COMPUTATIONAL METHODS IN LAGRANGIAN AND EULERIAN HYDROCODES

David J. Benson
Dept. of AMES R-011
University of California,
San Diego
La Jolla, CA 92093

Methods in Lagrangian and Eulerian Hydrocodes

INTRODUCTION

The purpose of this paper is to describe the algorithms that are currently being used in modern, production hydrocodes. A hydrocode may be loosely defined as a code for solving large deformation, finite strain transient problems that occur on a short time scale. Although the strength of the material may not be a major factor in the solution, it is usually included, a point which distinguishes hydrocodes from the Eulerian codes used for gas dynamics. In contrast to structural analysis codes, the energy equation is integrated in time, and the deviatoric and pressure terms in the stress tensor are usually modeled separately. The solution is advanced in time using an explicit integration scheme because stress waves and shocks are an important part of the solution, and they must be resolved accurately in both space and time. An alternative name for hydrocodes is “wave codes.” The shock physics are built into a hydrocode either by including a shock viscosity, by solving the Riemann problem on the element boundaries, or by using a flux limiter.

The original motivation for developing these codes was for solving defense problems, but the range of applications has broadened considerably in recent years. Current applications include simulating astrophysical events, such as the collapse of stars or the impact of a large meteorite on the earth, the shock compaction of powders, modeling void growth in ductile materials, and analyzing high strain rate experiments. Lagrangian hydrocodes are being used extensively by industry for solving problems in automobile crashworthiness, bird impact on aircraft, and manufacturing. Eulerian hydrocodes are perhaps five to ten years behind Lagrangian codes in terms of popularity largely because Eulerian codes have been subject to greater classification restrictions than Lagrangian codes. Eulerian codes have two great advantages over Lagrangian codes: 1) they allow new free surfaces to be created in a natural manner, and 2) Eulerian codes allow turbulent flow and arbitrarily large deformations. These two qualitities make the Eulerian formulation very attractive for many manufacturing problems, and as the classification issues disappear, the use of Eulerian codes will be more common in industry.

Despite the widespread use of hydrocodes, there are no textbooks outlining their computational methods and much of the literature is buried in the form of unpublished reports in government laboratories. Hence, the need for this paper.

No attempt has been made in this paper to set forth the historical development of hydrocodes, and the interested reader should read the surveys by Johnson and Anderson [53], Anderson [68] and Zukas [85]. As a consequence, many first order accurate algorithms, and the codes that still use them, are not discussed here. The emphasis is on the methods that are both currently popular and second order accurate. A few algorithms that will undoubtedly see wider usage in the near future are also presented.

To prevent this paper from becoming a book, many interesting topics have been omitted, including structural elements, numerical methods for integrating constitutive relations, adaptive methods, transmitting boundary conditions, ad infinitum.

The derivation of the mean stress quadrilateral element is presented from both the finite element and finite difference viewpoints. The derivations are not unique from either viewpoint, but the plane strain element that virtually all finite difference and finite element programs use seems to be the same. On the other hand, there is a wide variety of axisymmetric elements, although any given axisymmetric element can probably be derived from either viewpoint. The problem with most axisymmetric elements is they do not preserve the spherical symmetry in spherical problems. Various fixes have been developed, and most of them work well. Rather than present an exhaustive examination of the different algorithms, only the simplest formulation, which is based on area weighting, is presented.

Energy conservation is very important in hydrodynamic calculations, and the energy calculations in hydrocodes are discussed in detail. One advantage that the finite element method has over the finite difference method is a natural means for ensuring that the forces are always work conjugate to the displacements. For instance, it is demonstrated that a finite difference algorithm in an Eulerian code for the edge forces is not work conjugate with the edge displacements, and a remedy is proposed based on simple virtual work arguments.

Eulerian and ALE (Arbitrary Lagrangian Eulerian) codes allow material to flow through the mesh. The general strategy is to perform a Lagrangian timestep and to follow it with a remap step that maps the solution from the distorted Lagrangian mesh on to the spatially-fixed Eulerian mesh or the ALE mesh. The discussion on Eulerian and ALE codes is therefore focused on the remap step. In addition to the algorithms that are currently used, several other algorithms that hold promise are presented.

It is hoped that this paper will be of interest to both the finite difference and the finite element communities. There are very few algorithms that can not be derived from both the finite element and the finite difference viewpoints. The notation for both viewpoints is defined and used throughout the paper as appropriate.

Methods in Lagrangian and Eulerian Hydrocodes

Chapter 1

A REVIEW OF THE GOVERNING EQUATIONS

1.0 NOTATION

The current Cartesian coordinates of a point are denoted x, its spatial velocity is u, and the reference (original) coordinates are X. For two-dimensional plane strain and plane stress problems, the coordinates y and z are often used to simplify the notation and they correspond to x_1 and x_2. When the cylindrical coordinates r and z are used, the problem is assumed to be axisymmetric (i.e., no dependence on θ). The equations that are valid for both Cartesian and cylindrical coordinates are presented only in Cartesian coordinates, and (r, z) corresponds to (x_1, x_2) and (y, z). Material time derivatives are indicated by a dot above the symbol, e.g., $\dot{u}$ is the spatial acceleration. The Cauchy stress, which is the only stress that is used in this paper, is denoted σ, the internal energy per unit mass is e, the density is ρ, the body force per unit mass is f, and a concentrated force at X is $f^*(X,t)$. The summing convention is used unless otherwise stated.

For convenience, some of the common kinematic quantities are defined here. The deformation gradient, F_{ij}, is the derivative of the spatial coordinates with respect to the reference coordinates.

$$F_{ij} = \frac{\partial x_i}{\partial X_j} \tag{1.0.1}$$

The velocity gradient, L_{ij}, is defined in terms of the current spatial coordinates. It can be split into its symmetric component, D_{ij}, the deformation rate, and its skew component, W_{ij}, the spin.

$$\begin{aligned} L_{ij} &= \frac{\partial u_i}{\partial x_j} \\ D_{ij} &= \tfrac{1}{2}(L_{ij} + L_{ji}) \\ W_{ij} &= \tfrac{1}{2}(L_{ij} - L_{ji}) \end{aligned} \tag{1.0.2}$$

The deformation rate is also labelled the strain rate, $\dot{\epsilon}$, because it is conjugate to the Cauchy stress. This is something of a misnomer since the time integral of the deformation rate does not define a measure of large deformation strain.

Methods in Lagrangian and Eulerian Hydrocodes

1.1 THE LAGRANGIAN EQUATIONS

1.1.1 The Lagrangian mass, momentum and energy equations. The derivation of the equations of motion can be found in any book on mechanics, e.g., Ref. [1]. The conservation of mass, Eq. (1.1.1.1), is used to calculate the current density from the initial density, ρ_o, in a Lagrangian hydrocode, where J is the volumetric strain.

$$\begin{aligned} \rho J &= \rho_o \\ J &= \det\left(\frac{\partial x_i}{\partial X_j}\right) \end{aligned} \tag{1.1.1.1}$$

In two and three dimensions, the momentum equation is given by Eq. (1.1.1.2), where i and j range from 1 to the number of dimensions.

$$\rho \dot{u}_i = \sigma_{ij},_j + \rho f_i \tag{1.1.1.2}$$

The corresponding equation for axisymmetric problems is Eq. (1.1.1.3).

$$\begin{aligned} \rho \ddot{r} &= \sigma_{rr},_r + \sigma_{rz},_z + \frac{\sigma_{rr} - \sigma_{\theta\theta}}{r} \\ \rho \ddot{z} &= \sigma_{zz},_z + \sigma_{zr},_r + \frac{\sigma_{zr}}{r} \end{aligned} \tag{1.1.1.3}$$

The boundary conditions are either a specified displacement, a contact inequality constraint, or a traction, where the exterior normal is denoted n and the superscripts denote different surfaces. The boundaries where the displacement, contact and traction boundary conditions are applied are Γ_x, Γ_c, Γ_τ.

$$\begin{gathered} x_i(X,t) = g_i(X,t) \quad \text{on } \Gamma_x \\ (x_i^1 - x_i^2) \cdot n_i^1 \le 0 \quad \text{on } \Gamma_c \\ \sigma_{ij} n_j = \tau_i \quad \text{on } \Gamma_\tau \end{gathered} \tag{1.1.1.4}$$

The energy equation, Eq. (1.1.1.5), is used to update the internal energy for the equation of state and to check the energy conservation in a hydrocode.

$$\rho \dot{e} = \sigma_{ij} \dot{\epsilon}_{ij} + \rho f_i u_i \tag{1.1.1.5}$$

1.1.2 The weak form of the momentum equation. The finite element method uses the weak form of the momentum equation. In mechanics, the weak form is often referred to as the “principle of virtual work.” The derivation of the weak form presented in this section ignores the contact constraint, which will be covered separately in another section. For the moment, the discussion is restricted to Cartesian coordinates.

An arbitrary function, δx, subject to the constraint that $x + \delta x$ satisfies the boundary conditions, is referred to as an admissible "virtual displacement." A solution for x that satisfies Eq. (1.1.1.2) and Eq. (1.1.1.4) will also obviously satisfy Eq. (1.1.2.1) for any admissible virtual displacement.

$$\int_{\Omega} (\rho \dot{u}_i - \sigma_{ij},_{j} - \rho f_i)\delta x_i d\Omega + \int_{\Gamma_\tau} (\sigma_{ij} n_j - \tau_i)\delta x_i d\Omega - \sum_{k=1}^{n} f_i^*(X_k, t)\delta x_i(X_k) = 0 \quad (1.1.2.1)$$

By using the divergence theorem and the product differentiation rule, the two terms involving the Cauchy stress are combined.

$$\begin{gathered} \int_{\Omega} \sigma_{ij},_{j}\, \delta x_i d\Omega = \int_{\Gamma} \sigma_{ij} n_j \delta x_i d\Omega \\ \sigma_{ij}\delta x_i,_{j} = (\sigma_{ij}\delta x_i),_{j} - \sigma_{ij},_{j}\, \delta x_i \end{gathered} \quad (1.1.2.2)$$

Substitution of Eq. (1.1.2.2) into (1.1.2.1) results in the principle of virtual work, where $\delta\pi$ is denoted as the virtual work.

$$\delta\pi = \int_{\Omega} [(\rho \dot{u}_i - \rho f_i)\delta x_i - \sigma_{ij}\delta x_i,_{j}]d\Omega - \int_{\Gamma_\tau} \tau_i \delta x_i d\Gamma - \sum_{k=1}^{n} f_i^*(X_k, t)\delta x_i(X_k) = 0 \quad (1.1.2.3)$$

Note that the continuity requirements on the stress tensor and the virtual displacement changed between Eq (1.1.1.2) and Eq. (1.1.2.3) due to the use of the divergence theorem. The weak form provides the justification for using constant stress elements in the finite element method. The strong form of the momentum equation would seem to imply that the stress in the solution must be differentiable. Both shocks and the boundaries between constant stress elements have jump discontinuities in the stress, and therefore they are not differentiable solutions.

The weak form of the momentum equation for axisymmetric problems is derived in a similar manner, but with two small differences. First, for an axisymmetric slice of one radian, the volume element is $rdrdz$ and it is convenient to group δr and δz with the factor of r. Second, the divergence theorem is not applied to the stress terms involving the factor $1/r$. Note that the domain Ω and the boundary Γ are the two dimensional figures obtained by taking a slice through the axisymmetric body, and not the full three-dimensional volume, since the integration in the θ direction is eliminated through using a one radian slice and the axisymmetry of the problem.

$$
\begin{aligned}
\delta\pi_r &= \int_\Omega [(\rho\ddot{r}_i - \rho f_r - \frac{\sigma_{rr}-\sigma_{\theta\theta}}{r})r\delta r - \sigma_{rr}(r\delta r)_{,r} - \sigma_{rz}(r\delta r)_{,z}]drdz \\
&\quad - \int_{\Gamma_\tau} \tau_r r\delta r d\Gamma - \sum_{k=1}^{n} f_r^*(X_k,t)r\delta r(X_k) = 0 \\
\delta\pi_z &= \int_\Omega [(\rho\ddot{z}_i - \rho f_z - \frac{\sigma_{zr}}{r})r\delta z - \sigma_{zz}(r\delta z)_{,z} - \sigma_{zr}(r\delta z)_{,r}]drdz \\
&\quad - \int_{\Gamma_\tau} \tau_z r\delta z d\Gamma - \sum_{k=1}^{n} f_z^*(X_k,t)r\delta z(X_k) = 0
\end{aligned}
\tag{1.1.2.4}
$$

Other than the terms $-(\sigma_{rr}-\sigma_{\theta\theta})/r$ and $-\sigma_{zr}/r$, Eq. (1.1.2.4) is identical to Eq. (1.1.2.3) if the virtual displacements in Eq. (1.1.2.4) are replaced with $\delta r/r$ and $\delta z/r$. This similarity makes implementing the plain strain and axisymmetric cases in the same program trivial.

1.1.3 The equations of motion using a control volume. Some codes, such as PISCES [2] and CAVEAT [3], use second order accurate Godunov methods. For Godunov methods, a control volume formulation is more convenient to work with than either of the traditional strong or weak forms of the equations of motion.

$$
\begin{aligned}
\frac{d}{dt}\int_\Omega \rho d\Omega &= -\int_\Gamma \rho u_i n_i d\Gamma \\
\frac{d}{dt}\int_\Omega \rho u_i d\Omega &= -\int_\Gamma \rho u_i u_j n_j d\Gamma + \int_\Gamma \sigma_{ij} n_j d\Gamma + \int_\Omega \rho f_i d\Omega \\
\frac{d}{dt}\int_\Omega \rho e d\Omega &= -\int_\Gamma \rho e u_j n_j d\Gamma + \int_\Gamma n_i \sigma_{ij} u_j d\Gamma + \int_\Omega \rho f_i u_i d\Omega
\end{aligned}
\tag{1.1.3.1}
$$

1.2 THE ALE EQUATIONS

In this section, the governing equations are derived for the case when the reference coordinates move at an arbitrary velocity [9], [35], [36]. This formulation is referred to as the arbitrary Lagrangian-Eulerian (ALE) formulation as it contains both the Lagrangian and Eulerian equations as subsets. The velocity of the material is u, the velocity of the reference coordinates is v, and their difference, $u - v$, is denoted w. The Jacobian, J', is the relative differential volume between the reference and the spatial coordinates.

$$\frac{\partial J'}{\partial t} = J' \frac{\partial v_i}{\partial x_i} \tag{1.2.1}$$

The material time derivative can be expressed in terms of both the spatial and reference coordinates, where f^r means that f is expressed as a function of the reference coordinates.

$$\dot{f} = \frac{\partial f}{\partial t} + u_i \frac{\partial f}{\partial x_i} \tag{1.2.2a}$$

$$\dot{f} = \frac{\partial f^r}{\partial t} + (u_i - v_i) \frac{\partial f}{\partial x_i} \tag{1.2.2b}$$

The ALE equations are derived by substituting Eq. (1.2.2) into the equations in the previous section, but they are not in conservation form.

$$\frac{\partial \rho^r}{\partial t} = -\rho \frac{\partial u_i}{\partial x_i} - w_i \frac{\partial \rho}{\partial x_i} \tag{1.2.3a}$$

$$\rho \frac{\partial u_i^r}{\partial t} = (\sigma_{ij},_j + \rho b_i) - \rho w_j \frac{\partial u_i}{\partial x_j} \tag{1.2.3b}$$

$$\rho \frac{\partial e^r}{\partial t} = (\sigma_{ij} u_i,_j + \rho b_i u_i) - \rho w_j \frac{\partial e}{\partial x_j} \tag{1.2.3c}$$

To put them into conservation form, an additional identity is derived by multiplying Eq. (1.2.3a) by J', multiplying Eq. (1.2.1) by ρ, and adding them.

$$\frac{\partial J' \rho}{\partial t} = -J' \frac{\partial \rho w_i}{\partial x_i} \tag{1.2.3}$$

After multiplying Eq. (1.2.3) by f, Eq. (1.2.2b) by $\rho J'$, and arranging terms, the ALE equation for f is written in its general form. The first term on the right hand side of Eq. (1.2.4) is the source term for f, and the second term, the transport of f.

$$\frac{\partial (J' \rho f)}{\partial t} = J' \rho \dot{f} - J' \frac{\partial \rho f w_i}{\partial x_i} \tag{1.2.4}$$

The conservation form of the ALE equations is obtained by substituting the Lagrangian equations of the previous section into Eq. (1.2.4).

$$\frac{\partial \rho J'}{\partial t} = -J' \frac{\partial \rho w_i}{\partial x_i} \tag{1.2.5a}$$

$$\frac{\partial \rho J' u_i}{\partial t} = J'(\sigma_{ij},_j + \rho b_i) - J' \frac{\partial \rho u_i w_j}{\partial x_j} \tag{1.2.5b}$$

$$\frac{\partial \rho J' e}{\partial t} = J'(\sigma_{ij} u_i,_j + \rho b_i u_i) - J' \frac{\partial \rho e w_j}{\partial x_j} \tag{1.2.5c}$$

When w is zero, J' is one and the Lagrangian equations are recovered from Eq. (1.2.5). If the reference coordinates are the current spatial coordinates, w is v, J' is again one, and the Eulerian equations are recovered.

The ALE equations for the axisymmetric case are similar to Eq. (1.2.5). In addition to the extra terms involving the stress, the axisymmetric equations differ from the Cartesian equations because the velocity divergence w_r is replaced by rw_r.

1.3 THE RANKINE-HUGONIOT CONDITIONS

The physics across a shock or contact discontinuity are governed by the Rankine-Hugoniot conditions. The line integral derivation presented in this section follows Noh [4]. For a smooth shock front, the solution is discontinuous only in the direction normal to the shock front, making a one-dimensional analysis sufficient. Additional complexities arise at the intersection of shocks, and the interested reader is referred to [5].

For the moment, a single equation in conservation form is considered, where u is the unknown, f is the flux of u, and x is the single space dimension.

$$u,_t + f(u),_x = 0 \tag{1.3.1}$$

Based on the same arguments that were used to derive the weak form of the equations in Section 1.1, the integral of Eq. (1.3.1) over any section of the $x - t$ plane must be zero.

$$\int_B (u,_t + f(u),_x) dx dt = 0 \tag{1.3.2}$$

Equation (1.3.2) is converted to a boundary integral by applying Green's theorem. The boundary of the region B in the $x - t$ plane is denoted b.

$$\oint_b u dx - f(u) dt = 0 \tag{1.3.3}$$

The reason for calling f the flux function and Eq. (1.3.1) a conservation equation is apparent when a rectangular region, $[x_\ell, x_r] \times [t_a, t_b]$, in the $x - t$ plane is considered. Let U be the integral of u in the interval $[x_\ell, x_r]$ and apply Eq. (1.3.3).

$$\begin{aligned} U(t_b) - U(t_a) &= \int_{x_\ell}^{x_r} u(t_b)dx - \int_{x_\ell}^{x_r} u(t_a)dx \\ &= \int_{t_a}^{t_b} f(u(x_r))dt - \int_{t_a}^{t_b} f(u(x_\ell))dt \end{aligned} \tag{1.3.4}$$

The jump conditions are derived by considering an arbitrary region B that is cut by a line s into two regions B_1 and B_2, see Fig. 1. The corresponding boundaries are the union of b_1 with s and b_2 with s. The integrals over the two separate regions must each be zero, as is the integral over the original region B. Note that in applying Eq. (1.3.3) to region B_2, the curve s is traversed in the direction opposite to the one used for region B_1. For clarity, the curve s is subscripted by a 1 or 2 to indicate the direction.

$$\begin{aligned} \int_B &= \oint_{b_1} + \oint_{b_2} = 0 \\ &= \int_{B_1} + \int_{B_2} \\ &= \oint_{b_1} + \oint_{s_1} + \oint_{b_2} + \oint_{s_2} \\ &= (\oint_{b_1} + \oint_{b_2}) + (\oint_{s_1} + \oint_{s_2}) \\ &= \oint_{s_1} (udx - f(u)dt) + \oint_{s_2} (udx - f(u)dt) \end{aligned} \tag{1.3.5}$$

The jump condition can be written concisely by introducing some extra notation. The values of u and f in B_1 and B_2 are denoted by subscripting them with a 1 or 2, and the direction taken along s_2 is considered to be positive. The jumps, $[u]$ and $[f(u)]$, are defined as the differences $u_2 - u_1$ and $f_2 - f_1$.

$$\oint_s [u]dx - [f]dt = 0 \tag{1.3.6}$$

The region B was chosen arbitrarily, therefore the integrand in Eq. (1.3.6) must be zero at every point on every curve s. The slope, dx/dt, of the curve at a point is S.

$$S[u] - [f] = 0 \tag{1.3.7}$$

Except at shocks and contact discontinuities, the solution is continuous and Eq. (1.3.7) is trivially zero.

To apply Eqn. (1.3.7), the conservation equations for a continuum must be written in the spatial, or Eulerian, conservation form since the integrals in the derivation are performed on a fixed space-time domain. The density, velocity, pressure, and total energy per unit volume are ρ, u, P, and $\mathcal{E}$ respectively.

$$\begin{aligned} \rho,_t + (\rho u),_x &= 0 \\ \rho u,_t + (\rho u^2 + P),_x &= 0 \\ \mathcal{E},_t + (\mathcal{E} u + Pu),_x &= 0 \end{aligned} \tag{1.3.8}$$

Substituting Eqn. (1.3.8) into Eqn. (1.3.7) gives the standard jump conditions.

$$\begin{aligned} S[\rho] &= [\rho u] \\ S[\rho u] &= [\rho u^2 + P] \\ S[\mathcal{E}] &= [\mathcal{E} u + Pu] \end{aligned} \tag{1.3.9}$$

Shocks have discontinuities in all solution variables, while contact discontinuities have continuous pressures and normal velocities. It is well known that in linear hyperbolic equations that the solutions are discontinuous only across characteristics. Shocks are not, however, characteristics [5]. If the jump in the solution variables is known across the discontinuity, then Eqn. (1.3.8) can be used to calculate the velocity of the jump. For the special case of the contact discontinuity, the contact surface velocity will equal the particle velocity, u.

The equation of state for an ideal gas, Eqn. (1.3.11) was used by Hugoniot in 1889 to obtain an analytical solution for the pressure jump across a shock, Eqn. (1.3.12). The speed of sound in the ideal gas is a, and it equals $\sqrt{\gamma(\gamma-1)e}$, where e is the internal energy per unit mass.

$$P = (\gamma - 1)\rho e \tag{1.3.11}$$

$$P_2 - P_1 = \frac{\gamma+1}{4}\rho_1(\triangle u)^2 + \rho_1|\triangle u|\sqrt{\left(\frac{\gamma+1}{4}\right)^2(\triangle u)^2 + a_1^2} \tag{1.3.12}$$

For a linearly elastic fluid, Eqn. (1.3.13), it is also possible to find an analytical solution for the pressure jump across a shock, Eqn. (1.3.14). The bulk modulus is K, the initial density is ρ_0, and the speed of sound is $\sqrt{K/\rho}$.

$$P = K\left(\frac{\rho}{\rho_0} - 1\right) \tag{1.3.13}$$

$$P_2 - P_1 = \tfrac{1}{2}\rho_1(\triangle u)^2 + \rho_1 \triangle u \sqrt{\left(\frac{\triangle u}{2}\right)^2 + a_1^2} \tag{1.3.14}$$

The form of these classical solutions motivated the quadratic plus linear form that is currently used in most shock viscosities.

Chapter 2

LAGRANGIAN COMPUTATIONAL METHODS

2.1 AN OVERVIEW

A Lagrangian calculation embeds a computational mesh in the material domain and solves for the position of the mesh at discrete points in time. Since the mesh is embedded in the material, the motion of the material is inferred from the motion of the mesh. Both the Lagrangian finite element and finite difference methods are described in this section. Some of the finite difference methods are intended to be part of an Eulerian calculation and cannot be run separately for than one time step for multidimensional problems due to the geometrical simplifications they use.

All the methods that are currently popular for solving transient hydrodynamics problems are explicit. The solution is advanced from time t^n to time t^{n+1} without any iterations, and the difference between them is the time step, $\triangle t^{n+1/2}$. To minimize the storage required, the solution is stored for only one time t^n within the program. At the completion of a time step, the solution at the beginning of the time step is overwritten by the solution at the end of the step.

Centering in time. The central difference method is used in most hydrocodes to advance the position of the mesh in time. It is based on the second order accurate central difference approximation, which is readily derived from a Taylor series expansion. A superscript n indicates that the function is evaluated at t^n.

$$\begin{aligned} f^n &= f^{n+1/2} - f_{,t}^{n+1/2}\left(\frac{\triangle t^{n+1/2}}{2}\right) + \tfrac{1}{2} f_{,tt}^{n+1/2}\left(\frac{\triangle t^{n+1/2}}{2}\right)^2 \cdots \\ f^{n+1} &= f^{n+1/2} + f_{,t}^{n+1/2}\left(\frac{\triangle t^{n+1/2}}{2}\right) + \tfrac{1}{2} f_{,tt}^{n+1/2}\left(\frac{\triangle t^{n+1/2}}{2}\right)^2 \cdots \end{aligned} \qquad (2.1.1)$$

Subtracting the second equation from the first in Eq. (2.1.1) gives an explicit second order accurate integration rule.

$$f^{n+1} = f^n + f_{,t}^{n+1/2} \triangle t^{n+1/2} + \mathcal{O}\big((\triangle t^{n+1/2})^3\big) \tag{2.1.2}$$

Letting f be the displacement x, it is clear that the velocity must be staggered in time with respect to the displacement.

$$x^{n+1} = x^n + u^{n+1/2} \triangle t^{n+1/2} \tag{2.1.3}$$

Applying Eq. (2.1.2) again gives the integration rule for the velocity.

$$\begin{aligned} u^{n+1/2} &= u^{n-1/2} + \dot{u}^n \triangle t^n \\ &= u^{n-1/2} + (F^n/M) \triangle t^n \end{aligned} \tag{2.1.2}$$

Since the central difference method is explicit, it has a finite stable time step size. For most problems, the stepsize is given by the Courant limit, which is the smallest amount of time that is necessary for a sound wave to cross an element in the mesh. This number is a function of both the element geometry and speed of sound within the material. For problems involving hypervelocity impact, the impact velocity may exceed the sound speed and smaller time steps must be used. The time step size may also be limited by the contact algorithm, the magnitude of the shock viscosity, or an explosive burn.

A few codes, such as BBC [146], KRAKEN [102], and MESA [101], use a Runge-Kutta algorithm so that all the solution variables are centered at t^n, which is particularly attractive for Eulerian calculations. This scheme was first proposed by Truilio and Trigger [165] in 1961. The solution is advanced to $t^{n+1/2}$ by a half step with forward Euler, and the derivative is evaluated at the half step using the approximate solution. This time-centered derivative is used to advance the solution from n to $n+1$

$$\begin{aligned} f^{n+1/2} &= f^n + \tfrac{1}{2} f^n{}_{,t} \triangle t \\ f^{n+1} &= f^n + f^{n+1/2}{}_{,t} \triangle t \end{aligned} \tag{2.1.3}$$

Equation (2.1.3) advances the velocity from t^n to t^{n+1} while the displacements are advanced to t^{n+1} using the trapezoidal rule.

$$\begin{aligned} u^{n+1/2} &= u^n + \tfrac{1}{2}(F^n/M)\triangle t \\ u^{n+1} &= u^n + (F^{n+1/2}/M)\triangle t \\ x^{n+1} &= x^n + \tfrac{1}{2}(u^{n+1} + u^n)\triangle t \end{aligned} \tag{2.1.4}$$

The acceleration at t^n is a function of the stress, including both the deviatoric part from the strength of the material and the pressure from the equation of state. The pressure is a function of the internal energy and the density, which requires evaluation of the internal energy at t^n. The density at t^n is known since mass is conserved and the geometry, defined by x^n, is known. The constitutive equations for the strength of the material are expressed as differential equations which must also be integrated in time.

$$\sigma^{n+1} = \sigma^n + \dot{\sigma}^{n+1/2}\triangle t^{n+1/2} \tag{2.1.5}$$

To evaluate the stress rate at $t^{n+1/2}$, the strain rate must be known at $t^{n+1/2}$. The strain rate is the symmetric part of the velocity gradient, $\partial u/\partial x$. Although the velocity is known at $t^{n+1/2}$, the geometry is known only at t^n and t^{n+1} for the central difference method. A second order approximation to the midpoint geometry is given by the arithmetic average.

$$x^{n+1/2} = \tfrac{1}{2}\bigl(x^n + x^{n+1}\bigr) \tag{2.1.6}$$

While the information is available to integrate the constitutive equations with second order accuracy, the actual integration methods for most plasticity models are only first order. The most popular integration technique is the radial return algorithm developed by Maenchen and Sack [6], Wilkins [7], and Krieg and Key [8]. It is asymptotically exact as the strain increment over a time step approaches zero or infinity. The radial return algorithm is very accurate despite its theoretical first order accuracy, and it is computationally efficient and extremely robust.

2.2 THE MESH DEFINITIONS

The mesh is the discrete description of the continuous spatial domain. Three different types of meshes are considered, and they can be used with either a finite element or finite difference formulation.

The first mesh type is the logically regular mesh, and it is illustrated in Fig. 2. Two intersecting sets of lines, called k and ℓ lines, define a set of quadrilaterals. The sets of lines, which are generally not orthogonal, are numbered independently. Their intersections are the nodes, which are labelled by their ordered pair of intersecting lines, (k, ℓ). The quadrilateral regions bounded by the four nodes (k, ℓ), $(k+1, \ell)$, $(k+1, \ell+1)$, and $(k, \ell+1)$ are called "zones" or "cells." They are labelled by the average logical coordinates of the nodes, $(k + 1/2, \ell + 1/2)$. The quadrilateral elements can be subdivided into two or four triangles, but most hydrocodes work with quadrilateral zones. The displacements, velocities, and accelerations are evaluated at the nodes, while the remaining variables are evaluated at the centroids of the zones.

The second type of mesh is also defined by sets of intersecting k and ℓ lines. It differs from the preceding mesh in that the velocities are evaluated at the midpoints of the zone edges, and that only the velocity component normal to thae edge appears in the calculation. This type of mesh is frequently found in Eulerian hydrocodes because of its simplicity and its ability to resolve shear flows in only two elements (adjacent rows of elements can move in opposite directions). It does, however, have two serious drawbacks. The first is the mesh lines must be orthogonal. Most Eulerian hydrocodes are restricted to rectangular zones, although some codes allow the use of meshes based on orthogonal curvilinear coordinate systems. Since the mesh lines need to be orthogonal, these codes cannot be run in a Lagrangian mode for more than one time step unless the flow is only along one coordinate direction. The second is the difficulty associated with calculating the shear strain rate within a zone. Every Eulerian code that uses a mesh with edge-centered velocities seems to have its own unique way of evaluating the shear strain rate.

The last type of mesh is an unstructured mesh. The primitive object in this case is the "element" or zone, which is defined by a set of nodes. The elements are used to cover the spatial domain in a nonoverlapping manner, see Fig. 3. Elements can have both vertex and edge nodes. In constructing the mesh, a node must be either a vertex or edge node for all the elements connected to it. The nodes are numbered in an arbitrary manner since they are not defined in terms of intersecting mesh lines. Only elements that are defined solely by vertex nodes are discussed in this paper. The other elements are more complicated and have smaller stable time steps than the simplest elements, making them too expensive for hydrodynamic calculations on today's computers.

Unstructured meshes are most frequently used in finite element programs. Although the finite difference theory for an unstructured mesh is well known [9], [7], it is rarely used because of the complexity of vectorizing it on a supercomputer while using the finite difference formalism. Most of the finite difference programs that do use an unstructured mesh are “free-Lagrangian” codes [10] that dynamically change the mesh connectivity during the calculation.

2.3 A SUMMARY OF A LAGRANGIAN TIME STEP

The general flow of the calculations in a typical Lagrangian code is given below. This summary is somewhat premature in the sense that some of the terms used in it have not yet been defined, but it is presented at this stage to give some perspective before plunging into a detailed description of the individual steps.

1. Knowing the stress, pressure, hourglass forces, and shock viscosity at t^n in each zone or element, the forces at the nodes are calculated. The accelerations of the nodes are calculated by dividing the nodal forces by the nodal masses.

2. The acceleration is integrated to give the velocity at $t^{n+1/2}$.

3. The velocity is integrated to give the displacement at t^{n+1}.

4. The constitutive model for the strength of the material is integrated from t^n to t^{n+1} now that the motion of the material is known.

5. The artificial shock viscosity and hourglass viscosity are calculated from $u^{n+1/2}$.

6. The internal energy is updated based on the work done between t^n and t^{n+1}.

7. Based on the density and energy at t^{n+1}, the pressure is calculated from the equation of state.

8. A new time step size is calculated based on the speed of sound through each of the elements and their geometry.

9. Advance the time and return to step 1.

For formulations that use a Runge-Kutta method, steps 1 and 2 are replaced by step 1a, use forward Euler integration to advance the solution temporarily to $t^{n+1/2}$, and calculate the acceleration at $t^{n+1/2}$ and step 2a, advance the velocity to t^{n+1}. Step 5 is also merged into step 1a for the Runge-Kutta method. The only Lagrangian method that is a substantial departure from the general flow given above is Godunov’s method [11], which is discussed in a separate section. The major difference between most finite element and finite difference formulations is how the accelerations are calculated while the methods used in finite element and finite difference methods to update the stress, shock viscosity, and energy calculations are virtually identical. These topics are presented in their own sections and are completely independent of the finite element and finite difference sections.

2.4 THE FINITE ELEMENT METHOD

The publication of finite element textbooks has become something of a small industry. Most of the textbooks are restricted to small deformations and linear elasticity. Among the best books are the recent one by Hughes [12] and the latest edition of Zienkiewicz's and Taylor's textbook [13]. While Bathe's book [14] covers large deformations and plasticity, its emphasis is on implicit methods for structural analysis. Cook's introductory text [15] is popular one for teaching undergraduates. To see how mathematicians view the finite element method, the book by Johnson [16] is recommended.

2.4.1 Calculating the forces at the nodes. Before introducing the specialized notation of the finite element method, the force calculations are derived in a fairly general manner from the weak form of the momentum equation. The displacements, $x_{\alpha i}$, and velocities $u_{\alpha i}$, are known at the nodes, where α is the node number and i is the direction. It is assumed that the value of the displacements, velocities, and accelerations can be interpolated from their nodal values at any point in any element, where $N_\alpha(x)$ is the interpolation function associated with node α.

$$\begin{aligned} x_i(x) &= N_\alpha(x) x_{\alpha i} \\ u_i(x) &= N_\alpha(x) u_{\alpha i} \\ \dot{u}_i(x) &= N_\alpha(x) \dot{u}_{\alpha i} \end{aligned} \tag{2.4.1.1}$$

In a similar manner, the variation in displacement can be interpolated.

$$\delta x_i(x) = N_\alpha(x) \delta x_{\alpha i} \tag{2.4.1.2}$$

The gradient of the variation is evaluated by differentiating the interpolation functions.

$$\frac{\partial \delta x_i}{\partial x_j} = \frac{\partial N_\alpha(x)}{\partial x_j} \delta x_{\alpha i} \tag{2.4.1.3}$$

Substituting the above equations into the weak form of the momentum equation, Eq. (1.1.2.3), gives the following discrete form in Cartesian coordinates.

$$\Big[\int_\Omega \rho N_\beta \dot{u}_{\beta i} N_\alpha d\Omega + \int_\Omega \sigma_{ij}\frac{\partial N_\alpha(x)}{\partial x_j} d\Omega - \int_\Omega \rho f_i N_\alpha d\Omega - \int_{\Gamma_\tau} \tau_i N_\alpha d\Gamma - \sum_{k=1}^{n} f_i^*(X_k,t) N_\alpha(X_k)\Big]\delta x_{\alpha i} = 0 \tag{2.4.1.4}$$

Since the virtual displacements at the nodes, $\delta x_{\alpha i}$, are arbitrary, the individual terms enclosed by the square brackets must individually be zero. The accelerations of the nodes are calculated by solving the set of linear equations generated by a slight rearrangement of Eq. (2.4.1.4).

$$\Big[\int_\Omega \rho N_\beta N_\alpha d\Omega\Big]\Big\{\dot{u}_{\beta i}\Big\} = \Big\{-\int_\Omega \sigma_{ij}\frac{\partial N_\alpha(x)}{\partial x_j} d\Omega + \int_\Omega \rho f_i N_\alpha d\Omega + \int_{\Gamma_\tau} \tau_i N_\alpha d\Gamma + \sum_{k=1}^{n} f_i^*(X_k,t) N_\alpha(X_k)\Big\} \tag{2.4.1.5}$$

The first term, which is enclosed in square brackets, is the "consistent" mass matrix. It is consistent in the sense that it is the mass matrix that appears when the finite element method is applied to all the terms in the momentum equation in a straight forward manner. The equations are coupled because the consistent mass matrix is not diagonal and solving this system is unattractive due to the high computational cost. A "lumped" mass matrix, which is diagonal, is used instead of the consistent mass matrix in all hydrocodes. In its simplest and most common form, the mass of an element is distributed evenly to all its nodes. The consistent mass matrix is denoted M^c, the lumped mass matrix, M. Both have four subscripts, $\alpha i\beta j$, where αi refers to the ith displacement component of node α, and βj indicates the jth displacement component of node β. The mass of element A is M_A, an element has *nnodes* nodes, and it is implied in the summation over the elements for the lumped mass matrix that the range of the summation is only over the elements that have node α in their definition.

$$\begin{aligned} M^c_{\alpha i\beta j} &= \int_\Omega \rho N_\beta N_\alpha d\Omega \\ M_{\alpha i\beta j} &= \delta_{\alpha\beta}\delta_{ij}\frac{1}{nnodes}\sum_A M_A \end{aligned} \tag{2.4.1.6}$$

A lumped mass matrix is more than just a computational simplification; it also gives better answers for impulsive loads. Consider the following one-dimensional problem: a force of constant positive magnitude is suddenly turned on at t^0 at the left end of an elastic rod. In the continuum problem a stress wave propagates from the left to the right with a the speed of sound in the material, a. Material to the right of the point $x = at$ is undisturbed and the velocity of the material to the left of that point is positive. If a consistent mass matrix is used in Eq. (2.4.1.5), spurious oscillations will occur ahead of the stress wave. Only the first node should have a nonzero acceleration at t^0, but because of the inertial coupling between the nodes, every other node starting with the first will have a positive acceleration and the remainder will have a negative acceleration. This result is, of course, completely contrary to the physics of the problem. The precise mathematics describing when and why the consistent mass matrix does not give good solutions is discussed in Ref. [17].

2.4.2 The plane strain quadrilateral element. At this point it is worthwhile to introduce a specific element to illustrate the general finite element notation used in the remainder of the paper. The plane strain quadrilateral is chosen for this purpose because it is the element most commonly used in finite element hydrocodes and it is equivalent to the integral difference formulation used by Noh [9] and Wilkins [7]. While a triangular element is computationally simpler, it is more difficult to understand due to the local coordinate system that is embedded within the triangular element.

