stiffness, a tangent stiffness and a yield force (Figure 13.2). This discrete element uses the elastic stiffness model for unloading until the yield force is exceeded during unloading. The yield force is updated to track its maximum value which is equivalent to an isotropic hardening model. The force-displacement relation during loading may be written as

$$f = F_y\left(1-\frac{K_t}{K}\right)+K_t\Delta l \tag{13.24}$$

where F_y is the yield force and K_t is the tangent stiffness.

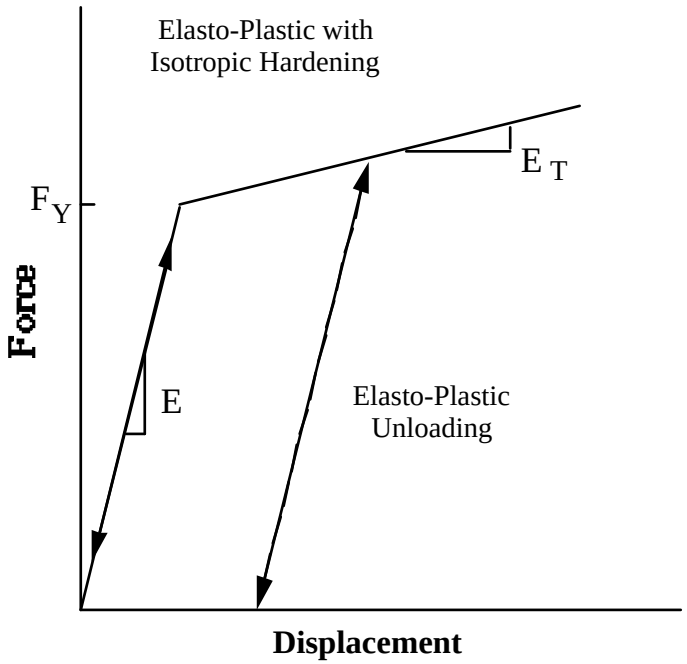


Figure 13.2 Loading and unloading force-displacement curves for elasto-plastic discrete element.

13.7 General Nonlinear

The general nonlinear discrete element allows the user to specify independent and nonsymmetrical piecewise linear loading and unloading paths (Figure 13.3(a)).

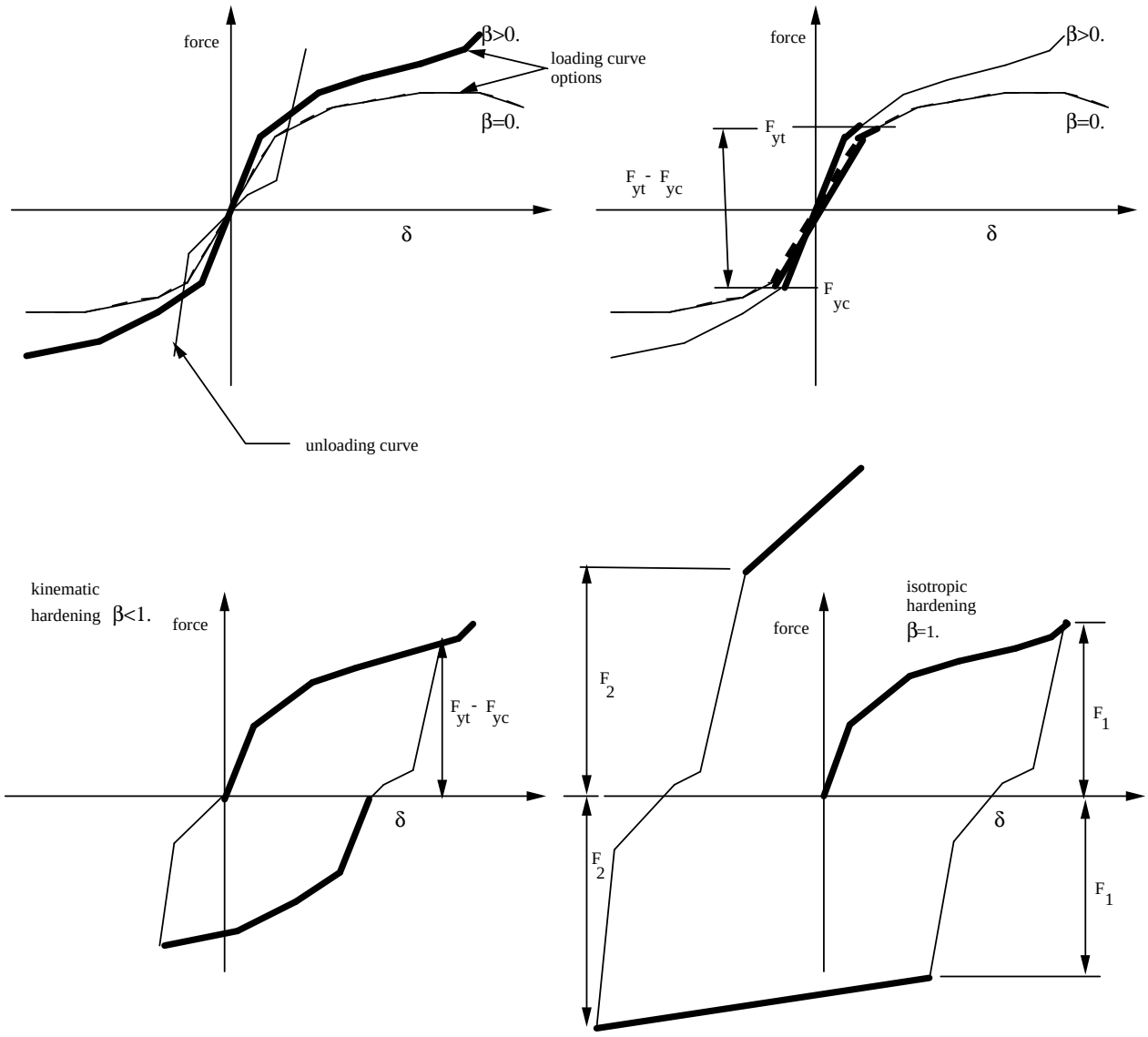


Figure 13.3 Loading and unloading force displacement curves for general nonlinear discrete element.

This element combines the features of the above-described nonlinear elastic and elasto-plastic discrete elements by allowing the user to specify independent initial yield forces in tension (F_{YT}) and in compression (F_{YC}). If the discrete element force remains between these initial yield values, the element unloads along the loading path (Figure 13.3(b)). This corresponds to the nonlinear elastic discrete element.

However, if the discrete element force exceeds either of these initial yield values, the specified unloading curve is used for subsequent unloading. Additionally, the initial loading and unloading curves are allowed to move in the force-displacement space by specifying a mixed hardening parameter β, where $\beta = 0$ corresponds to kinematic hardening (Figure 13.3(c)) and $\beta = 1$ corresponds to isotropic hardening (Figure 13.3(d)).

13.8 Linear Visco-Elastic

The linear viscoelastic discrete element [Schwer, Cheva, and Hallquist 1991] allows the user to model discrete components that are characterized by force relaxation or displacement creep behavior. The element's variable stiffness is defined by three parameters and has the form

$$K(t) = K_\infty + (K_0 - K_\infty)e^{-\beta t} \tag{13.25}$$

where K_∞ is the long duration stiffness, K_0 is the short time stiffness, and β is a decay parameter that controls the rate at which the stiffness transitions between the short and long duration stiffness (Figure 13.4).

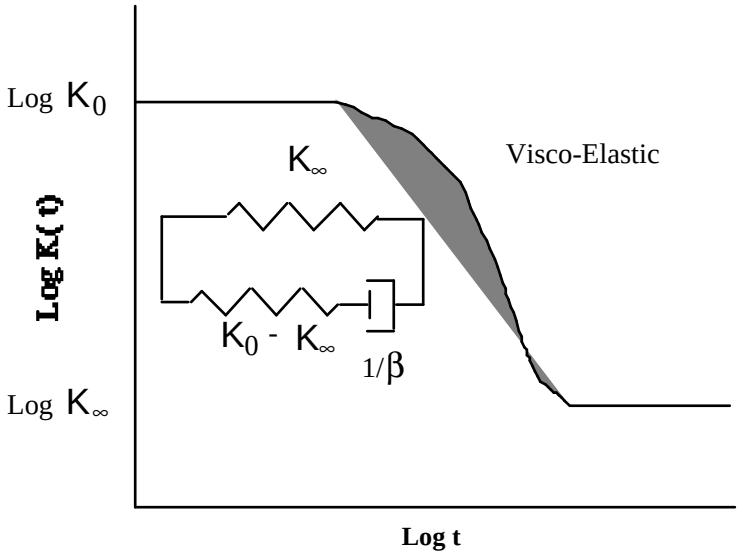


Figure 13.4. Typical stiffness relaxation curve used for the viscoelastic discrete element.

This model corresponds to a three-parameter Maxwell model (see insert in Figure 13.4) which consists of a spring and damper in series connected to another spring in parallel. Although this discrete element behavior could be built up using the above-

described linear elastic and linear viscous discrete elements, such a model would also require the user to specify the nodal mass at the connection of the series spring and damper. This mass introduces a fourth parameter which would further complicate fitting the model to experimental data.

13.9 Seat Belt Material

The seat belt capability reported here was developed by Walker and co-workers [Walker and Dallard 1991, Strut, Walker, et.al, 1991] and this section exerpted from their documentation. Each belt material defines stretch characteristics and mass properties for a set of belt elements. The user enters a load curve for loading, the points of which are *(Strain, Force)*. Strain is defined as engineering strain, i.e.

$$Strain = \frac{current\ \ length}{initial\ length} - 1.$$

Another similar curve is entered to describe the unloading behavior. Both loadcurves should start at the origin (0,0) and contain positive force and strain values only. The belt material is tension only with zero forces being generated whenever the strain becomes negative. The first non-zero point on the loading curve defines the initial yield point of the material. On unloading, the unloading curve is shifted along the strain axis until it crosses the loading curve at the 'yield' point from which unloading commences. If the initial yield has not yet been exceeded or if the origin of the (shifted) unloading curve is at negative strain, the original loading curves will be used for both loading and unloading. If the strain is less than the strain at the origin of the unloading curve, the belt is slack and no force is generated. Otherwise, forces will then be determined by the unloading curve for unloading and reloading until the strain again exceeds yield after which the loading curves will again be used.

A small amount of damping is automatically included. This reduces high frequency oscillation, but, with realistic force-strain input characteristics and loading rates, does not significantly alter the overall forces-strain performance. The damping forced opposes the relative motion of the nodes and is limited by stability:

$$D = \frac{.1 \times mass \times relative\ velocity}{timestep\ size}$$

In addition, the magnitude of the damping forces is limited to one tenth of the force calculated from the forces-strain relationship and is zero when the belt is slack. Damping forces are not applied to elements attached to sliprings and retractors.

The user inputs a mass per unit length that is used to calculate nodal masses on initialization.

A 'minimum length' is also input. This controls the shortest length allowed in any element and determines when an element passes through sliprings or are absorbed into the retractors. One tenth of a typical initial element length is usually a good choice.

13.10 Seat Belt Elements

Belt elements are single degree of freedom elements connecting two nodes and are treated in a manner similar to the spring elements. When the strain in an element is positive (i.e. the current length is greater then the unstretched length), a tension force is calculated from the material characteristics and is applied along the current axis of the element to oppose further stretching. The unstretched length of the belt is taken as the initial distance between the two nodes defining the position of the element plus the initial slack length. At the beginning of the calculation the seatbelt elements can be obtained within a retractor.

13.11 Sliprings

Sliprings are defined in the LS-DYNA3D input by giving a slipring ID and element ID's for two elements who share a node which is coincident with the slipring node. The slipring node may not be attached to any belt elements.

Sliprings allow continuous sliding of a belt through a sharp change of angle. Two elements (1 and 2 in Figure 13.5) meet at the slipring. Node B in the belt material remains attached to the slipring node, but belt material (in the form of unstretched length) is passed from element 1 to element 2 to achieve slip. The amount of slip at each timestep is calculated from the ratio of forces in elements 1 and 2. The ratio of forces is determined by the relative angle between elements 1 and 2 and the coefficient of friction, μ. The tension in the belts are taken as T_1 and T_2, where T_2 is on the high tension side and T_1 is the force on the low tension side. Thus if T_2 is sufficiently close to T_1 no slip occurs; otherwise, slip is just sufficient to reduce the ratio T_2/T_1 to $e^{\mu\Theta}$. No slip occurs if both elements are slack. The out-of-balance force at node B is reacted on the slipring node; the motion of node B follows that of slipring node.

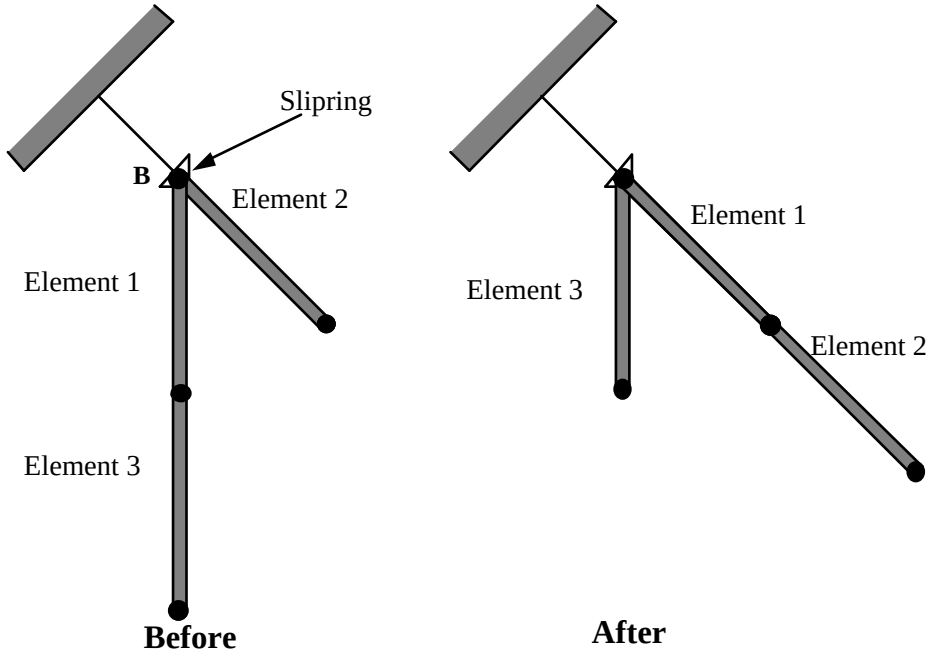


Figure 13.5. Elements passing through slipring.

If, due to slip through the slipring, the unstretched length of an element becomes less than the minimum length (as entered on the belt material card), the belt is remeshed locally: the short element passes through the slipring and reappears on the other side (see Figure 13.5). The new unstretched length of e1 is $1.1 \times$ minimum length. Force and strain in e2 and e3 are unchanged; force and strain in e1 are now equal to those in e2. Subsequent slip will pass material from e3 to e1. This process can continue with several elements passing in turn through the slipring.

To define a slipring, the user identifies the two belt elements which meet at the slipring, the friction coefficient, and the slipring node. The two elements must have a common node coincident with the slipring node. No attempt should be made to restrain or constrain the common node for its motion will automatically be constrained to follow the slipring node. Typically, the slipring node is part of the vehicle body structure and, therefore, belt elements should not be connected to this node directly, but any other feature can be attached, including rigid bodies.

13.12 Retractors

Retractors are defined by giving a node, the “retractor node” and an element ID of an element outside the retractor but with one node that is coincident with the retractor

node. Also sensor ID's must be defined for up to four sensors which can activate the seatbelt.

Retractors allow belt material to be payed out into a belt element, and they operate in one of two regimes: unlocked when the belt material is payed out or reeled in under constant tension and locked when a user defined force-pullout relationship applies.

The retractor is initially unlocked, and the following sequence of events must occur for it to become locked:

1. Any one of up to four sensors must be triggered. (The sensors are described below).
2. Then a user-defined time delay occurs.
3. Then a user-defined length of belt must be payed out (optional).
4. Then the retractor locks.

and once locked, it remains locked.

In the unlocked regime, the retractor attempts to apply a constant tension to the belt. This feature allows an initial tightening of the belt, and takes up any slack whenever it occurs. The tension value is taken from the first point on the force-pullout load curve. The maximum rate of pull out or pull in is given by $0.01 \times$ fed length per time step. Because of this, the constant tension value is not always be achieved.

In the locked regime, a user-defined curve describes the relationship between the force in the attached element and the amount of belt material payed out. If the tension in the belt subsequently relaxes, a different user-defined curve applies for unloading. The unloading curve is followed until the minimum tension is reached.

The curves are defined in terms of initial length of belt. For example, if a belt is marked at 10mm intervals and then wound onto a retractor, and the force required to make each mark emerge from the (locked) retractor is recorded, the curves used for input would be as follows:

0	Minimum tension (should be > zero)
10mm	Force to emergence of first mark
20mm	Force to emergence of second mark
.	.
.	.
.	.

Pyrotechnic pretensions may be defined which cause the retractor to pull in the belt at a predetermined rate. This overrides the retractor force-pullout relationship from the moment when the pretensioner activates.

If desired, belt elements may be defined which are initially inside the retractor. These will emerge as belt material is payed out, and may return into the retractor if sufficient material is reeled in during unloading.

Elements e2, e3 and e4 are initially inside the retractor, which is paying out material into element e1. When the retractor has fed L_{crit} into e1, where

L_{crit} = fed length - 1.1 × minimum length
(minimum length defined on belt material input)
(fed length defined on retractor input)

element e2 emerges with an unstretched length of 1.1 × minimum length; the unstretched length of element e1 is reduced by the same amount. The force and strain in e1 are unchanged; in e2, they are set equal to those in e1. The retractor now pays out material into e2.

If no elements are inside the retractor, e2 can continue to extend as more material is fed into it.

As the retractor pulls in the belt (for example, during initial tightening), if the unstretched length of the mouth element becomes less than the minimum length, the element is taken into the retractor.

To define a retractor, the user enters the retractor node, the ‘mouth’ element (into which belt material will be fed, e1 in Figure 13.6, up to 4 sensors which can trigger unlocking, a time delay, a payout delay (optional), load and unload curve numbers, and the fed length. The retractor node is typically part of the vehicle stricture; belt elements should not be connected to this node directly, but any other feature can be attached including rigid bodies. The mouth element should have a node coincident with the retractor but should not be inside the retractor. The fed length would typically be set either to a typical element initial length, for the distance between painted marks on a real belt for comparisons with high speed film. The fed length should be at least three times the minimum length.

If there are elements initially inside the retractor (e2, e3 and e4 in the Figure) they should not be referred to on the retractor input, but the retractor should be identified on the element input for these elements. Their nodes should all be coincident with the retractor node and should not be restrained or constrained. Initial slack will automatically be set to 1.1 × minimum length for these elements; this overrides any user-defined value.

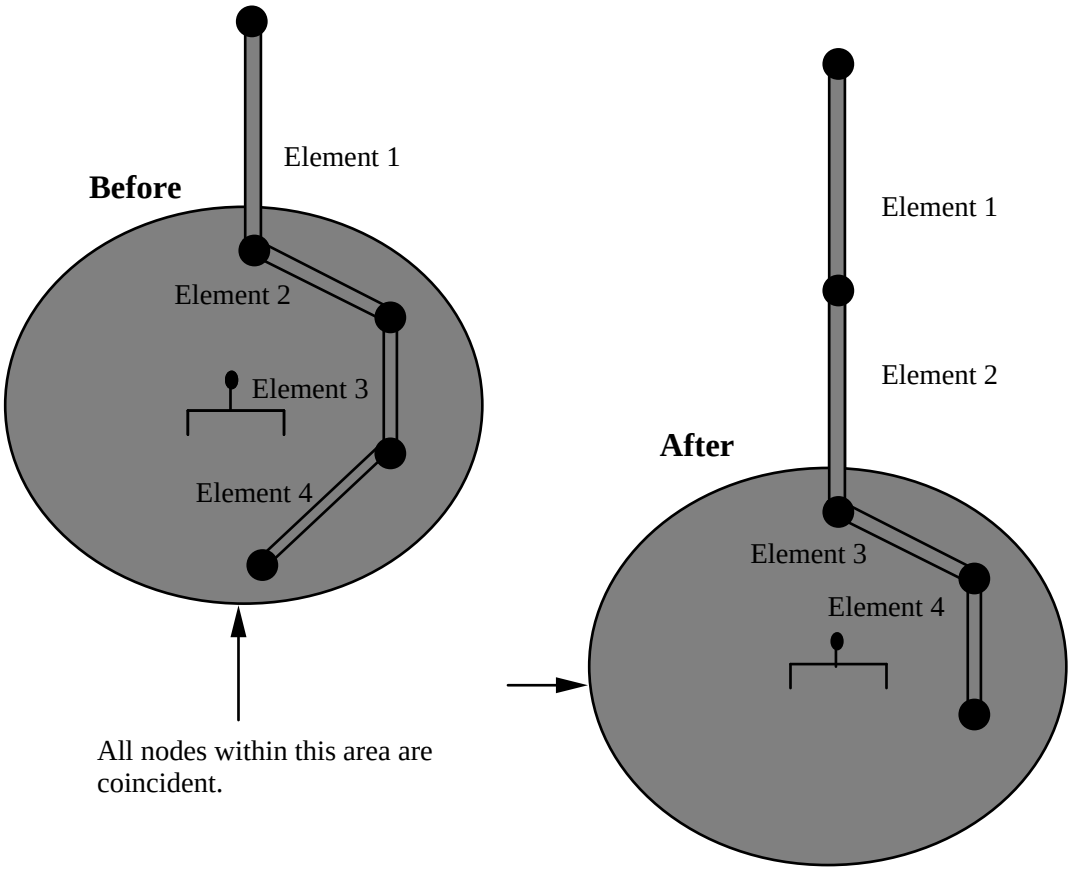


Figure 13.6. Elements in a retractor.

Weblockers can be included within the retractor representation simply by entering a ‘locking up’ characteristic in the force pullout curve, see Figure 13.7. The final section can be very steep (but must have a finite slope).

13.13 Sensors

Sensors are used to trigger locking of retractors and activate pretensioners. Four types of sensor are available which trigger according to the following criteria:

Type 1 – When the magnitude of x-, y-, or z- acceleration of a given node has remained above a given level continuously for a given time, the sensor triggers. This does not work with nodes on rigid bodies.

Type 2 – When the rate of belt payout from a given retractor has remained above a given level continuously for a given time, the sensor triggers.

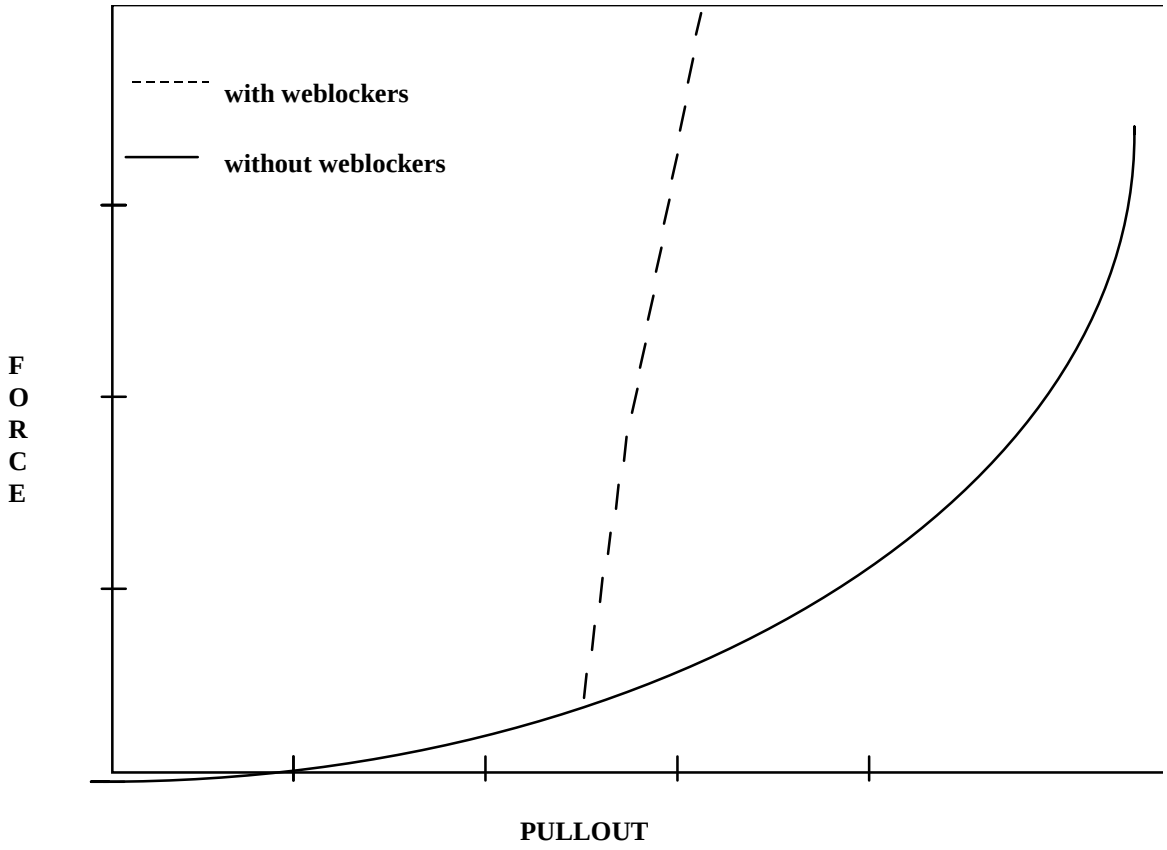


Figure 13.7 Retractor force pull characteristics.

Type 3 – The sensor triggers at a given time.

Type 4 – The sensor triggers when the distance between two nodes exceeds a given maximum or becomes less than a given minimum. This type of sensor is intended for use with an explicit mas/spring representation of the sensor mechanism.

By default, the sensors are inactive during dynamic relaxation. This allows initial tightening of the belt and positioning of the occupant on the seat without locking the retractor or firing any pretensioners. However, a flag can be set in the sensor input to make the sensors active during the dynamic relaxation phase.

13.14 Pretensioners

Pretensioners allow modelling of three types of active devices which tighten the belt during the initial stages of a crash. The first type represents a pyrotechnic device which spins the spool of a retractor, causing the belt to be reeled in. The user defines a

pull-in versus time curve which applies once the pretensioner activates. The remaining types represents preloaded springs or torsion bars which move the buckle when released. The pretensioner is associated with any type of spring element including rotational. Note that the preloaded spring, locking spring and any restraints on the motion of the associated nodes are defined in the normal way; the action of the pretensioner is merely to cancel the force in one spring until (or after) it fires. With the second type, the force in the spring element is cancelled out until the pretensioner is activated. In this case the spring in question is normally a stiff, linear spring which acts as a locking mechanism, preventing motion of the seat belt buckle relative to the vehicle. A preloaded spring is defined in parallel with the locking spring. This type avoids the problem of the buckle being free to 'drift' before the pretensioner is activated.

To activate the pretensioner the following sequence of events must occur:

1. Any one of up to four sensors must be triggered.
2. Then a user-defined time delay occurs.
3. Then the pretensioner acts.

13.15 Accelerometers

The accelerometer is defined by three nodes in a rigid body which defines a triad to measure the accelerations in a local system. The presence of the accelerometer means that the accelerations and velocities of node 1 will be output to **all** output files in local instead of global coordinates.

The local coordinate system is defined by the three nodes as follows:

- local **x** from node 1 to node 2
- local **z** perpendicular to the plane containing nodes, 1, 2, and 3 ($z = \mathbf{x} \times \mathbf{a}$), where **a** is from node 1 to node 3).
- local $\mathbf{y} = \mathbf{x} \times \mathbf{z}$

The three nodes should all be part of the same rigid body. The local axis then rotates with the body.

14. SIMPLIFIED ARBITRARY LAGRANGIAN-EULERIAN

Arbitrary Lagrangian-Eulerian (ALE) formulations may be thought of as algorithms that perform automatic rezoning. Users perform manual rezoning by

1. Stopping the calculation when the mesh is distorted,
2. Smoothing the mesh,
3. Remapping the solution from the distorted mesh to the smooth mesh.

An ALE formulation consists of a Lagrangian time step followed by a "remap" or "advection" step. The advection step performs an incremental rezone, where "incremental" refers to the fact that the positions of the nodes are moved only a small fraction of the characteristic lengths of the surrounding elements. Unlike a manual rezone, the topology of the mesh is fixed in an ALE calculation. An ALE calculation can be interrupted like an ordinary Lagrangian calculation and a manual rezone can be performed if an entirely new mesh is necessary to continue the calculation.

The accuracy of an ALE calculation is often superior to the accuracy of a manually rezoned calculation because the algorithm used to remap the solution from the distorted to the undistorted mesh is second order accurate for the ALE formulation while the algorithm for the manual rezone is only first order accurate.

In theory, an ALE formulation contains the Eulerian formulation as a subset. Eulerian codes can have more than one material in each element, but most ALE implementations are simplified ALE formulations which permit only a single material in each element. The primary advantage of a simplified formulation is its reduced cost per time step. When elements with more than one material are permitted, the number and types of materials present in an element can change dynamically. Additional data is necessary to specify the materials in each element and the data must be updated by the remap algorithms.

The range of problems that can be solved with an ALE formulation is a direct function of the sophistication of the algorithms for smoothing the mesh. Early ALE codes were not very successful largely because of their primitive algorithms for smoothing the mesh. In simplified ALE formulations, most of the difficulties with the mesh are associated with the nodes on the material boundaries. If the material boundaries are purely Lagrangian, i.e., the boundary nodes move with the material at all times, no smooth mesh maybe possible and the calculation will terminate. The algorithms for maintaining a smooth boundary mesh are therefore as important to the robustness of the calculations as the algorithms for the mesh interior.

The cost of the advection step per element is usually much larger than the cost of the Lagrangian step. Most of the time in the advection step is spent in calculating the material transported between the adjacent elements, and only a small part of it is spent on calculating how and where the mesh should be adjusted. Second order accurate monotonic advection algorithms are used in LS-DYNA3D despite their high cost per element because their superior coarse mesh accuracy which allows the calculation to be performed with far fewer elements than would be possible with a cheaper first order accurate algorithm.

The second order transport accuracy is important since errors in the transport calculations generally smooth out the solution and reduce the peak values in the history variables. Monotonic advection algorithms are constructed to prevent the transport calculations from creating new minimum or maximum values for the solution variables. They were first developed for the solution of the Navier Stokes equations to eliminate the spurious oscillations that appeared around the shock fronts. Although monotonic algorithms are more diffusive than algorithms that are not monotonic, they must be used for stability in general purpose codes. Many constitutive models have history variables that have limited ranges, and if their values are allowed to fall outside of their allowable ranges, the constitutive models are undefined. Examples include explosive models, which require the burn fraction to be between zero and one, and many elastoplasticity models, such as those with power law hardening, which require a non-negative plastic strain.

The overall flow of an ALE time step is:

1. Perform a Lagrangian time step.
2. Perform an advection step.
    a. Decide which nodes to move.
    b. Move the boundary nodes.
    c. Move the interior nodes.
    d. Calculate the transport of the element-centered variables.
    e. Calculate the momentum transport and update the velocity.

Each element solution variable must be transported. The total number of solution variables, including the velocity, is at least six and depends on the material models. For elements that are modeled with an equation of state, only the density, the internal energy, and the shock viscosity are transported. When the elements have strength, the six components of the stress tensor and the plastic strain must also be advected, for a total of ten

solution variables. Kinematic hardening, if it is used, introduces another five solution variables, for a total of fifteen.

The nodal velocities add an extra three solution variables that must be transported, and they must be advected separately from the other solution variables because they are centered at the nodes and not in the elements. In addition, the momentum must be conserved, and it is a product of the node-centered velocity and the element-centered density. This imposes a constraint on how the momentum transport is performed that is unique to the velocity field. A detailed consideration of the difficulties associated with the transport of momentum is deferred until later.

Perhaps the simplest strategy for minimizing the cost of the ALE calculations is to performs them only every few time steps. The cost of an advection step is typically two to five times the cost of the Lagrangian time step. By performing the advection step only every ten steps, the cost of an ALE calculation can often be reduced by a factor of three without adversely affecting the time step size. In general, it is not worthwhile to advect an element unless at least twenty percent of its volume will be transported because the gain in the time step size will not offset the cost of the advection calculations.

14.1 Mesh Smoothing Algorithms

The algorithms for moving the mesh relative to the material control the range of the problems that can be solved by an ALE formulation. The equipotential method which is used in LS-DYNA3D was developed by Winslow [1990] and is also used in the DYNA2D ALE code [Winslow 1963]. It, and its extensions, have proven to be very successful in a wide variety of problems. The following is extracted from reports prepared by Alan Winslow for LSTC.

14.1.1 Equipotential Smoothing of Interior Nodes

"Equipotential" zoning [Winslow, 1963] is a method of making a structured mesh for finite difference or finite element calculations by using the solutions of Laplace equations (later extended to Poisson equations) as the mesh lines. The same method can be used to smooth selected points in an unstructured three-dimensional mesh provided that it is at least locally structured. This chapter presents a derivation of the three-dimensional equipotential zoning equations, taken from the references, and gives their finite difference equivalents in a form ready to be used for smoothing interior points. We begin by reviewing the well-known two-dimensional zoning equations, and then discuss their extension to three dimensions.

In two dimensions we define curvilinear coordinates ξ,η which satisfy Laplace's equation:

$$\nabla^2\xi = 0 \tag{14.1.1a}$$

$$\nabla^2\eta = 0 \tag{14.1.1b}$$

We solve Equations (14.1.1) for the coordinates $x(\xi,\eta)$ and $y(\xi,\eta)$ of the mesh lines: that is, we invert them so that the geometric coordinates x,y become the dependent variables and the curvilinear coordinates ξ,η the independent variables. By the usual methods of changing variables we obtain

$$\alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} = 0 \tag{14.1.2a}$$

$$\alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} = 0 \tag{14.1.2b}$$

where

$$\alpha \equiv x_\eta^2 + y_\eta^2 , \quad \beta \equiv x_\xi x_\eta + y_\xi y_\eta , \quad \gamma \equiv x_\xi^2 + y_\xi^2 \tag{14.1.3}$$

Equations (14.1.2) can be written in vector form:

$$\alpha \vec{r}_{\xi\xi} - 2\beta \vec{r}_{\xi\eta} + \gamma \vec{r}_{\eta\eta} = 0 \tag{14.1.4}$$

where

$$\vec{r} \equiv x\hat{i} + y\hat{j} \ .$$

We difference Equations (14.1.4) and solve them numerically by an iterative method, since they are nonlinear. In (ξ,η) space we use a mesh whose curvilinear coordinates are straight lines which take on integer values corresponding to the usual numbering in a two-dimensional mesh. The numerical solution then gives us the location of the "equipotential" mesh lines.

In three dimensions x,y,z we add a third curvilinear coordinate ζ and a third Laplace equation

$$\nabla^2\zeta = 0 \tag{14.1.1c}$$

Inversion of the system of three equations (14.1.1) by change of variable is rather complicated. It is easier, as well as more illuminating, to use the methods of tensor analysis pioneered by Warsi [1982] Let the curvilinear coordinates be represented by ξ^i

(i=1,2,3). For a scalar function A(x,y,z), Warsi shows that the transformation of its Laplacian from rectangular Cartesian to curvilinear coordinates is given by

$$\nabla^2 A = \sum_{i,j=1}^{3} g^{ij} A_{\xi^i \xi^j} + \sum_{k=1}^{3} \left(\nabla^2 \xi^k \right) A_{\xi^h} \tag{14.1.5}$$

where a variable subscript indicates differentiation with respect to that variable. Since the curvilinear coordinates are each assumed to satisfy Laplace's equation, the second summation in Equation (14.1.5) vanishes and we have

$$\nabla^2 A = \sum_{i,j=1}^{3} g^{ij} A_{\xi^i \xi^j} \ . \tag{14.1.6}$$

If now we let A = x, y, and z successively, the left-hand side of (14.1.6) vanishes in each case and we get three equations which we can write in vector form

$$\sum_{i,j=1}^{3} g^{ij} \vec{r}_{\xi^i \xi^j} = 0 \ . \tag{14.1.7}$$

Equation (14.1.7) is the three-dimensional generalization of Equations (14.1.4), and it only remains to determine the components of the contravariant metric tensor g^{ij} in three dimensions. These are defined to be

$$g^{ij} \equiv \vec{a}^{\,i} \cdot \vec{a}^{\,j} \tag{14.1.8}$$

where the contravariant base vectors of the transformation from (x,y,z) to (ξ^1, ξ^2, ξ^3) are given by

$$\vec{a}^{\,i} \equiv \nabla \xi^i = \frac{\vec{a}_j \times \vec{a}_k}{\sqrt{g}} \tag{14.1.9}$$

(i,j,k cyclic). Here the covariant base vectors, the coordinate derivatives, are given by

$$\vec{a}_i \equiv \vec{r}_{\xi^i} \tag{14.1.10}$$

where

$$\vec{r} \equiv x\hat{i} + y\hat{j} + z\hat{k} \ .$$

Also,

$$g \equiv \det g_{ij} = \left[\vec{a}_1 \cdot \left(\vec{a}_2 \times \vec{a}_3\right)\right]^2 = J^2 \tag{14.1.11a}$$

where g_{ij} is the covariant metric tensor given by

$$g_{ij} \equiv \vec{a}_i \cdot \vec{a}_j \tag{14.1.11b}$$

and J is the Jacobian of the transformation.

Substituting (14.1.9) into (14.1.8), and using the vector identity

$$\left(\vec{a} \times \vec{b}\right) \cdot \left(\vec{c} \times \vec{d}\right) \equiv \left(\vec{a} \cdot \vec{c}\right) \cdot \left(\vec{b} \cdot \vec{d}\right) - \left(\vec{a} \cdot \vec{d}\right)\left(\vec{b} \cdot \vec{c}\right) \tag{14.1.12}$$

we get

$$gg^{ii} = \vec{a}_j^2 \vec{a}_k^2 - \left(\vec{a}_j \cdot \vec{a}_k\right)^2 = g_{jj} g_{kk} - \left(g_{jk}\right)^2 \tag{14.1.13a}$$

$$gg^{ij} = \left(\vec{a}_i \cdot \vec{a}_k\right)\left(\vec{a}_j \cdot \vec{a}_k\right) - \left(\vec{a}_i \cdot \vec{a}_j\right)\vec{a}_k^2 = g_{ik} g_{jk} - g_{ij} g_{kk} \ . \tag{14.1.13b}$$

Before substituting (14.1.10) into (14.1.13a, b), we return to our original notation:

$$\xi \equiv \xi^1 \ , \quad \eta \equiv \xi^2 \ , \quad \zeta \equiv \xi^3 \tag{14.1.14}$$

Then, using (14.1.10), we get

$$gg^{11} = \vec{r}_\eta^2 \vec{r}_\zeta^2 - \left(\vec{r}_\eta \cdot \vec{r}_\zeta\right)^2 \tag{14.1.15a}$$

$$gg^{22} = \vec{r}_\zeta^2 \vec{r}_\xi^2 - \left(\vec{r}_\zeta \cdot \vec{r}_\xi\right)^2 \tag{14.1.15b}$$

$$gg^{33} = \vec{r}_\xi^2 \vec{r}_\eta^2 - \left(\vec{r}_\xi \cdot \vec{r}_\eta\right)^2 \tag{14.1.15c}$$

for the three diagonal components, and

$$gg^{12} = \left(\vec{r}_\xi \cdot \vec{r}_\zeta\right)\left(\vec{r}_\eta \cdot \vec{r}_\zeta\right) - \left(\vec{r}_\xi \cdot \vec{r}_\eta\right)\vec{r}_\zeta^2 \tag{14.1.16a}$$

$$gg^{31} = \left(\vec{r}_\zeta \cdot \vec{r}_\eta\right)\left(\vec{r}_\xi \cdot \vec{r}_\eta\right) - \left(\vec{r}_\zeta \cdot \vec{r}_\xi\right)\vec{r}_\eta^2 \tag{14.1.16b}$$

$$gg^{31} = \left(\vec{r}_\zeta \cdot \vec{r}_\eta\right)\left(\vec{r}_\xi \cdot \vec{r}_\eta\right) - \left(\vec{r}_\zeta \cdot \vec{r}_\xi\right)\vec{r}_\eta^{\,2} \qquad (14.1.16c)$$

for the three off-diagonal components of this symmetric tensor.

When we express Equations (14.1.15) in terms of the Cartesian coordinates, some cancellation takes place and we can write them in the form

$$gg^{11} = (x_\eta y_\zeta - x_\zeta y_\eta)^2 + (x_\eta z_\zeta - x_\zeta z_\eta)^2 + (y_\eta z_\zeta - y_\zeta z_\eta)^2 \qquad (14.1.17a)$$

$$gg^{22} = (x_\zeta y_\xi - x_\xi y_\zeta)^2 + (x_\zeta z_\xi - x_\xi z_\zeta)^2 + (y_\zeta z_\xi - y_\xi z_\zeta)^2 \qquad (14.1.17b)$$

$$gg^{33} = (x_\xi y_\eta - x_\eta y_\xi)^2 + (x_\xi z_\eta - x_\eta z_\xi)^2 + (y_\xi z_\eta - y_\eta z_\xi)^2 \qquad (14.1.17c)$$

guaranteeing positivity as required by Equations (14.1.8). Writing out Equations (14.1.16) we get

$$\begin{aligned} gg^{12} = {} & (x_\xi x_\zeta + y_\xi y_\zeta + z_\xi z_\zeta)(x_\eta x_\zeta + y_\eta y_\zeta + z_\eta z_\zeta) \\ & - (x_\xi x_\eta + y_\xi y_\eta + z_\xi z_\eta)\left(x_\zeta^2 + y_\zeta^2 + z_\zeta^2\right) \end{aligned} \qquad (14.1.18a)$$

$$\begin{aligned} gg^{23} = {} & (x_\eta x_\xi + y_\eta y_\xi + z_\eta z_\xi)(x_\zeta x_\xi + y_\zeta y_\xi + z_\zeta z_\xi) \\ & - (x_\eta x_\zeta + y_\eta y_\zeta + z_\eta z_\zeta)\left(x_\xi^2 + y_\xi^2 + z_\xi^2\right) \end{aligned} \qquad (14.1.18b)$$

$$\begin{aligned} gg^{31} = {} & (x_\zeta x_\eta + y_\zeta y_\eta + z_\zeta z_\eta)(x_\xi x_\eta + y_\xi y_\eta + z_\xi z_\eta) \\ & - (x_\zeta x_\xi + y_\zeta y_\xi + z_\zeta z_\xi)\left(x_\eta^2 + y_\eta^2 + z_\eta^2\right). \end{aligned} \qquad (14.1.18c)$$

Hence we finally write Equations (14.1.7) in the form

$$g\left(g^{11}\vec{r}_{\xi\xi} + g^{22}\vec{r}_{\eta\eta} + g^{33}\vec{r}_{\zeta\zeta} + 2g^{12}\vec{r}_{\xi\eta} + 2g^{23}\vec{r}_{\eta\zeta} + 2g^{31}\vec{r}_{\zeta\xi}\right) = 0 \qquad (14.19)$$

where the gg^{ij} are given by Equations (14.1.17) and (14.1.18). Because Equations (14.1.7) are homogeneous, we can use gg^{ij} in place of g^{ij} as long as g is positive, as it must be for a nonsingular transformation. We can test for positivity at each mesh point by using Equation (14.1.20):

$$\sqrt{g} = J = \begin{vmatrix} x_\xi & y_\xi & z_\xi \\ x_\eta & y_\eta & z_\eta \\ x_\zeta & y_\zeta & z_\zeta \end{vmatrix} \qquad (14.1.20)$$

and requiring that J>0.

To check that these equations reproduce the two-dimensional equations when there is no variation in one dimension, we take ζ as the invariant direction, thus reducing (14.1.19) to

$$gg^{11}\vec{r}_{\xi\xi} + 2gg^{12}\vec{r}_{\xi\eta} + gg^{22}\vec{r}_{\eta\eta} = 0 \tag{14.1.21}$$

If we let ζ = z, then the covariant base vectors become

$$\vec{a}_1 = x_\xi \vec{i} + y_\xi \vec{j} \tag{14.1.22a}$$

$$\vec{a}_2 = x_\eta \vec{i} + y_\eta \vec{j} \tag{14.1.22b}$$

$$\vec{a}_3 = \vec{k} \tag{14.1.22c}$$

From (14.1.22), using (14.1.13), we get

$$gg^{11} = x_\eta^2 + y_\eta^2 \tag{14.1.23a}$$

$$gg^{22} = x_\xi^2 + y_\xi^2 \tag{14.1.23b}$$

$$gg^{12} = -(x_\xi x_\eta + y_\xi y_\eta)\ . \tag{14.1.23c}$$

Substituting (14.1.23) into (14.1.21) yields the two-dimensional equipotential zoning Equations (14.1.2).

Before differencing Equations (14.1.19) we simplify the notation and write them in the form

$$\alpha_1\vec{r}_{\xi\xi} + \alpha_2\vec{r}_{\eta\eta} + \alpha_3\vec{r}_{\zeta\zeta} + 2\beta_1\vec{r}_{\xi\eta} + 2\beta_2\vec{r}_{\eta\zeta} + 2\beta_3\vec{r}_{\zeta\xi} = 0 \tag{14.1.24}$$

where

$$\alpha_1 = (x_\eta y_\zeta - x_\zeta y_\eta)^2 + (x_\eta z_\zeta - x_\zeta z_\eta)^2 + (y_\eta z_\zeta - y_\zeta z_\eta)^2 \tag{14.1.25a}$$

$$\alpha_2 = (x_\zeta y_\xi - x_\xi y_\zeta)^2 + (x_\zeta z_\xi - x_\xi z_\zeta)^2 + (y_\zeta z_\xi - y_\xi z_\zeta)^2 \tag{14.1.25b}$$

$$\alpha_3 = (x_\xi y_\eta - x_\eta y_\xi)^2 + (x_\xi z_\eta - x_\eta z_\xi)^2 + (y_\xi z_\eta - y_\eta z_\xi)^2 \tag{14.1.25c}$$

$$\beta_1 = (x_\xi x_\zeta + y_\xi y_\zeta + z_\xi z_\zeta)(x_\eta x_\zeta + y_\eta y_\zeta + z_\eta z_\zeta) - (x_\xi x_\eta + y_\xi y_\eta + z_\xi z_\eta)(x_\zeta^2 + y_\zeta^2 + z_\zeta^2) \quad (14.1.25d)$$

$$\beta_2 = (x_\eta x_\xi + y_\eta y_\xi + z_\eta z_\xi)(x_\xi x_\zeta + y_\xi y_\zeta + z_\xi z_\zeta) - (x_\zeta x_\eta + y_\zeta y_\eta + z_\zeta z_\eta)(x_\xi^2 + y_\xi^2 + z_\xi^2) \quad (14.1.25e)$$

$$\beta_3 = (x_\eta x_\zeta + y_\eta y_\zeta + z_\eta z_\zeta)(x_\xi x_\eta + y_\xi y_\eta + z_\xi z_\eta) - (x_\xi x_\eta + y_\xi y_\eta + z_\xi z_\eta)(x_\eta^2 + y_\eta^2 + z_\eta^2) \quad (14.1.25f)$$

We difference Equations (14.1.24) in a cube in the rectangular $\xi\eta\zeta$ space with unit spacing between the coordinate surfaces, using subscript i to represent the ξ direction, j the η direction, and k the ζ direction, as shown in Figure 14.1.

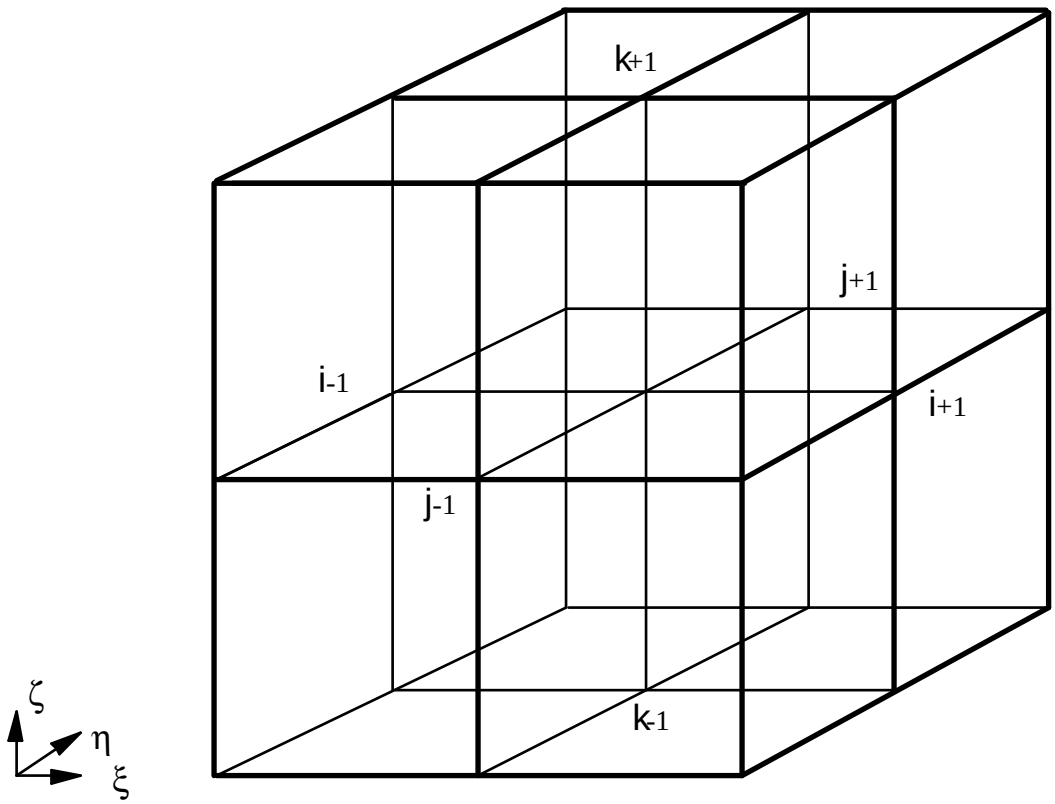


Figure 14.l.

Using central differencing, we obtain the following finite difference approximations for the coordinate derivatives:

$$\vec{r}_\xi = \left(\vec{r}_{i+1} - \vec{r}_{i-1}\right) \tag{14.1.26a}$$

$$\vec{r}_\eta = \left(\vec{r}_{j+1} - \vec{r}_{j-1}\right) \tag{14.1.26b}$$

$$\vec{r}_\zeta = \left(\vec{r}_{k+1} - \vec{r}_{k-1}\right) \tag{14.1.26c}$$

$$\vec{r}_{\xi\xi} = \left(\vec{r}_{i+1} - 2\vec{r} + \vec{r}_{i-1}\right) \tag{14.1.26d}$$

$$\vec{r}_{\eta\eta} = \left(\vec{r}_{j+1} - 2\vec{r} + \vec{r}_{j-1}\right) \tag{14.1.26e}$$

$$\vec{r}_{\zeta\zeta} = \left(\vec{r}_{k+1} - 2\vec{r} + \vec{r}_{k-1}\right) \tag{14.1.26f}$$

$$\vec{r}_{\xi\eta} = \frac{1}{4}\left[\left(\vec{r}_{i+1,\,j+1} + \vec{r}_{i-1,\,j-1}\right) - \left(\vec{r}_{i+1,\,j-1} + \vec{r}_{i-1,\,j+1}\right)\right] \tag{14.1.26g}$$

$$\vec{r}_{\eta\zeta} = \frac{1}{4}\left[\left(\vec{r}_{j+1,k+1} + \vec{r}_{j-1,k-1}\right) - \left(\vec{r}_{j+1,k-1} + \vec{r}_{j-1,k+1}\right)\right] \tag{14.1.26h}$$

$$\vec{r}_{\zeta\xi} = \frac{1}{4}\left[\left(\vec{r}_{I+1,k+1} + \vec{r}_{I-1,k-1}\right) - \left(\vec{r}_{I+1,k-1} + \vec{r}_{I-1,k+1}\right)\right] \tag{14.1.26i}$$

where for brevity we have omitted subscripts i, j, or k (e.g., k+1 stands for i, j, k+1). Note that these difference expressions use only coordinate planes that pass through the central point, and therefore do not include the eight corners of the cube.

Substituting Equations (14.1.26) into (14.1.24,14.1.25) and collecting terms, we get

$$\sum_{m=1}^{18} \omega_m(\vec{r}_m - \vec{r}) = 0 \tag{14.1.27}$$

where the sum is over the 18 nearest (in the transform space) neighbors of the given point. The coefficients ω_m are given in Table 14.1.

m	index	ω_m
1	i+1	α_1
2	i–1	α_1
3	j+1	α_2
4	j–1	α_2
5	k+1	α_3
6	k–1	α_3
7	i+1, j+1	$\beta_1/2$
8	i–1, j–1	$\beta_1/2$
9	i+1, j–1	$-\beta_1/2$
10	i–1, j+1	$-\beta_1/2$
11	j+1, k+1	$\beta_2/2$
12	j–1, k–1	$\beta_2/2$
13	j+1, k–1	$-\beta_2/2$
14	j–1, k+1	$-\beta_2/2$
15	i+1, k+1	$\beta_3/2$
16	i–1, k–1	$\beta_3/2$
17	i+1, k–1	$-\beta_3/2$
18	i–1, k+1	$-\beta_3/2$

Table 14.1: 3D Zoning Weight Coefficients

Equations (14.1.27) can be written

$$\vec{r}_m = \frac{\sum_m \omega_m \vec{r}_m}{\sum_m \omega_m} \tag{14.1.28}$$

expressing the position of the central point as a weighted mean of its 18 nearest neighbors. The denominator of (14.1.28) is equal to $2(\alpha_1+\alpha_2+\alpha_3)$ which is guaranteed to be positive by (14.1.25). This vector equation is equivalent to the three scalar equations

$$x = \frac{\sum_m \omega_m x_m}{\sum_m \omega_m} \tag{14.1.29a}$$

$$y = \frac{\sum_m \omega_m y_m}{\sum_m \omega_m} \tag{14.1.29b}$$

$$z = \frac{\sum_m \omega_m z_m}{\sum_m \omega_m} \tag{14.1.29c}$$

the same weights ω_m appearing in each equation.

These equations are nonlinear, since the coefficients are functions of the coordinates. Therefore we solve them by an iterative scheme, such as SOR. When applied to Equation (14.1.29a), for example, this gives for the $(n+1)^{st}$ iteration

$$x^{n+1} = (1-f)x^n + f\left(\frac{\sum_m \omega_m x_m}{\sum_m \omega_m}\right) \tag{14.1.30}$$

where the over relaxation factor f must satisfy $0<f<2$. In (14.1.30) the values of x_m at the neighboring points are the latest available values. The coefficients ω_m are recalculated before each iteration using Table 14.1 and Equations (14.1.25).

To smooth one interior point in a three-dimensional mesh, let the point to be smoothed be the interior point of Figure 14.1, assuming that its neighborhood has the logical structure shown. Even though Equations (14.1.29) are nonlinear, the ω_m do not involve the coordinates of the central point, since the α's and β's do not. Hence we simply solve Equations (14.1.29) for the new coordinates (x, y, z), holding the 18 neighboring points fixed, without needing to iterate.

If we wish to smooth a group of interior points, we solve iteratively for the coordinates using equations of the form (14.1.30).

14.1.2 Simple Averaging

The coordinates of a node is the simple average of the coordinates of its surrounding nodes.

$$\vec{x}_{SA}^{n+1} = \frac{1}{m^{tot}} \sum_{m=1}^{m^{tot}} \vec{x}_m^n \tag{14.1.31}$$

14.1.3 Kikuchi's Algorithm

Kikuchi proposed an algorithm that uses a volume-weighted average of the coordinates of the centroids of the elements surrounding a node. Variables that are subscripted with Greek letters refer to element variables, and subscripts with capital letters refer to the local node numbering within an element.

$$\vec{x}_\alpha^n = \frac{1}{8} \sum_A \vec{x}_A^n \tag{14.1.32a}$$

$$\vec{x}_K^{n+1} = \frac{\sum_{\alpha=1}^{\alpha^{tot}} V_a \vec{x}_\alpha^n}{\sum_{\alpha=1}^{\alpha^{tot}} V_a} \tag{14.1.32b}$$

14.1.4 Surface Smoothing

The surfaces are smoothed by extending the two-dimensional equipotential stencils to three dimensions. Notice that the form of Equation (14.1.2a) and (14.1.2b) for the x and y directions are identical. The third dimension, z, takes the same form. When Equation (14.1.2) is applied to all three dimensions, it tends to flatten out the surface and alter the total volume. To conserve the volume and retain the curvature of the surface, the point given by the relaxation stencil is projected on to the tangent plane defined by the normal at the node.

14.1.5 Combining Smoothing Algorithms

The user has the option of using a weighted average of all three algorithms to generate a composite algorithm, where the subscripts *E*, *SA*, and *K* refer to the equipotential, simple averaging, and Kikuchi's smoothing algorithm respectively, and w is the weighting factor.

$$\vec{x}^{n+1} = w_E \vec{x}_E^{n+1} + w_{SA} \vec{x}_{SA}^{n+1} + w_K \vec{x}_K^{n+1} \tag{14.1.33}$$

14.2 Advection Algorithms

LS-DYNA3D follows the SALE3D strategy for calculating the transport of the element-centered variables (i.e., density, internal energy, the stress tensor and the history variables). The van Leer MUSCL scheme [van Leer 1977] is used instead of the donor cell algorithm to calculate the values of the solution variables in the transport fluxes to achieve second order accurate monotonic results. To calculate the momentum transport, two algorithms have been implemented. The less expensive of the two is the one that is implemented in SALE3D, but it has known dispersion problems and may violate monotonocity (i.e., introduce spurious oscillations) [Benson 1992]. As an alternative, a significantly more expensive method [Benson 1992], which can be shown analytically to not have either problem on a regular mesh, has also implemented.

In this section the donor cell and van Leer MUSCL scheme are discussed. Both methods are one-dimensional and their extensions to multidimensional problems is discussed later.

14.2.1 Advection Methods in One Dimension

In this section the donor cell and van Leer MUSCL scheme are discussed. Both methods are one-dimensional and their extensions to multidimensional problems is discussed later.

The remap step maps the solution from a distorted Lagrangian mesh on to the new mesh. The underlying assumptions of the remap step are 1) the topology of the mesh is fixed (a complete rezone does not have this limitation), and 2) the mesh motion during a step is less than the characteristic lengths of the surrounding elements. Within the fluids community, the second condition is simply stated as saying the Courant number, C, is less than one.

$$C = \frac{u\Delta t}{\Delta x} = \frac{f}{V} \le 1 \tag{14.2.1}$$

Since the mesh motion does not occur over any physical time scale, Δt is arbitrary, and $u\Delta t$ is the transport volume, f, between adjacent elements. The transport volume calculation is purely geometrical for ALE formulations and it is not associated with any of the physics of the problem.

The algorithms for performing the remap step are taken from the computational fluids dynamics community, and they are referred to as “advection” algorithms after the first order, scalar conservation equation that is frequently used as a model hyperbolic problem.

$$\frac{\partial \phi}{\partial t} + a(x)\frac{\partial \phi}{\partial x} = 0 \qquad (14.2.2)$$

A good advection algorithm for the remap step is accurate, stable, conservative and monotonic. Although many of the solution variables, such as the stress and plastic strain, are not governed by conservation equations like momentum and energy, it is still highly desirable that the volume integral of all the solution variables remain unchanged by the remap step. Monotonicity requires that range of the solution variables does not increase during the remap. This is particularly important with mass and energy, where negative values would lead to physically unrealistic solutions.

Much of the research on advection algorithms has focused on developing monotonic algorithms with an accuracy that is at least second order. Not all recent algorithms are monotonic. For example, within the finite element community, the streamline upwind Petrov-Galerkin (SUPG) method developed by Hughes and coworkers [Brooks and Hughes 1982] is not monotonic. Johnson *et.al.* [1984] has demonstrated that the oscillations in the SUPG solution are localized, and its generalization to systems of conservation equations works very well for the Euler equations. Mizukami and Hughes [1985] later developed a monotonic SUPG formulation. The essentially non-oscillatory (ENO) [Harten 1989] finite difference algorithms are also not strictly monotonic, and work well for the Euler equations, but their application to hydrodynamics problems has resulted in negative densities [McGlaun 1990]. Virtually all the higher order methods that are commonly used were originally developed for solving the Euler equations, usually as higher order extensions to Godunov's method. Since the operator split approach is the dominant one in Eulerian hydrocodes, these methods are implemented only to solve the scalar advection equation.

The Donor Cell Algorithm. Aside from its first order accuracy, it is everything a good advection algorithm should be: stable, monotonic, and simple. The value of f_j^{ϕ} is dependent on the sign of a at node j, which defines the upstream direction.

$$\phi_{j+\frac{1}{2}}^{n+1} = \phi_{j+\frac{1}{2}}^{n} + \frac{\Delta t}{\Delta x}\left(f_j^{\phi} - f_{j+1}^{\phi}\right) \qquad (14.2.3a)$$

$$f_j^{\phi} = \frac{a_j}{2}\left(\phi_{j-\frac{1}{2}}^{n} + \phi_{j+\frac{1}{2}}^{n}\right) + \frac{|a_j|}{2}\left(\phi_{j-\frac{1}{2}}^{n} - \phi_{j+\frac{1}{2}}^{n}\right) \qquad (14.2.3b)$$

The donor cell algorithm is a first order Godunov method applied to the advection equation. The initial values of ϕ to the left and the right of node j are $\phi^n_{j-1/2}$ and $\phi^n_{j+1/2}(x)$, and the velocity of the contact discontinuity at node j is a_j.

The Van Leer MUSCL Algorithm. Van Leer [1977]introduced a family of higher order Godunov methods by improving the estimates of the initial values of left and right states for the Riemann problem at the nodes. The particular advection algorithm that is presented in this section is referred to as the MUSCL (monotone upwind schemes for conservation laws) algorithm for brevity, although MUSCL really refers to the family of algorithms that can be applied to systems of equations.

The donor cell algorithm assumes that the distribution of ϕ is constant over an element. Van Leer replaces the piecewise constant distribution with a higher order interpolation function, $\phi^n_{j+1/2}(x)$ that is subject to an element level conservation constraint. The value of ϕ at the element centroid is regarded in this context as the average value of ϕ over the element instead of the spatial value at $x_{j+1/2}$.

$$\phi^n_{j+1/2} = \int_{x_j}^{x_{j+1}} \phi^n_{j+1/2}(x)dx \tag{14.2.4}$$

To determine the range of ϕ, $\left[\phi^{\min}_{j+1/2}, \phi^{\max}_{j+1/2}\right]$, for imposing the monotonicity constraint, the maximum and minimum values of $\phi^n_{j-1/2}$, $\phi^n_{j+1/2}$, and $\phi^n_{j+3/2}$ are used. Monotonicity can be imposed in either of two ways. The first is to require that the maximum and minimum values of $\phi^n_{j+1/2}(x)$ fall within the range determined by the three elements. The second is to restrict the average value of ϕ in the transport volumes associated with element j + 1/2. While the difference may appear subtle, the actual difference between the two definitions is quite significant even at relatively low Courant numbers. The second definition allows the magnitude of the ϕ transported to adjacent elements to be larger than the first definition. As a consequence, the second definition is better able to transport solutions with large discontinuities. The magnitude of ϕ an algorithm is able to transport before its monotonicity algorithm restricts ϕ is a measure of the algorithm's "compressiveness."

The first step up from a piecewise constant function is a piecewise linear function, where x is now the volume coordinate. The volume coordinate of a point is simply the

volume swept along the path between the element centroid and the point. Conservation is guaranteed by expanding the linear function about the element centroid.

$$\phi^n_{j+\frac{1}{2}}(x) = S^n_{j+\frac{1}{2}}(x - x^n_{j+\frac{1}{2}}) + \phi^n_{j+\frac{1}{2}} \tag{14.2.5}$$

Letting $s^n_{j+\frac{1}{2}}$ be a second order approximation of the slope, the monotonicity limited value of the slope, $S^n_{j+\frac{1}{2}}$, according to the first limiting approach, is determined by assuming the maximum permissible values at the element boundaries.

$$S^n_{j+\frac{1}{2}} = \frac{1}{2}\Big(\mathrm{sgn}(s^L) + \mathrm{sgn}(s^R)\Big) \times \min\left(\left|s^L\right|, \left|s^n_{j+\frac{1}{2}}\right|, \left|s^R\right|\right) \tag{14.2.6a}$$

$$s^L = \frac{\phi^n_{j+\frac{1}{2}} - \phi^n_{j-\frac{1}{2}}}{\frac{1}{2}\Delta x_{j+\frac{1}{2}}} \tag{14.2.6b}$$

$$s^R = \frac{\phi^n_{j+\frac{3}{2}} - \phi^n_{j+\frac{1}{2}}}{\frac{1}{2}\Delta x_{j+\frac{1}{2}}} \tag{14.2.6c}$$

The second limiter is similar to the first, but it assumes that the maximum permissible values occur at the centroid of the transport volumes. Note that as stated in Equation (14.2.6), this limiter still limits the slope at the element boundary even if the element is the downstream element at that boundary. A more compressive limiter would not limit the slope based on the values of ϕ at the downstream boundaries. For example, if a_j is negative, only s^R would limit the value of s^n in Equation (14.2.6). If the element is the downstream element at both boundaries, then the slope in the element has no effect on the solution.

$$s^L = \frac{\phi^n_{j+\frac{1}{2}} - \phi^n_{j-\frac{1}{2}}}{\frac{1}{2}\Delta x_{j+\frac{1}{2}} - \frac{1}{2}\max(0, a_j \Delta t)} \tag{14.2.7a}$$

$$s^R = \frac{\phi^n_{j+\frac{3}{2}} - \phi^n_{j+\frac{1}{2}}}{\frac{1}{2}x_{j+\frac{1}{2}} + \frac{1}{2}\min(0, a_{j+1}\Delta t)} \tag{14.2.7b}$$

The flux at node j is evaluated using the upstream approximation of ϕ.

$$f_j^{\phi} = \frac{a_j}{2}\left(\phi_j^- + \phi_j^+\right) + \frac{|a_j|}{2}\left(\phi_j^- - \phi_j^+\right) \tag{14.2.8a}$$

$$\phi_j^+ = S^n_{j+1/2}(x^C - x^n_{j+1/2}) + \phi^n_{j+1/2} \tag{14.2.8b}$$

$$\phi_j^- = S^n_{j-1/2}(x^C - x^n_{j-1/2}) + \phi^n_{j-1/2} \tag{14.2.8c}$$

$$x^C = x_j^n + \frac{1}{2}\Delta t a_j \tag{14.2.8d}$$

The method for obtaining the higher order approximation of the slope is not unique. Perhaps the simplest approach is to fit a parabola through the centroids of the three adjacent elements and evaluate its slope at $x^n_{j+1/2}$. When the value of ϕ at the element centroids is assumed to be equal to the element average this algorithm defines a projection.

$$s^n_{j+1/2} = \frac{\left(\phi^n_{j+3/2} - \phi^n_{j+1/2}\right)\Delta x_j^2 + \left(\phi^n_{j+1/2} - \phi^n_{j-1/2}\right)\Delta x_{j+1}^2}{\Delta x_j \Delta x_{j+1}\left(\Delta x_j + \Delta x_{j+1}\right)} \tag{14.2.9a}$$

$$\Delta x_j = x^n_{j+1/2} - x^n_{j-1/2} \tag{14.2.9b}$$

14.2.2 Advection Methods in Three Dimensions

For programs that use a logically regular mesh, one-dimensional advection methods are extended to two and three dimensions by using a sequence of one-dimensional sweeps along the logically orthogonal mesh lines. This strategy is not possible for unstructured meshes because they don't have uniquely defined sweep directions through the mesh. CAVEAT [Addessio, *et. al.* 1986] uses one-dimensional sweeps in the spatial coordinate system, but their approach is expensive relative to the other algorithms and it does not always maintain spherical symmetry, which is an important consideration in underwater explosion calculations.

The advection in LS-DYNA3D is performed isotropically. The fluxes through each face of element A are calculated simultaneously, but the values of ϕ in the transport

volumes are calculated using the one-dimensional expressions developed in the previous sections.

$$\phi_A^{n+1} = \frac{1}{V_A^{n+1}} \left(V_A^n \phi_A^n + \sum_{j=1}^{6} f_j^{\phi} \right) \tag{14.2.10}$$

The disadvantage of isotropic advection is that there is no coupling between an element and the elements that are joined to it only at its corners and edges (i.e., elements that don't share faces). The lack of coupling introduces a second order error that is significant only when the transport is along the mesh diagonals.

The one-dimensional MUSCL scheme, which requires elements on either side of the element whose transport is being calculated, can not be used on the boundary elements in the direction normal to the boundary. Therefore, in the boundary elements, the donor cell algorithm is used to calculate the transport in the direction that is normal to the boundary, while the MUSCL scheme is used in the two tangential directions.

It is implicitly assumed by the transport calculations that the solution variables are defined per unit current volume. In LS-DYNA3D, some variables, such as the internal energy, are stored in terms of the initial volume of the element. These variables must be rescaled before transport, then the initial volume of the element is advected between the elements, and then the variables are rescaled using the new "initial" volumes. Hyperelastic materials are not currently advected in LS-DYNA3D because they require the deformation gradient, which is calculated from the initial geometry of the mesh. If the deformation gradient is integrated by using the midpoint rule, and it is advected with the other solution variables, then hyperelastic materials can be advected without any difficulties.

$$F^{n+1} = (I - \frac{\Delta t}{2L^{n+1/2}})^{-1} (I + \frac{\Delta t}{2L^{n+1/2}}) F^n \tag{14.2.11}$$

Advection of the Nodal Velocities. Except for the Godunov schemes, the velocity is centered at the nodes or the edges while the remaining variables are centered in the elements. Momentum is advected instead of the velocity in most codes to guarantee that momentum is conserved. The element-centered advection algorithms must be modified to advect the node-centered momentum. Similar difficulties are encountered when node-centered algorithms, such as the SUPG method [Brooks and Hughes 1982], are applied to element-centered quantities [Liu, Chang, and Belytschko, *to be published*]. There are two approaches: 1) construct a new mesh such that the nodes become the element centroids of

the new mesh and apply the element-centered advection algorithms, and 2) construct an auxiliary set of element-centered variables from the momentum, advect them, and then reconstruct the new velocities from the auxiliary variables. Both approaches can be made to work well, but their efficiency is heavily dependent on the architecture of the codes. The algorithms are presented in detail for one dimension first for clarity. Their extensions to three dimensions, which are presented later, are straight forward even if the equations do become lengthy. A detailed discussion of the algorithms in two dimensions is presented in Reference [Benson 1992].

Notation. Finite difference notation is used in this section so that the relative locations of the nodes and fluxes are clear. The algorithms are readily applied, however, to unstructured meshes. To avoid limiting the discussion to a particular element-centered advection algorithm, the transport volume through node i is f_i, the transported mass is $\tilde{f}_i$, and the flux of ϕ is $\phi_i f_i$. Most of the element-centered flux-limited advection algorithms calculate the flux of ϕ directly, but the mean value of ϕ in the transport volumes is calculated by dividing the $\phi_i f_i$, by the transport volume. A superscript "-" or "+" denotes the value of a variable before or after the advection. Using this notation, the advection of ϕ in one dimension is represented by Equation (14.2.12), where the volume is V.

$$\phi^+_{j+\frac{1}{2}} = \frac{(\phi^-_{j+\frac{1}{2}} V^-_{j+\frac{1}{2}} + \phi_i f_i - \phi_{i+1} f_{i+1})}{V^+} \qquad (14.2.12a)$$

$$V^+_{j+\frac{1}{2}} = V^-_{j+\frac{1}{2}} + f_i - f_{i+1} \qquad (14.2.12b)$$

The Staggered Mesh Algorithm. YAQUI [Amsden and Hirt 1973] was the first code to construct a new mesh that is staggered with respect to original mesh for momentum advection. The new mesh is defined so that the original nodes become the centroids of the new elements. The element-centered advection algorithms are applied to the new mesh to advect the momentum. In theory, the momentum can be advected with the transport volumes or the velocity can be advected with the mass.

$$v^+_j = \frac{(M^-_j v^-_j + v_{j-\frac{1}{2}} \tilde{f}_{j-\frac{1}{2}} - v_{j+\frac{1}{2}} \tilde{f}_{j+\frac{1}{2}})}{M^+_j} \qquad (14.2.13a)$$

$$v_j^+ = \frac{(M_j^- v_j^- + \{\rho v\}_{j-\frac{1}{2}} f_{j-\frac{1}{2}} - \{\rho v\}_{j+\frac{1}{2}} f_{j-\frac{1}{2}})}{M_j^+} \tag{14.2.13b}$$

$$M_j^+ = M_j^- + \tilde{f}_{j-\frac{1}{2}} - \tilde{f}_{j+\frac{1}{2}} \tag{14.2.13c}$$

A consistency condition, first defined by DeBar [1974], imposes a constraint on the formulation of the staggered mesh algorithm: if a body has a uniform velocity and a spatially varying density before the advection, then the velocity should be uniform and unchanged after the advection. The new mass of a node can be expressed in terms of the quantities used to advect the element-centered mass.

$$M_j^+ = \frac{1}{2}(M_{j-\frac{1}{2}}^+ + M_{j+\frac{1}{2}}^+) \tag{14.2.14}$$

$$M_j^+ = \frac{1}{2}(M_{j-\frac{1}{2}}^- + \rho_{j-1} f_{j-1} - \rho_j f_j + M_{j+\frac{1}{2}}^- + \rho_j f_j - \rho_{j+1} f_{j+1}) \tag{14.2.15a}$$

$$M_j^+ = M_j^- + \frac{1}{2}[(\rho_{j-1} f_{j-1} - \rho_j f_j) + (\rho_j f_j - \rho_{j+1} f_{j+1})] \tag{14.2.15b}$$

The staggered mass fluxes and transport volumes are defined by equating Equation (14.2.14) and Equation (14.2.15).

$$\rho_{j+\frac{1}{2}} f_{j+\frac{1}{2}} = \tilde{f}_{j+\frac{1}{2}} = \frac{1}{2}(\rho_j f_j + \rho_{j+1} f_{j+1}) \tag{14.2.16}$$

The density $\rho_{j+1/2}$ is generally a nonlinear function of the volume $f_{j+1/2}$, hence calculating $f_{j+1/2}$ from Equation (14.2.16) requires the solution of a nonlinear equation for each transport volume. In contrast, the mass flux is explicitly defined by Equation (14.2.16). Most codes, including KRAKEN [Debar 1974], CSQ [Thompson 1975], CTH [McGlaun 1989], and DYNA2D [Hallquist 1980], use mass fluxes with the staggered mesh algorithm because of their simplicity.

The dispersion characteristics of this algorithm are identical to the underlying element-centered algorithm by construction. This is not true, however, for some of the element-centered momentum advection algorithms. There are some difficulties in implementing the staggered mesh method in multidimensions. First, the number of edges defining a staggered element equals the number of elements surrounding the corresponding node. On an unstructured mesh, the arbitrary connectivity results in an

arbitrary number of edges for each staggered element. Most of the higher order accurate advection algorithms assume a logically regular mesh of quadrilateral elements, making it difficult to use them with the staggered mesh. Vectorization also becomes difficult because of the random number of edges that each staggered element might have. In the ALE calculations of DYNA2D, only the nodes that have a locally logically regular mesh surrounding them can be moved in order to avoid these difficulties [Benson 1992]. These difficulties do not occur in finite difference codes which process logically regular blocks of zones. Another criticism is the staggered mesh algorithm tends to smear out shocks because not all the advected variables are element-centered [Margolin 1989]. This is the primary reason, according to Margolin [1989], that the element-centered algorithm was adopted in SALE [Amdsden, Ruppel, and Hirt 1980].

The SALE Algorithm. SALE advects an element-centered momentum and redistributes its changes to the nodes [Amdsden, Ruppel, and Hirt 1980]. The mean element velocity, $\bar{v}_{j+1/2}$, specific momentum, $p_{j+1/2}$, element momentum, $P_{j+1/2}$, and nodal momentum are defined by Equation (14.2.17).

$$\bar{v}_{j+\frac{1}{2}} = \frac{1}{2}(v_j + v_{j+1}) \tag{14.2.17a}$$

$$p_{j+\frac{1}{2}} = \rho_{j+\frac{1}{2}} \bar{v}_{j+\frac{1}{2}} \tag{14.2.17b}$$

$$P_{j+\frac{1}{2}} = M_{j+\frac{1}{2}} \bar{v}_{j+\frac{1}{2}} \tag{14.2.17c}$$

Denoting the change in the element momentum $\Delta P_{j+1/2}$, the change in the velocity at a node is calculated by distributing half the momentum change from the two adjacent elements.

$$\Delta P_{j-\frac{1}{2}} = p_{j-1} f_{j-1} - p_j f_j \tag{14.2.18a}$$

$$P_j^+ = P_j^- + \frac{1}{2}(\Delta P_{j-\frac{1}{2}} + \Delta P_{j+\frac{1}{2}}) \tag{14.2.18b}$$

$$v_j^+ = \frac{P_j^+}{M_j^+} \tag{14.2.18c}$$

This algorithm can also be implemented by advecting the mean velocity, $\bar{v}_{j+\frac{1}{2}}$ with the transported mass, and the transported momentum $p_j f_j$ is changed to $\bar{v}_j \tilde{f}_j$.

The consistency condition is satisfied regardless of whether masses or volumes are used. Note that the velocity is not updated from the updated values of the adjacent element momenta. The reason for this is the original velocities are not recovered if $f_i = 0$, which indicates that there is an inversion error associated with the algorithm.

The HIS (Half Index Shift) Algorithm. Benson [1992] developed this algorithm based on his analysis of other element-centered advection algorithms. It is designed to overcome the dispersion errors of the SALE algorithm and to preserve the monotonicity of the velocity field. The SALE algorithm is a special case of a general class of algorithms. To sketch the idea behind the HIS algorithm, the discussion is restricted to the scalar advection equation. Two variables, $\Psi_{1,\,j+\frac{1}{2}}$ and $\Psi_{2,\,j+\frac{1}{2}}$ are defined in terms of a linear transformation of ϕ_j and ϕ_{j+1}. The linear transformation may be a function of the element j + 1/2.

$$\begin{Bmatrix} \Psi^-_{1,\,j+\frac{1}{2}} \\ \Psi^-_{2,\,j+\frac{1}{2}} \end{Bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{Bmatrix} \phi^-_j \\ \phi^-_{j+1} \end{Bmatrix} \tag{14.2.19}$$

This relation is readily inverted.

$$\begin{Bmatrix} \phi^+_j \\ \phi^+_{j+1} \end{Bmatrix} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \begin{Bmatrix} \Psi^+_{1,\,j+\frac{1}{2}} \\ \Psi^+_{2,\,j+\frac{1}{2}} \end{Bmatrix} \tag{14.2.20}$$

A function is monotonic over an interval if its derivative does not change sign. The sum of two monotonic functions is monotonic, but their difference is not necessarily monotonic. As a consequence, $\Psi^-_{1,\,j+\frac{1}{2}}$ and $\Psi^-_{2,\,j+\frac{1}{2}}$ are monotonic over the same intervals as ϕ^-_j if all the coefficients in the linear transformation have the same sign. On the other hand, ϕ^+_j is not necessarily monotonic even if $\Psi^+_{1,\,j+\frac{1}{2}}$ and $\Psi^+_{2,\,j+\frac{1}{2}}$ are monotonic because of the appearance of the negative signs in the inverse matrix. Monotonicity can be maintained by transforming in both directions provided that the transformation matrix is diagonal. Symmetry in the overall algorithm is obtained by using a weighted average of the values of ϕ_j calculated in elements j + 1/2 and j – 1/2.

A monotonic element-centered momentum advection algorithm is obtained by choosing the identity matrix for the transformation and by using mass weighting for the inverse relationship.

$$\begin{Bmatrix} \Psi_{1,\,j+\frac{1}{2}} \\ \Psi_{2,\,j+\frac{1}{2}} \end{Bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} v_j^- \\ v_{j+1}^- \end{Bmatrix} \tag{14.2.21}$$

To conserve momentum, Ψ is advected with the transport masses.

$$\Psi_{m,\,j+1/2}^{+} = \left(M_{j+1/2}^{-} \Psi_{m,\,j+1/2}^{-} + \Psi_{m,\,j}^{-} \tilde{f}_j - \Psi_{m,\,j+1}^{-} \tilde{f}_{j+1} \right) / M_{j+1/2}^{+} \tag{14.2.22}$$

$$v_j = \frac{1}{2M_j} \left(M_{j+1/2} \Psi_{1,\,j+1/2} + M_{j-1/2} \Psi_{2,\,j-1/2} \right) \tag{14.2.23}$$

Dispersion Errors. A von Neumann analysis [Trefethen 1982] characterizes the dispersion errors of linear advection algorithms. Since the momentum advection algorithm modifies the underlying element-centered advection algorithm, the momentum advection algorithm does not necessarily have the same dispersion characteristics as the underlying algorithm. The von Neumann analysis provides a tool to explore the changes in the dispersion characteristics without considering a particular underlying advection algorithm.

The model problem is the linear advection equation with a constant value of c. A class of solutions can be expressed as complex exponentials, where i is $\sqrt{-1}$, ω is the frequency, and χ is the wave number.

$$\frac{\partial \phi}{\partial t} + c \frac{\partial \phi}{\partial x} = 0 \tag{14.2.24a}$$

$$\phi(x,t) = e^{i(\omega t - \chi x)} \tag{14.2.24b}$$

For Equation (14.2.24), the dispersion equation is $\omega = c\chi$, but for discrete approximations of the equation and for general hyperbolic equations, the relation is $\omega = \omega\chi$. The phase velocity, c_p, and the group speed, c_g, are defined by Equation (14.2.25).

$$c_p = \frac{\omega}{\chi} \tag{14.2.25a}$$

$$c_g = \frac{\partial \omega}{\partial \chi} \tag{14.2.25b}$$

The mesh spacing is assumed to have a constant value J, and the time step, h, is also constant. The + and - states in the previous discussions correspond to times n and n+1 in the dispersion analysis. An explicit linear advection method that has the form given by Equation (14.2.26) results in a complex dispersion equation, Equation (14.2.27), where Π is a complex polynomial.

$$\phi_j^{n+1} = \phi_j^n + F(c, h, J, ..., \phi_{j-1}^n, \phi_j^n, \phi_{j+1}^n, ...) \tag{14.2.26}$$

$$e^{i\omega h} = 1 + P(e^{i\chi J}) \tag{14.2.27a}$$

$$\Pi(e^{i\chi J}) = \sum_j \beta_j e^{i\chi jJ} \tag{14.2.27b}$$

The dispersion equation has the general form given in Equation (14.2.28), where Π_r and Π_i denote the real and imaginary parts of Π, respectively.

$$\omega h = \tan^{-1}\left(\frac{\Pi_i}{1+\Pi_r}\right) \tag{14.2.28}$$

Recognizing that the relations in the above equations are periodic in ωh and χ^J, the normalized frequency and wave number are defined to simplify the notation.

$$\overline{\omega} = \omega h \qquad \overline{\chi} = \chi J \tag{14.2.29}$$

The von Neumann analysis of the SALE algorithm proceeds by first calculating the increment in the cell momentum.

$$P_{j+\frac{1}{2}}^n = \frac{1}{2}\left(v_j^n + v_{j+1}^n\right) \tag{14.2.30a}$$

$$P_{j+\frac{1}{2}}^n = \frac{1}{2}\left(1 + e^{-i\overline{\chi}}\right)v_j^n \tag{14.2.30b}$$

$$\Delta P_{j+\frac{1}{2}}^{n+1} = P_{j+\frac{1}{2}}^{n+1} - P_{j+\frac{1}{2}}^n \tag{14.2.30c}$$

$$\Delta P_{j+\frac{1}{2}}^{n+1} = \frac{1}{2}\left(1+e^{-i\bar{\chi}}\right)\Pi v_j^n \tag{14.2.30d}$$

The velocity is updated from the changes in the cell momentum.

$$v_j^{n+1} = v_j^n + \frac{1}{2}\left(\Delta P_{j+\frac{1}{2}}^{n+1} + \Delta P_{j-\frac{1}{2}}^{n+1}\right) \tag{14.2.31a}$$

$$v_j^{n+1} = \frac{1}{4}\left(1+e^{i\bar{\chi}}\right)\left(1+e^{-i\bar{\chi}}\right)\Pi v_j^n \tag{14.2.31b}$$

$$v_j^{n+1} = \frac{1}{2}\left(1+\cos(\bar{\chi})\right)\Pi v_j^n \tag{14.2.31c}$$

The dispersion relation for the SALE advection algorithm is given by Equation (14.2.32).

$$\bar{\omega} = \tan^{-1}\left(\frac{\frac{1}{2}\left(1+\cos(\bar{\chi})\right)\Pi_i}{1+\frac{1}{2}\left(1+\cos(\bar{\chi})\right)\Pi_r}\right) \tag{14.2.32}$$

By comparing Equation (14.2.28) and Equation (14.2.32), the effect of the SALE momentum advection algorithm on the dispersion is to introduce a factor λ, equal to $\frac{1}{2}\left(1+\cos(\bar{\chi})\right)\Pi$, into the spatial part of the advection stencil. For small values of $\bar{\chi}$, λ is close to one, and the dispersion characteristics are not changed, but when $\bar{\chi}$ is π, the phase and group velocity go to zero and the amplification factor is one independent of the underlying advection algorithm. Not only is the wave not transported, it is not damped out. The same effect is found in two dimensions, where λ, has the form $\frac{1}{4}\left(1+\cos(\bar{\chi})+\cos(\bar{\bar{\chi}})+\cos(\bar{\chi})\cos(\bar{\bar{\chi}})\right)$.

In contrast, none of the other algorithms alter the dispersion characteristics of the underlying algorithm. Benson has demonstrated for the element-centered algorithms that the SALE inversion error and the dispersion problem are linked. Algorithms that fall into the same general class as the SALE and HIS algorithms will, therefore, not have dispersion problems [Benson 1992].

Three-Dimensional Momentum Advection Algorithms. The momentum advection algorithms discussed in the previous sections are extended to three dimensions in a straight forward manner. The staggered mesh algorithm requires the construction of a staggered mesh and the appropriate transport masses. Based on the consistency arguments, the appropriate transport masses are given by Equation (14.2.33).

$$\tilde{f}_{j+1/2,k,l} = \frac{1}{8}\sum_{J=j}^{j+1}\sum_{K=k-\frac{1}{2}}^{k+\frac{1}{2}}\sum_{L=l-\frac{1}{2}}^{l+\frac{1}{2}} f_{J,K,L} \tag{14.2.33}$$

The SALE advection algorithm calculates the average momentum of the element from the four velocities at the nodes and distributes $1/8$ of the change in momentum to each node.

$$p_{j+\frac{1}{2},k+\frac{1}{2},l+\frac{1}{2}} = \frac{1}{8}\rho_{j+\frac{1}{2},k+\frac{1}{2},l+\frac{1}{2}}\sum_{J=j}^{j+1}\sum_{K=k}^{k+1}\sum_{L=l}^{l+1} v_{JKL} \tag{14.2.34a}$$

$$p_{j+\frac{1}{2},k+\frac{1}{2},l+\frac{1}{2}} = \frac{1}{8}M_{j+\frac{1}{2},k+\frac{1}{2},l+\frac{1}{2}}\sum_{J=j}^{j+1}\sum_{K=k}^{k+1}\sum_{L=l}^{l+1} v_{JKL} \tag{14.2.34b}$$

$$v^{+}_{j,k,l} = \frac{1}{M^{+}_{j,k,l}}\left(M^{-}_{j,k,l}v^{-}_{j,k,l} + \frac{1}{8}\sum_{J=j-\frac{1}{2}}^{j+\frac{1}{2}}\sum_{K=k-\frac{1}{2}}^{k+\frac{1}{2}}\sum_{L=l-\frac{1}{2}}^{l+\frac{1}{2}}\Delta P_{J,K,L}\right) \tag{14.2.34c}$$

The HIS algorithm is also readily extended to three dimensions. The variable definitions are given in Equation (14.2.35) and Equation (14.2.36), where the subscript A refers to the local numbering of the nodes in the element. In an unstructured mesh, the relative orientation of the nodal numbering within the elements may change. The subscript A is always with reference to the numbering in element j,k,l. The subscript $\tilde{A}$ is the local node number in an adjacent element that refers to the same global node number as A.

$$\Psi_{A,\,j+\frac{1}{2},k+\frac{1}{2},l+\frac{1}{2}} = v_{A,\,j+\frac{1}{2},k+\frac{1}{2},l+\frac{1}{2}} \tag{14.2.35}$$

$$v^{+}_{j,k,l} = \frac{1}{M^{+}_{j,k,l}}\sum_{J=j-\frac{1}{2}}^{j+\frac{1}{2}}\sum_{K=k-\frac{1}{2}}^{k+\frac{1}{2}}\sum_{L=l-\frac{1}{2}}^{l+\frac{1}{2}} M_{J,K,L}\Psi^{+}_{\tilde{A},J,K,L} \tag{14.2.36}$$

14.3 The Manual Rezone

The central limitation to the simplified ALE formulation is that the topology of the mesh is fixed. For a problem involving large deformations, a mesh that works well at early times may not work at late times regardless of how the mesh is distributed over the material domain. To circumvent this difficulty, a manual rezoning capability has been implemented in LS-DYNA3D. The general procedure is to 1) interrupt the calculation when the mesh is no longer acceptable, 2) generate a new mesh with INGRID by using the current material boundaries from LS-DYNA3D (the topologies of the new and old mesh are unrelated), 3) remap the solution from the old mesh to the new mesh, and 4) restart the calculation.

This chapter will concentrate on the remapping algorithm since the mesh generation capability is documented in the INGRID manual [Stillman and Hallquist 1992]. The remapping algorithm first constructs an interpolation function on the original mesh by using a least squares procedure, and then interpolates for the solution values on the new mesh.

The one point quadrature used in LS-DYNA3D implies a piecewise constant distribution of the solution variables within the elements. A piecewise constant distribution is not acceptable for a rezoner since it implies that for evenly moderately large changes in the locations of the nodes (say, displacements on the order of fifty percent of the elements characteristic lengths) that there will be no changes in the values of the element-centered solution variables. A least squares algorithm is used to generate values for the solution variables at the nodes from the element-centered values. The values of the solution variables can then be interpolated from the nodal values, ϕ_A, using the standard trilinear shape functions anywhere within the mesh.

$$\phi(\xi,\eta,\zeta) = \phi_A N_A(\xi,\eta,\zeta) \tag{14.3.1}$$

The objective function for minimization, J, is defined material by material, and each material is remapped independently .

$$J = \frac{1}{2}\int_V (\phi_A N_A - \phi)^2 \, dV \tag{14.3.2}$$

The objective function is minimized by setting the derivatives of J with respect to ϕ_A equal to zero.

$$\frac{\partial J}{\partial \phi_A} = \int_V (\phi_B N_B - \phi) N_A dV = 0 \tag{14.3.3}$$

The least square values of ϕ_A are calculated by solving the system of linear equations, Equation (14.3.4).

$$M_{AB}\phi_B = \int_V N_A \phi dV \tag{14.3.4a}$$

$$M_{AB} = \int_V N_A N_B dV \tag{14.3.4b}$$

The "mass matrix," M_{AB}, is lumped to give a diagonal matrix. This eliminates the spurious oscillations that occur in a least squares fit around the discontinuities in the solution (e.g., shock waves) and facilitates an explicit solution for ϕ_A. The integral on the right hand side of Equation (14.3.4a) is evaluated using one point integration. By introducing these simplification, Equation (14.3.4) is reduced to Equation (14.3.5), where the summation over α is restricted to the elements containing node A.

$$\phi_A = \frac{\sum_\alpha \phi_\alpha V_\alpha}{\sum_\alpha V_\alpha} \tag{14.3.5}$$

The value of ϕ_α is the mean value of ϕ in element α. From this definition, the value of ϕ_α is calculated using Equation (14.3.6).

$$\phi_\alpha = \frac{1}{V_\alpha} \int_{V_\alpha} \phi_A N_A dV \tag{14.3.6}$$

The integrand in Equation (14.3.6) is defined on the old mesh, so that Equation (14.3.6) is actually performed on the region of the old mesh that overlaps element α in the new mesh, where the superscript "*" refers to elements on the old mesh.

$$\phi_\alpha = \frac{1}{V_\alpha} \sum_\beta {}^{*} \int_{V_\alpha \mid V_\beta^*} \phi_A N_A dV^{*} \tag{14.3.7}$$

One point integration is currently used to evaluate Equation (14.3.7), although it would be a trivial matter to add higher order integration. By introducing this simplification,

Equation (14.3.7) reduces to interpolating the value of ϕ_α from the least squares fit on the old mesh.

$$\phi_\alpha = \phi_A N_A(\xi^*,\eta^*,\zeta^*) \tag{14.3.8}$$

The isoparametric coordinates in the old mesh that correspond to the spatial location of the new element centroid must be calculated for Equation (14.3.8). The algorithm that is described here is from Shapiro [1990], who references [Thompson and Maffeo 1985, Maffeo 1984, Maffeo 1985] as the motivations for his current strategy, and we follow his notation. The algorithm uses a "coarse filter" and a "fine filter" to make the search for the correct element in the old mesh efficient.

The coarse filter calculates the minimum and maximum coordinates of each element in the old mesh. If the new element centroid, (x_s,y_s,z_s), lies outside of the box defined by the maximum and minimum values of an old element, then the old element does not contain the new element centroid.

Several elements may pass the coarse filter but only one of them contains the new centroid. The fine filter is used to determine if an element actually contains the new centroid. The fine filter algorithm will be explained in detail for the two-dimensional case since it easier to visualize than the three-dimensional case, but the final equations will be given for the three-dimensional case.

The two edges adjacent to each node in Figure 14.2 (taken from [Shapiro 1990]) define four skew coordinate systems. If the coordinates for the new centroid are positive for each coordinate system, then the new centroid is located within the old element. Because of the overlap of the four positive quarter spaces defined by the skew coordinate systems, only two coordinate systems actually have to be checked. Using the first and third coordinate systems, the coordinates, α_i, are the solution of Equation (14.3.9).

$$V_s = V_1 + \alpha_1 V_{12} + \alpha_2 V_{14} \tag{14.3.9a}$$

$$V_s = V_3 + \alpha_3 V_{32} + \alpha_4 V_{34} \tag{14.3.9b}$$

Two sets of linear equations are generated for the α_i by expanding the vector equations.

$$\begin{bmatrix} x_2 - x_1 & x_4 - x_1 \\ y_2 - y_1 & y_4 - y_1 \end{bmatrix} \begin{Bmatrix} \alpha_1 \\ \alpha_2 \end{Bmatrix} = \begin{Bmatrix} x_s - x_1 \\ y_s - y_1 \end{Bmatrix} \tag{14.3.10a}$$

$$\begin{bmatrix} x_2 - x_3 & x_4 - x_3 \\ y_2 - y_3 & y_4 - y_3 \end{bmatrix} \begin{Bmatrix} \alpha_3 \\ \alpha_4 \end{Bmatrix} = \begin{Bmatrix} x_s - x_3 \\ y_s - y_3 \end{Bmatrix} \tag{14.3.10b}$$

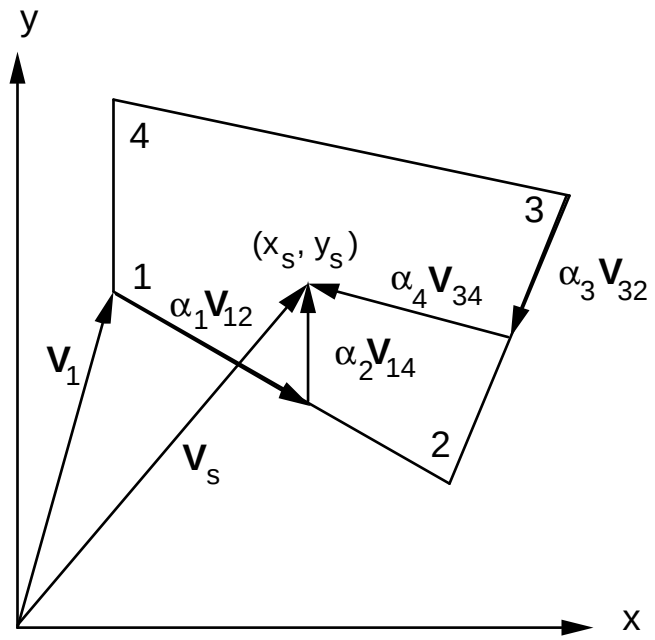


Figure 14.2: Skew Coordinate System

The generalization of Equation (14.3.9) to three dimensions is given by Equation (14.3.11), and it requires the solution of four sets of three equations. The numbering convention for the nodes in Equation (14.3.11) follows the standard numbering scheme used in LS-DYNA3D for eight node solid elements.

$$V_s = V_1 + \alpha_1 V_{12} + \alpha_2 V_{14} + \alpha_3 V_{15} \qquad (14.3.11a)$$

$$V_s = V_3 + \alpha_4 V_{37} + \alpha_5 V_{34} + \alpha_6 V_{32} \qquad (14.3.11b)$$

$$V_s = V_6 + \alpha_7 V_{62} + \alpha_8 V_{65} + \alpha_9 V_{67} \qquad (14.3.11c)$$

$$V_s = V_8 + \alpha_{10} V_{85} + \alpha_{11} V_{84} + \alpha_{12} V_{87} \qquad (14.3.11d)$$

The fine filter sometimes fails to locate the correct element when the mesh is distorted. When this occurs, the element that is closest to the new centroid is located by finding the element for which the sum of the distances between the new centroid and the nodes of the element is a minimum.

Once the correct element is found, the isoparametric coordinates are calculated using the Newton-Raphson method, which usually converges in three or four iterations.

$$\begin{bmatrix} x_A \dfrac{\partial N_A}{\partial \xi} & x_A \dfrac{\partial N_A}{\partial \eta} & x_A \dfrac{\partial N_A}{\partial \zeta} \\ y_A \dfrac{\partial N_A}{\partial \xi} & y_A \dfrac{\partial N_A}{\partial \eta} & y_A \dfrac{\partial N_A}{\partial \zeta} \\ z_A \dfrac{\partial N_A}{\partial \xi} & z_A \dfrac{\partial N_A}{\partial \eta} & z_A \dfrac{\partial N_A}{\partial \zeta} \end{bmatrix} \begin{Bmatrix} \Delta\xi \\ \Delta\eta \\ \Delta\zeta \end{Bmatrix} = \begin{Bmatrix} x_s - x_A N_A \\ y_s - y_A N_A \\ z_s - z_A N_A \end{Bmatrix} \tag{14.3.12}$$

$$\xi^{i+1} = \xi^i + \Delta\xi \tag{14.3.13a}$$

$$\eta^{i+1} = \eta^i + \Delta\eta \tag{14.3.13b}$$

$$\zeta^{i+1} = \zeta^i + \Delta\zeta \tag{14.3.13c}$$

15. STRESS UPDATE OVERVIEW

15.1 Jaumann Stress Rate

Stresses for material which exhibit elastic-plastic and soil-like behavior (hypoelastic) are integrated incrementally in time:

$$\sigma_{ij}\,(t + dt) = \sigma_{ij}\,(t) + \dot{\sigma}_{ij}dt \tag{15.1}$$

Here, and in equations which follow, we neglect the contribution of the bulk viscosity to the stress tensor. In Equation (15.1), the dot denotes the material time derivative given by

$$\dot{\sigma}_{ij} = \sigma_{ij}^{\nabla} + \sigma_{ik}\omega_{kj} + \sigma_{jk}\omega_{ki} \tag{15.2}$$

in which

$$\omega_{ij} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} - \frac{\partial v_j}{\partial x_i}\right) \tag{15.3}$$

is the spin tensor and

$$\sigma_{ij}^{\nabla} = C_{ijkl}\dot{\varepsilon}_{kl} \tag{15.4}$$

is the Jaumann (co-rotational) stress rate. In Equation 15.4, C_{ijkl} is the stress dependent constitutive matrix, v_i, is the velocity vector, and $\dot{\varepsilon}_{ij}$ is the strain rate tensor:

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) \tag{15.5}$$

In the implementation of Equation (15.1) we first perform the stress rotation, Equation (15.2), and then call a constitutive subroutine to add the incremental stress components σ_{ij}^{∇}. This may be written as

$$\sigma_{ij}^{n+1} = \sigma_{ij}^{n} + r_{ij}^{n} + \sigma_{ij}^{\nabla n+1/2}\Delta t^{n+1/2} \tag{15.6}$$

where

$$\sigma_{ij}^{\nabla n+\frac{1}{2}}\Delta t^{n+\frac{1}{2}} = C_{ijkl}\Delta\varepsilon_{kl}^{n+\frac{1}{2}}$$
$$\Delta\varepsilon_{ij}^{n+\frac{1}{2}} = \dot{\varepsilon}_{ij}^{n+\frac{1}{2}}\Delta t^{n+\frac{1}{2}} \tag{15.7}$$

and r_{ij}^{n} gives the rotation of the stress at time t^n to the configuration at t^{n+1}

$$r_{ij}^{n} = \left(\sigma_{ip}^{n}\omega_{pj}^{n+\frac{1}{2}} + \sigma_{jp}^{n}\omega_{pi}^{n+\frac{1}{2}}\right)\Delta t^{n+\frac{1}{2}} \tag{15.8}$$

In the implicit NIKE2D/3D [Hallquist 1981b] codes which we often use for low frequency structural response, we do a half-step rotation, apply the constitutive law, and complete the second half-step rotation on the modified stress. An exact rotation is performed rather than the approximate one represented by Equation (15.3), which is valid only for small rotations. A typical time step in the NIKE codes is usually 100 to 1000 or more times larger than the explicit DYNA3D time step; consequently, the direct use of Equation (15.7) could lead to very significant errors.

15.2 Jaumann Stress Rate Used With Equations of State

If pressure is defined by an equation of state as a function of relative volume, V, and energy, E, or temperature, T,

$$p = p\,(V, E) = p(V, T) \tag{15.9}$$

we update the deviatoric stress components

$$s_{ij}^{n+1} = \sigma_{ij}^{n} + r_{ij}^{n} + p^{n}\delta_{ij} + C_{ijkl}\dot{\varepsilon}_{kl}'^{n+\frac{1}{2}}\Delta t^{n+\frac{1}{2}} \tag{15.10}$$

where $\dot{\varepsilon}_{ij}'^{n+\frac{1}{2}}$ is the deviatoric strain rate tensor:

$$\dot{\varepsilon}_{ij}'^{n+\frac{1}{2}} = \dot{\varepsilon}_{ij} - \frac{1}{3}\dot{\varepsilon}_{kk}\delta \tag{15.11}$$

Before the equation of state (Equation (15.9)) is evaluated, we compute the bulk viscosity, q, and update the total internal energy e of the element being processed to a trial value e*:

$$e^{*n+1} = e^n - \frac{1}{2}\Delta v\left(p^n + q^{n-\frac{1}{2}} + q^{n+\frac{1}{2}} \right) + v^{n+\frac{1}{2}} s_{ij}^{n+\frac{1}{2}} \Delta\varepsilon_{ij}^{n+\frac{1}{2}} \tag{15.12}$$

where v is the element volume and

$$\begin{aligned} \Delta v &= v^{n+1} - v^n, \quad v^{n+\frac{1}{2}} = \frac{1}{2}\left(v^n + v^{n+1} \right) \\ s_{ij}^{n+\frac{1}{2}} &= \frac{1}{2}\left(s_{ij}^n + s_{ij}^{n+1} \right) \end{aligned} \tag{15.13}$$

The time-centering of the viscosity is explained by Noh [1976].

Assume we have an equation of state that is linear in internal energy of the form

$$p^{n+1} = A^{n+1} + B^{n+1} E^{n+1} \tag{15.14}$$

where

$$E^{n+1} = \frac{e^{n+1}}{v_0} \tag{15.15}$$

and υ_0 is the initial volume of the element. Noting that

$$e^{n+1} = e^{*n+1} - \frac{1}{2}\Delta v p^{n+1} \tag{15.16}$$

pressure can be evaluated exactly by solving the implicit form

$$p^{n+1} = \frac{\left(A^{n+1} + B^{n+1} E^{*n+1} \right)}{\left(1 + \frac{1}{2} B^{n+1} \frac{\Delta v}{v_0} \right)} \tag{15.17}$$

and the internal energy can be updated in Equation (15.16). If the equation of state is not linear in internal energy, a one-step iteration is used to approximate the pressure

$$p^{*n+1} = p\ (V^{n+1}, E^{*n+1}) \tag{15.18}$$

Internal energy is updated to n +1 using p^{*n+1} in Equation (15.16) and the final pressure is then computed:

$$p^{n+1} = p\ (V^{n+1}, E^{n+1}) \tag{15.19}$$

This is also the iteration procedure used in KOVEC [Woodruff 1973]. All the equations of state in DYNA3D are linear in energy except the ratio of polynomials.

15.3 Green-Naghdi Stress Rate

The Green-Naghdi rate is defined as

$$\sigma_{ij}^{\nabla} = \dot{\sigma}_{ij} + \sigma_{ik}\Omega_{kj} + \sigma_{jk}\Omega_{ki} = R_{ik}R_{jl}\dot{\tau}_{kl} \tag{15.20}$$

where Ω_{ij} is defined as

$$\Omega_{ij} = \dot{R}_{ik} R_{jk} \tag{15.21}$$

and R is found by application of the polar decomposition theorem

$$F_{ij} = R_{ik} U_{kj} = V_{ik} R_{kj} \tag{15.22}$$

F_{ij} is the deformation gradient matrix and U_{ij} and V_{ij} are the positive definite right and left stretch tensors:

$$F_{ij} = \frac{\partial x_i}{\partial X_j} \tag{15.23}$$

Stresses are updated for all materials by adding to the rotated Cauchy stress at time n.

$$\tau_{ij}^{n} = R_{ki}^{n} R_{lj}^{n} \sigma_{kl}^{n} \tag{15.24}$$

the stress increment obtained by an evaluation of the constitutive equations,

$$\Delta\tau_{ij}^{n+1/2} = C_{ijkl}\,\Delta d_{kl}^{n+1/2} \tag{15.25}$$

where

$$\Delta d_{ij}^{n+1/2} = R_{ki}^{n+1/2} R_{lj}^{n+1/2} \Delta\varepsilon_{kl}^{n+1/2} \tag{15.26}$$

C_{ijkl} = constitutive matrix

$\Delta\varepsilon_{kl}$ = increment in strain

and to obtain the rotated Cauchy stress at n+1, i.e.,

$$\tau_{ij}^{n+1} = \tau_{ij}^{n} + \Delta\tau_{ij}^{n+1/2} \tag{15.27}$$

The desired Cauchy stress at n+1 can now be found

$$\sigma_{ij}^{n+1} = R_{ik}^{n+1} R_{jl}^{n+1} \tau_{kl}^{n+1} \tag{15.28}$$

Because we evaluate our constitutive models in the rotated configuration, we avoid the need to transform history variables such as the back stress that arises in kinematic hardening.

In the computation of R, Taylor and Flanagan [1989] did an incremental update in contrast with the direct polar decomposition used in the NIKE3D code. Following their notation the algorithm is given by.

$$z_i = e_{ijk} V_{jm} \dot{\varepsilon}_{mk}$$

$$\varpi_i = e_{ijk}\omega_{jk} - 2\left[V_{ij} - \delta_{ij}V_{kk}\right]^{-1} z_j$$

$$\Omega_{ij} = \frac{1}{2} e_{ijk}\varpi_k$$

$$\left(\delta_{ik} - \tfrac{\Delta t}{2}\Omega_{ik}\right)R_{kj}^{n+1} = \left(\delta_{ik} + \tfrac{\Delta t}{2}\Omega_{ik}\right)R_{kj}^{n} \tag{15.29}$$

$$\dot{V}_{ij} = (\dot{\varepsilon}_{ik} + \omega_{ik})V_{kj} - V_{ik}\Omega_{kj}$$

$$V_{ij}^{n+1} = V_{ij}^{n} + \Delta t \dot{V}_{ij}$$

We have adopted the PRONTO3D approach in LS-DYNA3D due to numerical difficulties with the polar decomposition in NIKE3D. We believe the PRONTO3D approach is reliable. Several disadvantages of the PRONTO3D approach include 300+ operations (at least that is the number we got), the requirement of 15 additional variables per integration point, and if we rezone the material in the future the initial geometry will need to be remapped and the 15 variables initialized.

15.4 Elastoplastic Materials

At low stress levels in elastoplastic materials the stresses, σ_{ij}, depends only on the state of strain; however, above a certain stress level, called the yield stress, $\sigma_y(a_i)$,, nonrecoverable plastic deformations are obtained. The yield stress changes with increasing plastic deformations, which are measured by internal variables, a_i.

In the uniaxial tension test, a curve like that in Figure 15.1 is generated where logrithmic uniaxial strain is plotted against the uniaxial true stress which is defined as the applied load P divided by the actual cross-sectional area, A.

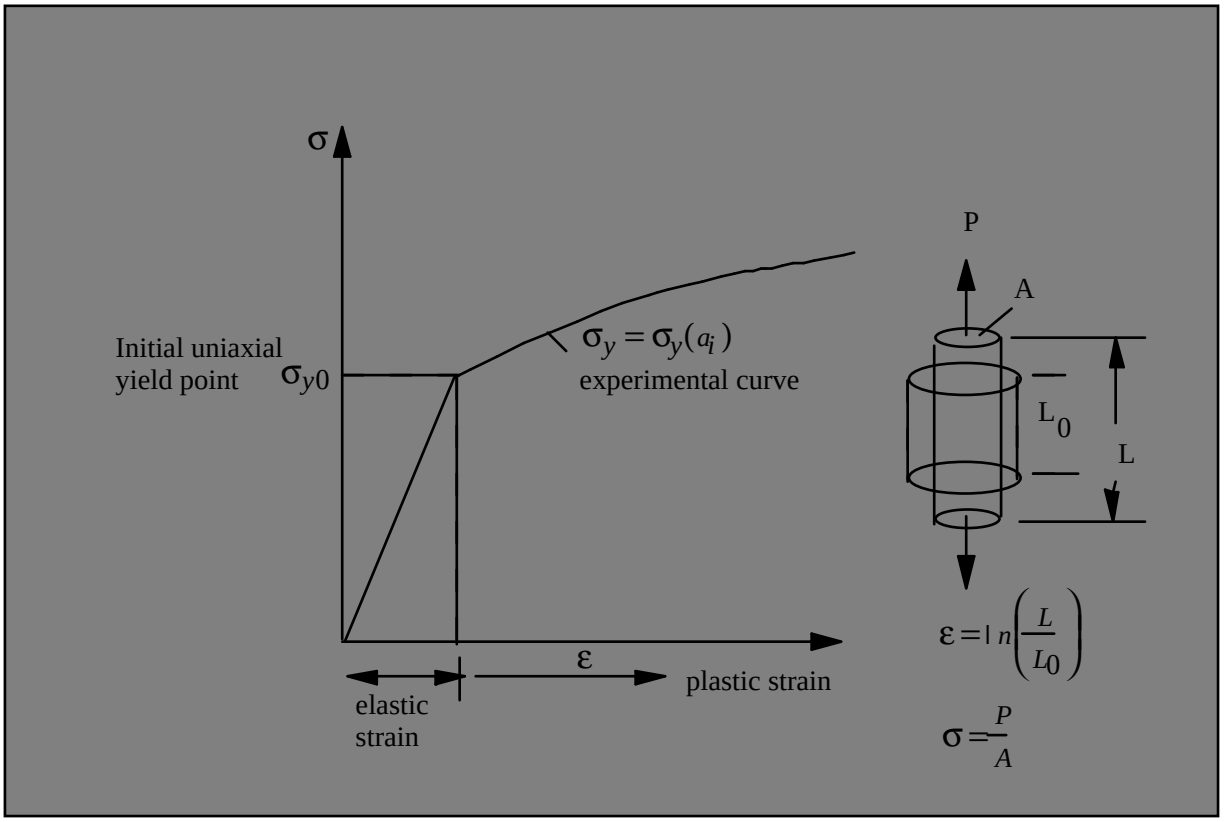


Figure 15.1. The uniaxial tension test demonstrates plastic behavior.

For the simple von Mises plasticity models the yield stress is pressure independent and the yield surface is a cylinder in principal stress space as shown in Figure 15.2. With isotropic hardening the diameter of the cylinder grows but the shape remains circular. In kinematic hardening the diameter may remain constant but will translate in the plane as a function of the plastic strain tensor, See Figure 15.3.

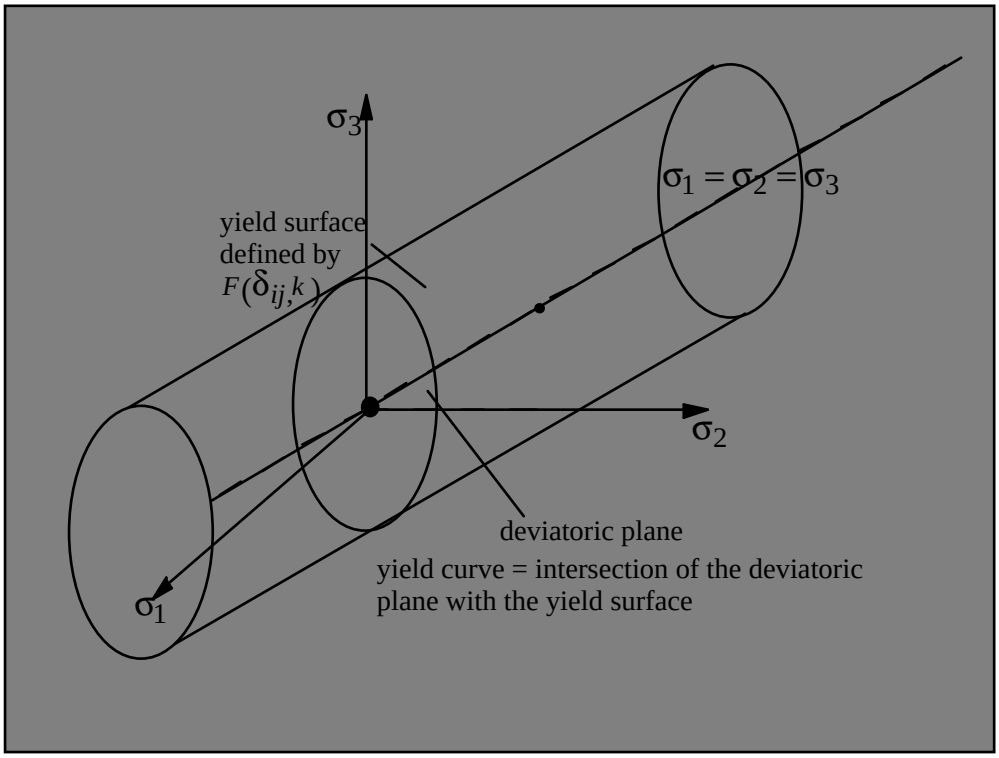


Figure 15.2. The yield surface in principal stress space in pressure independent.

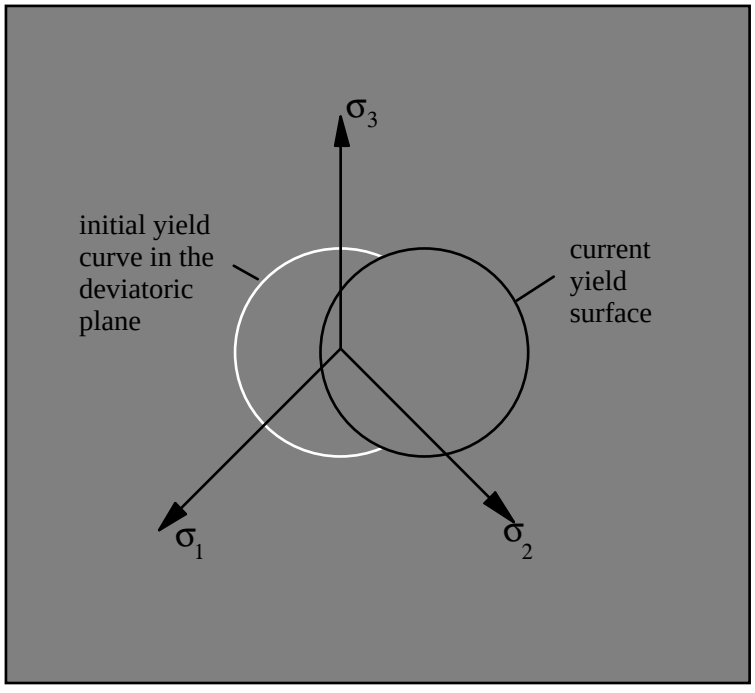


Figure 15.3. With kinematic hardening the yield surface may shift as a function of plastic strain.

The equation describing the pressure independent yield surface, F, is a function of the deviatoric stress tensor, s_{ij}, of a form given in Equation (15.30).

$$F\left(s_{ij}, a_i\right) = f\left(s_{ij}\right) - \sigma_y\left(a_i\right) = 0 \tag{15.30}$$

$f(s_{ij})$ = determines the shape,
$\sigma_y(a_i)$ = determines the translation and size.

The existence of a potential function, g, called the plastic potential, is assumed

$$g = g(s_{ij}) \tag{15.31}$$

Stability and uniqueness require that:

$$d\varepsilon_{ij}^{p} = \lambda \frac{\partial g}{\partial s_{ij}} \tag{15.32}$$

where λ is a proportionality constant.