The quadrilateral element has four nodes. In addition to the global numbering of the nodes, it is convenient to assign them the numbers 1 through 4 in a counter clockwise manner when considering a single element. A pair of local curvilinear coordinates, ξ^i, is introduced to simplify the interpolation functions. These coordinates are referred to as the "isoparametric" coordinates in the finite element literature. The range of the coordinates is -1 to +1 as shown in Fig. 3. The conditions on the interpolation functions N_α are that 1) they are continuous within the element, 2) their sum at any point within the element is 1, and 3) the value of N_α at node β is $\delta_{\alpha\beta}$. Linear interpolation cannot be used with a quadrilateral element because there are four known values at the nodes, and only three are required to define a plane. A set of bilinear interpolation functions that satisfies the interpolation requirements can be stated in terms of the isoparametric coordinates, where ξ^i_α refers to the value of ξ^i at node α.

$$N_\alpha(\xi^1, \xi^2) = \tfrac{1}{4}(1 + \xi^1_\alpha \xi^1)(1 + \xi^2_\alpha \xi^2) \qquad (2.4.2.1)$$

Expanding the sum $N_\alpha x_\alpha$ shows that the bilinear interpolation includes all the linear terms and a single quadratic cross term.

$$
\begin{aligned}
x_i(\xi^1,\xi^2) &= G_0^i + G_1^i\xi^1 + G_2^i\xi^2 + G_3^i\xi^1\xi^2 \\
G_0^i &= \tfrac{1}{4}(x_{1i} + x_{2i} + x_{3i} + x_{4i}) \\
G_1^i &= \tfrac{1}{4}(-x_{1i} + x_{2i} + x_{3i} - x_{4i}) \\
G_2^i &= \tfrac{1}{4}(-x_{1i} - x_{2i} + x_{3i} + x_{4i}) \\
G_3^i &= \tfrac{1}{4}(x_{1i} - x_{2i} + x_{3i} - x_{4i})
\end{aligned}
\tag{2.4.2.2}
$$

The spatial gradient of a field is calculated using the chain rule. In addition to evaluating the strain rates, the gradient is necessary for calculating the stress divergence term in Eq. (2.4.1.5).

$$
\frac{\partial g}{\partial x_i} = \frac{\partial N_\alpha(\xi^1,\xi^2)}{\partial \xi^1}\frac{\partial \xi^1}{\partial x_i} + \frac{\partial N_\alpha(\xi^1,\xi^2)}{\partial \xi^2}\frac{\partial \xi^2}{\partial x_i}
\tag{2.4.2.3}
$$

The derivatives $\partial\xi^i/\partial x_j$ are evaluated by inverting the Jacobian matrix $\partial x_i/\partial \xi^j$.

$$
\frac{\partial \xi^i}{\partial x_j} = \begin{bmatrix} G_1^1 + G_3^1\xi^2 & G_2^1 + G_3^1\xi^1 \\ G_1^2 + G_3^2\xi^2 & G_2^2 + G_3^2\xi^1 \end{bmatrix}
\tag{2.4.2.4}
$$

The integration of the stress divergence over the element in the weak form of the momentum equation, Eq. (1.1.2.1), is simplified by assuming that the stress is constant over the element. Implicit structural finite element programs use a bilinear distribution for the stress which must be numerically integrated with Gauss quadrature [12]. Constant stress elements are also referred to as 1-point elements because they correspond to using one Gauss integration point in the evaluation of the integrals. In three dimensions, constant stress elements and 1-point elements are not exactly the same, but the terms are still used interchangeably for the most part. The location x^o is the average of the four nodal coordinates, and A is the element area.

$$
F^\sigma = \int_\Omega \sigma_{ij}\frac{\partial N_\alpha(x)}{\partial x_j}d\Omega \approx \sigma_{ij}\frac{\partial N_\alpha(x^o)}{\partial x_j}A
\tag{2.4.2.5}
$$

When the inverse is evaluated at the origin of the isoparametric coordinates, the expressions for the partial derivatives become particularly simple. Of particular importance is the velocity gradient. The coordinates y and z are used to simplify the notation but the equation is also valid for cylindrical coordinates.

$$
\begin{aligned}
\left[\frac{\partial u_i}{\partial x_j}\right] &= \frac{1}{2A}\begin{bmatrix} z_{31}\dot{y}_{24} - z_{24}\dot{y}_{31} & y_{24}\dot{y}_{31} - y_{31}\dot{y}_{24} \\ z_{31}\dot{z}_{24} - z_{24}\dot{z}_{31} & y_{24}\dot{z}_{31} - y_{31}\dot{z}_{24} \end{bmatrix} \\
(\cdot)_{\alpha\beta} &\equiv (\cdot)_\alpha - (\cdot)_\beta
\end{aligned}
\tag{2.4.2.6}
$$

The velocity gradient can be rewritten in terms of a matrix multiplication.

$$\left[\frac{\partial u_i}{\partial x_j}\right] = B_{ij\alpha k} u_{\alpha k} \tag{2.4.2.7}$$

The gradient of the virtual displacement takes the same form. This greatly simplifies the evaluation of the forces on the nodes.

$$F^{\sigma}_{\alpha k} = \sigma_{ij} B_{ij\alpha k} A \tag{2.4.2.8}$$

The B matrix can be evaluated at any point within an element by starting with the general definitions of the derivatives and the abstract equation for the nodal forces that appears in all finite element textbooks is derived by substituting it into the weak form. The strain rates are also expressed as the product of the B matrix and the velocity field.

$$\begin{aligned} \{F^{\sigma}\} &= \int_{\Omega} [B]^t \{\sigma\} d\Omega \\ \{\dot{\epsilon}\} &= [B]\{u\} \end{aligned} \tag{2.4.2.9}$$

The B matrix for a constant stress element can be written out conveniently by numbering σ_{11}, σ_{22}, σ_{33} and σ_{12} as σ_1, σ_2, σ_3 and σ_4 and the corresponding strain rates $\dot{\epsilon}_{11}$, $\dot{\epsilon}_{22}$, $\dot{\epsilon}_{33}$, and $2\dot{\epsilon}_{12}$ as $\dot{\epsilon}_1$, $\dot{\epsilon}_2$, $\dot{\epsilon}_3$ and $2\dot{\epsilon}_4$. The velocities are ordered $(\dot{y}_1, \dot{z}_1 \ldots \dot{y}_4, \dot{z}_4)$.

$$[B] = \frac{1}{2A}\begin{bmatrix} z_{24} & 0 & z_{31} & 0 & -z_{24} & 0 & -z_{31} & 0 \\ 0 & -y_{24} & 0 & -y_{31} & 0 & y_{24} & 0 & y_{31} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -y_{24} & z_{24} & -y_{31} & z_{31} & y_{24} & -z_{24} & y_{31} & -z_{31} \end{bmatrix} \tag{2.4.2.10}$$

After carrying out the matrix multiplication, the force expressions are quite simple.

$$\begin{aligned} F_{y1} &= -F_{y3} = \tfrac{1}{2}(z_{24}\sigma_{yy} - y_{24}\sigma_{yz}) \\ F_{y2} &= -F_{y4} = \tfrac{1}{2}(z_{31}\sigma_{yy} - y_{31}\sigma_{yz}) \\ F_{z1} &= -F_{z3} = \tfrac{1}{2}(z_{24}\sigma_{zy} - y_{24}\sigma_{zz}) \\ F_{z2} &= -F_{z4} = \tfrac{1}{2}(z_{31}\sigma_{zy} - y_{31}\sigma_{zz}) \end{aligned} \tag{2.4.2.11}$$

The body force terms are evaluated by assuming that the body force is constant within an element.

$$F^{b}_{\alpha i} = \int_{\Omega} \rho f_i N_{\alpha} d\Omega \approx \tfrac{1}{4} M_{\alpha} f_i(x^o) \tag{2.4.2.12}$$

Boundary tractions are assumed to vary in a linear manner between two adjacent boundary nodes α and β separated by a distance L. The total force acting on the boundary segment between the two nodes is the average of the two values at the nodes. The boundary integral may be evaluated in a manner consistent with Eq. (2.4.1.2), or one half of the total force may be distributed to each node. The latter procedure is followed in DYNA2D [82].

$$\begin{aligned} &\left\{ \begin{aligned} F^\tau_{\alpha i} &= (\tfrac{1}{3}\tau_{\alpha i} + \tfrac{1}{6}\tau_{\beta i})L \\ F^\tau_{\beta i} &= (\tfrac{1}{6}\tau_{\alpha i} + \tfrac{1}{3}\tau_{\beta i})L \end{aligned} \right\} \quad \text{consistent} \\ &\left\{ \begin{aligned} F^\tau_{\alpha i} &= \tfrac{1}{2}(\tau_{\alpha i} + \tau_{\beta i})L \\ F^\tau_{\beta i} &= \tfrac{1}{2}(\tau_{\alpha i} + \tau_{\beta i})L \end{aligned} \right\} \quad \text{lumped} \end{aligned} \tag{2.4.2.13}$$

Concentrated forces are applied directly to the specified nodes.

$$F^c_{\alpha i} = f^*_{\alpha i} \tag{2.4.2.14}$$

2.4.3 The axisymmetric quadrilateral element. The most obvious implementation of an axisymmetric element is the direct application of the finite element method to Eq. (1.1.9). One disadvantage of using Eq. (1.1.9) is the resulting finite element formulation does not work well for spherically symmetric problems [18], see Fig. 4. The reason is the factor of r appearing in the differential volume element $2\pi r dr dz$ places too little weighting (it's actually zero in the limit) on the nodes along the z axis. Symmetry is achieved if the differential element is $drdz$, i.e., if an area weighting is used instead of a volume weighting in the weak form [18].

The axisymmetric element described in this section is based on area weighting since it preserves spherical symmetry. There are only two significant differences between the plane strain element and the axisymmetric element. The first is that the definition of the B matrix is altered by including the hoop strain rate. The second change is that the mass matrix is now time dependent.

The hoop strain rate, $\dot\epsilon_{\theta\theta}$, is $\dot r/r$. In a similar manner, the associated virtual displacement is $\delta r/r$. The value of r^0 is the arithmetic average of the radii of the four nodes.

$$[B] = \frac{1}{2A}\begin{bmatrix} z_{24} & 0 & z_{31} & 0 & -z_{24} & 0 & -z_{31} & 0 \\ 0 & -r_{24} & 0 & -r_{31} & 0 & r_{24} & 0 & r_{31} \\ A/2r^o & 0 & A/2r^o & 0 & A/2r^o & 0 & A/2r^o & 0 \\ -r_{24} & z_{24} & -r_{31} & z_{31} & r_{24} & -z_{24} & r_{31} & -z_{31} \end{bmatrix} \tag{2.4.3.1}$$

The strain rate is $[B]\{\dot r_1, \dot z_1 \ldots \dot r_4, \dot z_4\}$. Since the formulation is based on an area weighting as opposed to volume weighting (virtual work principal), the nodal forces are

$[B]^t\sigma A$ plus the additional terms with σ_{rr} and $\sigma_{\theta\theta}$ that appear as body force terms in Eq. (1.1.2.4).

$$\begin{aligned}
F_{r1} = -F_{r3} &= \tfrac{1}{2}(z_{24}\sigma_{rr} - r_{24}\sigma_{rz}) \pm \tfrac{A}{4}(\sigma_{rr} - \sigma_{\theta\theta})/r^o \\
F_{r2} = -F_{r4} &= \tfrac{1}{2}(z_{31}\sigma_{rr} - r_{31}\sigma_{rz}) \pm \tfrac{A}{4}(\sigma_{rr} - \sigma_{\theta\theta})/r^o \\
F_{z1} = -F_{z3} &= \tfrac{1}{2}(z_{24}\sigma_{zr} - r_{24}\sigma_{zz}) \pm \tfrac{A}{4}\sigma_{rz}/r^o \\
F_{z2} = -F_{z4} &= \tfrac{1}{2}(z_{31}\sigma_{zr} - r_{31}\sigma_{zz}) \pm \tfrac{A}{4}\sigma_{rz}/r^o
\end{aligned} \tag{2.4.3.2}$$

The inertial term in the area-weighted weak form is time dependent, but it is, however, very inexpensive to evaluate. The area of element B is A_B.

$$M_{\alpha i\beta j} = \delta_{\alpha\beta}\delta_{ij}\tfrac{1}{4}\textstyle\sum_B \rho_B A_B \tag{2.4.3.3}$$

The expressions for the body forces, boundary tractions, and the concentrated loads are the same as in the plane case, but the forces are specified per radian.

2.4.4 Assembling the global force vector. The global force vector is created by summing all the contributions from the elements, the concentrated loads, and the boundary tractions. The summation procedure is the same for the different types of loading, therefore only the force contribution from the elements is discussed in detail.

Before the summation begins, the global force vector, F, is initialized to zero. In the previous sections when the force contributions from a particular element were calculated, α ranged from 1 to 4 and the numbering was local to the element. The array $IX(A,\alpha)$ stores the value of the global node number for the local node numbered α in element A. Assuming that the number of elements is N, the number of nodes for the particular element type is n_N, and the number of space dimensions is n_s, the assembly procedure is readily expressed in terms of three nested DO loops.

```
      DO 30 A = 1, N
        DO 20 $i = 1, n_s$
          DO 10 $\alpha = 1, n_N$
          $F_{IX(A,\alpha)i} = F_{IX(A,\alpha)i} - F^\sigma_{\alpha i}$          (2.4.4.1)
10        CONTINUE
20      CONTINUE
30    CONTINUE
```

From the standpoint of vectorization, the procedure defined in Eq. (2.4.4.1) is poor. More efficient methods are discussed in Ref. [19].

2.4.5 The constant strain triangle. Triangular elements are used in a few finite element hydrocodes, the most popular one being EPIC [20]. The primary advantage of a triangular element over a quadrilateral element is robustness, while the major deficiency is a tendency to give an overly stiff answer. Quadrilaterals can be distorted into "bow ties" and "boomerangs" (see Fig. 5), configurations that make it impossible to continue the calculation. As long as the time step is small enough, it is impossible to push a node on a triangular element through its opposite edge. The reason for this robustness is the volume approaches zero as the node approaches its opposite edge, leading to an infinite resisting pressure within the element.

The problem associated with the stiffness of triangular elements is readily illustrated by considering the simple two element problem in Fig. 6. All the nodes except one are constrained and the material is incompressible. The incompressibility constraint for element 1 limits the motion of the node to the z direction and similarly, element 2 limits the motion to the y direction. The node therefore cannot move at all, and the block behaves as a rigid solid. More complicated problems with a larger number of elements can be constructed [12]. A number of ways to circumvent this problem have been proposed [21], [22]. The simplest is to use four elements for each quadrilateral region [21], but the computational cost of using four triangular elements in place of a single quadrilateral is comparatively high.

The simplest way to calculate the nodal forces for a triangular element is to condense nodes 3 and 4 in a quadrilateral element [12]. The third interpolation function for the three node element, $N_3^{\triangle}$, is the average of the third and fourth interpolation functions for the quadrilateral element.

$$\begin{aligned} N_1^{\triangle} &= N_1 \\ N_2^{\triangle} &= N_2 \\ N_3^{\triangle} &= \tfrac{1}{2}(N_3 + N_4) \end{aligned} \tag{2.4.5.1}$$

The B matrix for the triangle is calculated by averaging the contributions of the third and fourth nodes in the quadrilateral.

$$[B] = \begin{bmatrix} z_{23} & 0 & z_{31} & 0 & -z_{21} & 0 \\ 0 & -r_{23} & 0 & -r_{31} & 0 & r_{21} \\ -r_{23} & z_{23} & r_{31} & z_{31} & r_{21} & -z_{21} \end{bmatrix} \tag{2.4.5.2}$$

The same results are obtained by introducing the triangular coordinates r, s, and t.

$$\begin{aligned} N_1^{\triangle} &= r \\ N_2^{\triangle} &= s \\ N_3^{\triangle} &= t(r,s) = 1 - r - s \end{aligned} \tag{2.4.5.3}$$

2.5 FINITE DIFFERENCE METHODS

There is a large body of literature associated with finite difference methods in hydrodynamics. The difference methods that have been developed differ too much to give a complete survey. Only the integral difference technique developed by Noh [9] as implemented in HEMP [7] by Wilkins will be covered in detail. PISCES [2] also uses the integral difference method for its Lagrangian calculations, but it switches to a Godunov method for the Eulerian calculations. Many other finite difference codes, including SHALE [23], SALE [24], SALE3D [25], TENSOR [6], and YAQUI [26], are popular and each has its own finite difference formulation. A separate section will discuss a finite difference formulation with edge-centered velocities.

2.5.1 The general integral difference strategy. This difference technique can be used with either cylindrical or planar coordinates, and it can be extended to three dimensions. Noh derived the integral difference method by combining Green's theorem with the mean value theorem. Green's theorem gives the relation between the boundary integral and area integral.

$$\begin{aligned} \int_{\Omega} g_y dydz &= \oint_{\Gamma} gdz \\ \int_{\Omega} g_z dydz &= -\oint_{\Gamma} gdy \end{aligned} \tag{2.5.1.1}$$

A simple example is the calculation of the area of the domain Ω.

$$A(\Omega) = \oint_{\Gamma} ydz = -\oint_{\Gamma} zdy \tag{2.5.1.2}$$

The mean value theorem states that the mean value of a function is achieved somewhere in the domain.

$$\begin{aligned}\overline{g,_y} &= \frac{1}{A}\int_\Omega g,_y \, dydz \\ \overline{g,_z} &= \frac{1}{A}\int_\Omega g,_z \, dydz\end{aligned} \tag{2.5.1.3}$$

Combining the previous three equations gives the mean values of the derivatives in terms of boundary integrals.

$$\begin{aligned}\overline{g,_y} &= \frac{\oint_\Gamma gdz}{\oint_\Gamma ydz} \\ \overline{g,_z} &= -\frac{\oint_\Gamma gdy}{\oint_\Gamma ydz}\end{aligned} \tag{2.5.1.4}$$

The boundary integrals are evaluated by progressing around a closed path defined by N points (y_i, z_i, g_i). Linear interpolation is used to evaluate g between adjacent point. Different interpolation functions for g would yield different approximations for the derivatives.

$$\begin{aligned}\overline{g,_y} &= \frac{\sum_{i=1}^N (g_{i+1}+g_i)(z_{i+1}-z_i)}{\sum_{i=1}^N (y_{i+1}+y_i)(z_{i+1}-z_i)} \\ \overline{g,_z} &= -\frac{\sum_{i=1}^N (g_{i+1}+g_i)(y_{i+1}-y_i)}{\sum_{i=1}^N (y_{i+1}+y_i)(z_{i+1}-z_i)}\end{aligned} \tag{2.5.1.5}$$

2.5.2 The acceleration from the stress. The acceleration from the stress is calculated by applying Eq. (2.5.1.4) directly to the strong form of the momentum equation. For the moment, the discussion is restricted to planar coordinates.

$$\begin{aligned}\dot{u}_i &= \frac{1}{\rho}\sigma_{ij},_j \\ \ddot{y} &= \frac{1}{\int_\Omega \rho d\Omega}\left\{\oint_\Gamma \sigma_{yy}dz - \oint_\Gamma \sigma_{yz}dy\right\} \\ \ddot{z} &= \frac{1}{\int_\Omega \rho d\Omega}\left\{\oint_\Gamma \sigma_{zy}dz - \oint_\Gamma \sigma_{zz}dy\right\}\end{aligned} \tag{2.5.2.1}$$

The boundary integrals are evaluated in the same manner as in Eq. (2.5.1.5). Two of the possible paths for Eq. (2.5.2.1) are illustrated in Fig. 7. Despite the fact that both evaluate the stress at only one point within the zone, the accelerations at the nodes are different.

In plane geometry, the path used in HEMP [7] gives nodal accelerations that are identical to the constant stress quadrilateral used in DYNA2D [18]. The numbering of the nodes and zones is shown in Fig. 7.

$$\begin{aligned}
\ddot{y} &= \frac{1}{M}\sum_{i=1}^{N}\{\sigma^i_{yy}(z_{i+1}-z_i)-\ \sigma^i_{yz}(y_{i+1}-y_i)\} \\
\ddot{z} &= \frac{1}{M}\sum_{i=1}^{N}\{\sigma^i_{zy}(z_{i+1}-z_i)-\sigma^i_{zz}(y_{i+1}-y_i)\} \\
M &= \tfrac{1}{2}\sum_{i=1}^{N}\rho_i A_i
\end{aligned} \tag{2.5.2.2}$$

The equivalence between the finite element and finite difference formulations is demonstrated by considering the individual terms in Eq. (2.5.2.2). One half of the mass from each surrounding zone is attributed to the node in the finite difference formulation while only one quarter is used in the finite element formulation. Equation (2.5.2.2) is multiplied by $\frac{1}{4}\sum_{i=1}^{N}\rho_i A_i$ to determine the equivalent nodal force.

$$\begin{aligned}
F_y &= \tfrac{1}{2}\left\{\sum_{i=1}^{N}\sigma^i_{yy}(z_{i+1}-z_i)-\sum_{i=1}^{N}\sigma^i_{yz}(y_{i+1}-y_i)\right\} \\
F_z &= \tfrac{1}{2}\left\{\sum_{i=1}^{N}\sigma^i_{zy}(z_{i+1}-z_i)-\sum_{i=1}^{N}\sigma^i_{zz}(y_{i+1}-y_i)\right\}
\end{aligned} \tag{2.5.2.3}$$

A term by term comparison of Eq. (2.5.2.3) to (2.4.2.11) shows that they are equivalent (note that the numbering of the nodes differs between the two equations).

2.5.3 The strain rate. Unlike calculating the accelerations, where there are a number of integral paths, the only path that makes sense for the strain rates is around the perimeter of the zone. For planar geometry, this gives the same strain rate as the finite element method, see Eq. (2.4.2.10). The subscript i refers to the node number in Eq. (2.5.3.1), and the numbering is assumed to be counterclockwise.

$$\begin{aligned}
\dot\epsilon_{yy} = \overline{\dot y_{,y}} = &\ \frac{1}{2A}\sum_{i=1}^{N}(\dot y_{i+1}+\dot y_i)(z_{i+1}-z_i) \\
\dot\epsilon_{zz} = \overline{\dot z_{,z}} = &-\frac{1}{2A}\sum_{i=1}^{N}(\dot z_{i+1}+\dot z_i)(y_{i+1}-y_i) \\
2\dot\epsilon_{zy} = \overline{\dot z_{,y}} + \overline{\dot y_{,z}} = &\ \frac{1}{2A}\sum_{i=1}^{N}\{(\dot z_{i+1}+\dot z_i)(z_{i+1}-z_i)-(\dot y_{i+1}+\dot y_i)(y_{i+1}-y_i)\}
\end{aligned} \qquad (2.5.3.1)$$

2.5.4 Axisymmetry. As in the finite element formulation, the equations for the accelerations of a node simply add a few terms involving $1/r$ to the planar equations. The current version of HEMP uses the area-weighted average of the extra $1/r$ terms from the surrounding zones, resulting in a formulation that is identical to the area-weighted element in DYNA2D. In the original description of HEMP [7], the arithmetic average of the extra terms is used instead, leading to spherical symmetry problems similar those in a volume-weighted finite element formulation.

$$\begin{aligned}
\ddot r &= \frac{1}{M}\sum_{i=1}^{N}\left\{\sigma^i_{rr}(z_{i+1}-z_i)-\sigma^i_{rz}(r_{i+1}-r_i)+\frac{A}{2}\Big(\frac{\sigma^i_{rr}-\sigma^i_{\theta\theta}}{r^o_i}\Big)\right\} \\
\ddot z &= \frac{1}{M}\sum_{i=1}^{N}\left\{\sigma^i_{zr}(z_{i+1}-z_i)-\sigma^i_{zz}(r_{i+1}-r_i)+\frac{A}{2}\Big(\frac{\sigma^i_{rz}}{r^o_i}\Big)\right\} \\
M &= \tfrac{1}{2}\sum_{i=1}^{N}\rho_i A_i
\end{aligned} \qquad (2.5.4.1)$$

The hoop strain rate is calculated so that the trace of the strain rate tensor equals the volumetric strain rate.

$$\dot\epsilon_{\theta\theta} = \frac{\dot V}{V} - \dot\epsilon_{rr} - \dot\epsilon_{zz} \qquad (2.5.4.2)$$

2.5.5 Spherical symmetry. One of the difficulties that plagued early finite difference and finite element hydrocodes was the jetting of material along the z axis in spherically convergent calculations. Many different approaches are used in the finite difference community to eliminate this problem, while the finite element community seems to have adopted the area-weighted Petrov-Galerkin approach. Margolin, for instance, has derived a difference stencil for SHALE [23] that is identical to the one-point finite element and the integral difference stencil in plane strain and different in axisymmetry. He observes that the center of mass of the momentum control volume around a node is not coincident with the node, and he compensates for the difference by altering the nodal accelerations. The basic axisymmetric difference stencil in PISCES is the same as the original one used in HEMP. Hancock [2] alters the forces acting at the nodes based on an analysis he performed to determine the magnitude of the force contribution which generate the unsymmetric response. Both methods work well.

2.6 LARGE DEFORMATION STRENGTH CALCULATIONS

The stress tensor can be split into the deviatoric stress, σ'_{ij}, which is assumed to be due to the material strength, and the pressure, P, which is calculated from an equation of state.

$$\begin{aligned} \sigma_{ij} &= \sigma'_{ij} - \delta_{ij} P \\ \sigma'_{ii} &= 0 \\ \tfrac{1}{3}\sigma_{ii} &= -P \end{aligned} \tag{2.6.1}$$

An increment in the stress during a time step is composed of two parts: the first part is associated with the deformation of the material and it is described by a constitutive model, and the second part is associated with the rigid body rotation of the material. For the most part, constitutive models are defined in terms of differential equations where the strain rate is the independent variable. During a pure rigid body rotation, the calculated strain rate must be zero to avoid a change in the stress due to the rigid rotation.

2.6.1 The time-centered strain rate. The strain rate used in hydrocodes is evaluated at time $t^{n+1/2}$ not only because the velocity is evaluated at that time, but also to avoid calculating a nonzero strain rate during a pure rigid body rotation. The following proof that the time-centered strain rate is zero for rigid rotations is given by Hallquist [27]. Note that the arguments presented here hold for axisymmetric problems since the strain rate $\dot{\epsilon}_{\theta\theta}$ is unaffected by rotations in the $r-z$ plane.

The time-centered geometry is defined as the average of the geometries at t^n and t^{n+1}.

$$x_i^{n+1/2} = \tfrac{1}{2}(x_i^n + x_i^{n+1}) \tag{2.6.1.1}$$

The square of the distance separating two point $P(x^n)$ and $Q(x^n + dx^n)$ is calculated with the familiar Euclidean norm.

$$dS^2 = dx_i^n dx_i^n \tag{2.6.1.2}$$

At the time t^{n+1}, a similar expression can be evaluated.

$$ds^2 = dx_i^{n+1} dx_i^{n+1} \tag{2.6.1.3}$$

The geometry at t^n and t^{n+1} can be related to the geometry at $n + 1/2$ because of the continuity of the deformation. The differentials at t^n and t^{n+1} are given in terms of the differentials at $t^{n+1/2}$ by Eq. (2.6.1.4).

$$\begin{aligned} dx_i^n &= \frac{\partial x_i^n}{\partial x_j^{n+1/2}} dx_j^{n+1/2} \\ dx_i^{n+1} &= \frac{\partial x_i^{n+1}}{\partial x_j^{n+1/2}} dx_j^{n+1/2} \end{aligned} \tag{2.6.1.4}$$

Substituting Eq. (2.6.1.4) into (2.6.1.2) and (2.6.1.3) and subtracting gives the change in the length squared in terms of the centered geometry.

$$ds^2 - dS^2 = \left(\frac{\partial x_i^{n+1}}{\partial x_j^{n+1/2}} \frac{\partial x_i^{n+1}}{\partial x_k^{n+1/2}} - \frac{\partial x_i^n}{\partial x_j^{n+1/2}} \frac{\partial x_i^n}{\partial x_k^{n+1/2}} \right) dx_j^{n+1/2} dx_k^{n+1/2} \tag{2.6.1.5}$$

The term enclosed by the parentheses defines an increment in strain between the two times.

$$\triangle\epsilon_{jk} = \tfrac{1}{2} \left(\frac{\partial x_i^{n+1}}{\partial x_j^{n+1/2}} \frac{\partial x_i^{n+1}}{\partial x_k^{n+1/2}} - \frac{\partial x_i^n}{\partial x_j^{n+1/2}} \frac{\partial x_i^n}{\partial x_k^{n+1/2}} \right) \tag{2.6.1.6}$$

By using the central difference integration, the geometry at the two times is related to the centered geometry by Eq. (2.6.1.4).

$$\begin{aligned} x_i^{n+1} &= x_i^{n+1/2} + \tfrac{1}{2} \triangle x_i^{n+1/2} \\ x_i^n &= x_i^{n+1/2} - \tfrac{1}{2} \triangle x_i^{n+1/2} \\ \triangle x_i^{n+1/2} &= x_i^{n+1} - x_i^n = u_i^{n+1/2} \triangle t \end{aligned} \tag{2.6.1.7}$$

The centered strain rate is calculated by substituting Eq. (2.6.1.7) into Eq. (2.6.1.6) and dividing through by the time step, $\triangle t$.

$$\begin{aligned}\dot{\epsilon}_{jk}^{n+1/2} &= \frac{\triangle\epsilon_{jk}}{\triangle t} \\ &= \tfrac{1}{2}\left(\frac{\partial u_j^{n+1/2}}{\partial x_k^{n+1/2}} + \frac{\partial u_k^{n+1/2}}{\partial x_j^{n+1/2}}\right)\end{aligned} \tag{2.6.1.8}$$

For a rigid body rotation, the velocity field is given by Eq. (2.6.1.9), where W is a skew symmetric matrix. If Eq. (2.6.1.9) is substituted into Eq. (2.6.1.8), then the strain rate will be zero.

$$u_i = W_{ij}x_j \tag{2.6.1.9}$$

It is also true that the strain increment is zero in the time discrete case. Assume that the rigid body transformation between n and $n+1$ is the rotation matrix Q. Note that the midpoint geometry does not correspond to the rigid body rotation half way between n and $n+1$.

$$\begin{aligned}x_i^{n+1} &= Q_{ij}x_j^n \\ x_i^n &= \delta_{ij}x_j^n \\ x_i^{n+1/2} &= \tfrac{1}{2}(\delta_{ij} + Q_{ij})x_j^n \neq \sqrt{Q_{ij}}x_j^n\end{aligned} \tag{2.6.1.10}$$

The velocity is defined in terms of the difference between the n and $n+1$ geometries.

$$u_i^{n+1/2} = \frac{1}{\triangle t}(Q_{ij} - \delta_{ij})x_j^n \tag{2.6.1.11}$$

The first derivative term in Eq. (2.6.1.8) is evaluated using the chain rule.

$$\begin{aligned}\frac{\partial u_j^{n+1/2}}{\partial x_k^{n+1/2}} &= \frac{\partial u_j^{n+1/2}}{\partial x_\ell^n}\frac{\partial x_\ell^n}{\partial x_k^{n+1/2}} \\ &= \frac{1}{\triangle t}(Q_{j\ell} - \delta_{j\ell})\frac{\partial x_\ell^n}{\partial x_k^{n+1/2}}\end{aligned} \tag{2.6.1.12}$$

Letting the rotation matrix be defined in the traditional manner, the remaining derivative is readily evaluated.

$$\begin{aligned}Q &= \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \\ \begin{Bmatrix} x_1^n \\ x_2^n \end{Bmatrix} &= \frac{1}{D}\begin{bmatrix} 1+\cos\theta & -\sin\theta \\ \sin\theta & 1+\cos\theta \end{bmatrix}\begin{Bmatrix} x_1^{n+1/2} \\ x_2^{n+1/2} \end{Bmatrix} \\ D &= 1+\cos\theta\end{aligned} \tag{2.6.1.13}$$

The first term in the discrete centered strain rate is skew symmetric, just like the matrix W, therefore the strain rate is again zero.

$$\begin{aligned}\frac{\partial u_j^{n+1/2}}{\partial x_k^{n+1/2}} &= \frac{1}{D\triangle t}\begin{bmatrix}\cos\theta-1 & \sin\theta\\ -\sin\theta & \cos\theta-1\end{bmatrix}\begin{bmatrix}1+\cos\theta & -\sin\theta\\ \sin\theta & 1+\cos\theta\end{bmatrix}\\ &= \frac{2}{(1+\cos\theta)\triangle t}\begin{bmatrix}0 & \sin\theta\\ -\sin\theta & 0\end{bmatrix}\end{aligned} \tag{2.6.1.14}$$

The diagonal terms of the time-centered strain increment agree closely with the true strain, $\ln(L^{n+1}/L^n)$. In one dimension, the centered strain increment, $\triangle\epsilon$, the Green strain, $\triangle E$, and the Almansi strain, $\triangle e$, are given by Eq. (2.6.1.15).

$$\begin{aligned}\triangle\epsilon &= \tfrac{1}{2}\left[\left(\frac{L^{n+1}}{L^{n+1/2}}\right)^2-\left(\frac{L^n}{L^{n+1/2}}\right)^2\right]\\ \triangle E &= \tfrac{1}{2}\left[\left(\frac{L^{n+1}}{L^n}\right)^2-1\right]\\ \triangle e &= \tfrac{1}{2}\left[1-\left(\frac{L^n}{L^{n+1}}\right)^2\right]\end{aligned} \tag{2.6.1.15}$$

For an extension, $\left(\frac{L^{n+1}}{L^n}\right)$, of 2, the true, centered, Green, and Almansi strains have values of 0.6931, 0.6667, 1.5000, and 0.3750, respectively.

2.6.2 Stress rates. The material time derivative of the stress is not objective [28], i.e., it does not transform properly as a tensor under a superposed rigid body motion. Consider, for instance, a motion that differs from another by only a rigid body transformation.

$$x^* = T(t) + Q(t)x \tag{2.6.2.1}$$

The transformation for the stress tensors in the two coordinate systems is differentiated with respect to time.

$$\begin{aligned}\sigma_{ij}^* &= Q_{ik}\sigma_{k\ell}Q_{\ell j}\\ \dot{\sigma}_{ij}^* &= \dot{Q}_{ik}\sigma_{k\ell}Q_{\ell j}+Q_{ik}\dot{\sigma}_{k\ell}Q_{\ell j}+Q_{ik}\sigma_{k\ell}\dot{Q}_{\ell j}\end{aligned} \tag{2.6.2.2}$$

Note that the material time derivative of the rotated stress involves the time derivatives of the transformation matrix Q. In order to have a theory that transforms properly (i.e., “objectively”), the material time derivative is replaced by an objective stress rate in the constitutive law for a material. The derivative used within the mathematics community is the Lie derivative [29] (within the mechanics community, it is referred to as the Oldroyd derivative [30]), while the three most popular stress rates in the mechanics literature are the Jaumann rate, $\dot{\sigma}^J$, the Truesdell rate, $\dot{\sigma}^T$, and the corotational rate, $\dot{\sigma}^c$, which is also known as the Green-Naghdi rate. A constitutive model can be formulated in terms of any one of these objective rates, and changing from one rate to another requires that the constitutive model be reformulated. W is the skew symmetric part of the velocity gradient L, and Ω is $\dot{R}R^t$, where R is from the polar decomposition, $F = RU$ [31]. The second Piola-Kirchhoff stress, S, is $JF^{-1}\sigma F^{-t}$, and the rotated Cauchy stress, Σ, is $R\sigma R^t$.

$$\begin{aligned}
\dot{\sigma}^J_{ij} &= \dot{\sigma}_{ij} - W_{ik}\sigma_{kj} + \sigma_{ik}W_{kj} \\
\dot{\sigma}^T_{ij} &= \dot{\sigma}_{ij} - L_{ik}\sigma_{kj} + \sigma_{ik}L_{kj} + \sigma_{ij}\mathrm{tr}(D) = \frac{1}{J}F_{ik}\dot{S}_{k\ell}F_{j\ell} \\
\dot{\sigma}^c_{ij} &= \dot{\sigma}_{ij} - \Omega_{ik}\sigma_{kj} + \sigma_{ik}\Omega_{kj} = R_{ik}\dot{\Sigma}_{k\ell}R_{j\ell} \\
\Sigma_{ij} &= R_{ki}\sigma_{k\ell}R_{\ell j}
\end{aligned} \tag{2.6.2.3}$$

The Jaumann rate corresponds to the Lie derivative in a Cartesian coordinate system under the assumption that the flow is purely rotational. Truesdell's rate is obtained by differentiating the expression for the Cauchy stress in terms of the second Piola-Kirchhoff stress, and the Green-Naghdi rate is obtained by differentiating the expression for the Cauchy stress in terms of the rotated stress.

The choice of an objective rate for a hydrocode is generally based on convenience and most hydrocodes use the Jaumann rate. It is not popular in implicit programs because it leads to an unsymmetric stiffness matrix. The major criticism against the use of the Jaumann rate in explicit codes is the oscillatory solution it generates under pure shear when kinematic hardening is used in J_2 plasticity theory [32]. Kinematic hardening is used to model the Bauschinger effect [33], which is the reduction in the yield stress associated with reversed loading. Under monotonic shear loading the shear stress should also increase monotonically, but at large shear strains (>1), the shear stress oscillates if the Jaumann rate is used. Neither of the other two stress rates has this problem. The Truesdell rate is unattractive for plasticity because it implies that the Cauchy stress is an exponential function of the strain, which is also unacceptable. By the process of elimination, the Green-Naghdi rate is the best choice in some people's minds, and at least one hydrocode PRONTO2D [34] uses it exclusively while other codes, such as DYNA2D [82], use the Green-Naghdi rate only with the kinematic hardening plasticity model.

From the computational point of view, the Green-Naghdi rate is not very attractive. It uses information about the Lagrangian coordinates of the material points, X, through the deformation gradient. Large deformation problems eventually need to have their solution mapped from one mesh to another, either periodically through a rezone or continuously, as in an ALE or Eulerian formulation. Assuming the solution variables are mapped to a spatial accuracy of $\mathcal{O}(\triangle x^r)$, their gradients will be one order less accurate, $\mathcal{O}(\triangle x^{r-1})$. Since at least second order accuracy is desired, the mapping scheme must be at least third order accurate, a difficult feat. One way to avoid the difficulties associated with mapping the Lagrangian coordinates is to introduce the deformation gradient as a solution variable and to map the gradient instead of the Lagrangian coordinates. Both the Jaumann and Green-Naghdi rate implementations are presented so that computational costs are clear.

The Jaumann rate is quite simple to implement. First, the spin, W, is calculated using either the finite element or finite difference methods (the two formulations presented in detail will give identical answers). The strain rate (actually the rate of deformation), D, and the stress increment, $\dot{\sigma}\triangle t$, are evaluated at $t^{n+1/2}$ in order to integrate the stress with the second order accurate central difference method. Since the stress increment is a function of the current stress, the stress at $t^{n+1/2}$ must be calculated from the stress at t^n.

$$\sigma_{ij}^{n+1/2} = \sigma_{ij}^n - \tfrac{\triangle t}{2} W_{ik}\sigma_{kj}^n + \tfrac{\triangle t}{2}\sigma_{ik}^n W_{kj} \qquad (2.6.2.4)$$

The increment in the stress is calculated from the constitutive model, where q is a vector of internal state variables (e.g., internal energy or plastic strain).

$$\triangle\sigma_{ij} = \triangle t\dot{\sigma}_{ij}(\sigma, D, q) \qquad (2.6.2.5)$$

$$\sigma_{ij}^{*} = \sigma_{ij}^{n+1/2} + \triangle\sigma_{ij} \qquad (2.6.2.6)$$

The intermediate value of the stress is given the remaining increment of the rotation.

$$\sigma_{ij}^{n+1} = \sigma_{ij}^{*} - \tfrac{\triangle t}{2} W_{ik}\sigma_{kj}^{*} + \tfrac{\triangle t}{2}\sigma_{ik}^{*} W_{kj} \qquad (2.6.2.7)$$

Note that the Jaumann rate uses only the current spatial coordinates and velocities. During a mapping of the solution from one mesh to another, no additional variables other than the current spatial solution variables need to be mapped.