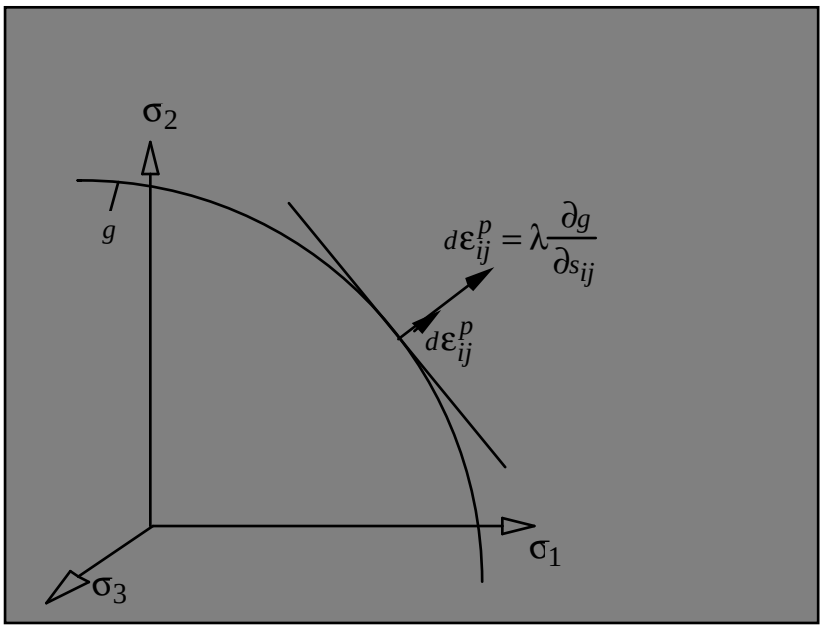


Figure 15.4. The plastic strain is normal to the yield surface.

As depicted in Figure 15.4 the plastic strain increments $d\varepsilon_{ij}^{p}$ are normal to the plastic potential function. This is the normality rule of plasticity.

The plastic potential g is identical with the yield condition F(s_{ij})

$$g \equiv F \tag{15.33}$$

Hence:

$$d\varepsilon_{ij}^{p} = \lambda \frac{\partial g}{\partial s_{ij}} \quad \rightarrow \quad d\varepsilon_{ij}^{p} = \lambda \cdot \frac{\partial f}{\partial s_{ij}} = \lambda \, \text{grad} \; f \tag{15.34}$$

and the stress increments ds_{ij} are normal to the plastic flow $\frac{\partial g}{\partial s_{ij}}$.

Post-yielding behavior from uniaxial tension tests typically show the following behaviors illustrated in Figure 15.5:

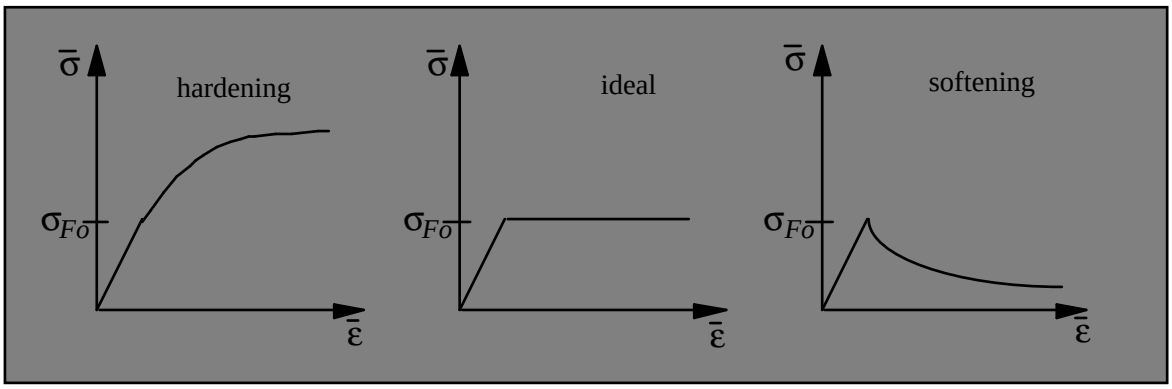


Figure 15.5. Hardening, ideal, and softening plasticity models.

The behavior of these hardening laws are characterized in Table 1. below. Although LS-DYNA3D permits softening to be defined and used, such softening behavior will result in strain localization and nonconvergence with mesh refinement.

15.5 Hyperelastic Materials

Stresses for elastic and hyperelastic materials are path independent; consequently, the stress update is not computed incrementally. The methods described here are well known and the reader is referred to Green and Adkins [1970] and Ogden [1984] for more details.

	HARDENING	IDEAL	SOFTENING
BEHAVIOR	$\sigma_y(a_i)$ is monotonic increasing	$\sigma_y(a_i)$ is constant	$\sigma_y(a_i)$ is monotonic decreasing
STABILITY	yes	yes	no
UNIQUENESS	yes	yes	no
APPLICATIONS	metals, concrete, rock with small deformations	crude idealization for steel, plastics, etc.	dense sand, concrete with large deformations

Table 15.1. Plastic hardening, ideal plasticity, and softening.

A retangular cartesian coordinate system is used so that the covarient and contravariant metric tensors in the reference (undeformed) and deformed configuration are

$$\begin{aligned} g_{ij} &= g^{ij} = \delta_{ij} \\ G_{ij} &= \frac{\partial x_k}{\partial X_i}\frac{\partial x_k}{\partial X_j} \\ G^{ij} &= \frac{\partial X_i}{\partial x_k}\frac{\partial X_j}{\partial x_k} \end{aligned} \tag{15.35}$$

The Green-St. Venant strain tensor and the principal strain invarients are defined as

$$\begin{aligned} \gamma_{ij} &= \tfrac{1}{2}\left(G_{ij} - \delta_{ij}\right) \\ I_1 &= \delta^{ij} G_{ij} \\ I_2 &= \tfrac{1}{2}\left(\delta^{ir}\delta^{js} G_{ri} G_{sj} - \delta^{ir}\delta^{js} G_{ij} G_{rs}\right) \\ I_3 &= \det G_{ij} \end{aligned} \tag{15.36}$$

For a compressible elastic material the existance of a strain energy functional, W, is assumed

$$W = W(I_1, I_2, I_3) \tag{15.37}$$

which defines the energy per unit undeformed volume. The stress measured in the deformed configuration is given as [Green and Adkins, 1970]:

$$s^{ij} = \Phi g^{ij} + \Psi B^{ij} + pG^{ij} \tag{15.38}$$

where

$$\Phi = \frac{2}{\sqrt{I_3}}\frac{\partial W}{\partial I_1}$$

$$\Psi = \frac{2}{\sqrt{I_3}}\frac{\partial W}{\partial I_2} \tag{15.39a}$$

$$p = 2\sqrt{I_3}\frac{\partial W}{\partial I_3}$$

$$B^{ij} = I_1\delta^{ij} - \delta^{ir}\delta^{js}G_{rs} \tag{15.39b}$$

This stress is related to the second Piola-Kirchhoff stress tensor:

$$S^{ij} = s^{ij}\sqrt{I_3} \tag{15.40}$$

Second Piola-Kirchhoff stresses are transformed to physical (Cauchy) stresses according to the relationship

$$\sigma_{ij} = \frac{\rho}{\rho_0}\frac{\partial x_i}{\partial X_k}\frac{\partial x_j}{\partial X_l}S_{kl} \tag{15.41}$$

15.6 Layered Composites

The composite models for shell elements in LS-DYNA3D include models for elastic behavior and inelastic behavior. The approach used here for updating the stresses also applies to the airbag fabric model.

To allow for an arbitrary orientation of the shell elements within the finite element mesh, each ply in the composite has a unique orientation angle, β, which measures the offset from some reference in the element. Each integration point through the shell

thickness, typically though not limited to one point per ply, requires the definition of β at that point. The reference is determined by the angle ψ, which can be defined for each element on the element card, and is measured from the 1-2 element side. Figures 15.6 and 15.7 depict these angles.

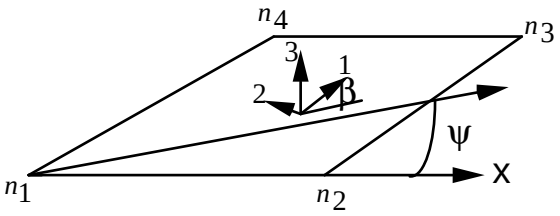


Figure 15.6. Orientation of material directions relative to the 1-2 side.

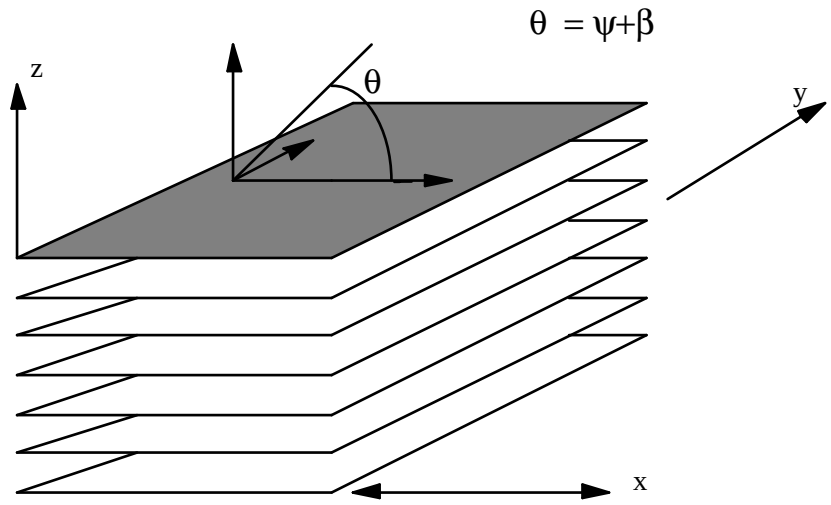


Figure 15.7. A multi-layer laminate can be defined. The angle β_i is defined for the ith lamina.

We update the stresses in the shell in the local shell coordinate system which is defined by the 1-2 element side and the cross product of the diagonals. Thus to transform the stress tensor into local system determined by the fiber directions entails a transformation that takes place in the plane of the shell.

In the implementation of the material model we first transform the Cauchy stress and velocity strain tensor d_{ij} into the coordinate system of the material denoted by the subscript L

$$\vec{\sigma}_L = q^t \sigma q$$

$$q_L = q^t d q \tag{15.42}$$

$$\sigma_L = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{32} & \sigma_{32} & \sigma_{33} \end{bmatrix} \qquad \varepsilon_L = \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{32} & d_{32} & d_{33} \end{bmatrix}$$

The Arabic subscripts on the stress and strain (σ and ε) are used to indicate the principal material directions where 1 indicates the fiber direction and 2 indicates the transverse fiber direction (in the plane). The orthogonal 3×3 transformation matrix is given by

$$q = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{15.43}$$

In shell theory we assume a plane stress condition, i.e., that the normal stress, σ_{33}, to the midsurface is zero. We can now incrementally update the stress state in the material coordinates

$$\sigma_L^{n+1} = \sigma_L^n + \Delta\sigma_L^{n+\frac{1}{2}} \tag{15.44}$$

where for an elastic material

$$\Delta\sigma_L^{n+\frac{1}{2}} = \begin{bmatrix} \Delta\sigma_{11} \\ \Delta\sigma_{22} \\ \Delta\sigma_{12} \\ \Delta\sigma_{23} \\ \Delta\sigma_{31} \end{bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} & 0 & 0 & 0 \\ Q_{12} & Q_{22} & 0 & 0 & 0 \\ 0 & 0 & Q_{44} & 0 & 0 \\ 0 & 0 & 0 & Q_{55} & 0 \\ 0 & 0 & 0 & 0 & Q_{66} \end{bmatrix} \begin{bmatrix} d_{11} \\ d_{22} \\ d_{12} \\ d_{23} \\ d_{31} \end{bmatrix}_L \Delta t \tag{15.45}$$

The terms Q_{ij} are referred to as reduced components of the lamina and are defined as

$$Q_{11} = \frac{E_{11}}{1 - \nu_{12}\nu_{21}}$$

$$Q_{22} = \frac{E_{22}}{1 - \nu_{12}\nu_{21}}$$

$$Q_{12} = \frac{\nu_{12}E_{11}}{1 - \nu_{12}\nu_{21}}$$

$$Q_{44} = G_{12}$$

$$Q_{55} = G_{23} \tag{15.46}$$

$$Q_{66} = G_{31}$$

Because of the symmetry properties,

$$\nu_{ji} = \nu_{ij} \frac{E_{jj}}{E_{ii}} \tag{15.47}$$

where υ_{ij} is Poisson's ratio for the transverse strain in jth direction for the material undergoing stress in the ith-direction, E_{ij} are the Young's modulii in the ith direction, and G_{ij} are the shear modulii.

After completion of the stress update we transform the stresses back into the local shell coordinate system.

$$\sigma = q\sigma_L q^t \tag{15.48}$$

15.7 Constraints on Orthotropic Elastic Constants

The inverse of the constitutive matrix C_l is generally defined in terms of the local material axes for an orthotropic material is given in terms of the nine independent elastic contants as

$$C_l^{-1} = \begin{bmatrix} \frac{1}{E_{11}} & -\frac{\upsilon_{21}}{E_{22}} & -\frac{\upsilon_{31}}{E_{33}} & 0 & 0 & 0 \\ -\frac{\upsilon_{12}}{E_{11}} & \frac{1}{E_{22}} & -\frac{\upsilon_{32}}{E_{33}} & 0 & 0 & 0 \\ -\frac{\upsilon_{13}}{E_{11}} & -\frac{\upsilon_{23}}{E_{22}} & \frac{1}{E_{33}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_{12}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{23}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{31}} \end{bmatrix} \tag{15.49}$$

As discussed by Jones [1975], the constants for a thermodynamically stable material must satisfy the following inequalities:

$$E_1, E_2, E_3, G_{12}, G_{23}, G_{31} > 0 \tag{15.50a}$$

$$C_{11}, C_{22}, C_{33}, C_{44}, C_{55}, C_{66} > 0 \tag{15.50b}$$

$$(1-\nu_{23}\nu_{32}), (1-\nu_{13}\nu_{31}), (1-\nu_{12}\nu_{21}) > 0 \tag{15.50c}$$

$$1-\nu_{12}\nu_{21}-\nu_{23}\nu_{32}-\nu_{31}\nu_{13}-2\nu_{21}\nu_{32}\nu_{13} > 0 \tag{15.50d}$$

Using Equation (15.47) and (15.50b) leads to:

$$\begin{aligned} |\nu_{21}| &< \left(\frac{E_{22}}{E_{11}}\right)^{1/2} \quad |\nu_{12}| < \left(\frac{E_{11}}{E_{22}}\right)^{1/2} \\ |\nu_{32}| &< \left(\frac{E_{33}}{E_{22}}\right)^{1/2} \quad |\nu_{23}| < \left(\frac{E_{22}}{E_{33}}\right)^{1/2} \\ |\nu_{13}| &< \left(\frac{E_{11}}{E_{33}}\right)^{1/2} \quad |\nu_{31}| < \left(\frac{E_{33}}{E_{11}}\right)^{1/2} \end{aligned} \tag{15.51}$$

16. MATERIAL MODELS

DYNA3D accepts a wide range of material and equation of state models, each with a unique number of history variables. Presently 37 material models are implemented, and space has been allotted for up to 10 user-specified models.

Material models 1 through 11 were adopted from DYNA2D to allow a common data base for two- and three-dimensional analyses. These models include the following material responses:

1. elastic,
2. orthotropic elastic [Cook 1974],
3. kinematic/isotropic plasticity [Krieg and Key 1976],
4. thermoelastoplastic [Hallquist 1979],
5. soil and crushable/non-crushable foam [Krieg 1972],
6. linear viscoelastic [Herrmann and Peterson 1968],
7. Blatz-Ko rubber [Blatz and Ko 1962],
8. high explosive burn [Giroux 1973],
9. hydrodynamic without deviatoric stresses,
10. elastoplastic hydrodynamic,
11. temperature dependent elastoplastic [Steinberg and Guinan 1978],
12. isotropic elastoplastic,
13. isotropic elastoplastic with failure,
14. soil and crushable foam with failure,
15. Johnson-Cook plasticity model,
16. pseudo TENSOR geological model,
17. elastoplastic with fracture,
18. power law isotropic plasticity,
19. strain rate dependent plasticity,
20. rigid,
21. thermal orthotropic,
22. composite damage model,
23. thermal orthotropic with 12 curves,
24. piecewise linear isotropic plasticity,
25. inviscid, two invariant geologic cap,
26. orthotropic crushable model,
27. Mooney-Rivlin rubber,
28. resultant plasticity,

29. force limited resultant formulation,
30. closed form update shell plasticity,
31. Frazer-Nash rubber model,
32. laminated glass model,
33. fabric,
37. anisotropic plasticity,
38. Blatz-Ko compressible foam
41-50. user-defined,
51. temperature- and rate-dependent plasticity,
52. Sandia damage model,
53. Low density polyurethane foam,
57. Urethane foam,
60. elastic with viscosity,
61. Maxwell/Kelvin viscoelastic with maximum strain,
62. viscous foam,
63. isotropic crushable foam.

Of these, materials 8, 9, 10, 11, and 15 require an equation of state designation. Materials 8 and 9 really are not materials at all. In the high explosive burn subroutine, burn fractions are computed based on lighting times determined in the initialization overlay. Material 9 allows a simple way of using an equation of state without material strength although materials 10 and 11 could accomplish the same thing with proper input.

We will now briefly discuss these models. Note that many of them were formulated by others and programmed by the author with vectorization for implementation in DYNA3D.

Material Model 1: Elastic

In this elastic material we compute the co-rotational rate of the deviatoric Cauchy stress tensor as

$$\overset{\nabla}{s}_{ij}^{n+1/2} = 2G\dot{\varepsilon}'^{\,n+1/2}_{ij} \tag{16.1.1}$$

and pressure

$$p^{n+1} = -K \ln V^{n+1} \tag{16.1.2}$$

where G and K are the elastic shear and bulk moduli, respectively, and V is the relative volume, i.e., the ratio of the current volume to the initial volume.

Material Model 2: Orthotropic Elastic

The material law that relates second Piola-Kirchhoff stress S to the Green-St. Venant strain E is

$$S = C\,E = T^t\,C_l\,T\,E \tag{16.2.1}$$

where T is the transformation matrix [Cook 1974].

$$T = \begin{bmatrix} l_1^2 & m_1^2 & n_1^2 & l_1m_1 & m_1n_1 & n_1l_1 \\ l_2^2 & m_2^2 & n_2^2 & l_2m_2 & m_2n_2 & n_2l_2 \\ l_3^2 & m_3^2 & n_3^2 & l_3m_3 & m_3n_3 & n_3l_3 \\ 2l_1l_2 & 2m_1m_2 & 2n_1n_2 & (l_1m_2+l_1m_1) & (m_1n_2+m_2n_1) & (n_1l_2+n_2l_1) \\ 2l_2l_3 & 2m_2m_3 & 2n_2n_3 & (l_2m_3+l_3m_2) & (m_2n_3+m_3n_2) & (n_2l_3+n_3l_2) \\ 2l_3l_1 & 2m_3m_1 & 2n_3n_1 & (l_3m_1+l_1m_3) & (m_3n_1+m_1n_3) & (n_3l_1+n_1l_3) \end{bmatrix} \tag{16.2.2}$$

l_i, m_i, n_i are the direction cosines

$$x'_i = l_ix_1 + m_ix_2 + n_ix_3 \quad i = 1,2,3 \tag{16.2.3}$$

and x'_i denotes the material axes. The constitutive matrix C_l is defined in terms of the material axes as

$$C_l^{-1} = \begin{bmatrix} \frac{1}{E_{11}} & -\frac{\upsilon_{21}}{E_{22}} & -\frac{\upsilon_{31}}{E_{33}} & 0 & 0 & 0 \\ -\frac{\upsilon_{12}}{E_{11}} & \frac{1}{E_{22}} & -\frac{\upsilon_{32}}{E_{33}} & 0 & 0 & 0 \\ -\frac{\upsilon_{13}}{E_{11}} & -\frac{\upsilon_{23}}{E_{22}} & \frac{1}{E_{33}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_{12}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{23}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{31}} \end{bmatrix} \tag{16.2.4}$$

where the subscripts denote the material axes, i.e.,

$$\upsilon_{ij} = \upsilon_{x'_i x'_j} \quad \text{and} \quad E_{ii} = E_{x'_i} \tag{16.2.5}$$

Since $\underset{\sim}{C}_l$ is symmetric

$$\frac{\upsilon_{12}}{E_{11}} = \frac{\upsilon_{21}}{E_{22}}, \text{ etc.} \tag{16.2.6}$$

The vector of Green-St. Venant strain components is

$$\underset{\sim}{E}^t = [E_{11}, E_{22}, E_{33}, E_{12}, E_{23}, E_{31}] \tag{16.2.7}$$

After computing S_{ij} we use Equation (15.32) to obtain the Cauchy stress. This model will predict realistic behavior for finite displacement and rotations as long as the strains are small.

This model is expensive to use and needs 18 history variables, 12 components of the strain displacement matrix computed in the initial configuration and 6 direction cosines.

Material Model 3: Elastic Plastic with Kinematic Hardening

Isotropic, kinematic, or a combination of isotropic and kinematic hardening may be obtained by varying a parameter, called β between 0 and 1. For β equal to 0 and 1, respectively, kinematic and isotropic hardening are obtained as shown in Figure 16.1. Krieg and Key [1976] formulated this model and the implementation is based on their paper.

In isotropic hardening, the center of the yield surface is fixed but the radius is a function of the plastic strain. In kinematic hardening, the radius of the yield surface is fixed but the center translates in the direction of the plastic strain. Thus the yield condition is

$$\phi = \frac{1}{2}\xi_{ij}\,\xi_{ij} - \frac{\sigma_y^{\,2}}{3} = 0 \qquad (16.3.1)$$

where

$$\xi_{ij} = s_{ij} - \alpha_{ij} \qquad (16.3.2)$$

$$\sigma_y = \sigma_0 + \beta E_p\,\varepsilon_{eff}^{p} \qquad (16.3.3)$$

The co-rotational rate of α_{ij} is

$$\alpha_{ij}^{\nabla} = (1-\beta)\frac{2}{3}E_p\,\dot{\varepsilon}_{ij}^{p} \quad . \qquad (16.3.4)$$

Hence,

$$\alpha_{ij}^{n+1} = \alpha_{ij}^{n} + \left(\alpha_{ij}^{\nabla^{n+1/2}} + \alpha_{ik}^{n}\,\Omega_{kj}^{n+1/2} + \alpha_{jk}^{n}\,\Omega_{ki}^{n+1/2}\right)\Delta t^{n+1/2} \quad . \qquad (16.3.5)$$

Strain rate is accounted for using the Cowper and Symonds [Jones 1983] model which scales the yield stress by a strain rate dependent factor

$$\sigma_y = \left[1 + \left(\frac{\dot{\varepsilon}}{C}\right)^{\frac{1}{p}}\right]\left(\sigma_0 + \beta E_p\,\varepsilon_{eff}^{p}\right) \qquad (16.3.6)$$

where p and C are user defined input constants and $\dot{\varepsilon}$ is the strain rate defined as:

$$\dot{\varepsilon} = \sqrt{\dot{\varepsilon}_{ij}\dot{\varepsilon}_{ij}} \qquad (16.3.7)$$

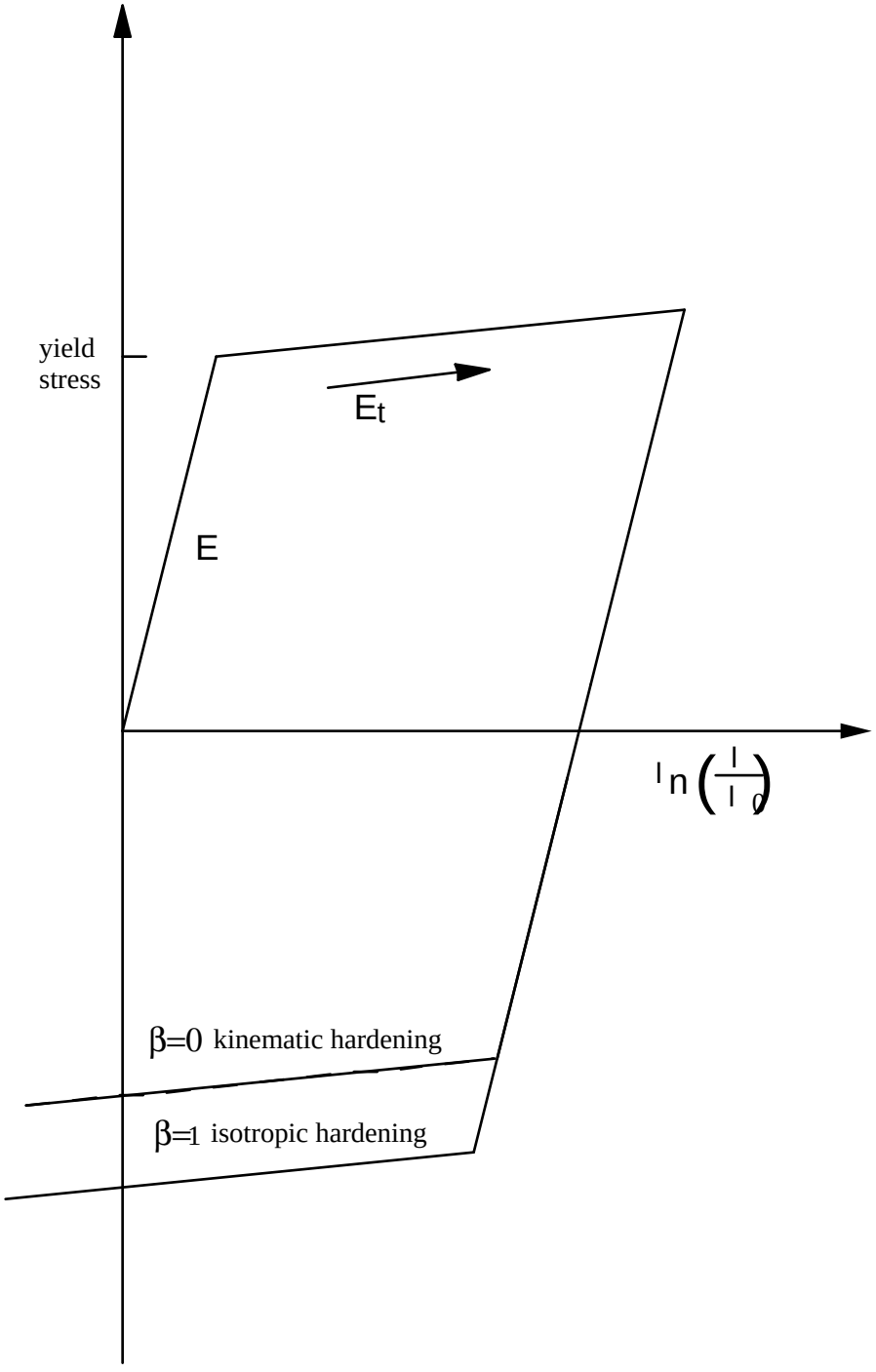


Figure 16.1 Elastic-plastic behavior with isotropic and kinematic hardening where ℓ_0 and ℓ are undeformed and deformed length of uniaxial tension specimen respectively.

The current radius of the yield surface σ_y is the sum of the initial yield strength σ_0, plus the growth $\beta E_p \varepsilon_{eff}^p$, where E_p is the plastic hardening modulus

$$E_p = \frac{E_t E}{E - E_t} \tag{16.3.8}$$

and ε_{eff}^p is the effective plastic strain

$$\varepsilon_{eff}^p = \int_0^t \left(\frac{2}{3} \dot{\varepsilon}_{ij}^p \dot{\varepsilon}_{ij}^p \right)^{1/2} dt \tag{16.3.9}$$

The plastic strain rate is the difference between the total and elastic (right superscript e) strain rates:

$$\dot{\varepsilon}_{ij}^p = \dot{\varepsilon}_{ij} - \dot{\varepsilon}_{ij}^e \tag{16.3.10}$$

In the implementation of this material model, the deviatoric stresses are updated elastically, as described for model 1 but repeated here for the sake of clarity:

$$\sigma_{ij}^* = \sigma_{ij}^n + C_{ijkl} \Delta\varepsilon_{kl} \tag{16.3.11}$$

where

σ_{ij}^* is the trial stress tensor,

σ_{ij}^n is the stress tensor from the previous time step,

C_{ijkl} is the elastic tangent modulus matrix,

$\Delta\varepsilon_{ij}$ is the incremental strain tensor.

and, if the yield function is satisfied, nothing else is done. If, however, the yield function is violated, an increment in plastic strain is computed, the stresses are scaled back to the yield surface, and the yield surface center is updated.

Let s_{ij}^* represent the trial elastic deviatoric stress state at n+1

$$s_{ij}^* = \sigma_{ij}^* - \frac{1}{3}\sigma_{kk}^* \tag{16.3.12}$$

and

$$\xi_{ij}^{*} = s_{ij}^{*} - \alpha_{ij} \quad . \tag{16.3.13}$$

Define the yield function,

$$\phi = \frac{3}{2}\xi_{ij}^{*}\,\xi_{ij}^{*} - \sigma_y^2 = \Lambda^2 - \sigma_y^2 \begin{cases} \le 0 \text{ for elastic or neutral loading} \\ > 0 \text{ for plastic harding} \end{cases} \tag{16.3.14}$$

For plastic hardening then

$$\varepsilon_{eff}^{p\,n+1} = \varepsilon_{eff}^{p\,n} + \frac{\Lambda - \sigma_y}{3G + E_p} = \varepsilon_{eff}^{p\,n} + \Delta\varepsilon_{eff}^{p} \tag{16.3.15}$$

scale back the stress deviators

$$\sigma_{ij}^{n+1} = \sigma_{ij}^{*} - \frac{3G\Delta\varepsilon_{eff}^{p}}{\Lambda}\xi_{ij}^{*} \tag{16.3.16}$$

and update the center:

$$\alpha_{ij}^{n+1} = \alpha_{ij}^{n} + \frac{(1-\beta)E_p\,\Delta\varepsilon_{eff}^{p}}{\Lambda}\,\xi_{ij}^{*} \tag{16.3.17}$$

16.3.1 Plane Stress Plasticity

The plane stress plasticity options apply to beams, shells and thick shells. Since the stresses and strain increments are transformed to the lamina coordinate system for the constitutive evaluation, the stress and strain tensors n are in the local coordinate system.

The application of the Jaumann rate to update the stress tensor allows for the possibility that the normal stress, σ_{33}, will not be zero. The first step in updating the stress tensor is to compute a trial plane stress update assuming that the incremental strains are elastic. In the above, the normal strain increment $\Delta\varepsilon_{33}$ is replaced by the elastic strain increment

$$\Delta\varepsilon_{33} = -\frac{\sigma_{33} + \lambda\left(\Delta\varepsilon_{11} + \Delta\varepsilon_{22}\right)}{\lambda + 2\mu} \tag{16.3.18}$$

where λ and μ are Lamé's constants.

When the trial stress is within the yield surface, the strain increment is elastic and the stress update is completed. Otherwise for the plastic plane stress case, secant iteration is used to solve Equation (16.3.16) for the normal strain increment ($\Delta\varepsilon_{33}$) required to produce a zero normal stress:

$$\sigma_{33}^{i} = \sigma_{33}^{*} - \frac{3G\Delta\varepsilon_{eff}^{p\,i}\xi_{33}}{\Lambda} \tag{16.3.19}$$

Here the superscript i indicates the iteration number.

The secant iteration formula for $\Delta\varepsilon_{33}$ (the superscript p is dropped for clarity) is

$$\Delta\varepsilon_{33}^{i+1} = \Delta\varepsilon_{33}^{i-1} - \frac{\Delta\varepsilon_{33}^{i} - \Delta\varepsilon_{33}^{i-1}}{\sigma_{33}^{i} - \sigma_{33}^{i-1}}\sigma_{33}^{i-1} \tag{16.3.20}$$

where the two starting values are obtained from the initial elastic estimate and by assuming a purely plastic increment, i.e.

$$\Delta\varepsilon_{33} = -(\Delta\varepsilon_{11} + \Delta\varepsilon_{22}) \tag{16.3.21}$$

These starting values bound the actual values of the normal strain increment.

The iteration procedure uses the updated normal stain increment to update first the deviatoric stress and then the other quantities needed to compute the next estimate of the normal stress in Equation (16.3.19). The iterations proceed until the normal stress σ_{33}^{i} is sufficiently small. The convergence criterion requires convergence of the normal strains:

$$\frac{\left|\Delta\varepsilon_{33}^{i} - \Delta\varepsilon_{33}^{i-1}\right|}{\left|\Delta\varepsilon_{33}^{i+1}\right|} < 10^{-4} \tag{16.3.22}$$

After convergence, the stress update is completed using the relationships given in Equations (16.3.16) and (16.3.17)

Material Model 4: Thermo-Elastic-Plastic

This model was adapted from the NIKE2D [Hallquist 1979] code. A more complete description of its formulation is given in the NIKE2D user's manual.

Letting T represent the temperature, we compute the elastic co-rotational stress rate as

$$\sigma_{ij}^{\nabla} = C_{ijkl}\left(\dot{\varepsilon}_{kl} - \dot{\varepsilon}_{kl}^{T}\right) + \dot{\theta}_{ij}\, dT \tag{16.4.1}$$

where

$$\dot{\theta}_{ij} = \frac{dC_{ijkl}}{dT} C_{klmn}^{-1}\, \dot{\sigma}_{mn} \tag{16.4.2}$$

and $C_{ijk\ell}$ is the temperature dependent elastic constitutive matrix:

$$C_{ijk\ell} = \frac{E}{(1+\upsilon)(1-2\upsilon)} \begin{bmatrix} 1-\upsilon & \upsilon & \upsilon & 0 & 0 & 0 \\ \upsilon & 1-\upsilon & \upsilon & 0 & 0 & 0 \\ \upsilon & \upsilon & 1-\upsilon & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\upsilon}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\upsilon}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\upsilon}{2} \end{bmatrix} \tag{16.4.3}$$

where υ is Poisson's ratio. The thermal strain rate can be written in terms of the coefficient of thermal expansion α as

$$\dot{\varepsilon}_{ij}^{T} = \left[\frac{d\alpha}{dT}\left(T - T_{ref}\right) + \alpha\right] \dot{T}\, \delta_{ij} \tag{16.4.4}$$

Note that α is defined with respect to a reference temperature T_{ref}. Since this has been a source of confusion for many users, an alternative definition of α has been implemented as an option:

$$\dot{\varepsilon}_{ij}^{T} = \alpha\, \dot{T}\, \delta_{ij} \tag{16.4.5}$$

When treating plasticity, we use a procedure analogous to that for material 3. We update the stresses elastically and check to see if we violate the isotropic yield function

$$\phi = \frac{1}{2} s_{ij}\ s_{ij} - \frac{\sigma_y(T)^2}{3} \tag{16.4.6}$$

where

$$\sigma_y(T) = \sigma_o(T) + E_p(T)\,\varepsilon_{eff}^p \tag{16.4.7}$$

The initial yield, σ_0, and plastic hardening modulus, E_p, are temperature dependent. If the behavior is elastic we do nothing; otherwise, we scale back the stress deviators by the factor f_s:

$$s_{ij}^{n+1} = f_s\, s_{ij}^* \tag{16.4.8}$$

where

$$f_s = \frac{\sigma_y}{\left(\frac{3}{2} s_{ij}^* s_{ij}^*\right)^{1/2}} \tag{16.4.9}$$

and update the plastic strain by the increment

(16.4.10)

Material Model 5: Soil and Crushable Foam

This model, due to Krieg [1972], provides a simple model for foam and soils whose material properties are not well characterized. We believe the other foam models in LS-DYNA3D are superior in their performance and are recommended over this model which simulates the crushing through the volumetric deformations. With low values for the yield stress nearly fluid like behavior is obtained.

A pressure-dependent flow rule governs the deviatoric behavior:

$$\phi_s = \frac{1}{2} s_{ij} s_{ij} - (a_0 + a_1 p\ a_2 p^2) \tag{16.5.1}$$

where a_0, a_1, and a_2 are user-defined constants. Volumetric yielding is determined by a tabulated curve of pressure versus volumetric strain. Elastic unloading from this curve is assumed to a tensile cutoff as illustrated in Figure 16.2.

Implementation of this model is straightforward. One history variable, the maximum volumetric strain in compression, is needed. If we are loading, the new compressive volumetric strain exceeds the stored value. If failure has occurred, pressure is left at the cutoff value and the deviatoric stresses are zeroed. If the yield condition is violated, the updated trial stresses, s_{ij}^*, are scaled back:

$$s_{ij}^{n+1} = \left(\frac{a_0 + a_1 p + a_2 p^2}{\frac{1}{2} s_{ij}\ s_{ij}} \right)^{1/2} s_{ij}^* \tag{16.5.2}$$

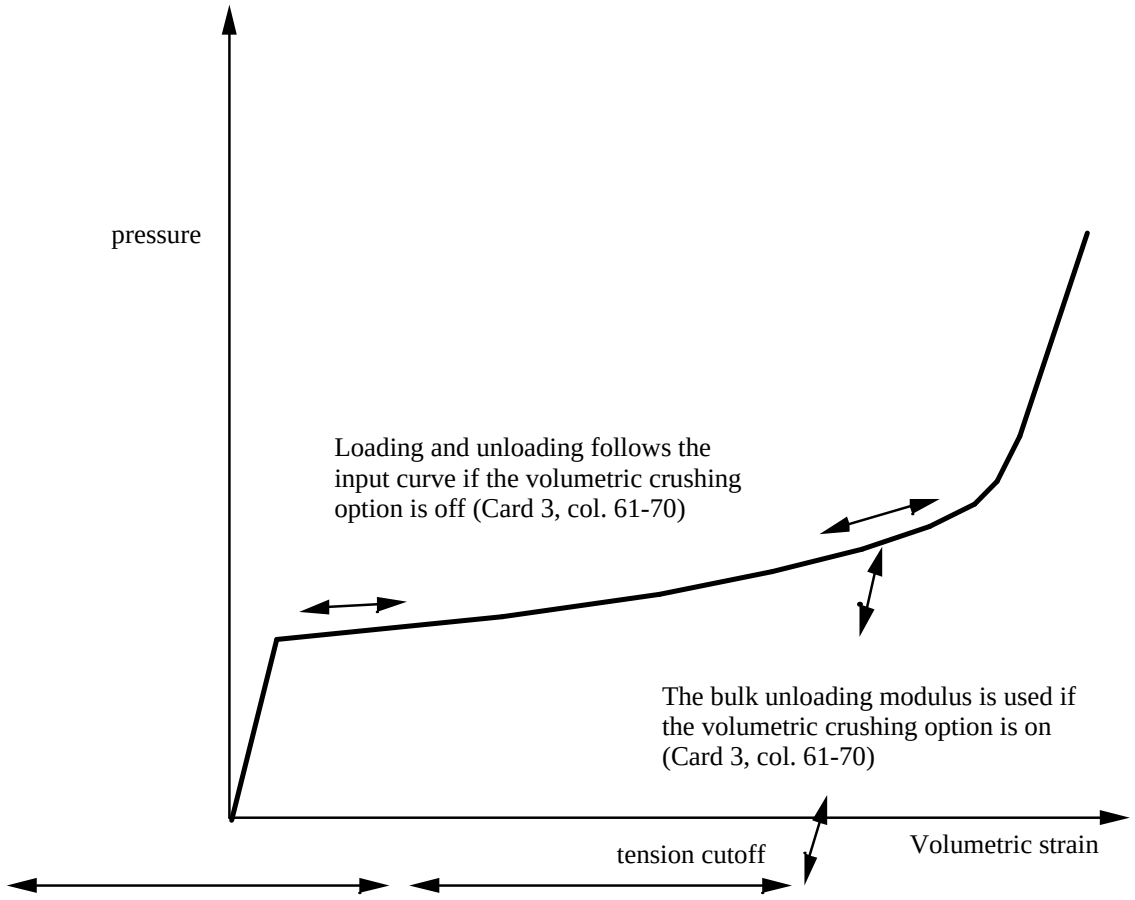


Figure 16.2. Volumetric strain versus pressure curve for soil and crushable foam model.

Material Model 6: Viscoelastic

In this model, linear viscoelastic behavior is assumed for the deviatoric behavior [Herrmann and Peterson 1968]:

$$s_{ij} = 2\int_0^t \phi(t-\tau)\frac{\partial \varepsilon'_{ij}(\tau)}{\partial \tau}\,d\tau \tag{16.6.1}$$

where

$$\phi(t) = G + (G_0 - G)e^{-\beta t} \tag{16.6.2}$$

is the shear relaxation modulus. A recursion formula is used to compute the new value of the hereditary integral at time t^{n+1} from its value at time t^n. Elastic bulk behavior is assumed:

$$p = K \ln V \tag{16.6.3}$$

where pressure is integrated incrementally.

Material Model 7: Continuum Rubber

The hyperelastic continuum rubber model was studied by Blatz and Ko [1962]. In this model, the second Piola-Kirchhoff stress is given by

$$S_{ij} = G\left(V^{-1} C_{ij} - V^{-\frac{1}{1-2\upsilon}} \delta_{ij} \right) \tag{16.7.1}$$

where G is the shear modulus, υ is Poisson's ratio, and C_{ij} is the right Cauchy-Green strain:

$$C_{ij} = \frac{\partial x_k}{\partial X_i} \frac{\partial x_k}{\partial X_j} \tag{16.7.2}$$

Material Model 8: Explosive Burn

Burn fractions, which multiply the equations of states for high explosives, control the release of chemical energy for simulating detonations. In the initialization phase, a lighting time t_l is computed for each element by dividing the distance from the detonation point to the center of the element by the detonation velocity D. If multiple detonation points are defined, the closest point determines t_l The burn fraction F is taken as the maximum

$$F = \max (F_1 , F_2) \tag{16.8.1}$$

where

$$F_1 = \begin{cases} \dfrac{2\,(t - t_l)D}{3\left(\dfrac{v_e}{A_{e_{\max}}}\right)} & \text{if } t > t_l \\ 0 & \text{if } t \le t_l \end{cases} \tag{16.8.2}$$

$$F_2 = \frac{1-V}{1-V_{CJ}} \tag{16.8.3}$$

where V_{CJ} is the Chapman-Jouguet relative volume and t is current time. If F exceeds 1, it is reset to 1. This calculation of the burn fraction usually requires several time steps for F to reach unity, thereby spreading the burn front over several elements. After reaching unity, F is held constant. This burn fraction calculation is based on work by Wilkins [1964] and is also discussed by Giroux [1973].

Material Model 9: Null Material

Equations of state can be called through this model to avoid deviatoric stress calculations. A pressure cutoff may be specified to set a lower bound on the pressure. This model has been very useful when combined with the reactive high explosive model where material strength is often neglected.

The null material should not be used to delete elements.

Material Model 10: Elastic-Plastic-Hydrodynamic

For completeness we give the entire derivation of this constitutive model based on radial return plasticity

The pressure, p, deviatoric strain rate, $\dot{\varepsilon}'_{ij}$, deviatoric stress rate, $\dot{s}_{ij}$, volumetric strain rate, and $\dot{\varepsilon}_v$, are defined in Equation (16.10.1):

$$p = -\frac{1}{3}\sigma_{ij}\,\delta_{ij} \qquad \dot{\varepsilon}'_{ij} = \dot{\varepsilon}_{ij} - \frac{1}{3}\dot{\varepsilon}_v$$

$$s_{ij} = \sigma_{ij} + p\delta_{ij} \qquad \dot{\varepsilon}_v = \dot{\varepsilon}_{ij}\,\delta_{ij} \tag{16.10.1}$$

$$s^{\nabla}_{ij} = 2\mu\,\dot{\varepsilon}'_{ij} = 2G\dot{\varepsilon}'_{ij}$$

The Jaumann rate of the deviatoric stress, s^{∇}_{ij}, is given by:

$$s^{\nabla}_{ij} = \dot{s}_{ij} - s_{ip}\Omega_{pj} - s_{jp}\Omega_{pi} \tag{16.10.2}$$

First we update s^n_{ij} to s^{n+1}_{ij} elastically

$$^{*}s^{n+1}_{ij} = s^n_{ij} + s_{ip}\Omega_{pj} + s_{jp}\Omega_{pi} + 2G\dot{\varepsilon}'_{ij}\,dt = \underbrace{s^n_{ij} + R_{ij}}_{s^{R^n}_{ij}} + \underbrace{2G\dot{\varepsilon}'_{ij}\,dt}_{2G\Delta\varepsilon'_{ij}} \tag{16.10.3}$$

where the left superscript, *, denotes a trial stress value. The effective trial stress is defined by

$$s^* = \left(\frac{3}{2}\,{}^{*}s^{n+1}_{ij}\,{}^{*}s^{n+1}_{ij}\right)^{1/2} \tag{16.10.4}$$

and if s^* exceeds yield stress σ_y, the Von Mises flow rule:

$$\phi = \frac{1}{2}s_{ij}\,s_{ij} - \frac{\sigma_y{}^2}{3} \le 0 \tag{16.10.5}$$

is violated and we scale the trial stresses back to the yield surface, i.e., a radial return

$$s^{n+1}_{ij} = \frac{\sigma_y}{s^*}\,{}^{*}s^{n+1}_{ij} = m\,{}^{*}s^{n+1}_{ij} \tag{16.10.6}$$

The plastic strain increment can be found by subtracting the deviatoric part of the strain increment that is elastic, $\frac{1}{2G}\left(s_{ij}^{n+1} - s_{ij}^{R^n}\right)$, from the total deviatoric increment, $\Delta\varepsilon'_{ij}$, i.e.,

$$\Delta\varepsilon_{ij}^{p} = \Delta\varepsilon'_{ij} - \frac{1}{2G}\left(s_{ij}^{n+1} - s_{ij}^{R^n}\right) \tag{16.10.7}$$

Recalling that,

$$\Delta\varepsilon'_{ij} = \frac{{}^{*}s_{ij}^{n+1} - s_{ij}^{R^n}}{2G} \tag{16.10.8}$$

and substituting Equation (16.10.8) into (16.10.7) we obtain,

$$\Delta\varepsilon_{ij}^{p} = \frac{\left({}^{*}s_{ij}^{n+1} - s_{ij}^{n+1}\right)}{2G} \tag{16.10.9}$$

Substituting Equation (16.10.6)

$$s_{ij}^{n+1} = m\,{}^{*}s_{ij}^{n+1}$$

into Equation (16.10.9) gives,

$$\Delta\varepsilon_{ij}^{p} = \left(\frac{1-m}{2G}\right){}^{*}s_{ij}^{n+1} = \frac{1-m}{2Gm}s_{ij}^{n+1} = d\lambda s_{ij}^{n+1} \tag{16.10.10}$$

By definition an increment in effective plastic strain is

$$\Delta\varepsilon^{p} = \left(\frac{2}{3}\Delta\varepsilon_{ij}^{p}\ \Delta\varepsilon_{ij}^{p}\right)^{1/2} \tag{16.10.11}$$

Squaring both sides of Equation (16.10.10) leads to:

$$\Delta\varepsilon_{ij}^{p}\ \Delta\varepsilon_{ij}^{p} = \left(\frac{1-m}{2G}\right)^{2}\ {}^{*}s_{ij}^{n+1}\ {}^{*}s_{ij}^{n+1} \tag{16.10.12}$$

or from Equations (16.10.4) and (16.10.11):

$$\frac{3}{2}\Delta\varepsilon^{p^2} = \left(\frac{1-m}{2G}\right)^2 \frac{2}{3} s^{*2} \tag{16.10.13}$$

Hence,

$$\therefore \Delta\varepsilon^p = \frac{1-m}{3G} s^* = \frac{s^* - \sigma_y}{3G} \tag{16.10.14}$$

where we have substituted for m from Equation (16.10.6)

$$m = \frac{\sigma_y}{s^*}$$

If isotropic hardening is assumed then:

$$\sigma_y^{n+1} = \sigma_y^n + E^p \Delta\varepsilon^p \tag{16.10.15}$$

and from Equation (16.10.14)

$$\Delta\varepsilon^p = \frac{\left(s^* - \sigma_y^{n+1}\right)}{3G} = \frac{\left(s^* - \sigma_y^n - E^p \Delta\varepsilon^p\right)}{3G} \tag{16.10.16}$$

Thus,

$$(3G + E^p)\,\Delta\varepsilon^p = (s^* - \sigma_y^n)$$

and solving for the incremental plastic strain gives

$$\Delta\varepsilon^p = \frac{\left(s^* - \sigma_y^n\right)}{\left(3G + E^p\right)} \tag{16.10.17}$$

The algorithm for plastic loading can now be outlined in five simple stress. If the effective trial stress exceeds the yield stress then

1. Solve for the plastic strain increment:

$$\Delta\varepsilon^p = \frac{\left(s^* - \sigma_y^n\right)}{\left(3G + E^p\right)}$$

2. Update the plastic strain:

$$\varepsilon^{p^{n+1}} = \varepsilon^{p^n} + \Delta\varepsilon^p \ .$$

3. Update the yield stress:

$$\sigma_y^{n+1} = \sigma_y^n + E^p \Delta\varepsilon^p$$

4. Compute the scale factor using the yield strength at time n+1:

$$m = \frac{\sigma_y^{n+1}}{s^*}$$

5. Radial return the deviatoric stresses to the yield surface:

$$s_{ij}^{n+1} = m^* s_{ij}^{n+1}$$

Material Model 11: Elastic-Plastic with Thermal Softening

Steinberg and Guinan [1978] developed this model for treating plasticity at high strain rates (10^5 s^{-1}) where enhancement of the yield strength due to strain rate effects is saturated out.

Both the shear modulus G and yield strength σ_y increase with pressure but decrease with temperature. As a melt temperature is reached, these quantities approach zero. We define the shear modulus before the material melts as

$$G = G_0\left[1 + bpV^{1/3} - h\left(\frac{E-E_c}{3R'} - 300\right)\right]e^{-\frac{fE}{E_m - E}} \tag{16.11.1}$$

where G_0, b, h, and f are input parameters, E_c is the cold compression energy:

$$E_c(X) = \int_0^X p\,dx - \frac{900\,R'\exp(ax)}{(1-X)^{2\left(\gamma_o - a - \frac{1}{2}\right)}}, \tag{16.11.2}$$

where

$$X = 1\text{-}V, \tag{16.11.3}$$

and E_m is the melting energy:

$$E_m(X) = E_c(X) + 3R'T_m(X) \tag{16.11.4}$$

which is a function of the melting temperature $T_m(X)$:

$$T_m(X) = \frac{T_{mo}\exp(2aX)}{(1-X)^{2\left(\gamma_o - a - 1/3\right)}} \tag{16.11.5}$$

and the melting temperature T_{m0} at $\rho = \rho_0$. The constants γ_0 and *a* are input parameters. In the above equation, R' is defined by

$$R' = \frac{R\rho_o}{A} \tag{16.11.6}$$

where R is the gas constant and A is the atomic weight. The yield strength σ_y is given by:

$$\sigma_y = \sigma'_0 \left[1 + b' p V^{1/3} - h\left(\frac{E - E_c}{3R'} - 300\right)\right] e^{-\frac{fE}{E_m - E}} \qquad (16.11.7)$$

If E_m exceeds E_i. Here, σ'_0 is given by:

$$\sigma'_0 = \sigma_0 \left[1 + \beta\left(\gamma_i + \bar{\varepsilon}^p\right)\right]^n \qquad (16.11.8)$$

where γ_i is the initial plastic strain, and b' and σ'_0 are input parameters. Where σ'_0 exceeds σ_{max}, the maximum permitted yield strength, σ'_0 is set to equal to σ_{max}. After the material melts, σ_y and G are set to zero.

DYNA3D fits the cold compression energy to a ten-term polynomial expansion:

$$E_c = \sum_{i=0}^{9} EC_i \, \eta^i \qquad (16.11.9)$$

where EC_i is the ith coefficient and $\eta = \frac{\rho}{\rho_o}$. The least squares method is used to perform the fit [Kreyszig 1972]. The ten coefficients may also be specified in the input.

Once the yield strength and shear modulus are known, the numerical treatment is similar to that for material model 10.

Material Model 12: Isotropic Elastic-Plastic

Computation of the deviatoric stress tensor is identical to that given for material model 10 except that the yield strength is given by

$$\sigma_y = \sigma_o + E_p \varepsilon_{eff}^p \tag{16.12.1}$$

Pressure is given by the expression

$$p^{n+1} = K\left(\frac{1}{V^{n+1}} - 1\right) \tag{16.12.2}$$

where K is the bulk modulus. This is perhaps the most cost effective plasticity model. Only one history variable, ε_{eff}^p , is stored with this model.

Material Model 13: Isotropic Elastic-Plastic with Failure

This highly simplistic failure model is occasionally useful. Material model 12 is called to update the stress tensor. Failure is initially assumed to occur if either

$$p^{n+1} < p_{min} \tag{16.13.1}$$

or

$$\varepsilon_{eff}^{p} > \varepsilon_{max}^{p} \tag{16.13.2}$$

where p_{min} and ε_{max}^{p} are user-defined parameters. Once failure has occurred, pressure may never be negative and the deviatoric components are set to zero:

$$s_{ij} = 0 \tag{16.13.3}$$

for all time. The failed element can only carry loads in compression.

Material Model 14: Soil and Crushable Foam with Failure

This material model provides the same stress update as model 5. However, if pressure ever reaches its cutoff value, failure occurs and pressure can never again go negative. In material model 5, the pressure is limited to its cutoff value in tension.

Material Model 15: Johnson and Cook Plasticity Model

Johnson and Cook express the flow stress as

$$\sigma_y = \left(A + B\,\bar{\varepsilon}^{\,p^n} \right)\left(1 + c\ \ln \dot{\varepsilon}^*\right)\left(1 - T*^m\right) \qquad (16.15.1)$$

where A, B, C, n, and m are user defined input constants, and:

$\bar{\varepsilon}^{\,p}$ = effective plastic strain

$\dot{\varepsilon}* = \dfrac{\dot{\bar{\varepsilon}}^{\,p}}{\dot{\varepsilon}_0}$ effective plastic strain rate for $\dot{\varepsilon}_0 = 1\,s^{-1}$

$T^* = \dfrac{T - T_{room}}{T_{melt} - T_{room}}$

Constants for a variety of materials are provided in Johnson and Cook [1983].

Due to the nonlinearity in the dependence of flow stress on plastic strain, an accurate value of the flow stress requires iteration for the increment in plastic strain. However, by using a Taylor series expansion with linearization about the current time, we can solve for σ_y with sufficient accuracy to avoid iteration.

The strain at fracture is given by

$$\varepsilon^{\,f} = \left[D_1 + D_2 \exp D_3\,\sigma^*\right]\left[1 + D_4\ \ln \varepsilon^*\right]\left[1 + D_5\,T^*\right] \qquad (16.15.2)$$

where D_i, i=1,…,5 are input constants and σ^* is the ratio of pressure divided by effective stress:

$$\sigma^* = \frac{p}{\sigma_{eff}} \qquad (16.15.3)$$

Fracture occurs when the damage parameter

$$D = \sum \frac{\Delta\bar{\varepsilon}^{\,p}}{\varepsilon^{\,f}} \qquad (16.15.4)$$

reaches the value 1.

Material Type 18: Power Law Isotropic Plasticity

Elastoplastic behavior with isotropic hardening is provided by this model. The yield stress σ_y is a function of plastic strain and obeys the equation:

$$\Delta\sigma_y = \beta k \left(\varepsilon_e + \varepsilon^p\right)^n \tag{16.18.1}$$

ε_e is the elastic strain to yield and $\bar{\varepsilon}^p$ is the effective plastic strain. Strain rate is accounted for by using the Cowper and Symonds model which scales the yield stress with the factor

$$\beta = 1 + \left(\frac{\dot{\varepsilon}}{C}\right)^{1/p} \tag{16.18.2}$$

where $\dot{\varepsilon}$ is the strain rate.

Material Type 19: Strain Rate Dependent Isotropic Plasticity

In this model, a load curve is used to describe the yield strength σ_0 as a function of effective strain rate $\dot{\bar{\varepsilon}}$ where

$$\dot{\bar{\varepsilon}} = \left(\frac{2}{3} \dot{\varepsilon}'_{ij} \, \dot{\varepsilon}'_{ij} \right)^{1/2}$$

and the prime denotes the deviatoric component. The yield stress is defined as

$$\sigma_y = \sigma_0 \left(\dot{\bar{\varepsilon}} \right) + E_p \, \bar{\varepsilon}^p$$

where $\bar{\varepsilon}^p$ is the effective plastic strain and E_h is given in terms of Young's modulus and the tangent modulus by

$$E_p = \frac{E \, E_t}{E - E_t} \ .$$

Material Model 22: Chang-Chang Composite Failure Model

Five material parameters are used in the three failure criteria. These are [Chang and Chang 1987a, 1987b]:

- S_1, longitudinal tensile strength
- S_2, transverse tensile strength
- S_{12}, shear strength
- C_2, transverse compressive strength
- α, nonlinear shear stress parameter.

S_1, S_2, S_{12}, and C_2 are obtained from material strength measurement. α is defined by material shear stress-strain measurements. In plane stress, the strain is given in terms of the stress as

$$\begin{aligned} \varepsilon_1 &= \frac{1}{E_1}\left(\sigma_1 - \upsilon_1\sigma_2\right) \\ \varepsilon_2 &= \frac{1}{E_2}\left(\sigma_2 - \upsilon_2\sigma_1\right) \\ 2\varepsilon_{12} &= \frac{1}{G_{12}}\tau_{12} + \alpha\tau_{12}^3 \end{aligned} \tag{16.22.1}$$

The third equation defines the nonlinear shear stress parameter α.

A fiber matrix shearing term augments each damage mode:

$$\bar{\tau} = \frac{\dfrac{\tau_{12}^2}{2G_{12}} + \dfrac{3}{4}\alpha\tau_{12}^4}{\dfrac{S_{12}^2}{2G_{12}} + \dfrac{3}{4}\alpha S_{12}^4} \tag{16.22.2}$$

which is the ratio of the shear stress to the shear strength.

The matrix cracking failure criteria is determined from

$$F_{\text{matrix}} = \left(\frac{\sigma_2}{S_2}\right)^2 + \bar{\tau} \tag{16.22.3}$$

where failure is assumed whenever $F_{matrix} > 1$. If $F_{matrix} > 1$, then the material constants E_2, G_{12}, υ_1, and υ_2 are set to zero.

The compression failure criteria is given as

$$F_{comp} = \left(\frac{\sigma_2}{2S_{12}}\right)^2 + \left[\left(\frac{C_2}{2S_{12}}\right)^2 - 1\right]\frac{\sigma_2}{C_2} + \bar{\tau} \qquad (16.22.4)$$

where failure is assumed whenever $F_{comb} > 1$. If $F_{comb} > 1$, then the material constants E_2, υ_1, and υ_2 are set to zero.

The final failure mode is due to fiber breakage.

$$F_{\text{fiber}} = \left(\frac{\sigma_1}{S_1}\right)^2 + \bar{\tau} \qquad (16.22.5)$$

Failure is assumed whenever $F_{fiber} > 1$. If $F_{fiber} > 1$, then the constants E_1, E_2, G_{12}, υ_1, and υ_2 are set to zero.

Material Model 24: Piecewise Linear Isotropic Plasticity

This model is quite similar to Model 10 but it includes strain rate effects. Deviatoric stresses are determined that satisfy the yield function

$$\phi = \frac{1}{2} s_{ij}\, s_{ij} - \frac{\sigma_y^{\,2}}{3} \le 0 \tag{16.10.1}$$

where

$$\sigma_y = \beta\left[\sigma_0 + f_h\left(\varepsilon_{eff}^{p}\right)\right] \tag{16.24.1}$$

where the hardening function $f_h\left(\varepsilon_{eff}^{p}\right)$ can be specified in tabular form as an option. Otherwise, linear hardening of the form

$$f_h\left(\varepsilon_{eff}^{p}\right) = E_p\left(\varepsilon_{eff}^{p}\right) \tag{16.10.3}$$

is assumed where E_p and ε_{eff}^{p} are given in Equations (16.3.6) and (16.3.7), respectively. The parameter β accounts for strain rate effects and is defined in a load curve that defines β versus strain rate.

In the implementation of this material model, the deviatoric stresses are updated elastically (see material model 1), the yield function is checked, and if it is satisfied the deviatoric stresses are accepted. If it is not, an increment in plastic strain is computed:

$$\Delta\varepsilon_{eff}^{p} = \frac{\left(\frac{3}{2} s_{ij}^{*}\, s_{ij}^{*}\right)^{1/2} - \sigma_y}{3G + E_p} \tag{16.10.4}$$

is the shear modulus and E_p is the current plastic hardening modulus. The trial deviatoric stress state s_{ij}^{*} is scaled back:

$$s_{ij}^{n+1} = \frac{\sigma_y}{\left(\frac{3}{2} s_{ij}^{*}\, s_{ij}^{*}\right)^{1/2}}\, s_{ij}^{*} \tag{16.10.5}$$

Material Model 26: Crushable Foam

This orthotropic material model does the stress update in the local material system denoted by the subscripts, a, b, and c. The material model requires the following input parameters:

- E, Young's modulus for the fully compacted material;
- ν, Poisson's ratio for the compacted material;
- σ_y, yield stress for fully compacted honeycomb;
- LCA, load curve number for sigma-aa versus either relative volume or volumetric strain (see Figure 16.3.);
- LCB, load curve number for sigma-bb versus either relative volume or volumetric strain (default: LCB = LCA);
- LCC, the load curve number for sigma-cc versus either relative volume or volumetric strain (default: LCC = LCA);
- LCS, the load curve number for shear stress versus either relative volume or volumetric strain (default LCS = LCA);
- $V_{f,}$ relative volume at which the honeycomb is fully compacted;
- E_{aau} , elastic modulus in the uncompressed configuration;
- E_{bbu} , elastic modulus in the uncompressed configuration;
- E_{ccu} , elastic modulus in the uncompressed configuration;
- G_{abu} , elastic shear modulus in the uncompressed configuration;
- G_{bcu} , elastic shear modulus in the uncompressed configuration;
- G_{cau} , elastic shear modulus in the uncompressed configuration;
- LCAB, load curve number for sigma-ab versus either relative volume or volumetric strain (default: LCAB = LCS);
- LCBC, load curve number for sigma-bc versus either relative volume or volumetric strain default: LCBC = LCS);
- LCCA, load curve number for sigma-ca versus either relative volume or volumetric strain (default: LCCA = LCS);
- LCSR, optional load curve number for strain rate effects.

The behavior before compaction is orthotropic where the components of the stress tensor are uncoupled, i.e., an a component of strain will generate resistance in the local a-direction with no coupling to the local b and c directions. The elastic moduli vary linearly with the relative volume from their initial values to the fully compacted values:

$$\begin{aligned} E_{aa} &= E_{aau} + \beta(E - E_{aau}) \\ E_{bb} &= E_{bbu} + \beta(E - E_{bbu}) \\ E_{cc} &= E_{ccu} + \beta(E - E_{ccu}) \\ G_{ab} &= G_{abu} + \beta(G - G_{abu}) \\ G_{bc} &= G_{bcu} + \beta(G - G_{bcu}) \\ G_{ca} &= G_{cau} + \beta(G - G_{cau}) \end{aligned} \tag{16.26.1}$$

where

$$\beta = \max\left[\min\left(\frac{1-V_{\min}}{1-V_f},1\right),0\right] \tag{16.26.2}$$

and G is the elastic shear modulus for the fully compacted honeycomb material

$$G = \frac{E}{2(1+\nu)} \tag{16.26.3}$$

The relative volume V is defined as the ratio of the current volume over the initial volume; typically, V = 1 at the beginning of a calculation. The relative volume, V_{min}, is the minimum value reached during the calculation.

The load curves define the magnitude of the average stress as the material changes density (relative volume). Each curve related to this model must have the same number of points and the same abscissa values. There are two ways to define these curves: as a function of relative volume V, or as a function of volumetric strain defined as:

$$\varepsilon_v = 1 - V \tag{16.26.4}$$

In the former, the first value in the curve should correspond to a value of relative volume slightly less than the fully compacted value. In the latter, the first value in the curve should

be less than or equal to zero corresponding to tension and should increase to full compaction. **When defining the curves, care should be taken that the extrapolated values do not lead to negative yield stresses.**

At the beginning of the stress update we transform each element's stresses and strain rates into the local element coordinate system. For the uncompacted material, the trial stress components are updated using the elastic interpolated moduli according to:

$$\begin{aligned}
\sigma_{aa}^{n+1^{trial}} &= \sigma_{aa}^{n} + E_{aa}\Delta\varepsilon_{aa} \\
\sigma_{bb}^{n+1^{trial}} &= \sigma_{bb}^{n} + E_{bb}\Delta\varepsilon_{bb} \\
\sigma_{cc}^{n+1^{trial}} &= \sigma_{cc}^{n} + E_{cc}\Delta\varepsilon_{cc} \\
\sigma_{ab}^{n+1^{trial}} &= \sigma_{ab}^{n} + 2G_{ab}\Delta\varepsilon_{ab} \\
\sigma_{bc}^{n+1^{trial}} &= \sigma_{bc}^{n} + 2G_{bc}\Delta\varepsilon_{bc} \\
\sigma_{ca}^{n+1^{trial}} &= \sigma_{ca}^{n} + 2G_{ca}\Delta\varepsilon_{ca}
\end{aligned} \tag{16.26.5}$$

Then we independently check each component of the updated stresses to ensure that they do not exceed the permissible values determined from the load curves, e.g., if

$$\left|\sigma_{ij}^{n+1^{trial}}\right| > \lambda\sigma_{ij}(V_{\min})$$

then

$$\sigma_{ij}^{n+1} = \sigma_{ij}(V_{\min})\frac{\lambda\sigma_{ij}^{n+1^{trial}}}{\left|\sigma_{ij}^{n+1^{trial}}\right|} \tag{16.26.6}$$

The parameter λ is either unity or a value taken from the load curve number, LCSR, that defines λ as a function of strain rate. Strain rate is defined here as the Euclidean norm of the deviatoric strain rate tensor.

For fully compacted material we assume that the material behavior is elastic-perfectly plastic and updated the stress components according to

$$s_{ij}^{trial} = s_{ij}^{n} + 2G\Delta\varepsilon_{ij}^{dev^{n+1/2}} \tag{16.26.7}$$

where the deviatoric strain increment is defined as

$$\Delta\varepsilon_{ij}^{dev} = \Delta\varepsilon_{ij} - \frac{1}{3}\Delta\varepsilon_{kk}\delta_{ij} \tag{16.26.8}$$

We next check to see if the yield stress for the fully compacted material is exceeded by comparing

$$s_{eff}^{trial} = \left(\frac{3}{2}s_{ij}^{trial}s_{ij}^{trial}\right)^{1/2} \tag{16.26.9}$$

the effective trial stress, to the yield stress σ_y. If the effective trial stress exceeds the yield stress, we simply scale back the stress components to the yield surface:

$$s_{ij}^{n+1} = \frac{\sigma_y}{s_{eff}^{trial}}s_{ij}^{trial} \tag{16.26.10}$$

We can now update the pressure using the elastic bulk modulus, K:

$$p^{n+1} = p^n - K\Delta\varepsilon_{kk}^{n+1/2} \tag{16.26.11}$$

$$K = \frac{E}{3(1-2\nu)}$$

and obtain the final value for the Cauchy stress

$$\sigma_{ij}^{n+1} = s_{ij}^{n+1} - p^{n+1}\delta_{ij} \tag{16.26.12}$$

After completing the stress update, we transform the stresses back to the global configuration.

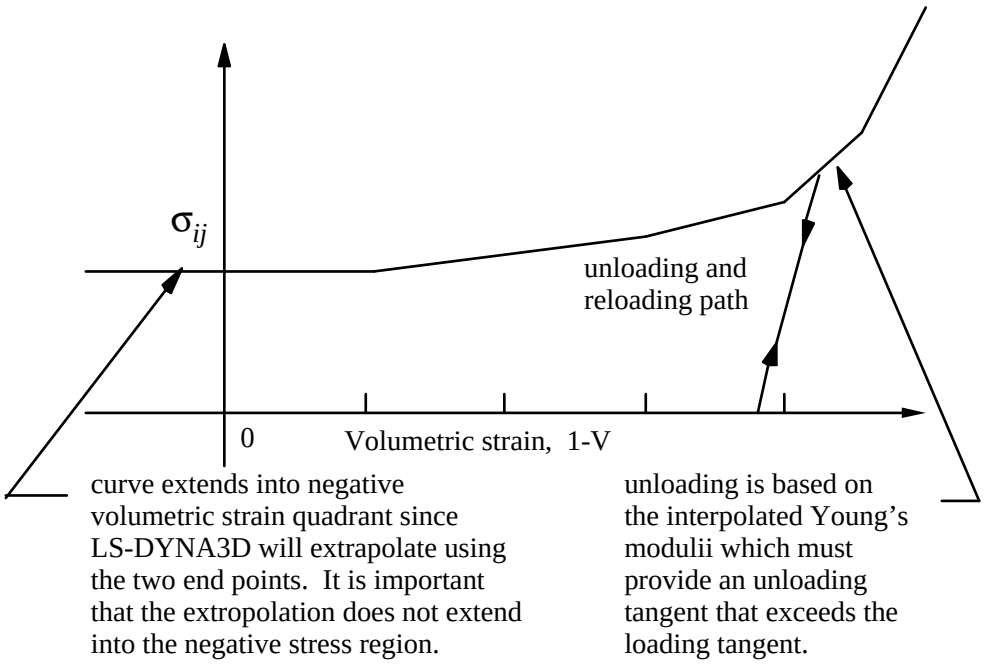


Figure 16.3. Stress quantity versus volumetric strain. Note that the "yield stress" at a volumetric strain of zero is nonzero. In the load curve definition, the "time" value is the volumetric strain and the "function" value is the yield stress.

Material Type 27: Incompressible Mooney-Rivlin Rubber

This material model, available for solid elements only, provides an alternative to the Blatz-Ko rubber model. The implementation is due to Maker [private communication]. The strain energy density function is defined as in terms of the input constants A, B, and υ as:

$$W(I_1, I_2, I_3) = A(I_1 - 3) + B(I_2 - 3) + C(\frac{1}{I_3^2} - 1) + D(I_3 - 1)^2 \qquad (16.27.1)$$

where

$$C = .5 * A + B \qquad (16.27.2a)$$

$$D = \frac{A(5\upsilon - 2) + B(11\upsilon - 5)}{2(1 - 2\upsilon)} \qquad (16.27.2b)$$

υ = Poisson's ratio

G=2(A+B) = shear modulus of linear elasticity

I_1, I_2, I_3 = strain invariants in terms of the principal stretches

$$I_1 = \lambda_1^2 + \lambda_2^2 + \lambda_3^2$$

$$I_2 = \lambda_1^2\lambda_2^2 + \lambda_2^2\lambda_3^2 + \lambda_3^2\lambda_1^2$$

$$I_3 = \lambda_1^2\lambda_2^2\lambda_3^2$$

Recommended values for Poisson's ratio are between .490 and .495 or higher. Lower values may lead to instabilities. In the derivation of the constants C and D incompressibility is assumed.

The derivation of the constants C and D is straigthforward [Feng, 1993] and is included here since we were unable to locate it in the literature. The principal components of Cauchy stress, σ_i, are given by [Ogden, 1984]

$$J\sigma_i = \lambda_i \frac{\partial W}{\partial \lambda_i}$$

For uniform dilation

$$\lambda_1 = \lambda_2 = \lambda_3 = \lambda$$

thus the pressure, *p*, is obtained (please note the sign convention),

$$p = \sigma_1 = \sigma_2 = \sigma_3 = \frac{2}{\lambda^3}\left(\lambda^2 \frac{\partial W}{\partial I_1} + 2\lambda^4 \frac{\partial W}{\partial I_2} + \lambda^6 \frac{\partial W}{\partial I_3}\right)$$

The relative volume, *V*, can be defined in terms of the stretches as:

$$V = \lambda^3 = \frac{\text{new volume}}{\text{old volume}}$$

For small volumetric deformations the bulk modulus, *K*, can be defined as the ratio of the pressure over the volumetric strain as the relative volume approaches unity:

$$K = \lim_{V \to 1}\left(\frac{p}{V-1}\right) K = \lim_{V \to 1}\left(\frac{p}{V-1}\right)$$

The partial derivatives of *W* lead to:

$$\frac{\partial W}{\partial I_1} = A$$

$$\frac{\partial W}{\partial I_2} = B$$

$$\frac{\partial W}{\partial I_3} = -2CI_3^{-3} + 2D(I_3 - 1) = -2C\lambda^{-18} + 2D\left(\lambda^6 - 1\right)$$

$$p = \frac{2}{\lambda^3}\left\{A\lambda^2 + 2\lambda^4 B + \lambda^6\left[-2C\lambda^{-18} + 2D\left(\lambda^6 - 1\right)\right]\right\}$$

$$= \frac{2}{\lambda^3}\left\{A\lambda^2 + 2\lambda^4 B - 2C\lambda^{-12} + 2D\left(\lambda^{12} - \lambda^6\right)\right\}$$

In the limit as the stretch ratio approaches 1 the pressure must approach zero:

$$\lim_{\lambda \to 1} p = 0$$

Therefore, $A + 2B - 2C = 0$ and

$$\therefore C = 0.5A + B$$

To solve for D we note that:

$$K = \lim_{V\to 1}\left(\frac{p}{V-1}\right) = \lim_{\lambda\to 1}\frac{\frac{2}{\lambda^3}\left\{A\lambda^2 + 2\lambda^4 B - 2C\lambda^{-12} + 2D\left(\lambda^{12} - \lambda^6\right)\right\}}{\lambda^3 - 1}$$

$$= 2\lim_{\lambda\to 1}\frac{A\lambda^2 + 2\lambda^4 B - 2C\lambda^{-12} + 2D\left(\lambda^{12} - \lambda^6\right)}{\lambda^6 - \lambda^3}$$

$$= 2\lim_{\lambda\to 1}\frac{2A\lambda + 8\lambda^3 B + 24C\lambda^{-13} + 2D\left(12\lambda^{11} - 6\lambda^5\right)}{6\lambda^5 - 3\lambda^2}$$

$$= \frac{2}{3}\left(2A + 8B + 24C + 12D\right) = \frac{2}{3}\left(14A + 32B + 12D\right)$$

We therefore obtain:

$$14A + 32B + 12D = \frac{3}{2}K = \frac{3}{2}\left(\frac{2G(1+\upsilon)}{3(1-2\upsilon)}\right) = \frac{2(A+B)(1+\upsilon)}{(1-2\upsilon)}$$

Solving for D we obtain the desired equation:

$$D = \frac{A(5\upsilon - 2) + B(11\upsilon - 5)}{2(1-2\upsilon)}$$

Material Type 28: Resultant Plasticity

This plasticity model, based on resultants as illustrated in Figure 16.4, is very cost effective but not as accurate as through-thickness integration. This model is available only with the C^0 triangular, Belytschko-Tsay shell, and the Belytschko beam element since these elements, unlike the Hughes-Liu elements, lend themselves very cleanly to a resultant formulation.

In applying this model to shell elements the resultants are updated incrementally using the midplane strains ε^m and curvatures κ:

$$\Delta n = \Delta t\, C \varepsilon^m \tag{16.28.1}$$

$$\Delta m = \Delta t \frac{h^3}{12} C \kappa \tag{16.28.2}$$

where the plane stress constitutive matrix is given in terms of Young's Modulus E and Poisson's ratio ν as:

$$C = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \tag{16.28.3}$$

Defining

$$\bar{n} = n_{xx}^2 - n_{xx}n_{yy} + n_{yy}^2 + 3n_{xy}^2 \tag{16.28.4}$$

$$\bar{m} = m_{xx}^2 - m_{xx}m_{yy} + m_{yy}^2 + 3m_{xy}^2 \tag{16.28.5}$$

$$\bar{m}\bar{n} = m_{xx}n_{xx} - \tfrac{1}{2}m_{xx}n_{yy} - \tfrac{1}{2}n_{xx}m_{yy} + m_y n_y + 3m_{xy}n_{xy} \tag{16.28.6}$$

the Ilyushin yield function becomes

$$f(m,n) = \bar{n} + \frac{4|\bar{m}\bar{n}|}{h\sqrt{3}} + \frac{16\bar{m}}{h^2} \le n_y^2 = h^2\sigma_y^2 \tag{16.28.7}$$

In our implementation we update the resultants elastically and check to see if the yield condition is violated:

$$\text{f(m,n)} > n_y^2 \tag{16.28.8}$$

If so, the resultants are scaled by the factor α:

$$\alpha = \sqrt{\frac{n_y^2}{f(m,n)}} \tag{16.28.9}$$

We update the yield stress incrementally:

$$\sigma_y^{n+1} = \sigma_y^n + E^P \Delta\varepsilon_{plastic}^{eff} \tag{16.28.10}$$

where E^P is the plastic hardening modulus which in incremental plastic strain is approximated by

$$\Delta\varepsilon_{plastic}^{eff} = \frac{\sqrt{f(m,n)} - n_y}{h\left(3G + E^p\right)} \tag{16.28.11}$$

Kennedy et al. report that this model predicts results that may be too stiff; users of this model should proceed cautiously.

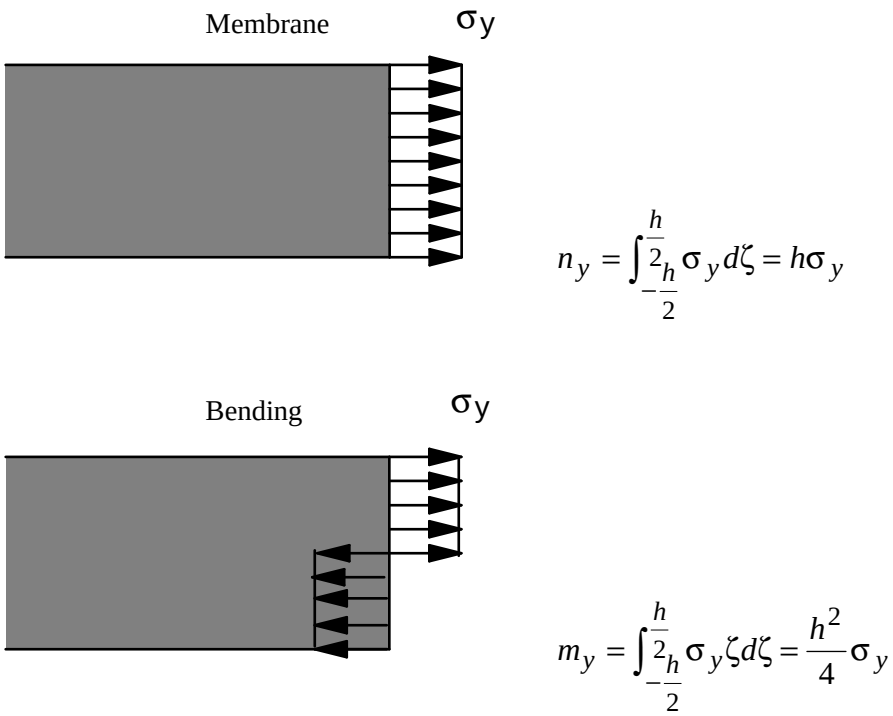


Figure 16.4. Full section yield using resultant plasticity.

In applying this material model to the Belytschko beam the flow rule changes to

$$f(m,n) = \hat{f}_x^2 + \frac{4\hat{m}_y^2}{3I_{yy}} + \frac{4\hat{m}_z^2}{3I_{zz}} \le n_y^2 = A^2\sigma_y^2 \qquad (16.28.12)$$

where a rectangular cross section has been assumed. The flow rule is checked after the resultants have been updated elastically according to Equations (4.16)-(4.18). The yield condition is checked [Equation (16.28.8)], and if it is violated, the resultants are scaled as described above.

This model is frequently applied to beams with non-rectangular cross sections. The accuracy of the results obtained should be viewed with some healthy suspicion. No work hardening is available with this model.

Material Type 31: Frazer-Nash Rubber Model

This model implements a hyperelastic constitutive law for solid elements described by Kenchington [1988]. The strain energy functional, U, is defined in terms of the constants C_{ijkl} as:

$$U = C_{100}\, I_1 + C_{200}\, I_1^2 + C_{300}\, I_1^3 + C_{400}\, I_1^4 + C_{010}\, I_2 + C_{020} I_2^2 + C_{110} I_1 I_2 + C_{210} I_1^2 I_2 + C_{001} I_3 + C_{101} I_1 I_3 \quad (16.31.1)$$

The partial derivatives of U with respect to a component of strain give the corresponding components of stress:

$$S_{ij} = \frac{\partial U}{\partial E_{ij}} = 2\frac{\partial U}{\partial C_{ij}} \quad (16.31.2)$$

Here S_{ij}, E_{ij}, and C_{ij} are the second Piola-Kirchhoff stress tensor, the Green-St. Venant strain tensor, and the right Cauchy-Green deformation tensor, respectively.

Material Type 37: Transversely Anisotropic Elastic-Plastic

This fully iterative plasticity model is available only for shell elements. The input parameters for this model are: Young's modulus E; Poisson's ratio υ; the yield stress; the tangent modulus E_t; and the anisotropic hardening parameter R.

Consider Cartesian reference axes which are parallel to the three symmetry planes of anisotropic behavior. Then the yield function suggested by Hill [1948] can be written

$$F(\sigma_{22}-\sigma_{33})^2 + G(\sigma_{33}-\sigma_{11})^2 + H(\sigma_{11}-\sigma_{22})^2 + 2L\sigma_{23}^2 + 2M\sigma_{31}^2 + 2N\sigma_{12}^2 - 1 = 0 \qquad (16.37.1)$$

where σ_{y1}, σ_{y2}, and σ_{y3}, are the tensile yield stresses and σ_{y12}, σ_{y23}, and σ_{y31} are the shear yield stresses. The constants F, G, H, L, M, and N are related to the yield stress by

$$\begin{aligned} 2L &= \frac{1}{\sigma_{23}^2} \\ 2M &= \frac{1}{\sigma_{y31}^2} \\ 2N &= \frac{1}{\sigma_{y12}^2} \\ 2F &= \frac{1}{\sigma_{y2}^2} + \frac{1}{\sigma_{y3}^2} - \frac{1}{\sigma_{y1}^2} \\ 2G &= \frac{1}{\sigma_{y3}^2} + \frac{1}{\sigma_{y1}^2} - \frac{1}{\sigma_{y2}^2} \\ 2H &= \frac{1}{\sigma_{y1}^2} + \frac{1}{\sigma_{y2}^2} - \frac{1}{\sigma_{y3}^2}. \end{aligned} \qquad (16.37.2)$$

The isotropic case of von Mises plasticity can be recovered by setting

$$F = G = H = \frac{1}{2\sigma_y^2}$$

and $L = M = N = \dfrac{3}{2\sigma_y^2}$.

For the particular case of transverse anisotropy, where properties do not vary in the x_1-x_2 plane, the following relations hold:

(16.37.3)

$$2F = 2G = \frac{1}{\sigma_{y3}^2}$$

$$2H = \frac{2}{\sigma_y^2} - \frac{1}{\sigma_{y3}^2}$$

$$N = \frac{2}{\sigma_y^2} - \frac{1}{2}\frac{1}{\sigma_{y3}^2}$$

where it has been assumed that $\sigma_{y1} = \sigma_{y2} = \sigma_y$.

Letting $K = \frac{\sigma_y}{\sigma_{y3}}$, the yield criterion can be written

$$F(\sigma) = \sigma_e = \sigma_y , \tag{16.37.4}$$

where

$$F(\sigma) \equiv \left[\sigma_{11}^2 + \sigma_{22}^2 + K^2\sigma_{33}^2 - K^2\sigma_{33}(\sigma_{11}+\sigma_{22}) - \left(2-K^2\right)\sigma_{11}\sigma_{22} + 2L\sigma_y^2\left(\sigma_{23}^2+\sigma_{31}^2\right) + 2\left(2-\frac{1}{2}K^2\right)\sigma_{12}^2\right]^{1/2}$$

The rate of plastic strain is assumed to be normal to the yield surface so $\dot{\varepsilon}_{ij}^p$ is found from

$$\dot{\varepsilon}_{ij}^p = \lambda\frac{\partial F}{\partial \sigma_{ij}} . \tag{16.37.5}$$

Now consider the case of plane stress, where $\sigma_{33} = 0$. Also, define the anisotropy input parameter R as the ratio of the in-plane plastic strain rate to the out-of-plane plastic strain rate:

$$R = \frac{\dot{\varepsilon}_{22}^p}{\dot{\varepsilon}_{33}^p} . \tag{16.37.6}$$

It then follows that

$$R = \frac{2}{K^2} - 1 . \tag{16.37.7}$$

Using the plane stress assumption and the definition of R, the yield function may now be written

$$F(\sigma)=\left[\sigma_{11}^2+\sigma_{22}^2-\frac{2R}{R+1}\sigma_{11}\sigma_{22}+2\frac{2R+1}{R+1}\sigma_{12}^2\right]^{1/2}. \qquad (16.37.8)$$

Material Type 38: Blatz-Ko Compressible Foam

The strain energy functional for the compressible foam model is given by

$$W(I_1, I_2, I_3) = \frac{\mu}{2}\left(\frac{I_2}{I_3} + 2\sqrt{I_3} - 5\right) \tag{16.38.1}$$

Blatz and Ko [1962] suggested this form for a 47 percent volume polyurethane foam rubber with a Poisson's ratio of 0.25. The second Piola-Kirchhoff stresses are given as

$$S^{ij} = \mu\left[\left(I\delta_{ij} - G_{ij}\right)\frac{1}{I_3} + \left(\sqrt{I_3} - \frac{I_2}{I_3}\right)G^{ij}\right] \tag{16.38.2}$$

Material Type 51: Temperature and Rate Dependent Plasticity

The kinematics associated with this model are discussed by Bammann [1989]. The description below is taken nearly verbatim from Bammann [1989].

With the assumption of linear elasticity we can write

$$\overset{o}{\sigma} = \lambda \mathrm{tr}\,(D^e)1 + 2\mu D^e \tag{16.51.1}$$

where the Cauchy stress σ is convected with the elastic spin W^e as

$$\overset{o}{\sigma} = \dot{\sigma} - W^e\sigma + \sigma W^e \tag{16.51.2}$$

This is equivalent to writing the constitutive model with respect to a set of directors whose direction is defined by the plastic deformation (Bammann and Aifantis [1987], Bammann and Johnson [1987]). Decomposing both the skew-symmetric and symmetric parts of the velocity gradient into elastic and plastic parts, we write for the elastic stretching D^e and the elastic spin W^e

$$D^e = D - D^p\,, \qquad W^e = W - W^p \tag{16.51.3}$$

Within this structure it is now necessary to prescribe an equation for the plastic spin W^p in addition to the normally prescribed flow rule for D^p As proposed, we assume a flow rule of the form

$$D^p = f(T)\sinh\left[\frac{|\xi| - \kappa - Y(T)}{V(T)}\right]\frac{\xi}{|\xi|} \tag{16.51.4}$$

where T is the temperature, κ is the scalar hardening variable, ξ is the difference between the deviatoric Cauchy stress σ' and the tensor variable α:

$$\xi = \sigma' - \frac{2}{3}\alpha \tag{16.51.5}$$

and f(T), Y(T), V(T) are scalar functions whose specific dependence upon the temperature is given below.

The evolution of the internal variables α and κ are prescribed in a 'hardening minus recovery' format as

$$\dot{\alpha} = h(T)D^p - \left[r_d(T)\left|D^p\right| + r_s(T)\right]|\alpha|\alpha \tag{16.51.6}$$

$$\dot{\kappa} = H(T)D^p - \left[R_d(T)\left|D^p\right| - R_s(T)\right]\kappa^2 \qquad (16.51.7)$$

where h and H are the hardening moduli, r_s (T) and R_s (T) are scalar functions describing the diffusion-controlled static or thermal recovery, and r_d (T) and R_d (T) are the functions describing dynamic recovery.

If we assume that $W^p = 0$, we recover the Jaumann stress rate which results in the prediction of an oscillatory shear stress response in simple shear when coupled with a Präger kinematic hardening assumption [Johnson and Bammann 1984]. Alternatively we can choose,

$$W^p = R^T \dot{U} U^{-1} R\ , \qquad (16.51.8)$$

which recovers the Green-Naghdi rate of Cauchy stress. The model employing this rate allows a reasonable prediction of directional softening for some materials but in general underpredicts the softening and does not accurately predict the axial stresses which occur in the torsion of a thin-walled tube.

The final equation necessary to complete our description of high strain rate deformation is one which allows us to compute the temperature change during the deformation. In the absence of a coupled thermomechanical finite element code, we assume adiabatic temperature change and follow the empirical assumption that 90% of the plastic work is dissipated as heat. Hence

$$\dot{T} = \frac{.9}{\rho C_v}\left(\sigma \cdot D^p\right), \qquad (16.51.9)$$

where ρ is the density of the material and C_v the specific heat.

In terms of the input parameters, the functions defined above become:

$$\begin{aligned}
V(T) &= C1\ \exp(-C2/T)\\
Y(T) &= C3\ \exp(C4/T)\\
f(T) &= C5\ \exp(-C6/T)\\
rd(T) &= C7\ \exp(-C8/T)\\
h(T) &= C9\ \exp(C10/T)\\
rs(T) &= C11\exp(-C12/T)\\
RD(T) &= C13\exp(-C14/T)\\
H(T) &= C15\exp(C16/T)\\
RS(T) &= C17\exp(-C18/T)
\end{aligned}$$

and the heat generation coefficient is

$$HC = \frac{.9}{\rho C_V} \tag{16.51.10}$$

Material Type 52: Sandia Damage Model

This model is the same as model 51 but it includes a damage function for degrading the strength of the material. The evolution of the damage parameter ϕ is defined by

$$\dot{\phi} = \beta\left[\frac{1}{(1-\phi)^N} - (1-\phi)\right]\left|D^p\right| \tag{16.52.1}$$

in which

$$\beta = \sinh\left[\frac{2(2N-1)p}{(2N-1)\bar{\sigma}}\right]$$

where p is the pressure and $\bar{\sigma}$ is the effective stress. More details of the model can be found in Bammann et al. [1990].

Material Type 53: Low Density Polyurethane Foam

A rigid, low density, closed cell, polyurethane foam model developed at Sandia Laboratories [Neilsen, Morgan, and Krieg, 1987] has been recently implemented for modeling impact limiters in shipping containers. A number of such foams were tested at Sandia and reasonable fits to the experimental data were obtained. We will briefly outline the equations here and refer the reader the the original documents for more details.

In some respects this model is similar to the crushable honeycomb model in that the components of the stress tensor are uncoupled until full volumetric compaction is achieved. However, unlike the honeycomb model this material possesses no directionality but includes the effects of confined air pressure in its overall response characteristics..

$$\sigma_{ij} = \sigma_{ij}^{sk} - \delta_{ij}\sigma^{air} \tag{16.53.1}$$

where σ_{ij}^{sk} is the skeletal stress and σ^{air} is the air pressure computed from the equation:

$$\sigma^{air} = -\frac{p_0\gamma}{1+\gamma-\phi} \tag{16.53.2}$$

where p_0 is the initial foam pressure usually taken as the atmospheric pressure and γ defines the volumetric strain

$$\gamma = V \quad 1 + \gamma_0 \tag{16.53.3}$$

where V is the relative volume and γ_0 is the initial volumetric strain which is typically zero. The yield condition is applied to the principal skeletal stresses which are updated independently of the air pressure. We first obtain the skeletal stresses:

$$\sigma_{ij}^{sk} = \sigma_{ij} + \sigma_{ij}\,\sigma^{air} \tag{16.53.4}$$

and compute the trial stress, σ^{skt}

$$\sigma_{ij}^{skt} = \sigma_{ij}^{sk} + E\dot{\varepsilon}_{ij}\Delta t \tag{16.53.5}$$

where E is Young's modulus. Since Poisson's ratio is zero, the update of each stress component is uncoupled and 2G=E where G is the shear modulus. The yield condition is applied to the principal skeletal stresses such that if the magnitude of a principal trial stress component, σ_i^{skt}, exceeds the yield stress, σ_y, then

$$\sigma_i^{sk} = \min(\sigma_y, \left|\sigma_i^{skt}\right|)\frac{\sigma_i^{skt}}{\left|\sigma_i^{skt}\right|} \tag{16.53.6}$$

The yield stress is defined by

$$\sigma_y = a + b(1 + c\gamma) \tag{16.53.7}$$

where a, b, and c are user defined input constants. After scaling the principal stresses they are transformed back into the global system.and the final stress state is computed

$$\sigma_{ij} = \sigma_{ij}^{sk} - \delta_{ij}\sigma^{air} \ . \tag{16.53.8}$$

Material Type 57: Urethane Foam

A urethane foam model is available to model highly compressible foams such as those used in seat cushions and as padding on the Side Impact Dummy (SID). The compressive behavior is illustrated in Figure 16.5 where a hystersis on unloading is shown. This behavior under uniaxial loading is assumed not to significantly couple in the transverse directions. In tension the material behaves in a linear fashion until tearing occurs. Although our implementation may be somewhat unique, it was motivated by Storakers [Storakers 1986] and Shkolnikov [1991].

The model uses tabulated input data for the loading curve where the nominal stresses are defined as a function of the elongations, ε_i, which are defined in terms of the principal stretches, λ_i , as:

$$\varepsilon_i = \lambda_i - 1 \tag{16.57.1}$$

The stretch ratios are found by solving for the eigenvalues of the left stretch tensor, V_{ij}, which is obtained via a polar decomposition of the deformation gradient matrix, F_{ij}. Recall that,

$$F_{ij} = R_{ik}U_{kj} = V_{ik}R_{kj} \tag{14.22}$$

The update of V_{ij} follows the numerically stable approach of Equations (14.29), [Taylor and Flanagan 1989].

For compressive elongations, the corresponding values of the nominal stresses, τ_i, are interpolated from the tabulated function,

$$\tau_i = g(\alpha_j)F(\varepsilon_i) \tag{16.57.2}$$

where $g(\alpha_j,\varepsilon_i)$ is scaling function of state variables α_j and the elongations. The function $g(\alpha_j,\varepsilon_i)$ equals 1 for loading and is ≤ 1 for unloading. By the use of two input shape factors the observed unloading behavior of the foams can be closely approximated. When hysteretic unloading is used, the reloading will follow the unloading curve if the decay constant, β, is set to zero. If β is nonzero the decay over time, *t*, to the original loading curve is governed by the expression:

$$1.\text{-}e^{-\beta t} \tag{16.57.3}$$

If the elongations are tensile, the nominal stresses are given by

$$\tau_i = E\varepsilon_i \tag{16.57.4}$$

and the Cauchy stresses in the principal system become

$$\sigma_i = \frac{\tau_i}{\lambda_j \lambda_k} \tag{16.57.5}$$

The stresses can now be transformed back into the global system for the nodal force calculations.

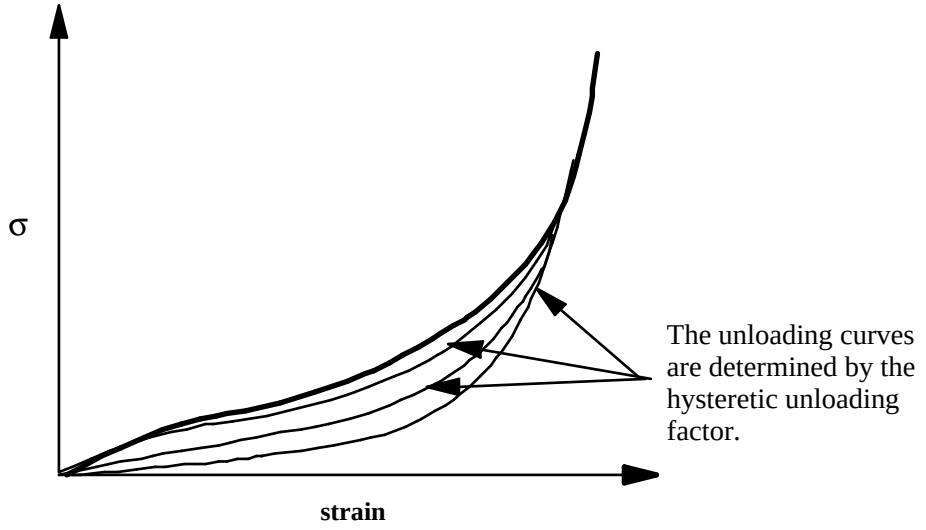


Figure 16.5.

Material Type 60: Elastic With Viscosity

This material model was developed to simulate the forming of glass products (e.g., car windshields) at high temperatures. Deformation is by viscous flow but elastic deformations can also be large. The material model, in which the viscosity may vary with temperature, is suitable for treating a wide range of viscous flow problems and is implemented for brick and shell elements.

Volumetric behavior is treated as linear elastic. The deviatoric strain rate is considered to be the sum of elastic and viscous strain rates:

$$\underset{\sim}{\dot{\varepsilon}'}_{\sim\ total} = \underset{\sim}{\dot{\varepsilon}'}_{\sim\ elastic} + \underset{\sim}{\dot{\varepsilon}'}_{\sim\ viscous} = \frac{\underset{\sim}{\dot{\sigma}'}}{2G} + \frac{\underset{\sim}{\sigma'}}{2\nu} \tag{16.60.1}$$

where G is the elastic shear modulus, v is the viscosity coefficient, and ~ indicates a tensor. The stress increment over one time step dt is

$$\underset{\sim}{d\sigma'} = 2G\,\underset{\sim}{\dot{\varepsilon}'}_{\sim\ total}\,dt - \frac{G}{\upsilon}dt\,\underset{\sim}{\sigma'} \tag{16.60.2}$$

The stress before the update is used for $\underset{\sim}{\sigma'}$. For shell elements, the through-thickness strain rate is calculated as follows.

$$d\sigma_{33} = 0 = K\left(\dot{\varepsilon}_{11} + \dot{\varepsilon}_{22} + \dot{\varepsilon}_{33}\right)dt + 2G\dot{\varepsilon}'_{33}\,dt - \frac{G}{\upsilon}dt\sigma'_{33} \tag{16.60.3}$$

where the subscript ij=33 denotes the through-thickness direction and K is the elastic bulk modulus. This leads to:

$$\dot{\varepsilon}_{33} = -a\left(\dot{\varepsilon}_{11} + \dot{\varepsilon}_{22}\right) + bp \tag{16.60.4}$$

$$a = \frac{\left(K - \frac{2}{3}G\right)}{\left(K + \frac{4}{3}G\right)} \tag{16.60.5}$$

$$b = \frac{Gdt}{\upsilon\left(K + \frac{4}{3}G\right)} \tag{16.60.6}$$

in which p is the pressure defined as the negative of the hydrostatic stress.

Material Type 61: Maxwell/Kelvin Viscoelastic with Maximum Strain

The shear relaxation behavior is described for the Maxwell model by:

$$G(t) = G_\infty + (G_0 - G_\infty)\, e^{-\beta t}$$

A Jaumann rate formulation is used

$$\overset{\nabla}{\sigma}'_{ij} = 2 \int_0^t G(t-\tau)\, D'_{ij}(\tau) dt$$

where the prime denotes the deviatoric part of the stress rate, $\overset{\nabla}{\sigma}_{ij}$, and the strain rate D_{ij}. For the Kelvin model the stress evolution equation is defined as:

$$\dot{s}_{ij} + \frac{1}{\tau} s_{ij} = (1+\delta_{ij}) G_0 \dot{e}_{ij} + (1+\delta_{ij}) \frac{G_\infty}{\tau} \dot{e}_{ij}$$

Material Type 62: Viscous Foam

This model was written to represent the energy absorbing foam found on certain crash dummies. This model was added to model the 'Confor Foam' on the ribs of the Eurosid.

The model consists of a nonlinear elastic stiffness in parallel with a viscous damper. The elastic stiffness is intended to limit total crush while the viscosity absorbs energy. The stiffness E_2 exists to prevent timestep problems. Both E_1 and V_2 are nonlinear with crush as follows:

$$E_1^t = E_1\left(V^{-n_1}\right)$$

$$V_2^t = V_2\left(abs(1-V)\right)^{n_2}$$

where V is the relative volume defined by the ratio of the current to initial volume. Typical values are (units of N, mm, s)

E_1=0.0036 n_1=4.0	V_2=0.0015	
E_2=100.0	n_2=0.2	n=0.05

Material Type 63: Crushable Foam

The intent of this model is model crushable foams in side impact and other applications where cyclic behavior is unimportant.

This isotropic foam model crushes one-dimensionally with a Poisson's ratio that is essentially zero. The stress versus strain behavior is depicted in Figure 16.6 where an example of unloading from point **a** to the tension cutoff stress at **b** then unloading to point **c** and finally reloading to point **d** is shown. At point **d** the reloading will continue along the loading curve. It is important to use nonzero values for the tension cutoff to prevent the disintegration of the material under small tensile loads. For high values of tension cutoff the behavior of the material will be similar in tension and compression.

In the implementation we assume that Young's modulus is constant and update the stress assuming elastic behavior.

$$\sigma_{ij}^{trial} = \sigma_{ij}^{n} + E\dot{\varepsilon}_{ij}^{n+\frac{1}{2}}\Delta t^{n+\frac{1}{2}} \tag{16.63.1}$$

The magnitudes of the principal values, $\sigma_i^{trial}, i = 1,3$ are then checked to see if the yield stress, σ_y, is exceeded and if so they are scaled back to the yield surface:

$$if \quad \sigma_y < \left|\sigma_i^{trial}\right| \quad \text{then} \quad \sigma_i^{n+1} = \sigma_y \frac{\sigma_i^{trial}}{\left|\sigma_i^{trial}\right|} \tag{16.63.2}$$

After the principal values are scaled, the stress tensor is transformed back into the global system. As seen in Figure 16.6, the yield stress is a function of the natural logarithm of the relative volume, V, i.e., the volumetric strain.

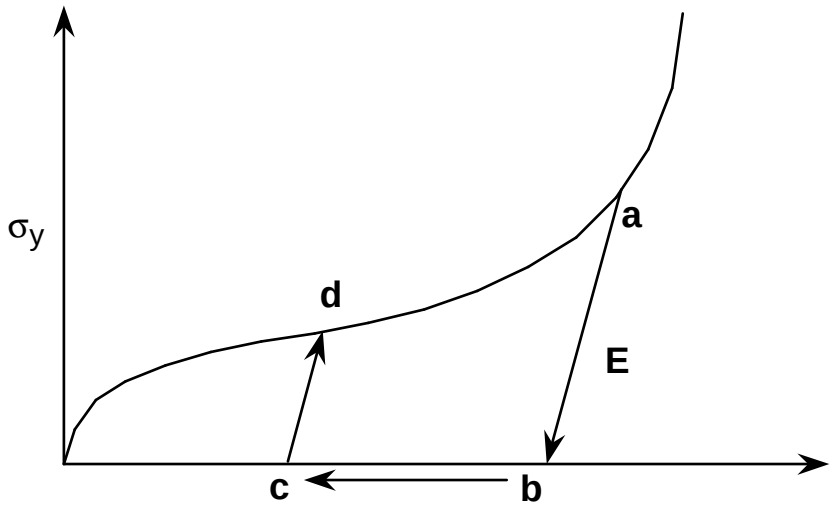


Figure 16.6. Yield stress versus volumetric strain curve for the crusable foam.

17. EQUATION OF STATE MODELS

The nine equation of state forms are:

1: linear polynomial,
2: JWL high explosive,
3: Sack “Tuesday” high explosive,
4: Gruneisen,
5: ratio of polynomials,
6: linear polynomial with energy deposition,
7: ignition and growth of reaction in high explosives,
8: tabulated compaction,
9: tabulated.

With the exception of the ignition and growth form, these equations of state are completely vectorized. The forms of the first five equations of state are given in the KOVEC user’s manual [Woodruff 1973] as well as below.

17.1 Equation of State Form 1: Linear Polynomial

This polynomial equation of state, linear in internal energy, is given by

$$p = C_0 + C_1\mu + C_2\mu^2 + C_3\mu^3 + (C_4 + C_5\mu + C_6\mu^2)E \qquad (17.1.1)$$

where

$$\mu = \frac{1}{V} - 1. \qquad (17.1.2)$$

In expanded elements, we set the coefficients of μ^2 to zero, i.e., $C_2 = C_6 = 0$.

17.2 Equation of State Form 2: JWL High Explosive

The JWL equation of state defines pressure as a function of relative volume and internal energy as

$$p = A\left(1-\frac{\omega}{R_1V}\right)e^{-R_1V} + B\left(1-\frac{\omega}{R_2V}\right)e^{-R_2V} + \frac{\omega E}{V} \qquad (17.2.1)$$

where ω_1, A, B, R_1, and R_2 are input parameters. This equation of state is normally used for determining the pressure of the detonation products of high explosives in applications involving metal accelerations. Input parameters for this equation are given by Dobratz [1981] for a variety of high explosive materials.

17.3 Equation of State Form 3: Sack "Tuesday" High Explosives

Pressure of detonation products is given in terms of the relative volume and internal energy as [Woodruff 1973]:

$$p = \frac{A_3}{V^{A_1}} e^{-A_2 V} \left(1 - \frac{B_1}{V}\right) + \frac{B_2}{V} E \qquad (17.3.1)$$

where A_1, A_2, A_3, B_1, and B_2 are user-defined input parameters.

17.4 Equation of State Form 4: Gruneisen

The Gruneisen equation of state with cubic shock velocity-particle velocity defines pressure for compressed material as

$$p = \frac{\rho_0 C^2 \mu \left[1+\left(1-\frac{\gamma_0}{2}\right)\mu - \frac{a}{2}\mu^2\right]}{\left[1-(S_1-1)\mu - s_2\frac{\mu^2}{\mu+1} - s_3\frac{\mu^3}{(\mu+1)^2}\right]} + (\gamma_0 + a\mu)E \qquad (17.4.1)$$

and for expanded materials as

$$p = \rho_0 C^2 \mu + (\gamma_0 + a\mu)E \qquad (17.4.2)$$

where C is the intercept of the u_s-u_p curve, S_1, S_2, and S_3 are the coefficients of the slope of the u_s-u_p curve, γ_0 is the Gruneisen gamma, and a is the first order volume correction to γ_0. Constants C, S_1, S_2, S_3, γ_0, and a are all input parameters.

17.5 Equation of State Form 5: Ratio of Polynomials

The ratio of polynomials equation of state defines the pressure as

$$p = \frac{F_1 + F_2 E + F_3 E^2 + F_4 E^3}{F_5 + F_6 E + F_7 E^2}\left(1 + \alpha\mu\right) \tag{17.5.1}$$

where

$$F_i = \sum_{j=0}^{n} A_{ij} m^j . \quad n = 4 \text{ if } i < 3 \qquad n = 3 \text{ if } i \geq 3 \tag{17.5.2}$$

$$\mu = \frac{\rho}{\rho_0 - 1}$$

In expanded zoned F_1 is replaced by $F'_1 = F_1 + \beta\mu^2$ Constants A_{ij}, α, and β are user input.

17.6 Equation of State Form 6: Linear with Energy Deposition

See Form 1 for the equation of state. Internal energy is increased according to an energy deposition rate versus time curve defined in the user input.

17.7 Equation of State Form 7: Ignition and Growth Model

A JWL equation of state defines the pressure in the unreacted high explosive as

$$P_e = A_e\left(1-\frac{\omega_e}{R1_e V_e}\right)e^{-R1_e V_e} + B_e\left(1-\frac{\omega_e}{R2_e V_e}\right)e^{-R2_e V_e} + \frac{\omega_e E}{V_e} \qquad (17.7.1)$$

where V_e is the relative volume, E_e is the internal energy, and the constants A_e, B_e, ω_e, $R1_e$, and $R2_e$ are input constants. Similarly, the pressure in the reaction products is defined by another JWL form

$$P_p = A_p\left(1-\frac{\omega_p}{R1_p V_p}\right)e^{-R1_p V_p} + B_p\left(1-\frac{\omega_e}{R2_p V_p}\right)e^{-R2_p V_p} + \frac{\omega_p E}{V_p} \qquad (17.7.2)$$

The mixture of unreacted explosive and reaction products is defined by the fraction reacted F(F = 0 implies no reaction, F = 1 implies complete conversion from explosive to products). The pressures and temperature are assumed to be in equilibrium, and the relative volumes are assumed to be additive:

$$V = (1–F)\, V_e + FV_p \qquad (17.7.3)$$

The rate of reaction is defined as

$$\frac{\partial F}{\partial t} = I\,(FCRIT–F)^y\left(V_e^{-1}-1\right)^3\left[1+G\left(V_e^{-1}-1\right)\right] + H\,(1–F)^y F^x P^z\left(V_p^{-1}-1\right)^m \qquad (17.7.4)$$

where I, G, H, x, y, z, and m (generally m = 0) are input constants.

The JWL equations of state and the reaction rates have been fitted to one- and two-dimensional shock initiation and detonation data for four explosives: PBX-9404, RX-03-BB, PETN, and cast TNT. The details of calculational method are described by Cochran and Chan [1979]. The detailed one-dimensional calculations and parameters for the four explosives are given by Lee and Tarver [1980]. Two-dimensional calculations with this model for PBX 9404 and LX-17 are discussed by Tarver and Hallquist [1981].

17.8 Equation of State Form 8: Tabulated Compaction

The tabulated compaction model is linear in internal energy. Pressure is defined by

$$p = C(\varepsilon_V) + \gamma T(\varepsilon_V)E \qquad (17.8.1)$$

in the loading phase. The volumetric strain, ε_V, is given by the natural logarithm of the relative volume. Unloading occurs along the unloading bulk modulus to the pressure cutoff. Reloading always follows the unloading path to the point where unloading began, and continues on the loading path (Figure 17.1). Up to ten points and as few as two may be used when defining the tabulated functions, and the pressure is extrapolated if necessary.

17.9 Equation of State Form 9: Tabulated

The tabulated equation of state model is linear in internal energy. Pressure is defined by

$$p = C(\varepsilon_V) + \gamma T(\varepsilon_V)E \qquad (17.9.1)$$

The volumetric strain ε_V is given by the natural algorithm of the relative volume. Up to 10 points and as few as 2 may be used when defining the tabulated functions. The pressure is extrapolated if necessary. Loading and unloading are along the same curve unlike equation of state form 8.

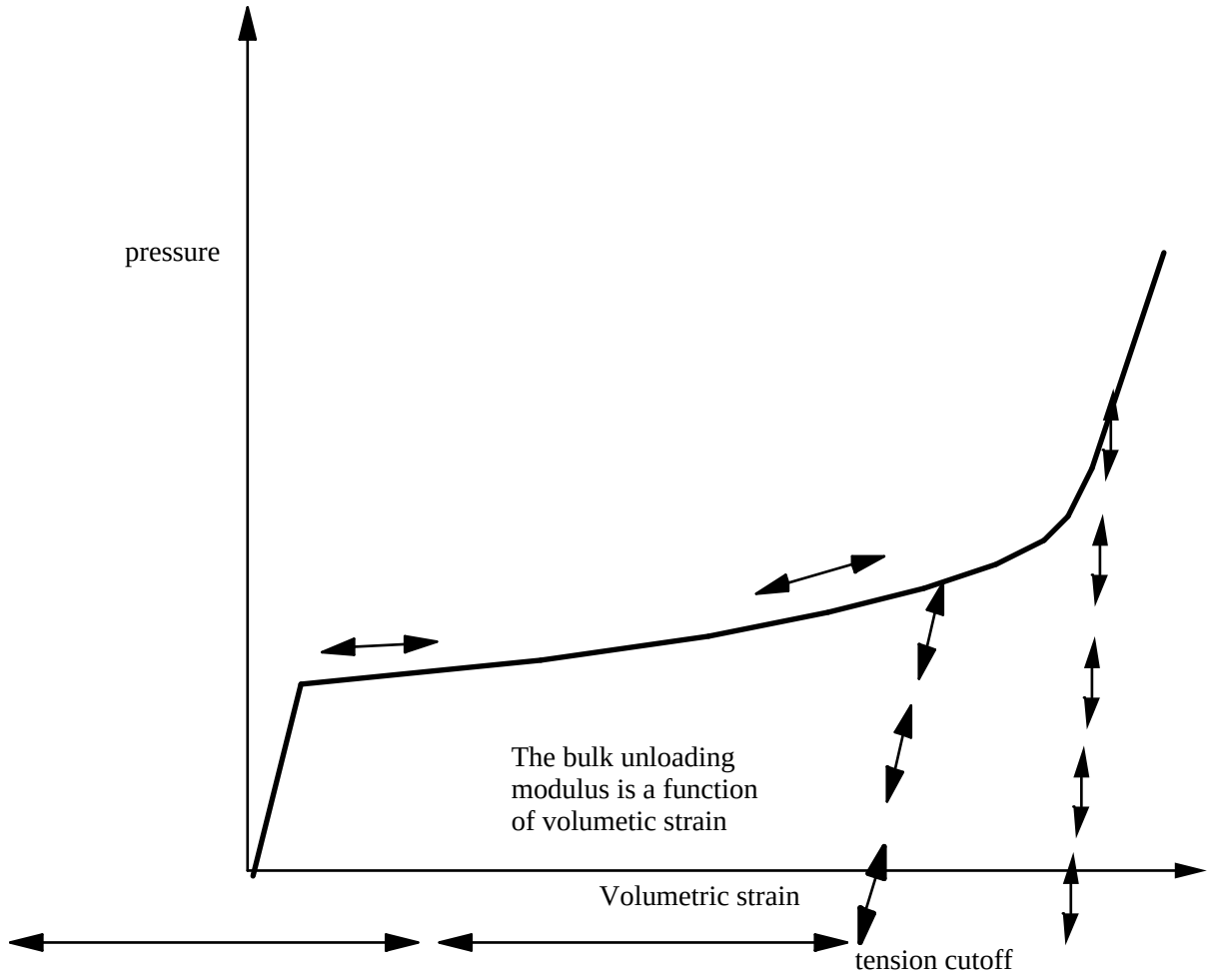


Figure 17.1. Pressure versus volumetric strain curve for equation of state form 8 with compaction. In the compacted states, the bulk unloading modulus depends on the peak volumetric strain.

18. ARTIFICIAL BULK VISCOSITY

Bulk viscosity is used to treat shock waves. Proposed in one spatial dimension by von Neumann and Richtmyer [1950], the bulk viscosity method is now used in nearly all wave propagation codes. A viscous term q is added to the pressure to smear the shock discontinuities into rapidly varying but continuous transition regions. With this method the solution is unperturbed away from a shock, the Hugoniot jump conditions remain valid across the shock transition, and shocks are treated automatically. In our discussion of bulk viscosity we draw heavily on works by Richtmyer and Morton [1967], Noh [1976], and Wilkins [1980]. The following discussion of the bulk viscosity applies to solid elements since strong shocks are not normally encountered in structures modeled with shell and beam elements.

18.1 Shock Waves

Shock waves result from the property that sound speed increases with increasing pressure. A smooth pressure wave can gradually steepen until it propagates as a discontinuous disturbance called a shock. Shocks lead to jumps in pressure, density, particle velocity, and energy.

Consider a planar shock front moving through a material. Mass, momentum, and energy are conserved across the front. Application of these conservation laws leads to the well known Rankine-Hugoniot jump conditions

$$u_s = \frac{\rho(u - u_0)}{\rho - \rho_0} \tag{18.1a}$$

$$\rho - \rho_0 = \rho_0 u_s (u - u_0) \tag{18.1b}$$

$$e - e_0 = \frac{p - p_0}{2} \frac{\rho - \rho_0}{\rho_0 \rho} \tag{18.1c}$$

where Equation (18.1c) is an expression of the energy jump condition using the results of mass conservation, Equation (18.1a), and momentum conservation, Equation (18.1b). Here u_s is the shock velocity, u is the particle velocity, ρ is the density, e is the specific internal energy, p is the pressure, and the subscript 0 indicates the state ahead of the shock.

The energy equation relating the thermodynamic quantities density, pressure, and energy must be satisfied for all shocks. The equation of state

$$p = p(\rho,e) \tag{18.2}$$

which defines all equilibrium states that can exist in a material and relating the same quantities as the energy equation, must also be satisfied. We may use this equation to eliminate energy from Equation (18.1c) and obtain a unique relationship between pressure and compression. This relation, called the Hugoniot, determines all pressure-compression states achievable behind the shock. Shocking takes place along the Rayleigh line and not the Hugoniot (Figure 18.1) and because the Hugoniot curve closely approximates an isentrope, we may often assume the unloading follows the Hugoniot. Combining Equations (18.1a) and (18.1b), we see that the slope of the Rayleigh line is related to the shock speed:

$$u_s = \frac{1}{\rho_0}\left[\frac{p_1 - p_0}{\frac{1}{\rho_0} - \frac{1}{\rho}}\right]^{1/2} \tag{18.3}$$

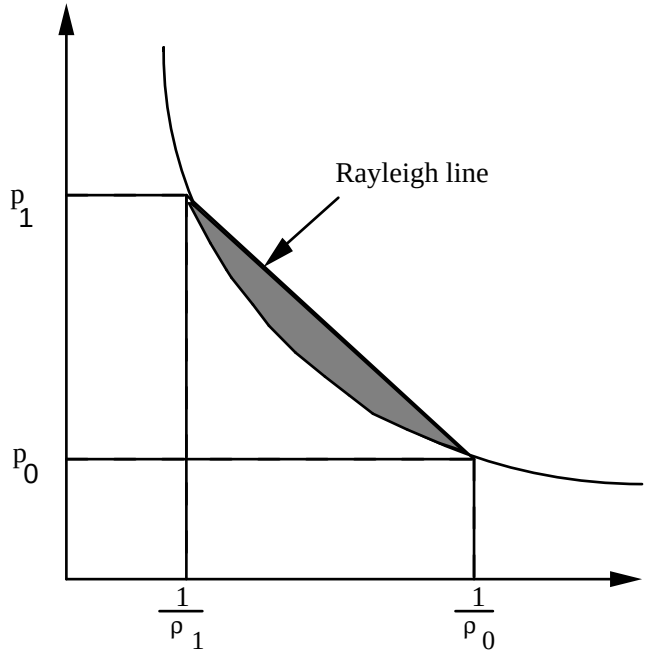


Figure 18.1. Shocking takes place along the Rayleigh line, and release closely follows the Hugoniot. The cross-hatched area is the difference between the internal energy behind the shock and the internal energy lost on release.