One drawback to the half-step rotation form of centering is the stress may not lie exactly on the yield surface after the second half rotation. The magnitude of the error is second order in $\triangle t$, and Hancock has reported that it can be significant for some problems [166]. The simplest solution for this problem is to use "sloppy centering," i.e., by adding in the full rotational contribution to the stress ($\triangle t$ instead of $1/2\triangle t$ in Eq. (2.6.2.4)) before $\triangle\sigma_{ij}$ is evaluated. A second approach, which is more expensive, is to rotate the stress with an orthonormal Q. An example of this approach is the implementation of the Green-Naghdi rate.

The implementation of the Green-Naghdi rate is not quite as standardized as the Jaumann rate. Probably the simplest implementation is the one used in NIKE2D [27], an implicit nonlinear finite element program. The gradient of the Lagrangian coordinates is calculated using Eq. (2.4.2.7) with the Lagrangian coordinates replacing the velocities.

$$\frac{\partial X_i}{\partial x_j} = B_{ij\alpha k} X_{\alpha k} \tag{2.6.2.8}$$

The deformation gradient, F, is evaluated by inverting $\frac{\partial X_i}{\partial x_j}$. While the polar decomposition is fairly complicated in three dimensions, it is very simple in two [37], [38].

$$\begin{aligned} F_{ij} &= R_{ik} U_{kj} \\ R_{ki} F_{ij} &= U_{kj} = U_{jk} = F_{ji} R_{ik} \end{aligned} \tag{2.6.2.9}$$

The rotation matrix is calculated by substituting its general form into Eq. (2.6.2.9).

$$\begin{aligned} \text{let} \quad [R] &= \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \\ \sin\theta &= \frac{F_{12} - F_{21}}{\mathcal{F}} \\ \cos\theta &= \frac{F_{11} + F_{22}}{\mathcal{F}} \\ \mathcal{F} &= \sqrt{(F_{12} - F_{21})^2 + (F_{11} + F_{22})^2} \end{aligned} \tag{2.6.2.10}$$

Three evaluations of the polar decomposition are needed. The geometry at $n + 1/2$ is again evaluated as the simple average of the geometry at n and $n + 1$.

$$\begin{aligned}
\Sigma_{ij}^{n} &= R_{ki}^{n}\sigma_{k\ell}^{n}R_{\ell j}^{n} \\
\Sigma_{ij}^{n+1/2} &= R_{ki}^{n+1/2}\sigma_{k\ell}^{n}R_{\ell j}^{n+1/2} \\
d_{ij}^{n+1/2} &= R_{ki}^{n+1/2}D_{k\ell}^{n+1/2}R_{\ell j}^{n+1/2} \\
\triangle\Sigma_{ij} &= \dot{\Sigma}_{ij}(\Sigma^{n+1/2}, d^{n+1/2})\triangle t \\
\Sigma_{ij}^{n+1} &= \Sigma_{ij}^{n} + \triangle\Sigma_{ij} \\
\sigma_{ij}^{n+1} &= R_{ik}^{n+1}\Sigma_{k\ell}^{n+1}R_{j\ell}^{n+1}
\end{aligned} \tag{2.6.2.11}$$

The deformation gradient is integrated with the central difference algorithm if it is treated as a solution variable.

$$\begin{aligned}
\dot{F}_{ij}^{n+1/2} &= D_{ik}^{n+1/2}F_{kj}^{n+1/2} \\
F_{ij}^{n+1} &= F_{ij}^{n} + \triangle t\dot{F}_{ij}^{n+1/2} \\
F_{ij}^{n+1/2} &= \frac{\partial\frac{1}{2}(x_i^{n+1}+x_i^{n})}{\partial X_j} = \tfrac{1}{2}\left(F_{ij}^{n+1}+F_{ij}^{n}\right)
\end{aligned} \tag{2.6.2.12}$$

$$\text{therefore } (\delta_{ik} - \tfrac{1}{2}\triangle tD_{ik}^{n+1/2})F_{kj}^{n+1} = (\delta_{i\ell} + \tfrac{1}{2}\triangle tD_{i\ell}^{n+1/2})F_{k\ell}^{n}$$

This approach still requires that the polar decomposition be performed on the deformation gradient. The polar decomposition is simple in two dimensions, but in three dimensions, it is computationally expensive.

An alternative formulation used by Flanagan and Taylor in PRONTO2D [34] to directly integrate R is based on relations derived by Dienes [39] . V, from the left polar decomposition, $F = VR$, is assumed to be known. Following Dienes' work, the vector z is calculated, where e_{ijk} is the permutation operator.

$$\begin{aligned}
z_i &= e_{ijk}V_{jm}D_{mk} \\
\omega_i &= e_{ijk}W_{jk} - 2\left(V - \mathrm{I}\ \mathrm{tr}(V)\right)_{ij}^{-1} z_j \\
\Omega_{ij} &= \tfrac{1}{2}e_{ijk}\omega_k
\end{aligned} \tag{2.6.2.13}$$

An incremental rotation matrix is calculated using a centered approximation that was originally implemented by Hallquist in NIKE2D[27]. A full analysis of this approximation was given by Hughes and Winget [40].

$$Q_{ij} = \left(\mathrm{I} - \frac{\triangle t}{2}\Omega\right)_{ik}^{-1}\left(\mathrm{I} + \frac{\triangle t}{2}\Omega\right)_{kj} \tag{2.6.2.14}$$

The symmetric tensor V is integrated to complete the calculation. At the start of the analysis, V is the identity matrix. Note that both V and R must be stored for each element.

$$\begin{aligned} \dot{V}_{ij} &= L_{ik}V_{kj} - V_{ik}\Omega_{kj} \\ V_{ij} &= V_{ij} + \triangle t\dot{V}_{ij} \end{aligned} \tag{2.6.2.15}$$

In contrast to the time-centered integration in Eq. (2.6.2.11), Flanagan and Taylor use only R^{n+1} to evaluate the stress at t^{n+1}.

$$\begin{aligned} d_{ij} &= R^{n+1}_{ki} D^{n+1/2}_{k\ell} R^{n+1}_{\ell j} \\ \triangle\Sigma_{ij} &= \dot{\Sigma}_{ij}(\Sigma, d) \\ \Sigma^{n+1}_{ij} &= \Sigma^{n}_{ij} + \triangle\Sigma_{ij} \\ \sigma^{n+1}_{ij} &= R^{n+1}_{ik}\Sigma^{n+1}_{k\ell}R^{n+1}_{j\ell} \end{aligned} \tag{2.6.2.16}$$

One advantage of treating V and R as history variables and remapping them is it avoids the reduction in the accuracy of R associated with using the remapped Lagrangian coordinates, X, in the difference formula for the deformation gradient. The rotation tensor can be remapped using θ, or each component can be remapped independently provided that R is orthonormalized later. Remapping the two additional tensors, however, does add significantly to the computational cost.

2.7 ENERGY CALCULATIONS

The pressure, P, is a function of the internal energy per unit mass, e, and density, ρ, and possibly some set of additional history variables, h. The evolution of the history variables depends on the particular equation of state. To simplify the discussion, it is assumed that the pressure is a function only of the internal energy and density in the remainder of this discussion. From the standpoint of simplifying the calculations, it is convenient to use the total internal energy of an element, E.

$$\begin{aligned} E &\equiv \int_\Omega \rho e d\Omega \\ e &= \frac{E}{M} \end{aligned} \tag{2.7.1}$$

Not only does the use of the total internal energy simplify the discussion of the energy calculations, it makes the implementation of equations of state that use the internal energy per unit volume or the temperature easier since the only change is to replace Eq. (2.7.1) with the appropriate equation.

The contributions to the energy equation from the body forces and heat conduction are ignored for simplicity. Furthermore, the work associated with the strength of the material, the shock viscosity, q, and the pressure are treated separately. The tensors are written in terms of their deviatoric, $(\cdot)'$, and mean components, $\frac{1}{3}\mathrm{tr}(\cdot)$. All the tensors are assumed to be symmetric, although the shock viscosity terms may be unsymmetric [41], [42]. The appropriate changes to the energy calculation for an unsymmetric shock viscosity are discussed later.

$$\begin{aligned}\rho\dot{e} &= \sigma_{ij}\dot{\epsilon}_{ij}\\ &= \left(\sigma'_{ij} - P\delta_{ij} + q'_{ij} + \tfrac{1}{3}\mathrm{tr}(q)\delta_{ij}\right)\left(\dot{\epsilon}'_{ij} + \tfrac{1}{3}\mathrm{tr}(\dot{\epsilon})\right)\\ &= \left(\sigma'_{ij} + q'_{ij}\right)\dot{\epsilon}'_{ij} + \left(\tfrac{1}{3}\mathrm{tr}(q) - P\right)\frac{\dot{V}}{V}\end{aligned} \tag{2.7.2}$$

The stress from the strength of the material is assumed to be purely deviatoric, and the fact that the trace of the deformation rate is equal to the volumetric strain rate is used to simplify Eq. (2.7.2).

The pressure is evaluated at time t^n, which forces the internal energy to also be centered at t^n. While it is highly desirable for the numerical algorithm to conserve the total energy of the problem exactly, the goal is elusive because the velocity is centered at $t^{n+1/2}$ and the internal energy is centered at t^n. The conservation of the total energy therefore depends not only on the numerical algorithm, but also on the definition chosen for the total energy.

2.7.1 Forward Euler integration. The first hydrocodes integrated the energy equation with the forward Euler method, Eq. (2.7.1.1).

$$\begin{aligned}\triangle E &= \triangle t\left[\left(\sigma'^n_{ij} + q'^n_{ij}\right)\dot{\epsilon}'^{n+1/2}_{ij}V^n\right] + \left(\tfrac{1}{3}\mathrm{tr}(q^n) - P^n\right)\triangle V\\ E^{n+1} &= E^n + \triangle E\end{aligned} \tag{2.7.1.1}$$

This method does not conserve energy exactly and it is only first order accurate.

2.7.2 Central difference rule. Wilkins uses the central difference rule in HEMP [7], and it is also used in DYNA2D, DYNA3D [18], PRONTO2D [34], and many other Lagrangian hydrocodes. The integration of the internal energy is performed exactly if the stress, shock viscosity, and the pressure vary linearly over the time step.

$$
\begin{aligned}
E^{n+1} &= E^n + \triangle t\big[\left(\sigma_{ij}^{\prime n+1/2} + q_{ij}^{\prime n+1/2}\right) \dot{\epsilon}_{ij}^{\prime n+1/2} V^{n+1/2}\big] \\
&\quad + \left(\tfrac{1}{3}\mathrm{tr}(q^{+1/2}) - P^{n+1/2}\right) \triangle V^{n+1/2} \\
\sigma_{ij}^{\prime n+1/2} &= \tfrac{1}{2}(\sigma_{ij}^{\prime n} + \sigma_{ij}^{\prime n+1}) \\
q_{ij}^{\prime n+1/2} &= \tfrac{1}{2}(q_{ij}^{\prime n} + q_{ij}^{\prime n+1}) \\
P_{ij}^{\prime n+1/2} &= \tfrac{1}{2}(P_{ij}^{\prime n} + P_{ij}^{\prime n+1})
\end{aligned}
\tag{2.7.2.1}
$$

The energy at t^{n+1} in Eq. (2.7.2.1) is a function of the pressure at t^{n+1}, which makes the equation implicit in E^{n+1} since the pressure is a function of the energy. The deviatoric stress and shock viscosity are assumed to be independent of or at least comparatively insensitive to the value of e^{n+1}. An intermediate value of energy, $\tilde{E}$, which includes all the known terms in the right hand side of Eq. (2.7.2.1) allows the discrete energy equation to be expressed in a simpler form.

$$
\begin{aligned}
E^{n+1} &= \tilde{E} - \tfrac{1}{2}P^{n+1}(E^{n+1}/M, \rho^{n+1})\triangle V^{n+1/2} \\
\tilde{E} &= E^n + \triangle t\big[\left(\sigma_{ij}^{\prime n+1/2} + q_{ij}^{\prime n+1/2}\right) \dot{\epsilon}_{ij}^{\prime n+1/2} V^{n+1/2}\big] \\
&\quad + \left(\tfrac{1}{3}\mathrm{tr}(q^{n+1/2}) - \tfrac{1}{2}P^n\right) \triangle V^{n+1/2}
\end{aligned}
\tag{2.7.2.2}
$$

Most of the commonly used equations of state, such as the Mie-Grüneisen equation of state, are linear in the internal energy and can be written as the sum of two functions, A_1 and A_2. The equation of state is written in terms of the total internal energy of the element for generality, and a scale factor of $1/M$ or $1/V$ is necessary in A_2 to put it terms of the energy per unit mass or volume.

$$
P(e, \rho) = A_1(\rho) + A_2(\rho)E \tag{2.7.2.3}
$$

Substituting Eq. (2.7.2.3) into Eq. (2.7.2.2) gives an explicit formula for the energy at t^{n+1}.

$$
E^{n+1} = \frac{\tilde{E}}{1 + \frac{1}{2}\triangle V^{n+1/2} A_2(\rho^{n+1})} \tag{2.7.2.4}
$$

When the pressure is not a linear function of the internal energy, a simple successive substitution algorithm is so frequently used to solve the nonlinear equation for the energy that its origins are difficult to trace. The subscript i refers to the iteration number and P_0^{n+1} is P^n.

$$E_{i+1}^{n+1} = \tilde{E} - \tfrac{1}{2}P(\rho, E_i^{n+1}/M)\triangle V \tag{2.7.2.5}$$

The method is stable as long as the magnitude of the energy increment, $|\triangle E_i|$, from each iteration decreases. A Taylor expansion of Eq. (2.7.2.5) gives the necessary condition for this to happen. This condition is usually met, and only a few iterations are necessary for convergence.

$$\begin{aligned}\triangle E_i &= E_i^{n+1} - E_{i-1}^{n+1} \\ &\approx -\frac{1}{2\rho}\frac{\partial P}{\partial e}\frac{\triangle V}{V}\triangle E_{i-1} \\ 1 \geq |\frac{1}{2\rho}\frac{\partial P}{\partial e}\frac{\triangle V}{V}|\end{aligned} \tag{2.7.2.6}$$

The central difference energy calculation is not exactly equal to the central difference approximation of the work done by an element on its nodes. To demonstrate the level of approximation, only the planar case is considered.

$$\begin{aligned}\triangle E &= -\tfrac{1}{2}(F_{\alpha i}^{n+1} + F_{\alpha i}^{n})u_{\alpha i}^{n+1/2}\triangle t \\ &= \tfrac{1}{2}(B_{k\ell\alpha i}^{n+1}\sigma_{k\ell}^{n+1}V^{n+1} + B_{k\ell\alpha i}^{n}\sigma_{k\ell}^{n}V^{n})u_{\alpha i}^{n+1/2}\triangle t \\ &\approx \tfrac{1}{2}B_{k\ell\alpha i}^{n+1/2}V^{n+1/2}(\sigma_{k\ell}^{n+1} + \sigma_{k\ell}^{n})u_{\alpha i}^{n+1/2}\triangle t \\ &= \tfrac{1}{2}(\sigma_{k\ell}^{n+1} + \sigma_{k\ell}^{n})\dot{\epsilon}_{k\ell}^{n+1/2}V^{n+1/2}\triangle t\end{aligned} \tag{2.7.2.7}$$

The approximation appearing in Eq. (2.7.2.7) is usually a very good one because the time step limitation prevents the element geometry from changing greatly during any one time step.

The total energy is conserved according to Eq. (2.7.2.8) if the kinetic energy, T^n, at t^n is defined in a particular (and not necessarily intuitive) manner. This definition of the kinetic energy dates at least back to the paper [165] by Truillo and Trigger in 1961. One drawback to this definition is that the kinetic energy can be negative.

$$\begin{aligned}\triangle W &= \tfrac{1}{2}\left(F_{\alpha i}^{n+1} + F_{\alpha i}^{n}\right)\dot{x}_{\alpha i}^{n+1/2}\triangle t \\ &= \tfrac{1}{2}\sum_{\alpha i} M_{\alpha i}\left(\frac{u_{\alpha i}^{n+3/2} - u_{\alpha i}^{n+1/2}}{\triangle t} + \frac{u_{\alpha i}^{n+1/2} - u_{\alpha i}^{n-1/2}}{\triangle t}\right)u_{\alpha i}^{n+1/2}\triangle t \\ &= \tfrac{1}{2}\sum_{\alpha i} M_{\alpha i}\left(u_{\alpha i}^{n+3/2}u_{\alpha i}^{n+1/2} - u_{\alpha i}^{n+1/2}u_{\alpha i}^{n-1/2}\right) \\ &= T^{n+1} - T^{n}\end{aligned} \tag{2.7.2.8}$$

$$T^n = \tfrac{1}{2}\sum_{\alpha i} M_{\alpha i} u_{\alpha i}^{n+1/2} u_{\alpha i}^{n-1/2} \tag{2.7.2.9}$$

A Taylor expansion about t^n demonstrates that Eq. (2.7.2.9) is a second order accurate approximation of the kinetic energy at t^n.

$$\begin{aligned} u_{\alpha i}^{n-1/2} &= u_{\alpha i}^{n} - \dot{u}_{\alpha i}^{n}\triangle t/2\ldots \\ u_{\alpha i}^{n+1/2} &= u_{\alpha i}^{n} + \dot{u}_{\alpha i}^{n}\triangle t/2\ldots \\ T^n &= \tfrac{1}{2}\sum_{\alpha i} M_{\alpha i}\left\{(u_{\alpha i}^{n})^2 - (\triangle t)^2(\dot{u}_{\alpha i}^{n})^2/4\ldots\right\} \end{aligned} \tag{2.7.2.10}$$

The more intuitive approximation of the kinetic energy, which just averages the velocity at $t^{n-1/2}$ and $t^{n+1/2}$ for the velocity at t^n is also second order accurate, but it is no more accurate than the approximation in Eq. (2.7.2.9).

$$\begin{aligned} T^n &= \tfrac{1}{8}\sum_{\alpha i} M_{\alpha i}(u_{\alpha i}^{n-1/2} + u_{\alpha i}^{n+1/2})^2 \\ &= \tfrac{1}{2}\sum_{\alpha i} M_{\alpha i}\left\{(\dot{x}^n)^2 + (\triangle t)^2\ddot{u}^n/4\ldots\right\} \end{aligned} \tag{2.7.2.11}$$

2.7.3 The operator split. The previous method requires the solution of an equation that is implicit in the energy. It also requires the shock viscosity at the previous time step, which is not a problem for Lagrangian calculations. In Eulerian calculations, it is necessary to account for the transport of all the solution variables between the elements. The value of the shock viscosity at t^n is calculated directly from the velocities at $t^{n-1/2}$ and it would not need to be mapped if it was only needed for the acceleration calculations. The energy calculations in Section 2.7.2 require the previous value, making the additional computational cost of mapping of the shock viscosity necessary.

An additional problem with mapping the viscosities can occur. The viscosity is added to convert kinetic energy into internal energy within a shock. If the viscosity is remapped into an adjacent element that is expanding, it will reduce the internal energy of that element, an undesirable result. An alternative approach is derived by considering the change in the kinetic energy associated with applying the nodal forces at time t^n.

$$\begin{aligned}
-\triangle E = \triangle T &= \tfrac{1}{2} M_{\alpha i} \left[(u_{\alpha i}^{n+1/2})^2 - (u_{\alpha i}^{n-1/2})^2 \right] \\
&= \tfrac{1}{2} \sum_{\alpha i} M_{\alpha i} (u_{\alpha i}^{n+1/2} - u_{\alpha i}^{n-1/2})(u_{\alpha i}^{n+1/2} + \dot{x}_{\alpha i}^{n-1/2}) \\
&= \tfrac{1}{2} \sum_{\alpha i} F_{\alpha i}^{n} \triangle t (u_{\alpha i}^{n+1/2} + u_{\alpha i}^{n-1/2}) \qquad (2.7.3.1) \\
&= \tfrac{1}{2} \sum_{\alpha i} -B_{k\ell\alpha i}^{n} \sigma_{k\ell}^{n} V^{n} \triangle t (u_{\alpha i}^{n+1/2} + u_{\alpha i}^{n-1/2}) \\
E^{n+1} &= E^{n} + \triangle E
\end{aligned}$$

By construction, the total energy is conserved if it is defined as the sum of the internal energy at t^{n+1} and the kinetic energy at $t^{n+1/2}$. This procedure can be rewritten to look more like the differential energy equation provided that the strain rate at the discrete times is correctly defined.

$$\begin{aligned}
\triangle E &= \tfrac{1}{2} (\dot{\epsilon}_{k\ell}^{n+1/2} + \dot{\epsilon}_{k\ell}^{n-1/2}) \sigma_{k\ell}^{n} V^{n} \triangle t \\
\dot{\epsilon}_{k\ell}^{(\cdot)} &\equiv B_{k\ell\alpha i}^{n} u_{\alpha i}^{(\cdot)}
\end{aligned} \qquad (2.7.3.2)$$

It is desirable to include the work performed by the deviatoric stress and the shock viscosity into the energy used for the equation of state evaluation. This is achieved via an operator split: it is assumed that the nodes can be accelerated sequentially. First the nodes are accelerated by the deviatoric stress and the shock viscosity.

$$\dot{u}_{\alpha i}^{n} = - \sum_{elements} B_{k\ell\alpha i}^{n} (\sigma_{k\ell}^{n} + q_{k\ell}^{n}) / M_{\alpha i} \qquad (2.7.3.3)$$

An intermediate value of the velocity is calculated using the partial forces and the full time step.

$$\tilde{u}_{\alpha i} = u_{\alpha i}^{n-1/2} + \triangle t \dot{u}_{\alpha i}^{n} \qquad (2.7.3.4)$$

The energy is updated using Eq. (2.7.3.1) with $\tilde{u}_{\alpha i}$ replacing $u_{\alpha i}^{n+1/2}$, giving an energy increment $\triangle E^{\sigma}$. This gives an intermediate value for the energy, $\tilde{E}$. It is used to evaluate the equation of state, and the pressure is used to accelerate the nodes.

$$\dot{\tilde{u}}_{\alpha i} = \sum_{elements} B_{kk\alpha i}^{n} P / M_{\alpha i} \qquad (2.7.3.5)$$

The final value for the velocity is evaluated by adding the contribution from the pressure.

$$u_{\alpha i}^{n+1/2} = \tilde{u}_{\alpha i} + \triangle t \dot{\tilde{u}}_{\alpha i} \tag{2.7.3.6}$$

Using the velocities $u_{\alpha i}^{n+1/2}$ and $\tilde{u}_{\alpha i}$ in Eq. (2.7.3.1), the energy increment $\triangle E^P$ is added to advance $\tilde{E}$ to E^{n+1}.

This procedure is conservative and the final velocity is not altered by the operator split if the dependence of pressure forces on $\tilde{u}$ through $\tilde{E}$ is ignored. To demonstrate these results, the forces due to the stress and the viscosity are denoted F^σ and the forces due to the pressure, F^P. The second result is trivially obtained by noting that the explicit central difference method is linear in the acceleration.

$$\begin{aligned} \tilde{u}_{\alpha i} &= u_{\alpha i}^{n-1/2} + \triangle t F_{\alpha i}^\sigma / M_{\alpha i} \quad \text{no sum on } \alpha i \\ u_{\alpha i}^{n+1/2} &= \tilde{u}_{\alpha i} + \triangle t F_{\alpha i}^P / M_{\alpha i} \\ &= u_{\alpha i}^{n-1/2} + \triangle t \left(F_{\alpha i}^\sigma + F_{\alpha i}^P \right) / M_{\alpha i} \end{aligned} \tag{2.7.3.7}$$

The first result is obtained by adding and subtracting the kinetic energy associated with the intermediate velocity from the difference in kinetic energy at $t^{n+1/2}$ and $t^{n-1/2}$.

$$\begin{aligned} \triangle T &= \tfrac{1}{2} \sum_{\alpha i} M_{\alpha i} \Big[\big((u_{\alpha i}^{n+1/2})^2 - (\tilde{u}_{\alpha i})^2 \big) + \big((\tilde{u}_{\alpha i})^2 - (u_{\alpha i}^{n-1/2})^2 \big) \Big] \\ &= \tfrac{1}{2} \triangle t \sum_{\alpha i} \Big[F_{\alpha i}^P \big(u_{\alpha i}^{n+1/2} + \tilde{u}_{\alpha i} \big) + F_{\alpha i}^\sigma \big(\tilde{u}_{\alpha i} + u_{\alpha i}^{n-1/2} \big) \Big] \\ &= -\triangle E^P - \triangle E^\sigma \end{aligned} \tag{2.7.3.8}$$

2.7.4 Runge-Kutta methods. As discussed in a previous section, some hydrocodes use a second order accurate Runge-Kutta method to advance the solution in time. In addition to centering all the solution variables at a common location in time, it has the additional advantages that no iterations are necessary on the energy equation and the method conserves energy. While the Runge-Kutta scheme requires that the pressure and stress be evaluated twice per timestep, the fact that it avoids an iterative solution of the energy equation may make it more efficient than the central difference method for problems that have tabular or complicated equations of state.

The energy at $t^{n+1/2}$ is evaluated by integrating from t^n with forward Euler. The increment in the energy from t^n to t^{n+1} is calculated by using Eq. (2.7.2.1), but the actual quantities centered at $t^{n+1/2}$ are used instead of the average of the t^n and t^{n+1} values.

This seemingly small difference between the Runge-Kutta and central difference method accounts for the lack of iterations in the Runge-Kutta method, i.e., the coupling between E^{n+1} and P^{n+1} is eliminated.

The energy conservation proof follows the proof in Eq. (2.7.3.1) very closely except for some slight differences in the velocity and stress centering. Note that the kinetic energy and the internal energy are centered at the same time, and that the kinetic energy must always be positive (in contrast to the central difference method).

$$\begin{aligned} -\triangle E = \triangle T &= \tfrac{1}{2} M_{\alpha i} \left[(u_{\alpha i}^{n+1})^2 - (u_{\alpha i}^{n})^2 \right] \\ &= \tfrac{1}{2} \sum_{\alpha i} M_{\alpha i} (u_{\alpha i}^{n+1} - u_{\alpha i}^{n})(u_{\alpha i}^{n+1} + u_{\alpha i}^{n}) \\ &= \tfrac{1}{2} \sum_{\alpha i} F_{\alpha i}^{n+1/2} \triangle t (u_{\alpha i}^{n+1} + u_{\alpha i}^{n}) \\ &= \tfrac{1}{2} \sum_{\alpha i} -B_{k\ell\alpha i}^{n+1/2} \sigma_{k\ell}^{n+1/2} V^{n+1/2} \triangle t (u_{\alpha i}^{n+1} + u_{\alpha i}^{n}) \\ E^{n+1} &= E^{n} + \triangle E \end{aligned} \tag{2.7.4.1}$$

2.7.5 Energy calculations for axisymmetric problems. The methods presented above are almost unchanged for axisymmetric problems except for the volume calculations. When one point integration is used in the finite element method, the area of the element is approximated as $4J(x^o)$, which is exact for a planar quadrilateral element. For the axisymmetric case, the approximate volume is $4r^o J(r^o, z^o)$, which is not exact, and the error is the largest near the axis. In the finite difference method, the volume is usually evaluated exactly by dividing the quadrilateral into triangles defined by the nodes 1-2-3 and 1-3-4. The volume for one radian of θ for a quadrilateral is given by adding the two triangular volumes. It is common practice to also use Eq. (2.7.5.1) in finite element programs.

$$\begin{aligned} V &= \tfrac{1}{3}(r_1 + r_2 + r_3) A_{123} + \tfrac{1}{3}(r_1 + r_3 + r_4) A_{134} \\ A_{ijk} &= \tfrac{1}{2}(r_{ji} z_{ki} - r_{ki} z_{ji}) \end{aligned} \tag{2.7.5.1}$$

The proofs of energy conservation in the previous sections relied on a constant mass matrix, but the mass matrix is time dependent when area-weighted finite element or finite difference methods are used to solve axisymmetric problems. This subtlety has been studied in detail by Burton [166], who has developed methods that are energy conservative by adding momentum flux terms into the energy updates. For most applications, the energy error is small, but Crowley demonstrated that it can be quite large in a blast wave problem.

2.8 THE SHOCK VISCOSITY IN ONE DIMENSION

Researchers have realized ever since shock calculations were first attempted [43] that special computational methods are necessary to handle the jump discontinuities associated with shocks. A standard approach in hydrodynamic calculations is the introduction of a viscous term [43] into the equations. Within the context of strong shocks in an ideal gas, the viscosity can be related to the thermodynamic irreversibility of the shock [44]. In the acoustic limit there is no entropy increase, and the role of the viscosity is to damp out the postshock "ringing" that occurs in the discrete solution. It can be shown that the weak form of a hyperbolic system of equations can admit more than one solution [167]. If the system is modified to include viscous terms, the physically meaningful solution to the original problem is the limiting solution as the viscous terms approach zero and satisfy and entropy condition [168]. By adding a viscosity, even in accoustic problems, the numerical solution of the weak form also converges to the physically meaningful solution.

Although the addition of a shock viscosity has proven to be a simple and robust approach to solving shock problems, the approach is not without its drawbacks. Shocks are typically spread over four to six elements, and the density of the mesh must be fine enough so that the six element shock width is small in comparison to the dimensions of the problem. Adaptive methods have been developed so that a fine mesh is used only within the shock. In explicit calculations, the time step size is limited by the size of the smallest element, consequently, the elements within the shock control the time step for adaptive methods. The efficiency of the method is heavily dependent on the coarseness of the mesh outside the shock and the speed with which the mesh can be changed. On supercomputers, continuously re-blocking the elements for vectorization adds an additional burden on adaptive methods.

Shock viscosities introduce errors and problems of their own. In strong shocks, "wall heating" and "shockless heating" occur [45], leading to errors in the energy behind, and even in front of, the shock. Errors in the energy lead to errors in the density and the shock speed.

Instabilities in the solution can be created, especially when the elements are distorted or have large aspect ratios [41]. Shock viscosities that produce thinner shocks or have less heating error than the standard formulations generally worsen the instabilities. While distorted elements and elements with large aspect ratios are undesirable for other reasons, they are common in problems involving large deformations.

2.8.1 The Von Neumann and Richtmyer Viscosity. The paper by Von Neumann and Richtmyer [43] was concerned only with one-dimensional planar shocks. They wanted a numerical method that would automatically treat shocks without the need for introducing an internal boundary condition that moved with the shock. Quoting their paper, their approach "utilizes the well-known effect on shocks of dissipative mechanisms, such as viscosity and heat conduction. When viscosity is taken into account, for example, the shocks are seen to be smeared out ... Our idea is to introduce (artificial) dissipative terms ... to give the shocks a thickness comparable ... to the spacing of the points of the network." They introduced a pressure-like term, designated q, that is added consistently to the pressure, P, in all of the governing equations.

$$\begin{aligned}\rho\frac{\partial u}{\partial t} &= -\frac{\partial(P+q)}{\partial x}\\ \frac{\partial \varepsilon}{\partial t} &= -(P+q)\frac{\partial u}{\partial x}\end{aligned}\qquad(2.8.1.1)$$

The particular form they chose is quadratic in the strain rate. The dimensionless constant, c, is close to unity, and the mesh spacing is denoted $\triangle x$.

$$q = -\rho(c\triangle x)^2\frac{\partial u}{\partial x}\left|\frac{\partial u}{\partial x}\right|\qquad(2.8.1.2)$$

Using the standard approximation for the strain rate in a two node element, the viscosity can be written in terms of the velocity jump across the element.

$$\frac{\partial u}{\partial x} = \frac{\triangle u}{\triangle x}\qquad(2.8.1.3)$$

$$q = -\rho c^2\triangle u\,|\triangle u|\qquad(2.8.1.4)$$

The viscosity term is positive for shocks and negative for rarefactions. Rarefactions in gases usually do not need the viscosity for a smooth solution. It is common practice today to turn on the q only in compression for a gas, and set it to zero everywhere else. For acoustic shocks in solids, the q is sometimes left on all the time (for both compression and expansion) to add extra damping. This practice is followed throughout the remainder of this paper except when otherwise specified.

Von Neumann and Richtmyer solved the governing equations with the viscosity included for a steady shock in an ideal gas. The length scale introduced by the mesh spacing,

$\triangle x$, appears in their solution. Using the analytical solution, they were able to prove for ideal gases that their viscosity eliminated all discontinuities, and that the shock profile was sinusoidal. They also demonstrated that the shock was about six elements wide and independent of the shock strength. Most of the jump in the solution occurs over three to four elements in practice. Their analytical solution satisfies the Rankine-Hugoniot equations when the pre- and postshock conditions are evaluated at $\pm\infty$. The viscosity method is convergent in the sense that the error in satisfying the Rankine-Hugoniot equations can be held to any desired value with a fine enough mesh.

Treating the viscosity as a pressure term the momentum and energy equations allows the work done by the viscosity to be identified with the thermodynamic irreversibility of the shock by comparing the energy equation to the first law [44].

$$\begin{aligned} \mathrm{d}\varepsilon &= -(P+q)\mathrm{d}V \\ T\mathrm{d}s &= \mathrm{d}\varepsilon + P\mathrm{d}V \\ \rightarrow\ T\mathrm{d}s &= -q\mathrm{d}V \end{aligned} \tag{2.8.1.5}$$

2.8.2 Artificial Heat Conduction. As noted by Von Neumann and Richtmyer, heat conduction also plays a role in smearing out shocks. Noh [45] advocates adding an artificial heat conduction term to the energy equation. He convincingly demonstrates that the artificial heat conduction eliminates most of the heating errors that occur with all viscosities. Before the internal energy of an element is updated at the end of the time step, an intermediate value of the internal energy is used to calculate the thermal diffusion in the shock. This procedure requires the intermediate energy for all the elements in the shock front.

Implementing artificial heat conduction in a finite difference code is not difficult, but it is a problem in a finite element code due to the lack of a logically regular mesh. When a given element is being updated, there is no guarantee that all the intermediate energies of the surrounding elements have been calculated. Two passes could be made over all the elements, but that would be prohibitively expensive. In finite difference programs, it is possible to order the calculations over strips along a logical mesh direction in such a way as to guarantee that all of the necessary energies have been calculated.

2.8.3 The HEMP Artificial Viscosity. HEMP [46] uses a combination of linear and quadratic viscosities, see Landshoff [47], with a method for calculating the shock direction developed by Richards and Wilkins [48]. Landshoff noticed that the quadratic viscosity works reasonably well, but small oscillations still occur after the shock and that while linear viscosities were used in some papers, they were overly diffusive. He used a linear combination of the two viscosities that was better than either of them separately. Wilkins [46] credits Kuropatenko [49] with giving a physical basis for the combination of linear and quadratic viscosities. Kuropatenko noted the similarity between the analytical solutions for the pressure jump across an ideal gas and an elastic solid. For shocks where the velocity jump is much greater than the speed of sound, the pressure jump is quadratic.

$$\begin{aligned} P_2 - P_1 &= \frac{\gamma+1}{2}\rho_1(\triangle u)^2 \qquad \text{for an ideal gas} \\ P_2 - P_1 &= \rho_1(\triangle u)^2 \qquad \text{for an elastic fluid} \end{aligned} \tag{2.8.3.1}$$

The viscosity, q, is defined to be the pressure jump across the shock, $P_2 - P_1$. Comparing analytical solutions to Eqn. (2.8.3.1), the theoretical values for c_0 are $(\gamma+1)/2$ for an ideal gas and 1.0 for an elastic fluid.

A linear viscosity is obtained by considering the limits of the analytical solutions when the velocity jump is much smaller than the sound speed.

$$q = P_2 - P_1 = \rho_1 a_1 |\triangle u| \qquad \text{for both materials} \tag{2.8.3.2}$$

Combining Eq. (2.8.3.1) and Eq. (2.8.3.2) into a single term, and allowing two material dependent coefficients, gives the quadratic plus linear viscosity that is in common use today. The particular form chosen represents a doubly asymptotic expansion of the shock Hugoniot. In the remainder of this paper, this form will be referred to as the standard one-dimensional viscosity.

$$q = c_0\rho(\triangle u)^2 + c_L a\rho|\triangle u| \tag{2.8.3.3}$$

Typical values used in calculations for c_0 and c_L are 1.5 and 0.06 respectively. Note that the linear term is small in comparison to its theoretical value. The reason for a small value of c_L is discussed in the next section.

2.8.4 The Shock Viscosity as an Approximate Riemann Solver. Christensen closes the loop by interpreting the artificial viscosity as an approximate Riemann solver [50]. The derivation presented here has been superseded by a more sophisticated one that gives sharper results [51], and it will be published in a forthcoming paper by Christensen. Three assumptions are made here, two of which are found in Dukowicz's work: 1) a rarefaction can be approximated by the shock Hugoniot, referred to as the “two-shock approximation,” and 2) the shock velocity, U_s, is a linear function of the particle velocity, U_p. The second assumption is an accurate approximation for a wide variety of materials. The states to the right and left of the Riemann problem are denoted by subscripts + and −, the values at the contact discontinuity by a superscript ∗, and the characteristic velocity is C. Using the two shock approximation, the solution to the Riemann problem satisfies Eq. (2.8.4.1).

$$(u^* - u_\pm) \pm \frac{P^* - P_\pm}{(\rho C)_\pm} = 0 \tag{2.8.4.1}$$

Letting C_- and C_+ be equal (this third assumption makes this a symmetric rather than an upwind approximation), the contact pressure can be calculated.

$$P^* = \frac{\rho_- P_+ + \rho_+ P_-}{\rho_- + \rho_+} + \frac{\rho_+ \rho_-}{\rho_- + \rho_+} C(u_+ - u_-) \tag{2.8.4.2}$$

Assuming that the linear $U_s - U_p$ relationship holds, then the characteristic velocity is a linear function of the velocity jump, and the contact pressure is a quadratic function of the velocity jump.

$$C = a + A|u_+ - u_-| \tag{2.8.4.3}$$

$$P^* = \frac{\rho_- P_+ + \rho_+ P_-}{\rho_- + \rho_+} + \frac{\rho_+ \rho_-}{\rho_- + \rho_+} \left(a|u_+ - u_-| + A(u_+ - u_-)^2\right) \tag{2.8.4.4}$$

The terms involving the densities can be thought of as averages.

$$\begin{aligned} \bar{P} &= \frac{\rho_- P_+ + \rho_+ P_-}{\rho_- + \rho_+} \\ \bar{\rho} &= \frac{\rho_+ \rho_-}{\rho_- + \rho_+} \end{aligned} \tag{2.8.4.5}$$

Identifying terms between Eqn. (2.8.4.5) and standard formulation, $P^* - \bar{P}$ is the shock viscosity, $\bar{P}$ is the element average pressure, and $\bar{\rho}$ is the element density.

2.8.5 The Christensen flux-limited viscosity. The one-dimensional flux-limited viscosity developed by Christensen [50] was originally developed for an Eulerian hydrocode using a Lagrangian-Eulerian operator split on a rectangular mesh with edge-centered velocities. The one-dimensional viscosity is applied in the y and z directions independently. In the viscosity literature, such an approach is referred to as a “split viscosity” in reference to the spatial operator split.

The major difference between the standard viscosity and the new viscosity is the approximation of the jump in the velocity at the element centroid. The standard approximation is just the difference between the two nodal velocities defining element i; see Fig. 8 for a graphical interpretation.

$$\triangle u = u_{i+1} - u_i \tag{2.8.5.1}$$

The jump in the velocity can be written in terms of the strain rate, a form which proves convenient in two and three dimensions. In one dimension, $\triangle X$ is the length of the element, while in two or three dimensions it represents a characteristic length of the element.

$$\triangle u = \triangle X \frac{\partial u}{\partial x} \tag{2.8.5.2}$$

When the standard approximation is used, the typical values for c_0 and c_L for an ideal gas with a γ of 5/3 are 1.5 and .06, respectively. The theoretical values, based on Kuropatenko's q are 1.3 and 1.0, respectively. If the theoretical value of c_L is used in a calculation, the shock is smeared out over more elements than is necessary. This shortcoming can be traced back to the poor quality of the approximation in Eq. (2.8.5.1).