For the material of Figure 18.1, increasing pressure increases shock speed.

Consider a γ-law gas with the equation of state

$$p \quad (\gamma - 1)\rho e \tag{18.4}$$

where γ is the ratio of specific heats. Using the energy jump condition, we can eliminate e and obtain the Hugoniot

$$\frac{p}{p_0} = p^* = \frac{2V_0 + (\gamma - 1)(V_0 - V)}{2V - (\gamma - 1)(V_0 - V)} \tag{18.5}$$

where V is the relative volume. Figure 18.2 shows a plot of the Hugoniot and adiabat where it is noted that for p* = 1 the slopes are equal. Thus for weak shocks, Hugoniot and adiabat agree to the first order and can be ignored in numerical calculations. However, special treatment is require for strong shocks, and in numerical calculations this special treatment takes the form of bulk viscosity.

18.2 Bulk Viscosity

In the presence of shocks, the governing partial differential equations can give multiple weak solutions. In their discussion of the Rankine-Hugoniot jump conditions, Richtmyer and Morton [1967] report that the unmodified finite difference (element) equations often will not produce even approximately correct answers. One possible solution is to employ shock fitting techniques and treat the shocks as interior boundary conditions. This technique has been used in one spatial dimension but is too complex to extend to multi-dimensional problems of arbitrary geometry. For these reasons the pseudo-viscosity method was a major breakthrough for computational mechanics.

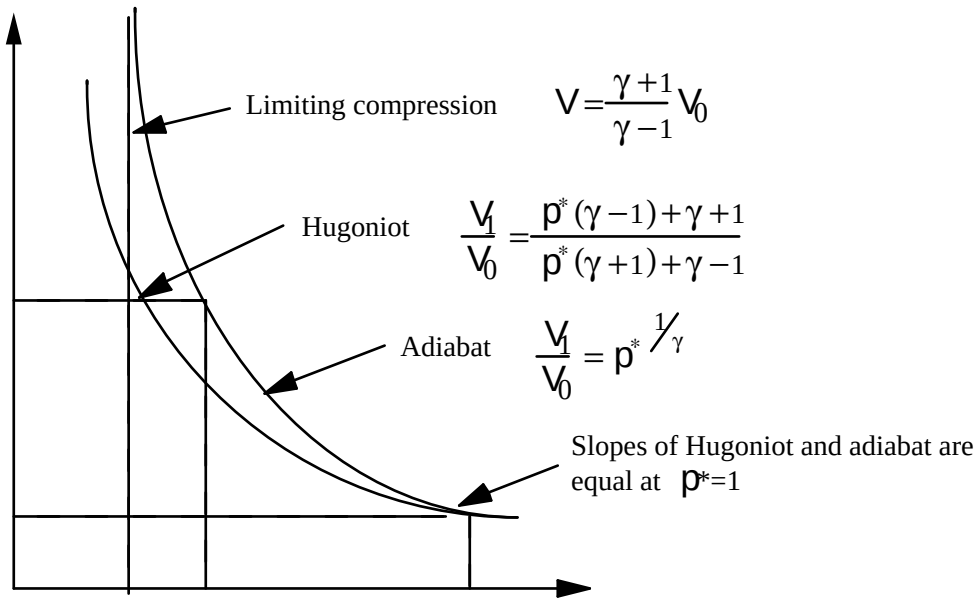


Figure 18.2 Hugoniot curve and adiabat for a g-law gas (from [Noh 1976]).

The viscosity proposed by von Neumann and Richtmyer [1950]in one spatial dimension has the form

$$q = C_0\rho(\Delta x)^2\left(\frac{\partial \dot{x}}{\partial x}\right)^2 \qquad \text{if } \frac{\partial \dot{x}}{\partial x} < 0$$

$$q = 0 \qquad \text{if } \frac{\partial \dot{x}}{\partial x} \geq 0 \tag{18.6}$$

where C_0 is a dimensionless constant and q is added to the pressure in both the momentum and energy equations. When q is used, they proved the following for steady state shocks: the hydrodynamic equations possess solutions without discontinuities; the shock thickness is independent of shock strength and of the same order as the Δx used in the calculations; the q term is insignificant outside the shock layer; and the jump conditions are satisfied. According to Noh, it is generally believed that these properties:

> "hold for all shocks, and this has been borne out over the years by countless numerical experiments in which excellent agreement has been obtained either with exact solutions or with hydrodynamical experiments."

In 1955 Landshoff [1955] suggested the addition of a linear term to the q of von Neumann and Richtmyer leading to a q of the form

$$q = C_0 \rho \Delta x^2 \left(\frac{\partial \dot{x}}{\partial x} \right)^2 - C_l \rho \Delta x a \frac{\partial \dot{x}}{\partial x} \qquad \text{if } \frac{\partial \dot{x}}{\partial x} < 0$$

$$q = 0 \qquad \text{if } \frac{\partial \dot{x}}{\partial x} \geq 0 \tag{18.7}$$

where C_l is a dimensionless constant and a is the local sound speed. The linear term rapidly damps numerical oscillations behind the shock front (Figure 18.3). A similar form was proposed independently by Noh about the same time.

In an interesting aside, Wilkins [1980] discusses work by Kuropatenko who, given an equation of state, derived a q by solving the jump conditions for pressure in terms of a change in the particle velocity, Δu. For an equation of state of the form

$$p = K\left(\frac{\rho}{\rho_0} - 1 \right) \tag{18.8}$$

pressure across the shock front is given by [Wilkins 1980]

$$p = p_0 + \rho_0 \frac{(\Delta u)^2}{2} + \rho_0 |\Delta u| \left[\frac{(\Delta u)^2}{2} + a^2 \right]^{1/2} \tag{18.9}$$

where a is a sound speed

$$a = \left(\frac{K}{\rho_0} \right)^{1/2} \tag{18.10}$$

For a strong shock, $\Delta u^2 >> a^2$, we obtain the quadratic form

$$q = \rho_0 \Delta u^2 \tag{18.11}$$

and for a weak shock, $\Delta u^2 << a^2$, the linear form

$$q = \rho_0 a \Delta u \tag{18.12}$$

Thus linear and quadratic forms for q can be naturally derived. According to Wilkins, the particular expressions for q obtained by Kuropatenko offer no particular advantage over the expressions currently used in most computer programs.

In extending the one-dimensional viscosity formulations to multi-dimensions, most code developers have simply replaced the divergence of the velocity with $\dot{\varepsilon}_{kk}$, the trace of the strain rate tensor, and the characteristic length with the square root of the area A, in two dimensions and the cubic root of the volume v in three dimensions. These changes also give the default viscosities in the LS-DYNA2D/3D codes:

$$q = \rho l\left(C_0 l\,\dot{\varepsilon}_{kk}^2 - C_l a\dot{\varepsilon}_{kk}\right) \quad \text{if } \dot{\varepsilon}_{kk} < 0$$

$$q = 0 \qquad \text{if } \dot{\varepsilon}_{kk} \geq 0 \tag{18.13}$$

where C_0 and C_1 are dimensionless constants which default to 1.5 and .06, respectively where $\ell = \sqrt{A}$ in 2D and $\sqrt[3]{v}$ in 3D, a is the local sound speed, C_0 defaults to 1.5 and C_ℓ defaults to .06.

In converging two- and three-dimensional geometries, the strain rate $\dot{\varepsilon}_{kk}$ is negative and the q term in Equation (18.13) is nonzero, even though no shocks may be generated. This results in nonphysical q heating. When the aspect ratios of the elements are poor (far from unity), the use of a characteristic length based on $\sqrt{A}$ or $\sqrt[3]{V}$ can also result in nonphysical q heating and even occasional numerical instabilities. Wilkins uses a q, based in part on earlier work by Richards [1965] that extends the von Neumann and Richtmyer formulations in a way that avoids these problems. This latter q may be added in the future if the need arises.

Wilkins' q is defined as:

$$q = C_0\,\rho\, l^2\left(\frac{ds}{dt}\right)^2 - C_l\rho\, l\, a^*\frac{ds}{dt} \quad \text{if } \dot{\varepsilon}_{kk} < 0$$

$$q = 0 \qquad \text{if } \frac{ds}{dt} \geq 0 \tag{18.14}$$

where l and $\frac{ds}{dt}$ are the thickness of the element and the strain rate in the direction of the acceleration, respectively, and a^* is the sound speed defined by $(p/\rho)^{1/2}$ if $p > 0$. We use the local sound speed in place of a^* to reduce the noise at the low stress levels that are typical of our applications.

Two disadvantages are associated with Equation (18.14). To compute the length parameter and the strain rate, we need to know the direction of the acceleration through the element. Since the nodal force vector becomes the acceleration vector in the explicit integration scheme, we have to provide extra storage to save the direction. In three dimensions our present storage capacity is marginal at best and sacrificing this storage for storing the direction would make it even more so. Secondly, we need to compute ℓ and $\frac{ds}{dt}$ which results in a noticeable increase in computer cost even in two dimensions. For most problems the additional refinement of Wilkins is not needed. However, users must be aware of the pitfalls of Equation (18.13), i.e., when the element aspect ratios are poor or the deformations are large, an anomalous q may be generated.

19. TIME STEP CONTROL

During the solution we loop through the elements to update the stresses and the right hand side force vector. We also determine a new time step size by taking the minimum value over all elements.

$$\Delta t^{n+1} = \text{amin}\ \{\Delta t_1, \Delta t_2, ..., \Delta t_N\} \tag{19.1}$$

where N is the number of elements. For stability reasons the scale factor α is typically set to a value of .90 (default) or some smaller value.

19.1 Time Step Calculations for Solid Elements

A critical time step size, Δt_e, is computed for solid elements from

$$\Delta t_e = \frac{L_e}{\left\{\left[Q + \left(Q^2 + c^2\right)^{1/2}\right]\right\}} \tag{19.2}$$

where Q is a function of the bulk viscosity coefficients C_0 and C_1:

$$Q = \begin{cases} C_1 c + C_0 L_e \left|\dot{\varepsilon}_{kk}\right| & \text{for } \dot{\varepsilon}_{kk} < 0 \\ 0 & \text{for } \dot{\varepsilon}_{kk} \geq 0 \end{cases} \tag{19.3}$$

L_e is a characteristic length:

8 node solids: $L_e = \dfrac{\upsilon_e}{A_{e\max}}$

4 node tetrahedras: L_e = minimum altitude

υ_e, is the element volume, $A_{e_{max}}$ is the area of the largest side, and c is the adiabatic sound speed:

$$c = \left[\frac{4G}{3\rho_0} + \frac{\partial p}{\partial \rho}\right)_s\Bigg]^{1/2} \tag{19.4}$$

where ρ is the specific mass density. Noting that:

$$\frac{\partial p}{\partial \rho}\bigg)_s = \frac{\partial p}{\partial \rho}\bigg)_{\mathrm{E}} + \frac{\partial p}{\partial \mathrm{E}}\bigg)_{\rho}\frac{\partial \mathrm{E}}{\partial \rho}\bigg)_s \tag{19.5}$$

and that along an isentrope the incremental energy, E, in the units of pressure is the product of pressure, p, and the incremental relative volume, dV:

$$d\mathrm{E} = -pdV \tag{19.6}$$

we obtain

$$c = \left[\frac{4G}{3\rho_0} + \frac{\partial p}{\partial \rho}\bigg)_{\mathrm{E}} + \frac{pV^{\,2}}{\rho_0}\frac{\partial p}{\partial \mathrm{E}}\bigg)_{\rho}\right]^{1/2} \tag{19.7}$$

For elastic materials with a constant bulk modulus the sound speed is given by:

$$c = \sqrt{\frac{E(1-\upsilon)}{(1+\upsilon)(1-2\upsilon)\rho}} \tag{19.8}$$

where E is Young's modulus, and υ is Poisson's ratio.

19.2 Time Step Calculations for Beam and Truss Elements

For the Hughes-Liu beam and truss elements, the time step size is give by:

$$\Delta t_e = \frac{L}{c} \tag{19.9}$$

where L is the length of the element and c is the wave speed:

$$c = \sqrt{\frac{E}{\rho}} \tag{19.10}$$

For the Belytschko beam the time step size given by the longitudinal sound speed is used (Equation (19.9)), unless the bending-related time step size given by [Belytschko and Tsay 1982]

$$\Delta t_e = \frac{.5L}{c\sqrt{3I\left[\frac{3}{12I + AL^2} + \frac{1}{AL^2}\right]}} \tag{19.11}$$

is smaller, where I and A are the maximum value of the moment of inertia and area of the cross section, respectively.

Comparison of critical time steps of the truss versus the elastic solid element shows that it if Poisson's ratio, υ, is nonzero the solid elements give a considerably smaller stable time step size. If we define the ratio, α, as:

$$\alpha = \frac{\Delta t_{continuum}}{\Delta t_{rod}} = \frac{C_{rod}}{C_{continuum}} = \sqrt{\frac{(1+\upsilon)(1-2\upsilon)}{1-\upsilon}}, \tag{19.12}$$

we obtain the results in Table 19.1 where we can see that as υ approaches .5 $\alpha \to 0$.

υ	0	0.2	0.3	0.4	0.45	0.49	0.50
α	1.	0.949	0.862	0.683	0.513	0.242	0.0

Table 19.1 Comparison of critical time step sizes for a truss versus a solid element.

19.3 Time Step Calculations for Shell Elements

For the shell elements, the time step size is given by:

$$\Delta t_e = \frac{L_s}{c} \tag{19.13}$$

where L_s is the characteristic length and c is the sound speed:

$$c = \sqrt{\frac{E}{\rho\left(1-\nu^2\right)}} \tag{19.14}$$

Three user options exists for choosing the characteristic length. In the default or first option the characteristic length is given by:

$$L_s = \frac{(1+\beta)A_s}{\max(L_1, L_2, L_3, (1-\beta)L_4)} \tag{19.15}$$

where $\beta = 0$ for quadrilateral and 1 for triangular shell elements, A_s is the area, and L_i, (i= 1…4) is the length of the sides defining the shell elements. In the second option a more conservative value of L_s is used:

$$L_s = \frac{(1+\beta)A_s}{\max(D_1, D_2)} \tag{19.16}$$

where D_i (i=1,2) is the length of the diagonals. The third option provides the largest time step size and is frequently used when triangular shell elements have very short altitudes. The bar wave speed, Equation (19.10), is used to compute the time step size and L_s is give by

$$L_s = \max\left[\frac{(1+\beta)A_s}{\max(L_1, L_2, L_3, (1-\beta)L_4)}, \min(L_1, L_2, L_3, L_4 + \beta 10^{20})\right] \tag{19.17}$$

A comparison of critical time steps of truss versus shells is given in Table 19.2 with β defined as:

$$\beta = \frac{\Delta t_{2D-continuum}}{\Delta t_{rod}} = \frac{C_{rod}}{C} = \sqrt{1-\upsilon^2} \tag{19.18}$$

υ	0	0.2	0.3	0.4	0.5
β	1.0	0.98	0.954	0.917	0.866

Table 19.2 Comparison of critical time step sizes for a truss versus a shell element.

19.4 Time Step Calculations for Solid Shell Elements

A critical time step size, Δt_e is computed for solid shell elements from

$$\Delta t_e = \frac{\upsilon_e}{cA_{e_{\max}}} \tag{19.19}$$

where υ_e is the element volume, $A_{e_{max}}$ is the area of the largest side, and c is the plane stress sound speed given in Equation (19.14).

19.5 Time Step Calculations for Discrete Elements

For spring elements such as that in Figure 19.1 there is no wave propagation speed c to calculate the critical time step size.

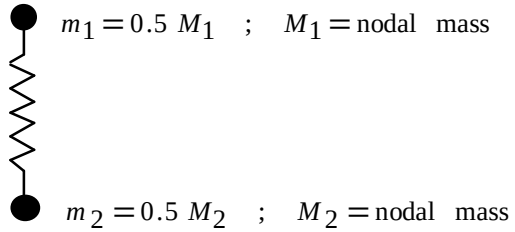


Figure 19.1 Lumped spring mass system.

The eigenvalue problem for the free vibration of spring with nodal masses m_1 and m_2, and stiffness, k, is given by:

$$\begin{bmatrix} k & -k \\ -k & k \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} - \omega^2 \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{19.20}$$

Since the determinant of the charactistic equation must equal zero, we can solve for the maximum eigenvalue:

$$\det \begin{bmatrix} k - \omega^2 m_1 & -k \\ -k & k - \omega^2 m_2 \end{bmatrix} = 0 \quad \rightarrow \quad \omega_{max}^2 = \frac{k(m_1 + m_2)}{m_1 \cdot m_2} \tag{19.21}$$

Recalling the critical time step of a truss element:

$$\left.\begin{aligned} \Delta t &\le \frac{\ell}{c} \\ \omega_{max} &= \frac{2c}{\ell} \end{aligned}\right\} \quad \Delta t \le \frac{2}{\omega_{max}} \tag{19.22}$$

and approximating the spring masses by using 1/2 the actual nodal mass, we obtain:

$$\Delta t = 2\sqrt{\frac{m_1 m_2}{m_1 + m_2} \frac{1}{k}} \tag{19.23}$$

Therefore, in terms of the nodal mass we can write the critical time step size as:

$$\Delta t_e = 2\sqrt{\frac{2M_1M_2}{k\left(M_1+M_2\right)}} \tag{19.24}$$

The springs used in the contact interface are not checked for stability.

20. BOUNDARY AND LOADING CONDITIONS

20.1 Pressure Boundary Conditions

Consider pressure loadings on boundary ∂b_1 in Equation (2.4). To carry out the surface integration indicated by the integral

$$\int_{\partial b_1} N^{t} t \, ds \tag{20.1}$$

a Gaussian quadrature rule is used. To locate any point of the surface under consideration, a position vector, r, is defined:

$$r = f_1(\xi,\eta)i_1 + f_2((\xi,\eta)i_2 + f_3(\xi,\eta)i_3 \tag{20.2}$$

where

$$f_i(\xi,\eta) = \sum_{j=1}^{4} \phi_j x_i^j \tag{20.3}$$

and $\mathbf{i}_1$, $\mathbf{i}_2$, $\mathbf{i}_3$ are unit vectors in the x_1, x_2, x_3 directions (see Figure 20.1).

Nodal quantities are interpolated over the four-node linear surface by the functions

$$\phi_i = \frac{1}{4}(1+\xi\xi_i)(1+\eta\eta_i) \tag{20.4}$$

so that the differential surface area ds may be written in terms of the curvilinear coordinates as

$$ds = |J| d\xi d\eta \tag{20.5}$$

where $|J|$ is the surface Jacobian defined by

$$|J| = \left|\frac{\partial r}{\partial \xi} \times \frac{\partial r}{\partial \eta}\right| = \left(EG - F^{2}\right)^{1/2} \tag{20.6}$$

in which

$$E = \frac{\partial r}{\partial \xi} \cdot \frac{\partial r}{\partial \xi}$$
$$F = \frac{\partial r}{\partial \xi} \cdot \frac{\partial r}{\partial \eta} \tag{20.7}$$
$$G = \frac{\partial r}{\partial \eta} \cdot \frac{\partial r}{\partial \eta}$$

A unit normal vector **n** to the surface segment is given by

$$n = |\mathrm{J}|^{-1} \left(\frac{\partial r}{\partial \xi} \times \frac{\partial r}{\partial \eta} \right) \tag{20.8}$$

and the global components of the traction vector can now be written

$$t_i = n_i \sum_{j=1}^{4} \phi_j p^j \tag{20.9}$$

where p_j is the applied pressure at the jth node.

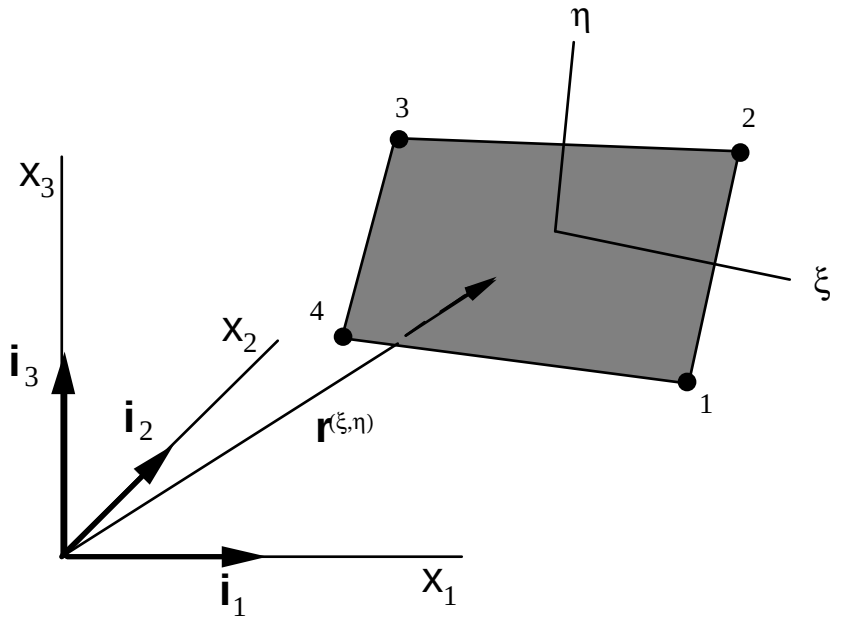


Figure 20.1. Parametric representation of a surface segment.

The surface integral for a segment is evaluated as:

$$\int_{-1}^{1}\int_{-1}^{1} N^t t |J| d\xi\, d\eta \tag{20.10}$$

One such integral is computed for each surface segment on which a pressure loading acts. Note that the Jacobians cancel when Equations (20.8) and (20.7) are put into Equation (20.10). Equation (20.10) is evaluated with one-point integration analogous to that employed in the volume integrals. The area of an element side is approximated by $4|J|$ where $|J| = |J(0, 0)|$.

20.2 Transmitting Boundaries

Transmitting boundaries are available only for problems that require the modeling of semi-infinite or infinite domains with solid elements and therefore are not available for beam or shell elements. Applications of this capability include problems in geomechanics and underwater structures.

The transmitting or silent boundary is defined by providing a complete list of boundary segments. In the approach used, discussed by Cohen and Jennings [1983] who in turn credit the method to Lysmer and Kuhlemeyer [1969], viscous normal shear stresses in Equation (20.11) are applied to the boundary segments:

$$\sigma_{normal} = -\rho c_d V_{normal} \tag{20.11a}$$

$$\sigma_{shear} = -\rho c_s V_{tan} \tag{20.11b}$$

where ρ, c_d, and c_s are the material density, dilatational wave speed, and the shear wave speed of the transmitting media respectively. The magnitude of these stresses is proportional to the particle velocities in the normal, v_{normal}, and tangential, v_{tan}, directions. The material associated with each transmitting segment is identified during initialization so that unique values of the constants ρ, c_d, and c_s can be defined automatically.

20.3 Kinematic Boundary Conditions

In this subsection, the kinematic constraints are briefly reviewed. LS-DYNA3D tracks reaction forces for each type of kinematic constraint and provides this information as output if requested. For the prescribed boundary conditions, the input energy is integrated and included in the external work.

20.3.1 Displacement Constraints

Translational and rotational boundary constraints are imposed either globally or locally by setting the constrained acceleration components to zero. If nodal single point constraints are employed, the constraints are imposed in a local system. The user defines the local system by specifying a vector u_l in the direction of the local x-axis x_l, and a local in-plane vector v_l. After normalizing u_l, the local x_l, y_l and z_l axes are given by:

$$x_l = \frac{u_l}{\|u_l\|} \tag{20.12}$$

$$z_l = \frac{x_l \times v_l}{\|x_l \times v_l\|} \tag{20.13}$$

$$y_l = z_l \times x_l \tag{20.14}$$

A transformation matrix q is constructed to transform the acceleration components to the local system:

$$q = \begin{bmatrix} x_l^t \\ y_l^t \\ z_l^t \end{bmatrix} \tag{20.15}$$

and the nodal translational and rotational acceleration vectors a_I and $\dot{\omega}_I$, for node I are transformed to the local system:

$$a_{I_l} = qa_I \tag{20.16a}$$

$$\dot{\omega}_{I_l} = q\dot{\omega}_I \tag{20.16b}$$

and the constrained components are zeroed. The modified vectors are then transformed back to the global system:

$$a_I = q^t a_{I_l} \tag{20.17a}$$

$$\dot{\omega}_I = q^t \dot{\omega}_{I_l} \tag{20.17b}$$

20.3.2 Prescribed Displacements, Velocities, and Accelerations

Prescribed displacements, velocities, and accelerations are treated in a nearly identical way to displacement constraints. After imposing the zero displacement constraints, the prescribed values are imposed as velocities at time, $t^{n+1/2}$. The acceleration versus time curve is integrated or the displacement versus time curve is differentiated to generate the velocity versus time curve. The prescribed nodal components are then set.

20.4 Body Force Loads

Body force loads are used in many applications. For example, in structural analysis the base accelerations can be applied in the simulation of earthquake loadings, the gun firing of projectiles, and gravitational loads. The latter is often used with dynamic relaxation to initialize the internal forces before proceeding with the transient response calculation. In aircraft engine design the body forces are generated by the application of an angular velocity of the spinning structure. The generalized body force loads are available if only part of the structure is subjected to such loadings, e.g., a bird striking a spinning fan blade.

For base accelerations and gravity we can fix the base and apply the loading as part of the body force loads element by element according to Equation (20.18)

$$f_{ebody} = \int_{\upsilon_m} \rho \mathrm{N}^t \, N a_{base} d\upsilon = m_e a_{base} \tag{20.18}$$

where a_{base} is the base acceleration and m_e is the element (lumped) mass matrix.

21. TIME INTEGRATION

21.1 Backgound

Consider the single degree of freedom damped system in Figure 21.1

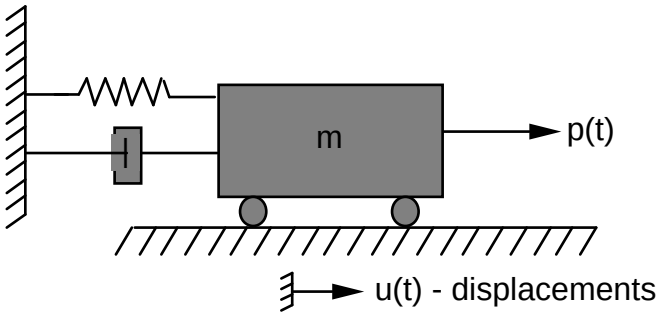


Figure 21.1 Single degree of freedom damped system.

Forces acting on mass m are shown in Figure 21.2.

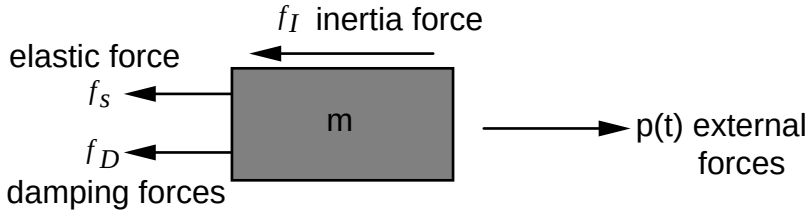


Figure 21.2 Forces acting of mass, m.

The equations of equilibrium are obtained from d'Alembert's principle

$$f_I + f_D + f_{\text{int}} = p(t) \;\; f_I + f_D + f_{\text{int}} = p(t) \tag{21.1}$$

$$\begin{aligned} f_I &= m\ddot{u} \quad ; \quad \ddot{u} = \frac{d^2u}{dt^2} && \text{acceleration} \\ f_D &= c\dot{u} \quad ; \quad \dot{u} = \frac{du}{dt} && \text{velocity} \\ f_{\text{int}} &= k \cdot u \quad ; \quad u && \text{displacement} \end{aligned} \tag{21.2}$$

where c is the damping coefficient, and k is the linear stiffness.

The equations of motion for linear behavior lead to a linear ordinary differential equation, o.d.e.:

$$m\ddot{u} + c\dot{u} + ku = p(t) \tag{21.3}$$

but for the nonlinear case the internal force varies as a nonlinear function of the displacement, leading to a nonlinear o.d.e:

$$m\ddot{u} + c\dot{u} + f_{\text{int}}(u) = p(t) \tag{21.4}$$

Analytical solutions of linear ordinary differential equations are available, so instead we consider the dynamic response of linear system subjected to a harmonic loading. It is convenient to define some commonly used terms:

Harmonic loading: $p(t) = p_0 \sin \varpi t$

Circular frequency: $\omega = \sqrt{\frac{k}{m}}$ for single degree of freedom

Natural frequency: $f = \frac{\omega}{2\pi} = \frac{1}{T}$ T = period (21.5)

Damping ratio: $\xi = \frac{c}{c_{cr}} = \frac{c}{2m\omega}$

Damped vibration frequency: $\omega_0 = \omega\sqrt{1-\xi^2}$

Applied load frequency: $\beta = \frac{\varpi}{\omega}$

The closed form solution is:

$$u(t) = \underbrace{u_0 \cos \omega t + \frac{\dot{u}_0}{\omega} \sin \omega t}_{\text{homogenous solution}} + \underbrace{\underbrace{\frac{p_0}{k} \frac{1}{1-\beta^2} (\sin \varpi t - \underbrace{\beta \sin \omega t}_{\text{transient}})}_{\text{steady state}}}_{\text{particular solution}} \tag{21.6}$$

with the initial conditions:

u_0 = initial displacement

$\dot{u}_0$ = initial velocity

$\frac{p_0}{k}$ = static displacement

For nonlinear problems, only numerical solutions are possible. LS-DYNA3D uses the explicit central difference scheme to integrate the equations of motion.

21.2 The Central Difference Method

The semi-discrete equations of motion at time n are:

$$Ma^n = P^n - F^n + H^n \tag{21.7}$$

where M is the diagonal mass matrix, P^n accounts for external and body force loads, F^n is the stress divergence vector, and H^n is the hourglass resistance. To advance to time t^{n+1} we use central difference time integration:

$$a^n = M^{-1}\,(P^n - F^n + H^n) \tag{21.8}$$

$$v^{n+\frac{1}{2}} = v^{n-\frac{1}{2}} + a^n \Delta t^n \tag{21.9}$$

$$u^{n+1} = u^n + v^{n+\frac{1}{2}} \Delta t^{n+\frac{1}{2}} \tag{21.10}$$

where

$$\Delta t^{n+\frac{1}{2}} = \frac{\left(\Delta t^n + \Delta t^{n+1}\right)}{2} \tag{21.11}$$

and **v** and **u** are the global nodal velocity and displacement vectors, respectively. We update the geometry by adding the displacement increments to the initial geometry:

$$x^{n+1} = x^0 + u^{n+1} \tag{21.12}$$

We have found that, although more storage is required to store the displacement vector the results are much less sensitive to round-off error.

21.3 Stability of Central Difference Scheme

The stability of the central difference scheme is determined by looking at the stability of a linear system. The system of linear equations in uncoupled into the modal equations where the modal matrix of eigenvectors, ϕ, are normalized with respect to the mass and linear stiffness matrices *K*, and *M*, respectively, such that:

$$\begin{aligned} \phi^T M\phi &= I \\ \phi^T K\phi &= \omega^2 \end{aligned} \tag{21.13}$$

With this normalization, we obtain for viscous proportional damping the decoupling of the damping matrix, *C*:

$$\phi^T C\phi = 2\xi\omega \tag{21.14}$$

The equations of motion in the modal coordinates *x* are:

$$\ddot{x} + 2\xi\omega\dot{x} + \omega^2 x = \underbrace{\phi^T p}_{=Y} \tag{21.15}$$

With central differences we obtain for the velocity and acceleration:

$$\dot{x}_n = \frac{x_{n+1} - x_{n-1}}{2\Delta t} \tag{21.16}$$

$$\ddot{x}_n = \frac{x_{n+1} - 2x_n + x_{n-1}}{\Delta t^2} \tag{21.17}$$

Substituting $\dot{x}_n$ and $\ddot{x}_n$ into equation of motion at time t^n leads to:

$$x_{n+1} = \frac{2-\omega^2\Delta t^2}{1+2\xi\omega\Delta t^2} x_n - \frac{1-2\xi\omega\Delta t}{1+2\xi\omega\Delta t} x_{n-1} + \frac{\Delta t^2}{1+2\xi\omega\Delta t^2} Y_n \tag{21.18}$$

$$x_n = x_n \tag{21.19}$$

which in matrix form leads to

$$\begin{bmatrix} x_{n+1} \\ x_n \end{bmatrix} = \begin{bmatrix} \frac{2-\omega^2\Delta t^2}{1+2\xi\omega\Delta t} & -\frac{1-2\xi\omega\Delta t}{1+2\xi\omega\Delta t} \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_n \\ x_{n-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta t^2}{1+2\xi\omega\Delta t^2} \\ 0 \end{bmatrix} Y_n \tag{21.20}$$

or

$$\hat{x}_{n+1} = \quad A \quad \hat{x}_n \quad + \quad L \quad Y_n \tag{21.21}$$

where, A is the time integration operator for discrete equations of motion. After m time steps with L=0 we obtain:

$$\hat{x}_m = A^m \hat{x}_0 \tag{21.22}$$

As m approaches infinity, A must remain bounded.

A spectral decomposition of A gives:

$$A^m = \left(P^T J P\right)^m = P^T J^m P \tag{21.23}$$

where, P, is the orthonormal matrix containing the eigenvectors of A, and J is the Jordan form with the eigenvalues on the diagonal. The spectral radius, $\rho(A)$, is the largest eigenvalue of A = max [diag. (J)]. We know that J^m, is bounded if and only if:

$$\left|\rho(A) \le 1\right| \tag{21.24}$$

Consider the eigenvalues of A for the undamped equation of motion

$$\mathrm{Det}\left[\begin{vmatrix} 2-\omega^2\Delta t^2 & -1 \\ 1 & 0 \end{vmatrix} - \lambda \begin{vmatrix} 1 & -1 \\ 1 & 0 \end{vmatrix}\right] = 0 \tag{21.25}$$

$$-\left(2-\omega^2\Delta t^2 - \lambda\right)\cdot\lambda + 1 = 0 \tag{21.26}$$

$$\lambda = \frac{2-\omega^2\Delta t^2}{2} \pm \sqrt{\frac{\left(2-\omega^2\Delta t^2\right)^2}{4} - 1} \tag{21.27}$$

The requirement that $\left|\lambda\right| \le 1$ leads to:

$$\Delta t \le \frac{2}{\omega_{\max}} \tag{21.28}$$

as the critical time step. For the damped equations of motion we obtain:

$$\Delta t \le \frac{2}{\omega_{\max}}\left(\sqrt{1+\xi^2} - \xi\right). \tag{21.29}$$

Thus, damping reduces the critical time step size. The time step size is bounded by the largest natural frequency of the structure which in turn is bounded by the highest frequency of any individual element in the finite element mesh.

21.4 Subcycling (Mixed Time Integration)

The time step size, Δt, is always limited by a single element in the finite element mesh. The idea behind subcycling is to sort elements based on their step size into groups whose step size is some even multiple of the smallest element step size, $2^{(n-1)}\Delta t$, for integer values of n greater than or equal to 1. For example, in Figure 21.3 the mesh on the right because of the thin row of elements is three times more expensive than the mesh on the left

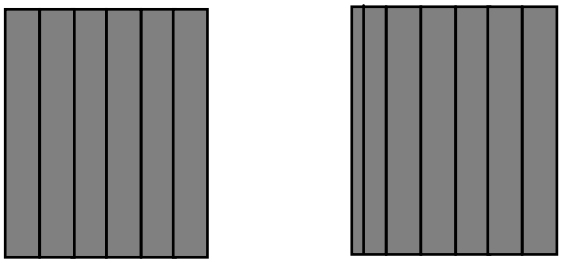


Figure 21.3. The righthand mesh is much more expensive to compute than the lefthand due to the presence of the thinner elements.

The subcycling in LS-DYNA3D is based on the linear nodal interpoation partition subcycling algorithm of Belytschko, Yen, and Mullen [1979], and Belytschko [1980]. In their implementation the steps are:

1. Assign each node, i, a time step size, Δt_j, according to:
   $\Delta t_i = \min \left(2/\omega_j \right)$ over all elements j, connected to node i
2. Assign each element, j, a time step size, Δt_j, according to:
   $\Delta t_j = \min (\Delta t_i)$ over all nodes, i, of element, j
3. Group elements into blocks by time step size for vectorization.

In LS-DYNA3D we desire to used constant length vectors as much as possible even if it means updating the large elements incrementally with the small time step size. We have found that doing this decreases costs and stability is unaffected.

Hulbert and Hughes [1988] reviewed seven subcycling algorithms in which the partitioning as either node or element based. The major differences within the two subcycling methods lie in how elements along the interface between large and small elements are handled, a subject which is beyond the scope of this theoretical manual. Nevertheless, they concluded that three of the methods including the linear nodal

interpolation method chosen for LS-DYNA3D, provide both stable and accurate solutions for the problems they studied. However, there was some concern about the lack of stability and accuracy proofs for any of these methods when applied to problems in structural mechanics.

The implementation of subcycling currently includes the following element classes and contact options:

- Solid elements
- Beam elements
- Shell elements
- Brick shell elements
- Penalty based contact algorithms.

but intentionally excludes discrete elements since these elements generally contribute insignificantly to the calculational costs. The interface stiffnesses used in the contact algorithms are based on the minimum value of the slave node or master segment stiffness and, consequently, the time step size determination for elements on either side of the interface is assumed to be decoupled; thus, caling penalty values to larger values when subcycling is active can be a dangerous exercise indeed. Nodes that are included in constaint equations, rigid bodies, or in contact with rigid walls are always assigned the smallest time step sizes.

To explain the implementation and the functioning of subcycling, we consider the beam shown in Figure 21.4 where the beam elements on the right (material group 2) have a Courant time step size exactly two times greater than the elements on the left. The nodes attached to material group 2 will be called minor cycle nodes and the rest, major cycle nodes. At time step n=mk all nodal displacements and element stresses are known, where m is the ratio between the largest and smallest time step size, k is the number of major time steps, and n is the number of minor time steps. In Figures 21.5 and 21.6, the update of the state variables to the next major time step k+1 is depicted. The stress state in the element on the material interface in group 1 is updated during the minor cycle as the displacement of the major cycle node shared by this element is assumed to vary linearly during the minor cycle update. This linear variation of the major cycle nodal displacements during the update of the element stresses improves accuracy and stablility.

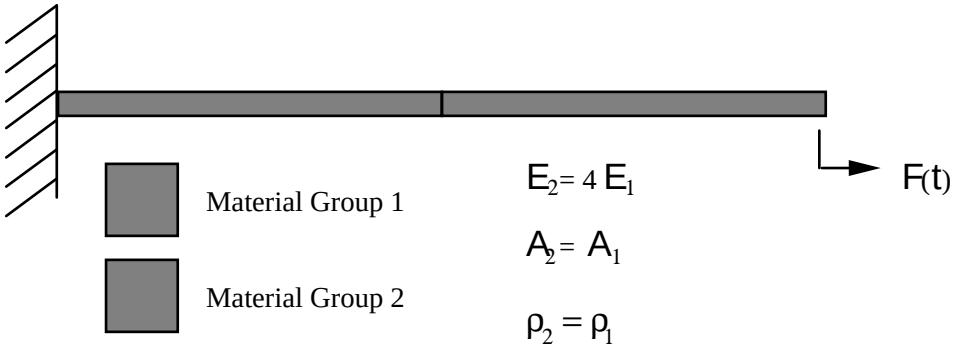


Figure 21.4. Subcycled beam problem from Hulbert and Hughes [1988].

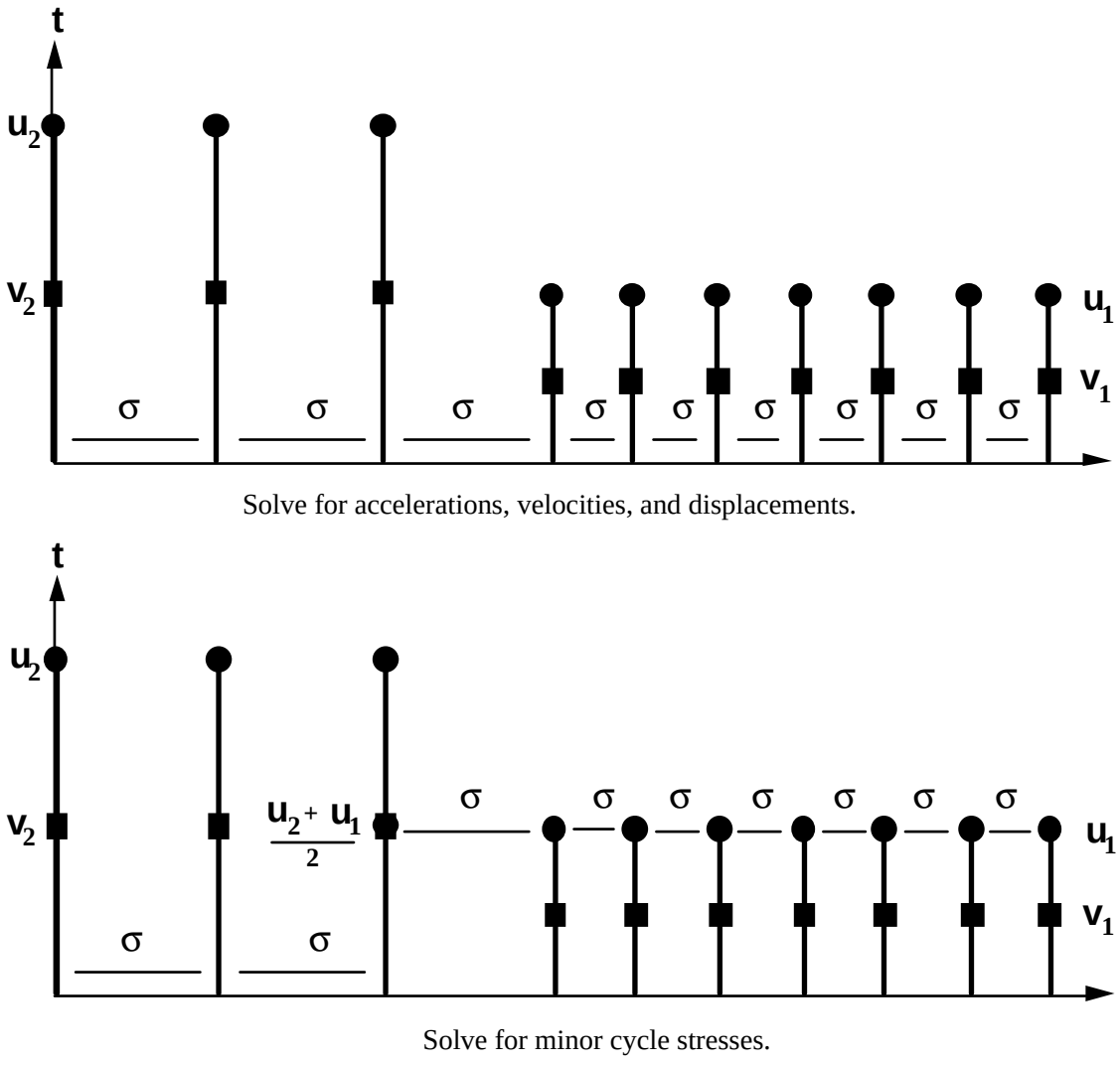
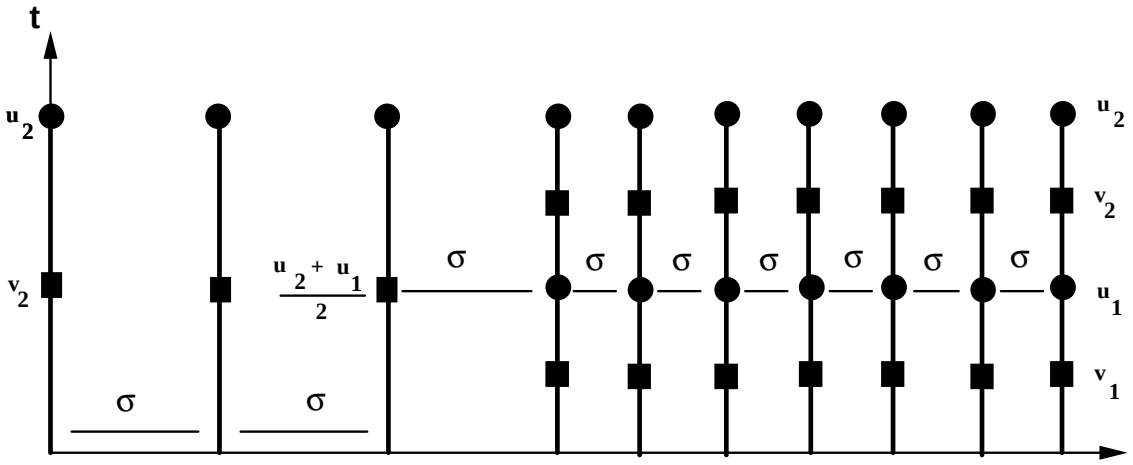
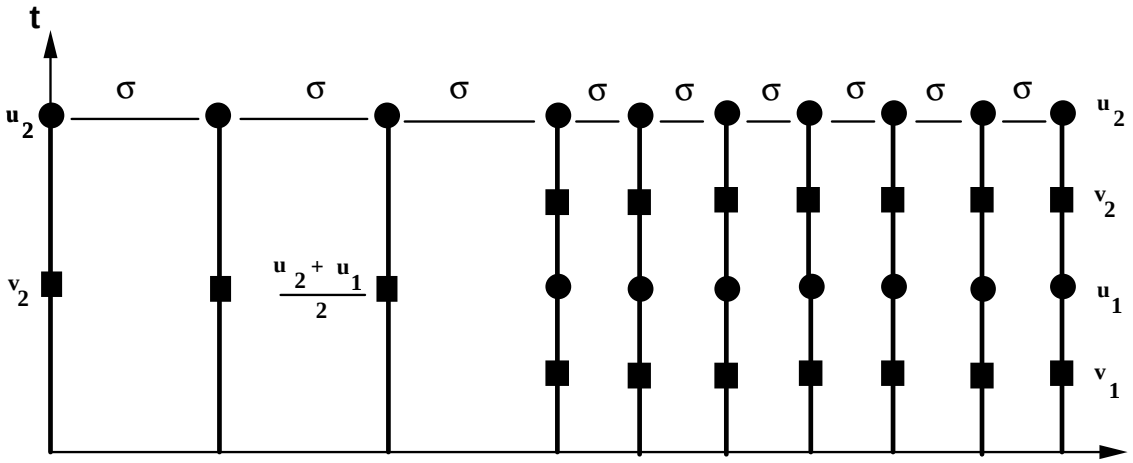


Figure 21.5. Timing diagram for subcycling algorithm based on linear nodal interpolations.



Solve for minor cycle accelerations, velocities, and displacements.



Update stresses for all elements.

Figure 21.6. Timing diagram for subcycling algorithm based on linear nodal interpolations.

In the timing study results in Table 21.1, fifty solid elements were included in each group for the beam depicted in Figure 21.4, and the ratio between E2 to E1 was varied from 1 to 128 giving a major time step size greater than 10 times the minor. Note that as the ratio between the major and minor time step sizes approaches infinity the reduction in cost should approach 50 percent for the subcycled case, and we see that this is indeed the case. The effect of subcycling for the more expensive fully integrated elements is greater as may be expected. The overhead of subcycling for the case where E1=E2 is relatively large. This provides some insight into why a decrease in speed is often observed when subcycling is active. For subcycling to have a significant effect, the ratio of the major to minor time step size should be large and the number of elements having the minor step size should be small. In crashworthiness applications the typical mesh is very well planned and generated to have uniform time step sizes; consequently, subcycling will probably give a net increase in costs.

Case 1 ONE POINT INTEGRATION WITH ELASTIC MATERIAL MODEL	Number of cycles	cpu time(secs)
E2= E1	178 178	4.65 5.36 (+15.%)
E2=4E1	367 367	7.57 7.13 (-6.0%)
E2=16E1	714 715	12.17 10.18 (-20.%)
E2=64E1	1417 1419	23.24 16.39 (-29.%)
E2=128E1	2003 2004	31.89 22.37 (-30.%)

Case 2 EIGHT POINT INTEGRATION WITH ORTHOTROPIC MATERIAL MODEL	Number of cycles	cpu time(secs)
E2= E1	180 180	22.09 22.75 (+3.0%)
E2=4E1	369 369	42.91 34.20 (-20.%)
E2=16E1	718 719	81.49 54.75 (-33.%)
E2=64E1	1424 1424	159.2 97.04 (-39.%)
E2=128E1	2034 2028	226.8 135.5 (-40.%)

Table 21.1 Timing study showing effects of the ratio of the major to minor time step size.

The impact of the subcycling implementation in the software has a very significant effect on the internal structure. The elements in LS-DYNA3D are now sorted three times

- By element number in ascending order.
- By material number for large vector blocks.
- By connectivity to insure disjointness for right hand side vectorization which is very important for efficiency.

Sorting by Δt, interact with the second and third sorts and can result in the creation of much smaller vector blocks and result in higher cost per element time step. During the simulation elements can continuously change in time step size and resorting may be required to maintain stability; consequently, we must check for this continuously. Sorting cost, though not high when spread over the entire calculation, can become a factor that results in higher overall cost if done too frequently especially if the factor, m, is relatively small and the ratio of small to large elements is large.

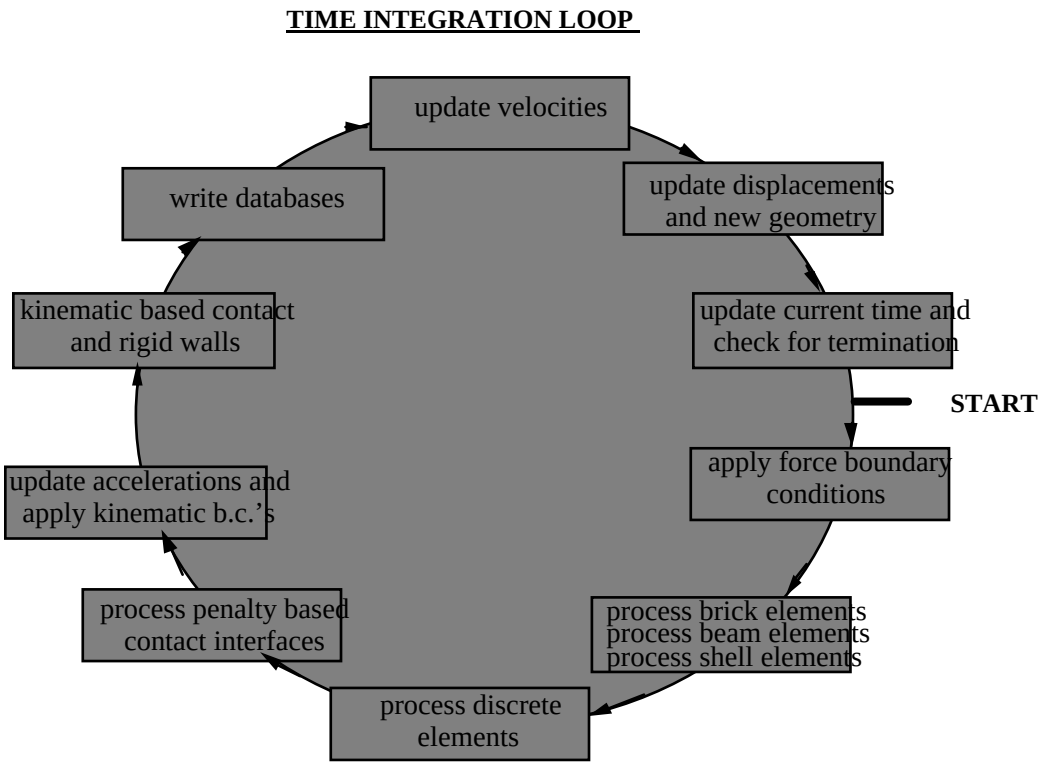


Figure 21.7. The time integration loop in LS-DYNA3D.

22. RIGID BODY DYNAMICS

A detailed discussion of the rigid body algorithm is presented by Benson and Hallquist [1986] and readers are referred to this publication for more information. The equations of motion for a rigid body are given by:

$$M\ddot{X}_i^{cm} = F_i^x \tag{22.1}$$

$$J_{ij}\dot{\omega}_j + e_{ijk}\omega_j J_{kl}\omega_l = F_i^{\omega} \tag{22.2}$$

where M is the diagonal mass matrix, J is the inertia tensor, X^{cm} is the location of the center of mass, ω is the angular velocity of the body, and F^x and F^{ω} are the generalized forces and torques. We can readily solve for the rigid body accelerations:

$$\ddot{X}_i^{cm} = \frac{F_i^x}{M} \tag{22.3}$$

$$\dot{\omega}_j = J_{ij}^{-1}\left[F_i^{\omega} - e_{ijk}\omega_j J_{kl}\omega_l\right] \tag{22.4}$$

There are three central issues associated with implementing Equations (22.1) and (22.2) in a structural dynamics program:

- calculating M and J from the mesh defining the body;
- calculating F^x and F^{ω}. This issue is especially critical when structures include rigid bodies contiguous with finite element.
- updating the displacements, velocities and inertia tensor in a manner that does not deform the rigid body.

A rigid body is defined using a finite element mesh by specifying that all of the elements in a region are rigid. In many of the equations that follow, summations are performed over all of the nodes associated with all of the elements in a rigid body, and these special summations are denoted $\sum_{\alpha}^{RB}$.

The rigid body mass is readily calculated from Equation (22.5):

$$M = \sum_{\alpha,\beta}^{RB} M_{\alpha 1 \beta 1} \tag{22.5}$$

where in explicit analyses the mass matrix is diagonal. The most popular mass lumping procedures scale the rotation masses to increase the allowable integration step size under the central difference stability criterion. Scaling should not be performed on the mass matrix entries associated with the rigid body nodes because the rigid body elements do not affect the step size and scaling reduces the accuracy of the calculated inertia tensor. For thin shells, the contributions to the inertia tensor from the rotational degrees of freedom are usually small and can be neglected. Thus, the inertia tensor is found by a nodal summation of the product of the point masses with their moment arms.

$$J_{ij} = e_{irs} e_{jnm} M_{\alpha r \beta n} \bar{x}_{\alpha s} \bar{x}_{\beta m} \tag{22.6}$$

where

$$\bar{x}_{\alpha i} = x_{\alpha i} - X_i^{cm} \tag{22.7}$$

The displacement of the rigid body is measured from its center of mass to eliminate the coupling between the translational and rotational momentum equations. Its location is initialized by calculating it from the mesh:

$$X_i^{cm} = \frac{\sum_{\alpha}^{RB} M_{\alpha i \beta i} x_{\beta i}}{M} \quad \text{(no sum on } i\text{)} \tag{22.8}$$

The initial velocities of the nodes are readily calculated for a rigid body from:

$$\dot{x}_{\alpha i} = \dot{X}_i^{cm} + e_{ijk} \omega_j A_{kn}(\theta) \bar{x}_{\alpha j} \tag{22.11}$$

where **A** the transformation from the rotated reference configuration to the global coordinate frame, θ the measure of the rotation of the body, and ω the angular velocity of the body in the global coordinate frame. At the initial time, **A** is the identity transformation. In deriving Equation (22.11), the reference configuration is assumed to be co-aligned with the global reference frame. For arbitrary orientations of the body, the inertia tensor must be transformed each time step based on the incremental rotations using the standard rules of second-order tensors:

$$J_{ij}^{n+1} = A(\Delta\theta)_{ik}\, A(\Delta\theta)_{jm}\, J_{km}^{n} \tag{22.12}$$

where $\mathbf{J}^{n+1}$ is the inertia tensor components in the global frame. The transformation matrix **A** is not stored since the formulation is incremental. The forces and torques are found by summing over the nodes:

$$F_i^x = \sum_{\alpha}^{RB} f_{\alpha i} \tag{22.13}$$

$$F_i^{\omega} = e_{ijk}\, \bar{x}_{\alpha j}\, f_{\alpha k} \tag{22.14}$$

The summations are performed over all of nodes in the rigid body, including the nodes on the boundary between the rigid body and a nonlinear finite element mesh. The summation automatically accounts for all of the forces (concentrated loads, gravity, impact forces, surface traction, etc.) including the interface forces between the rigid body and any contiguous mesh. It is the simplicity of the force and torque accumulations that makes rigid bodies so computationally attractive.

After calculating the rigid body accelerations from Equation (22.3) and (22.4), the rigid body velocities are updated as described in Section 21. The incremental rotation matrix of Equation (22.11) is calculated using the Hughes-Winget algorithm:

$$\Delta\theta_i^{(n+1)} \cong \Delta t^{\,n+1/2}\omega_i^{(n+1/2)} \tag{22.15}$$

$$A^{(n+1)}(\Delta\theta)_{ij} \cong \delta_{ij} + \left(\delta_{ik} - \frac{1}{2}\Delta S_{ik}\right)^{-1}\Delta S_{kj} = \delta_{ij} + \frac{1}{2D}\left(2\delta_{ik} + \Delta S_{ik}\right)\Delta S_{kj} \tag{22.16}$$

$$\Delta S_{ij} = e_{ikj}\Delta\theta_k^{(n+1)} \tag{22.17}$$

$$2D = 2 + \frac{1}{2}\left(\Delta\theta_1^2 + \Delta\theta_2^2 + \Delta\theta_3^2\right) \tag{22.18}$$

The coordinates of the nodes are incrementally updated

$$x_{\alpha i}^{(n+1)} = x_{\alpha i}^{(n)} + \left(X_i^{\,cm(n+1)} - X_i^{\,cm(n)}\right) + \left(A\left(\Delta\theta^{(n+1)}\right)_{ij} - \delta_{ij}\right)\bar{x}_{\alpha j}^{(n)} \tag{22.19}$$

where

$$\bar{x}^{(n)} = x^{(n)} - X^{cm(n)}$$

The velocities of the nodes are calculated by differencing the coordinates

$$\dot{x}_{\alpha i}^{(n+\frac{1}{2})} = \frac{\left(x_{\alpha i}^{(n+1)} - x_{\alpha i}^{(n)} \right)}{\Delta t^{n+\frac{1}{2}}} \tag{22.20}$$

A direct integration of the rigid body accelerations into velocity and displacements is not used for two reasons: (1) calculating the rigid body accelerations of the nodes is more expensive than the current algorithm, and (2) the second-order accuracy of the central difference integration method would introduce distortion into the rigid bodies. Since the accelerations are not needed within the program, they are calculated by a post-processor using a difference scheme similar to the above.

22.1 Deformable to Rigid Material Switching

Occasionally in practice, long duration, large rigid body motions arise that are prohibitively expensive to simulate if the elements in the model are deformable. Such a case could occur possibly in automotive rollover where the time duration of the rollover would dominate the cost relative to the post impact response that occurs much later.

To permit such simulations to be efficiently handled a capability to switch a subset of materials from deformable to rigid and back to deformable is available. In practice the suspension system and tires would remain deformable A flag is set in the input to let LS-DYNA3D know that all materials in the model have the potential to become rigid bodies sometime during the calculation. When this flag is set a cost penalty is incurred on vector machines since the blocking of materials in the element loops will be based on the part ID rather than the material type ID. Normally this cost is insignificant relative to the cost reduction due to this unique feature.