A better approximation to the velocity jump can be developed by using a Taylor expansion of the velocities at the nodes and keeping only the first order terms; see Fig. 8. The vertical bar with the subscript, $|_i$, indicates that the derivative is evaluated at node i.

$$\triangle u = (u_{i+1} - \frac{\triangle X}{2} \frac{\partial u}{\partial x}\bigg|_{i+1}) - (u_i + \frac{\triangle X}{2} \frac{\partial u}{\partial x}\bigg|_i) \tag{2.8.5.3}$$

The difficulty with using Eq. (2.8.5.3) is in evaluating the spatial derivative of the velocity at the nodes, $\partial u/\partial x|_i$. The lower order elements typically used in explicit calculations are only C_0 continuous and special approximations of the derivative must be used. Christensen adopted the methodology developed by van Leer [52] for higher order, monotonic advection methods. The term "flux-limited" which Christensen gave his new viscosity is derived from the flux limiting procedure that van Leer created in order to impose monotonicity on his advection algorithms.

Van Leer [52] developed a general class of higher order advection methods, and with them, a general class of derivative approximations. Only the particular second order approximation used in the viscosity will be discussed. Christensen uses a more sophisticated third order approximation, but it is appropriate only for meshes that are logically regular, i.e., two-dimensional meshes with a regular checkerboard pattern. In one dimension, three consecutive nodes are considered: $i-1$, i, and $i+1$. Two first order slope approximations, S_L and S_R, are available at node i by using backward and forward differencing.

$$\begin{aligned} S_L &= \frac{J(u_i - u_{i-1})}{x_i - x_{i-1}} \\ S_R &= \frac{J(u_{i+1} - u_i)}{x_{i+1} - x_i} \end{aligned} \tag{2.8.5.4}$$

A second order approximation, S_C, of the slope may be developed by passing a parabola through the three points and evaluating the slope at node i.

$$S_C = \frac{(\frac{x_{i+1}-x_i}{x_i-x_{i-1}})(u_i - u_{i-1}) + (\frac{x_i-x_{i-1}}{x_{i+1}-x_i})(u_{i+1} - u_i)}{x_{i+1} - x_{i-1}} \tag{2.8.5.5}$$

Monotonicity determines which of the three slopes is used at a given node. Defining u_{max} and u_{min} as the maximum and minimum of the velocities at nodes $i-1$, i, and $i+1$, the final slope approximation, S_i, is determined by demanding that range of Taylor expansion of u about node i lie between u_{max} and u_{min} on the domain $\{x^*|(x_{i-1}-x_i)/J \leq x^* - x_i \leq (x_{i+1} - x_i)/J\}$. The value of J for advection is 2.0. A more diffusive result is obtained if J is set to 1.0, which corresponds to limiting the second order derivative approximation by the left and right first order approximations. A value of 0.0 for J reduces the formulation to the standard formulation, i.e., S_i is zero.

$$u_{min} \leq u_i + S_i x^* \leq u_{max} \qquad \forall\ x^* \tag{2.8.5.6}$$

Since the expansion is linear, only the endpoints of the domain need to be tested to see if a particular value of S_i satisfies Eq. (2.8.5.6). A particular choice that satisfies Eq. (2.8.5.6), which is used here, is given by Eq. (2.8.5.7).

$$S_i = \tfrac{1}{2}(\mathrm{sgn}(S_L) + \mathrm{sgn}(S_R))\mathrm{min}(|S_L|, |S_C|, |S_R|) \tag{2.8.5.7}$$

The behavior of this slope approximation is the crucial ingredient in the success of the new viscosity. When the solution is smooth, the slope is the second order approximation and the predicted jump is small. For a perfectly linear velocity profile, the predicted velocity jump is zero, in contrast to the jump of $u_{i+1} - u_i$ predicted by the standard approximation. The amount of dissipation introduced by the viscosity in a region undergoing a smooth compression is small. Near a sharp discontinuity, the limiting procedure comes into play, and the slopes on either side of the discontinuity approach zero. The result is a large jump in the velocity across the element, and a correspondingly large value for q.

Since Eq. (2.8.5.7) reduces the unwanted dissipation in the smooth part of the solution, the theoretical values for c_0 and c_L, determined by the Rankine-Hugoniot relations, can be used for strong shocks. For ideal gases, these values are $\frac{1}{2}(\gamma+1)$ and 1 for c_0 and c_L respectively. Weak shocks in elastic materials still require some reduction in the viscosity coefficients for optimum results. Another possibility is to use a still higher order expansion of the velocity field.

Noh proposed three test problems involving an ideal gas ($\gamma = 5/3$) in his article on shock errors [45]. The initial state of the gas is the same in all three problems: the internal energy is 0.0, the density is 1.0, and the gas has a uniform velocity of -1.0 in the radial direction. The only difference between the problems is their geometry. The first problem is planar ($n = 1$), the second is cylindrical ($n = 2$), and the third is spherical ($n = 3$). A shock wave starts at the origin and propagates with a velocity of 1/3 to the right, the internal energy behind the shock is 1/2, and the density behind the shock is 4^n. Ahead of the shocks, there is isentropic heating associated with the geometrical convergence in the cylindrical and spherical problems, see Fig. 9. Only the first and third problems will be used in this paper. The first problem is useful for determining the shock width and sharpness, and the third problem illustrates problems associated with shockless heating. The density is used to compare the accuracy of the different viscosity formulations.

In the remainder of this section on viscosities, the coefficients c_0 and c_L for the standard formulation are 1.5 and 0.06. For the Christensen viscosity, they are the theoretical values, 1.3 and 1.0, from Hugoniot's ideal gas solution.

A comparison of the standard viscosity and the flux-limited viscosity for the planar shock problem is shown in Fig. 10. Both give accurate answers for the density behind the shock, but it is clear that the new viscosity gives a much sharper shock. For the spherical shock problem, shown in Fig. 11, the results are much the same. Overheating is evident by the low density after the shock in Fig. 11 when the standard formulation is used and the viscosity is treated as a pressure in the energy equation.

2.9 TWO-DIMENSIONAL SHOCK VISCOSITIES

2.9.1 The Standard Extension. The simplest extension of the viscosity from one dimension to two is to consider the viscosity as a pressure term. To implement the viscosity in two dimensions, it is necessary to calculate a velocity jump across the shock in the element. This task is simplified by calculating the jump as the product of a characteristic element length and a strain rate normal to the shock front, where the normal direction is x_s.

$$\triangle u = \triangle X \frac{\partial u}{\partial x_s} \tag{2.9.1.1}$$

The simplest approximations, which are used in what will be referred to as the standard formulation in two dimensions, are given in Eq (2.9.1.2).

$$\begin{aligned} \triangle X &= \sqrt{A} \\ \frac{\partial u}{\partial x_s} &= \mathrm{tr}(\dot{\epsilon}) \end{aligned} \tag{2.9.1.2}$$

The viscosity term is treated as a pressure and the work is calculated in the same manner as for pressures. This results in shock overheating because the energy calculation includes the irreversible viscous work due to the nonzero hoop strain rate in axisymmetric problems despite the fact that the axisymmetric assumption prevents shocks in that direction.

2.9.2 The HEMP Viscosity. Wilkins and Sharp were aware of the problems associated with Eq. (2.9.1.2) and sought to remedy them by introducing more sophisticated approximations [46]. With reference to Fig. 12, the direction of the shock, $\vec{n}$, is calculated using the nodal accelerations from the last time step. In cylindrical coordinates, the coordinates y and z are replaced with r and z, but the equations below remain otherwise unchanged.

$$\begin{aligned} \ddot{y} &= \ddot{y}_1 + \ddot{y}_2 + \ddot{y}_3 + \ddot{y}_4 \\ \ddot{z} &= \ddot{z}_1 + \ddot{z}_2 + \ddot{z}_3 + \ddot{z}_4 \\ n_1 = \mathrm{cos}\alpha &= \frac{\ddot{y}}{\sqrt{\ddot{y}^2 + \ddot{z}^2}} \\ n_2 = \mathrm{sin}\alpha &= \frac{\ddot{z}}{\sqrt{\ddot{y}^2 + \ddot{z}^2}} \end{aligned} \tag{2.9.2.1}$$

The strain rate in the shock direction is $\vec{n}^t[\dot{\epsilon}]\vec{n}$.

$$\frac{\partial u}{\partial y_s} = \dot{\epsilon}_{yy}\mathrm{cos}^2\alpha + \dot{\epsilon}_{zz}\mathrm{sin}^2\alpha + 2\dot{\epsilon}_{yz}\mathrm{cos}\alpha\mathrm{sin}\alpha \tag{2.9.2.2}$$

A characteristic length, $\triangle X$, is calculated by dividing the element area by the average thickness of the element in the direction normal to the shock front, where the d_i are defined in Fig. 12.

$$\triangle X = \frac{2A}{d_1 + d_2 + d_3 + d_4} \tag{2.9.2.3}$$

This viscosity works well when the aspect ratio of the element is near one. There is less overheating with this viscosity with the standard extension because the viscosity is not turned on in elements that are undergoing an isentropic compression due to spherical or cylindrical convergence. When the element aspect ratio is large, small changes in the direction of the shock result in large changes in $\triangle X$, and the viscosity is unstable.

2.9.3 Tensor viscosities. To overcome the limitations associated with a scalar or bulk shock viscosities, tensor viscosities have been developed that incorporate the direction of the shock in their formulation. To simplify the discussion, two sets of basis vectors are introduced. The first set is the standard Euclidean vectors E_i. The second set is local to the element or zone. Corresponding to the ξ^1 or the k lines is the basis vector e_1, for ξ^2 or the ℓ lines, it is e_2. To be consistent with the tensor notation in this section, the Cartesian coordinates x_i are written with superscripts, x^i. In all other sections, the coordinates will retain their subscripts to avoid any confusion with exponentials.

$$\begin{aligned} e_1(k+1/2,\ell) &= (x^i_{(k+1,\ell)} - x^i_{(k,\ell)})E_i \\ e_2(k,\ell+1/2) &= (x^i_{(k,\ell+1)} - x^i_{(k,\ell)})E_i \\ e_1(k+1/2,\ell+1/2) &= \tfrac{1}{2}(e_1(k+1/2,\ell) + e_1(k+1/2,\ell+1)) \\ e_2(k+1/2,\ell+1/2) &= \tfrac{1}{2}(e_2(k,\ell+1/2) + e_2(k+1,\ell+1/2)) \end{aligned} \tag{2.9.3.1}$$

More compactly, the relationship between e_i and E_j is defined by a set of displacement dependent coefficients X^j_i.

$$e_i = X^j_i E_j \tag{2.9.3.2}$$

The formulation of a particular tensor viscosity is considered to have three parts: 1) the calculation of a velocity jump vector across the shock, 2) the construction of an appropriate Hugoniot function, $W(\triangle\vec{u})$, and 3) the assembly of the first two parts into a tensor form. In some cases, $W(\triangle\vec{u})$ is constructed to satisfy some set of desirable characteristics and it is not related to the Hugoniot function as it was discussed in previous sections.

For a mesh with edge-centered velocities, the one-dimensional tensor viscosity can be extended to two dimensions by evaluating the one-dimensional viscosity independently in the x^1 and x^2 directions. Noting that the E_i are orthogonal, the tensor viscosity in Eq. (2.9.3.3) is diagonal.

$$Q^{ij} E_i \otimes E_j = \sum_{k=1}^{2} q(\triangle u^k) E_k \otimes E_k \tag{2.9.3.3}$$

Three different tensor shock viscosities appear in the literature for TENSOR. The first viscosity [6] separates the response into volumetric and deviatoric contributions. The volumetric part is very similar to the simple two-dimensional extension discussed previously, denoted here as q^{s2}, while the deviatoric is a linear function of the deviatoric strain rate, $\dot{\epsilon}'_{ij}$, the density, and the characteristic element length, $\triangle X$.

$$Q^{ij} E_i \otimes E_j = q^{s2} E_k \otimes E_k + C\rho\triangle X \dot{\epsilon}'_{ij} E_i \otimes E_j \tag{2.9.3.4}$$

The second viscosity [54] is similar, but the magnitude of the velocity jump is defined as $\sqrt{(\triangle u^1)^2 + (\triangle u^2)^2}$, and $(\triangle u^i)^2$ is calculated directly as the square of the velocity jump in the direction normal to e_i.

The most recent viscosity in TENSOR [55] was motivated by the desire to include directional properties into the viscosity, to handle multiple shocks, and to make the viscosity invariant under rigid body rotation. For axisymmetric problems there is no shock in the θ direction and no terms are included in that direction so as to avoid shock overheating in spherically convergent problems. Exclusive of the calculation of the strain rates, there is no difference between the planar and axisymmetric shock viscosities.

Directionality is included in this formulation by working in the coordinate system associated with the directions of the principal strain rates, $\dot{\hat{\epsilon}}_i$, denoted here as $\hat{e}_i$. A characteristic length, $\triangle\hat{X}_i$, is calculated for each principal direction. The jump in the two principal directions, $\triangle\hat{u}^i$, includes a function that is intended to discriminate between shocks and steady flow situations. The coefficient, C_f, is set to a value near 1.0.

$$\begin{aligned} \triangle\hat{u}^i &= f(\dot{\hat{\epsilon}}_i)\dot{\hat{\epsilon}}_i \triangle\hat{X}^i \qquad \text{no sum on } i \\ f(\dot{\hat{\epsilon}}_i) &= \frac{2}{1 + |(\dot{\hat{\epsilon}}_1 - \dot{\hat{\epsilon}}_2)/C_f \frac{\partial V}{\partial t}|} \end{aligned} \tag{2.9.4.5}$$

The one-dimensional standard formulation, $q^{s1}(\triangle u)$, is applied independently to the two principal directions, resulting in a split viscosity. Shear viscosity terms appear when this split viscosity is rotated into the global coordinate system.

$$Q^{ij} E_i \otimes E_j = \sum_{k=1}^{2} q^{s1}(\triangle u^k)\hat{e}_k \otimes \hat{e}_k \tag{2.9.4.6}$$

In his paper [56], Schulz presents arguments in favor of a tensor viscosity, and derives an edge-centered tensor viscosity. While the edge viscosity developed by Schulz can be transformed into the global coordinate system, it is calculated in the mesh curvilinear coordinates. After performing "a fair amount of manipulation," the contributions of the viscosity tensor divergence can be written in terms of the curvilinear coordinates, and it is computationally never necessary to work with the viscosity tensor in the global coordinates. In Schulz's paper, $\mathbf{R}_k$, $\mathbf{R}_\ell$, $\bar{\mathbf{R}}_\ell$, and $-\bar{\mathbf{R}}_k$ correspond to e_1, e_2, e^1 and e^2 in this paper, and q_A and q_B correspond to q^1 and q^2.

In the curvilinear coordinate system of the mesh, the Schulz viscosity is a split formulation. The jacobian, J, by virtue of Schulz's choice of coordinates, is equal to $e_1 e^1$ and $e_2 e^2$.

$$\begin{aligned} Q^{ij} E_i \otimes E_j &= \frac{1}{J}\sum_{k=1}^{2} q^k e_k \otimes e^k \\ q^k &= 2\rho \left(\frac{\partial u^k}{\partial \xi^k}\right)_- \cdot \left| \frac{\partial}{\partial \xi^k}\left(\frac{\partial u^k}{\partial \xi^k}\right)_- \right| \\ \left(\frac{\partial u^k}{\partial \xi^k}\right)_- &= \min\left(0,\ \frac{\partial u^k}{\partial \xi^k}\right) \\ \frac{\partial u^k}{\partial \xi^k} &\equiv \frac{\partial \vec{u}}{\partial \xi^k} \cdot \frac{e^k}{|e^k|} \end{aligned} \tag{2.9.4.7}$$

The central difference approximation of $\partial \vec{u}/\partial \xi^k$ is simply $\triangle \vec{u}$ for the pair of nodes defining the edge, and $\partial u^k/\partial \xi^k$ is the physical component of $\triangle \vec{u}$ parallel to the edge. A length scale enters the viscosity through the fact that e_k is not a unit vector, but has a magnitude equal to the length of the appropriate element edge. By grouping the appropriate terms into the Hugoniot, the edge viscosity can be written in terms of the Hugoniot, the velocity jumps along the element edges and the curvilinear basis vectors.

$$\begin{aligned} Q^{ij} E_i \otimes E_j &= \frac{1}{J} \sum_{k=1}^{2} W(\vec{u}) \triangle u^k e_k \otimes \frac{e^k}{|e^k|} \\ W(\vec{u}) &= 2\rho \left| \frac{\partial}{\partial \xi^k} \left(\frac{\partial u^k}{\partial \xi^k} \right)_- \right| \end{aligned} \tag{2.9.4.8}$$

Schulz departed from the standard Hugoniot approximation to eliminate the shockless heating error. A uniform expansion or contraction is considered to be reversible, and the viscosity should not be turned on. In such a situation the velocity is proportional to the displacement in cartesian coordinates, resulting in the second derivative of the velocity with respect to the displacement being zero. When a uniform mesh is used, the second derivative of the velocity with respect to the logical mesh coordinates is also zero, and the Hugoniot in Eq. (2.9.4.8) is zero. For a nonuniform mesh, the Hugoniot will not be zero. This situation is discussed by Schulz.

To summarize, the Schulz viscosity is a split viscosity formulation in the logical mesh coordinates. The Hugoniot is defined in such a way as to eliminate shockless heating. Shear viscosities appear when the viscosity is transformed into the global cartesian coordinate system. In general, the viscosity tensor is not symmetric.

Margolin introduced a zone-centered viscosity based on the Schulz edge-centered viscosity. In order to improve the performance of the viscosity with elements having a large aspect ratio, he scales W.

$$Q^{ij} E_i \otimes E_j = \rho W(\vec{u}) \frac{\partial u^i}{\partial x^k} X^k_m X^j_m E_i \otimes E_j \tag{2.9.4.9}$$

$$W(\vec{u}) = \max \left[\frac{c_1 a + c_2 \left| \frac{\partial u^i}{\partial x^k} X^k_1 \right|}{|e_1|}, \frac{c_1 a + c_2 \left| \frac{\partial u^i}{\partial x^k} X^k_2 \right|}{|e_2|} \right] \tag{2.9.4.10}$$

Margolin tested both the nonsymmetric form and a symmetric form obtained by averaging the off diagonal terms. He found that the nonsymmetric form gave better solutions, but that special care had to be taken with the boundary conditions. The symmetric form was less sensitive to the boundary conditions, but it introduced spurious motion perpendicular to the shock through its shear terms.

A two-dimensional implementation of the Christensen viscosity was developed by Benson [42]. The velocity jumps are calculated in a one-dimensional manner along the element directions. Finite difference notation is used here for clarity since the finite element notation does not provide a convenient means of describing quantities centered along the element edges. The derivatives are evaluated using the monotonic algorithm of van Leer in the same manner as the Christensen viscosity.

$$\begin{aligned} \triangle u^1 &= \left(u^1 - \frac{\triangle\xi^1}{2}\frac{\partial u^1}{\partial\xi^1}\right)\Big|_{(k+1,\ell+1/2)} - \left(u^1 + \frac{\triangle\xi^1}{2}\frac{\partial u^1}{\partial\xi^1}\right)\Big|_{(k,\ell+1/2)} \\ \triangle u^2 &= \left(u^2 - \frac{\triangle\xi^2}{2}\frac{\partial u^2}{\partial\xi^2}\right)\Big|_{(k+1/2,\ell+1)} - \left(u^2 + \frac{\triangle\xi^2}{2}\frac{\partial u^2}{\partial\xi^2}\right)\Big|_{(k+1/2,\ell)} \end{aligned} \tag{2.9.4.11}$$

The two-dimensional velocity jump is readily expressed in the Cartesian coordinate system.

$$\begin{aligned} \triangle\vec{u} = \quad & (\triangle u^1 x^1{,_{\xi^1}} + \triangle u^2 x^1{,_{\xi^2}})E_1 \\ & + (\triangle u^1 x^2{,_{\xi^1}} + \triangle u^2 x^2{,_{\xi^2}})E_2 \end{aligned} \tag{2.9.4.12}$$

For clarity, the components of the jump in the cartesian coordinate system will be $\triangle u^1_E$ and $\triangle u^2_E$.

The shock directionality is introduced in the same manner as in the TENSOR viscosity. After calculating the shock direction from the strain rates, two one-dimensional shock viscosities are calculated, where quantities in the principal directions are given a hat: $(\hat{\cdot})$.

$$\begin{aligned} \hat{Q}^{11} &= -c_0\rho(\triangle\hat{u}_1)^2 - c_L a\rho|\triangle\hat{u}_1| \\ \hat{Q}^{22} &= -c_0\rho(\triangle\hat{u}_2)^2 - c_L a\rho|\triangle\hat{u}_2| \\ \hat{Q}^{12} &= \hat{Q}^{21} = 0 \end{aligned} \tag{2.9.4.13}$$

Benson demonstrated that shear viscosity terms must be introduced into constant stress quadrilateral elements to eliminate spurious element distortion after the shock [42]. The particular form of the shear viscosity that was chosen in his formulation is based on a simple linear differential equation which has a solution that damps out the spurious shear strain rates.

$$\frac{\partial}{\partial t}\left(\frac{\partial\hat{u}^i}{\partial\hat{x}^j}\right) + \lambda\frac{\partial\hat{u}^i}{\partial\hat{x}^j} = 0 \qquad \lambda > 0 \tag{2.9.4.14}$$

The final expressions for the shear viscosity terms are unsymmetric.

$$\hat{Q}^{12} = \alpha\frac{(\hat{x}^1_{13}\hat{x}^2_{13} + \hat{x}^1_{24}\hat{x}^2_{24})\hat{Q}^{11} + \rho(2A^2\lambda - \dot{A})[\hat{x}^1_{13}\hat{u}^1_{24} - \hat{x}^1_{24}\hat{u}^1_{13}]}{(\hat{x}^1_{13})^2 + (\hat{x}^1_{24})^2} \tag{2.9.4.15}$$

$$\hat{Q}^{21} = \alpha \frac{(\hat{x}^1_{13}\hat{x}^2_{13} + \hat{x}^1_{24}\hat{x}^2_{24})\hat{Q}^{22} + \rho(2A^2\lambda - \dot{A})[\hat{x}^2_{24}\hat{u}^2_{13} - \hat{x}^2_{13}\hat{u}^2_{24}]}{(\hat{x}^2_{13})^2 + (\hat{x}^2_{24})^2} \tag{2.9.4.16}$$

The energy calculation is modified slightly because of the lack of symmetry associated with the tensor viscosity. Instead of using the strain rate, the velocity gradient, which is unsymmetric like the viscosity, is used in the energy equation.

$$\dot{E}_Q = Q^{ij}\frac{\partial u^i}{\partial x^j}V \tag{2.9.4.17}$$

No changes are necessary for using this viscosity formulation in axisymmetric problems. The governing equations in axisymmetry have the additional terms $(\sigma^{rr} - \sigma^{\theta\theta})/r$ in the r equation and σ^{rz}/r in the z equation. These additional terms only contribute a rigid body acceleration to the entire element when the area-weighted Petrov-Galerkin finite element method is used. A consistent use of the tensor viscosity would suggest that similar viscous terms should be added for axisymmetric problems. The purpose of adding the viscosity is to resist the compression of the element and to add the proper amount of thermodynamic irreversibility. Adding the axisymmetric viscous terms to the equations would not achieve either end, but they would result in large, spurious rigid body accelerations in the region of a shock. Neglecting to add them does not affect the internal energy calculation because the strain rates are independent of the rigid body motion.

2.10 HOURGLASS CONTROL

One consequence of using constant stress quadrilateral elements is the appearance of zero energy modes. These modes are also called “hourglass” and “keystone” modes after their shapes. The modes are not resisted by the stresses within the element, and they can grow without bound. Unless some mechanism is included to control them, they will ultimately turn the elements inside out.

The existence and nature of the hourglass modes can be deduced from very primitive considerations. A single element has eight degrees of freedom, two for each node. The three rigid body modes, two translational and one rotational, must not be resisted by the stresses. This leaves five independent modes of deformation to be resisted, but in two dimensions there are only three stresses ($\sigma_{\theta\theta}$ does not resist the hourglass modes in axisymmetry) to resist them. Two of those modes, the hourglass modes, are therefore not resisted by the element.

Consider an initially square element, and move the lower two nodes towards each other by an amount δ, and the upper two nodes δ outwards. The two material lines associated with ξ^i through the element centroid remain undistorted and the strain at the centroid is zero. No resisting stress will be developed in response to this distortion. This mode of distortion is the x_1 hourglass mode, and the x_2 mode is simply the x_1 mode rotated through a quarter turn. Note that the centroid of the element is the only location within the element where the strain is zero. For the x_1 mode, the only strain component is ϵ_{11} and it is a linear function of x_2. In simple beam bending theory [57], the strain has the same linear distribution through the thickness. The hourglass modes therefore can be thought of as bending modes for the case of rectangular elements. Since bending is often an important part of the solution, we want to control the unbounded growth of the hourglass modes, but not to suppress them so much that the structural response is overly stiff.

The existence of hourglass modes has been known since the earliest two-dimensional finite difference calculations. The usual approach to controlling hourglass modes is to add viscous terms to the momentum equations that oppose the modes. Despite the large body of literature on the subject, the actual form of the hourglass viscosity is not critical for most problems. One exception is the method by Flanagan and Belytschko [58], which uses a stiffness instead of a viscosity. Belytschko [59] has shown that the proper choice for an hourglass stiffness results in superconvergent behavior for small strain, linearly elastic problems. While the stiffness form works better than the viscosity for many structural problems [58], its usefulness is limited in axisymmetric hydrodynamics problems [34]. In the hydrocode literature other formulations have been presented by many others, including Wilkins [7], Hallquist [18], Margolin [60], and Maenchen and Sack [6].

The hourglass control has two parts: 1) defining the mode shapes and 2) defining the resisting force. Neither part is uniquely defined in the literature, and depending on the problems, different forms are appropriate.

2.10.1 Defining the hourglass modes. The hourglass velocity is the velocity field minus the velocity field associated with the rigid body and straining modes [6], [7]. Since there are eight degrees of freedom in a quadrilateral element, the velocity field can be expressed in terms of eight modes. The choice of the eight modes is not unique so long as they are linearly independent. From inspection, one choice is the two translational modes, the rigid rotation mode, the pure shear mode, the two stretching modes in the y and z directions, and the two hourglass modes. The shear and rotation modes include both y and z velocity components, while the other modes include only one velocity component (aside from the hourglass modes, which are to be determined). To simplify the analysis, the rigid rotation and the pure shear modes are replaced by the y and z shear modes. The pure shear mode is one half of the z and y shear modes, while the rigid rotation is half the difference between the two.

Letting the translational, shear, and stretching modes in the ith direction be denoted as T^i, τ^i, and σ^i respectively, their shapes can be described by vectors using the same ordering as the velocity components for Eq. (2.4.2.10).

$$\begin{aligned}
T^y &= \tfrac{1}{4}[\ 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0\ ] \\
T^z &= \tfrac{1}{4}[\ 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1\ ] \\
\tau^y &= \tfrac{1}{2A}[-y_{24} \quad 0 \quad -y_{31} \quad 0 \quad y_{24} \quad 0 \quad y_{31} \quad 0\ ] \\
\tau^z &= \tfrac{1}{2A}[\ 0 \quad z_{24} \quad 0 \quad z_{31} \quad 0 \quad -z_{24} \quad 0 \quad -z_{31}] \\
\sigma^y &= \tfrac{1}{2A}[\ z_{24} \quad 0 \quad z_{31} \quad 0 \quad -z_{24} \quad 0 \quad -z_{31} \quad 0\ ] \\
\sigma^z &= \tfrac{1}{2A}[\ 0 \quad -y_{24} \quad 0 \quad -y_{31} \quad 0 \quad y_{24} \quad 0 \quad y_{31}]
\end{aligned} \qquad (2.10.1.1)$$

These modes are used to calculate the magnitude of the deformation modes and do not constitute velocity profiles. For instance, the velocity profile for shear in the y direction is proportional to z, but the τ^y mode defined above clearly does not have this form (e.g., nodes 1 and 3 have the same magnitudes). The dot product of the nodal velocities with τ^y gives the magnitude of the shear component in the y direction in the specified bilinear velocity field.

Perhaps the best way to think of these modes is as covectors. Their dot product with the nodal velocities gives the magnitude of the contribution of their associated deformation pattern to the velocity field. By construction, the resulting velocity field is linear.

$$\{u_{\alpha i}\}^{linear} = (T^y\cdot u)T_y + (T^z\cdot u)T_z + (\tau^y\cdot u)\tau_y + (\tau^z\cdot u)\tau_z + (\sigma^y\cdot u)\sigma_y + (\sigma^z\cdot u)\sigma_z \quad (2.10.1.2)$$

The associated vectors are fairly simple. The centroid location is (y_o, z_o), and it is calculated by taking the dot product of the nodal coordinates with $T^y/4$ and $T^z/4$.

$$
\begin{aligned}
T_y &= [\;\; 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \;\;] \\
T_z &= [\;\; 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \;\;] \\
\tau_y &= [\, z_1 - z_o \quad 0 \quad z_2 - z_o \quad 0 \quad z_3 - z_o \quad 0 \quad z_4 - z_o \quad 0 \;\;] \\
\tau_z &= [\;\; 0 \quad y_1 - y_o \quad 0 \quad y_2 - y_o \quad 0 \quad y_3 - y_o \quad 0 \quad y_1 - y_o \,] \\
\sigma_y &= [\, y_1 - y_o \quad 0 \quad y_2 - y_o \quad 0 \quad y_3 - y_o \quad 0 \quad y_4 - y_o \quad 0 \;\;] \\
\sigma_z &= [\;\; 0 \quad z_1 - z_o \quad 0 \quad z_2 - z_o \quad 0 \quad z_3 - z_o \quad 0 \quad z_1 - z_o \,]
\end{aligned}
\tag{2.10.1.3}
$$

Six of the eight basis vectors and covectors are known and it remains to complete the spaces by adding an additional pair of vectors and covectors. Note that neither the set of vectors or covectors are mutually orthogonal. The definitions of the remaining vectors are not unique.

By inspection, two hourglass vectors, Γ_i, are found to be orthogonal to the six covectors. Any velocity vector that is a multiple of these vectors will produce a zero strain rate. The spaces are completed by calculating Γ^i.

$$
\begin{aligned}
\Gamma_y &= [+1 \quad 0 \quad -1 \quad 0 \quad +1 \quad 0 \quad -1 \quad 0] \\
\Gamma_z &= [\;\; 0 \quad +1 \quad 0 \quad -1 \quad 0 \quad +1 \quad 0 \quad -1]
\end{aligned}
\tag{2.10.1.4}
$$

Flanagan and Belytschko chose the Γ_i to complete the space because the Γ_i represent the difference between the interpolated bilinear velocity field and the linear velocity field defined by the six vectors. The covectors are calculated by Gram-Schmidt orthogonalization using the Γ_i as the initial approximation to Γ^i. By noting that the Γ_i are already orthogonal to rigid body modes, only the orthogonalization with respect to the straining modes needs to be performed. Since the two directions are considered independently, there are only four nonzero coefficients in each Γ^i. The two are related by a simple interchange between their y and z components, and therefore only one of the actual orthogonalization operations needs to be performed.

$$
\begin{aligned}
\Gamma^y &= \Gamma_y - (\tau_y \cdot \Gamma_y)\tau^y - (\sigma_y \cdot \Gamma_y)\sigma^y \\
\Gamma^z &= \Gamma_z - (\tau_z \cdot \Gamma_z)\tau^z - (\sigma_z \cdot \Gamma_z)\sigma^z
\end{aligned}
\tag{2.10.1.5}
$$

The orthogonalization can also be written in terms of the B matrix and the nodal coordinates [58]. An explicit expression for the hourglass covectors is given by Flanagan and Belytschko [58] and Taylor and Flanagan [34]. Since the mode in the y direction is a function of only the y velocities, and the z mode, the z velocities, only the four nonzero components are given.

$$\Gamma = \frac{1}{A}\left\{\begin{array}{l} y_2(z_3 - z_4) + y_3(z_4 - z_2) + y_4(z_2 - z_3) \\ y_3(z_1 - z_4) + y_4(z_3 - z_1) + y_1(z_4 - z_3) \\ y_4(z_1 - z_2) + y_1(z_2 - z_4) + y_2(z_4 - z_1) \\ y_1(z_3 - z_2) + y_2(z_1 - z_3) + y_3(z_2 - z_1) \end{array}\right\} \tag{2.10.1.6}$$

Kosloff and Frazier [61] chose to make the Γ_i orthogonal with respect to the rigid body and straining vectors. Again considering each coordinate direction independently, this gives three orthogonality conditions on a vector with four nonzero components, a_i. Kosloff and Frazier arbitrarily choose the last component, a_4, to be 1.0, solve the resulting three equations, and normalize the final vector.

$$\begin{aligned} a_1 + a_2 + a_3 &= -1 \\ a_1 y_1 + a_2 y_2 + a_3 y_3 &= -y_4 \\ a_1 x_1 + a_2 x_2 + a_3 x_3 &= -x_4 \end{aligned} \tag{2.10.1.7}$$

$$a'_i = \frac{2a_i}{\sqrt{a_1^2 + a_2^2 + a_3^2 + a_4^2}} \tag{2.10.1.8}$$

In both cases, when the elements are rectangular, the vector and covectors are equal to within a scale factor. In the finite difference community [2], [7], [60], and DYNA2D [18], the hourglass vectors and covectors are taken to be the Γ_i defined in Eq. (2.10.2.4).

2.10.2 Defining the resisting forces. As in the definition of the modes, there are a variety of ways to remove the hourglass modes. Viscous or elastic forces can be added to the momentum equation [2], [6], [7], [18], [34] or the mode shapes can be used as stencils to filter out the hourglass velocities [60].

The simplest approach, and the one that is the most popular in Lagrangian hydrocodes, is to add viscous damping to the momentum equation. The magnitude of the hourglass velocity, q_i, is calculated by dotting the velocities with the Γ^i.

$$q_i = \Gamma^i_{\alpha k} u_{\alpha k} \tag{2.10.2.1}$$

Defining an hourglass force, Q^i, that is conjugate to the the hourglass velocity results in an expression for the nodal hourglass forces. Kosloff and Frazier [61] are concerned only with small deformation elasticity, and instead of using the velocity, they use the displacements.

$$
\begin{aligned}
\dot{W}^i_{hg} &= Q^i q_i = F^i_{\alpha k} u_{\alpha k} \quad \text{no sum on } i \\
&= Q^i \Gamma^i_{\alpha k} u_{\alpha k} \qquad (2.10.2.2) \\
F^i_{\alpha k} &= Q^i \Gamma^i_{\alpha k}
\end{aligned}
$$

The remaining difficulty is to define the magnitude of the conjugate hourglass force. Kosloff and Frazier [61] choose a stiffness that will recover the fully integrated element stiffness for small deformation elasticity. The stiffness coefficient E is Young's modulus for plane stress and $4\mu(\lambda+\mu)/(\lambda+2\mu)$ for plane strain.

$$
Q^i = \frac{E}{12} A q_i \qquad (2.10.2.3)
$$

Flanagan and Belytschko [58] define damping and stiffness rates. The stiffness form is the hourglass analog of a hypoelastic material. The element thickness is t, and the damping and stiffness parameters are ϵ and κ respectively.

$$
\begin{aligned}
Q^i &= \epsilon t \sqrt{\frac{\rho(\lambda+2\mu) B : B}{2}}\, \dot{q}_i \quad \text{damping} \\
\dot{Q}^i &= \kappa t \frac{\lambda+2\mu}{2A} B : B \dot{q}_i \quad \text{stiffness rate}
\end{aligned} \qquad (2.10.2.4)
$$

The viscous coefficients used in DYNA2D [18] and PISCES [2] are similar. Note that the values of ϵ in the two forms are simply coefficients and do not have the same values.

$$
\begin{aligned}
Q^i &= \tfrac{1}{4} \epsilon \sqrt{A} \rho \dot{q}_i \quad \text{DYNA2D} \\
Q^i &= \left(\frac{\epsilon \rho A}{\triangle t}\right) \dot{q}_i \quad \text{PISCES}
\end{aligned} \qquad (2.10.2.5)
$$

Margolin filters the velocity element by element [60]. Rather than subtracting all the hourglassing velocity, only a fraction, β, of it is removed each time step. This number can be thought of as the ratio of the time step size to a relaxation time. Each node on a logically regular mesh is filtered four times, once for each surrounding element, and therefore β must be less than $\frac{1}{4}$ to prevent overcorrecting the modes.

$$
u = u - \beta[(\Gamma^y \cdot u)\Gamma_y + (\Gamma^z \cdot u)\Gamma_z] \qquad (2.10.2.6)
$$

2.10.3 Nonlinear hourglass controls. The hourglass controls discussed in the previous sections use a constant set of coefficients that are heuristic functions of the elastic properties of the material. These methods were designed to suppress the hourglass modes without degrading the accuracy of the element. Belytschko and Bindeman [169] have recently proposed a nonlinear assumed strain stabilization algorithm for plane strain with the coefficients based on the tangent moduli at the current stress state.

Working in the corotational frame, they construct an assumed strain by using an augmented B matrix, $\hat{B}$, which can be shown to satisfy the orthogonality conditions required by the Hu-Washizu principle [57],[170]. The constitutive model is evaluated using the usual strain rate, Bu, and the hourglass stress rates are $C[\hat{B} - B]u$, where C is the matrix of the current material moduli. By taking advantage of the orthogonality properties of the derivatives of the bilinear term in the shape functions, the calculation of the hourglass force rates can be simplified to the product of a stabilization matrix and the nodal velocities.

2.10.4 Comparing the hourglass controls. The differences between the choices of hourglass base vectors do not seem to be critical based on experiences with DYNA2D [18]. For linearly elastic problems, however, it is possible to choose the hourglass stiffness with the Flanagan and Belytschko vectors so that the fully integrated stiffness is exactly recovered [59]. In most cases, both the damping and stiffness hourglass forces work equally well, but it is possible to construct problems where either fail [58], [34]. The stiffness form apparently doesn't work well for axisymmetric problems, and Flanagan and Taylor [34] state that the instability is due to nonstrain terms in the Petrov-Galerkin formulation. The new formulation by Belytschko and Bindeman performs only moderately better than the formulation by Flanagan and Belytschko for elastic-plastic problems, but it is vastly superior for linear problems [169].

2.11 RECTANGULAR MESH FORMULATIONS WITH EDGE-CENTERED VELOCITIES

In the interests of efficiency, many Eulerian hydrocodes use a mesh of rectangular elements. The velocity is centered on the cell edges, and only the normal velocity component is known. This choice gives better results than node-centered velocities for some problems where the pressure is much larger than the deviatoric stress because it can optimally resolve a shear flow [50]. Rows of adjacent cells can move in opposite directions without any coupling because of the velocity centering. The expressions for the shear strain rates, however, are not as obvious for edge-centered velocities as for node centered velocities.

Finite difference notation is used in this section because it is the most natural notation for describing how different quantities are centered. The additional assumption is made that the mesh is rectangular and aligned with the coordinate axis. Formulations for general orthogonal coordinate systems have been proposed, but the necessary introduction of the metric tensor complicates the formulations without giving any deeper insight into the difference scheme.

Most of the algorithms associated with the rectangular mesh are identical to the general ones described above. From the conjugate relationship between the forces and the velocities, and the stresses and the strain rates, it should only be necessary to characterize the B matrix to completely define the edge-centered algorithm, but as will be shown, the forces and the velocities are not exactly conjugate in many finite difference algorithms. This means that there is an error in the work terms, but it is usually small in comparison to the other errors in the energy calculations.

2.11.1 The volume calculation. The initial volume of a cell is simply calculated from the coordinates of the edges.

$$\begin{aligned} V^n(i+\tfrac{1}{2}, j+\tfrac{1}{2}) &= (y(i+1)-y(i))(z(j+1)-z(j)) \quad \text{planar} \\ &= (r^2(i+1)-r^2(i))(z(j+1)-z(j)) \quad \text{axisymmetric} \end{aligned} \tag{2.11.1.1}$$

Since logical segments of the mesh lines move independently, the mesh lines are not continuous after a time step and the volumes of the cells are not defined unambiguously. The simplest and almost universally used method to calculate the final volume is to add the differential volume changes associated with the motion of the edges. For clarity, the logical coordinates of the mesh are enclosed in parentheses and separated by commas. The commas within the parentheses do not indicate partial differentiation.

$$\begin{aligned} V^{n+1}_{(i+1/2,j+1/2)} = V^{n}_{(i+1/2,j+1/2)} \quad &\text{planar} \\ +\triangle t\big[(y^n_{(i+1,j+1/2)} - y^n_{(i,j+1/2)})&(\dot{z}^{n+1/2}_{(i+1/2,j+1)} - \dot{z}^{n+1/2}_{(i+1/2,j)}) \\ +(\dot{y}^{n+1/2}_{(i+1,j+1/2)} - \dot{y}^{n+1/2}_{(i,j+1/2)})&(z^n_{(i+1/2,j+1)} - z^n_{(i+1/2,j)})\big] \\ V^{n+1}_{(i+1/2,j+1/2)} = V^{n}_{(i+1/2,j+1/2)} \quad &\text{axisymmetric} \\ +\triangle t\big[((r^n_{(i+1,j+1/2)})^2 - (r^n_{(i,j+1/2)})^2)&(\dot{z}^{n+1/2}_{(i+1/2,j+1)} - \dot{z}^{n+1/2}_{(i+1/2,j)}) \\ +2(\dot{r}^{n+1/2}_{(i+1,j+1/2)}\overline{r}_{(i+1)} - \dot{r}^{n+1/2}_{(i,j+1/2)}\overline{r}_{(i)})&(z^n_{(i+1/2,j+1)} - z^n_{(i+1/2,j)})\big] \\ \overline{r}_{(i)} = \tfrac{1}{2}(r^n_{(i,j+1/2)} + r^{n+1}_{(i,j+1/2)}) &= r^n_{(i,j+1/2)} + \tfrac{1}{2}\triangle t\dot{r}^{n+1/2}_{(i,j+1/2)} \end{aligned} \tag{2.11.1.2}$$

2.11.2 Defining the normal and hoop strain rates. The calculation of $\dot{\epsilon}_{yy(i+1/2,j+1/2)}$ (or $\dot{\epsilon}_{rr(i+1/2,j+1/2)}$) and $\dot{\epsilon}_{zz(i+1/2,j+1/2)}$ is straight forward.

$$
\begin{aligned}
\dot\epsilon^{n+1/2}_{yy(i+1/2,j+h2)} &= \frac{\dot y^{n+1/2}_{(i+1,j+1/2)} - \dot y^{n+1/2}_{(i,j+1/2)}}{y^{n+1/2}_{(i+1,j+1/2)} - y^{n+1/2}_{(i,j+1/2)}} \\
\dot\epsilon^{n+1/2}_{zz(i+1/2,j+h2)} &= \frac{\dot z^{n+1/2}_{(i+1,j+1/2)} - \dot z^{n+1/2}_{(i,j+1/2)}}{z^{n+1/2}_{(i+1,j+1/2)} - z^{n+1/2}_{(i,j+1/2)}}
\end{aligned}
\tag{2.11.2.1}
$$

The hoop strain rate is calculated by subtracting the strain rates in the r and z directions from the volumetric strain rate. Away from the z axis, the hoop strain rate simplifies to $\frac{1}{2}(\dot r^{n+1/2}_{(i+1,j+1/2)} + \dot r^{n+1/2}_{(i,j+1/2)})/\tilde r_{(i+1/2,j+1/2)}$, the direct analog to the finite element expression in Eq. (2.4.2.1).

$$
\begin{aligned}
\dot\epsilon^{n+1/2}_{\theta\theta(i+1/2,j+1/2)} &= \frac{\dot V^{n+1/2}_{(i+1/2,j+1/2)}}{V^{n+1/2}_{(i+1/2,j+1/2)}} - \dot\epsilon^{n+1/2}_{rr(i+1/2,j+1/2)} - \dot\epsilon^{n+1/2}_{zz(i+1/2,j+1/2)} \\
&= \frac{C_{(i+1,j+1/2)}\dot r^{n+1/2}_{(i+1,j+1/2)} + C_{(i,j+1/2)}\dot r^{n+1/2}_{(i,j+1/2)}}{\tilde r_{(i+1/2,j+1/2)}} \\
\tilde r_{(i+1/2,j+1/2)} &= \tfrac{1}{2}(r_{(i+1,j+1/2)} + r_{(i,j+1/2)}) \\
C_{(i+1,j+1/2)} &= \left(\frac{\overline r_{(i+1,j+1/2)} - \tilde r_{(i+1/2,j+1/2)}}{r^{n+1/2}_{(i+1,j+1/2)} - r^{n+1/2}_{(i,j+1/2)}}\right) \\
C_{(i,j+1/2)} &= \left(\frac{\tilde r_{(i+1/2,j+1/2)} - \overline r_{(i,j+1/2)}}{r^{n+1/2}_{(i+1,j+1/2)} - r^{n+1/2}_{(i,j+1/2)}}\right)
\end{aligned}
\tag{2.11.2.2}
$$

2.11.3 The shear strain rates. The calculation of the shear strain rates in a formulation with edge-centered velocities is complicated by the fact that some of the materials in the problem may not have any strength. In order to illustrate these difficulties, the algorithm used in CSQ [62] is presented in detail. While each edge-centered finite difference code has its own unique formulaton, the CSQ algorithm is fairly representative, and most of the differences between different codes are associated with the treatment of materials that lack shear strength. To simplify the presentation, only the calculation of $\partial\dot y/\partial z$ is discussed. The calculation of $\partial\dot z/\partial y$ proceeds in the same manner, and the shear strain rate is the average of the two terms.

Since the edges defining zone $(i+1/2, j+1/2)$ only have information on the velocities normal to the edges, the shear strain rates must include velocity information from other edges. To demonstrate the problems associated with naively differencing the velocity field, the problem illustrated in Fig. 13 is considered. The central difference approximation for $\partial\dot{z}/\partial y$ in zone $(i+1/2, j+1/2)$ is readily expressed in terms of the velocities at edges $(i, j+1/2)$, $(i+1, j+1/2)$, $(i, j+3/2)$, and $(i+1, j+3/2)$.

$$\frac{\partial\dot{z}}{\partial y} = \frac{\dot{z}_{(i,j+3/2)} + \dot{z}_{(i+1,j+3/2)} - \dot{z}_{(i,j-1/2)} - \dot{z}_{(i+1,j-1/2)}}{2(y_{(i+1/2,j+3/2)} - y_{(i+1/2,j-1/2)})} \tag{2.11.3.1}$$

Suppose that the material in row $j+1/2$ is steel while the material above and below the row is air. As illustrated in Fig. 14, there is nothing physically wrong with the air above and below the steel moving in opposite directions, but Eq. (2.11.3.1) calculates a large, nonphysical strain rate in the steel. Clearly the formulation must take into account in a dynamic manner the strength associated with the materials immediately adjacent to the zone under consideration. The basic strategy is to interpolate values of $\dot{z}$ for edges $(i+1/2, j+1)$ and $(i+1/2, j)$ from the edges used in Eq. (2.11.3.1) in such a manner that only the materials that have shear strength contribute to the shear strain rate.

If the material in zone $(i+1/2, j+3/2)$ can support shear, then the tangential velocity is given by Eq. (2.11.3.2a), otherwise it is approximated by Eq. (2.11.3.2b).

$$\dot{z}_{(i+1/2,j+1)} = \tfrac{1}{4}(\dot{z}_{(i,j+3/2)} + \dot{z}_{(i+1,j+3/2)} + \dot{z}_{(i,j+1/2)} + \dot{z}_{(i+1,j+1/2)}) \tag{2.11.3.2a}$$

$$\dot{z}_{(i+1/2,j+1)} = \tfrac{1}{2}(\dot{z}_{(i,j+1/2)} + \dot{z}_{(i+1,j+1/2)}) \tag{2.11.3.2b}$$

The velocity at edge $(i+1/2, j)$ is interpolated in a similar manner. A zone can support a shear stress if any of the materials in it has shear strength. An alternative approach taken in some codes is to use a mass weighting for the velocities based on the generally true observation that dense materials have more shear strength than light materials. The derivative $\partial\dot{z}/\partial y$ is evaluated by differencing the tangential velocities at the upper and lower zone edges.

$$\frac{\partial\dot{z}}{\partial y} = \frac{\dot{z}_{(i+1/2,j+1)} - \dot{z}_{(i+1/2,j)}}{y_{(i+1/2,j+1)} - y_{(i+1/2,j)}} \tag{2.11.3.3}$$

An alternative approach to defining the shear strain rate is calculate the components of the B matrix from the equations that specify how the edges are accelerated. As an example, an alternative shear strain rate for CTH [63], can be derived from its momentum equation in the y direction. To simplify the derivation, the logic associated with shear strengths of the adjacent zones is omitted. The acceleration calculations are considered in greater detail in the next section.

$$\begin{aligned}\dot{y}^{n+1/2}_{(i,j+1/2)} =& \dot{y}^{n-1/2}_{(i,j+1/2)} + \\ & \triangle t\left\{V^n_{(i+1/2,j+1/2)} + V^n_{(i-1/2,j+1/2)}\right\} \\ & \left\{2\frac{\sigma^n_{yy(i+1/2,j+1/2)} - \sigma^n_{yy(i-1/2,j+1/2)}}{y^n_{(i+1,j+1/2)} - y^n_{(i,j+1/2)}} + \right. \\ & \left.\frac{\sigma^n_{yz(i,j+1)} - \sigma^n_{yz(i,j)}}{z^n_{(i,j+1)} - z^n_{(i,j)}}\right\} / \\ & \left\{M^n_{(i+1/2,j+1/2)} + M^n_{(i-1/2,j+1/2)}\right\}\end{aligned} \tag{2.11.3.4}$$

The average shear stresses are positioned at the corners, and an additional relation is needed between the cell-centered stresses and the corner stresses.

$$\begin{aligned}\sigma^n_{yz(i,j)} = \tfrac{1}{4}\{&\sigma^n_{yz(i-1/2,j-1/2)} + \sigma^n_{yz(i-1/2,j+1/2)} \\ &+\sigma^n_{yz(i+1/2,j-1/2)} + \sigma^n_{yz(i+1/2,j+1/2)}\}\end{aligned} \tag{2.11.3.5}$$

The force due to the shear stresses acting at edge $(i, j + 1/2)$ is derived by multiplying through Eq. (2.11.3.4) by the mass of the edge, and it is simplified by restricting the equation to planar geometry.

$$\begin{aligned}F^y_{(i,j+1/2)} = \tfrac{1}{8}(y^n_{(i+1,j+1/2)} - y^n_{(i-1,j+1/2)})(&\sigma^n_{yz(i-1/2,j+3/2)} + \sigma^n_{yz(i+1/2,j+3/2)} \\ &-\sigma^n_{yz(i-1/2,j-1/2)} - \sigma^n_{yz(i+1/2,j-3/2)})\end{aligned} \tag{2.11.3.6}$$

The general force relation associated with the shear stresses can be used to define the corresponding B matrix terms.

$$F^y_{(i,j+1/2)} = -\sum_{k,\ell} B_{(k,\ell,i,j+1/2)}\sigma^n_{yz(k\ell)}V_{(k,\ell)} \tag{2.11.3.7}$$

The derivative $\partial\dot{y}/\partial z$ is calculated from Eq. (2.11.3.8) and the necessary entries can be calculated from Eq. (2.11.3.6) and Eq. (2.11.3.7) by permuting the indicies.

$$\frac{\partial\dot{z}}{\partial y} = \sum_{k,\ell} B_{(i+1/2,j+1/2,k,\ell)}\dot{z}_{(k,\ell)} \tag{2.11.3.8}$$

If all the surrounding zones can carry shear stresses, then the finite difference expression currently used in CTH for the strain rate is given by Eq. (2.11.3.9a) while the expression derived by virtual work principals is given by Eq. (2.11.3.9b).

$$\frac{\partial \dot{z}}{\partial y} = \frac{\dot{z}_{(i,j+3/2)} + \dot{z}_{(i+1,j+3/2)} - \dot{z}_{(i,j-1/2)} - \dot{z}_{(i+1,j-1/2)}}{2(y_{(i+1/2,j+1)} - y_{(i+1/2,j)})} \tag{2.11.3.9a}$$

$$\begin{aligned}\frac{\partial \dot{z}}{\partial y} =& \Big\{ (z^n_{(i+2,j+1/2)} - z^n_{(i,j+1/2)})(\dot{z}^{n+1/2}_{(i+1,j+3/2)} - \dot{z}^{n+1/2}_{(i+1,j-1/2)}) \\ &+ (z^n_{(i+1,j+1/2)} - z^n_{(i-1,j+1/2)})(\dot{z}^{n+1/2}_{(i,j+3/2)} - \dot{z}^{n+1/2}_{(i,j-1/2)}) \Big\} \Big/ \\ &\Big\{ 8(z^n_{(i+1/2,j+1)} - z^n_{(i+1/2,j)})(y^n_{(i+1,j+1/2)} - y^n_{(i,j+1/2)}) \Big\}\end{aligned} \tag{2.11.3.9b}$$

2.11.4 Accelerating the edges. The acceleration for edge $(i, j + 1/2)$ caused by σ_{yy} and σ_{yz} is given by Eq. (2.11.3.4), and a similar relation holds for edge $(i + 1/2, j)$ and the stresses σ_{zz} and σ_{yz}. To complete the description of the edge accelerations, the case where not all the surrounding zones can support a shear stress must be considered. The algorithm discussed here is used in CSQ [62] and it is representative of the algorithms used in most Eulerian finite difference codes.

The shear stresses are distributed to the zone edges based on the ability of the surrounding zones to support shear stresses. The shear stress on edge $(i+1/2, j+1)$ is nonzero only if both zones $(i+1/2, j+1/2)$ and $(i+1/2, j+3/2)$ can support a shear stress, where the volume fraction of zone $(i + 1/2, j + 1/2)$ that can support shear stresses is denoted $\Phi^n_{(i+1/2,j+1/2)}$.

$$\begin{aligned}\sigma^n_{yz(i+1/2,j+1)} = \tfrac{1}{2}\min\big(&\Phi^n_{(i+1/2,j+1/2)}, \Phi^n_{(i+1/2,j+3/2)}\big) \\ &\big(\sigma^n_{yz(i+1/2,j+1/2)} + \sigma^n_{yz(i+1/2,j+3/2)}\big)\end{aligned} \tag{2.11.4.1}$$

A similar expression is used to evaluate the shear stresses on the other three edges. The acceleration contributed by the shear stresses to the zone center is calculated by a simple central difference.

$$\ddot{y}^n_{(i+1/2,j+1/2)} = \frac{\triangle x^n_{(i+1/2,j+1/2)}}{M^n_{(i+1/2,j+1/2)}}(\sigma^n_{yz(i+1/2,j+1)} - \sigma^n_{yz(i+1/2,j)}) \tag{2.11.4.2}$$

The cell-centered shear contributions to the accelerations are distributed to the zone edges in a simple manner.

$$\ddot{y}^n_{(i,j+1/2)} = \tfrac{1}{2}(\ddot{y}^n_{(i+1/2,j+1/2)} + \ddot{y}^n_{(i-1/2,j+1/2)}) \tag{2.11.4.3}$$

2.12 CONTACT ALGORITHMS

Contact algorithms are used to prevent surfaces from interpenetrating in Lagrangian calculations. Eulerian codes do not have contact algorithms since contact is implicitly handled by the treatment of the mixed zones (zones with more than one material). While there is an extensive body of literature on contact algorithms [66], most of it is devoted to static problems. The literature devoted to hydrodynamic contact algorithms is rather small, and much of it dates back to the early finite difference work.

The contacting surfaces are often referred to a “slide lines” even in three dimensions. One surface is designated the master surface and the other, the slave. The corresponding nodes on the surfaces are also designated master and slave nodes. This nomenclature assumes that there are two distinct surfaces. Special contact algorithms are necessary if a surface is to be allowed to come into contact with itself [64] because the two surface algorithms for determining the contact points will not work with a single surface. By convention, the numbering of the surface nodes is counterclockwise around the boundary so that the outward normal is always to the right as the boundary is traversed.

Contact algorithms can be categorized based on 1) the contact search, 2) the contact and release conditions, and 3) the contact forces. The contact search is the dominant cost for many contact algorithms. In two dimensions the search is fairly simple, but it can become quite complicated in three dimensions [64]. In most cases, two surfaces are not allowed to support a tensile force between them. Not all contact algorithms calculate the interface forces directly, making it necessary to establish a set of conditions to determine when the surfaces are in contact. Two general approaches to the contact forces are found in the literature. The first calculates an interface force between the surfaces and applies it as a force boundary condition to the master surface. Typical examples of this approach are the Lagrange multiplier method [65] and the penalty method [66]. The second approach merges the slave and master surfaces by mapping the masses and stresses from the slave onto the master surface. Most of the finite difference hydrocodes, including HEMP [7] and TENSOR [6], use this approach.

2.12.1 Interface geometry. The most convenient way to describe a contact algorithm is in terms of a local coordinate system defined by the normal and tangential directions of the master segment. In the next few sections the normal direction is z in the local coordinate system, and the tangential direction is y. The relationship between the local and global coordinate systems is defined by the angle $\theta_{(\ell,\ell+1)}$, where ℓ and $\ell+1$ are the two nodes defining the master segment.

$$\begin{Bmatrix} y \\ z \end{Bmatrix}_{global} = \begin{bmatrix} \cos\theta_{(\ell,\ell+1)} & -\sin\theta_{(\ell,\ell+1)} \\ \sin\theta_{(\ell,\ell+1)} & \cos\theta_{(\ell,\ell+1)} \end{bmatrix} \begin{Bmatrix} y \\ z \end{Bmatrix}_{local}$$

$$\theta_{(\ell,\ell+1)} = \tan^{-1}\left(\frac{z_{g(\ell)} - z_{g(\ell+1)}}{y_{g(\ell)} - y_{g(\ell+1)}}\right) \tag{2.12.1.1}$$

The normal and tangential directions at a node ℓ are defined in a similar manner. The angle of the local coordinate system at a node is defined by using the two nodes adjacent to it.

$$\theta_{(\ell-1,\ell+1)} = \tan^{-1}\left(\frac{z_{g(\ell-1)} - z_{g(\ell+1)}}{y_{g(\ell-1)} - y_{g(\ell+1)}}\right) \tag{2.12.1.2}$$

All displacements, velocities, accelerations, and forces used in equations defining the various contact algorithms are written in terms of the local coordinate system of the master segment unless otherwise specified. In addition to simplifying the explanations, it is usually more computationally efficient to work in the local coordinate system. The equations defining the algorithms therefore are more representative of the FORTRAN expressions in actual codes than the more complicated equations necessary to express the algorithms in the global coordinate system.

2.12.2 Interface force methods. The simplest geometry of interest is a single slave node in contact with a master segment defined by two master nodes, see Fig. 15. While the geometry is drawn so that the master segment is horizontal, no generality is lost because a local coordinate system can always be constructed. Concentrated forces are defined which act on the slave and the two master nodes. For the moment only the normal forces associated with enforcing the contact condition are considered.

The contact tractions acting along the contact surfaces are equal and opposite for the master and slave surfaces, and their resulting contribution to the momentum equations is zero. The linear and angular momentum equations give two relations between the three nodal forces for the discrete case.

$$\begin{aligned} F_1^m &= -(1-\xi)F^s \\ F_2^m &= -\xi F^s \\ \xi &= \frac{y_s - y_1^m}{y_2^m - y_1^m} \end{aligned} \tag{2.12.2.1}$$

The only remaining task is the calculation of F^s. The methods most commonly used are the penalty method [66], Lagrange multipliers [65], imposing a displacement constraint at the next time level (a form of explicit Lagrange multipliers) [34], [67], and imposing a velocity constraint [22].

The penalty method is the simplest conceptually. Rather than impose the contact constraint exactly, a resisting force proportional to the amount of error in the constraint is applied to the slave node.

$$F^s = -kz^s \tag{2.12.2.2}$$

As the value of the surface stiffness, k, increases, the closer the penalty method comes to enforcing the contact constraint exactly. A large value of k reduces the stable time step size. One choice for the stiffness that has worked well in DYNA2D [18] is dependent on the mesh size and stiffness of the master side, where f is a scale factor (which defaults to 0.1), K_i is the bulk modulus of the ith master segment, A_i is the area of the element that has the master segment as a boundary, and V_i is the element volume.

$$k_i = \frac{fK_iA_i^2}{V_i} \tag{2.12.2.3}$$

Belytschko and Neal [69] limit the value of f on the basis of a one-dimensional stability analysis. Their numerical experiments indicate that instabilities can still occur unless the time step size is reduced if f is near its limiting value.

$$\begin{aligned} f &\le \frac{2C(1-C)}{(1-2C)} \\ C &= \frac{1}{c^2} \quad \text{where } c \text{ is the Courant number} \end{aligned} \tag{2.12.2.4}$$

In addition to limiting f, they limit the maximum penalty force. Their analysis considers only two nodes interacting, which is sufficient for their pinball algorithm. The pinball algorithm, which is discussed later, replaces the line segment topology of the contact surfaces with sets of spheres embedded in the elements. Their contact calculations, therefore, consider the forces between the pairs of contacting spheres, where F^s is the force acting between the pinballs, M^i are the element masses of the master and slave elements, and u^i are the velocities of the pinballs.

$$F^s \le \frac{M^sM^m|u^m - u^s|}{\triangle t(M^s + M^m)} \tag{2.12.2.5}$$

The penalty method applies concentrated forces at the nodes instead of a distributed force along the contact surfaces. This can lead to a degree of mesh sensitivity that is reduced by using the algorithm proposed by Simo [70]. Instead of evaluating the penalty forces only at the nodes, he integrates the penalty force with Gaussian quadrature on an average surface defined by the master and slave surfaces.

In hypervelocity impact calculations, where hypervelocity refers to impact velocities on the order of the wave speed in the material, the time step must be reduced to a small fraction (∼.1) of the Courant limit regardless of the contact algorithm. A scale factor in Eq. (2.12.2.3) on the order of $10 \sim 100$ can be used without further reductions in the time step size. The penalty algorithm is routinely used by Hallquist [71] and Belytschko [69], [72] in hypervelocity penetration calculations.

The Lagrange multiplier method adds additional constraint equations that must be solved simultaneously with the equations of motion. As the master and slave move relative to each other, both the structure and number of equations change. The displacement constraint for the three node model problem is given by Eq. (2.12.2.6).

$$\phi = y^s - [(1-\xi)y_1^m + \xi y_2^m] = 0 \tag{2.12.2.6}$$

The force acting on the node k is $\lambda \partial\phi/\partial y^k$, where k is one of the two master nodes or the slave node. A comparison of the constraint forces with Eq. (2.12.2.1) demonstrates that the Lagrange multiplier method satisfies the conditions imposed by the conservation of momentum. Note that if λ is negative, the nodes are in tension and the contact constraint must be turned off.

$$\begin{aligned} F^s &= \lambda \\ F_1^m &= -(1-\xi)\lambda \\ F_1^m &= -\xi\lambda \end{aligned} \tag{2.12.2.7}$$

Hydrocodes use a diagonal mass matrix to avoid the cost of solving a linear equation set, and no production hydrocode directly solves the coupled set of equations. Carpenter, Taylor, and Katona [171] use Gauss-Seidel to solve the equations in their research code and Belytschko [73] is currently experimenting with iterative methods.

In EPIC [22], Johnson requires that the final velocity of the contact point on the master surface equal the final velocity of the slave node. Although he works in terms of momentum increments, his formulation is readily recast in terms of interface forces. Johnson's velocity increments, $\triangle u$, correspond to the $\triangle tF/M$ terms in the following equations. The velocities already include the velocity increment over the time step from all the other force contributions.

$$\dot{z}^s + \triangle t\frac{F^s}{M^s} - (1-\xi)[\dot{z}_1^m + \triangle t\frac{F_1^m}{M_1^m}] - \xi[\dot{z}_2^m + \triangle t\frac{F_1^m}{M_2^m}] = 0 \tag{2.12.2.8}$$

The slave node force, which again represents a tensile force if it is negative, is solved for by substituting Eq. (2.12.2.1) into Eq. (2.12.2.8).

$$F^s = -\frac{\dot{z}^s - (1-\xi)\dot{z}_1^m - \xi\dot{z}_2^m}{\frac{\triangle t}{M^s} + (1-\xi)^2\frac{\triangle t}{M_1^m} + \xi^2\frac{\triangle t}{M_2^m}} \tag{2.12.2.9}$$

This approach does not enforce the displacement constraint directly. If the slave node is not exactly on the master surface at the beginning of the time step, forcing the slave node and the contact point to have the same velocity is not sufficient to bring the slave node to the master surface within the next time step. As with virtually all explicit contact algorithms, the displacement constraint is not satisfied exactly. This algorithm is frequently used for penetration problems with the eroding slideline option [22] in EPIC, where small errors in the displacement constraints are less important than stability.

Several researchers have proposed methods that may be classed as explicit Lagrange multiplier algorithms since the Lagrange multiplier at t^n is calculated in an explicit manner such that the contact constraints are satisfied at t^{n+1}. It appears that the first publication of this idea in a general multidimensional setting was in 1987 by Flanagan and Taylor [34]. Zhong [67] published a collection of papers for his thesis in 1988, which contains similar ideas, but he differs in the manner in which he solves the system of equations generated by the contact constraints. More recently, Carpenter, Taylor, and Katona [171] have proposed a Gauss-Seidel strategy for solving the equations that converges in only a few iterations.

The explanation of the method in the PRONTO manual [34] does not do justice to their algorithm since they motivate the development of their method by an example of a single node hitting a rigid wall. Instead, a significantly more complicated problem is considered to motivate their algorithm.

Before proceeding, a discussion of the time centering of their contact forces and displacement constraints is in order. The contact algorithm is called after all other force contributions have been accounted for and the preliminary accelerations have been calculated. The contact algorithm adjusts the accelerations (forces) at t^n so that the displacement constraints are satisfied at time t^{n+1}. This separation of the contact force calculation from the other forces is similar to the operator split used to sequentially update the accelerations with the contributions from the viscosity and the pressure.

Consider a one-dimensional problem with N slave nodes with varying masses M_i^s and velocities u_i^s striking a single master surface with a mass M^m and a velocity u^m. These velocities are evaluated at $t^{n+1/2}$ and include the effects of all the forces at t^n other than the contact forces. At t^n the slave nodes, with coordinates x_i^s, are either in perfect contact or outside of the master surface, and at t^{n+1}, all the nodes will penetrate the surface unless the contact forces are applied. The conditions for the slave nodes being exactly coincident with the master surface at t^{n+1} are given by N displacement constraints.

$$x_i^s + \triangle t(u_i^s + \triangle t\frac{F_i^s}{M_i^s}) - x^m - \triangle t(u^m + \triangle t\frac{F^m}{M^m}) = 0 \tag{2.12.2.10}$$

Note that the time step sizes $\triangle t^n$ and $\triangle t^{n+1/2}$ are set to a common value of $\triangle t$, a condition that is generally not true. This is an unnecessary approximation introduced by Flanagan and Taylor, but it probably does not introduce a significant amount of error into the solution. The predicted displacements at t^{n+1}, $\hat{x}$, are introduced to simplify the notation.

$$\begin{aligned} \hat{x}_i^s + (\triangle t)^2\frac{F_i^s}{M_i^s} - \hat{x}^m - (\triangle t)^2\frac{F^m}{M^m} &= 0 \\ \hat{x}_i^s &= x_i^s + \triangle t u_i^s \\ \hat{x}^m &= x^m + \triangle t u^m \end{aligned} \tag{2.12.2.11}$$

The momentum equation provides the necessary $N+1$ equation.

$$\sum_i F_i^s + F^m = 0 \tag{2.12.2.12}$$

The resulting set of equations is diagonal with a border on the right and lower edges. It is readily solved explicitly because of its simple structure. To keep the terms similar to the equations in the PRONTO manual, the displacement constraints have been divided by $(\triangle t)^2$. The coefficient matrix is A, the right hand side is b, and the force vector is ordered $\{F^s, F^m\}$.

$$\begin{aligned}
A_{(i,i)} &= \frac{1}{M_i^s} \\
A_{(i,N+1)} &= -\frac{1}{M^m} \\
A_{(N+1,i)} &= 1 \\
A_{(N+1,N+1)} &= -1 \\
b_{(i)} &= -\frac{(\hat{x}_i^s - \hat{x}^m)}{(\triangle t)^2} \\
b_{(N+1)} &= 0
\end{aligned} \tag{2.12.2.13}$$

The partitioned solution gives the master force in terms of the sum of a set of forces, f_i^s, that would exactly bring the nodes to rest against a rigid barrier.

$$\begin{aligned}
f_i^s &= M_i^s \frac{(\hat{x}_i^s - \hat{x}^m)}{(\triangle t)^2} \\
F^m &= M^m \left(\frac{\sum_i f_i^s}{M^m + \sum_i M_i^s} \right)
\end{aligned} \tag{2.12.2.14}$$

The solution for F_i^s is obtained by substituting Eq. (2.12.2.14) into the original equations. Flanagan and Taylor express the solution in terms of accelerations.

$$\begin{aligned}
\dot{u}^m &= \frac{F^m}{M^m} = \frac{\sum_i f_i^s}{M^m + \sum_i M_i^s} \\
\dot{u}_i^s &= \frac{F_i^s}{M_i^s} = \dot{u}^m - \frac{f_i^s}{M_i^s}
\end{aligned} \tag{2.12.2.15}$$

When the master surface is defined by two nodes, the same problem structure occurs, but the coefficient matrix has nonzero edges two columns and two rows wide. To decouple the system, f_i^s is divided between the two master nodes according to Eq. (2.12.2.1), and the two master nodes are treated independently. The contributions of the slave masses are also divided according to the same rule. This decoupling procedure is applied to all pairs of slave nodes and master segments, and the master node accelerations are calculated according to Eq. (2.12.2.15). The slave node accelerations are calculated by using the second equation in Eq. (2.12.2.15), where $\dot{u}^m$ is the acceleration of the contact point on the master surface. In two dimensions, the same procedure is followed for each coordinate direction where the global force components are calculated from Eq. (2.12.1.1).

Zhong [67], [74] has independently developed and implemented into DYNA3D a contact algorithm that is very similar to the Flanagan-Taylor algorithm. He distinguishes between the time step sizes $\triangle t^n$ and $\triangle t^{n+1/2}$. Instead of distributing f^s to the master nodes by using the interpolation functions, he constructs a defense node with a mass M^d and solves the two node contact problem.

$$\begin{aligned} M^d &= \sum_i \frac{N_i(\xi)}{\sum_j N_j^2(\xi)} M_i^m \\ N_1(\xi) &= 1 - \xi \\ N_2(\xi) &= \xi \end{aligned} \tag{2.12.2.16}$$

The interface force, F^s, is calculated between the slave node (“hitting node” in his papers) and the defense node. Flanagan and Taylor account for the forces associated with several slave nodes contributing contact forces to a particular master node by distributing and then summing the current forces contributions from the slave nodes to each master node. In contrast, Zhong uses the accumulated slave forces from the previous step and calculates an incremental change to the slave contact forces. The slave contact forces are then distributed to the master nodes by mass weighting.

$$F_i^m = -\frac{N_i(\xi)}{\sum_j N_j^2(\xi)} \frac{M_i^m}{M} F^s = -\frac{N_i(\xi) M_i^m}{\sum_j N_j(\xi) M_j^m} F^s \tag{2.12.2.17}$$

According to Hallquist [71], Zhong is using his algorithm in an iterative manner to enforce the contact constraints to required degree of accuracy. The magnitude of the contact force is proportional to the degree of error in the contact inequality and it is calculated completely independently of the other forces. After the first pass of the contact algorithm, the geometry is updated based on the contact forces. If the updated geometry at $n+1$ is again treated as the $\hat{x}$ geometry and the contact calculation is repeated (holding time constant), then the second pass should reduce the contact error at $n+1$ even more. In this manner, the Zhong algorithm is used as a fixed point algorithm where the number of iterations is determined by the norm of the contact error. While there is no formal proof for its convergence, its successful application to a broad range of problems argues that it is convergent.

Carpenter, Taylor, and Katona [171] have proposed a Gauss-Seidel algorithm for solving the system of equations generated by the explicit Lagrange multiplier method. Their scheme effectively decouples the contact constraints from each other so that during a Gauss-Seidel sweep, the contact segments can be processed independently. They have demonstrated that while their Gauss-Seidel algorithm coverges linearly, only a few sweeps are needed in practice to acheive a good solution.

2.12.3 The finite difference hydrodynamic contact algorithms. The finite difference hydrodynamics community is generally concerned more with the accurate propagation of stress waves across a contact interface than enforcing the contact constraints exactly. The philosophical differences between the hydrodynamic algorithms and the ones in the previous section are somewhat similar to the differences between the mixed and displacement based finite element formulations. As a consequence, hydrodynamic algorithms differ considerably from the algorithms used in structural analysis. Instead of calculating an interface force, the master and slave surfaces are merged locally in the contacting regions by mapping the stress and mass of the slave surface onto the master surface. After updating the motion of the master surface, the slave surface is projected onto it. Note that the master surface acceleration is governed by the surrounding stresses rather than the constraint forces as in the previous section. A detailed review of the hydrodynamic methods is presented in Ref. [75]. This section only summarizes the major ideas in the previous review.

The first step is the evaluation of the pressure contributed by the slave elements on the boundary of the master segments. Wilkins assumes in HEMP [7] a constant pressure distribution along the master segment and interpolates the value of the pressure at the midpoint of the master segment from the two slave elements that are closest to the segment midpoint, see Fig. 16.

$$\begin{aligned} p_{(\ell+1/2)} &= (1-\xi)p_{(k-1/2)} + \xi p_{(k+1/2)} \\ \xi &= \frac{L'}{L} \end{aligned} \tag{2.12.3.1}$$

The pressures in Eq. (2.12.3.1) are the normal stress components using the normal defined by the master segment. Note that this expression for the pressure does not include any contribution from the hoop stress.

$$p = -(\sigma_{yy}\mathrm{sin}^2\theta_{(\ell,\ell+1)} + \sigma_{zz}\mathrm{cos}^2\theta_{(\ell,\ell+1)} - \sigma_{yz}\mathrm{sin}2\theta_{(\ell,\ell+1)}) \tag{2.12.3.2}$$

Bertholf and Benzley create a new mesh for TOODY on the slave side such that the new slave nodes are exactly opposite the master nodes [76]. TOODY and TENSOR [6] appear to be the first attempts to use an adaptive mesh for large deformation contact problems. The TENSOR algorithm also creates a new mesh in the same manner. Since the new mesh is used only for the contact algorithm, none of the diffusion problems associated with the continuous advection of material through a mesh are encountered. Referring to Fig. 17, the solution variables, ϕ, for the new element opposite the master segment $(\ell, \ell+1)$ are the length-weighted averages of the original slave elements.

$$\begin{aligned} \phi &= \sum (\tilde{y}_{(k)} - \tilde{y}_{(k+1)}) \phi_{(k+1/2)} / y_{(\ell+1)} - y_{(\ell)} \\ \tilde{y}_{(k)} &= \min\big(\max(y_{(k)}, y_{(\ell+1)}), y_{(\ell)}\big) \end{aligned} \tag{2.12.3.3}$$

The same interpolation procedure is used in DYNA2D [75], but the normal pressure is calculated from the force contributions of the slave elements. Only the nodal force contributions from element $(k+1/2)$ are used for $p_{(k+1/2)}$; the nodal forces in Eq. (2.12.3.4) are not from the assembled force vector.

$$p_{(k+1/2)} = (F^{\sigma}_{(k)} + F^{\sigma}_{(k+1)}) / L_{(k,k+1)} \tag{2.12.3.4}$$

The mass distribution from the slave surface to the master surface is the second step in the hydrodynamic algorithms. Wilkins originally assumed a constant mass distribution, a simplification he acknowledged is not generally valid, and introduced an amplification factor, Z, that is used to scale the master node masses.

$$Z = \frac{\sum_{i=1}^{master} M_i^m + \sum_{i=1}^{slave} M_i^s}{\sum_{i=1}^{master} M_i^m} \tag{2.12.3.5}$$

Since the TOODY and TENSOR algorithms construct new slave meshes that are contiguous with the master meshes, the master nodes are treated as interior nodes and the standard mass lumping scheme is used for the master node masses.

The mass is distributed from the slave surface to the master surface in DYNA2D by using the TOODY algorithm for distributing the pressures, but with respect to a staggered mesh. The local coordinate system is defined by the slave nodes k and $k+1$, and the length of the slave segment is $L_{(k+1/2)}$.

$$\begin{aligned} \triangle M_\ell^m &= \sum (\tilde{y}_{(k)} - \tilde{y}_{(k+1)}) M^s_{(k+1/2)} / L_{(k+1/2)} \\ \tilde{y}_{(k)} &= \min\big(\max(y_{(k)}, y_{(\ell+1/2)}), y_{(\ell-1/2)}\big) \\ M^s_{(k+1/2)} &= \tfrac{1}{2} \rho_{(k+1/2)} A_{(k+1/2)} \end{aligned} \tag{2.12.3.6}$$

The master surface is accelerated during the third step. HEMP treats the master surface as a free boundary with an applied pressure. In contrast, the TOODY algorithm treats the master nodes as interior nodes surrounded by the master surface elements and the new mesh created from the slave elements. Both the TENSOR and DYNA2D algorithms add an additional interface force term based on the Coriolis acceleration. The Coriolis terms are derived from requiring that the time derivative of the the normal velocity difference across the contact surface be zero, where the velocity $\dot{z}_\ell^{s*}$ is the velocity of the slave node in the new mesh across from master node ℓ.

$$\frac{d}{dt}\left(\dot{z}_\ell^{s*} - \dot{z}_\ell^{m}\right) = 0 \tag{2.12.3.7}$$

The difference in the accelerations of master node ℓ and the new node ℓ^* is the Coriolis term, a_c, arising from the differentiation in Eq. (2.12.3.7) of the transformation matrix defined in Eq. (2.12.1.1).

$$\begin{aligned} \ddot{z}_\ell^{s*} - \ddot{z}_\ell^{m} &= a_c = 2\omega(\dot{y}_\ell^{s*} - \dot{y}_\ell^{m}) \\ \omega &= \frac{\dot{z}_{(\ell-1)} - \dot{z}_{(\ell+1)}}{z_{(\ell-1)} - z_{(\ell+1)}} \end{aligned} \tag{2.12.3.8}$$

The stress divergence term for the new node, F_ℓ^{s*}, is the force on node ℓ determined from the pressure boundary condition defined by Eq. (2.12.3.4). The mass associated with node ℓ^* is $\triangle M_\ell^m$, defined in Eq. (2.12.3.6). An interface force, $\triangle f_\ell$, is calculated to enforce Eq. (2.12.3.8).

$$\begin{aligned} \frac{F_\ell^{s*} + \triangle f_\ell}{\triangle M_\ell^m} - \frac{F_\ell^m - \triangle f_\ell}{M_\ell^m} &= a_c \\ \triangle f_\ell = &\frac{\triangle M_\ell^m F_\ell^m + F_\ell^{s*} M_\ell^m + M_\ell^m \triangle M_\ell^m a_c}{\triangle M_\ell^m + M_\ell^m} \end{aligned} \tag{2.12.3.9}$$

The interface force $\triangle f_\ell$ is subtracted from the master surface nodes, and the master surface is accelerated using the original master surface masses.