For rigid body switching to work properly the choice of the shell element formulation is critical. The Hughes-Liu elements cannot currently be used for two reasons. First, since these elements compute the strains from the rotations of the nodal fiber vectors from one cycle to the next, the nodal fiber vectors should be updated with the rigid body motions and this is not done. Secondly, the stresses are stored in the global system as opposed to the co-rotational system. Therefore, the stresses would also need to

be transformed with the rigid body motions or zeroed out. The co-rotational elements of Belytschko and co-workers do not reference nodal fibers for the strain computations and the stresses are stored in the co-rotational coordinate system which eleminates the need for the transformations; consequently, these elements can be safely used. The membrane elements and airbag elements are closely related to the Belytschko shells and can be safely used with the switching options.

The beam elements have nodal triads that are used to track the nodal rotations and to calculate the deformation displacements from one cycle to the next. These nodal triads are updated every cycle with the rigid body rotations to prevent non-physical behavior when the rigid body is switched back to deformable. This applies to all beam element formuations in LS-DYNA3D. The Belytschko beam formulations are preferred for the switching options for like the shell elements, the Hughes-Liu beams keep the stresses in the global system . Truss elements like the membrane elements are trivially treated and pose no difficulties.

The brick elements store the stresses in the global system and upon switching the rigid material to deformable the element stresses are zeroed to eliminate spurious behavior.

The implementation of the addresses many potential problems and has worked well in practice. The current restrictions can be eliminated if the need arises and anyway should pose no insurmountable problems. We will continue to improve this capability if we find that it is becoming a popular option.

23. CONTACT-IMPACT ALGORITHM

23.1 Introduction

The treatment of sliding and impact along interfaces has always been an important capability in the DYNA3D codes. Three distinct methods for handling this have been implemented which we will refer to as the kinematic constraint method, the penalty method and the distributed parameter method. Of these, the first approach is now used only for tying interfaces. The relative merits of each approach are discussed below.

Interfaces are defined in three dimensions by listing in arbitrary order all triangular and quadrilateral segments that comprise each side of the interface. One side of the interface is designated as the slave side, and the other is designated as the master side. Nodes lying in those surfaces are referred to here as slave and master nodes, respectively. In the penalty method, this distinction is irrelevant, but in the other methods the slave nodes are constrained to slide on the master surface after impact and must remain on the master surface until a tensile force develops between the node and the surface.

23.2 Kinematic Constraint Method

The kinematic constraint method which uses the impact and release conditions of Hughes et al.[1976] was implemented first in DYNA2D [Hallquist 1976b] and finally extended to three dimensions in DYNA3D. Constraints are imposed on the global equations by a transformation of the nodal displacement components of the slave nodes along the contact interface. This transformation has the effect of eliminating the normal degree of freedom of nodes. To preserve the efficiency of the explicit time integration, the mass is lumped to the extent that only the global degrees of freedom of each master node are coupled. Impact and release conditions are imposed to insure momentum conservation.

Problems arise with this method when the master surface zoning is finer than the slave surface zoning as shown in two dimension in Figure 23.1. Here certain master nodes can penetrate through the slave surface without resistance and create a kink in the slide line. Such kinks are relatively common with this formulation, and, when interface pressures are high, these kinks occur whether one or more quadrature points are used in the element integration. It may be argued, of course, that better zoning would minimize such problems; but for many problems that are of interest, good zoning in the initial configuration may be very poor zoning later. Such is the case, for example, when gaseous products of a high explosive gas expand against the surface of a structural member.

23.3 Penalty Method

The penalty method is used in the explicit programs DYNA2D and DYNA3D as well as in the implicit programs NIKE2D and NIKE3D. The method consists of placing normal interface springs between all penetrating nodes and the contact surface. With the exception of the spring stiffness matrix which must be assembled into the global stiffness matrix, the implicit and explicit treatments are similar. The NIKE2D/3D and DYNA2D/3D programs compute a unique modulus for the element in which it resides. In our opinion, pre-empting user control over this critical parameter greatly increases the success of the method.

Quite in contrast to the nodal constraint method, the penalty method approach is found to excite little if any mesh hourglassing. This lack of noise is undoubtedly attributable to the symmetry of the approach. Momentum is exactly conserved without the necessity of imposing impact and release conditions. Furthermore, no special treatment of intersecting interfaces is required, greatly simplifying the implementation.

The interface stiffness is chosen to be approximately the same order of magnitude as the stiffness of the interface element normal to the interface. Consequently the computed time step size is unaffected by the existence of the interfaces. However, if interface pressures become large, unacceptable penetration may occur. By scaling up the stiffness and scaling down the time step size, we may still solve such problems using the penalty approach. Since this increases the number of time steps and hence the cost, a sliding-only option has been implemented for treating explosive-structure interaction problems thereby avoiding use of the penalty approach. This latter option is based on a specialization of the third method described below.

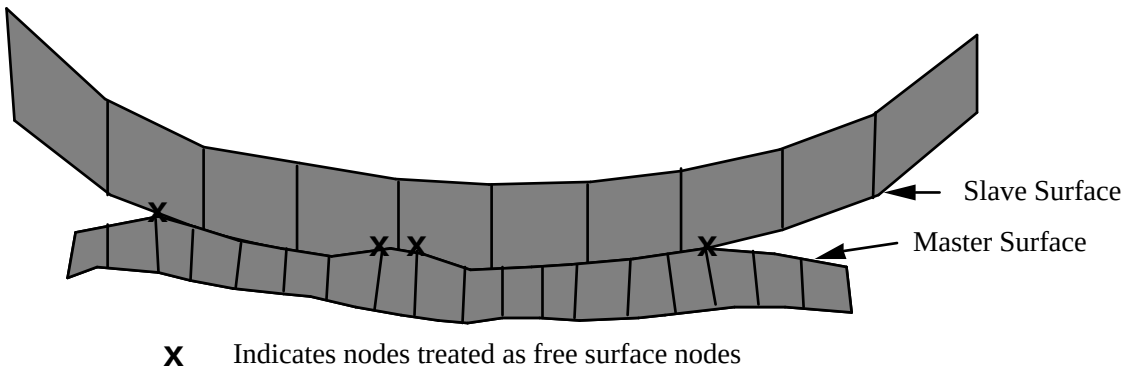


Figure 23.1. Nodes of the master slide surface designated with an “x” are treated as free surface node in the nodal constraint method.

23.4 Distributed Parameter Method

This method is used in DYNA2D, and a specialization of it is the sliding only option in DYNA3D. Motivation for this approach came from the TENSOR [Burton et al. 1982] and HEMP [Wilkins 1964] programs which displayed fewer mesh instabilities than DYNA2D with the nodal constraint algorithm. The first DYNA2D implementation of this last algorithm is described in detail by Hallquist [1978]. Since this early publication, the method has been moderately improved but the major ideas remain the same.

In the distributed parameter formulation, one-half the slave element mass of each element in contact is distributed to the covered master surface area. Also, the internal stress in each element determines a pressure distribution for the master surface area that receives the mass. After completing this distribution of mass and pressure, we can update the acceleration of the master surface. Constraints are then imposed on slave node accelerations and velocities to insure their movement along the master surface. Unlike the finite difference hydro programs, we do not allow slave nodes to penetrate; therefore we avoid "put back on" logic. In another simplification, our calculation of the slave element relative volume ignores any intrusion of the master surfaces. The HEMP and TENSOR codes consider the master surface in this calculation.

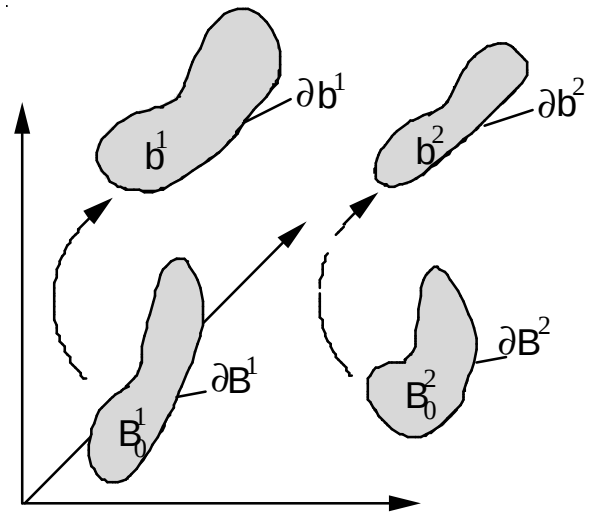


Figure 23.2. Reference and deformed configuration.

23.5 Preliminaries

Consider the time-dependent motion of two bodies occupying regions B^1 and B^2 in their undeformed configuration at time zero. Assume that the intersection

$$B^1 \cap B^2 = \emptyset \tag{23.1}$$

is satisfied. Let ∂B^1 and ∂B^2 denote the boundaries of B^1 and B^2, respectively. At some later time these bodies occupy regions b^1 and b^2 bounded by ∂b^1 and ∂b^2 as shown in Figure 23.2. Because the deformed configurations cannot penetrate,

$$(b^1 - \partial b^1) \cap b^2 = \emptyset \tag{23.2}$$

As long as $(\partial b^1 \cap \partial b^2) = \emptyset$, the equations of motion remain uncoupled. In the foregoing and following equations, the right superscript α (= 1, 2) denotes the body to which the quantity refers.

Before a detailed description of the theory is given, some additional statements should be made concerning the terminology. The surfaces ∂b^1 and ∂b^2 of the discretized bodies b^1 and b^2 become the master and slave surfaces respectively. Choice of the master and slave surfaces is arbitrary when the symmetric penalty treatment is employed. Otherwise, the more coarsely meshed surface should be chosen as the master surface unless there is a large difference in mass densities in which case the side corresponding to the material with the highest density is recommended. Nodal points that define ∂b^1 are called master nodes and nodes that define ∂b^2 are called slave nodes. When $\partial b^1 \cap \partial b^2 \neq \emptyset$ the constraints are imposed to prevent penetration. Right superscripts are implied whenever a variable refers to either the master surface ∂b^1, or slave surface, ∂b^2; consequently, these superscripts are dropped in the development which follows.

23.6 Slave Search

The slave search is common to all interface algorithms implemented in DYNA3D. This search finds for each slave node its nearest point on the master surface. Lines drawn from a slave node to its nearest point will be perpendicular to the master surface, unless the point lies along the intersection of two master segments, where a segment is defined to be a 3- or 4-node element of a surface.

Consider a slave node, n_s, sliding on a piecewise smooth master surface and assume that a search of the master surface has located the master node, m_s, lying nearest to n_s. Figure 23.3 depicts a portion of a master surface with nodes m_s and n_s labeled. If m_s and n_s do not coincide, n_s can usually be shown to lie in a segment s_i via the following tests:

$$(c_i \times s) \cdot (c_i \times c_{i+1}) > 0$$
$$(c_i \times s) \cdot (s \times c_{i+1}) > 0 \qquad (23.3)$$

where vector c_i and c_{i+1} are along edges of s_i and point outward from m_s. Vector **s** is the projection of the vector beginning at m_s, ending at n_s, and denoted by **g**, onto the plane being examined (see Figure 23.4).

$$s = g - (g \cdot m)m \qquad (23.4)$$

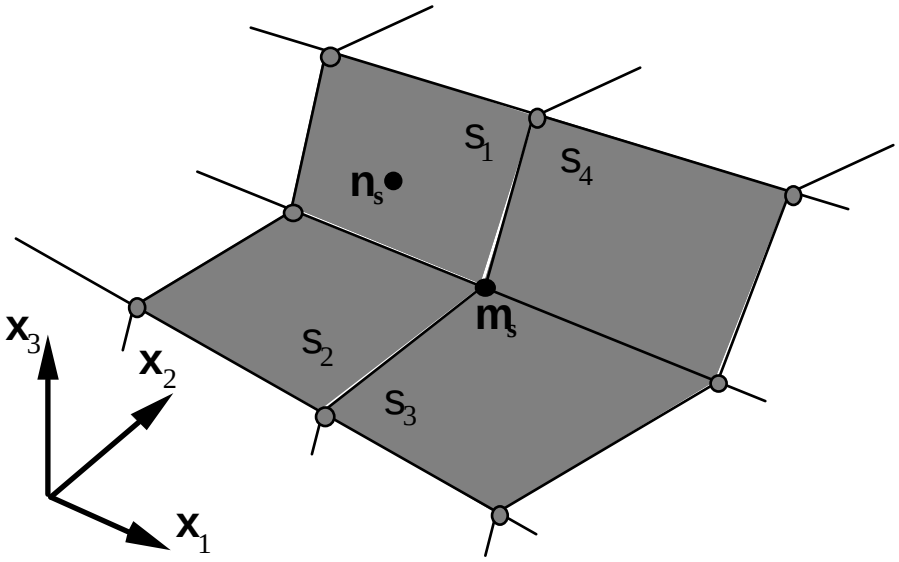


Figure 23.3. In this figure, four master segments can harbor slave node n_s given that m_s is the nearest master node.

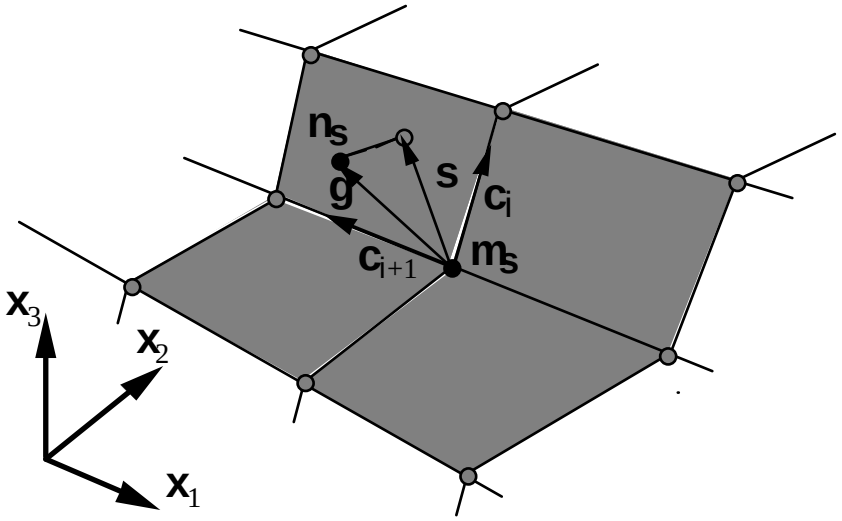


Figure 23.4. Projection of **g** onto master segment s_i .

where for segment s_i

$$m = \frac{c_i \times c_{i+1}}{|c_i \times c_{i+1}|} \tag{23.5}$$

Since the sliding constraints keep n_s close but not necessarily on the master surface and since n_s may lie near or even on the intersection of two master segments, the inequalities of Equations (23.3) may be inconclusive, i.e., they may fail to be satisfied or more than one may give positive results. When this occurs n_s is assumed to lie along the intersection which yields the maximum value for the quantity

$$\frac{g \cdot c_i}{|c_i|} \quad i = 1,2,3,4,... \tag{23.6}$$

When the contact surface is made up of badly shaped elements, the segment apparently identified as containing the slave node actually may not, as shown in Figure 23.5.

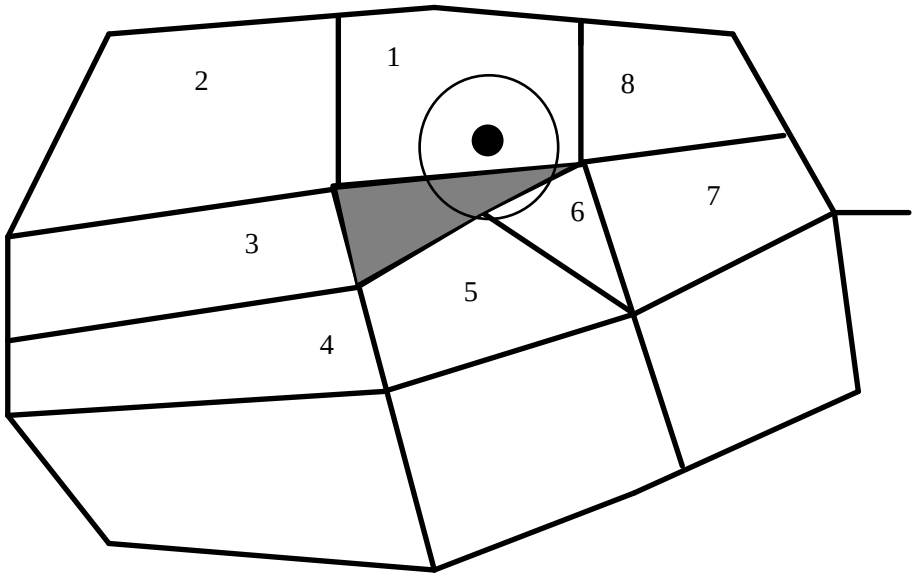


Figure 23.5. When the nearest node fails to contain the segment that harbors the slave node, segments numbered 1-8 are searched in the order shown.

Assume that a master segment has been located for slave node n_s and that n_s is not identified as lying on the intersection of two master segments. Then the identification of the contact point, defined as the point on the master segment which is nearest to n_s, becomes nontrivial. Each master surface segment, s_i is given the parametric representation of Equation (19.2), repeated here for clarity:

$$r = f_1(\xi,\eta)i_1 + f_2(\xi,\eta)i_2 + f_3(\xi,\eta)i_3 \tag{19.2}$$

where

$$f_i(\xi,\eta) = \sum_{j=1}^{4} \phi_j x_i^j \tag{19.3}$$

Note that r_i is at least once continuously differentiable and that

$$\frac{\partial r}{\partial \xi} \times \frac{\partial r}{\partial \eta} \neq 0 \tag{23.7}$$

Thus **r** represents a master segment that has a unique normal whose direction depends continuously on the points of s_i.

Let **t** be a position vector drawn to slave node n_s and assume that the master surface segment s_i has been identified with n_s. The contact point coordinates (ξ_c, η_c) on s_i must satisfy

$$\frac{\partial r}{\partial \xi}(\xi_c, \eta_c)\cdot\left[t - r(\xi_c, \eta_c)\right] = 0 \tag{23.8a}$$

$$\frac{\partial r}{\partial \eta}(\xi_c, \eta_c)\cdot\left[t - r(\xi_c, \eta_c)\right] = 0 \tag{23.8b}$$

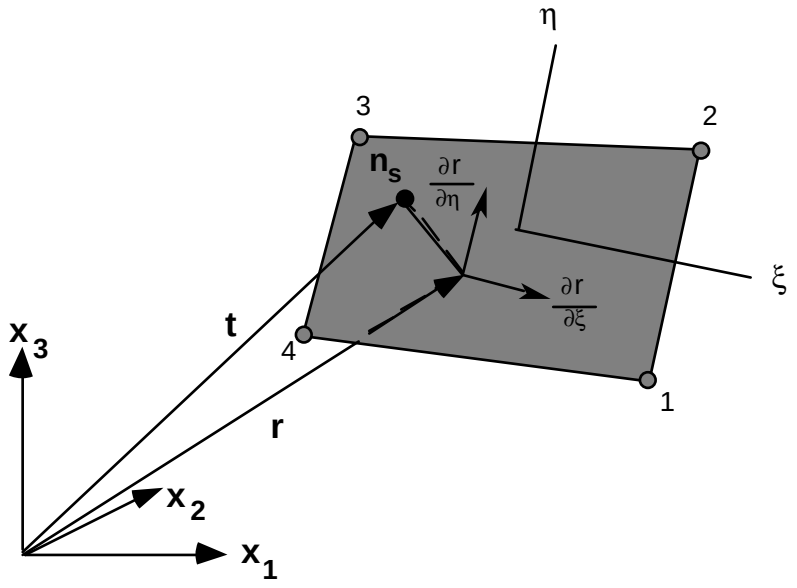


Figure 23.6. Location of contact point when n_s lies above master segment.

The physical problem is illustrated in Figure 23.6 which shows n_s lying above the master surface. Equations (23.8a) and (23.8b) are readily solved for ξ_c and η_c. One way to accomplish this is to solve Equation (23.8a) for ξ_c in terms of η_c, and substitute the results into Equation (23.8b). This yields a cubic equation in η_c which is presently solved numerically in LS-DYNA3D. In the near future we hope to implement a closed form solution for the contact point.

The equations are solved numerically. When two nodes of a bilinear quadrilateral are collapsed into a single node for a triangle, the Jacobian of the minimization problem is singular at the collapsed node. Fortunately, there is an analytical solution for triangular segments since three points define a plane. Newton-Raphson iteration is a natural choice for solving these simple nonlinear equations. The method diverges with distorted

elements unless the initial guess is accurate. An exact contact point calculation is critical in post-buckling calculations to prevent the solution from wandering away from the desired buckling mode.

Three iterations with a least-squares projection are used to generate an initial guess:

$$\xi_0 = 0, \quad \eta_0 = 0,$$

$$\begin{bmatrix} r_{,\xi} \\ r_{,\eta} \end{bmatrix} \begin{bmatrix} r_{,\xi} & r_{,\eta} \end{bmatrix} \begin{Bmatrix} \Delta\xi \\ \Delta\eta \end{Bmatrix} = \begin{bmatrix} r_{,\xi} \\ r_{,\eta} \end{bmatrix} \left\{ r\left(\xi_i, \eta_i\right) - t \right\}, \tag{23.9}$$

$$\xi_{i+1} = \xi_{i+1} + \Delta\xi, \quad \eta_{i+1} = \eta_i + \Delta\eta$$

followed by the Newton-Raphson iterations which are limited to ten iterations, but which usually converges in four or less.

$$[H] \begin{Bmatrix} \Delta\xi \\ \Delta\eta \end{Bmatrix} = - \begin{Bmatrix} r_{,\xi} \\ r_{,\eta} \end{Bmatrix} \left\{ r\left(\xi_i, \eta_i\right) - t \right\},$$

$$[H] = \begin{Bmatrix} r_{,\xi} \\ r_{,\eta} \end{Bmatrix} \begin{bmatrix} r_{,\xi} & r_{,\eta} \end{bmatrix} + \begin{bmatrix} 0 & r \cdot r_{,\xi\eta} \\ r \cdot r_{,\xi\eta} & 0 \end{bmatrix}, \tag{23.10}$$

$$\xi_{i+1} = \xi_i + \Delta\xi, \quad \eta_{i+1} = \eta_i + \Delta\eta.$$

In concave regions, a slave node may have isoparametric coordinates that lie outside of the [-1, +1] range for all of the master segments, yet still have penetrated the surface. A simple strategy is used for handling this case, but it can fail. The contact segment for each node is saved every time step. If the slave node contact point defined in terms of the isoparametric coordinates of the segment, is just outside of the segment, and the node penetrated the isoparametric surface, and no other segment associated with the nearest neighbor satisfies the inequality test, then the contact point is assumed to occur on the edge of the segment. In effect, the definition of the master segments are extended so that they overlap by a small amount. In the hydrocode literature, this approach is similar to the slide line extensions used in two dimensions. This simple procedure works well for most cases, but it can fail in situations involving sharp concave corners.

23.7 Sliding with Closure and Separation

Because this is perhaps the most general and most used interface algorithm, we choose to discuss it first. In applying this penalty method, each slave node is checked for penetration through the master surface. If the slave node does not penetrate, nothing is done. If it does penetrate, an interface force is applied between the slave node and its contact point. The magnitude of this force is proportional to the amount of penetration. This may be thought of as the addition of an interface spring.

Penetration of the slave node n_s through the master segment which contains its contact point is indicated if

$$l = n_i \cdot [t - r(\xi_c, \eta_c)] < 0 \tag{23.11}$$

where

$$n_i = n_i(\xi_c, \eta_c)$$

is normal to the master segment at the contact point.

If slave node n_s has penetrated through master segment s_i we add an interface force vector f_s :

$$\mathbf{f}_s = -l k_i \mathbf{n}_i \qquad \text{if } l < 0 \tag{23.12}$$

to the degrees of freedom corresponding to n_s and

$$f_m^i = \phi_i(\xi_c, \eta_c) f_s \qquad \text{if } l < 0 \tag{23.13}$$

to the four nodes (i = 1,2,3,4) that comprise master segment s_i. The stiffness factor k_i for master segment s_i is given in terms of the bulk modulus K_i, the volume V_i, and the face area A_i of the element that contains s_i as

$$k_i = \frac{f_{si} K_i A_i^2}{V_i} \tag{23.14}$$

where f_{si} is a scale factor for the interface stiffness and is normally defaulted to .10. Larger values may cause instabilities unless the time step size is scaled back in the time step calculation.

23.8 Recent Improvements in Surface-to-Surface Contact

A number of recent changes have been made in the surface-to-surface contact including contact searching, accounting for thickness, and contact damping. These changes have been implemented primarily to aid in the analysis of sheet metal forming problems.

23.8.1 Improvements to the Contact Searching

In metal forming applications, problems with the contact searching were found when the rigid body stamping dies were meshed with elements having very poor aspect ratios. The nearest node algorithm described above can break down since the nearest node is not necessarily anywhere near the segment that harbors the slave node as is assumed in Figure 23.5 (see Figure 23.7). Such distorted elements are commonly used in rigid bodies in order to define the geometry accurately.

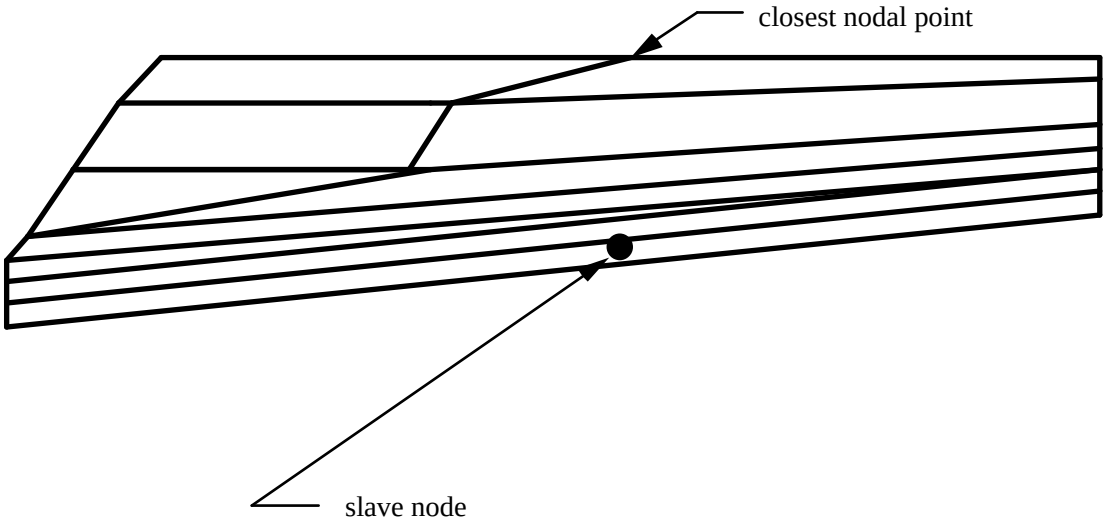


Figure 23.7. Failure to find the contact segment can be caused by poor aspect ratios in the finite element mesh.

To circumvent the problem caused by bad aspect ratios, an expanded searching procedure is used in which we attempt to locate the nearest segment rather than the nearest nodal point. We first sort the segments based on their centroids as shown in Figure 23.8 using a one-dimensional bucket sorting technique.

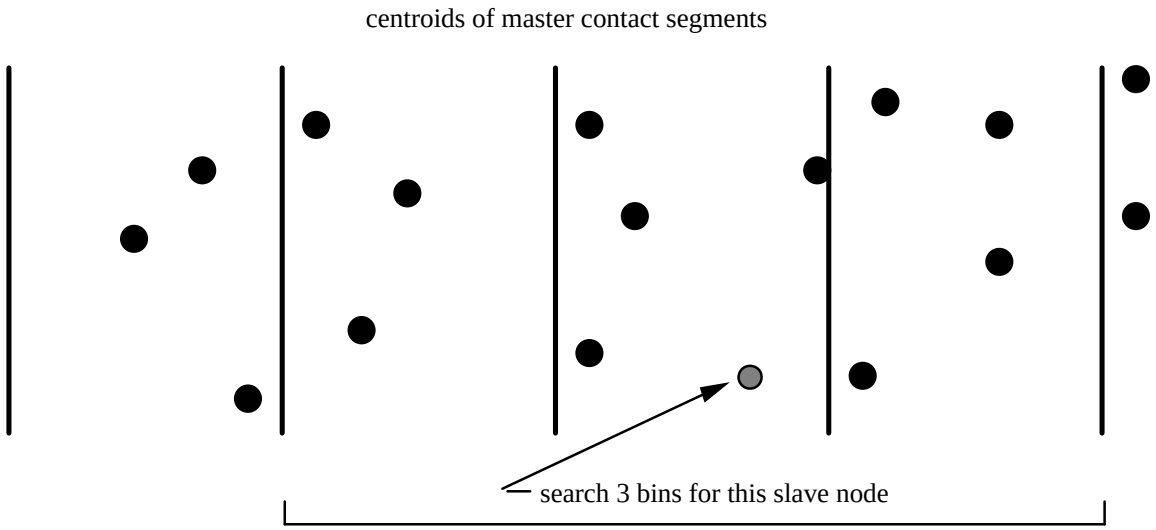


Figure 23.8 One-dimensional bucket sorting identifies the nearest segments for each slave node.

Once a list of possible candidates are identified for a slave node, it is necessary to locate the possible segments that contain the slave node of interest. For each quadrilateral segment, four points are constructed at the centroids of the four triangles each defined by 3 nodes as shown in Figure 23.9 where the black point is the centroid of the quadrilateral. These centroids are used to find the nearest point to the slave node and hence the nearest segment. The nodes of the three nearest segments are then examined to identify the three nearest nodes. Just one node from each segment is allowed to be a nearest node.

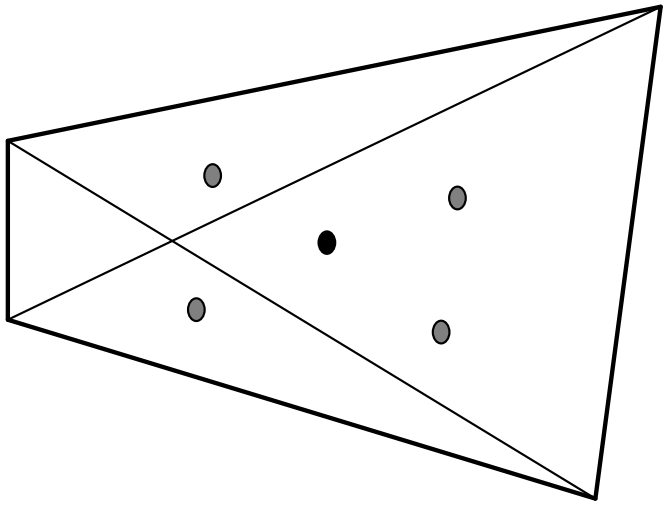


Figure 23.9 Interior points are constructed in the segments for determining the closest point to the slave node.

When the nearest segment fails to harbor the slave node, the adjacent segments are checked. The old algorithm checks the segments labeled 1-3 (Figure 23.10), which do not contain the slave node, and fails.

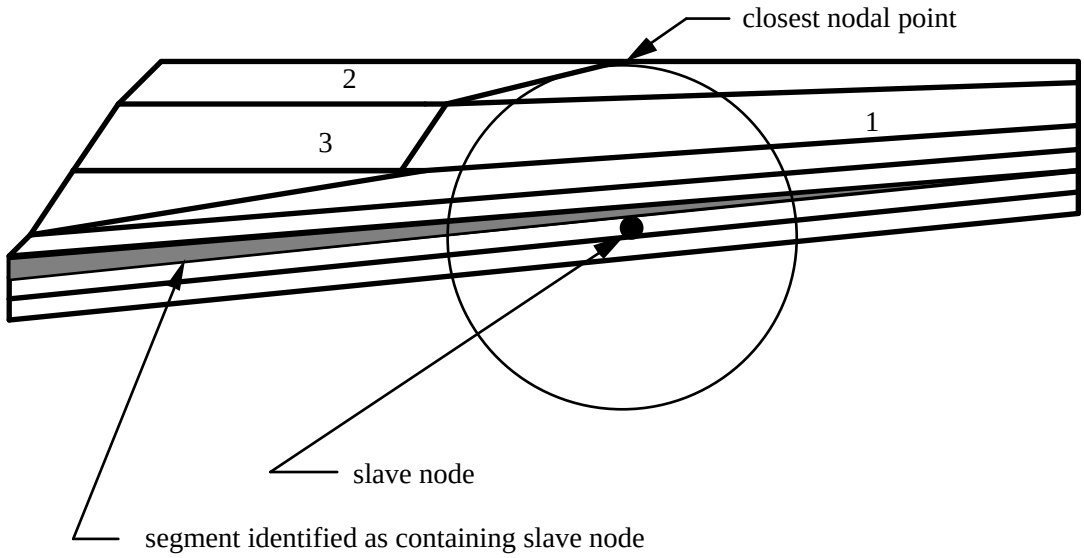


Figure 23.10. In case the stored segment fails to contain the node, the adjacent segments are checked.

23.8.2 Accounting for the Shell Thickness

Shell thickness effects are important when shell elements are used to model sheet metal. Unless thickness is considered in the contact, the effect of thinning on frictional interface stresses due to membrane stretching will be difficult to treat. In the treatment of thickness we project both the slave and master surfaces based on the mid-surface normal projection vectors as shown in Figure 23.11. The surfaces, therefore, must be offset by an amount equal to 1/2 their total thickness (Figure 23.12). This allows DYNA3D to check the node numbering of the segments automatically to ensure that the shells are properly oriented.

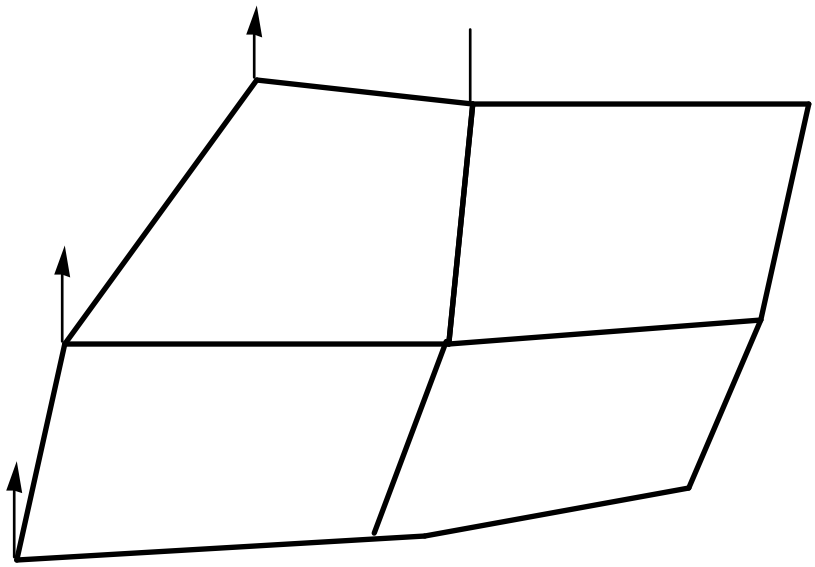


Figure 23.11. Contact surface is based on midsurface normal projection vectors.

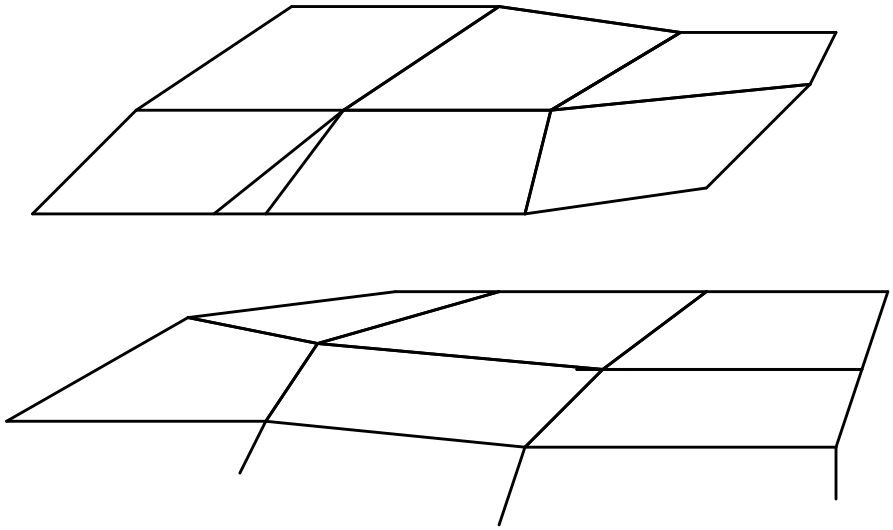


Figure 23.12. The slave and master surfaces must be offset in the input by one-half the total shell thickness. This also allows the segments to be oriented automatically.

Thickness changes in the contact are accounted for if and only if the shell thickness change option is flagged in the input. Each cycle, as the shell elements are

processed, the nodal thicknesses are stored for use in the contact algorithms. The interface stiffnesses may change with thickness depending on the input options used.

Type 5 contact considers nodes interacting with a surface. This algorithm calls exactly the same subroutines as surface-to-surface but not symmetrically: i.e., the subroutines are called once, not twice. To account for the nodal thickness, the maximum shell thickness of any shell connected to the node is taken as the nodal thickness and is updated every cycle. The projection of the node is done normal to the contact surface as shown in Figure 23.13.

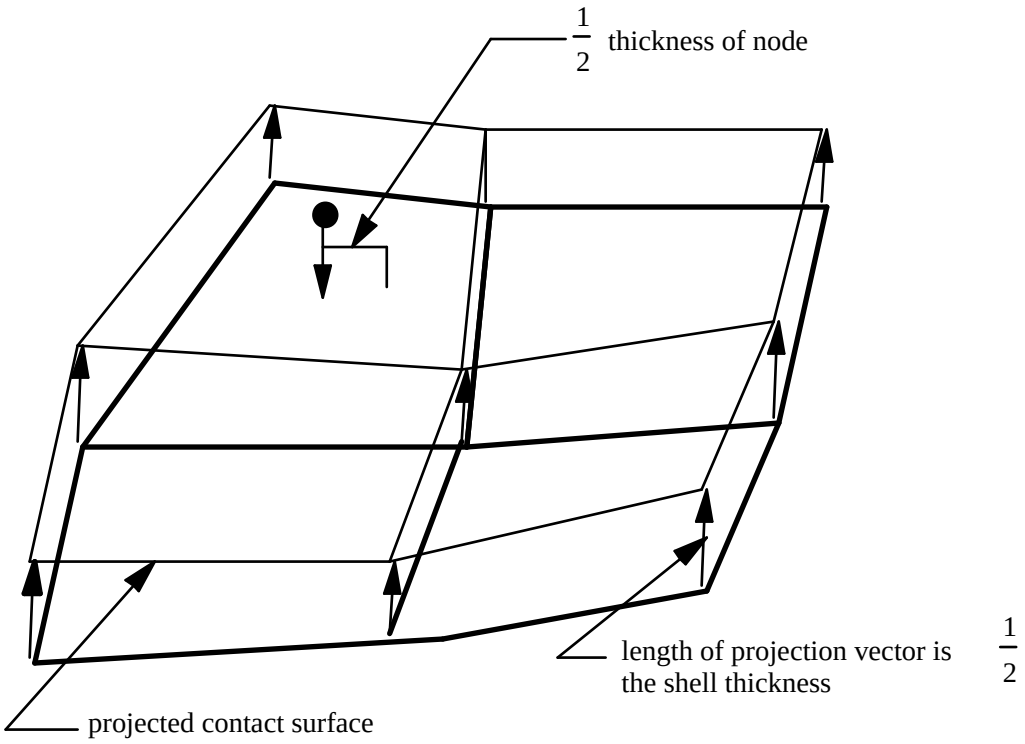


Figure 23.13 In a type 5 contact, thickness can also be taken into account.

23.8.3 Contact Damping

Viscous contact damping has been added to all contact options including single surface contact. The intention is to damp out oscillations normal to the contact surfaces during metal forming operations. The input requires a damping value as a percentage of critical, $2m\omega$, where m is the mass and ω is the natural frequency. Letting k denote the interface stiffness, we compute the natural frequency for the interface from Equation 23.15.

$$\omega = \sqrt{\frac{k\left(m_{slave} + m_{master}\right)}{m_{slave} m_{master}}} \quad m = \min\left\{m_{slave}, m_{master}\right\} \tag{23.15}$$

The master mass m_{master} is interpolated from the master nodes of the segment containing the slave node using the basis functions evaluated at the contact point.

23.8.4 Friction

Friction in LS-DYNA3D is based on a Coulomb formulation. Let f* be the trial force, f_n the normal force, k the interface stiffness, μ the coefficient of friction, and f^n the frictional force at time n. Then

$$F_y = \mu |f_n| \tag{23.16}$$

$$\Delta e = r^{n+1}\left(\xi_c^{n+1}, \eta_c^{n+1}\right) - r^{n+1}\left(\xi_c^{n}, \eta_c^{n}\right) \tag{23.17}$$

$$f^* = f^n - k\Delta e \tag{23.18}$$

$$f^{n+1} = f^* \qquad \text{if } |f^*| \le F_y \tag{23.19}$$

$$f^{n+1} = \frac{F_y f^*}{\left|f^*\right|} \quad \text{if } \left|f^*\right| > F_y \tag{23.20}$$

An exponential interpolation function smooths the transition between the static and dynamic coefficients of friction where v is the relative velocity between the slave node and the master segment:

$$\mu = \mu_d + (\mu_s - \mu_d)e^{-c|v|} \tag{23.21}$$

where

$$v = \frac{\Delta e}{\Delta t} \quad v = \frac{\Delta e}{\Delta t} \tag{23.22}$$

Δt is the time step size, and c is a decay constant.

The interface shear stress that develops as a result of Coulomb friction can be very large and in some cases may exceed the ability of the material to carry such a stress. We therefore allow another limit to be placed on the value of the tangential force:

$$f^{n+1} = \min\left(f^{n+1}_{Coulomb}, \kappa A_{master}\right) \tag{23.23}$$

where A_{master} is the area of the master segment and κ is the viscous coefficient. Since more than one node may contribute to the shear stress of a segment, we recognize that the stress may still in some cases exceed the limit κ.

23.9 Tied Interfaces

Sudden transitions in zoning are permitted with the tied interfaces as shown in Figure 23.14 where two meshes of solid elements are joined. This feature can often decrease the amount of effort required to generate meshes since it reduces the need to match nodes across interfaces of merged parts.

Tied interfaces include four interface options of which three are in the Sliding Interface Definition Section in the LS-DYNA3D User's Manual. These are:

- Type 2 for tying surfaces with translational degrees of freedom.
- Type 6 for tying translational degrees of freedom of nodes to a surface
- Type 7 for tying both translational and rotational degrees of freedom of nodes

The fourth option is in the "Tie-Breaking Shell Definitions" Section of the user's manual and is meant as a way of tying edges of adjacent shells together. Unlike Type 7 this latter option does not require a surface definition, simply nodal lines, and includes a failure model based on plastic strain which can be turned off by setting the plastic failure strain to a high value. The first two options, which are equivalent in function but differ in the input definition, can be properly applied to nodes of elements which lack rotational degrees of freedom. The latter options must be used with element types that have rotational degrees of freedom defined at their nodes such as the shell and beam elements. One important application of Type 7 is that it allows edges of shells to be tied to shell surfaces. In such transitions the shell thickness is not considered.

Since the constraints are imposed only on the slave nodes, the more coarsely meshed side of the interface is recommended as the master surface. Ideally, each master node should coincide with a slave node to ensure complete displacement compatibility along the interface, but in practice this is often difficult if not impossible to achieve. In other words, master nodes that do not coincide with a slave node can interpenetrate through the slave surface.

Implementation of tied interface constraints is straightforward. Each time step we loop through the tied interfaces and update each one independently. First we distribute the

nodal forces and nodal mass of each slave node to the master nodes which define the segment containing the contact point,. i.e., the increments in mass and forces

$$\Delta M_m^i = \phi_i(\xi_{c,}\eta_c)M_s \quad i=1,...,4 \tag{23.24}$$

$$\Delta f_m^i = \phi_i(\xi_{c,}\eta_c)f_s \tag{23.25}$$

are added to the mass and force vector of the master surface. After the summation over all slave nodes is complete we can compute the acceleration of the master surface. The acceleration of each slave node a_{i_S} is then interpolated from the master segment containing its contact points:

$$a_{i_s} = \sum_{j=1}^{4} \phi_j(\xi_c,\eta_c)a_i^j \tag{23.26}$$

Velocities and displacements are now updated normally.

The interpolated contact point, (ξ_C, η_C), for each slave node is computed once since its relative position on the master segment is constant for the duration of the calculation If the closest point projection of the slave node to the master surface is non-orthogonal, values of (ξ_C, η_C) greater than unity will be computed. To allow for slight errors in the mesh definition, the slave node is left unconstrained if the magnitude of the contact point exceeds 1.02. Great care should be exercised in setting up tied interfaces to ensure that the slave nodes are covered by master segments.

Conflicting constraints must be avoided. Care should be taken not to include nodes that are involved in a tied interfaces in another tied interface, in constraint sets such as nodal constraint sets, in linear constraint equations, and in spot welds. Furthermore, tied interfaces between rigid and deformable bodies are not permitted. LS-DYNA3D checks for conflicting constraints on nodal points and if such conflicts are found, the calculation will terminate with an error message identifying the conflict. Nodes in tied interfaces should not be included as slave nodes in rigid wall definitions since interactions with stonewalls will cause the constraints that were applied in the tied interface logic to be violated. We do not currently check for this latter condition is LS-DYNA3D.

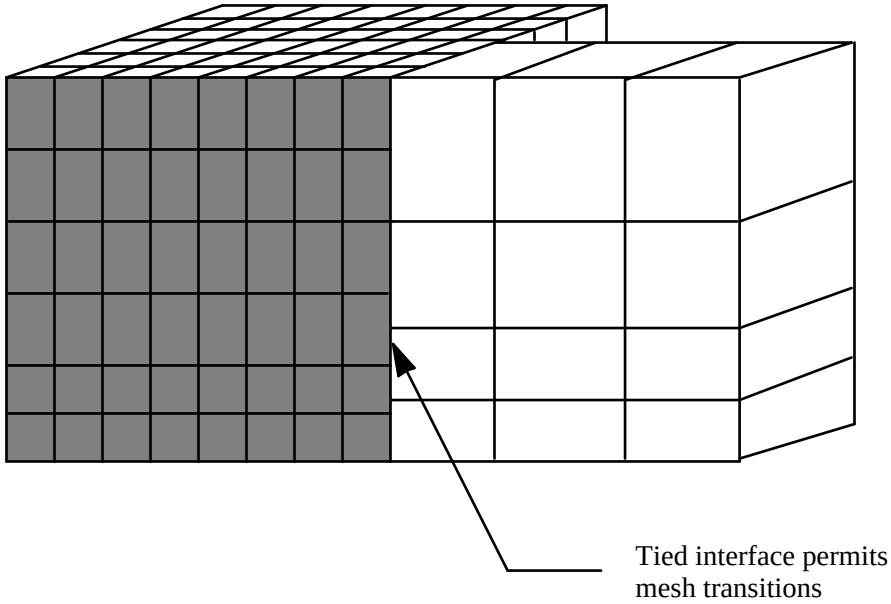


Figure 23.14. Tied interface used for a mesh transition.

23.10 Sliding-only Interfaces

This option is seldom useful in structural calculations. Its chief usefulness is for treating interfaces where the gaseous detonation products of a high explosive act on solid material. The present algorithm, though simple, has performed satisfactorily on a number of problems of this latter type. We briefly outline the approach here since the algorithm is still experimental and subject to change.

The method consists of five steps. In the first step, the mass per unit area (mass/area) and pressure are found at each node on the slave surface. Next, the contact point for each master node is found, and the slave mass/area and slave pressure at each master node is interpolated from the slave surface. In the third step, this pressure distribution is applied to the master surface to update its acceleration. In the fourth step, the normal component of the acceleration at each node on the master surface is scaled by its z-factor defined as the mass/area of the master surface at the master node divided by the sum of the mass/area of the slave surface at the master node. The last step consists of resetting the normal acceleration and velocity components of all slave nodes to ensure compatibility.

23.11 Bucket Sorting

Bucket sorting is now used extensively in both the surface to surface and single surface contact algorithms. Version 920 of LS-DYNA3D no longer contains one

dimensional sorting. Presently two separate but similar bucket sorts are in LS-DYNA3D. In the first and older method we attempt to find for each node the three nearest nodes. In the newer method which is systematically replacing the older method we locate the nearest segment.

The reasons for eliminating slave node tracking by incremental searching is illustrated in Figure 23.16 where surfaces are shown which cause the incremental searches to fail. In LS-DYNA3D tied interfaces are used extensively in many models creating what appears to the contact algorithms to be topologically disjoint regions. For robustness, our new algorithms account for such mesh transitions with only minor cost penalties. With bucket sorting incremental searches may still be used but for reliability they are used after contact is achieved. As contact is lost, the bucket sorting for the affected nodal points must resume.

In a direct search of a set of N nodes to determine the nearest node, the number of distance comparisons required is N – 1. Since this comparison needs to be made for each node, the total number of comparisons is N(N – 1), with each of these comparisons requiring a distance calculation

$$l^2 = (x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2 \tag{23.27}$$

that uses eight mathematical operations. The cumulative effect of these mathematical operations for N(N – 1) compares can dominate the solution cost at less than 100 elements.

The idea behind a bucket sort is to perform some grouping of the nodes so that the sort operation need only calculate the distance of the nodes in the nearest groups. Consider the partitioning of the one dimensional domain shown in Figure 23.16. With this partitioning the nearest node will either reside in the same bucket or in one of the two adjoining buckets. The number of distance calculations is now given by

$$\frac{3N}{a} - 1 \tag{23.28}$$

where a is the number of buckets. The total number of distance comparisons for the entire one dimensional surface is

$$N\left(\frac{3N}{a} - 1\right) \tag{23.29}$$

Thus, if the number of buckets is greater than 3, then the bucket sort will require fewer

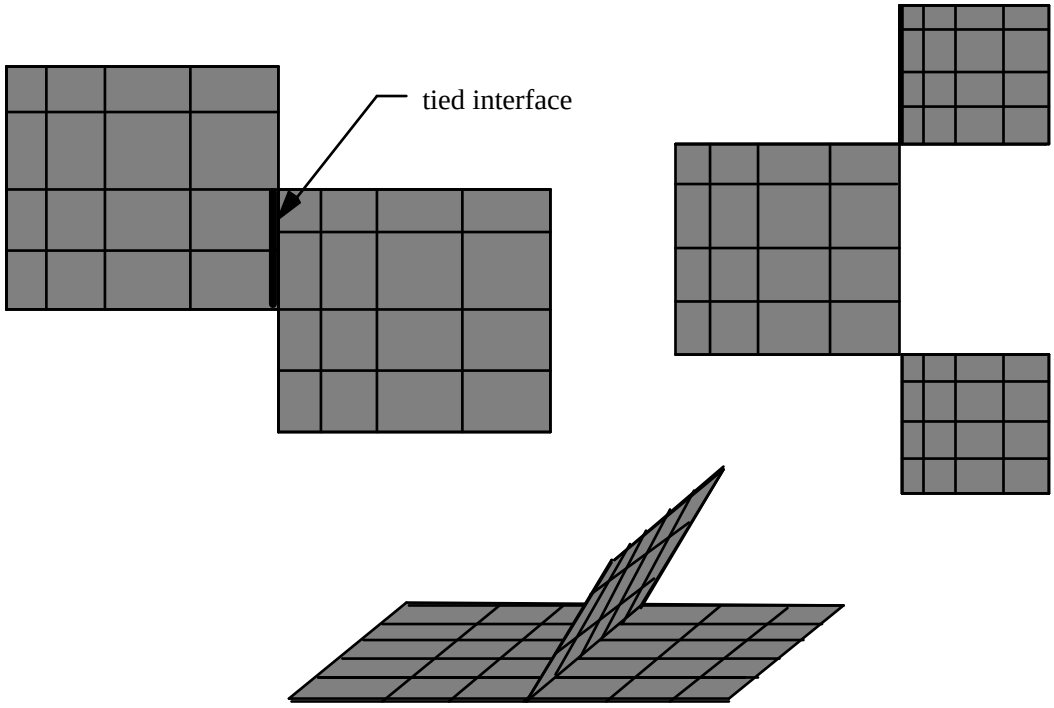


Figure 23.15. Incremental searching may fail on surfaces that are not simply connected. The new contact algorithm in LS-DYNA3D avoid incremental searching for nodal points that are not in contact and all these cases are considered.

distance comparisons than a direct sort. It is easy to show that the corresponding number of distance comparisons for two-dimensional and three-dimensional bucket sorts are given by

$$N\left(\frac{9N}{ab}-1\right) \quad \text{for 2D} \tag{23.30}$$

$$N\left(\frac{27N}{abc}-1\right) \quad \text{for 3D} \tag{23.31}$$

where b and c are the number of partitions along the additional dimension.

The cost of the grouping operations, needed to form the buckets, is nearly linear with the number of nodes N. For typical LS-DYNA3D applications, the bucket sort is 100 to 1000 times faster than the corresponding direct sort. However, the sort is still an expensive part of the contact algorithm, so that, to further minimize this cost, the sort is performed every ten or fifteen cycles and the nearest three nodes are stored. Typically, three to five percent of the calculational costs will be absorbed in the bucket sorting when most surface segments are included in the contact definition.

23.11.1Bucket Sorting in TYPE 4 Single Surface Contact

We set the number of buckets in the x, y, and z coordinate directions to NX, NY, and NZ, respectively. Letting, LMAX represent the longest characteristic length (found by checking the length of the segment diagonals and taking a fraction thereof) over all segments in the contact definition, the number of buckets in each direction is given by

$$NX = \frac{x_{\max} - x_{\min}}{LMAX} \tag{23.32}$$

$$NY = \frac{y_{\max} - y_{\min}}{LMAX} \tag{23.33}$$

$$NZ = \frac{z_{\max} - z_{\min}}{LMAX} \tag{23.34}$$

where the coordinate pairs (x_{min} ,x_{max}), (y_{min} ,y_{max}), and (z_{min} ,z_{max}) define the extent of the contact surface and are updated each time the bucket searching is performed. In order to dynamically allocate memory effectively with FORTRAN we further restrict the number of buckets such that the total number of buckets does not exceed the number of nodes in the contact surface, NSN or 5000:

$$\text{NX} \cdot \text{NY} \cdot \text{NZ} \leq \text{MIN (NSN,5000)} \tag{23.35}$$

If the characteristic length, LMAX, is large due to an oversized contact segment or an instability leading to a node flying off into space, the bucket sorting can be slowed down considerably since the number of buckets will be reduced. In older versions of DYNA3D this led to the error termination message “More than 1000 nodes in bucket.”

The formulas give by Belytschko and Lin [1985] are used to find the bucket containing a node with coordinates (x,y,z). The bucket pointers are given by

$$PX = NX \cdot \frac{(x - x_{\min})}{(x_{\max} - x_{\min})} + 1 \tag{23.36}$$

$$PY = NY \cdot \frac{(y - y_{\min})}{(y_{\max} - y_{\min})} + 1 \tag{23.37}$$

$$PZ = NZ \cdot \frac{(z - z_{\min})}{(z_{\max} - z_{\min})} + 1 \tag{23.38}$$

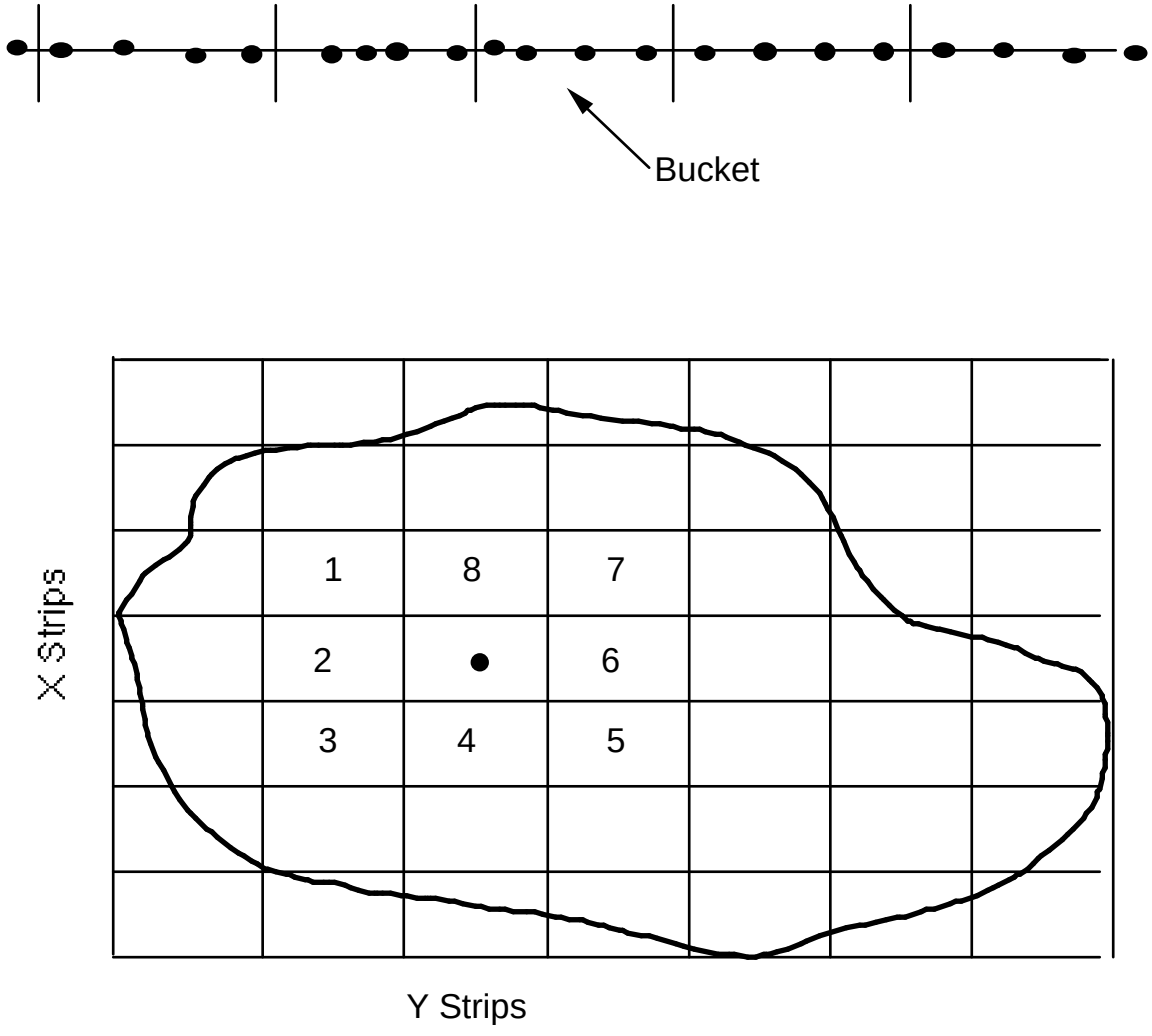


Figure 23.16. One- and two-dimensional bucket sorting.

and are used to compute the bucket number given by

$$NB = PX + (PY - 1) \cdot PX + (PZ - 1) \cdot PX \cdot PY \tag{23.39}$$

For each nodal point, k, in the contact surface we locate the three nearest neighboring nodes by searching all nodes in buckets from

$$\begin{aligned} &MAX(1,PX - 1),\ MIN(NX,PX + 1) \\ &MAX(1,PY - 1),\ MIN(NY,PY + 1) \\ &MAX(1,PZ - 1),\ MIN(NZ,PZ + 1) \end{aligned} \tag{23.40}$$

A maximum of twenty-seven buckets are searched. Nodes that share a contact segment with k are not considered in this nodal search. By storing the three nearest nodes and rechecking these stored nodes every cycle to see if the nearest node has changed, we avoid performing the bucket sorting every cycle. Typically, sorting every five to fifteen cycles is adequate. Implicit in this approach is the assumption that a node will contact just one surface. For this reason the single surface contact (TYPE 4 in LS-DYNA3D) is not applicable to all problems. For example, in metal forming applications both surfaces of the workpiece are often in contact.