The final acceleration of the master surface is the sum of the original tangential component of the acceleration and the normal acceleration calculated by the contact algorithm. During the calculation of the normal and tangential components of the acceleration, the normal at the master node ℓ is $\theta_{(l-1,l+1)}$. The velocity and displacement of the master surface are calculated using the updated acceleration.

Slave node normal velocities and displacements are usually interpolated from the master surface while the tangential components are calculated from the slave node tangential accelerations. HEMP projects the slave nodes onto the master surface along the logically perpendicular mesh line (i.e., if the contact surface is a k line, the slave node (k,ℓ) is projected onto the master surface along the direction defined by (k,ℓ) and $(k,\ell+1)$). The velocity of the slave nodes is simply determined by differencing the coordinates at the n and $n+1$ time steps. In TOODY, the slave nodes are moved tangentially according to their velocities and then projected normally onto the master surface. As with HEMP, the final slave velocities are calculated by differencing. The TENSOR procedure is identical to the TOODY procedure except the normal velocity is interpolated from the master surface. DYNA2D uses the most complicated update for the slave node velocities. The tangential component of the velocities are calculated from the slave node accelerations, and the normal acceleration is calculated using Eq. (2.12.3.8) complete with the Coriolis term. The normal velocities are integrated using the normal accelerations and combined in the familiar manner with the tangential velocities. Instead of projecting the slave nodes to the master surface, an additional increment of one percent of the velocity needed to close the gap in one time step is added to the normal velocity component.

Hallquist et al. [75] demonstrated that some care must be used in the choice of a normal direction for the slave nodes. When the normal direction is chosen to be the normal of the master segment, hourglassing and spurious shearing results. Better solutions are obtained when the normal direction of slave node k is defined by $\theta_{(k-1,k+1)}$, see Fig. 19. Similar results were found with a single surface contact algorithm by Benson and Hallquist [64] which uses the penalty formulation.

The last method to be discussed here calculates the interface forces based on Godunov's method and is implemented in CAVEAT [3], a Godunov ALE code. Since the Godunov methodology, which advances the solution by solving a series of Riemann problems, is discussed in detail later, only the general strategy of the contact method is presented in this section. The Riemann problem, mentioned in Section 1.3, is a generalization of the shocktube problem. The velocity, density, and energy are discontinuous across an interface, which in this case is the contact point between the master and slave surfaces. An exact or approximate Riemann solver calculates the pressure and the normal velocity at the contact discontinuity.

The CAVEAT algorithm [172] solves the Riemann problem for the contact pressure and velocity between each pair of overlapping master and slave segments along the slide line. The contributions from each overlapping pair are summed appropriately for each contact segment to obtain the pressure and normal velocity, and there an optional spatial smoothing available. Once the pressure and velocity are available on the contact boundary, the boundary zones in the same manner as the interior zones, see section 2.14. This method should be, by construction, nearly optimal for transmitting shocks across contact interfaces since it is based on solving the Riemann problem.

2.12.4 Contact and release conditions. The simplest contact and release conditions are those used in the penalty method: the penalty force is turned on whenever the contact inequality constraint is violated. This is also the condition used in the Flanagan-Taylor algorithm, but with the $\hat{x}$ coordinates defining the contact geometry. The description of the TENSOR contact algorithm [6] is for a slip line (the nodes are always in contact, but relative slip is allowed) and the contact condition is trivial. Lagrange multiplier algorithms turn on the contact when the contact inequality is violated and turn it off when the Lagrange multiplier indicates a tensile force between the surfaces. In DYNA2D, and in most of the hydrocode algorithms, the surfaces are released when the relative acceleration between the slave node and the master segment indicates separation.

DYNA2D checks all the slave nodes not in contact with the master surface and reduces the time step if it necessary so that no slave node penetrates during the next time step. When a slave node impacts a master segment, momentum conservation is used to calculate the normal post-impact velocity, $\dot{z}^{+}$, of the master surface, where $\triangle M_{\ell i}^{m}$ is the mass contribution from slave node i to master node ℓ.

$$\dot{z}_{\ell}^{+} = \frac{M^{m}\dot{z}_{\ell}^{m} + \sum \triangle M_{\ell i}^{m}\dot{z}_{i}^{s}}{M^{m} + \sum \triangle M_{\ell i}^{m}} \tag{2.12.4.1}$$

Hughes, et al. [77] also change the velocity at impact, but they use the velocity of a contact discontinuity. One of the assumptions of their work is the slave nodes directly strike a master node, therefore they do not address the issue of distributing the velocity change associated with the impact to two master nodes defining a master segment. The wave velocity in the master surface is c^{m} and in the slave surface, c^{s}. Release conditions based on a local wave analysis are also employed. Although their contact algorithm is not implemented in a hydrocode, their use of a wave analysis in a contact algorithm is unique outside of the CAVEAT work.

$$\dot{z}_{\ell}^{+} = \frac{\rho^{m}c^{m}\dot{z}_{\ell}^{m} + \rho^{s}c^{s}\dot{z}_{k}^{s}}{\rho^{m}c^{m} + \rho^{s}c^{s}} \tag{2.12.4.2}$$

Multiplying the numerator and denominator by $\triangle t$ gives a characteristic depth into the master and slave surfaces. Since the Courant number limits the time step size, the characteristic depth is less than the full depth of the element. Comparing Eq. (2.12.4.1) to

(2.12.4.2), the characteristic depth associated with the momentum balance performed in DYNA2D is one half the element depth.

2.12.5 Symmetry. Symmetry is highly desirable in a contact algorithm primarily for geometrical robustness. Problems involving corner contact generally benefit from a symmetric treatment, and symmetric problems may become unsymmetric if the contact algorithm is not symmetric. There are no hard and fast rules for choosing which surface is the master and which is the slave. A particular surface may be chosen as the master surface if it is: 1) the denser material, 2) the stronger material, 3) the side with the coarsest mesh, and 4) the material which will move the least. In many problems none of these rules leads to an unquestionable choice for the master surface. For example, in a problem with variable zoning, the appropriate side for the master surface may change along the contact region.

The penalty method is usually implemented in a completely symmetric manner by simply making two passes over the contact surfaces during a time step, with the second pass reversing the definitions of the master and slave surfaces. Even though this doubles the cost of the contact algorithm, a symmetric treatment has been found valuable enough that it is used in both the DYNA2D and DYNA3D penalty algorithms without any option for a nonsymmetric treatment.

Flanagan and Taylor [34] partition the time step with a factor β, which has a value between zero and one. During the first β fraction of the time step, the contact forces are calculated using the master and slave surfaces as they are designated in the input data. At $t^n + \beta\triangle t$, the definitions of the contact surfaces reversed, the new geometry is calculated, and new contact forces are calculated with the new geometry. No discussion was presented on the merits for different choices of β, nor was there any discussion of whether or not they encountered aliasing errors associated with using the original choice of the master surface at the beginning of each step.

The TENSOR algorithm comes the closest to being perfectly symmetric of all the finite difference hydrodynamic algorithms. After two passes are made with the surface definitions reversed on the second pass, the two surfaces are accelerated independently. The final projection to enforce the contact condition is, however nonsymmetric, since the original slave surface is always projected onto the master surface.

2.12.6 Friction. Only the simplest models for friction are presented here, and the interested reader is referred to the work by Kikuchi and Oden [66], and Zhong [67] for more complicated formulations. Although friction laws in continuum mechanics are generally expressed in terms of a normal pressure and a resisting traction stress, the friction laws for the contact algorithms are expressed in terms of concentrated forces. The classic Coulomb friction model states that the relative slip between to surfaces is zero if the tangential force is below $\mu_s F^s$, where μ_s is the coefficient of static friction. If the tangential force is greater than $\mu_s F^s$, relative slip occurs and the magnitude of the force of friction opposing it is $\mu_d F^s$, where μ_d is the dynamic coefficient of friction.

The two major difficulties in an explicit implementation of friction are first, determining if a node is sticking or slipping, and second, calculating the tangential force for the sticking condition. Implicit formulations sometimes experience difficulties in converging because of the extremely nonlinear nature of the stick-slip condition.

PRONTO calculates the sticking force and the sliding force, and chooses the minimum of the two [34]. The relative tangential velocity is calculated as the difference between the slave node velocity and the velocity of its contact point on the master surface.

$$\triangle \dot{y} = \dot{y}^s - [(1-\xi)\dot{y}_1^m + \xi \dot{y}_2^m] \tag{2.12.6.1}$$

The sticking force is the tangential force applied to the master surface necessary to eliminate the relative velocity in a single time step.

$$F_s^s = -\frac{M^s \triangle \dot{y}}{\triangle t} \tag{2.12.6.2}$$

The coefficient of dynamic friction is the same one used in HONDO II [78].

$$\mu_d = \mu_\infty + (\mu_0 - \mu_\infty) e^{-\gamma \triangle \dot{y}} \tag{2.12.6.3}$$

The minimum of the sticking force and the frictional force is chosen, where F_t^s denotes the final tangential frictional force.

$$F_t^s = \frac{F_s^s}{|F_s^s|} \min(\mu_d F^s, |F_s^s|) \tag{2.12.6.4}$$

The nodes can oscillate if the force of friction is simply $\pm \mu F^s$ because the on-off nature of the classic law does not permit intermediate values that can just cancel small motion. A method for smoothing the discontinuous nature of the friction force, therefore, is necessary. The sticking force calculated by Eq. (2.12.6.2) is a viscous force with a damping coefficient $M^s/\triangle t$ that acts to damp any possible oscillations.

A different method, developed by Hallquist, that is analogous to an elastic-plastic constitutive law, is used in DYNA2D and DYNA3D [64]. The relative slip between the master and slave surfaces is calculated by keeping track of the isoparametric coordinates and master segment number for every slave node that is in contact with the master surface.

$$\triangle y^s = \big(y^s(\xi^{n+1}) - y^s(\xi^n)\big) \tag{2.12.6.5}$$

This increment in displacement is treated as a measure of "strain" and the radial return algorithm is used to return the tangential force to the "yield surface." The elastic modulus, k_i, is the same as the master surface stiffness used for the normal force calculation in the penalty method.

$$\begin{aligned} F_y^s &= \mu_d |F^s| \\ F_t^{trial} &= F_t^s(n) - k_i \triangle y^s \\ F_t^s(n+1) &= \left\{ \begin{matrix} F_t^{trial} \text{ if } |F_t^{trial}| \leq F_y^s \\ \text{sign}(F_t^{trial}) F_y^s \text{ if } |F_t^{trial}| > F_y^s \end{matrix} \right\} \end{aligned} \tag{2.12.6.6}$$

As a slave node moves across a master segment, the friction force builds to a maximum determined by the normal force. If the slave node suddenly stops, the tangential force does not drop to zero, but remains at the value it had during the last time step. The node must move a small increment in the opposite direction to its previous motion or leave the master surface to reduce the force of friction to zero. As Zhong points out [67], this model can be viewed as implementing the sticking case with the penalty method over a radius of $\mu_d F^s / k_i$ with a penalty stiffness of k_i. The relative slip is the violation of the sticking constraint in the same manner as the surface penetration is a violation of the contact inequality. This algorithm requires slightly more storage than the PRONTO algorithm because the friction force, the isoparametric coordinate, and the master segment are stored. When there is no friction, only the nearest master node to a slave node is stored.

2.12.7 Simplifications and approximations. To reduce the computational cost of contact problems, researchers have simplified the problem geometry [69], [75] and the contact force calculations [75].

Belytschko and Neal [69] replace the contact surfaces with spheres centered in the elements in their pin-ball algorithm. The penetration for two spheres is the sum of the radii minus the distance between the centers, and the unit normal is along the line connecting the centers. Belytschko and Neal take advantage of the simplified geometry not only to reduce the cost of the force calculations, but also to simplify the vectorization of their contact search (Hallquist has also completely vectorized the search strategies within DYNA2D and DYNA3D [71]).

The force calculations in the DYNA2D hydrodynamic contact algorithm [75] can be simplified by approximating the normal force and slave mass distributions. The mass density, m_k^*, at each slave node is its nodal mass divided by half the distance between its two adjacent nodes, and the force density, f_k^*, is approximated in the same manner from the nodal forces. The mass density and stress density for a slave segment is the average of the values at its nodes. The mass and normal force is then mapped onto the master surface in the manner described in the previous sections. These simplifications significantly reduce the cost of the calculation without adversely affecting the accuracy.

If two surfaces are tied together, additional simplifications are possible. The contact point of the slave node on the master surface is fixed in terms of its isoparametric coordinates. The mass and force at the slave node are distributed to the master segment nodes in the same manner as in Eq. (2.12.2.1). This simplification works best when the slave mesh is an integer multiple of the master mesh, and hourglassing problems can occur if this is not the case. This simplification can not be used in a contact-impact algorithm because it invariably excites the hourglass modes. Even the hydrodynamic contact algorithms that allow sliding but prohibit separation can not be simplified beyond the level discussed in the previous paragraph without hourglassing problems.

2.12.8 Eroding surface contact algorithms. Penetration calculations are difficult to perform with a Lagrangian code because it does not permit the mesh to tear. To overcome this difficulty, and to model the effects of erosion in a hypervelocity impact, Johnson [22] developed an eroding slideline capability for EPIC. When elements satisfy a material dependent erosion criteria (which may be based on plastic strain, maximum principal stress, element distortion, etc) they are deleted from the mesh and their vertices are free to move as point masses. After an element is deleted, its contact segments are deleted from the slidelines and the new free surfaces, which were generated by the element's deletion, are added to the slideline definitions. Both the target and the penetrator can erode, and their erosion criteria are usually different. Experiments have demonstrated that this method provides a reasonably accurate prediction of the hole size in the target, and the final velocity and mass of the penetrator [80]. The contact forces are calculated by using Eq. (2.12.2.9).

Belytschko and coworkers have worked on improving the efficiency of Johnson's eroding slideline. They determine the eroded surface from all the nodes with nonzero normal vectors [72]. After zeroing out an accumulation array, they loop through all the elements and calculate their contribution to the normal of their nodes. If a node is in the interior of the mesh, then the sum of all the element contributions is zero, while the sum for the exterior nodes is nonzero. The calculation of the element contributions to the normals vectorizes easily, increasing the algorithm's efficiency. The efficiency of the contact search is increased by using a two-dimensional hash code to determine which master segments are close a particular slave node. The most recent development is the "pin ball" contact algorithm [69] discussed in the previous section.

2.12.9 Single surface contact algorithms. When the deformations are very large, a surface may buckle and come into contact with itself. The standard search algorithms will not work in this case, and a special single surface contact algorithm [64] developed by Benson and Hallquist can be used. A list of the three closest nodes to each node is generated by using a three-dimensional nested bucket sort every ten time steps. The nodes that share a common element with a node are automatically eliminated from the list. Each node has a list of contact segments with that node as part of their definition. The list of candidate contact segments for a node is the list of all the contact segments associated with the three closest nodes, and the actual contact segment is determined by using the efficient local search developed for the regular two surface contact algorithm [75].

Zhong [67] has recently developed a heirarchical search method for single surface contact that he claims is significantly faster than the nested bucket sort. Hallquist [71] has, however, recently rewritten and vectorized the single surface contact algorithm code and improved its efficiency by a factor of two to five. Since these contact algorithms are complicated, there is till room for improving their speed and robustness. It impossible to predict at this time which algorithm will turn out to be superior in the long run.

2.13 TIME STEP CONTROL

The time step size for the central difference integration method is determined by stability considerations. In hydrodynamic problems, the typical stable time step sizes are so small that accuracy is not a consideration. For a one-dimensional problem the stable time step is limited by the Courant condition, where $\triangle x$ is the length of the smallest element, and a is the speed of sound.

$$\triangle t \leq \frac{\triangle x}{a} \tag{2.13.1}$$

An alternative analysis [12] expresses the same result in a different form, where ω is the highest natural frequency in the system.

$$\omega \triangle t \leq 2 \tag{2.13.2}$$

The introduction of a shock viscosity requires that the Courant condition be modified. A detailed analysis [81] for one dimension was performed by Hicks for WONDY in 1978. Associated with the viscosity is a damping coefficient, c. The definition given here for c assumes the standard quadratic form for the shock viscosity is being used.

$$\begin{aligned} \triangle t &\leq \frac{\triangle x}{c + \sqrt{c^2 + a^2}} \\ c &= \left\{ \begin{matrix} c_L a + c_o|\triangle u| \quad \text{if } \triangle u < 0 \\ c_L a \quad \text{if } \triangle u \geq 0 \end{matrix} \right\} \end{aligned} \tag{2.13.3}$$

The time step for each element is evaluated according to Eq. (2.13.3), and the stable time step for the system is the minimum value over all the elements. The time step is often multiplied by safety factor (<1) to ensure its stability. For instance, the default safety factor in DYNA2D is 0.67 [82], in SHALE [23], 0.25, in TENSOR [6], 0.5, in HEMP [7], .333, and in DYNA3D [83], 0.90 (0.67 if there are explosive materials). In hypervelocity impact problems, the maximum of the sound speed and the material velocity replaces the sound speed in Eq. (2.13.3) in SHALE, while MESA [101] uses the sum of the absolute value of the material velocity and the sound speed.

The value of the sound speed appearing in Eq. (2.13.3) is usually based on the elastic value of the shear modulus and a contribution from the equation of state.

$$a = \sqrt{\frac{4G}{3\rho_0} + \frac{\partial P}{\partial \rho}\Big|_E + \frac{pV^2}{\rho_0}\frac{\partial P}{\partial E}\Big|_\rho} \tag{2.13.4}$$

In PRONTO [34], Flanagan and Taylor dynamically calculate hypoelastic material constants based on the material response. This strategy gives them a larger time step in elements that are deforming plastically since the plastic moduli are smaller than the elastic values.

$$
\begin{aligned}
3\hat{K} = 3\hat{\lambda} + 2\hat{\mu} &= \frac{\triangle\sigma_{kk}}{\triangle t\dot{\epsilon}_{mm}} \\
2\hat{\mu} &= \frac{\sigma'_{ij}\dot{\epsilon}'_{ij}}{\triangle t\dot{\epsilon}'_{k\ell}\dot{\epsilon}'_{k\ell}} \qquad (2.13.5) \\
a^2 &= \frac{\hat{\lambda} + 2\hat{\mu}}{\rho} = \frac{3\hat{K} + 4\hat{\mu}}{3\rho}
\end{aligned}
$$

The increments in the strain, $\dot{\epsilon}_{ij}\triangle t$, must be large enough for the difference expressions in Eq. (2.13.5) to be accurate. When the volumetric or deviatoric strain increments are too small, or the calculated dilitational modulus is negative, different strategies may be used, but in most cases they resort to an initial estimate of $\lambda + 2\mu$.

In two dimensions, the form of the stability condition remains the same but the characteristic length of the element, $\triangle x$, is not uniquely defined. Hallquist uses the element area divided by the longest diagonal in DYNA2D, Margolin uses the characteristic length defined by Wilkins for the Wilkins shock viscosity [46] in SHALE [23], and Flanagan and Taylor divide the element area by $\sqrt{(d_{13}^2 + d_{24}^2)/2}$, where d_{ij} is the distance between nodes i and j. For rectangular elements the two diagonals are of equal length, resulting in DYNA2D and PRONTO calculating the same characteristic length. The PRONTO characteristic length is based on the bounds of the element eigenvalues and, therefore, is always conservative. Flanagan and Taylor rely on the conservative nature of the characteristic length to compensate for any possible overestimation of the time step that occurs due to the material sound speed estimate in Eq. (2.13.5). Margolin's characteristic length incorporates both the element geometry and the shock direction. In problems where the element aspect ratios are large, the characteristic length may vary greatly with small changes in the shock direction. The large swings in the characteristic length have been a problem with the Wilkins viscosity, and are a potential problem in calculating the time step size in Lagrangian codes. Since SHALE is an ALE (Arbitrary Lagrangian Eulerian) code, it can dynamically reduce the aspect ratio of a problem element by moving the mesh.

2.14 GODUNOV METHODS

Godunov methods are an interesting alternative to the standard finite element and finite difference methods for problems that are dominated by shocks. Most Godunov methods are designed to solve the Euler equations, i.e., they neglect the strength of the material (in fact, most specialize their work to ideal gas problems and emphasize aerodynamic applications). Only three groups have attempted to include strength [2], [3], [84], and their work is emphasized at the expense of neglecting the many other researchers who are concentrating on the Euler equations.

Godunov [11] was interested in eliminating the oscillations that occur near shocks in numerical solutions. The method he proposed is only first order accurate, but later researchers, e.g, [52], [86], have created higher order accurate algorithms based on his ideas. His method can be used in both a Lagrangian and Eulerian context, but in this section only the Lagrangian approach is considered.

2.14.1 Godunov methods in one dimension. To introduce the Godunov method, the discussion is restricted to the Euler equations in one planar dimension with a unit cross sectional area, and heat conduction and the body forces are neglected. The computational mesh for the Godunov method is similar to the ones used in the standard finite element and finite difference methods except that the velocities are centered in the elements with the density and the internal energy, and the nodes are used only to define the boundaries of the elements. In the literature on the Godunov method, the element centers are frequently given integer numbers, but the numbering scheme used in this section remains consistent with the previous sections and integers denote nodes.

The motion of the nodes in the Godunov method is determined by solving the Riemann problem at each node every time step. At node i, the initial left and right states of the Riemann problem are given by the values of the velocity, density, and energy in elements $i-1/2$ and $i+1/2$. The solution of the Riemann problem is obtained numerically using either an exact, iterative method or one of a number of approximate methods [2], [87]. The solution consists of three waves: the inner wave is the contact discontinuity, and node i moves with it, and the outer two waves are shock waves or rarefaction fans moving through the adjacent elements $i-1/2$ and $i+1/2$. The velocity of each node is set equal to its associated contact discontinuity velocity, u_i^*, and integrated to obtain the displacement of the node at t^{n+1}. In addition to the contact velocity, the pressure at the contact discontinuity, P_i^*, is calculated, and it is used with the velocity to update the internal energy and the element velocity. Note that the solution for the contact discontinuity is centered at time $n+1/2$. The work calculation uses the total energy, $T+E$, where T and E are the kinetic and internal energy of the element respectively.

$$\begin{aligned}
u^{n+1}_{i+1/2} &= u^n_{i+1/2} + \frac{\triangle t}{M_{i+1/2}}\left(P^*_{i-1/2} - P^*_{i+1/2}\right) \\
x^{n+1}_i &= x^n_i + \triangle t u^*_{i+1/2} \\
T^{n+1}_{i+1/2} + E^{n+1}_{i+1/2} &= T^n_{i+1/2} + E^n_{i+1/2} + P^*_{i-1/2}u^*_{i-1/2} - P^*_{i+1/2}u^*_{i+1/2} \\
\rho^{n+1}_{i+1/2} &= \frac{M_{i+1/2}}{x^{n+1}_{i+1} - x^{n+1}_i}
\end{aligned} \tag{2.14.1.1}$$

The time step is limited so that waves with positive velocities from any node i do not interact with waves with negative velocities from node $i+1$.

The original Godunov scheme is first order accurate even when the Riemann problem is solved exactly. Higher order methods are obtained by improving the approximation of the left and right states in the Riemann problem, e.g., [52], [86], [88]. The original Godunov scheme represents the solution as a piecewise constant function. A more accurate method is obtained if the solution is represented as a piecewise linear [52] or parabolic function [86]; see Fig. 20.

The numerical values of the solution variables associated with the elements represent element averages instead of amplitude of the piecewise constant function. Letting $q^n_{i+1/2}$ be the numerical value of one of the solution variables in element $i+1/2$, and $q_{i+1/2}(x)$ be the piecewise representation of q in element $i+1/2$, two constraints must be imposed on $q_{i+1/2}(x)$ to avoid spurious oscillations in the solution: First, the interpolation is conservative in the sense that the integral of the distribution must equal the element average. Second, the interpolation of q is monotonic. The values for q_{min} and q_{max} are usually taken to be the minimum and maximum values of the element averages of elements $i-3/2$, $i+1/2$, and $i+3/2$.

$$\begin{aligned}
\int_{x^n_i}^{x^n_{i+1}} q_{i+1/2}(x)dx &= q^n_{i+1/2}(x^n_{i+1} - x^n_i) \\
q_{min} \leq q_{i+1/2}(x) \leq q_{max} & \qquad \forall x \in [x^n_i, x^n_{i+1}]
\end{aligned} \tag{2.14.1.2}$$

The interpolation functions are nonlinear functions of the element averages, and therefore the higher order Godunov methods are nonlinear even for linear problems. A detailed discussion of how these functions are created is deferred until discussion on advection in the next chapter, but it should be noted that most of the nonlinearity in the algorithms is associated with imposing the monotonicity constraint. The basic strategy for the case of a linear interpolation function involves two steps: the first step is to calculate a second order approximation of the slope, and the second is the imposition of the monotonicity constraint by scaling the magnitude of the slopes in different zones on a selective basis.

2.14.2 Introducing strength into Godunov methods. The application of Godunov methods to materials with strength is relatively recent, and therefore the approaches are much less standardized than for the Euler equations. Hancock in PISCES [2] and Dukowicz et al. in the original version of CAVEAT [87] split the stress into the pressure and the deviatoric contributions. They use a second order Godunov method for the pressure, and a finite difference formulation for the deviatoric part of the stress. While their approaches are conceptually similar, there are numerous differences in the implementation. As an alternative to the differencing scheme for the strength, CAVEAT can solve two independent Riemann problems for the normal and transverse stresses and velocities on the edge of a cell [173]. Trangenstein and Colella [84] developed a second order Godunov method by constructing an approximate Riemann solver for a hypoelastic material. In contrast to most, they use a total Lagrangian formulation. The algorithm they describe in the literature [84], [89] is one-dimensional, but a two-dimensional version does exist based a two-dimensional method for general systems of hyperbolic equations developed by Colella [90].

The algorithm developed by Trangenstein and Colella consists of eight steps:

1. A characteristic analysis and time step estimation.
2. A monotonized slope computation.
3. The tracing of the characteristics.
4. The flux calculation.
5. Conservative differencing.
6. A rotational update.
7. Stress and strain rate calculations.
8. The stress update.

At the beginning of the time step, the cell-centered values of the velocities, stress, rotation, and deformation gradient are known.

A characteristic analysis is performed on the quasilinear forms of the mass, momentum, and energy equations. The primitive variables in the analysis are the velocity, internal energy, and the stress. Deflation is used to reduce the characteristic analysis to finding the eigenvalues and eigenvectors of the acoustic tensor. The acoustic tensor must be derived for each material model, although Trangenstein and Colella have simplified the task by deriving its general form for the general class of hypoelastic plasticity models with yield surfaces. After the eigenvalue problem is solved for the acoustic tensor, the characteristic directions, X, and speeds, Λ, are calculated for the full system of equations. The time step is based on the fastest elastic wavespeed.

The second step requires the calculation of the spatial derivatives of the primitive variables. Trangenstein and Colella calculate the derivatives in terms of the characteristic directions. Monotonicity is enforced by using the flux-limiting procedure developed by van Leer [52], which is discussed in detail later. The enforcement of the monotonicity condition introduces the numerical equivalent of the shock viscosity used in the standard finite difference formulations. By evaluating the derivatives in the directions of the characteristics, the viscosity is added only to those waves that have spurious oscillations, which minimizes the smearing of shock waves and contact discontinuities. A central difference approximation, Eq. (2.14.2.1), is used for the second order derivative in the van Leer algorithm. The vector of primitive variables are $\phi_{j+1/2}$, where subscript denotes the cell. In a similar manner, $[X_{j+1/2}]$ is the matrix of eigenvectors for the $j+1/2$th zone, and $c_{j+1/2}$ is the vector of central differences in the characteristic directions.

$$c_{j+1/2} = [X_{j+1/2}]^{-1}(\phi_{j+3/2} - \phi_{j-1/2})/(2\triangle x) \tag{2.14.2.1}$$

Trangenstein and Colella have found it necessary to selectively introduce extra viscosity into the solution by reducing the derivatives to zero in a zone when the loading conditions (with respect to the yield surface) in the adjacent zones are not the same.

The left and right states at each node for the Riemann problem are calculated by using the slopes from Eq. (2.14.2.1), the characteristic directions and the speeds.

$$\begin{aligned} \phi^{n+1/2}_{j+1} &= \phi^n_{j+1/2} + \tfrac{1}{2}\frac{\partial \phi}{\partial x}\triangle x + \tfrac{1}{2}\frac{\partial \phi}{\partial t}\triangle t \\ &= \phi^n_{j+1/2} + \tfrac{1}{2}[X_{j+1/2}]\{I\triangle x - \Lambda\triangle t\}c_{j+1/2} \end{aligned} \tag{2.14.2.2}$$

Upwinding is achieved by using only the characteristics with a negative velocity to construct the state at the left edge of each element (the right state for each node, Φ^R_j), and similarly, the characteristics with a positive velocity, the state at the right edge, Φ^L_{j+1}.

An approximate Riemann solver, which is sufficient for weak waves, is used to calculate the stationary state. For strong shocks their approximate Riemann solver produces oscillations and a more viscous scheme must be used. First, the jump across the node is expressed in terms of the upwind eigenvectors of each zone, $[\tilde{X}_{j+1/2}]$, resulting in a set of linear equations for the vector $\{a^+, a^-\}$.

$$[\tilde{X}_{j+1/2}]a^+ + [\tilde{X}_{j-1/2}]a^- = \Phi_j^R - \Phi_j^L \tag{2.14.2.3}$$

The stationary state can be expressed in terms of either the left or right values, and the average is used for the numerical flux.

$$\begin{aligned} f_j^\phi &= \tfrac{1}{2}(f_j^L + f_j^R) \\ f_j^L &= \Phi_j^L + [\tilde{X}_{j-1/2}]a^- \\ f_j^R &= \Phi_j^R - [\tilde{X}_{j+1/2}]a^+ \end{aligned} \tag{2.14.2.4}$$

The fluxes in Eq. (2.14.2.4) are used to update the solution variables in a conservative manner.

The spin and deformation rate are calculated at $t^{n+1/2}$ and the Hughes-Winget formula [40] is used to integrate the rotation matrix. After the stress is updated to $t^{n+1/2}$ using an objective stress rate, the constitutive equation is evaluated, and then the stress is given the additional rotational increment to bring it to t^{n+1}.

Most of the algorithms for integrating constitutive equations are only first order accurate, but they are usually constructed so that they are asymtotically correct for large increments in the strain. In order to maintain the second order accuracy in time, Trangenstein and Colella have developed their own integration algorithms for J_2 plasticity [84] and the cap model for soils [90]. Trangenstein drops from a second order algorithm to a first order algorithm for the cap model when a shock is detected [90].

2.14.3 Two-dimensional Godunov methods with strength. PISCES [2], a two-dimensional coupled Eulerian-Lagrangian code, was one of the first production codes to implement van Leer's second order accurate Godunov scheme. The Godunov method is applied only to the pressure terms, and the deviatoric terms are handled by a finite difference scheme. This separation makes the implementation of new material models in PISCES as simple as in any finite difference or finite element code while the advantages of the Godunov method for strong shocks are retained. The Godunov method is used only in the Lagrangian step in the Eulerian code, while the purely Lagrangian calculations are performed using a finite difference algorithm similar to the one in HEMP.

CAVEAT [3] is a production ALE code with strength while CAVEAT-GT [91] is a research-oriented, free-Lagrange code without strength. Both use a Godunov method for the pressure terms, and the original strength implementation in CAVEAT, with a philosophy similar to PISCES, uses a finite difference method for the strength terms. CAVEAT uses quadrilaterals, while CAVEAT-GT uses triangles. The general flow of all three programs is similar within the Lagrangian Godunov step:

1. A monotonized slope computation of the solution variables.
2. The “left” and “right” states for the Riemann problem are calculated for each zone edge.
3. The Riemann problem in the direction normal to the edge is solved using an approximate Riemann solver for strong shocks.
4. Stress and strain rate calculations at the edges of each zone (original CAVEAT and PISCES only).
5. The motion of the mesh is calculated.
6. The stress, density, energy, and history variables are updated within each zone.

The details associated with the strength calculations (step 4) and the mesh motion (step 5) are the major points of difference. The first two steps are similar to steps 2 and 3 of the previous section, but in contrast to the work of Trangenstein and Colella, the slopes for the linear distributions are calculated in the spatial coordinate system. An alternative to step 4 in CAVEAT is the solution of the transerve Riemann problem [173], which is described below.

One of the underlying assumptions in all Godunov codes is that the shock direction is normal to the cell edge. Clearly this is not true, but numerical experiments have been conducted by Dukowicz and Melz at Los Almos [174], and their results indicate that the numerical solutions are not very sensitive to the mesh orientation relative to the shock.

The Riemann solvers in PISCES [2] and CAVEAT [3] assume that the shock Hugoniot can be approximated by a quadratic relationship between the pressure and the velocity, where $*$ indicates the contact discontinuity and s indicates either the left or right state. The wave velocity is approximated with the same linear relation, Eq. (2.8.4.3), that was used in the shock viscosity.

$$\begin{aligned} C_s &= a_s + A_s|u^* - u_s| \\ P^* - P_s &= \rho_s C_s (u^* - u_s) \end{aligned} \tag{2.14.3.1}$$

The isentropic speed of sound is a, and the strong shock parameter, A is the strong shock limit of $(\rho_2/\rho_1)/(\rho_2/\rho_1 - 1)$. The Riemann solver calculates P^* and u^* from the intersections of the shock Hugoniots defined by Eq. (2.14.3.1) for the left and right states. For materials that have a linear relationship between the particle velocity and the shock speed, the relation given by Eq. (2.14.3.1) is exact. The major difference between the Riemann solvers in PISCES and CAVEAT (and CAVEAT-GT) is the extension of the Hugoniot for rarefactions. Considering a rightward wave, Hancock sets A_R to zero when $u^* - u_R$ is negative while Dukowicz continues it with a quadratic, Eq. (2.14.3.2), that maintains a positive slope. Similar strategies are used for the left state.

$$\begin{aligned} P^* - P_R &= \rho_R A_R |u^* - u^*_{min}|(u^* - u^*_{min}) \\ u^*_{min} &= u_R - \frac{a_r}{2A_R} \end{aligned} \tag{2.14.3.2}$$

When the transverse Riemann problem is solved in CAVEAT, the transverse wave speed, a_T, is assumed to be independent of the velocity jump. This assumption reduces the Riemann problem to two linear equations, where v^* is the velocity transverse to the edge, and τ^* is the shear stress. When this option is chosen, the coefficients for the wave speed in Eq. (2.14.3.1) are modified appropriately to account for the material strength, and P in Eq. (2.14.3.2) is replaced by the normal stress, σ.

$$\begin{aligned} v^* &= \frac{\tau_R - \tau_L + (\rho^* a_T)_R v_R + (\rho^* a_T)_L v_L}{(\rho^* a_T)_R + (\rho^* a_T)_L} \\ \tau^* &= \frac{(\rho^* a_T)_R (\rho^* a_T)_L (v_R - v_L) + (\rho^* a_T)_R \tau_L (\rho^* a_T)_L \tau_R}{(\rho^* a_T)_R + (\rho^* a_T)_L} \end{aligned} \tag{2.14.3.3}$$

The stress and strain calculations at the zone edges differ significantly in PISCES and CAVEAT (referring now to the original difference implementation), the major difference being that PISCES evaluates the stresses at the zone centers in addition to the zone edges. The strain rates and the spin are evaluated at a zone edge in both codes by differencing the velocities of the two adjacent zone centers and the two vertices defining the edge. The deviatoric stresses at the beginning of time step n in PISCES are the density-weighted average of the left and right values calculated by using the linear van Leer approximations. While the material properties, φ, (e.g., the yield stress) are also density weighted, note that the density weighting is different from the weighting used for the stresses.

$$\begin{aligned} \sigma^n_{ij} &= \frac{\rho_R \sigma^n_{L,ij} + \rho_L \sigma^n_{R,ij}}{\rho_R + \rho_L} \\ \varphi^n_{ij} &= \frac{\rho_L \varphi^n_L + \rho_L \varphi^n_L}{\rho_R + \rho_L} \end{aligned} \tag{2.14.3.4}$$

Associated with each zone $(i+1/2, j+1/2)$ in the original CAVEAT difference formulation are the deviatoric stresses and their history variables centered at the edges $(i+1, j+1/2)$ and $(i+1/2, j+1)$. Their values at the beginning of a time step are simply the values they had at the end of the previous time step (assuming the calculation is purely Lagrangian). Based on the values of the strain rates, the spin, the deviatoric stresses, and the history variables, the stresses at the edges are integrated in the same manner as in a standard Lagrangian code. The total stress at an edge is equal to $\sigma'^{n+1} - P^*$. The element-centered solution variables are updated using the boundary integral form of the governing equations, Eq. (1.1.3.1) in a difference form that is directly analagous to Eq. (2.14.1.1). PISCES uses the contact velocity uniformly in the boundary integrals, while CAVEAT uses the velocities of the vertices in the energy calculation.

2.14.4 Godunov methods in cylindrical and spherical geometries. The extension of the Godunov method to two and three-dimensional problems in Cartesian coordinates is straight forward, but there are two additional issues associated with the cylindrical and spherical geometries. The first issue, highlighted by Roe [93], is how the Riemann problem should be solved since the shock is no longer planar and he has proposed a modified Riemann solver for the cylindrical geometry. Both CAVEAT and PISCES use the same Riemann solver for both planar and cylindrical geometries on the basis that for a short period of time, at a particular point in space and time, the behavior of the contact discontinuity is the same for both geometries. The authors of both codes have carefully compared the analytical and the numerical solutions of cylindrical test problems to demonstrate the validity of their approximation.

Hancock initially obtained poor answers when he applied a first order Godunov method to a cylindrical or spherical problem in one dimension [94]. He tested his Godunov code by coasting a single element towards the centerline and he found that very large pressures were generated. On examining the results of his calculation, he determined that the cause of his problem was the manner in which he calculated the velocities at the edges. For the one element problem, a first order method assigns the average element velocity to both the left and right edges. These boundary conditions are inconsistent with the isochoric motion of the exact solution. To overcome this difficulty he interpolated the velocity within an element on the basis that the fluid motion within the element is isochoric. Letting α equal 1, 2 or 3 to indicate planar, cylindrical, or spherical geometry respectively, the velocity interpolation is given by Eq. (2.14.4.1).

$$\dot{r} = \left(\frac{r_{i+1/2}}{r}\right)^{\alpha-1} \dot{r}_{i+1/2} \tag{2.14.4.1}$$

To put it another way, he assumes that the volume flux, not the velocity, is constant within the element. For his second order Godunov scheme, he interpolates the volume flux, $\dot{r}r$, with a monotonic linear function instead of the velocity.

2.14.5 Updating the vertex velocities. The velocities of the vertices, or nodes, are needed in Lagrangian calculations to update the location of the mesh, and they may be used in an ALE or Eulerian calculations to determine the transport volumes between elements. Since the velocities are not available directly, they must be calculated from the edge velocities.

The default algorithm in CAVEAT [3] is a least squares fit, which is both simple and robust. When only the velocity normal to the cell edges is calculated, the error between the normal velocity and the projection of the vertex velocity is minimized in the least squares sense. The resulting functional, Eq. (2.14.5.1), gives two linear equations, Eq. (2.14.5.2), where n_α is the normal vector, u^* is the velocity for edge α, and u is the unknown vertex velocity. The weighting factor, W_α, is the sum of the material densities on either side of the edge.

$$\min \tfrac{1}{2} \sum_\alpha W_\alpha (u \cdot n_\alpha - u^*_\alpha)^2 \tag{2.14.5.1}$$

$$\sum_\alpha W_\alpha n_{\alpha i}(u_j n_{\alpha j} - u^*_\alpha) = 0 \quad i = 1, 2 \tag{2.14.5.2}$$

One limitation of this approach is it may generate spurious velocities transverse to the flow if the mesh is distorted, e.g., the Saltzman shock tube problem. Dukowicz and Melz have interpreted these velocities in terms of spurious vorticities [174] and Benson, in terms of spurious shears [42] on a distorted mesh.

When the velocity tangent to the edge is available through solving the transverse Riemann problem, the contact velocity vector can be rotated into the global coordinate system and Eq. (2.14.5.1) and Eq. (2.14.5.2) are generalized.

$$\min \tfrac{1}{2} \sum_{\alpha i} W_\alpha (u_i - u^*_{\alpha i})^2 \tag{2.14.5.3}$$

$$\sum_\alpha W_\alpha(u_i - u^*_{\alpha i}) = 0 \quad i = 1,2 \tag{2.14.5.4}$$

The result in this case is a simple weighted average.

$$u_i = \frac{\sum_\alpha W_\alpha u^*_{\alpha i}}{\sum_\alpha W_\alpha} \quad i = 1,2 \tag{2.14.5.5}$$

Dukowicz and Meltz recently developed a new method [174] for calculating the vertex velocities that eliminates the spurious vorticities by directly integrating the vorticity equation. In their work, they consider a couple of different ways to use the vorticity equation, but only the basic strategy is summarized here. The velocity of a vertex is defined in terms of two potentials, ϕ and A.

$$\begin{aligned} u &= \nabla\phi + \nabla \times A \\ \nabla^2\phi &= \nabla \cdot u \\ \nabla^2 A &= -\nabla \times u \end{aligned} \tag{2.14.5.6}$$

The divergence of the velocity field is known from u^*. While the curl of the velocity is also known in principle, it is integrated from the vorticity equation, where ω is the vorticity vector. In two dimensions, ω has only the nonzero component out of the plane. The numerical integration procedure for Eq. (2.14.5.7) is designed to diffuse the vorticity, thereby reducing the distortion locally.

$$\frac{d\omega}{dt} + \omega\nabla \cdot u = \nabla P \times \nabla\frac{1}{\rho} \tag{2.14.5.7}$$

The two Laplace's equations in Eq. (2.14.5.6) are discretized using a finite difference stencil and the resulting discrete equations are solved with the conjugate gradient method. Despite the cost of solving the equations, the method is practical and the quality of the velocity solution is much better than the least squares fit.

2.15 TVD METHODS

The standard approach to increasing the order of accuracy of the Godunov methods it to construct higher order, conservative, monotonic interpolation functions to determine the left and right states. The combination of restrictions associated with choosing a polynomial interpolation function (the invariable choice), and imposing the conservation and monotonicity constraints greatly limits the number of free parameters that can be tuned to optimize the resolution of shock and contact discontinuities. In addition, the solution of the Riemann problem can be quite difficult for real materials. One approach to circumventing these problems is the total variation diminishing (TVD) methods developed for solving the Euler equations. Their formulation for pure advection is discussed in greater detail in the next chapter, but the essential idea is fairly simple.

First order methods for the Euler equations tend to be montonic, but diffusive, while second order methods are accurate at the expense of having oscillations in the solution. The idea of a TVD limiter, ϕ, is to introduce into the solution the minimum amount of damping that is necessary to eliminate the oscillations by using a linear combination of first and second order methods. On a conceptual level, this idea is illustrated by Eq. (2.15.1), where S is a state variable and $\mathcal{O}$ indicates the order of the method. Note that TVD methods limit the flux, while Godunov methods limit the data, to enforce monotonicity.

$$S^{n+1} = S^n + \phi \cdot \mathcal{O}(\triangle x) + (1-\phi) \cdot \mathcal{O}(\triangle x^2) \tag{2.15.1}$$

Kashiwa and Lee [175] have replaced the Riemann solution with a TVD scheme in a version of CAVEAT [138]. Their update of the cell-centered quantities is the same as in the Godunov method. Since TVD schemes are formulated in terms of limiting fluxes, the appropriate fluxes for the velocity and the pressure must be defined. Kashiwa and Lee use the momentum and energy equations.

$$\begin{aligned} \frac{du}{dt} &= -1/\rho \nabla P \\ \frac{dP}{dt} &= -\rho c^2 \nabla \cdot u \end{aligned} \tag{2.15.2}$$

The first order fluxes are derived from the same assumptions as the Riemann solver developed by Dukowicz [87], while the second order fluxes are derived by applying the Lax-Wendroff scheme [176] to Eq. (2.15.2). In contrast to the second order Godunov methods, the subscripts + and $-$ denote cell-centered values and not the values extrapolated to the edge of the cell.

$$\begin{aligned} u^* &= u_0^* - \phi \left[\frac{\triangle x \nabla P}{(\rho C)_+ + (\rho C)_-} \right] - (1-\phi) \left[\triangle t \frac{\nabla P}{2\rho} \right] \\ P^* &= P_0^* - \phi \left[\frac{(\rho C)_+ (\rho C)_- \triangle x \nabla \cdot u}{(\rho C)_+ + (\rho C)_-} \right] - (1-\phi) \left[\triangle t \frac{\rho C^2 \nabla \cdot u}{2} \right] \end{aligned} \tag{2.15.3}$$

The initial values u_0^* and $P{*}_0$ are calculated by using the leading terms from a method of characteristic solution.

$$\begin{aligned} u_0^* &= \frac{(\rho C)_- u_- + (\rho C)_+ u_+}{(\rho C)_+ + (\rho C)_-} \\ P_0^* &= \frac{(\rho C)_- P_+ + (\rho C)_+ P_-}{(\rho C)_+ + (\rho C)_-} \end{aligned} \tag{2.15.4}$$

The limiter is based on work by Davis [178], and it is only turned on when the divergence is negative so the first order terms are a direct analog to a shock viscosity.

$$\phi = \left\{ \begin{matrix} \phi' & \text{if } D_{ii} \leq 0 \\ 0 & \text{otherwise} \end{matrix} \right\} \tag{2.15.5}$$

$$\phi' = 1 - \max[0, \min(D_{ii}/D_{jj-}, D_{ii}/D_{jj_+}, D_{ii-}/D_{jj}, D_{ii+}/D_{jj})] \tag{2.15.6}$$

The Godunov and TVD schemes in CAVEAT have the same shock resolution in one dimension, but the TVD scheme is better at propagating shocks that are oblique to the mesh in two dimensions, e.g., spherical blast calculations on a rectangular mesh. The superior isotropy of the TVD scheme relative to the Godunov scheme is probably due to using the velocity divergence, which is an isotropic indicator of shocks, instead of the one-dimensional velocity jumps in the limiter.

2.16 PARTICLE METHODS

Particle methods can be characterized as methods where the solution variables are attributed to Lagrangian point masses instead of computational cells. There have been many different variations on this theme, and the interested reader is referred to Ref. [179] for a collection of excellent papers on recent advances in the area. The advantages of particle methods are there is no need to track the material interfaces, and the calculation will continue to run regardless of the amount of turbulence in the solution. The difficulties are ringing of the particles, fluctuations from transporting discrete particles, resolving thin layers of material, and a lack of translational and rotational invariance [180]. Recent work has eliminated or reduced these difficulties, but the shock resolution of particle methods is inferior to the best finite difference and finite element methods. Among the best formulations today are FLIP (Fluid-Implicit Particle) [181] and SPH (Smooth Particle Hydrodynamics) [182].

Some of the particle formulations consist of a standard finite difference Lagrange step followed by an advection step that uses particles to resolve material interfaces in cells with more than one material. The discussion of particle advection is deferred to later.

2.17 FREE LAGRANGE AND ADAPTIVE METHODS

Free Lagrange methods overcome the difficulties associated with turbulent flow by automatically changing the definition of the mesh as it becomes distorted [10], [179]. The most successful free Lagrange codes, such as CAVEAT-GT [x], use the same finite difference or finite element Lagrange steps as conventional Lagrangian codes. The main challenge in these formulations is to create adaptive mesh algorithms that are robust, efficient, and which produce good meshes. These algorithms fall into two main classes, and both may exist within a single code [91]: 1) nearly Lagrangian mesh updates, which adjust the mesh locally, and 2) complete mesh regenerators, which are frequently based on the Vornoi diagrahm [10]. While the algorithms are quite simple conceptually, a presentation with enough detail to appreciate them is beyond the scope of this paper, and the interested reader is referred to Refs. [10], [91], and [179]. The transport algorithms, which are also used by Eulerian and ALE codes, are considered in detail later.

Chapter 3

ALGORITHMS FOR EULERIAN AND ARBITRARY LAGRANGIAN EULERIAN CALCULATIONS

Hydrocodes are frequently applied to problems that ivolve deformations that are too severe to be handled by the same Lagrangian mesh during the entire calculation. These problems involve large stretchs, resulting in large changes in the aspect ratios of elements, turbulent flow, which may invert elements, and material failure, which results in the generation of new free surfaces. At some point in the calculation, a new mesh must be generated and the old solution must be mapped from the old mesh on to the new mesh. The frequency and the restrictions placed on the generality of the remap strategy define the differences between Eulerian, arbitrary Lagrangian-Eulerian (ALE), and rezoned meshes. Rezoning requires the intervention of the user, either through a graphical interface or through a command file to define the new mesh, and the success of the method depends heavily on the skill and patience of the user. Problems which involve a change in the topology (e.g., a penetration calculation) during the calculation are especially difficult for manual rezoning since a smooth transition from one topology to another may require time step by time step intervention.

Eulerian methods were at one time regarded as a last resort for solving a problem because of their poor resolution of material interfaces. This has changed, however, with the introduction of high resolution interface tracking algorithms. The best interface tracking algorithms are now also used in the graphics postprocessor. An example of the accuracy of the interface trackers is shown in Fig. 21, a MESA calculation of the shape of an explosively formed projectile.

The advection algorithms, which are used to calculate the material transport between elements, are second order accurate and they are no longer the limiting factor in the accuracy of Eulerian calculations. Furthermore, Eulerian and ALE calculations do not have the accuracy losses associated with highly distorted elements that plagues Lagrangian calculations. The cost of the calculations has also been reduced considerably by performing the Eulerian remap only after several Lagrangian steps instead of every step. For problems involving large deformations (and consequently, distorted Lagrangian meshes), the accuracy of an Eulerian calculation is equal to a Lagrangian calculation, and it may be significantly better for problems that have a changing topology.

The mesh used in Eulerian hydrocodes is fixed in space. In addition to accounting for the changes in the element solution due to source or Lagrangian terms, Eulerian codes must account for the transport of material through the mesh. There are two different schools of thought on how Eulerian codes should be structured: One approach updates the solution variables in a single step. An early example of this approach is the HELP code [95], and much of the recent research on solving the Euler equations has emphasized this approach, e.g., [96], [97], [98]. The other, used by Noh in CEL [9], separates the Lagrangian and Eulerian terms into two steps. A Lagrangian step is performed first which allows the mesh to follow the material and distort. The solution on the distorted mesh is mapped back onto the original Eulerian (spatially fixed) mesh in a separate step that is referred to as the “remap”, “Eulerian”, or “advection” step. This strategy is often referred to as an “operator split” because the governing equations are solved by solving a sequence of problems, each one associated with just one of the differential operators in the original equations. Chorin et al. [99] discuss the advantages and disadvantages of this strategy in the context of a wide variety of physical problems.

If the solution is mapped onto a different mesh that moves in a manner that may be independent of the material motion, the method is referred to as an “arbitrary Lagrangian-Eulerian” (ALE) method [24]. The algorithms for the remap step for Eulerian and ALE codes are identical. Eulerian codes are often faster on a cpu per element basis because their spatially fixed Eulerian mesh. Since the geometry of the Eulerian mesh is fixed, the geometric calculations associated with it can be performed once and stored. Additional simplifications are available if the mesh is rectangular. When edge-centered velocities are used, the Lagrange step is very simple, especially if strength is ignored. In addition, the remap step is cheaper because only half as many velocity components are remapped in comparison with formulations using node-centered velocities. This last advantage disappears when the comparison is made with Godunov schemes, which center the velocities in the elements.

Virtually all Eulerian hydrocodes, including CEL [98], JOY [100], HULL, PISCES [2], CSQ [62], CTH [63], MESA [101], and KRAKEN [102] have separate Lagrangian and remap steps. Most ALE codes, including CALE, CAVEAT [3], DYNA2D [103], HEMP [104], SALE [24], and SALE3D [25] also perform two steps. HELP, as mentioned before, does not, nor does the recent ALE research performed by Liu et al. [105]. The primary reason for the popularity of the operator split is the simplicity in the centering of the solution variables for obtaining second order accuracy in an explicit calculation. When unsplit Eulerian algorithms are updating the solution at a particular spatial location, they use information that is upstream of the flow. The upstream location is dependent on the direction and magnitude of the flow. For instance, to update the stress, an upstream or "upwinded" strain rate is used. The flow velocity is, however, dependent on the stress, resulting in a coupling between the location of the upwinded strain rate and the updated stress. This coupling makes unsplit Eulerian algorithms appear to be naturally implicit. Second order accurate methods for the Euler equations have, however, been constructed, e.g., [96], [97], [106].

Both ALE and Eulerian formulations may have elements containing more than one material, while virtually all Lagrangian formulations assume that each element is restricted to a single material. The complexity of handling elements with several materials, or "mixed elements" (which are not to be confused with elements that are based on mixed variational principles), adds a significant computational cost. Some ALE codes are restricted to single materials within an element, and they are referred to as "simplified ALE" codes. The drawback of a simplified formulation is the range of problems a simplified ALE code may address is not much greater than a pure Lagrange code since the material boundaries remain Lagrangian in both cases.

In the remainder of this chapter the operator split formulations are emphasized because of their popularity. Since the Lagrangian step is discussed in Chapter 2, this chapter focuses on the transport of material between elements. One dimensional methods are discussed first, and then their extensions to two dimensions. A separate section discusses the special difficulties associated with the transport of momentum. Much of the computational cost in hydrocodes is associated with the treatment of elements containing more than one material. The algorithms associated with the stress updates and the interface tracking are discussed. General remapping techniques are considered at the end of the chapter.

Methods in Lagrangian and Eulerian Hydrocodes

3.1 THE OPERATOR SPLIT

Operator splitting is convenient method for breaking complicated problems into a series of less complicated problems [107]. The separation of the Lagrangian and Eulerian parts of an Eulerian calculation is one application of an operator split. It is also frequently used to construct multidimensional advection algorithms from one-dimensional algorithms [98]. Although it is not usually thought of in the operator split context, the radial return algorithm [6], [7], [8] used in most finite element programs to integrate the J_2 plasticity equations may also be viewed as an operator split [108].

The linear advection equation with a source provides a convenient way to illustrate how the operator split is used in Eulerian codes.

$$\frac{\partial \phi}{\partial t} + c\frac{\partial \phi}{\partial x} = f \qquad\qquad \phi(x,0) = \phi_0(x) \tag{3.1.1}$$

The field variable is ϕ, c is the constant flow velocity, and f is the source. The operator split divides Eq. (3.1.1) into two equations that are solved sequentially. The first equation, the "Lagrangian" equation, contains the source term, and the second one, the "Eulerian" equation, contains the convective term.

$$\begin{aligned} \frac{\partial \phi}{\partial t} &= f \\ \frac{\partial \phi}{\partial t} + c\frac{\partial \phi}{\partial x} &= 0 \end{aligned} \tag{3.1.2}$$

In an Eulerian hydrocode, the first equation is solved with the same algorithms that are used in a Lagrangian code. The mesh moves with the material during this step. Codes that use node-centered velocities can usually be run in a purely Lagrangian mode so that their Eulerian and Lagrangian solutions can be directly compared. To solve the second equation, the mesh is moved from its current position to its original spatial position, and the volume of material transported between adjacent elements is calculated. The transport of the mass, energy, momentum, stress, and other field quantities is handled by the advection algorithms.

The computational flow of an Eulerian calculation based on an operator split proceeds in the following stages:

1. The solution is advanced in time with a Lagrangian step.
2. The remap or Eulerian step maps the solution from the Lagrangian mesh onto the new mesh in a series of substeps.

a. A new mesh is generated. The usual strategies are either to relax the Lagrangian mesh [109] in an ALE calculation, or to use a mesh fixed in space in an Eulerian calculation.

b. The element-centered solution variables are remapped with the advection algorithms. If necessary, the stresses are returned to the yield surface [110].

c. The node or edge-centered momenta are remapped. The momentum is used instead of the velocity to guarantee the conservation of momentum, and the velocities are calculated by dividing the values of the momentum at the edges or nodes by the associated masses. The momentum is advected after the element-centered mass advection because the new masses are needed to calculate the new velocities from the momentum.

The primary disadvantage of the operator split strategy is the limitation on accuracy. It is derived from an error analysis [107], which is presented here in a simplified manner. Consider a set of simultaneous, first order differential equations, with x being a vector of dependent variables, t the independent variable, and A and B are constant matrices.

$$\dot{x} = (A+B)x \qquad \text{solution : } \; x(t) = e^{(A+B)t}x(0) \tag{3.1.3}$$

An operator split sequentially solves the following equations each time step:

$$\begin{aligned} \dot{x} &= Ax \\ \dot{x} &= Bx \end{aligned} \tag{3.1.4}$$

Considering only a single time step, the operator split solution is given by Eq. (3.1.5).

$$x(\triangle t) = e^{B\triangle t}e^{A\triangle t}x(0) \tag{3.1.5}$$

A Taylor expansion is used to compare the solution in Eq. (3.1.5) to the exact solution in Eq. (3.1.3). The identity matrix is denoted 1.

$$\begin{aligned} e^{(A+B)\triangle t} &= 1 + (A+B)\triangle t + \tfrac{1}{2}(A^2 + AB + BA + B^2)\triangle t^2 \ldots \\ e^{B\triangle t}e^{A\triangle t} &= 1 + (A+B)\triangle t + \tfrac{1}{2}(A^2 + 2BA + B^2)\triangle t^2 \ldots \end{aligned} \tag{3.1.6}$$

The difference between $2BA$ and $AB + BA$ in the $\triangle t^2$ term means that the solution in Eq. (3.1.5) is only first order accurate. In the present context, A is the Lagrangian differential operator, and B is the Eulerian differential operator, or A and B represent k

and ℓ sweeps of the one-dimensional advection algorithm. In either case, both operators are nonlinear, and the above arguments are repeated using a linearization of the nonlinear equations about a particular solution point.

A second order accurate algorithm is generated by reversing the order of the operators every half time step.

$$\begin{aligned}
\dot{x} &= Ax \quad 0 \le t < \triangle t/2 \\
\dot{x} &= Bx \quad 0 \le t < \triangle t/2 \\
\dot{x} &= Bx \quad \triangle t/2 \le t < \triangle t \\
\dot{x} &= Ax \quad \triangle t/2 \le t < \triangle t \\
x &= e^{A\triangle t/2}e^{B\triangle t/2}e^{B\triangle t/2}e^{A\triangle t/2}x(0)
\end{aligned} \tag{3.1.7}$$

Two points need to be made. First, although the operator split notation provides a convenient format for discussing the Lagrangian and Eulerian steps, they don't represent a true operator split. There is no time step associated with the Eulerian step – it is simply a projection of the solution from one mesh onto another. The Eulerian step may be thought of as a continuous rezoning of the mesh. Time evolves only during the Lagrangian step and the Lagrangian step is rarely fully second order accurate in time. Most material models are integrated in time with a first order method such as the radial return algorithm [8]. Second, the advection algorithms are usually implemented in a second order accurate manner. The one-dimensional sweeps are spatially second order accurate, and the sequence of the directional sweeps is reversed after every complete sweep so that Eq. (3.1.7) applies. In contrast, the constant stress elements in the Lagrangian step are not spatially second order accurate if the elements are distorted.

3.2 UNSPLIT FINITE ELEMENT METHODS

3.2.1 The weak form of the ALE equations. Several finite element methods have been developed from the weak form of the ALE equations [35], [36], [105], [183]. These methods do not use an operator split to separate the Lagrangian and Eulerian terms, nor do they use it to generate multidimensional advection algorithms from one-dimensional algorithms. Although unsplit methods have a theoretical accuracy advantage over operator split methods, the advantage remains largely theoretical.

The mass, momentum, and energy weak forms are given by Eq. (3.2.1), and correspond to Eq. (1.2.3).

$$\int_\Omega \frac{\partial \rho^r}{\partial t}\delta\rho d\Omega = -\int_\Omega \rho\frac{\partial u_i}{\partial x_i}\delta\rho d\Omega - \int_\Omega w_i\frac{\partial \rho}{\partial x_i}\delta\rho d\Omega \tag{3.2.1a}$$

$$\begin{aligned}\int_\Omega \rho\frac{\partial u^r}{\partial t}\delta u_i d\Omega = &\int_\Omega \sigma_{ij}\frac{\partial \delta u_i}{\partial x_j} + \rho b_i\delta u_i d\Omega\\ &-\int_\Omega \rho w_j\frac{\partial u_i}{\partial x_j}\delta u_i d\Omega + \int_{\Gamma_\tau}\tau_i\delta u_i d\Gamma\end{aligned} \tag{3.2.1.b}$$

$$\begin{aligned}\int_\Omega \rho\frac{\partial e^r}{\partial t}\delta e d\Omega = &\int_\Omega(\sigma_{ij}\frac{\partial u_i}{\partial x_j} + \rho b_i u_i)\delta e d\Omega\\ &-\int_\Omega \rho w_j\frac{\partial e}{\partial x_j}\delta e d\Omega\end{aligned} \tag{3.2.1.c}$$

The greatest divergence in the formulations occurs in the advection of the element-centered variables. For constant stress elements, the spatial gradient of the element-centered variables is zero on the element interior and undefined at the element boundaries. The most attractive method for circumventing this difficulty was proposed by Liu, Belytschko, and Chang [184]. Their weak form for the evolution of the element-centered variables is presented here, where the auxilary variable, y_{ijk}, is defined to be the stress-velocity product, $\sigma_{ij}w_k$. Similar auxilary variables are necessary for all the history variables.

The strong form of the stress rate is given by Eq. (3.2.2), where for generality, the rotational terms and the material tangent matrix are added together and the rate is expressed as a function of the velocity gradient. Similar equations hold for the evolution of the other element-centered variables.

$$\frac{\partial \sigma^r_{ij}}{\partial t} = C_{ijk\ell}\frac{\partial u_k}{\partial x_\ell} - w_k\frac{\partial \sigma_{ij}}{\partial x_k} \tag{3.2.2}$$

Introducing the stress-velocity product results in an equivalent strong form, Eq. (3.2.3.

$$\frac{\partial \sigma^r_{ij}}{\partial t} = C_{ijk\ell}\frac{\partial u_k}{\partial x_\ell} - y_{ijk},_k + \sigma_{ij}\frac{\partial w_k}{\partial x_k} \tag{3.2.3}$$

The weak form of Eq. (3.2.3) is used in conjunction with Eq (3.2.4), which enforces the definition of y_{ijk} in the weak sense.

$$\int_\Omega \frac{\partial \sigma^r_{ij}}{\partial t}\delta\sigma_{ij}d\Omega = \int_\Omega(C_{ijk\ell}\frac{\partial u_k}{\partial x_\ell} - y_{ijk},_k + \sigma_{ij}\frac{\partial w_k}{\partial x_k})\delta\sigma_{ij}d\Omega \tag{3.2.4}$$

$$\int_{\Omega} y_{ijk}\delta y_{ijk} d\Omega = \int_{\Omega} \sigma_{ij} w_k \delta y_{ijk} d\Omega \tag{3.2.5}$$

In general, the matrices in the discrete equations generated by the weak forms are time dependent and, like the consistent mass matrix, not diagonal. Special weighting functions, integration rules, or lumping procedures are necessary to decouple the equations.

3.2.2 Finite element algorithms. The velocities are centered at the nodes, resulting in a continuous velocity field, while the remaining variables are discontinuous functions across the element boundaries. Within the finite element literature, most of the advection algorithms are designed for continuous fields. Some algorithms have been developed for discontinuous fields [185], [186], [187], but they have not been used in hydrocodes. The discussion presented here is limited to the streamline upwind Petrov Galerkin (SUPG) methods developed by Hughes and coworkers since they are the only higher-order accurate finite element methods that have been applied to problems in solid mechanics, e.g., [184]. Many of the finite element ALE formulations have adopted monotonic, element-centered, finite difference algorithms for their work, and those algorithms are discussed in a later section.

The standard Galerkin formulation does not work well for problems dominated by advection because the stability condition is a function of the element Peclet number. The Peclet number, α, is defined as $h||u||/(2k)$, where h is the element size, and k is the cofficient of diffusion. The stability enhancing properties of upwinding are introduced in the SUPG [111], or "streamline diffusion" methods, by using weighting functions that are biased towards the upwind direction. These methods are applied to general hyperbolic systems, including the ALE equations given in the previous section. A typical SUPG weighting function, $\delta\phi_\alpha$, adds additional terms to the interpolation function N where τ is a flow dependent function.

$$\delta\phi_\alpha = N_\alpha + \tau \frac{u_i}{||u||} \frac{\partial N_\alpha}{\partial x_i} \tag{3.2.2.1}$$

Additional terms may be incorportated to improve the resolution of shocks [188], which makes the method nonlinear even for linear problems. For the special case of the scalar advection equation, the original SUPG formulation is a linear, higher order accurate method, which means that it will not be monotonic. Mathematical analysis has shown that the oscillations are localized [112], [189], but even local violations of monotonicity may not be acceptable for problems involving highly nonlinear material models.

Mizukami and Hughes [128] proposed a SUPG method for triangular elements that incorporates a special set of rules for choosing τ. The boundary of each element is divided into vertex and edge zones. The weighting functions are the linear interpolation functions plus constants, C_α. The flow is associated with a particular vertex or edge zone by taking the dot product of the velocity vector at the element centroid with the gradient of the interpolation function. The constants, which in most cases have the values 2/3 or $-1/3$, depend on which zone is associated with the flow.

$$\delta\phi_\alpha = N_\alpha + C_\alpha \tag{3.2.2.2}$$

One especially interesting aspect of this work is their demonstration that the upwinding may have to be in a direction other than along the streamline to guarantee a monotonic solution.

The advection of the element-centered variables is the largest obstacle to attaining second order accurate finite element ALE methods. The discontinuous finite element method, which provides the general theoretical framework for advecting the element-centered variables, was originally developed for the neutron transport equation [185]. For constant stress elements, this method is equivalent to the finite difference donor cell algorithm, which is only first order accurate. Modifications of the SUPG methods have been developed [184], and they can be adjusted between the second order accurate, but unstable, central difference algorithm and the first order accurate donor cell algorithm.

To acheive a specified level of accuracy, the discontinuous finite element method requires more degrees of freedom than its node-centered, continuous counterpart. To be able to represent the same fields as a node-centered formulation, each element must have the same number of degrees of freedom as it has nodes. For example, a four node quadrilateral would require four degrees of freedom, while the ratio of nodes to elements on a logically regular mesh approaches one. To achieve second order accuracy, the discontinuous finite element formulation must work with four times as much information. This requirement is supported by the results obtained for a related problem. Most finite difference advection algorithms are element-centered, and they must be modified to advect the node-centered velocities. One approach is to construct auxilary element-centered variables, advect them, and then reconstruct the velocities from the element-centered auxilary variables. It has been proven (see section 3.5 or [133]) that four auxilary variables must be constructed to advect each component of the velocity field without dispersion.

3.3 FINITE DIFFERENCE ADVECTION METHODS IN ONE DIMENSION

The remap step maps the solution from a distorted Lagrangian mesh onto the new mesh. The underlying assumptions of the remap step are 1) the topology of the mesh is fixed (a complete rezone, discussed later, does not have this limitation), and 2) the mesh motion during a step is less than the dimensions of the elements. Within the fluid community, the second condition is simply stated as saying the Courant number, C, is less than one.

$$C \equiv \frac{u\triangle t}{\triangle x} \leq 1 \tag{3.3.1}$$

Since the mesh motion does not occur over any physical time scale, $\triangle t$ is arbitrary, and $u\triangle t$ 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 are referred to as "advection" algorithms after the first order, scalar conservation equation that is frequently used as a model hyperbolic problem.

$$\frac{\partial \phi}{\partial t} + a(x)\frac{\partial \phi}{\partial x} \tag{3.3.2}$$

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 [111] is not monotonic. Johnson [112] 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 [128] later developed a monotonic SUPG formulation. The essentially non-oscillatory (ENO) [88] algorithms are also not strictly monotonic, and work well for the Euler equations, but their application to hydrodynamics problems has resulted in negative densities [113]. Harten's work on ENO algorithms is discussed because their potential and the belief that their difficulties will be overcome in the future. All the higher order methods discussed here 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. In the interests of simplicity, these methods, therefore, are discussed only in the context of the scalar advection equation.

3.3.1 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.

$$\begin{aligned}\phi_{j+1/2}^{n+1} &= \phi_{j+1/2}^{n} + \frac{\triangle t}{\triangle x}(f_j^\phi - f_{j+1}^\phi) \\ f_j^\phi &= \frac{a_j}{2}(\phi_{j-1/2}^n + \phi_{j+1/2}^n) + \frac{|a_j|}{2}(\phi_{j-1/2}^n - \phi_{j+1/2}^n)\end{aligned} \tag{3.3.1.1}$$

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 $\phi_{j-1/2}^n$ and $\phi_{j+1/2}^n$, and the velocity of the contact discontinuity at node j is a_j.

3.3.2 The van Leer MUSCL algorithm. Van Leer 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 [52]. 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, $\phi_{j+1/2}^n(x)$, 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 $x_{j+1/2}$.

$$\phi^n_{j+1/2}\triangle x_{j+1/2} = \int_{x_j}^{x_{j+1}} \phi^n_{j+1/2}(x)dx \tag{3.3.2.2}$$

To determine the range of ϕ, $[\phi^{min}_{j+1/2}, \phi^{max}_{j+1/2}]$, 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 tranport solutions with large discontinuities. The magnitude of ϕ an algorithm is able to transport before its monotonicity algorithm restricts it is a measure of its “compressiveness.”

The first step up from a piecewise constant function is a piecewise linear function, where x is now the volume coordinate, see section 3.4.1. Conservation is guaranteed by expanding the linear function about the element centroid.

$$\phi^n_{j+1/2}(x) = S^n_{j+1/2}(x - x^n_{j+1/2}) + \phi^n_{j+1/2} \tag{3.3.2.3}$$

Letting $s^n_{j+1/2}$ be a second order approximation of the slope, the limited value of the slope, $S^n_{j+1/2}$, according to the first limiting approach, is determined by assuming the maximum permissible values at the element boundaries.

$$\begin{aligned} S^n_{j+1/2} &= \tfrac{1}{2}\{\text{sgn}(s^L) + \text{sgn}(s^R)\}\text{min}\left(|s^L|, |s^n_{j+1/2}|, |s^R|\right) \\ s^L &= \frac{1}{2\triangle x_{j+1/2}}\left(\phi^n_{j+1/2} - \phi^n_{j-1/2}\right) \\ s^R &= \frac{1}{2\triangle x_{j+1/2}}\left(\phi^n_{j+3/2} - \phi^n_{j+1/2}\right) \end{aligned} \tag{3.3.2.4}$$

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 Eq. (3.3.2.5), 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 Eq. (3.3.2.4). If the element is the downstream element at both boundaries, then the slope in the element has no effect on the solution.

$$\begin{aligned} s^L &= \frac{\phi^n_{j+1/2} - \phi^n_{j-1/2}}{x_{j+1/2} - (x_j + \frac{1}{2}\max(0, a_j)\triangle t)} \\ s^R &= \frac{\phi^n_{j+3/2} - \phi^n_{j+1/2}}{(x_{j+1} - \frac{1}{2}\min(0, a_{j+1})\triangle t) - x_{j+1/2}} \end{aligned} \tag{3.3.2.5}$$

The flux at node j is evaluated using the upstream approximation of ϕ.

$$\begin{aligned} f^\phi_j &= \frac{a_j}{2}(\phi^-_j + \phi^+_j) + \frac{|a_j|}{2}(\phi^-_j - \phi^+_j) \\ \phi^+_j &= S^n_{j+1/2}(x^c - x^n_{j+1/2}) + \phi^n_{j+1/2} \\ \phi^-_j &= S^n_{j-1/2}(x^c - x^n_{j-1/2}) + \phi^n_{j-1/2} \\ x^c &= x^n_j + \tfrac{1}{2}\triangle t a_j \end{aligned} \tag{3.3.2.6}$$

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 $x_{j+1/2}$. When the value of ϕ at the element centroids is assumed to be equal to the element average this algorithm defines a projection.

$$\begin{aligned} s^n_{j+1/2} &= \frac{(\phi^n_{j+3/2} - \phi^n_{j+1/2})\triangle x^2_j + (\phi^n_{j+1/2} - \phi^n_{j-1/2})\triangle x^2_{j+1}}{\triangle x_j \triangle x_{j+1}(\triangle x_j + \triangle x_{j+1})} \\ \triangle x_j &= x^n_{j+1/2} - x^n_{j-1/2} \end{aligned} \tag{3.3.2.7}$$

Christensen [50] fits a cubic polynomial through the four element centroids surrounding node j and interpolates $\phi(x^n_j)$. He reports that this approximation gives him close to third order accuracy.

$$s^n_{j+1/2} = \frac{\phi(x^n_{j+1}) - \phi(x^n_j)}{\triangle x_{j+1/2}} \tag{3.3.2.8}$$

Harten proposes a method for calculating the slope in his paper on ENO methods [88], which is described in a later section.

All these methods rely on reconstructing a distribution of ϕ from its cell averages and evaluating its derivative at the element centroid. Van Leer proposed in his paper [52] and Hancock implemented in PISCES [2] a method that treats the slope as an additional history variable. In addition to keeping extra storage for the slope, the slope must also be advected.

Van Leer extended his ideas to parabolic approximations, and the interested reader should read Ref. [52].

3.3.3 The Piecewise Parabolic Method. Colella and Woodward developed the piecewise parabolic method (PPM) [86] which is similar in flavor to the MUSCL algorithm. The major difference between the algorithms is the addition of some tests to detect discontinuities in the solution, and the ability to alter the approximation of ϕ based on the test results. A parabola is defined within in the element by demanding that the interpolation be conservative and that it passes through the two values defined at the left and right element edges, $\phi_{L,j+1/2}$ and $\phi_{R,j+1/2}$.

$$\begin{aligned}\phi(x) &= \phi_{L,j+1/2} + \xi(\triangle\phi_{j+1/2} + \phi_{6,j+1/2}(1-\xi)) \\ \xi &= \frac{x - x_j}{\triangle x_{j+1/2}} \\ \triangle\phi_{j+1/2} &= \phi_{R,j+1/2} - \phi_{L,j+1/2} \\ \phi_{6,j+1/2} &= 6\big[\phi_{j+1/2} - \tfrac{1}{2}(\phi_{R,j+1/2} + \phi_{L,j+1/2})\big]\end{aligned} \tag{3.3.3.1}$$

In regions where the solution is smooth $\phi_{L,j+1/2}$ and $\phi_{R,j-1/2}$ are equal and the interpolation is continuous at the nodes. The monotonicity constraint is enforced by altering the values of $\phi_{L,j+1/2}$ and $\phi_{R,j+1/2}$, resulting in discontinuities in ϕ at the nodes. A quartic polynomial, H_4, is fit through the integral of ϕ from node $j-2$ to node $j+2$, and $\tilde{\phi}_j$ is initially set to $dH_4/dx|_{x_j}$. This procedure is very similar to Harten's method of generating interpolating functions for his ENO schemes. Colella and Woodward replace the term associated with the average slope of the parabola in element $j+1/2$ appearing in $dH_4/dx|_{x_j}$ with van Leer's limited slope approximation to obtain a sharper representation of discontinuities in the solution.

The first application of the monotonicity constraint is the restriction of ϕ_j to the range defined by the adjacent elements.

$$\phi_j = \max\big(\min(\phi_{j-1/2}, \phi_{j+1/2}), \min(\max(\phi_{j-1/2}, \phi_{j+1/2}), \tilde{\phi}_j)\big) \tag{3.3.3.2}$$

The initial values of $\phi_{L,j+1/2}$ and $\phi_{R,j+1/2}$ are set to ϕ_j. If $\phi_{j+1/2}$ is a local minimum or maximum, the interpolation function is reset to a constant, resulting in a donor cell algorithm just like in the MUSCL scheme. When a new maximum or minimum is created by the choice of the $\phi_{L,j+1/2}$ and $\phi_{R,j+1/2}$, one of the values is reset so that monotonicity is regained and the derivative at the opposite edge of the reset value of ϕ is equal to zero.

If an element is determined to be inside a discontinuity, the parabola is replaced by the linear interpolation developed by van Leer. A discontinuity is considered to occur when the third derivative of ϕ is large and the second derivative changes sign within the element. Furthermore, the first and third derivatives of the solution must have opposite signs to avoid discontinuities being falsely detected within small plateaus of a generally monotonic solution.

The amount of material transported between elements is readily calculated by integrating the parabolic distribution over the transported volume.

3.3.4 Total variation diminishing (TVD) schemes. The advection methods presented in the previous sections are largely geometrical in nature. They are constructed by introducing a piecewise continuous interpolation function and constraining it to be conservative and monotonic. One of the reasons that these methods are popular is the distribution of ϕ is known pointwise by construction, giving the MUSCL and PPM schemes a physical interpretation. By the same token, the geometrical approach is somewhat limited (although the ENO schemes are extremely promising) because polynomials seem to be the only family of interpolation functions that are convenient.