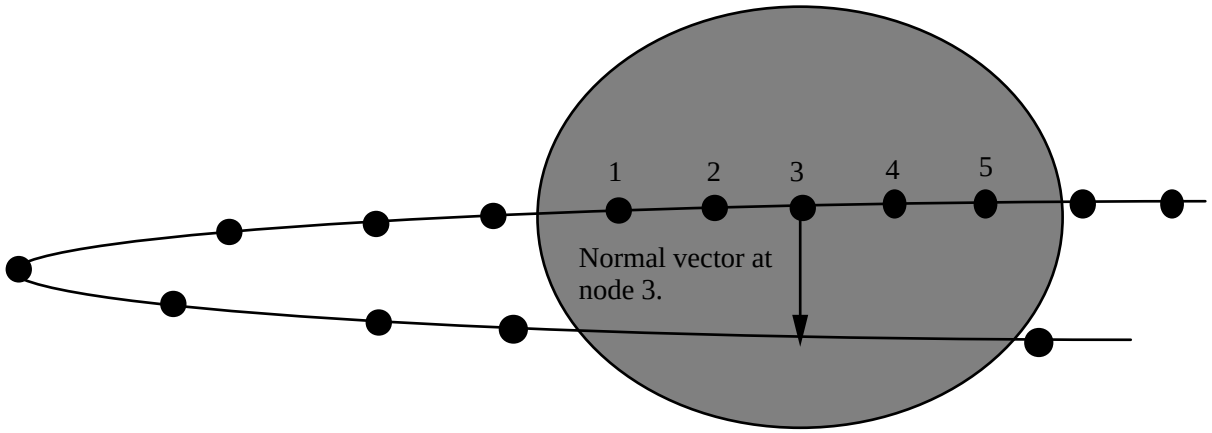


Figure 23.17. Nodes 2 and 4 share segments with node 3 and therefore the two nearest nodes are 1 and 5. The nearest contact segment is not considered since its nodes are not members of the nearest node set.

The nearest contact segment to a given node, k, is defined to be the first segment encountered when moving in a direction normal to the surface away from k. A major deficiency with the nearest node search is depicted in Figure 23.17 where the nearest nodes are not even members of the nearest contact segment. Obviously, this would not be

problem for a more uniform mesh. To overcome this problem we have adopted segment based searching in both surface to surface and single surface contact.

23.11.2Bucket Sorting in Surface to Surface and TYPE 13 Single Surface Contact

The procedure is roughly the same as before except we no longer base the bucket size on LMAX which can result in as few as one bucket being generated. Rather, the product of the number of buckets in each direction always approaches NSN or 5000 whichever is smaller,

$$NX \cdot NY \cdot NZ \rightarrow MIN(NSN,5000) \tag{23.41}$$

where the coordinate pairs (x_{min}, x_{max}), (y_{min}, y_{max}), and (z_{min}, z_{max}) span the entire contact surface. In the new procedure we loop over the segments rather than the nodal points. For each segment we use a nested DO LOOP to loop through a subset of buckets from IMIN to IMAX, JMIN to JMAX, and KMIN to KMAX where

$$\begin{aligned} IMIN &= MIN(PX1,PX2,PX3,PX4) \\ IMAX &= MAX\ (PX1,PX2,PX3,PX4) \\ \\ JMIN &= MIN(PY1,PY2,PY3,PY4) \\ JMAX &= MAX\ (PY1,PY2,PY3,PY4) \\ \\ KMIN &= MIN(PZ1,PZ2,PZ3,PZ4) \\ KMAX &= MAX\ (PZ1,PZ2,PZ3,PZ4) \end{aligned} \tag{23.42}$$

and PXk, PYk, PZk are the bucket pointers for the kth node. Figure 23.18 shows a segment passing through a volume that has been partitioned into buckets.

We check the orthogonal distance of all nodes in the bucket subset from the segment. As each segment is processed, the minimum distance to a segment is determined for every node in the surface and the two nearest segments are stored. Therefore the required storage allocation is still deterministic. This would not be the case if we stored for each segment a list of nodes that could possibly contact the segment.

We have now determined for each node, k, in the contact surface the two nearest segments for contact. Having located these segments we permanently store the node on these segments which is nearest to node k. When checking for interpenetrating nodes we check the segments surrounding the node including the nearest segment since during the steps between bucket searches it is likely that the nearest segment may change. It is possible to bypass nodes that are already in contact and save some computer time;

however, if multiple contact per node are admissible then bypassing the search may lead to unacceptable errors.

23.12 Single Surface Contact Algorithms in LS-DYNA3D

The single surface contact algorithms evolved from the surface to surface contact algorithms and the post contact searching follows the procedures employed for the surface to surface contact. Type 4 contact in LS-DYNA3D uses the following steps where NSEG is the number of contact segments and NSN is the number of nodes in the interface:

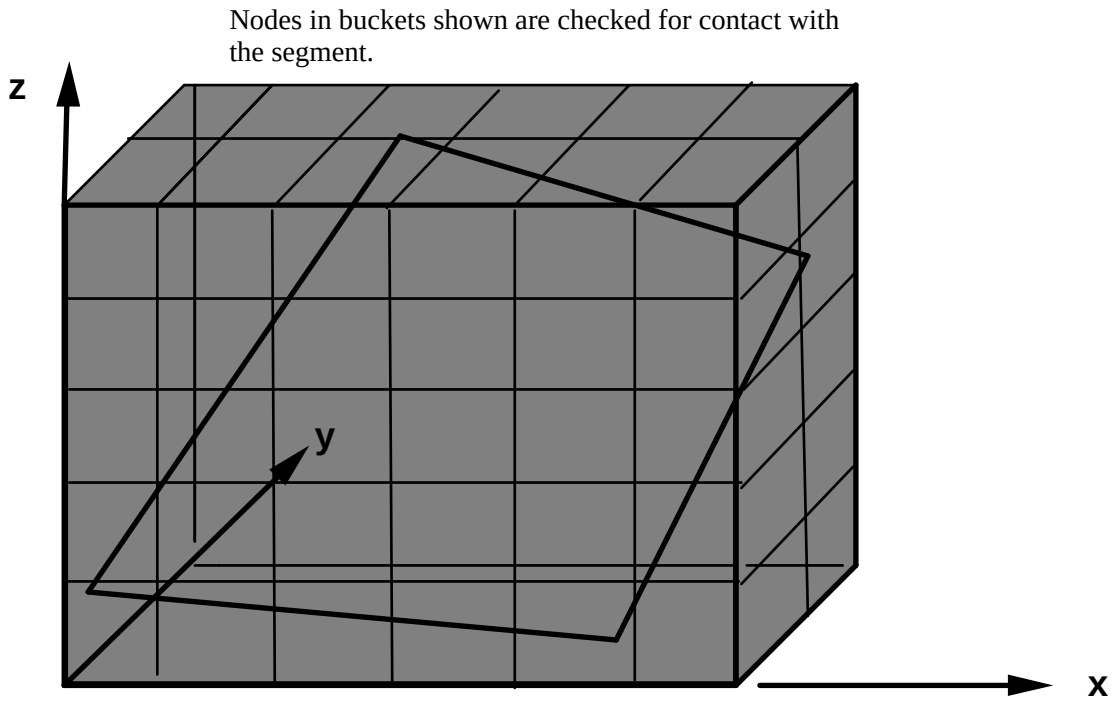


Figure 23.18. The orthogonal distance of each slave node contained in the box from the segment is determined. The box is subdivided into sixty buckets.

I. Loop through the contact segments from 1 to NSEG

1. Compute the normal segment vectors and accumulate an area weighted average at the nodal points to determine the normal vectors at the nodal points.

II. Loop through the slave nodes from 1 to NSN

1. Check all nearest nodes, stored from the bucket sort, and locate the node which is nearest.

2. Check to see if nearest node is within a penetration tolerance determined during the bucket sort, if not, proceed to the end of the loop.

3. For shell elements, determine if the nearest node is approaching the segment from the positive or negative side based on the right hand rule. Project both the node and the contact segment along the nodal normal vectors to account for the shell thickness.

4. Check for interpenetrating nodes and if a node has penetrated apply a nodal point force that is proportional to the penetration depth.

End of Loop

Of course, several obvious limitations of the above procedure exists. The normal vectors that are used to project the contact surface are meaningless for nodes along an intersection of two or more shell surfaces (Please see the sketch at the bottom of Figure 23.15). In this case the normal vector will be arbitrarily skewed depending on the choice of the numbering of the connectivities of the shells in the intersecting surfaces. Secondly, by considering the possibility of just one contact segment per node, metal forming problems cannot be handled within one contact definition. For example, if a workpiece is constrained between a die and a blankholder then at least some nodal points in the workpiece must necessarily be in contact with two segments-one in the die and the other in the workpiece. These two important limitations have motivated the development of the new bucket sorting procedure described above and the modified single surface contact procedure, type 13.

A major change in type 13 contact from type 4 is the elimination of the normal nodal vector projection by using the segment normal vector as shown in Figure 23.19.

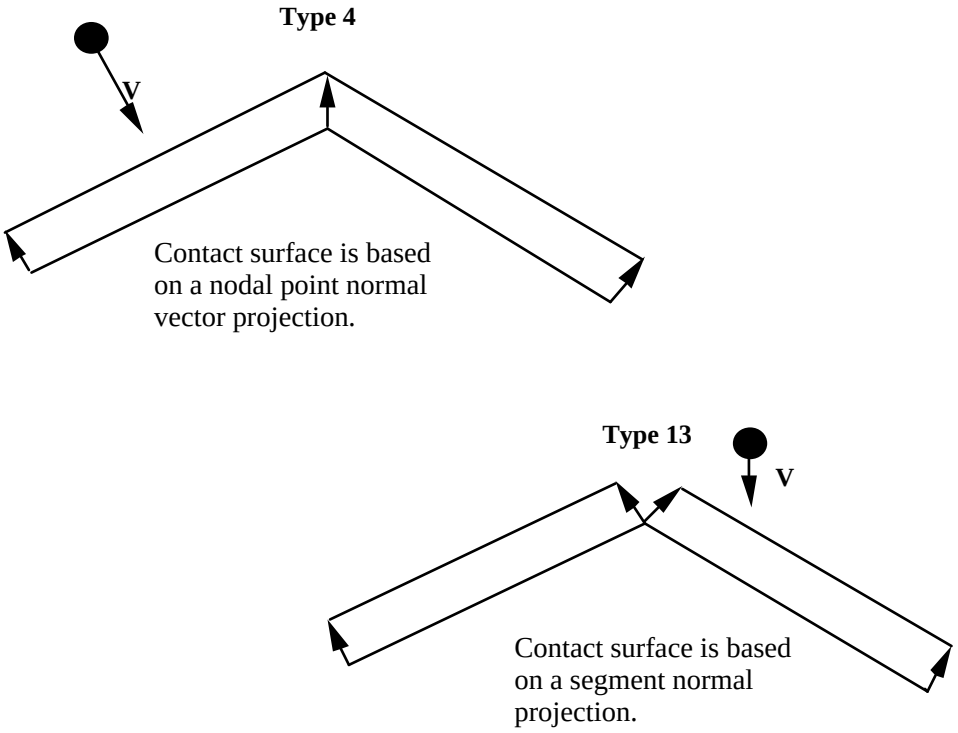


Figure 23.19. Projection of the contact surface for a node approaching from above is shown for types 4 and 13 contact.

Segment numbering within the contact surface is arbitrary when the segment normal is used greatly simplifying the model input generation. However, additional complexity is introduced since special handling of the nodal points is required at segment intersections where nodes may approach undetected as depicted in Figure 23.20. In this example an energy growth equal to $k\Delta L/2$, where k is the interface stiffness value, is generated as the penetrating node is forced back to the contact surface. This increase in energy shows up as a negative energy contribution in the ASCII interface energy file, SLEOUT. Failure to detect potential interpenetrations before they occur can result an overall energy growth during a simulation. It has been the introduction of additional coding to solve this and related problems, for example, the node on node case where multiple segments join in a warped surface, that has resulted is a decrease in efficiency since this new contact type was introduced. Without the special coding most problems ran successfully but even an infrequent failure cannot be tolerated in a production environment and causes a severe loss of confidence in the methodology. Generally, a slightly higher CPU cost is better (except in bench marking for speed, an endeavor which seems to be a hobby for some).

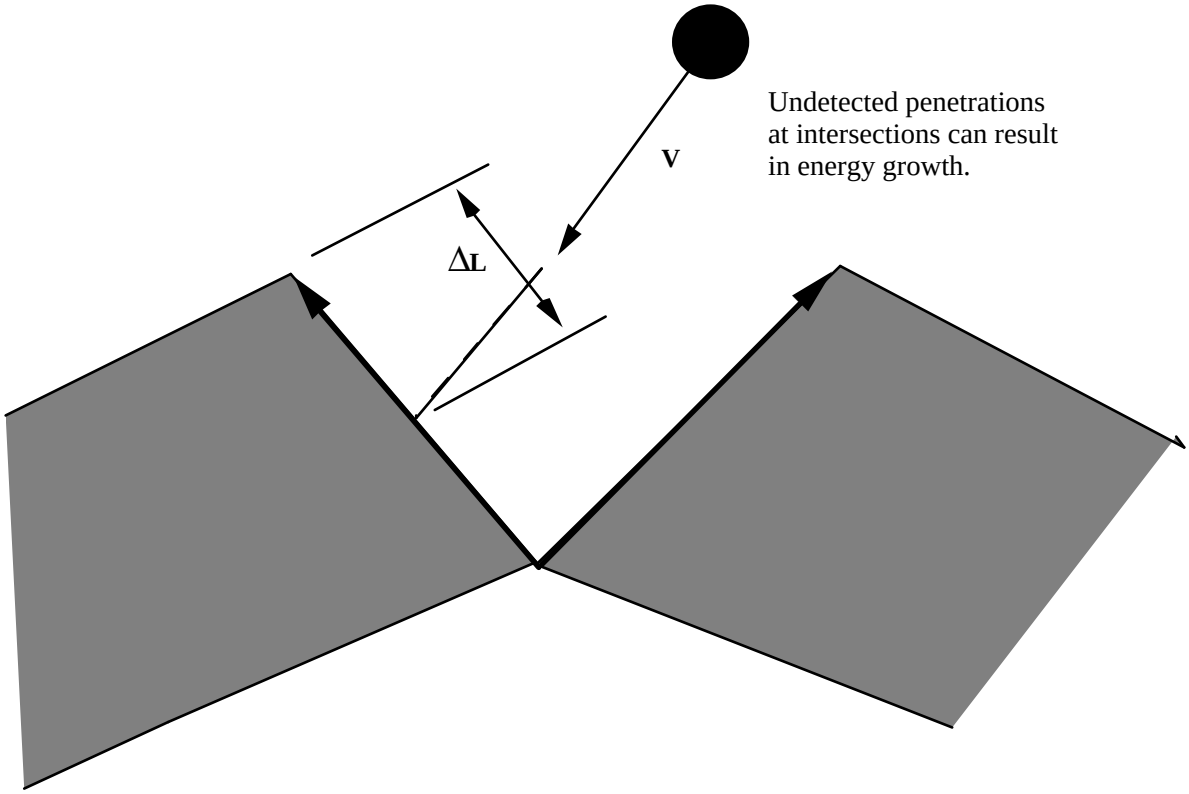


Figure 23.20. The lack of a continuous contact surface requires special care in cases where nodes can interpenetrate between contiguous segments.

Assuming the segment based bucket sort has been completed and closest segments are known for all slave nodes then the procedure for processing the type 13 contact simplifies to:

I. Loop through the slave nodes from 1 to NSN

1. If node is in contact, check to see if the contact segment has changed and if so, then update the closest segment information and the orientation flag which remembers the side in contact. Since no segment orientation information is stored this flag may change as the node moves from segment to segment.

2. Check the closest segment to see if the node is in contact if not then proceed to the end of the loop. If the slave node or contact segment connectivity are members of a shell element, project both the node and the contact segment along the segment normal vector to account for the shell thickness. A nodal thickness is stored for each node and a segment thickness is stored for each segment. A zero thickness is stored for solid elements. The thickness can be optionally updated to account for membrane thinning.

3. Check for interpenetrating nodes and if a node has penetrated apply a nodal point force that is proportional to the penetration depth.

End of Loop

Note that type 13 contact does not require the calculation of nodal normal vectors.

23.13 Surface to Surface Constraint Algorithm

The constraint algorithm that we implemented is based on the algorithm developed by Taylor and Flanagan [1989]. This involves a two-pass symmetric approach with a partitioning parameter, β, that is set between negative and positive unity where β=1 and β=-1 correspond to one way treatments with the master surface accumulating the mass and forces from the slave surface (for β=1) and visa versa (for β= -1). The searching algorithms are those used in the other contact algorithms for the surface to surface contact.

In this constraint approach the accelerations, velocities, and displacements are first updated to a trial configuration without accounting for interface interactions. After the update, a penetration force is computed for the slave node as a function of the penetration distance ΔL:

$$f_p = \frac{m_s \Delta L}{\Delta t^2} n \tag{23.43}$$

where n is the normal vector to the master surface.

We desire that the response of the normal component of the slave node acceleration vector, a_s, of a slave node residing on master segment k be consistent with the motion of the master segment at its contact segment (s_c,t_c), i.e.,

$$a_s = \phi_1(s_c,t_c)\, a^1_{nk} + \phi_2(s_c,t_c)\, a^2_{nk} + \phi_3(s_c,t_c)\, a^3_{nk} + \phi_4(s_c,t_c)\, a^4_{nk} \tag{23.44}$$

For each slave node in contact with and penetrating through the master surface in its trial configuration, its nodal mass and its penetration force given by Equation (2.43) is accumulated to a global master surface mass and force vector:

$$\left(m_k + \sum_s m_{ks} \right) a_{nk} = \sum_s f_{ks} \tag{23.45}$$

where

$$m_{ks} = \phi_k m_s$$

$$f_{ks} = \phi_k f_p \tag{23.46}$$

After solving Equation (2.45) for the acceleration vector, a_{nk}, we can obtain the acceleration correction for the slave node as

$$a_{ns} = a_s - \frac{f_p}{m_s} \tag{23.47}$$

The above process is repeated after reversing the master and slave definitions. In the final step the averaged final correction to the acceleration vector is found

$$a_n^{final} = \frac{1}{2}(1-\beta)a_n^{1st\ pass} + \frac{1}{2}(1+\beta)a_n^{2nd\ pass} \tag{23.48}$$

and used to compute the final acceleration at time n+1

$$a^{n+1} = a^{trial} + a_n^{final} \tag{23.49}$$

Friction, as described by Taylor and Flanagan [1989], is included in our implementation. Friction resists the relative tangential velocity of the slave node with respect to the master surface. This relative velocity if found by subtracting from the relative velocity:

$$v_r = v_s - \left(\phi_1 v_k^1 + \phi_2 v_k^2 + \phi_3 v_k^3 + \phi_4 v_k^4\right) \tag{23.50}$$

the velocity component normal to the master segment:

$$v_t = v_r - (n \cdot v_r)n \tag{23.51}$$

A trial tangential force is computed that will cancel the tangential velocity

$$f^* = \frac{m_s \upsilon_t}{\Delta t} \tag{23.52}$$

where υ_t is the magnitude of the tangential velocity vector

$$\upsilon_t = \sqrt{v_t \cdot v_t} \tag{23.53}$$

The magnitude of the tangential force is limited by the magnitude of the product of the Coulomb friction constant with the normal force defined as

$$f_n = m_s a_{ns} \cdot n \tag{23.54}$$

The limiting force is therefore

$$F_y = m|f_n| \tag{23.16}$$

and

$$f^{n+1} = f^* \qquad \text{if} \quad |f^*| \le F_y \tag{23.19}$$

$$f^{n+1} = \frac{F_y f^*}{|f^*|} \qquad \text{if} \quad |f^*| > F_y \tag{23.20}$$

Therefore, using the above equations the modification to the tangential acceleration component of the slave node is given by

$$a_t = \min\left(\mu a_{nt} \cdot n, \frac{|v_s|}{\Delta t}\right) \tag{23.55}$$

which must act in the direction of the tangential vector defined as

$$n_t = \frac{v_t}{v_t} \tag{23.56}$$

The corrections to both the slave and master node acceleration components are:

$$a_{ts} = a_t n_t$$

$$a_{tk} = -\phi_k \frac{a_s m_s}{m_k} n_t \tag{23.57}$$

The above process is again repeated after reversing the master and slave definitions. In the final step the averaged final correction to the acceleration vector is found

$$a_t^{final} = \tfrac{1}{2}(1-\beta) a_t^{1st\ pass} + \tfrac{1}{2}(1+\beta) a_t^{2nd\ pass} \tag{23.58}$$

and is used to compute the final acceleration at time n+1

$$a^{n+1} = a^{trial} + a_n^{final} + a_t^{final} \tag{23.59}$$

A significant disadvantage of the constraint method relative to the penalty method appears if an interface node is subjected to additional constraints such as spot welds, constraint equations, tied interfaces, and rigid bodies. Rigid bodies can often be used with this contact algorithm if their motions are prescribed as is the case in metal forming. For the more general cases involving rigid bodies, the above equations are not directly applicable since the local nodal masses of rigid body nodes are usually meaningless. Subjecting the two sides of a shell surface to this constraint algorithm will also lead to erroneous results since an interface node cannot be constrained to move simultaneously on two mutually independent surfaces. In the latter case the constraint technique could be used on one side and the penalty method on the other.

The biggest advantage of the constraint algorithm is that interface nodes remain on or very close to the surfaces they are in contact with. Furthermore, elastic vibrations that can occur in penalty formulations are insignificant with the constraint technique. The problem related to finding good penalty constants for the contact are totally avoided by the latter approach. Having both methods available is possibly the best option of all.

23.14 Planar Rigid Boundaries

The rigid boundary represents the simplest contact problem and is therefore treated separately. As shown in Figure 23.19 the boundary is flat, finite or infinite in extent and is defined by an outward normal unit vector **n** with the origin of **n** at a corner point on the wall if the wall is finite or at an arbitrary point on the wall if the wall extends to infinity. The finite wall is rectangular with edges of length L and M. Unit vectors **l** and **m** lie along these edges. A subset of nodes is defined, usually boundary nodes of the calculational model, that are not allowed to penetrate. Let *k* represent one such boundary node and let $\mathbf{r}_k^{n+1}$ be the position vector from the origin of **n** to k after locally updating the coordinates. Each time step prior to globally updating the velocities and accelerations we check k to ensure that the nodes lies within the wall by checking that both inequalities are satisfied:

$$\begin{aligned} r_k^{n+1} \cdot l &\le L \\ r_k^{n+1} \cdot m &\le M \end{aligned} \tag{23.50}$$

This test is skipped for the infinite rigid wall. Assuming that the inequality is satisfied, we then check the penetration condition to see if k is penetrating through the wall,

$$r_k^{n+1} \cdot n < 0 \tag{23.51}$$

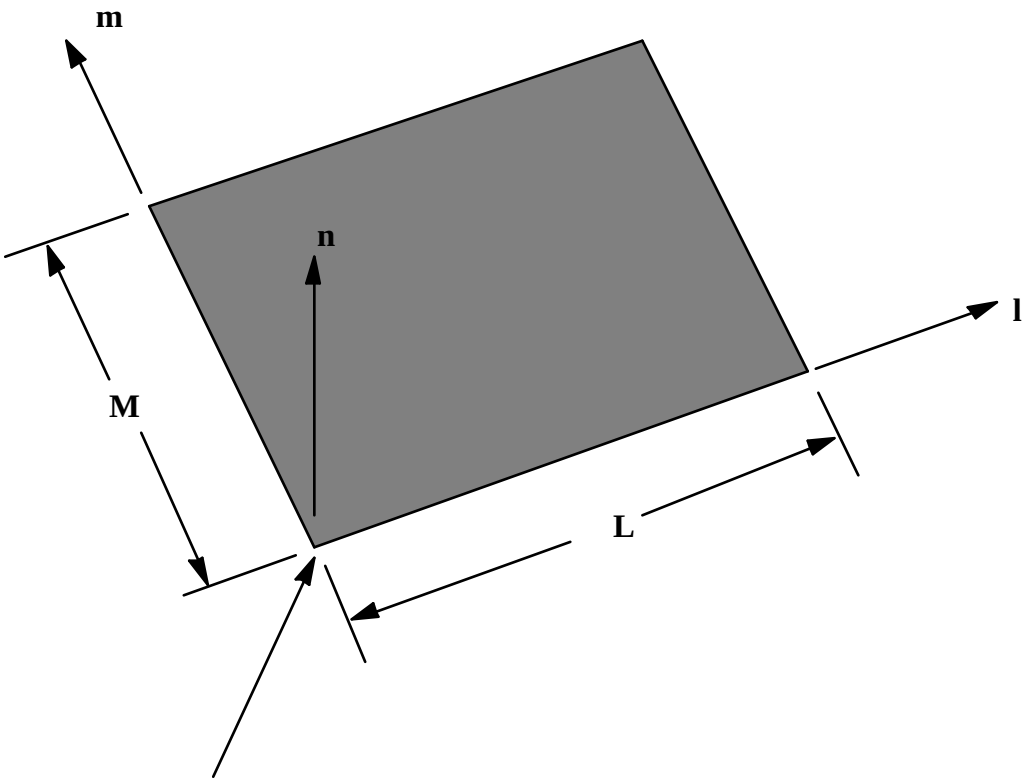


Figure 23.21. Vector **n** is normal to the stonewall. An optional vector **l** can be defined such that **m**=**n**× **l**. The extent of the stonewall is limited by defining **L** and **M**. A zero value for either of these lengths indicates that the stonewall is infinite in that direction.

and if so, the velocity and acceleration components normal to the wall are set to zero:

$$\begin{aligned} a^{\,n}_{k_{new}} &= a^{\,n}_{k_{old}} - \left(a^{\,n}_{k_{old}} \cdot n\right) n \\ v^{\,n}_{k_{new}} &= v^{\,n}_{k_{old}} - \left(v^{\,n}_{k_{old}} \cdot n\right) n \end{aligned} \tag{23.51}$$

Here $\mathbf{a}_k$ and $\mathbf{v}_k$ are the nodal acceleration and velocity of node k, respectively. This procedure for stopping nodes represents a perfectly plastic impact resulting in an irreversible energy loss. The total energy dissipated is found by taking the difference between the total kinetic energy of all the nodal points slaved to the rigid wall before and after impact with the wall. This energy is computed and accumulated in LS-DYNA3D and is printed in the GLSTAT (global statistics) file.

The tangential motion of the boundary node may be either unconstrained, fully constrained, or subjected to Coulomb friction while it is in contact with the rigid boundary.

Coulomb friction acts along a vector defined as:

$$n_t = \frac{v^n_{k_{new}}}{\sqrt{v^n_{k_{new}} \cdot v^n_{k_{new}}}} \tag{23.52}$$

The magnitude of the tangential force which is applied to oppose the motion is given as

$$f_t = \min\left(\frac{m_s \sqrt{v^n_{k_{new}} \cdot v^n_{k_{new}}}}{\Delta t}, \mu |f_n| \right) \tag{23.53}$$

i.e., the maximum value required to hold the node in the same relative position on the stonewall or the product of the coefficient of friction and the magnitude of the normal force which ever is less. In Equation (23.53)m_s is the mass of the slave node and f_n is the normal force.

23.15 Geometric Rigid Boundaries

Geometric rigid walls are similar to the finite planar rigid walls but can be subjected to a prescribed motion only. As the surface moves into the structure, external work is generated which is integrated and added to the overall energy balance. In addition to the external work plastic work also is generated as nodes contact the wall. Contact can occur any of surfaces which enclose the volume.

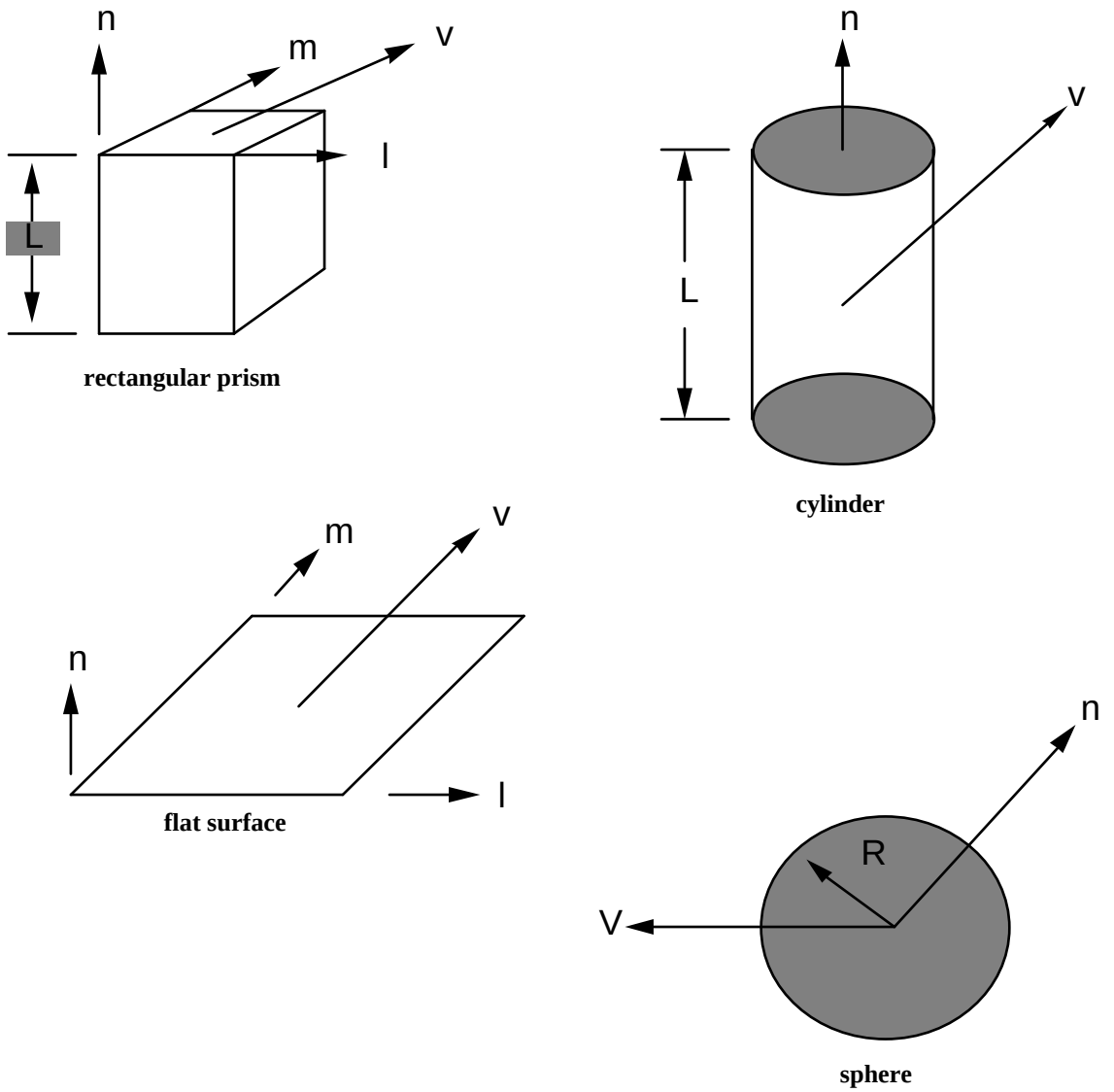


Figure 23.22 Vector **n** determines the orientation of the generalized stonewalls. For the prescribed motion options the wall can be moved in the direction **V** as shown.

24. GEOMETRIC CONTACT ENTITIES

Contact algorithms in LS-DYNA3D currently can treat any arbitrarily shaped surface by representing the surface with a faceted mesh. Occupant modeling can be treated this way by using fine meshes to represent the head or knees. The generality of the faceted mesh contact suffers drawbacks when modeling occupants, however, due to storage requirements, computing costs, and mesh generation times. The geometric contact entities were added as an alternate method to model cases of curved rigid bodies impacting deformable surfaces. Much less storage is required and the computational cost decreases dramatically when compared to the more general contact.

Geometric contact entities are developed using a standard solids modeling approach. The geometric entity is defined by a scalar function G(x,y,z). The solid is determined from the scalar function as follows:

G(x,y,z) > 0	The point (x,y,z) is outside the solid.	(24.1)
G(x,y,z) = 0	The point (x,y,z) is on the surface of the solid.	(24.2)
G(x,y,z) < 0	The point (x,y,z) is inside the solid.	(24.3)

Thus, by a simple function evaluation, a node can be immediately determined to be outside the solid or in contact. Figure 24.1 illustrates this for a cylinder.

If the node is in contact with the solid, a restoring force must be applied to eliminate further penetration. A number of methods are available to do this such as Lagrange multipliers or momentum based methods. The penalty methods was selected because it is the simplest and most efficient method. Also, in our applications the impact velocities are at a level where the penalty methods provide almost the identical answer as the exact solution.

Using the penalty method, the restoring force is proportional to the penetration distance into the solid and acts in the direction normal to the surface of the solid. Thus, the penetration distance and the normal vector must be determined. The surface normal vector is conveniently determined from the gradient of the scalar function.

$$\vec{N}(x,y,z) = \frac{\frac{\partial G}{\partial x} i + \frac{\partial G}{\partial y} j + \frac{\partial G}{\partial z} k}{\sqrt{\left(\frac{\partial G}{\partial x}\right)^2 + \left(\frac{\partial G}{\partial y}\right)^2 + \left(\frac{\partial G}{\partial z}\right)^2}} \tag{24.4}$$

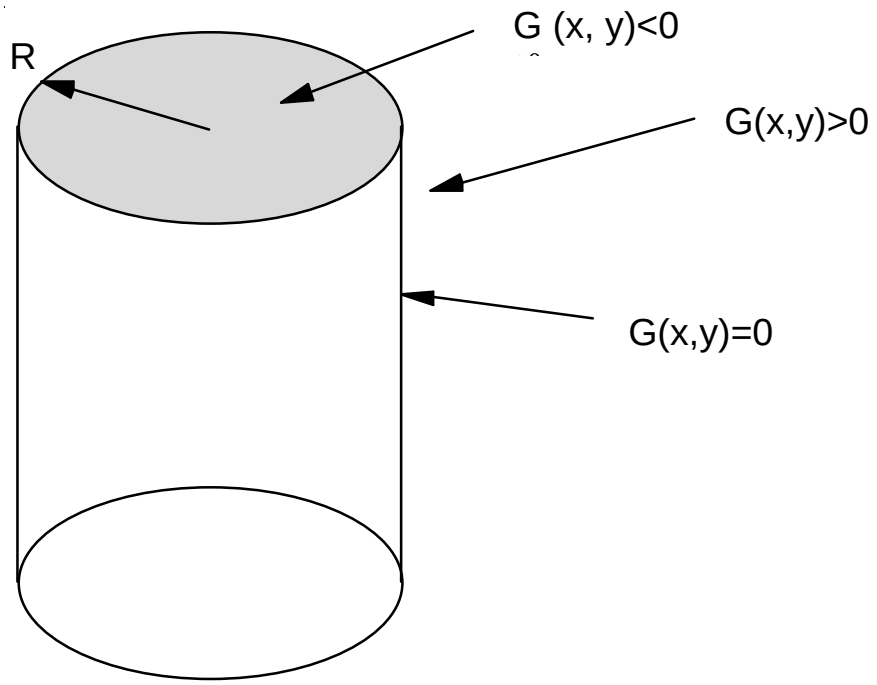


Figure 24.1. Determination of whether a node is interior or exterior to the cylindrical surface $G(x,y)=x^2+y^2-R^2$

for all (x,y,z) such that G(x,y,z) = 0. The definition of G(x,y,z) guarantees that this vector faces in the outward direction. When penetration does occur, the function G(x,y,z) will be slightly less than zero. For curved surfaces this will result in some errors in calculating the normal vector, because it is not evaluated exactly at the surface. In an implicit code, this would be important, however, the explicit time integration scheme in DYNA3D uses such a small time step that penetrations are negligible and the normal function can be evaluated directly at the slave node ignoring any penetration.

The penetrations distance is the last item to be calculated. In general, the penetration distance, d, is determined by.

$$d = \left| \vec{X}_n - \vec{X}'_n \right| \tag{24.5}$$

where,

$\vec{X}_n$ is the location of node n and

$\vec{X}'_n$ is the nearest point on the surface of the solid.

To determine $\vec{X}_n'$, a line function is defined which passes through $\vec{X}_n$ and is normal to the surface of the solid:

$$\vec{L}(s) = \vec{X}_n + s\vec{N}\left(\vec{X}_n\right) \tag{24.6}$$

Substituting the line function into the definition of the Equation (24.2) surface of a solid body gives:

$$G\left(\vec{X}_n + s\vec{N}\left(\vec{X}_n\right)\right) = 0 \tag{24.7}$$

If Equation (24.7) has only one solution, this provides the parametric coordinates s which locates $\vec{X}_n'$. If Equation (24.7) has more than one root, then the root which minimizes Equation (24.5) locates the point $\vec{X}_n'$.

The penalty method defines the restoring forces as:

$$\vec{f} = pd\vec{N}\left(\vec{X}_n'\right) \tag{24.8}$$

where p is a penalty factor and is effectively a spring constant. To minimize the penetration of the slave node into the solid, the constant p is set large, however, it should not be set so large that the Courant stability criteria is violated. This criteria for the slave node tells us that:

$$\Delta t \le \frac{2}{\omega_{\max}} = \frac{2}{\sqrt{\frac{K_n}{m_n}}} = 2\sqrt{\frac{m_n}{K_n}} \tag{24.9}$$

where, K_n is the stiffness of node n and m_n is the mass of node n.

The penalty factor, p, is determined by choosing a value which results in a penalty/ slave mass oscillator which has a characteristic time step that is ten times larger than the Courant time step:

$$10\Delta t = 2\sqrt{\frac{m_n}{p_n}} \tag{24.10}$$

Solving for p_n gives:

$$p_n = \frac{4m_n}{\left(100\Delta t^{\,2}\right)} \tag{24.11}$$

Inclusion of any structural elements into the occupant model will typically result in very large stiffnesses due to the small time step and the $(1/\Delta t)^2$ term. Thus the method is highly effective even with impact velocities on the order of 1km/sec.

The scalar function $G\left(\vec{X}\right)$ is frequently more conveniently expressed as $g(\vec{x})$ where, *g* is the function defined in local coordinates and $\vec{x}$ is the position in local coordinates. The local entity is related to the global coordinates by:

$$\vec{x} = [T]\left(\vec{X}_j - \vec{Q}_j\right) \tag{24.12}$$

where $\vec{Q}_j$ is the offset and [*T*] is a rotation matrix. The solid scalar function and the penetration distance can be evaluated in either local or global coordinates with no difference to the results. When working in local coordinates, the gradient of the local scalar function provides a normal vector which is in the local system and must be transformed into the global by:

$$\vec{N}\left(\vec{\bar{X}}\right) = [T]^T \vec{n}\left(\vec{x}\right) \tag{24.13}$$

An ellipsoid is defined by the function:

$$G(x,y,z) = \left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 + \left(\frac{z}{c}\right)^2 - 1 \tag{24.14}$$

The gradient of G is $\frac{2x}{a^2}i + \frac{2y}{a^2}j + \frac{2z}{a^2}k$ and the normal vector is:

$$n(x,y,z) = \frac{\left(\frac{x}{a^2}i + \frac{y}{b^2}j + \frac{z}{c^2}k\right)}{\sqrt{\frac{x^2}{a^4} + \frac{y^2}{b^4} + \frac{z^2}{c^4}}} \tag{24.15}$$

Substituting Equations (24.6) and (24.14) into Equation (24.2) gives:

$$\left[\left(\frac{n_x}{a}\right)^2+\left(\frac{n_y}{b}\right)^2+\left(\frac{n_z}{c}\right)^2\right]s^2+2\left[\frac{n_x x_n}{a^2}+\frac{n_y y_n}{b^2}+\frac{n_z z_n}{c^2}\right]s$$
$$+\left[\left(\frac{x_n}{a}\right)^2+\left(\frac{y_n}{b}\right)^2+\left(\frac{z_n}{c}\right)^2-1\right]=0$$

Solving this quadratic equation for *s* provides the intercepts for the nearest point on the ellipsoid and the opposite point of the ellipsoid where the normal vector, $\vec{X}_n$, also points toward.

Currently, this method has been implemented for the case of an infinite plane, a cylinder, a sphere, and an ellipsoid with appropriate simplifications. The ellipsoid is intended to be used with rigid body dummy models. The methods are, however, quite general so that many more shapes could be implemented. A direct coupling to solids modeling packages should also be possible in the future.

25. NODAL CONSTRAINTS

In this section nodal constraints and linear constraint equations are described.

25.1 Nodal Constraint Sets

This option forces groups of nodes to move together with a common translational acceleration in either one or more degrees of freedom. The implementation is straightforward with the common acceleration defined by

$$a_{i_{common}} = \frac{\sum_{j}^{n} M_j a_i^j}{\sum_{j}^{n} M_j} \tag{25.1}$$

where n is the number of nodes, a_i^j is the acceleration of the jth constrained node in the ith direction, and $a_{\mathrm{i_{common}}}$ is the common acceleration.

25.2 Linear Constraint Equations

Linear constraint equations of the form:

$$\sum_{k=1}^{n} C_k u_k = C_0 \tag{25.2}$$

can be defined where n is the number of constrained degrees of freedom, u_k is a constrained nodal displacement, and the C_k are user-defined coefficients. Unless LS-DYNA3D is initialized by linking to an implicit code to satisfy this equation at the beginning of the calculation, the constant C_0 is assumed to be zero. The first constrained degree of freedom is eliminated from the equations of motion:

$$u_1 = C_0 - \sum_{k=2}^{n} \frac{C_k}{C_1} u_k \tag{25.3}$$

Its velocities and accelerations are given by

$$\dot{u}_1 = -\sum_{k=2}^{n} \frac{C_k}{C_1}\dot{u}_k$$

$$\ddot{u}_1 = -\sum_{k=2}^{n} \frac{C_k}{C_1}\ddot{u}_k \tag{25.4}$$

respectively. In the implementation a transformation matrix $\underset{\sim}{L}$ is constructed relating the unconstrained $\underset{\sim}{u}$ and constrained $\underset{\sim}{u}_c$ degrees of freedom. The constrained accelerations used in the above equation are given by:

$$\underset{\sim}{u}_c = \left[\underset{\sim}{L}^t \underset{\sim}{M} \underset{\sim}{L}\right]^{-1} \underset{\sim}{L}^t \underset{\sim}{F} \tag{25.5}$$

where $\underset{\sim}{M}$ is the diagonal lumped mass matrix and $\underset{\sim}{F}$ is the righthand side force vector. This requires the inversion of the condensed mass matrix which is equal in size to the number of constrained degrees of freedom minus one. The inverse of the condensed mass matrix is computed in the initialization phase and stored in core.

26. VECTORIZATION AND PARALLELIZATION

26.1 Vectorization

In 1978, when the author first vectorized DYNA3D on the CRAY-1, a four fold increase in speed was attained. This increase was realized by recoding the solution phase to process vectors in place of scalars. It was necessary to process elements in groups rather than individually as had been done earlier on the CDC-7600 supercomputers.

Since vector registers on Cray computers are 64 words long, vector lengths of 64 or some multiple of 64 are appropriate. In LS-DYNA3D, groups of 128 elements or possibly some larger integer multiple of 64 are utilized. Larger groups give a marginally faster code, but can reduce computer time sharing efficiency because of increased core requirements. If elements within the group reference more than one material model, subgroups are formed for consecutive elements that reference the same model. LS-DYNA3D internally sorts elements by material to maximize vector lengths.

Conceptually, vectorization is straightforward. Each scalar operation that is normally executed once for one element, is repeated for each element in the group. This means that each scalar is replace by an array, and the operation is put into a DO-loop. For example, the nodal force calculation for the hexahedron element appeared in a scalar version of DYNA3D as:

```
E11 = SGV1*PX1+SGV4*PY1+SGV6*PZ1
E21 = SGV2*PY1+SGV4*PX1+SGV5*PZ1
E31 = SGV3*PZ1+SGV6*PX1+SGV5*PY1
E12 = SGV1*PX2+SGV4*PY2+SGV6*PZ2
E22 = SGV2*PY2+SGV4*PX2+SGV5*PZ2
E32 = SGV3*PZ2+SGV6*PX2+SGV5*PY2
E13 = SGV1*PX3+SGV4*PY3+SGV6*PZ3
E23 = SGV2*PY3+SGV4*PX3+SGV5*PZ3
E33 = SGV3*PZ3+SGV6*PX3+SGV5*PY3
E14 = SGV1*PX4+SGV4*PY4+SGV6*PZ4
E24 = SGV2*PY4+SGV4*PX4+SGV5*PZ4
E34 = SGV3*PZ4+SGV6*PX4+SGV5*PY4
```

and in the vectorized version as:

```
      DO 110 I=LFT, LLT
      E11(I)  = SGV1(I)*PX1(I)+SGV4(I)*PY1(I)+SGV6(I)*PZ1(I)
      E21(I)  = SGV2(I)*PY1(I)+SGV4(I)*PX1(I)+SGV5(I)*PZ1(I)
      E31(I)  = SGV3(I)*PZ1(I)+SGV6(I)*PX1(I)+SGV5(I)*PY1(I)
      E12(I)  = SGV1(I)*PX2(I)+SGV4(I)*PY2(I)+SGV6(I)*PZ2(I)
      E22(I)  = SGV2(I)*PY2(I)+SGV4(I)*PX2(I)+SGV5(I)*PZ2(I)
      E32(I)  = SGV3(I)*PZ2(I)+SGV6(I)*PX2(I)+SGV5(I)*PY2(I)
      E13(I)  = SGV1(I)*PX3(I)+SGV4(I)*PY3(I)+SGV6(I)*PZ3(I)
      E23(I)  = SGV2(I)*PY3(I)+SGV4(I)*PX3(I)+SGV5(I)*PZ3(I)
      E33(I)  = SGV3(I)*PZ3(I)+SGV6(I)*PX3(I)+SGV5(I)*PY3(I)
      E14(I)  = SGV1(I)*PX4(I)+SGV4(I)*PY4(I)+SGV6(I)*PZ4(I)
      E24(I)  = SGV2(I)*PY4(I)+SGV4(I)*PX4(I)+SGV5(I)*PZ4(I)
110   E34(I)  = SGV3(I)*PZ4(I)+SGV6(I)*PX4(I)+SGV5(I)*PY4(I)
```

where $1 \le LFT \le LLT \le n$. Elements LFT to LLT inclusive use the same material model and n is a integer multiple of 64.

Gather operations are vectorized on most supercomputers. In the gather operation, variables needed for processing the element group are pulled from global arrays into local vectors. For example, the gather operation:

```
      DO 10 I=LFT, LLT
        X1(I) =  X(1,IX1(I))
        Y1(I) =  X(2,IX1(I))
        Z1(I) =  X(3,IX1(I))
       VX1(I) =  V(1,IX1(I))
       VY1(I) =  V(2,IX1(I))
       VZ1(I) =  V(3,IX1(I))
        X2(I) =  X(1,IX2(I))
        Y2(I) =  X(2,IX2(I))
        Z2(I) =  X(3,IX2(I))
       VX2(I) =  V(1,IX2(I))
       VY2(I) =  V(2,IX2(I))
       VZ2(I) =  V(3,IX2(I))
        X3(I) =  X(1,IX3(I))
        X3(I) =  X(2,IX3(I))
        X3(I) =  X(3,IX3(I))
```

```
                .
                .
                .
          X8(I) = X(1,IX8(I))
          Y8(I) = X(2,IX8(I))
          Z8(I) = X(3,IX8(I))
         VX8(I) = V(1,IX8(I))
         VY8(I) = V(2,IX8(I))
10       VZ8(I) = V(3,IX8(I))
```

initializes the nodal velocity and coordinate vector for each element in the subgroup LFT to LLT. In the scatter operation, element nodal forces are added to the global force vector. The force assembly does not vectorize unless special care is taken as described below.

In general the element force assembly is given in FORTRAN by:

```
      DO 30 I=1,NODFRC
      DO 20 N=1,NUMNOD
      DO 10 L=LFT,LLT
      RHS(I,IX(N,L))=RHS(I,IX(N,L))+FORCE(I,N,L)
   10 CONTINUE
   20 CONTINUE
   30 CONTINUE
```

where NODFRC is the number of force components per node (3 for solid elements, 6 for shells), LFT and LLT span the number of elements in the vector block, NUMNOD is the number of nodes defining the element, FORCE contains the force components of the individual elements, and RHS is the global force vector. This loop does not vectorize since the possibility exists that more that one element may contribute force to the same node. FORTRAN vector compilers recognize this and will vectorize only if directives are added to the source code. If all elements in the loop bounded by the limits LFT and LLT are disjoint, the compiler directives can be safely added. We therefore attempt to sort the elements as shown in Figure 26.1 to guarantee disjointness.

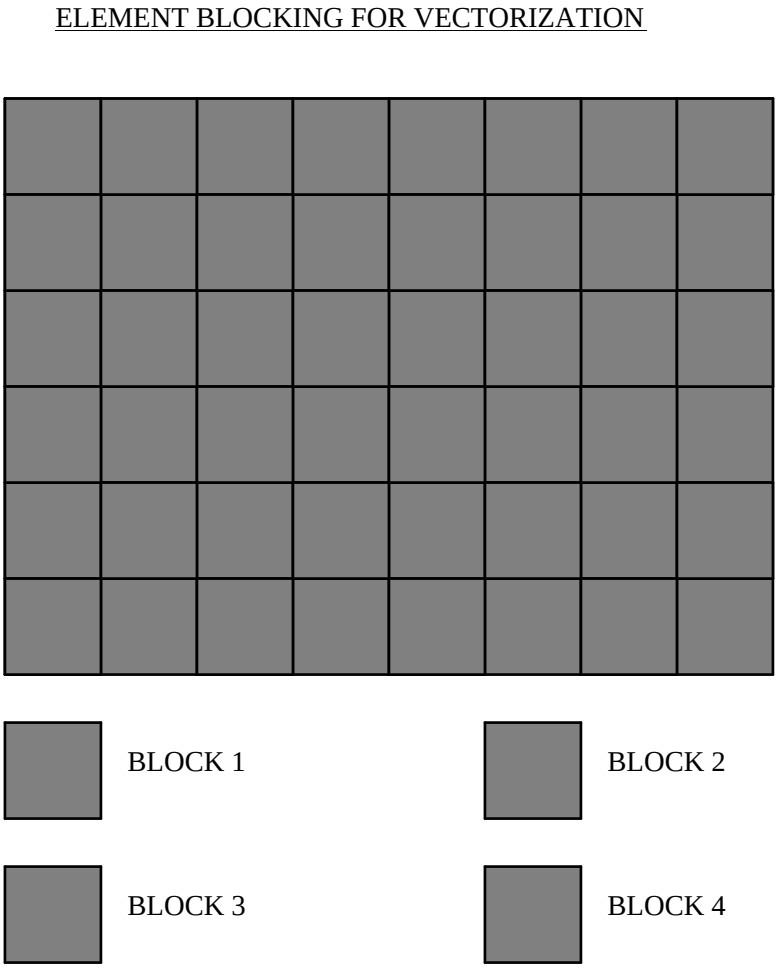


Figure 26.1 Group of 48 elements broken into 4 disjoint blocks.

The current implementation was strongly motivated by Benson [1989] and by work performed at General Motors [Ginsberg and Johnson 1988, Ginsberg and Katnik 1989], where it was shown that substantial improvements in execution speed could be realized by blocking the elements in the force assembly. Katnik implemented element sorting in a public domain version of DYNA3D for the Belytschko shell element and added compiler directives to force vectorization of the scatter operations associated with the addition of element forces into the global force vector. The sorting was performed immediately after the elements were read in so that subsequent references to the stored element data were sequential. Benson performed the sorting in the element loops via indirect addressing. In LS-DYNA3D the published GM approach is taken.

Implementation of the vectorization of the scatter operations is implemented in for all elements including the solid, shell, membrane, beam, and truss elements.
The sorting is completely transparent to the user.

26.2 Parallelization

In parallelization the biggest hurtle is overcoming Amdahl's law for multitasking [Cray Research Inc. 1990]

$$S_m = \frac{1}{f_s + \frac{f_p}{N}}$$

where

S_m = maximum expected speedup from multitasking
N = number of processors available for parallel execution
f_p = fraction of a program that can execute in parallel
f_s = fraction of a program that is serial

Table 26.1 shows that to obtain a speed factor of four on eight processors it is necessary to have eighty-six percent of the job running in parallel. Obviously, to gain the highest speed factors the entire code must run in parallel.

LS-DYNA3D has been substantially reorganized to permit parallel execution on computers with multiple processors such as the Cray-YMP and the Convex C-240. Major affected portions include the elements, surface to surface contact, and single surface contact. which typically can account for 85% of the CPU charges in a crashworthiness application. Several areas still remain to be parallelized including the stonewalls, and rigid bodies.

In 1986, Benson parallelized the brick elements in DYNA3D using the Cray Multi-Tasking. The extensive modifications that were required destroyed portablility and far from impressive speed-ups were attained. Fortunately, we learned much from this experience (mainly what not to do) and this has permitted us to carefully prepare the internal structure for the parallel version which is now being installed at selected sites as a

%	N=2	N=4	N=8	N=16	N=32	N=64	N=128	N=256
86.0%	1.75	2.82	4.04	5.16	5.99	6.52	6.82	6.98
90.0%	1.82	3.08	4.71	6.40	7.80	8.77	9.34	9.66
92.0%	1.85	3.23	5.13	7.27	9.20	10.60	11.47	11.96
94.0%	1.89	3.39	5.63	8.42	11.19	13.39	14.85	15.71
96.0%	1.92	3.57	6.25	10.00	14.29	18.18	21.05	22.86
98.0%	1.96	3.77	7.02	12.31	19.75	28.32	36.16	41.97
99.0%	1.98	3.88	7.48	13.91	24.43	39.26	56.39	72.11
99.2%	1.98	3.91	7.58	14.29	25.64	42.55	63.49	84.21
99.4%	1.99	39.3	7.68	14.68	26.98	46.44	72.64	101.19
99.6%	1.99	3.95	7.78	15.09	28.47	51.12	84.88	126.73
99.7%	1.99	3.96	7.84	15.31	29.28	53.83	92.69	145.04
99.8%	2.00	3.98	7.89	15.53	30.13	56.84	102.07	169.54
99.9%	2.00	3.99	7.94	15.76	31.04	60.21	113.58	203.98
100.0%	2.00	4.00	8.00	16.00	32.00	64.00	128.00	256.00

Table 26.1. Maximum theoretical speedup S_m, on N CPUs with parallelism [Cray Research Inc. 1990].

development version (Version 91.1). During the past several years Cray Research has developed auto-tasking for their computers with the advantages that standard FORTRAN is used for portability, DO loops will run in parallel if a compiler directive preceding the do loop is present, and the code is organized to minimize the overhead of initiating a process by using large element blocks.

In the element loops element blocks with vector lengths of 64 or some multiple are assembled and sent to separate processors. All elements are processed in parallel. On the average a speed factor of 7.8 has been attained in each element class corresponding to 99.7% parallelization.

A significant complication in parallelizing code is that a variable can sometimes be updated simultaneously by different processors with incorrect results. To force the processors to access and update the variable in a sequential manner, a GUARD compiler directive must be introduced. This results in an interruption to the parallel execution and can create a bottleneck. By sorting the data in the parallel groups or by allocating

additional storage it is usually possible to eradicate the GUARDS from the coding. The effort may not be worth the gains in execution speed.

The element blocks are defined at the highest level and each processor updates the entire block including the right hand side force assembly. The user currently has two options on the YMP: GUARD compiler directives prevent simultaneous updates of the RHS vector (recommended for single CPU processors or when running in a single CPU mode on a multi-processor), or assemble the right hand side in parallel and let LS-DYNA3D prevent conflicts between CPU's. This provides the highest speed and is recommended, i.e., no GUARDS.

The speed up in execution is dependent on having a large problem. Parallelization has been done with vector machines as the target machine where vector speed up is typically 10 times or greater over scalar. Vectorization comes first. If the problem is large enough then parallelization is automatic. Vector lengths are 128 so, for example, if 256 beam elements are used only a factor of 2 in speed can be anticipated while processing beam elements. Large contact surfaces will run in parallel, small surfaces having under 100 segments will not. The speed up in the contact subroutines has only registered 7 on 8 processors due to the presence of GUARD statements around the force assembly. Because real models often use many special options that will not even vectorize efficiently it is unlikely that more than 95% of a given problem will run in parallel-at least in the near future.

27. AIRBAGS

Additional information on the airbag modeling and comparisons with experimental data can be found in a report [Hallquist, Stillman, Hughes, and Tarver 1990] based on research sponsored by the Motor Vehicles Manufacturers Association (MVMA).

27.1 Control Volume Modeling

A direct approach for modeling the contents of the airbag would be to discretize the interior of the airbag using solid elements. The total volume and pressure-volume relationship, of the airbag would then be the sum of all the elemental contributions. Although this direct approach could be applied in a straight forward manner to an inflated airbag, it would become very difficult to implement during the inflation phase of the airbag deployment. Additionally, as the model is refined, the solid elements would quickly overwhelm all other computational costs and make the numerical simulations prohibitively expensive.

An alternative approach for calculating the airbag volume, that is both applicable during the inflation phase and less computationally demanding, treats the airbag as a control volume. The control volume is defined as the volume enclosed by a surface. In the present case the 'control surface' that defines the control volume is the surface modeled by shell or membrane elements comprising the airbag fabric material.

Because the evolution of the control surface is known, i.e. the position, orientation, and current surface area of the airbag fabric elements are computed and stored at each time step, we can take advantage of these properties of the control surface elements to calculate the control volume, i.e. the airbag volume. The area of the control surface can be related to the control volume through Green's Theorem

$$\iiint \phi \frac{\partial \psi}{\partial x} dxdydz = -\iiint \psi \frac{\partial \phi}{\partial x} dxdydz + \oint \phi \psi n_x d\Gamma \tag{27.1}$$

where the first two integrals are integrals over a closed volume, i.e dv = dxdydz, the last integral is an integral over the surface enclosing the volume, and n_x is the direction cosine between the surface normal and the x direction (corresponding to the x-partial derivative); similar forms can be written for the other two directions. The two arbitrary functions ϕ and ψ need only be integrated over the volume and surface.

The integral form of the volume can be written as

$$V = \iiint dxdydz \tag{27.2}$$

Comparing the first of the volume integrals in Equation (27.1) to Equation (27.2), we can easily obtain the volume integral from Equation (27.1) by choosing for the two arbitrary functions

$$\phi = 1 \tag{27.3}$$

$$\psi = x \tag{27.4}$$

leading to

$$V = \int\int\int dxdydz = \oint x n_x d\Gamma \tag{27.5}$$

The surface integral in Equation (27.5) can be approximated by a summation over all the elements comprising the airbag, i.e.

$$\oint x n_x d\Gamma \approx \sum_{i=1}^{N} \bar{x}_i n_{ix} A_i \tag{27.6}$$

where for each element i: $\bar{x}_i$ is the average x coordinate, n_{ix} is the direction cosine between the elements normal and the x direction, and A_i is the surface area of the element.

Although Equation (27.5) will provide the exact analytical volume for an arbitrary direction, i.e. any n, the numerical implementation of Equation (27.5), and its approximation Equation (27.6), has been found to produce slightly different volumes, differing by a few percent, depending on the choice of directions: if the integration direction is nearly parallel to a surface element, i.e. the direction cosine is nearly zero, numerical precision errors affect the volume calculation. The implementation uses as an integration direction, a direction that is parallel to the maximum principle moment of inertia of the surface. Numerical experiments have shown this choice of integration direction produces more accurate volumes than the coordinate or other principle inertia directions.

Because airbag models may contain holes, e.g. holes for inflation and deflation, and Green's Theorem only applies to closed surfaces, a special treatment is needed for calculating the volume of airbags with holes. This special treatment consists of the following:

- The n-sized polygon defining the hole is identified automatically, using edge locating algorithms in LS-DYNA3D.
- The n-sized polygon is projected onto a plane, i.e. it is assumed to be flat; this is a good approximation for typical airbag hole geometries. Planar symmetry should work with the control volume capability for one symmetry plane.
- The area of the flat n-sided polygon is calculated using Green's Theorem in two dimensions.
- The resulting holes are processed as another surface element in the airbag control volume calculation.

27.2 Equation of State Model

As explained above, at each time step in the calculation the current volume of the airbag is determined from the control volume calculation. The pressure in the airbag corresponding to the control volume is determined from an equation of state (EOS) that relates the pressure to the current gas density (volume) and the specific internal energy of the gas.

The equation of state used for the airbag simulations is the usual 'Gamma Law Gas Equation of State',

$$p = (k - 1)\,\rho e \tag{27.7}$$

where p is the pressure, k is a constant defined below, ρ is the density, and e is the specific internal energy of the gas. The derivation of this equation of state is obtained from thermodynamic considerations of the adiabatic expansion of an ideal gas. The incremental change in internal energy, dU, in n moles of an ideal gas due to an incremental increase in temperature, dT, at constant volume is given by (see for example page 42 of Reference [1]),

$$dU = nc_v dT \tag{27.8}$$

where c_v is the specific heat at constant volume. Using the ideal gas law we can relate the change in temperature to a change in the pressure and total volume, v, as

$$d(pv) = nRdT \tag{27.9}$$

where R is the universal gas constant. Solving the above for dT and substituting the result into Equation (27.8) gives

$$dU = \frac{c_v d(pv)}{R} = \frac{d(pv)}{(k-1)} \tag{27.10}$$

where we have used the relationship

$$R = c_p - c_v \tag{27.11}$$

and the notation

$$k = \frac{c_p}{c_v} \tag{27.12}$$

Equation (27.10) may be rewritten as

$$dU = \frac{\rho_0 v_0}{k-1} d\left(\frac{p}{\rho}\right) \tag{27.13}$$

and integrated to yield

$$e = \frac{U}{\rho_0 v_0} = \frac{p}{\rho(k-1)} \tag{27.14}$$

Solving for the pressure

$$p = (k - 1)\, \rho e \tag{27.15}$$

The equation of state and the control volume calculation can only be used to determine the pressure when the specific internal energy is also known. The evolution equation for the internal energy is obtained by assuming the change in internal energy is given by

$$dU = - pdv \tag{27.16}$$

where the minus sign is introduced to emphasize that the volume increment is negative when the gas is being compressed. This expression can be written in terms of the specific internal energy as

$$de = \frac{dU}{\rho_0 v_0} = -\frac{pdv}{\rho_0 v} \tag{27.17}$$

Next we divide the above by the equation of state, Equation (27.15), to obtain

$$\frac{de}{e} = -\frac{\rho(k-1)dv}{\rho_0 v_0} = -\frac{(k-1)dv}{v} \tag{27.18}$$

which may be integrated to yield

$$\ln e = (1 - k) \ln V \tag{27.19}$$

or evaluating at two states and exponentiating both sides yields

$$e_2 = e_1\left(\frac{v_2}{v_1}\right)^{(1-k)} \tag{27.20}$$

The specific internal energy evolution equation, Equation (27.20), the equation of state, Equation (27.15), and the control volume calculation completely define the pressure-volume relation for an inflated airbag.

27.3 Airbag Inflation Model

Airbag inflation models have been used for many years in occupant simulation codes such as CAL3D [Fleck, 1981].

The inflation model we chose to implement in LS-DYNA3D is due to Wang and Nefske[1988] and more recent improvements to the model in LS-DYNA3D were suggested by Wang [1992]. In their development they consider the mass flow due to the vents and leakage through the bag. We assume that the mass flow rate and the temperature of the gas going into the bag from an inflator are provided as a tabulated functions of time.

A pressure relation is defined:

$$Q = \frac{p_e}{p_2} \tag{27.21}$$

where p_e is the external pressure and p_2 is the internal pressure in the bag. A critical pressure relationship is defined as:

$$Q_{crit} = \left(\frac{2}{k+1}\right)^{k/k-1} \tag{27.22}$$

where k is the ratio of specific heats:

$$k = \frac{c_p}{c_v}$$

If

$$Q \le Q_{crit} \quad \text{then} \quad Q = Q_{crit} \tag{27.23}$$

Wang and Nefske define the mass flow through the vents and leakage by

$$\dot{m}_{23} = C_{23} A_{23} \frac{p_2}{R\sqrt{T_2}} Q^{1/k} \sqrt{2 g_c \left(\frac{kR}{k-1} \right) \left(1 - Q^{k-1/k} \right)}$$

and

$$\dot{m}'_{23} = C'_{23} A'_{23} \frac{p_2}{R\sqrt{T_2}} Q^{1/k} \sqrt{2 g_c \left(\frac{kR}{k-1} \right) \left(1 - Q^{k-1/k} \right)} \tag{27.24}$$

where C_{23}, A_{23}, C'_{23}, A'_{23}, R and g_c are the vent orifice coefficient, vent orifice area, the orifice coefficient for leakage, the area for leakage, the gas constant, and the gravitational conversion constant, respectively. The internal temperature of the airbag gas is denoted by T_2. We note that both A_{23} and A'_{23} can be defined as a function of pressure [Wang, 1992] or if they are input as zero they are computed within LS-DYNA3D. This latter option requires detailed modelling of the airbag with all holes included.

A uniform temperature and pressure is assumed; therefore, in terms of the total airbag volume V_2 and air mass, m_2, the perfect gas law is applied:

$$p_2 V_2 = m_2 R T_2 \tag{27.25}$$

Solving for T_2:

$$T_2 = \frac{p_2 V_2}{m_2 R} \tag{27.26}$$

and substituting Equation (27.26) into equations (27.24), we arrive at the mass transient equation:

$$\dot{m}_{out} = \dot{m}_{23} + \dot{m}'_{23} = \mu \sqrt{2 p_2 \rho} \sqrt{\frac{k\left(Q^{\frac{2}{k}} - Q^{k+1/k} \right)}{k-1}} \tag{27.27}$$

where

ρ = density of airbag gas
μ = bag characterization parameter
$\dot{m}_{out}$ = total mass flow rate out of bag

In terms of the constants used by Wang and Nefske:

$$\mu = \sqrt{g_c}\left(C_{23}A_{23} + C'_{23}A'_{23}\right) \tag{27.28}$$

We solved these equations iteratively, via function evaluation. Convergence usually occurs in 2 to 3 iterations.

The mass flow rate and gas temperature are defined in load curves as a function of time. Using the mass flow rate we can easily compute the increase in internal energy:

$$\dot{E}_{in} = c_p \dot{m}_{in} T_{in} \tag{27.29}$$

where T_{in} is the temperature of the gas flowing into the airbag. Initializing the variables pressure, p, density, ρ, and energy, E, to their values at time n, we can begin the iterations loop to compute the new pressure , p^{n+1}, at time n+1.

$$\begin{aligned}
p^{n+\frac{1}{2}} &= \frac{p^{n} + p^{n+1}}{2} \\
\rho^{n+\frac{1}{2}} &= \frac{\rho^{n} + \rho^{n+1}}{2} \\
E^{n+\frac{1}{2}} &= \frac{E^{n} + E^{n+1}}{2} \\
Q^{n+\frac{1}{2}} &= \max\left(\frac{p_e}{p_2^{n+\frac{1}{2}}}, Q_{crit}\right)
\end{aligned} \tag{27.30}$$

The mass flow rate out of the bag, $\dot{m}_{out}$ can now be computed:

$$\dot{m}_{out}^{n+\frac{1}{2}} = \mu\sqrt{2p_2^{n+\frac{1}{2}}\rho^{n+\frac{1}{2}}}\sqrt{\frac{k\left(Q^{n+\frac{1}{2}\,2/k} - Q^{n+\frac{1}{2}\,k+1/k}\right)}{k-1}} \tag{27.31}$$

where

$$p_2^{n+1/2} = p^{n+1/2} + p_e \tag{27.32}$$

and the total mass updated:

$$\begin{aligned} m^{n+1} &= m^n + \Delta t\left(\dot{m}_{in}^{n+1/2} - m_{out}^{n+1/2}\right) \\ m^{n+1/2} &= \frac{m^n + m^{n+1}}{2} \end{aligned} \tag{27.33}$$

The energy exiting the airbag is given by:

$$\dot{E}_{out}^{n+1/2} = \dot{m}_{out}^{n+1/2}\frac{E^{n+1/2}}{m^{n+1/2}} \tag{27.34}$$

we can now compute our new energy at time n+1

$$E^{n+1} = E^n + \Delta t\left(\dot{E}_{in}^{n+1/2} - \dot{E}_{out}^{n+1/2}\right) - p^{n+1/2}\Delta V^{n+1/2} \tag{27.35}$$

where $\Delta V^{n+1/2}$ is the change in volume from time n to n+1. The new pressure can now be computed:

$$p^{n+1} = (k-1)\frac{E^{n+1}}{V^{n+1}} \tag{27.36}$$

which is the gamma-law (where k=γ) gas equation. This ends the iteration loop.

28. DYNAMIC RELAXATION AND SYSTEM DAMPING

Dynamic relaxation allows LS-DYNA3D to approximate solutions to linear and nonlinear static or quasi-static processes. Control parameters must be selected with extreme care or bad results can be obtained. The current methods are not compatible with displacement or velocity boundary conditions, but various body loads, thermal loads, pressures, and nodal loads are allowed. The solutions to most nonlinear problems are path dependent, thus results obtained in the presence of dynamic oscillations may not be the same as for a nonlinear implicit code, and they may diverge from reality.

In LS-DYNA3D we have two methods of damping the solution. The first named "dynamic relaxation " is used in the beginning of the solution phase to obtain the initial stress and displacement field prior to beginning the analysis. The second is system damping which can be applied anytime during the solution phase either globally or on a material basis.

28.1 Dynamic Relaxation for Initialization

In this phase only a subset of the load curves is used to apply the static load which are flagged in the load curve section of the manual. The calculation begins and executes like a normal LS-DYNA3D calculation but with damping incorporated in the update of the displacement field.

Our development follows the work of Underwood [1986] and Papadrakakis [1981] with the starting point being the dynamic equilibrium equation, Equation (20.1) with the addition of a damping term, at time n:

$$\begin{aligned} M \cdot a^n + C \cdot v^n + Q^n(d) = 0 \\ Q^n(d) = F^n - P^n - H^n \end{aligned} \tag{28.1}$$

where we recall that M is the mass matrix, C is the damping matrix, n indicates the nth time step, a is the acceleration, v the velocity, and d is the displacement vector. With Δt as the fixed time increment we get for the central difference scheme:

$$v^{n+1/2} = \frac{\left(d^{n+1} - d^n\right)}{\Delta t} \quad ; \quad a^n = \frac{\left(v^{n+1/2} - v^{n-1/2}\right)}{\Delta t} \tag{28.2}$$

For v^n we can assume an averaged value

$$v^{n} = \frac{1}{2}\left(v^{n+1/2} + v^{n-1/2} \right) \tag{28.3}$$

and obtain

$$v^{n+1/2} = \left(\frac{1}{\Delta t} M + \frac{1}{2} C \right)^{-1} \left[\left(\frac{1}{\Delta t} M - \frac{1}{2} C \right) v^{n-1/2} - Q^{n} \right] \tag{28.4}$$

$$d^{n+1} = d^{n} + \Delta t \cdot v^{n+1/2} \tag{28.5}$$

In order to preserve the explicit form of the central difference integrator, M and C must be diagonal. For the dynamic relaxation scheme C has the form

$$C = c \cdot M \tag{28.6}$$

If Equation (28.6) is substituted into (28.4) the following form is achieved

$$v^{n+1/2} = \frac{2 - c \cdot \Delta t}{2 + c \cdot \Delta t} v^{n-1/2} + \frac{2\Delta t}{2 + c\Delta t} \cdot M^{-1} \cdot Q^{n} \tag{28.7}$$

Since M is diagonal, each solution vector component may be computed individually form

$$v_i^{n+1/2} = \frac{2 - c\Delta t}{2 + c\Delta t} v_i^{n-1/2} + \frac{2\Delta t}{2 + c\Delta t} \frac{Q_i^{n}}{m_i} \tag{28.8}$$

As a starting procedure it is suggested by Underwood

$$v^{o} = 0 \quad ; \quad d^{o} = 0 \tag{28.9}$$

Since the average value is used for v^{n}, which must be zero at the beginning for a quasi-static solution

$$v^{-1/2} = -v^{1/2} \tag{28.10}$$

thus the velocity at time $+1/2$ is

$$v^{1/2} = -\frac{\Delta t}{2} M^{-1} Q^{o} \tag{28.11}$$

A damping coefficient must now be selected to obtain convergence to the static solution in minimal time. The best estimate for damping values is based on the frequencies of the structure. One choice is to focus on an optimal damping parameter as suggested by Papadrakakis [1981]. Then dynamic relaxation is nothing else but a critically damped dynamic system

$$C = C_{cr} = 2 \cdot \omega_{min} \cdot m \tag{28.12}$$

with m as modal mass. The problem is finding the dominant eigenvalue in the structure related to the "pseudo-dynamic" behavior of the structure. As the exact estimate would be rather costly and not fit into the explicit algorithm, an estimate must be used. Papadrakakis suggests

$$\lambda_D = \frac{\left\| d^{n+1} - d^n \right\|}{\left\| d^n - d^{n-1} \right\|} \tag{28.13}$$

When this quantity has converged to an almost constant value, the minimum eigenvalue of the structure can be estimated:

$$\omega_{min}^2 = -\frac{\left(\lambda_D^2 - \lambda_D \cdot \beta + \alpha\right)}{\lambda_D \cdot \gamma} \tag{28.14}$$

where

$$\alpha = \frac{2 - c\Delta t}{2 + c\Delta t}$$
$$\beta = \alpha + 1$$
$$\gamma = \frac{2\Delta t^2}{2 + c\Delta t}$$

The maximum eigenvalue determines the time step and is already known from the model

$$\omega_{max}^2 = \frac{4.0}{(\Delta t)^2} \tag{28.15}$$

Now the automatic adjustment of the damping parameter closely follows the paper of Papadrakakis, checking the current convergence rate compared to the optimal

convergence rate. If the ratio is reasonably close, then an update of the iteration parameters is performed.

$$c = \frac{4.0}{\Delta t} \frac{\sqrt{\omega_{\min}^2 \cdot \omega_{\max}^2}}{\left(\omega_{\min}^2 + \omega_{\max}^2\right)} \tag{28.16}$$

As is clearly visible from Equation (28.16) the value of highest frequency has always a rather high influence on the damping ratio. This results in a non-optimal damping ratio, if the solution is dominated by the response in a very low frequency compared to the highest frequency of the structure. This is typically the case in shell structures, when bending dominates the solution. It was our observation that the automatic choice following Papadrakakis results in very slow convergence for such structures, and this is also mentioned by Underwood for similar problems. The damping ratio should then be fully adjusted to the lowest frequency by hand by simply choosing a rather high damping ratio. An automatic adjustment for such cases is under preparation. For structures with dominant frequencies rather close to the highest frequency, convergence is really improved with the automatically adjusted parameter.

If the automated approach is not used then we apply the damping as

$$v^{n+1/2} = \eta v^{n-1/2} + a^n \Delta t \tag{28.20}$$

where η is an input damping factor (defaulted to .995). The factor, η, is equivalent to the corresponding factor in Equations (28.7- 28.8).

The relaxation process continues until a convergence criterion based on the global kinetic energy is met, i.e., convergence is assumed if

$$E_{ke} < cvtol \cdot E_{ke}^{\max} \tag{28.21}$$

where cvtol is the convergence tolerance (defaulted to .001). The kinetic energy excludes any rigid body component. Initial velocities assigned in the input are stored during the relaxation. Once convergence is attained the velocity field is initialized to the input values. A termination time for the dynamic relaxation phase may be included in the input and is recommended since if convergence fails, LS-DYNA3D will continue to execute indefinitely.

28.2 System Damping

With system damping the Equation (20.2) is modified as:

$$a^n = M^{-1}\left(P^n - F^n + H^n - F^n_{damp} \right) \tag{28.22}$$

where

$$F^n_{damp} = D_s m v \tag{28.23}$$

As seen from Figure 28.1 and as discussed above the best damping constant for the system is usually the critical damping constant: Therefore,

$$D_s = 2\,\omega_{min} \tag{28.24}$$

is recommended.

28.3 Dynamic RelaxationHow Fast Does it Converge?

The number of cycles required to reduce the amplitude of the dynamic response by a factor of 10 can be approximated by [see Stone, Krieg, and Beisinger 1985]

$$ncycle = 1.15\frac{\omega_{max}}{\omega_{min}} \tag{28.25}$$

Structural problems which involve shell and beam elements can have a very large ratio and consequently very slow convergence.

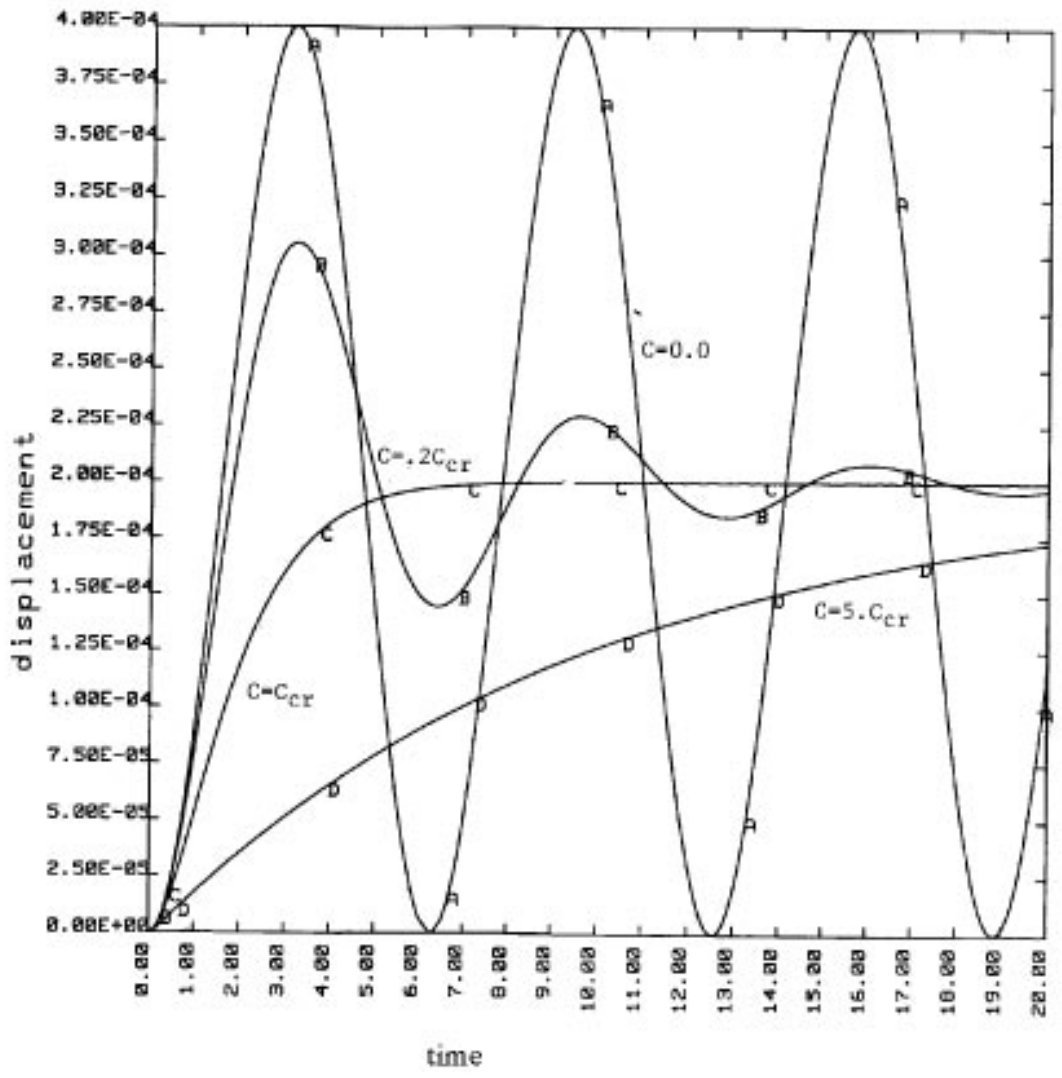


Figure 28.1 Displacement versus time curves with a variety of damping coefficients applied to a one degree-of-freedom oscillator.

29. HEAT CONDUCTION

LS-DYNA3D can be used to solve for the steady state or transient temperature field on three dimensional geometries. Material properties may be temperature dependent and either isotopic or orthotropic. A variety of time and temperature dependent boundary conditions can be specified including temperature, flux, convection, and radiation. The implementation of heat conduction into LS-DYNA3D is based on the work of Shapiro [1985].

29.1 Conduction of Heat in an Orthotropic Solid

The differential equation of conduction of heat in a three-dimensional solid is given by

$$\rho c \frac{\partial \theta}{\partial t} = \frac{\partial}{\partial x}\left[k_x \frac{\partial \theta}{\partial x}\right] + \frac{\partial}{\partial y}\left[k_y \frac{\partial \theta}{\partial y}\right] + \frac{\partial}{\partial y}\left[k_z \frac{\partial \theta}{\partial z}\right] + q_g \text{ in } \Omega, \tag{29.1}$$

subject to the boundary condition

$$k_x \frac{\partial \theta}{\partial x} n_x + k_y \frac{\partial \theta}{\partial y} n_y + k_z \frac{\partial \theta}{\partial z} n_z + \beta\theta = \gamma \text{ on } \Gamma, \tag{29.2}$$

and with the initial condition:

$$\theta = \theta(x, y, z) \text{ at } t = t_0 . \tag{29.3}$$

Equations (29.1-29.3) represent the strong form of a boundary value problems to be solved for the temperature field within the solid.

DYNA3D employs essentially the same theory as TOPAZ [Shapiro, 1985] in solving Equation (29.1) by the finite element method. Those interested in a more detailed description of the theory are referred to the TOPAZ User's Manual. Brick elements are integrated with a 2×2×2 Gauss quadrature rule, with temperature dependence of the properties accounted for at the Gauss points. Time integration is performed using a generalized trapezoidal method shown by Hughes to be unconditionally stable for nonlinear problems. Fixed point iteration with relaxation is used to satisfy equilibrium in nonlinear problems.

The finite element method provides the following equations for the numerical solution of Equations (29.1-29.3).

$$\left[\frac{C_{n+\alpha}}{\Delta t}+\alpha H_{n+\alpha}\right]\{\theta_{n+1}-\theta_n\}=\{F_{n+\alpha}-H_{n+\alpha}\theta_n\} \tag{29.4}$$

where

$$[C]=\sum_e\left[C_{ij}^e\right]=\sum_e\int_{\Omega^e}N_i\,\rho c\,N_j\,d\Omega \tag{29.5}$$

$$[H]=\sum_e\left[H_{ij}^e\right]=\sum_e\left[\int_{\Omega^e}\nabla^T N_i K\nabla N_j\,d\Omega+\int_{\Gamma^e}N_i\beta N_j\,d\Gamma\right] \tag{29.6}$$

$$[F]=\sum_e\left[F_i^e\right]=\sum_e\left[\int_{\Omega^e}N_i q_g\,d\Omega+\int_{\Gamma^e}N_i\gamma\,d\Gamma\right] \tag{29.7}$$

The parameter α is take to be in the interval [0,1]. Some well known members of this α-family are

α	Method
0	forward difference; forward Euler
$1/2$	midpoint rule; Crank-Nicolson
$2/3$	Galerkin
1	backward

29.2 Thermal Boundary Condtions

Boundary conditions are represented by

$$k_x\frac{\partial\theta}{\partial x}n_x+k_y\frac{\partial\theta}{\partial y}n_y+k_z\frac{\partial\theta}{\partial z}n_z=\gamma-\beta\theta=\dot{q}'' \tag{29.8}$$

By convection heat flow is positive in the direction of the surface outward normal vector. Surface definition is in accordance with the right hand rule. The outward normal vector point to the right as one progresses from node N_1 to N_2 to N_3 and finally to N_4 as shown in Figure 29.1.

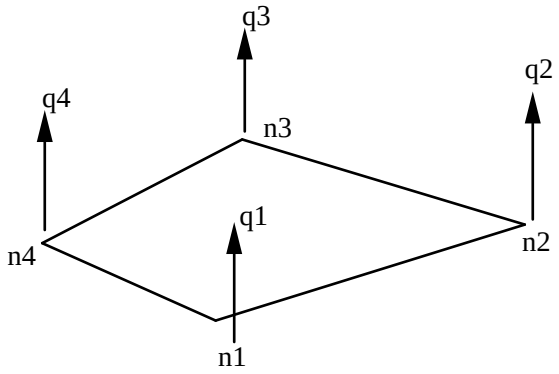


Figure 29.1 Convection heat flow.

Boundary condition can be functions of temperature or time. More than one boundary condition can be specified over the same surface such as in a case of combined convection and radiation. For situations where it is desired to specify adiabatic (i.e. $\dot{q}'' = 0$) conditions such as at an insulated surface or on a line of symmetry, no boundary condition need be specified. This is the default boundary condition in LS-DYNA3D.

Temperature boundary condition can be specified on any node whether on the physical boundary or not.

Flux: Sets $\dot{q}'' = q_f$ where q_f is defined at the node point comprising the flux boundary condition surfaces.

Radiation: A radiation boundary condition is calculated using a radiant-heat-transfer coefficient. Set $\dot{q}'' = h_r$ (T –were h_r is a radiant-heat-transfer coefficient defined as

The exchange factor, **F**, is a characterization of the effect of the system geometry, emissivity and reflectivity on the capability of radiative transport between surfaces. The radiation boundary condition data cards require specification of the product, $f = \mathbf{F}\sigma$, and T for the boundary surface.

Convection: A convection boundary condition is calculated using $\dot{q}'' = h(T - T_\infty)$

where

h	heat transfer coefficient.
$(T - T_\infty)$	temperature potential.

DYNA3D evaluates h at the film temperature

$$T = \frac{1}{2}\left(T_{surf} + T_{\infty}\right) . \tag{29.9}$$

29.3 Thermal Energy Balances

Various energy terms are printed and written into the plot file for post processing using the code TAURUS. The energy terms are:

- change in material internal energy for time step,
- change in material internal energy from initial time,
- heat transfer rates on boundary condition surfaces,
- heat transfer rates on enclosure radiation surfaces,
- x, y, and z fluxes at all nodes.

29.4 Heat Generation

Volumetric heat generation rates may be specified by element, by material, or both (in which case the effect is additive). Volumetric heat generation rates can be function of time or temperature.

29.5 Initial Conditions

Initial temperature condition can be specified on the nodal data input cards or on the nodal temperature initial condition cards. If no temperatures are specified, the default is 0. For nonlinear steady state problems the temperature initial condition serves as a first guess for the equilibrium iterations.

29.6 Material Properties

Heat capacity and thermal conductivity may be function of temperature. Since the density and heat capacity appear only as a product in the governing equations, the temperature dependence of the density may be included in the temperature dependence of the heat capacity. Material properties are evaluated at the element Gauss point temperature.

The thermal conductivity may be either isotropic or orthotropic. For an orthotropic material, the three material axes $\left(x_1', x_2', x_3'\right)$ are orthogonal and the thermal conductivity tensor K′ is diagonal.

The thermal conductivity tensor K in the global coordinate system is related by

$$K_{ij} = K'_{ij}\beta_{mi}\beta_{nj}$$

where

$$\beta_{ij} = \cos\left(x'_i, x_j\right)$$

29.7 Nonlinear Analysis

In a nonlinear problem, C, H may F any be functions of θ and iteration is required to solved Equation 4. Functional iteration with under relaxation is used. The nonlinear solution scheme consists of two steps. The first step call "reformation" is the assembly and triangularization of the coefficient matrix in equation 4.

$$\left[\frac{C_{n+\alpha}}{\Delta t} + \alpha H_{n+\alpha}\right] \tag{5}$$

d on experience or a trial-and-error process.

In a steady state nonlinear problem, an initial guess should be made of the final temperature distribution and included in the input file as an initial condition. If your guess is good, a considerable savings in computation time is achieved.

29.8 Transient Analysis

DYNA3D has a fixed thermal time step.

Units

Any consistent set of units with the governing equation may be used. Examples are:

Quantity	Units		
temperature	K	C	F
space	m	cm	ft
time	s	s	hr
density	kg/m^3	g/cm^3	Lb_m/ft^3
heat capacity	J/kg k	cal/g c	Btu/Lb_mF
thermal conductivity	W/m K	cal/s cm C	Btu/hr ft F
thermal generation	W/M^3	cal/s cm^3	Btu/hr ft^3
heat flux	W/m^2	cal/s cm^2	Btu/hr ft^2

This step is computationally expensive. The second step called an "equilibrium iteration" is the formation of the right hand vector $\{F - H_{n+\alpha}\theta_n\}$ in Equation 4 and back substitution to solve for $\{\theta_{n+1} - \theta_n\}$. This step is computationally inexpensive.

For strongly nonlinear problems (e.g. radiation boundary condition) it is necessary to perform a reformation for each equilibrium iteration. For weakly nonlinear problems (e.g. material property nonlinearities) it is computationally advantageous to perform a reformation only at the beginning of the time step and then perform as many equilibrium iterations as required for convergence. Further still, a reformation of Equation 5 can be performed and used over several time steps. The decision as to the number of reformation and equilibrium iteration is best to use on a particular problem must be made base

REFERENCES

Addessio, F.L., D.E. Carroll, J.K. Dukowicz, F.H. Harlow, J.N. Johnson, B.A.. Kashiwa, M.E. Maltrud, H.M. Ruppel, "CAVEAT: A Computer Code for Fluid Dynamics Problems with Large Distortion and Internal Slip," Rept. LA-10613-MS, UC-32, Los Alamos National Laboratory (1986)

Ahmad, S., Irons, B.M. and Zienkiewicz, O.C., "Analysis of Thick and Thin Shell Structures by Curved Finite Elements," Int. J. Numer. Meths. Eng. **2**, (1970)

Amsden, A. A., and Hirt, C. W., "YAQUI: An Arbitrary Lagrangian-Eulerian Computer Program for Fluid Flow at All Speeds," Los Alamos Scientific Laboratory, LA-5100, (1973).

Amdsden, A. A., Ruppel, H. M., and Hirt, C. W., "SALE: A Simplified ALE Computer Program for Fluid Flow at All Speeds," Los Alamos Scientific Laboratory, (1980).

Argyris, J.H., Kelsey, S., and Kamel, H., "Matrix Methods of Structural Analysis: A Precis of recent Developments," Matrix Methods of Structural Analysis, Pergamon Press , 1964.

Bammann, D. J., "Modeling the Temperature and Strain Rate Dependent Large Deformation of Metals," Proceedings of the 11th US National Congress of Applied Mechanics, Tucson, AZ., 1989.

Bammann, D. J. and Johnson, G., "On the Kinematics of Finite-Deformation Plasticity," Acta Mechanica **69**, 97-117 (1987).

Bammann, D.J., Chiesa, M.L., McDonald, A., Kawahara, W.A., Dike, J.J. and Revelli, V.D., "Predictions of Ductile Failure in Metal Structures," in AMD-Vol. 107, Failure Criteria and Analysis in Dynamic Response, edited by. H.E. Lindberg, 7-12 (1990).

Bathe, K. J., Finite Element Procedures in Engineering Analysis, Prentice-Hall (1982).

Bathe, K. J. and Wilson, E.L., Numerical Methods in Finite Element Analysis, Prentice-Hall (1976).

Bathe, K.J. and Dvorkin, E.N., "A Continuum Mechanics Based Four Node Shell Element for General Nonlinear Analysis," Int. J. Computer-Aided Eng. and Software, Vol. 1, 77-88, (1984).

Bazeley, G.P., Cheung, W.K., Irons, B.M. and Zienkiewicz, O.C., "Triangular Elements in Plate Bending—Conforming and Nonconforming Solutions in Matrix Methods and Structural Mechanics," Proc. Conf. on Matrix Methods in Structural Analysis, Rept. AFFDL-R-66-80, Wright Patterson AFB, 547-576 (1965).

Belytschko, T., "Transient Analysis," Structural Mechanics Computer Programs, edited by W. Pilkey et al., University Press of Virginia, pp. 255-276, 1974.

Belytschko, T., Lin, J., "A New Interaction Algorithm with Erosion for EPIC-3,", Contract Report BRL-CR-540, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, Maryland, (1985)

Belytschko, T., Lin, J. and Tsay, C.S., "Explicit Algorithms for Nonlinear Dynamics of Shells," Comp. Meth. Appl. Mech. Eng. **42**, 225-251 (1984) [**a**].

Belytschko, T., Stolarski,H. and Carpenter,N., "A C_0 Triangular Plate Element with One-Point Quadrature," International Journal for Numerical Methods in Engineering **20**, 787-802 (1984) [**b**].

Belytschko, T., Schwer, L. and Klein, M. J., "Large Displacement Transient Analysis of Space Frames," International Journal for Numerical and Analytical Methods in Engineering **11**, 65-84 (1977).

Belytschko, T. and Tsay, C.S., "Explicit Algorithms for Nonlinear Dynamics of Shells," AMD-Vol.48, ASME, 209-231 (1981).

Belytschko, T. and Tsay, C.S., "WHAMSE: A Program for Three-Dimensional Nonlinear Structural Dynamics," Report EPRI NP-2250, Project 1065-3, Electric Power Research Institute, Palo Alto, CA. (1982).

Belytschko, T., and Tsay, C. S., "A Stabilization Procedure for the Quadrilateral Plate Element with One-Point Quadrature," Int. J. Num. Method. Eng. **19**, 405-419 (1983).

Belytschko, T., Yen, H. R. and Mullen R. (1979), "Mixed Methods for Time Integration," Computer Methods in Applied Mechanics and Engineering, 17, pp. 259-175.

Belytschko, T., (1980), "Partitioned and Adaptive Algorithms for Explicit Time Integration," in Nonlinear Finite Element Analysis in Structural Mechanics, ed. by Wunderlich, W. Stein, E, and Bathe, J. J., pp. 572-584.

Belytschko, T., Wong, B.L. and Chiang, H.Y., "Improvements in Low-Order Shell Elements for Explicit Transient Analysis," Analytical and Computational Models of Shells, A.K. Noor, T. Belytschko, and J. Simo, editors, ASME, CED-Vol. 3, pp. 383-398, (1989).

Belytschko, T., Wong, B.L. and Chiang, H.Y., "Advances in One-Point Quadrature Shell Elements," Comp. Meths. Appl.. Mech. Eng., 96, 93-107, (1992).

Benson, D.J., "Vectorizing the Right-Hand Side Assembly in an Explicit Finite Element Program," Comp. Meths. Appl.. Mech. Eng., 73, 147-152, (1989).

Benson, D. J. and Hallquist, J.O., "A Simple Rigid Body Algorithm for Structural Dynamics Program," Int. J. Numer. Meth. Eng., 22, (1986).

Benson, D.J. and Hallquist J.O., "A Single Surface Contact Algorithm for the Postbuckling Analysis of Shell Structures," Comp. Meths. Appl.. Mech. Eng., 78, 141-163, (1990).

Benson, D. J., "Momentum Advection on a Staggered Mesh," Journal of Computational Physics, Vol. 100, No. 1, (1992).

Benson, D. J., "Vectorization Techniques for Explicit Arbitrary Lagrangian Eulerian Calculations," Computer Methods in Applied Mechanics and Engineering, (1992).

Blatz, P.J., and Ko, W.L., "Application of Finite Element Theory to the Deformation of Rubbery Materials," Trans. Soc. of Rheology **6**, 223-251 (1962).

Brooks, A. N., and Hughes, T. J. R., "Streamline Upwind/Petrov-Galerkin Formulations for Convection Dominated Flows with Particular Emphasis on the Incompressible Navier-Stokes Equations," Computer Methods in Applied Mechanics and Engineering, Vol. 32, pp. 199-259, (1982).

Burton, D.E. et al., "Physics and Numerics of the TENSOR Code," Lawrence Livermore National Laboratory, Internal Document UCID-19428, (July 1982).

Chang, F.K. and Chang, K.Y., "Post-Failure Analysis of Bolted Composite Joints in Tension or Shear-Out Mode Failure," J. of Composite Materials **21**, 809-833 (1987).[**a**]

Chang, F.K. and Chang, K.Y., "A Progressive Damage Model for Laminated Composites Containing Stress Concentration," J. of Composite Materials **21**, 834-855 (1987).[**b**]

Cochran, S.G. and Chan, J., "Shock Initiation and Detonation Models in One and Two Dimensions," University of California, Lawrence Livermore National Laboratory, Rept. UCID-18024 (1979).

Cohen, M. and Jennings, P.C., "Silent Boundary Methods for Transient Analysis", in Computational Methods for Transient Analysis, T. Belytschko and T.J.R. Hughes, editors, North-Holland, New York, pp. 301-360 (1983).

Cook, R. D., Concepts and Applications of Finite Element Analysis, John Wiley and Sons, Inc. (1974).

Couch, R., Albright, E. and Alexander, "The JOY Computer Code," University of California, Lawrence Livermore National Laboratory, Rept. UCID-19688 (1983).

Cray Research Inc., "CF77 Compiling System, Volume 4: Parallel Processing Guide," SG-3074 4.0, Mendota Heights, Mn, (1990).

DeBar, R. B., "Fundamentals of the KRAKEN Code," Lawrence Livermore Laboratory, UCIR-760, (1974).

Dobratz, B.M., "LLNL Explosives Handbook, Properties of Chemical Explosives and Explosive Simulants," University of California, Lawrence Livermore National Laboratory, Rept. UCRL-52997 (1981).

Englemann, B.E. and Whirley, R.G., "A New Explicit Shell Element Formulation for Impact Analysis," In Computational Aspects of Contact Impact and Penetration, Kulak, R.F., and Schwer, L.E., Editors, Elmepress International, Lausanne, Switzerland, 1991, pgs. 51-90.

Englemann, B.E., Whirley, R.G., and Goudreau, G.L., "A Simple Shell Element Formulation for Large-Scale Elastoplastic Analysis," In Analytical and Computational Models of Shells, Noor, A.K., Belytschko, T., and Simo, J.C., Eds., CED-Vol. 3, ASME, New York, New York, (1989).

Farhoomand, I., and Wilson, E.L., "A Nonlinear Finite Element Code for Analyzing the Blast Response of Underground Structures," U.S. Army Waterways Experiment Station, Contract Rept. N-70-1 (1970).

Feng, W.W. Private communication, Livermore, CA, 1993.

Flanagan, D.P. and Belytschko, T., "A Uniform Strain Hexahedron and Quadrilateral and Orthogonal Hourglass Control," Int. J. Numer. Meths. Eng. **17**, 679-706 (1981)

Fleck, J.T., "Validation of the Crash Victim Simulator," Volumes I through IV, Report No. DOT-HS-806 279 (1981).

Ginsberg, M. and Johnson, J.P., "Benchmarking the Performance of Physical Impact Simulation Software on Vector and Parallel Computers," Proc. of the Supercomputing 88: Vol. II, Science and Applications, Computer Society Press, (1988).

Ginsberg, M., and Katnik, R.B., "Improving Vectorization of a Crashworthiness Code," SAE Technical Paper No. 891985, Passenger Car Meeting and Exposition, Dearborn, Mi. (1989).

Giroux, E.D., "HEMP User's Manual," University of California, Lawrence Livermore National Laboratory, Rept. UCRL-51079 (1973).

Hallquist, J.O., “Preliminary User’s Manuals for DYNA3D and DYNAP (Nonlinear Dynamic Analysis of Solids in Three Dimension),” University of California, Lawrence Livermore National Laboratory, Rept. UCID-17268 (1976) and Rev. 1 (1979).[**a**]

Hallquist, J.O., “A Procedure for the Solution of Finite Deformation Contact-Impact Problems by the Finite Element Method,” University of California, Lawrence Livermore National Laboratory, Rept. UCRL-52066 (1976).

Hallquist, J.O., “A Numerical Procedure for Three-Dimensional Impact Problems,” American Society of Civil Engineering, Preprint 2956 (1977).

Hallquist, J.O., “A Numerical Treatment of Sliding Interfaces and Impact,” in: K.C. Park and D.K. Gartling (eds.) Computational Techniques for Interface Problems, AMD Vol. 30, ASME, New York (1978).

Hallquist, J.O., “NIKE2D: An Implicit, Finite-Element Code for Analyzing the Static and Dynamic Response of Two-Dimensional Solids,” University of California, Lawrence Livermore National Laboratory, Rept. UCRL-52678 (1979).[**b**]

Hallquist, J.O., “User's Manual for DYNA2D— An Explicit Two-Dimensional Hydrodynamic Finite Element Code with Interactive Rezoning,” University of California, Lawrence Livermore National Laboratory, Rept. UCID-18756 (1980).

Hallquist, J.O., “User's Manual for DYNA3D and DYNAP (Nonlinear Dynamic Analysis of Solids in Three Dimensions),” University of California, Lawrence Livermore National Laboratory, Rept. UCID-19156 (1981).[**a**]

Hallquist, J. O., “NIKE3D: An Implicit, Finite-Deformation, Finite-Element Code for Analyzing the Static and Dynamic Response of Three-Dimensional Solids,” University of California, Lawrence Livermore National Laboratory, Rept. UCID-18822 (1981).[**b**]

Hallquist, J.O., “DYNA3D User's Manual (Nonlinear Dynamic Analysis of Solids in Three Dimensions),” University of California, Lawrence Livermore National Laboratory, Rept. UCID-19156 (1982; Rev. 1: 1984; Rev. 2: 1986).

Hallquist, J.O., “Theoretical Manual for DYNA3D,” University of California, Lawrence Livermore National Laboratory, Rept. UCID-19401 (March, 1983).

Hallquist, J.O., “DYNA3D User's Manual (Nonlinear Dynamic Analysis of Solids in Three Dimensions),” University of California, Lawrence Livermore National Laboratory, Rept. UCID-19156 (1988, Rev. 4).

Hallquist, J.O., “DYNA3D User's Manual (Nonlinear Dynamic Analysis of Solids in Three Dimensions),” Livermore Software Technology Corporation, Rept. 1007 (1990).

Hallquist, J.O., Benson, D.J., and Goudreau, G.L., "Implementation of a Modified Hughes-Liu Shell into a Fully Vectorized Explicit Finite Element Code," Proceedings of the International Symposium on Finite Element Methods for Nonlinear Problems, University of Trondheim, Trondheim, Norway (1985).

Hallquist, J.O. and Benson, D.J., "A Comparison of an Implicit and Explicit Implementation of the Hughes-Liu Shell," Finite Element Methods for Plate and Shell Structures, T.J.R. Hughes and E. Hinton, Editors, 394-431, Pineridge Press Int., Swanea, U.K. (1986).

Hallquist, J.O. and Benson, D.J., "DYNA3D User's Manual (Nonlinear Dynamic Analysis of Solids in Three Dimensions)," University of California, Lawrence Livermore National Laboratory, Rept. UCID-19156 (1986, Rev. 2).

Hallquist, J.O. and Benson, D.J., "DYNA3D User's Manual (Nonlinear Dynamic Analysis of Solids in Three Dimensions)," University of California, Lawrence Livermore National Laboratory, Rept. UCID-19156 (1987, Rev. 3).

Hallquist, J.O., Stillman, D.W., Hughes, T.J.R., and Tarver, C.,"Modeling of Airbags Using MVMA/DYNA3D," LSTC Report, (1990)

Harten, A., "ENO Schemes with Subcell Resolution," J. of Computational Physics, Vol. ~83, pp. ~148-184, (1989).

Herrmann, L.R. and Peterson, F.E., "A Numerical Procedure for Viscoelastic Stress Analysis," Seventh Meeting of ICRPG Mechanical Behavior Working Group, Orlando, FL, CPIA Publication No. 177 (1968).

Hill, R., "A Theory of the Yielding and Plastic Flow of Anisotropic Metals," Proceedings of the Royal Society of London, Series A., **193,** p281 (1948).

Hughes, T.J.R., The Finite Element Method, Linear Static and Dynamic Finite Element Analysis, Prentice-Hall Inc., Englewood Cliffs, N.J. (1987).

Hughes, T.J.R., "Generalization of Selective Integration Procedures to Anisotropic and Nonlinear Media," Int. J. Numer. Meth. Eng. **15**, 9 (1980).

Hughes, T.J.R. and Carnoy, E., "Nonlinear Finite Element Shell Formulation Accounting for Large Membrane Strains," AMD-Vol.48, ASME, 193-208 (1981).

Hughes, T.J.R. and Liu, W.K., "Nonlinear Finite Element Analysis of Shells: Part I. Two-Dimensional Shells." Comp. Meths. Appl. Mechs. **27**, 167-181 (1981).

Hughes, T.J.R. and Liu, W.K., "Nonlinear Finite Element Analysis of Shells: Part II. Three-Dimensional Shells." Comp. Meths. Appl. Mechs. **27**, 331-362 (1981).

Hughes, T.J.R., Liu,W.K., and Levit, I., "Nonlinear Dynamics Finite Element Analysis of Shells." Nonlinear Finite Element Analysis in Struct. Mech., Eds. W. Wunderlich, E. Stein, and K.J. Bathe, Springer-Verlag, Berlin, 151–168 (1981).

Hughes, T.J.R., Taylor, R. L., Sackman, J. L., Curnier, A.C., and Kanoknukulchai, W., "A Finite Element Method for a Class of Contact-Impact Problems," J. Comp. Meths. Appl. Mechs. Eng. **8**, 249-276 (1976).

Hughes, T.J.R., and Winget, J., "Finite Rotation Effects in Numerical Integration of Rate Constitutive Equations Arising in Large-Deformation Analysis," Int. J. Numer. Meths. Eng. **15**, 1862-1867, (1980).

Hulbert, G.M. and Hughes, T.J.R.," Numerical Evaluation and Comparison of Subcycling Algorithms for Structural Dynamics," Technical Report, DNA-TR-88-8, Defense Nuclear Agency, Washington, D.C. (1988).

Johnson, G. C. and Bammann, D. J., "A discussion of stress rates in finite deformation problems, " Int. J. Solids Struct, **20**, 725-737 (1984).

Johnson, G.R. and Cook,W. H., "A Constitutive Model and Data for Metals Subjected to Large Strains, High Strain Rates and High Temperatures," presented at the Seventh International Symposium on Ballistics, The Hague, The Netherlands, April 1983.

Johnson, C., Navert, U., and Pitkaranta, J., "Finite Element Methods for Linear Hyperbolic Problems," Computer Methods in Applied Mechanics and Engineering, Vol. 45, pp. 285-312, (1984).

Jones, N., "Structural Aspects of Ship Collisions," Chapter 11, in Structural Crashworthiness, Eds. N. Jones and T Wierzbicki, Butterworths, London, 308-337, (1983).

Jones, R.M., Mechanics of Composite Materials, Hemisphere Publishing Co., New York, (1975)

Kenchington, G. J., "A Non-Linear Elastic Material Model for DYNA3D," Proceedings of the DYNA3D Users Group Conference, September 1988, published by Boeing Computer Services (Europe) Limited.

Kennedy, J. M., Belytschko,T., and Lin, J. I., "Recent Developments in Explicit Finite Element Techniques and their Applications to Reactor Structures," Nuclear Engineering and Design **97**, 1-24 (1986).

Kosloff, D. and Frazier, G.A.,"Treatment of Hourglass Patterns in Low Order Finite Element Codes," Int. J. Numer. Anal. Meth. Geomech. **2**, 57-72 (1978)

Kreyszig, E., Advanced Engineering Mathematics, John Wiley and Sons, New York, New York (1972).

Krieg, R.D.,"A Simple Constitutive Description for Cellular Concrete," Sandia National Laboratories, Albuquerque, NM, Rept. SC-DR-72-0883 (1972).

Krieg, R.D. and Key, S.W., Implementation of a Time Dependent Plasticity Theory into Structural Computer Programs, Vol. 20 of Constitutive Equations in Viscoplasticity: Computational and Engineering Aspects (American Society of Mechanical Engineers, New York, N.Y., 1976), pp. 125-137.

Landshoff, R., "A Numerical Method for Treating Fluid Flow in the Presence of Shocks," Los Alamos Scientific Laboratory, Rept. LA-1930 (1955).

Lee, E.L. and Tarver, C.M., "Phenomenological Model of Shock Initiation in Heterogeneous Explosives," Phys. of Fluids, **23**, 2362 (1980).

Liu, W. K., Chang, H., and Belytschko, T., "Arbitrary Lagrangian-Eulerian Petrov-Galerkin Finite Elements for Nonlinear Continua," Computer Methods in Applied Mechanics and Engineering, to be published.

Lysmer, J. and Kuhlemeyer, R.L., "Finite Dynamic Model for Infinite Media", J. Eng. Mech. Div. ASCE, 859-877 (1969).

Maenchen, G. and Sack, S., "The Tensor Code," Meth. Comp. Phys. **3**, (Academic Press), 181-263 (1964).

Maffeo, R., "TRANCITS Program User's Manual," General Electric Company, Cincinnati, OH, NASA Report CR-174891, (1985).

Maffeo, R., "Burner Liner Thermal/Structural Load Modeling," General Electric Company, Cincinnati, OH, NASA Report CR-174892, (1984).

Maker, B. N. Private communication, Lawrence Livermore National Laboratory. Dr. Maker programmed and implemented the compressible Mooney-Rivlin rubber model.

Marchertas, A. H. and Belytschko,T. B., "Nonlinear Finite Element Formulation for Transient Analysis of Three Dimensional Thin Structures," Report ANL-8104, LMFBR Safety, Argonne National Laboratory, Argonne, IL 1974.

Margolin, L. G., personal communication to D. Benson, (1989).

McGlaun, personal communication to D. Benson, Sandia National Laboratories, (1990).

McGlaun, J. M., Thompson, S. L., and Elrick, M. G., "CTH: A Three-Dimensional Shock Wave Physics Code," Proc. of the 1989 Hypervelocity Impact Symposium, (1989).

Mindlin, R.D., "Influence of Rotary Inertia and Shear on Flexural Motions of Isotropic, Elastic Plates," J. Appl. Mech. **18**, 31-38 (1951).

Mizukami, A., and Hughes, T. J. R., "A Petrov-Galerkin Finite Element Method for Convection Dominated Flows: An Accurate Upwinding Technique for Satisfying the Maximum Principle," Computer Methods in Applied Mechanics and Engineering, Vol. 50, pp. 181-193, (1985).

Nagtegaal, J.C. , Parks, D.M., and Rice J.R., "On Numerically Accurate Finite Element Solution in the Fully Plastic Range", Computer Methods in Applied Mechanics and Engineering, 4, 153 (1974).

Newman, W.M., and Sproull, R.F., Principles of Interactive Computer Graphics, McGraw-Hill, New York, (1979)

Newmark, N., "A Method of Computation for Structural Dynamics," Journal of the Engineering Mechanics Division, Proceedings of the American Society of Civil Engineers, 67-94, 1959.

Neilsen, M.K., Morgan, H.S., and Krieg, R.D., "A Phenomenological Constitutive Model for Low Density Polyurethane Foams," Rept. SAND86-2927, Sandia National Laboratories, Albuquerque, N.M., (1987)

Noh, W.F., "Numerical Methods in Hydrodynamic Calculations," University of California, Lawrence Livermore National Laboratory, Rept. UCRL-52112 (1976)

Ogden, R.W., *Non-Linear Elastic Deformations*, Ellis Horwood Ltd., Chichester, Great Britian, (1984).

Papadrakakis, M. "A Method for the Automated Evaluation of the Dynamic Relaxation Parameters," Comp. Meth. Appl. Mech. Eng. 25, pgs. 35-48 (1981).

Pian, T.H.H. and Sumihara, K., "Rational Approach for Assumed Stress Elements," Int. J. Num. Meth. Eng., Vol. 20, 1685-1695, (1984).

Przemieniecki, J. S., Theory of Matrix Structural Analysis, McGraw-Hill Book Company, New York, NY, 1986.

Richards, G.T., "Derivation of a Generalized von Neumann Pseudo-Viscosity with Directional Properties," University of California, Lawrence Livermore National Laboratory, Rept. UCRL-14244 (1965).

Richtmyer, R.D. and Morton,K.W., Difference Equations for Initial-Value Problems, Interscience Publishers, New York, 1967.

Schwer, L.E., Cheva, W., and Hallquist, J.O., "A Simple Viscoelastic Model for Energy Absorbers Used in Vehicle-Barrier Impacts," In Computational Aspects of Contact Impact and Penetration, Kulak, R.F., and Schwer, L.E., Editors, Elmepress International, Lausanne, Switzerland, 1991, pgs. 99-117.

Shapiro, A. B. “TOPAZ3D - A Three Dimensional Finite Element Heat Transfer Code,” University of California, Lawrence Livermore National Laboratory, Report UCID-20481 (1985).

Shapiro, A. B., “REMAP: a Computer Code That Transfers Node Information Between Dissimilar Grids,” Lawrence Livermore National Laboratory, UCRL-ID-104090, (1990).

Steinberg, D.J. and Guinan, M.W., “A High-Strain-Rate Constitutive Model for Metals,” University of California, Lawrence Livermore National Laboratory, Rept. UCRL-80465 (1978).

Stillman, D. W., and Hallquist, J. O., “LS-INGRID: A Pre-Processor and Three-Dimensional Mesh Generator for the Programs LS-DYNA3D, LS-NIKE3D, and TOPAZ3D,” Version 3.1, Livermore Software Technology Corporation, (1992).

Stone, C.M., Krieg, R.D., and Beisinger, Z.E., “Sancho, A Finite Element Computer Program for the Quasistatic, Large Deformation, Inelastic Response of Two-Dimensional Solids,” Sandia Report, SAND 84-2618, UC-32, Albuquerque, New Mexico, (1985).

Sturt, R.M.V. and B.D. Walker, J.C. Miles, A. Giles, and N. Grew, “Modelling the Occupant in a Vehicle Context-An Integrated Approach,” 13th International ESV Conference, Paris, November 4-7 (1991).

Tarver, C.M. and Hallquist, J.O., “Modeling of Two-Dimensional Shock Initiation and Detonation Wave Phenomena in PBX 9404 and LX-17,” University of California, Lawrence Livermore National Laboratory, Rept. UCID84990 (1981).

Taylor, L.M. and Flanagan, D.P., “PRONTO3D A Three-Dimensional Transient Solid Dynamics Program,” Sandia Report: SAND87-1912, UC-32, (1989).

Thompson, S. L., “CSQ -- A Two Dimensional Hydrodynamic Program with Energy Flow and Material Strength,” Sandia Laboratories, SAND74-0122, (1975).

Thompson, R., L., and Maffeo, R. L., “A Computer Analysis Program for Interfacing Thermal and Structural Codes,” NASA Lewis Research Center, Report NASA-TM-87021, (1985).

Trefethen, L., N., “Group Velocity in Finite Difference Schemes,” SIAM Review, Vol. 24, No. 2, (1982).

Underwood, P. “Dynamic Relaxation,” Computational Method for Transient Analysis, Belytschko, T. and Hughes, T.J.R., Eds., Vol. 1, pgs. 245-263, (1986). 6.

Van Leer, B., "Towards the Ultimate Conservative Difference Scheme. IV. A New Approach to Numerical Convection," Journal of Computational Physics, Vol. 23, pp. 276-299, (1977).

von Neumann, J. and Richtmyer, R.D., "A Method for the Numerical Calculation of Hydrodynamical Shocks," J. Appl. Phys., **21**, 232 (1950).

Walker, B.D. and P.R.B. Dallard, "An integrated Approach to Vehicle Crashworthiness and Occupant Protection Systems", SAE International Congress and Exposition, Detroit, Michigan, (910148), February 25-March 1 (1991).

Wang, J. T. and O. J. Nefske, "A New CAL3D Airbag Inflation Model," SAE paper 880654, 1988.

Wang, J. T. private communication (1992).

Warsi, Z. U. A., "Basic Differential Models for Coordinate Generation," in Symposium on the Numerical Generation of Curvilinear Coordinate Systems, Nashville, Tenn., (1982).

Wilkins, M.L., "Calculations of Elastic Plastic Flow," Meth. Comp. Phys. **3**, (Academic Press), 211-263 (1964).

Wilkins, M.L., "Use of Artificial Viscosity in Multidimensional Fluid Dynamic Calculations," J. Comp. Phys. **36**, 281 (1980).

Wilkins, M.L., Blum, R.E., Cronshagen, E. and Grantham, P., "A Method for Computer Simulation of Problems in Solid Mechanics and Gas Dynamics in Three Dimensions and Time," University of California, Lawrence Livermore National Laboratory, Rept. UCRL-51574 (1974).

Winslow, A. M., "Equipotential Zoning of Two-Dimensional Meshes," Lawrence Radiation Laboratory, UCRL-7312, (1963).

Winslow, A. M., "Equipotential Zoning of The Interior of a Three-Dimensional Mesh," Report to LSTC, (1990).

Woodruff, J.P., "KOVEC User's Manual," University of California, Lawrence Livermore National Laboratory, Rept. UCRL-51079 (1973).

Yunus, S.M., Pawlak, T.P., and Cook, R.D.,, " Solid Elements with Rotational Degrees of Freedom, Part I-Hexahedron Elements," To be published, (1989)