An alternative approach is to calculate the fluxes with a low order monotonic scheme and add as much of the fluxes from a higher order scheme as possible without destroying the monotonicity of the solution. The multiplier that scales the higher order fluxes is nonlinearly dependent on the solution, and it is referred to as the "flux limiter." This approach does not require that the entire solution profile be constructed on a pointwise basis, and is therefore somewhat less restrictive in its range of formulations. Boris and Book [114] introduced this approach with their flux corrected transport (FCT) algorithm. Many others have also taken this approach, including van Leer [115], Roe [116], Chakravarthy and Osher [117], and Harten [118]. As with the approaches in the previous sections, these methods are generally discussed in the context of solving systems of hyperbolic conservation equations. A mathematical framework was established by Harten [118] for proving when advection algorithms would be oscillation free or TVD (total variation diminishing). He defined the total variation, $TV(\phi^n)$, of a solution and required that the total variation decrease with time for an oscillation free solution.

$$\begin{aligned} TV(\phi^{n+1}) &\leq TV(\phi^n) \\ TV(\phi^n) &= \sum_k |\phi^n_{k+1} - \phi^n_k| \end{aligned} \tag{3.3.4.1}$$

The sufficient conditions derived by Harten [118] for the general three point scheme defined in Eq. (3.3.4.2) to be TVD are given by Eq. (3.3.4.3).

$$\phi^{n+1}_{i+1/2} = \phi^n_{i+1/2} - C_i(\phi^n_{i+1/2} - \phi^n_{i-1/2}) + D_{i+1}(\phi^n_{i+3/2} - \phi^n_{i+1/2}) \tag{3.3.4.2}$$

$$0 \le C_i \quad 0 \le D_i \quad 0 \le C_i + D_i \le 1 \tag{3.3.4.3}$$

Sweby [119] considers the scalar general conservation law, Eq. (3.3.4.4), and a class of first order advection algorithms that Osher [120] termed E-schemes because they converge to the correct, entropy satisfying solution, where h^E_j are the numerical fluxes. The simplest example of an E-scheme is the donor cell algorithm.

$$\frac{\partial \phi}{\partial t} + \frac{\partial f(\phi)}{\partial x} = 0 \tag{3.3.4.4}$$

$$\phi^{n+1}_{k+1/2} = \phi^n_{k+1/2} + \frac{\triangle t}{\triangle x}(h^E_i - h^E_{i+1}) \tag{3.3.4.5}$$

Using Sweby's notation, the coefficients C and D for a second order accurate scheme are expressed as functions of the limiter and the solution.

$$\begin{aligned}
C_i &= \nu^+_i\{1 + \alpha^+_i[\varphi(r^+_{i+1/2})/r^+_{i+1/2} - \varphi(r^+_{i-1/2})]\} \\
D_i &= -\nu^-_i\{1 + \alpha^-_i[\varphi(r^-_{i-1/2})/r^-_{i-1/2} - \varphi(r^-_{i+1/2})]\} \\
r^+_{i+1/2} &= \frac{\alpha^+_i(\triangle f_i)^+}{\alpha^+_{i+1}(\triangle f_{i+1})^+} \qquad \alpha^+_i = \tfrac{1}{2}(1 - \nu^+_i) \\
r^-_{i+1/2} &= \frac{\alpha^-_{i+1}(\triangle f_{i+1})^-}{\alpha^-_i(\triangle f_i)^-} \qquad \alpha^-_i = \tfrac{1}{2}(1 + \nu^-_i) \\
(\triangle f_i)^+ &= -(h^E_i - f(\phi_{i+1/2})) \qquad \nu^+_i = \frac{\triangle t(\triangle f_i)^+}{\triangle x(\phi_{i+1/2} - \phi_{i-1/2})} \\
(\triangle f_i)^- &= \quad (h^E_i - f(\phi_{i-1/2})) \qquad \nu^-_i = \frac{\triangle t(\triangle f_i)^-}{\triangle x(\phi_{i+1/2} - \phi_{i-1/2})}
\end{aligned} \tag{3.3.4.6}$$

Sweby established bounds on the limiter $\varphi(r)$, where r is the ratio of two consecutive differences, which guaranteed second order accuracy, see Fig. 22.

$$r_{i+1/2} = \frac{\phi_{i+1/2} - \phi_{i-1/2}}{\phi_{i+3/2} - \phi_{i+1/2}} \tag{3.3.4.7}$$

Although not all flux-limited schemes fall within the bounds of the figure (not all limiters can be expressed in terms of r only), Sweby was able to show that the schemes proposed by van Leer [115], Roe [116], Chakravarthy and Osher [117] fit within his general framework. There is obviously an infinite number of second order flux-limited schemes that fall within the bounds defined by Sweby. Chakravarthy and Osher [117] also unified a wide class of TVD schemes using a somewhat different approach and described the MUSCL scheme as a TVD scheme that preprocesses or limits the data instead of the fluxes.

3.3.5 Essentially non-oscillatory schemes with subcell resolution. Harten [88] has developed a class of essentially non-oscillatory (ENO) schemes that generalize Godunov's scheme to higher order accuracy. They are an interesting extension of the earlier geometrical approaches taken by van Leer [52] and Colella and Woodward [88]. Like MUSCL and PPM, a conservative interpolating polynomial, $R(x, \phi)$, is generated in a manner that depends nonlinearly on ϕ. The ENO schemes are not TVD, but they are very close.

$$TV\big(R(\cdot;\phi)\big) \leq TV(\phi) + \mathcal{O}(\triangle x^{1+P}), \qquad P > 0 \tag{3.3.5.1}$$

Harten [88] has recently extended this work to include the ability capture jump discontinuities within cells (elements). For a scheme of using an interpolating polynomial of degree r, it is exact for all initial data of discontinuous piecewise polynomials less than or equal to r. Although this method is not currently used in any production hydrocodes, it holds a great deal of promise. Although it looses any accuracy it has above second order when it is used in conjunction with a spatial operator split in multiple dimensions, many real problems are dominated by flows primarily in one dimension.

The interpolating polynomial, R, is generated by differentiating a polynomial, H, that approximates the integral of ϕ in one space dimension.

$$\begin{aligned}
\Phi(x) &= \int_{x_1}^{x} \phi(x)dx \\
\Phi_i &= \Phi(x_i) = \sum_{k=2}^{i}(x_k - x_{k-1})\phi_{k-1/2} \\
\Phi_1 &= 0 \\
H(x_i, \Phi) &= \Phi_i \\
R(x, \phi) &= \frac{d}{dx}H(x, \Phi)
\end{aligned} \tag{3.3.5.2}$$

The following ENO construction is from Harten [88]. Other researchers have also developed ENO schemes, and the interested reader is referred to the original paper for details. Following Harten's notation, $H_m(x,\phi)$ for element $j+1/2$ is a piecewise polynomial function of x of degree m which is defined in terms of the unique polynomial, $q_{j+1/2}$, that interpolates Φ on the set of consecutive $m+1$ points in $S_m(i)$, where i is the first node in the domain of interpolation.

$$S_m(i) \equiv \{x_i, x_{i+1}, \ldots, x_{i+m}\} \tag{3.3.5.3}$$

To ensure that the polynomial passes through ϕ at nodes j and $j+1$, the range of $i(j)$ is defined to always include the two nodes of the element.

$$j - m + 1 \le i(j) \le j \tag{3.3.5.4}$$

For each element, the allowable range defined by Eq. (3.3.5.4) gives m possible choices for the interpolating polynomial. The choice of a particular polynomial is based on a smoothness criteria and Harten provides two candidate criteria based on the estimated magnitude of the mth derivative. This information is based on a table of Newton's divided differences [122]. Letting $D\phi[S_k(i)]$ denote the kth divided difference of the polynomial interpolating the points $S_k(i)$, the first criteria simply chooses the $i(j)$ in the range defined by Eq. (3.3.5.4) that has the smallest absolute value of $D\phi[S_m(i)]$.

The second criteria, which is the one recommended by Harten, is hierarchical. Let $i_k(j)$ be the choice for a kth degree polynomial on $S_k(i_k(j))$. For k equal to 1, the necessary choice for $i_1(j)$ is j. To choose $i_{k+1}(j)$, two stencils for consideration are obtained by adding a point to either end of $S_k(i_k(j))$.

$$\begin{aligned} S^L_{k+1} &= S_{k+1}(i_k(j) - 1) \\ S^R_{k+1} &= S_{k+1}(i_k(j)) \end{aligned} \tag{3.3.5.5}$$

Harten selects the smoother of the two stencils and the final value of $i(j)$ is $i_m(j)$.

$$i_{k+1}(j) = \left\{ \begin{array}{ll} i_k(j) - 1 & \text{if } |D\phi[S^L_{k+1}]| < |D\phi[S^R_{k+1}]| \\ i_k(j) & \text{otherwise} \end{array} \right\} \tag{3.3.5.6}$$

When a discontinuity is detected within an element $j+1/2$, $R_{j-1/2}$ interpolates ϕ to the left of the discontinuity, and $R_{j+3/2}$ is used to the right. The location of the discontinuity, which is not known at this point and which never needs to be calculated for our purposes, is designated θ.

$$\hat{R}_{j+1/2}(x,\phi) = \left\{ \begin{array}{lll} R_{j-1/2}(x,\phi) & \text{for} & x_j \leq x < \theta \\ R_{j+3/2}(x,\phi) & \text{for} & \theta < x \leq x_{j+1} \end{array} \right\} \tag{3.3.5.7}$$

The location of the discontinuity is determined by demanding that the integral of $\hat{R}_{j+1/2}$ equal $\Phi_{j+1} - \Phi_j$. If some criteria determines that there is a discontinuity in element $j+1/2$, there is no guarantee from the construction of R that a θ exists in element $j+1/2$ which will satisfy the conservation constraint. Such a location does exist if the integral of one of the adjacent R is too large and the other is too small.

$$[H_m^{j-1/2}(x_{j+1}) - \Phi_j][\Phi_{j+1} - H_m^{j+3/2}(x_j)] \leq 0 \quad \text{if} \quad x_j \leq \theta \leq x_{j+1} \tag{3.3.5.8}$$

The original interpolating function, $R_{j+1/2}$, is used if Eq. (3.3.5.8) is not satisfied despite the results of the discontinuity criteria. If desired, the location of the discontinuity can be calculated by solving the nonlinear algebraic conservation equation, Eq. (3.3.5.9).

$$F_{j+1/2}(\theta) = H_m^{j-1/2}(\theta) - H_m^{j+3/2}(\theta) - (\Phi_{j+1} - \Phi_j) = 0 \tag{3.3.5.9}$$

Defining $\sigma_{j+1/2}$ as some measure of non-smoothness, a discontinuity is assumed to occur in element $j+1/2$ if the interpolation in that element is less smooth than in its two adjacent elements.

$$\begin{aligned} \sigma_{j+1/2} &= \left| \frac{d^k}{dx^k} R^{j+1/2} \right| \qquad \text{for example} \\ \sigma_{j+1/2} &> \max(\sigma_{j-1/2}, \sigma_{j+3/2}) \;\; \text{at a discontinuity} \end{aligned} \tag{3.3.5.10}$$

The actual flux calculation does not require the precise location of θ. To demonstrate , assume that Eq. (3.3.5.8) and (3.3.5.10) are satisfied. By definition, the flux is the time-averaged integral of ϕ between $x_{j+1} - a\triangle t$ and x_{j+1}. The presence of a discontinuity within the transport volume can be detected by checking for a sign change in $F_{j+1/2}$.

$$F_{j+1/2}(x_{j+1} - a\triangle t) \cdot F_{j+1/2}(x_{j+1}) \leq 0 \quad \text{if} \quad x_{j+1} - a\triangle t \leq \theta \leq x_{j+1} \tag{3.3.5.11}$$

Assuming that the inequality Eq. (3.3.5.11) is satisfied, i.e., $x_{j+1} - a\triangle t \leq \theta \leq x_{j+1}$, ϕ_{j+1}^{+} is calculated using Eq. (3.3.5.12a), otherwise Eq. (3.3.5.12b) is used.

$$\phi^{+}_{j+1} = \frac{1}{\triangle t}[\Phi_{j+1} - \Phi_j - \{H_m^{j-1/2}(x_{j+1} - a\triangle t) - \Phi_j\}] \qquad (3.3.5.12a)$$

$$\phi^{+}_{j+1} = \frac{1}{\triangle t}[\Phi_{j+1} - H_m^{j+3/2}(x_{j+1} - a\triangle t)] \qquad (3.3.5.12b)$$

Although the ENO schemes are very attractive, investigators [190], [191] have recently demonstrated that ENO schemes may be unstable or suffer a loss of convergence even for smooth functions. The investigators chose test functions that caused ENO to choose a stencil that was downstream of the flow. The major difficulty appears to be that some of the difference stencils that ENO is allowed to choose are linearly unstable. When ENO is restricted to linearly stable stencils, it seems to regain all its good properties.

3.3.6 Other nonoscillatory schemes. Several other advection schemes have been recently proposed that have the nice properties of an ENO scheme, but which appear to be more efficient and stable. The one discussed here is monotone if all the data used in the stencil are monotone, but it permits flux values larger than adjacent cell values when a strict extremum is detected. This method is therefore not sign preserving: near a strict minimum, the interpolant of a strictly positive, oscillatory, function, such as density, may go negative. For individual zones that have this problem, a different limiter may be preferred.

A particularly nice framework for describing the TVD, UNO2 [192], and SONIC [193] algorithms has been devised by Huynh [193], [194]. It is based on a geometrical interpretation of the limiters, and it is expressed mathematically in terms of the median function, which can be readily vectorized. The median of an odd number of values is defined as the one that has the same number of values above and below. For three numbers, it is the one which lies between the other two. The minmod function is easily expressed in terms of the median of two numbers and 0, and the median function for three arbitrary numbers can be expressed in terms of the minmod function.

$$\begin{aligned} \text{minmod}(x,y) &= \text{median}(x,y,0) \\ \text{median}(x,y,z) &= x + \text{minmod}(y-x, z-x) \\ &= y + \text{minmod}(x-y, z-y) \\ &= z + \text{minmod}(x-z, y-z) \end{aligned} \qquad (3.3.6.1)$$

The minmod function can be expressed in terms of the three standard FORTRAN intrinsic functions min, abs, and sign.

$$\text{minmod}(x, y) = [\text{sign}(\tfrac{1}{2}, x) + \text{sign}(\tfrac{1}{2}, y)]\text{min}\{\text{abs}(x), \text{abs}(y)\} \tag{3.3.6.2}$$

Huynh has created a family of algorithms that he collectively refers to as SONIC, and the one that has the best performance overall is SONIC-A [193]. It maintains the peak values on an advected profile better than MUSCL, while avoiding the diffusivity of UNO2 and the overcompressiveness of the Superbee limiter. Following Huynh [194], the first and second divided differences are defined in Eq. (3.3.6.3).

$$\begin{aligned}
s_j &= \frac{\phi_{j+1/2} - \phi_{j-1/2}}{x_{j+1/2} - x_{j-1/2}} \\
s_{j+1/2} &= \text{minmod}(s_j, s_{j+1}) \\
d_{j+1/2} &= \frac{s_{j+1} - s_j}{x_{j+1} - x_j} \\
d_j &= \text{minmod}(d_{j-1/2}, d_{j+1/2})
\end{aligned} \tag{3.3.6.3}$$

Using the nonoscillatory parabolic interpolation ideas from UNO2, the derivatives, p', associated with the two monotonic parabolas to the left and right are evaluated at the element centroid. A limiting value, $t_{j+1/2}$ is also calculated using the minmod function.

$$\begin{aligned}
p'_j(x_{j+1/2}) &= s_j + d_j(x_{j+1/2} - x_{j-1/2}) \\
p'_{j+1}(x_{j+1/2}) &= s_{j+1} + d_{j+1}(x_{j+1/2} - x_{j+3/2}) \\
t_{j+1/2} &= \text{minmod}(p'_j, p'_{j+1})
\end{aligned} \tag{3.3.6.4}$$

A tentative value of the derivative is defined as the arithmetic average of the slopes from the two parabolas, a limiting slope, t_{max}, is defined, and the final slope is calculated using the minmod function. The flux at node j is calculated using the slope from Eq. (3.3.6.5) in Eq. (3.3.2.6).

$$\begin{aligned}
S &= \tfrac{1}{2}[p'_j + p'_{j+1}] \\
t_{max} &= \text{sgn}(t_j)\text{max}(2|s_j|, |t_j|) \\
S^n_{j+1/2} &= \text{minmod}(S, t_{max})
\end{aligned} \tag{3.3.6.5}$$

3.4 TWO-DIMENSIONAL ADVECTION

3.4.1 Volume and mass coordinates. The spatial coordinate system is not a very convenient measure of distance even in one dimension if the underlying geometry is not planar. Consider, for instance, using the van Leer MUSCL algorithm on a one-dimensional problem in cylindrical coordinates. If the linear distribution of ϕ is expanded about the spatial centroid, $r_{i+1/2}$, of the element, then the distribution is conservative only if $S^n_{i+1/2}$ is equal to zero (donor cell). An arbitrary slope can be used, however, if the linear distribution is expanded about the volume centroid. Instead of calculating the spatial location of all the volume centroids, volume coordinates can be used instead.

$$x_i \equiv \int_{r_1}^{r_i} r dr \tag{3.4.1.1}$$

For an arbitrary cross section, the volume coordinates at each node are defined by the volume integral.

$$x_i \equiv \int_{r_1}^{r_i} \frac{dV}{dr} dr \tag{3.4.1.2}$$

A similar problem arises in measuring the distance along a particular row of elements in space. Again, volume coordinates can be defined, where $J(y,z;i,j)$ is the Jacobian relating the spatial coordinates y and z to the mesh coordinates i and j. The superscript $(j,j+1)$ denotes that the volume coordinates are defined in the i direction between j lines j and $j+1$. In general, $x_i^{(j,j+1)} - x_{i-1}^{(j,j+1)}$ is not equal to $x_i^{(k,k+1)} - x_{i-1}^{(k,k+1)}$ when j and k are not equal.

$$x_i^{(j,j+1)} \equiv \int_1^i \int_j^{j+1} J(y,z;i,j) di dj \tag{3.4.1.3}$$

Volume coordinates can also be defined in the j direction. Mass coordinates are similar in definition to the volume coordinate except that the density appears in the integral of Eq. (3.4.1.3).

The advection algorithms discussed in the previous sections can be used for arbitrary one-dimensional flows without any other changes than replacing the spatial coordinate x with the volume coordinates and replacing the one-dimensional transport volume, $a\triangle t$, with the transport volume between elements.

3.4.2 Spatial operator split. The simplest approach to extending a one-dimensional advection algorithm to two dimensions is the spatial operator split, or more commonly, the “alternating direction” method [98]. In the context of the operators in Eq. (3.2.3), A and B are $-a_y\partial\phi/\partial y$ and $-a_z\partial\phi/\partial z$. Consider for the moment the advection between element $(i+1/2, j+1/2)$ and its eight adjacent elements, and assume that the velocities are edge centered, see Fig. 2. If the flow is restricted to the i or j mesh directions, then the one-dimensional algorithms discussed in the previous section can be applied directly. In general the flow is two-dimensional, and some volume should be transported between elements that share only a vertex. Note that by limiting the motion of an edge to its normal direction, there is no unambiguously defined vertex volume as there is for meshes with node-centered velocities. If the one-dimensional advection algorithms are applied simultaneously in the two mesh directions, no material is tranported between diagonal elements in a single step. The velocity at which a signal propagates for isotropic advection along the diagonal is slower than in the two mesh directions, resulting in a loss of spherical symmetry.

The vertex coupling is introduced by using an operator split. A first order application of the operator split is obtained by advecting in the i direction followed by advecting in the j direction. To simplify the following discussion, only the volume transported between elements is considered, and the factor of $\triangle t$ is implied in the transport volume f. Since the advection algorithms are nonlinear, a precise estimate of how much of ϕ was transported between diagonal elements would require the use a Taylor expansion. The complexity of the analysis would be greatly increased but the additional insight would be insignificant. Assuming the flow is positive in the i and j directions, during the i direction sweep, a volume of $f_{(i+1,j+1/2)}$ is transported between element $(i+1/2, j+1/2)$ and element $(i+3/2, j+1/2)$. During the j sweep, element $(i+3/2, j+1/2)$ donates a volume of $f_{(i+3/2,j+1)}$ to element $(i+3/2, j+3/2)$. The volume transported indirectly from element $(i+1/2, j+1/2)$ to element $(i+3/2, j+3/2)$ is calculated assuming uniform mixing, where the volume of the elements after the first sweep is superscripted with $n+1/2$.

$$f_{(i+1,j+1)} = f_{(i+1,j+1/2)} f_{(i+3/2,j+1)} / V^{n+1/2}_{(i+3/2,j+1/2)} \tag{3.4.2.1}$$

For a uniform flow on a rectangular mesh, the vertex transport volume simplifies to $(\triangle t)^2 \dot{y}\dot{z}$. Aliasing errors are introduced into the calculation if the i sweep always precedes the j sweep. These errors are eliminating by switching the order of the sweeps every time step: at t^n, i, j, at t^{n+1}, $j, i, \ldots$ etc.

A second order algorithm requires that the sweeps be split into three stages: 1) a sweep in the i direction using one half the transport volumes, 2) a j sweep using the full transport volume (in theory, this step should be split into two steps, each using one half the transport volume), and 3) another i sweep using one half the volumes. During the first sweep material from element $(i+1/2, j+1/2)$ is transported to element $(i+3/2, j+1/2)$. During the second sweep, material originally in element $(i+1/2, j+1/2)$ is transported from element $(i+3/2, j+1/2)$ to element $(i+3/2, j+3/2)$, and material is transported from element $(i+1/2, j+1/2)$ to element $(i+1/2, j+3/2)$. On the third sweep, additional material from element $(i+1/2, j+1/2)$ is transported to element $(i+3/2, j+3/2)$ via element $(i+1/2, j+3/2)$. Ignoring terms higher than second order in $\triangle t$, the transport volume from element $(i+1/2, j+1/2)$ to element $(i+3/2, j+3/2)$ is calculated.

$$f_{(i+1,j+1)} = \tfrac{1}{2}\left\{ \frac{f_{(i+1/2,j+1)} f_{(i+1,j+3/2)}}{V^{n+1/2}_{(i+1/2,j+3/2)}} + \frac{f_{(i+1,j+1/2)} f_{(i+3/2,j+1)}}{V^{n+1/2}_{(i+3/2,j+1/2)}} \right\} \tag{3.4.2.2}$$

For a uniform flow with elements of a constant size, this estimate of the corner volume is the same as the previous estimate, Eq. (3.4.2.1). There are, however, cubic terms in the full expression for $f_{(i+1,j+1)}$ in Eq. (3.4.2.2) that become significant in large Courant number flows. Considering the special case of a uniform flow on a mesh with square elements of unit volume, the full expression is given by Eq. (3.4.2.3).

$$\begin{aligned} f_{(i+1,j+1)} =& \tfrac{1}{2} f_{(i+1,j+1/2)} f_{(i+3/2,j+1)} (1 - \tfrac{1}{2} f_{(i+2,j+3/2)}) \\ &+ \tfrac{1}{2} f_{(i+1/2,j+1)} f_{(i+1,j+3/2)} (1 - \tfrac{1}{2} f_{(i+1,j+1/2)}) \end{aligned} \tag{3.4.2.3}$$

In addition, the three-sweep procedure transports material from element $(i+1/2, j+1/2)$ to elements $(i+5/2, j+1/2)$ and $(i+5/2, j+3/2)$. For the donor cell algorithm, the amount of material can be quite significant at large Courant numbers. The higher order monotonic advection algorithms drop to the first order donor cell algorithm in regions with large discontinuities, e.g., shocks and contact discontinuities, and the greatest amount of diffusion, therefore, will be experienced at the these discontinuities. Shocks tend to eliminate the smearing introduced by the advection algorithms because of the continual convergence of their characteristics, but the resolution of the contact discontinuities is generally poor. In most hydrocodes the advection is performed in only two sweeps with the order alternating every time step.

The proper implementation of an alternating direction advection method involves some subtle considerations when the mesh is nonorthogonal. The best interface tracking algorithms divide an element into regions of the appropriate volumes with straight line segments. The obvious way to implement an alternating direction method is to define the transport volumes as the regions swept out by the edges during a time step. Note that the transport volumes may overlap at the nodes, but that the total volume that is transported will be correct. After the first direction is advected, the interface must be recalculated to account for the transport. The question therefore arises as to what the proper geometry is after the first direction.

Another problem can occur even when the mesh is orthogonal. Consider a square element in a flow that translates along the line $y = x$. The transport volume on the right edge is defined by the four locations $x^{n+1}_{(i+1,j)}$, $x^{n+1}_{(i+1,j+1)}$, $x^{n}_{(i+1,j)}$, and $x^{n}_{(i+1,j+1)}$, see Fig. 23. The interfaces are calculated using the geometry at t^{n+1}, but the transport volume contains a triangular region, $x^{n}_{(i+1,j)}$, $x^{n+1}_{(i+1,j)}$, and $(y^{n}_{(i+1,j)}, z^{n+1}_{(i+1,j)})$ that lies entirely outside of domain of the element at t^{n+1}. What material should be assigned to that region? A blind application of the interface tracker destroys the conservation of the different materials, although the total volume of all the materials is conserved.

The simplest way to eliminate these problems is to use volume coordinates. While this strategy works well with a smooth mesh, the transformation between volume coordinates and spatial coordinates can lead to highly distorted material interfaces on distorted meshes. A different strategy has been developed by Norm Johnson [138] for CAVEAT, see Fig. 23. He calculates the transport volumes on all element edges due to the velocity field in one of the spatial directions (r or z), advects the material isotropically, and then does the same thing for the other direction (z or r). Note that the operator split is in terms of spatial directions, not logical mesh directions. The geometry of the elements is well defined at all times. One problem with this strategy is that symmetry is lost in spherical calculations. For problems where this symmetry is important, isotropic advection is used in CAVEAT.

To avoid the second problem, he first calculates the magnitude of the transport volume at an edge and then creates a region having the same volume that lies entirely within the element. The new transport volume is constructed by calculating the position of a line parallel to the edge such that its intersection with the element encloses the proper volume, see Fig. 23b. This calculation uses the same algorithms used to determine the location of the material interfaces. Extensive testing of this transport volume construction has demonstrated that it works very well. For rectangular meshes, this construction appears to be essentially equivalent to using volume coordinates.

3.4.3 Unsplit two-dimensional advection methods. Advection methods that do not use operator splitting are referred to as "unsplit" methods despite the fact that they were never split in the first place. One of the first higher order monotonic algorithms was the flux-corrected transport of Boris and Book [114]. It was generalized to two dimensions in an unsplit manner by Zalesak [123], and implemented in an adaptive Eulerian programs for gas dynamics by Löhner [124] and Oden [125]. One particularly interesting aspect of this work is that FCT was originally developed as an edge-centered algorithm and it was converted to a node-centered finite element method by doing little more than altering some nomeclature. This method has not been used extensively in multimaterial hydrocodes because other methods, such as the MUSCL scheme, are better in one dimension and the spatial operator split works well in most cases.

Smolarkiewicz's multipass donor cell algorithm [126] was recently extended to two dimensions in an unsplit manner by Margolin and Smolarkiewicz [127] and implemented in SHALE [23]. The essential idea is to estimate the truncation error from the donor cell advection and subtract it. A sequence of donor cell steps is performed after the initial advection with each step using an antidiffusive velocity field that is calculated from the field being advected.

Mizukami and Hughes [128] developed a monotonic finite element advection algorithm for unstructured meshes based on SUPG (Streamline Upwind Petrov Galerkin) approach to computational fluid dynamics. It advects node-centered fields, which is useful for the momentum advection, but attempts at using the simpler SUPG formulation for element-centered quantities reduced the method to an interpolated donor cell algorithm even in one dimension [105].

Many other researchers have developed unsplit, multidimensional advection algorithms that are not mentioned here. Oil reservoir simulation is one application area in particular where unsplit methods are very valuable, and the interested reader is urged to survey their literature for additional methods. The work by Bell, Dawson, and Shubin [129] on a Godunov method for scalar conservation equations is of particular interest. It presents some of the most accurate pure convection solutions obtained to date.

The major drawback to unsplit algorithms is their cost. Another difficulty is the enforcement of the monotonicity constraint along the mesh diagonals without being overly diffusive because a linear approximation of ϕ will always have its maximum and minimum at the vertices. As computers become faster, unsplit methods will probably become more popular.

A relatively efficient unsplit method, which uses a time-centering strategy similar to the Runge-Kutta method, has been proposed recently by Christensen [50] that gives vertex coupling. The advection algorithms are applied in both directions simultaneously, but with time-centered values of ϕ used in the transport volumes. To obtain the time-centered values of ϕ, the advection algorithms are applied from t^n to $t^{n+1/2}$, or equivalently, half the full transport volume are used in the calculation.

$$
\begin{aligned}
\phi^{n+1/2}_{(i+1/2,j+1/2)} =& \frac{1}{V^{n+1/2}_{(i+1/2,j+1/2)}} \Big\{ \phi^{n}_{(i+1/2,j+1/2)} V^{n}_{(i+1/2,j+1/2)} \\
& + f(\phi^n)_{(i,j+1/2)} - f(\phi^n)_{(i+1,j+1/2)} \\
& + f(\phi^n)_{(i+1/2,j)} - f(\phi^n)_{(i+1/2,j+1)} \Big\} \\
\phi^{n+1}_{(i+1/2,j+1/2)} =& \frac{1}{V^{n+1}_{(i+1/2,j+1/2)}} \Big\{ \phi^{n}_{(i+1/2,j+1/2)} V^{n}_{(i+1/2,j+1/2)} \\
& + f(\phi^{n+1/2})_{(i,j+1/2)} - f(\phi^{n+1/2})_{(i+1,j+1/2)} \\
& + f(\phi^{n+1/2})_{(i+1/2,j)} - f(\phi^{n+1/2})_{(i+1/2,j+1)} \Big\}
\end{aligned}
\tag{3.4.3.1}
$$

If the terms cubic in the volumes are again ignored, then Eq. (3.4.3.1) transports the same volume as given by Eq. (3.4.2.2), but with both volumes centered at $n + 1/2$. The particular advantage of this approach is it works on meshes that do not have a logically regular structure.

A similar idea was recently published by Colella [90] for hyperbolic systems of equations, which he calls the CTU (Corner Transport Upwind) scheme. It is based on tracing the characteristics of the advection equation in two dimensions. Although his publication is recent, it has been in use since 1983 [131]. A scheme identical to the two-dimensional scalar advection algorithm was derived in a different manner by van Leer [132]. Following Colella's presentation, for a problem with a constant velocity field of (u, ν) on a uniform mesh, the second order advection update can be written in terms of the time-centered values defined in Eq. (3.4.3.2).

$$\begin{aligned}
\phi^{n+1}_{(i+1/2,j+1/2)} &= \phi^{n}_{(i+1/2,j+1/2)} + \frac{u\triangle t}{\triangle y}(\phi^{n+1/2}_{(i,j+1/2)} - \phi^{n+1/2}_{(i+1,j+1/2)}) \\
&+ \frac{\nu\triangle t}{\triangle z}(\phi^{n+1/2}_{(i+1/2,j)} - \phi^{n+1/2}_{(i+1/2,j+1)}) \\
\phi^{n+1/2}_{(i+1,j+1/2)} &= \phi^{n}_{(i+1,j+1/2)} + \tfrac{1}{2}(\triangle y - u\triangle t)\frac{\triangle^y\phi_{(i+1/2,j+1/2)}}{\triangle y} \\
&- \left\{\frac{\nu\triangle t}{2\triangle z}(\phi^{n}_{(i+1/2,j+1/2)} - \phi^{n}_{(i+1/2,j-3/2)})\right\} \\
\phi^{n+1/2}_{(i+1/2,j+1)} &= \phi^{n}_{(i+1/2,j+1)} + \tfrac{1}{2}(\triangle z - \nu\triangle t)\frac{\triangle^z\phi_{(i+1/2,j+1/2)}}{\triangle z} \\
&- \left\{\frac{u\triangle t}{2\triangle y}(\phi^{n}_{(i+1/2,j+1/2)} - \phi^{n}_{(i-1/2,j+1/2)})\right\}
\end{aligned} \tag{3.4.3.2}$$

The terms $\triangle^y\phi_{(i+1/2,j+1/2)}/\triangle y$ and $\triangle^z\phi_{(i+1/2,j+1/2)}/\triangle z$ are montonic approximations to the derivatives of ϕ in the y and z directions. The terms that do not appear in the first order unsplit algorithm are enclosed by curly brackets.

In the previous sections we have interpreted the advection operation as a projection of the solution on the Lagrangian mesh at t^n onto the spatially fixed Eulerian mesh. The extra terms arising in the second order unsplit algorithm defined by Eq. (3.4.3.2) can be given a geometrical interpretation. For $\phi^{n+1/2}_{(i+1,j+1/2)}$, the extra term is $-\frac{\nu\triangle t}{2\triangle z}(\phi^{n}_{(i+1/2,j+1/2)} - \phi^{n}_{(i+1/2,j-3/2)})$, which accounts for the fact that the centroid of the transport volume at $(i+1, j+1/2)$ is shifted in the positive z direction by the flow. In a similar manner, the additional term for $\phi^{n+1/2}_{(i+1/2,j+1)}$ is associated with the displacement of the transport volume centroid associated with the y direction flow.

3.5 MOMENTUM ADVECTION

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 [111], are applied to element-centered quantities [105]. 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 two dimensions, which are briefly discussed, are straight forward even if the equations do become lengthy. A detailed discussion of the algorithms in two dimensions is presented in Ref. [133].

3.5.1 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 Eq. (3.5.1.1), where the volume is V.

$$\begin{aligned}\phi^{+}_{j+1/2} &= \frac{\phi^{-}_{j+1/2}V^{-}_{j+1/2} + \phi_j f_j - \phi_{j+1} f_{j+1}}{V^{+}_{j+1/2}} \\ V^{+}_{j+1/2} &= V^{-}_{j+1/2} + f_j - f_{j+1}\end{aligned} \tag{3.5.1.1}$$

3.5.2 The staggered mesh algorithm. YAQUI [26] 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.

$$\begin{aligned}v^{+}_{j} &= \frac{M^{-}_{j} v^{-}_{j} + v_{j-1/2}\tilde{f}_{j-1/2} - v_{j+1/2}\tilde{f}_{j+1/2}}{M^{+}_{j}} \\ v^{+}_{j} &= \frac{M^{-}_{j} v^{-}_{j} + (\rho v)_{j-1/2} f_{j-1/2} - (\rho v)_{j+1/2} f_{j+1/2}}{M^{+}_{j}} \\ M^{+}_{j} &= M^{-}_{j} + \tilde{f}_{j-1/2} - \tilde{f}_{j+1/2}\end{aligned} \tag{3.5.2.1}$$

A consistency condition, first defined by DeBar [102], 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.

$$\begin{aligned} M_j^+ &= \tfrac{1}{2}[M_{j-1/2}^+ + M_{j+1/2}^+] \\ &= \tfrac{1}{2}\big[(M_{j-1/2}^- + \rho_{j-1}f_{j-1} - \rho_j f_j) + (M_{j+1/2}^- + \rho_j f_j - \rho_{j+1}f_{j+1})\big] \\ &= M_j^- + \tfrac{1}{2}\big[(\rho_{j-1}f_{j-1} - \rho_j f_j) + (\rho_j f_j - \rho_{j+1}f_{j+1})\big] \end{aligned} \tag{3.5.2.3}$$

The staggered mass fluxes and transport volumes are defined by equating Eq. (3.5.2.3) and Eq. (3.5.2.2).

$$\begin{aligned} \rho_{j+1/2}f_{j+1/2} = \tilde{f}_{j+1/2} &= \tfrac{1}{2}(\rho_j f_j + \rho_{j+1}f_{j+1}) \\ &= \tfrac{1}{2}(\tilde{f}_j + \tilde{f}_{j+1}) \end{aligned} \tag{3.5.2.4}$$

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 Eq. (3.5.2.4) requires the solution of a nonlinear equation for each transport volume. In contrast, the mass flux is explicitly defined by Eq. (3.5.2.4). Most codes, including KRAKEN [102], CSQ [62], CTH [63], and DYNA2D [103], 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 two dimensions. 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 [134]. 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 [135]. This is the primary reason, according to Margolin [135], that the element-centered algorithm was adopted in SALE [24].

3.5.3 The SALE algorithm. SALE advects an element-centered momentum and redistributes its changes to the nodes [24]. The mean element velocity, $\overline{v}_{j+1/2}$, specific momentum, $p_{j+1/2}$, element momentum, $P_{j+1/2}$, and nodal momentum are defined by Eq. (3.5.3.1).

$$\begin{aligned}\overline{v}_{j+1/2} &= \tfrac{1}{2}(v_j + v_{j+1}) \\ p_{j+1/2} &= \rho_{j+1/2}\overline{v}_{j+1/2} \\ P_{j+1/2} &= M_{j+1/2}\overline{v}_{j+1/2} \\ P_j &= \quad M_j v_j\end{aligned} \tag{3.5.3.1}$$

Denoting the change in the element momentum $\triangle P_{j+1/2}$, the change in the velocity at a node is calculated by distributing half the momentum change from the two adjacent elements.

$$\begin{aligned}\triangle P_{j-1/2} &= p_{j-1}f_{j-1} - p_j f_j \\ P_j^+ &= P_j^- + \tfrac{1}{2}(\triangle P_{j-1/2} + \triangle P_{j+1/2}) \\ v_j^+ &= \frac{P_j^+}{M_j^+}\end{aligned} \tag{3.5.3.2}$$

This algorithm can also be implemented by advecting the mean velocity, $\overline{v}_{j+1/2}$ with the transported mass, and the transported momentum $p_j f_j$ is changed to $\overline{v}_j \tilde{f}_j$.

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 evident when the velocities is calculated from the momenta and no transport occurs.

$$\begin{aligned}v_j^+ &= \frac{\frac{1}{2}\left[\frac{1}{2}M_{j-1/2}^-(v_{j-1}^- + v_j^-) + \frac{1}{2}M_{j+1/2}^-(v_j^- + v_{j+1}^-)\right]}{M_j^+} \\ &= \tfrac{1}{2}v_j^- + \frac{M_{j-1/2}^+ v_{j-1}^- + M_{j+1/2}^+ v_{j+1}^-}{4M_j^+}\end{aligned} \tag{3.5.3.3}$$

Since the original velocities are not recovered, Eq. (3.5.3.3) indicates that there is an inversion error associated with the algorithm.

3.5.4 The SHALE algorithm. Margolin and Beason [136] found that the SALE algorithm created spurious oscillations near regions of steep gradients. They attributed the oscillations to the inversion error illustrated in Eq. (3.5.3.3). To eliminate the inversion error, the SHALE algorithm advects two variables, one representing the average momentum, P, and the other, the gradient of the momentum, S. Note that the element-centered momentum defined here differs from the definition used by the SALE algorithm.

$$\begin{aligned} P_j &= P_{j+1/2} + S_{j+1/2}(x_j - x_{j+1/2}) = M_j v_j \\ P_{j+1} &= P_{j+1/2} + S_{j+1/2}(x_{j+1} - x_{j+1/2}) = M_{j+1} v_{j+1} \\ P_{j+1/2} &= \tfrac{1}{2}(P_{j+1} + P_j) = \tfrac{1}{2}(M_j v_j + M_{j+1} v_{j+1}) \\ S_{j+1/2} &= \frac{P_{j+1} - P_j}{x_{j+1} - x_j} \end{aligned} \tag{3.5.4.1}$$

As with the SALE algorithm, the velocity is updated incrementally. Since this algorithm does not have the inversion errors associated with the SALE algorithm, the velocity could also be updated from the final values of S and P. The velocity at node j can be expressed in terms of S and P of element $j + 1/2$ or $j - 1/2$ as in Eq. (3.5.4.1), but the two expressions are averaged to obtain a symmetric algorithm.

$$\begin{aligned} P_j^+ =& P_j^- + \tfrac{1}{2}\big[\triangle P_{j-1/2} + \triangle S_{j-1/2}(x_j - x_{j-1/2}) \\ &+ \triangle P_{j+1/2} + \triangle S_{j+1/2}(x_j - x_{j+1/2})\big] \\ v_j^+ =& P_j^+ / M_j^+ \end{aligned} \tag{3.5.4.2}$$

The primary disadvantage of this algorithm is that it is not monotonic unless the flux limiters of the underlying element-centered advection algorithms are modified. Margolin has successfully rewritten the limiters for his advection algorithms [135], but he has indicated no systematic method that can be applied to all algorithms.

3.5.5 The HIS (Half Index Shift) algorithm. Benson [133] developed this algorithm based on his analysis of the two 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 SHALE 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, $\Psi_{(1,j+1/2)}$ and $\Psi_{(2,j+1/2)}$ 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 $j + 1/2$.

$$\begin{Bmatrix} \Psi^-_{(1,j+1/2)} \\ \Psi^-_{(2,j+1/2)} \end{Bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{Bmatrix} \phi_j^- \\ \phi_{j+1}^- \end{Bmatrix} \tag{3.5.5.1}$$

This relation is readily inverted.

$$\left\{ \begin{array}{c} \phi_j^+ \\ \phi_{j+1}^+ \end{array} \right\} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \left\{ \begin{array}{c} \Psi^+_{(1,j+1/2)} \\ \Psi^+_{(2,j+1/2)} \end{array} \right\} \tag{3.5.5.2}$$

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, $\Psi^-_{(1,j+1/2)}$ and $\Psi^-_{(2,j+1/2)}$ 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 $\Psi^+_{(1,j+1/2)}$ and $\Psi^+_{(2,j+1/2)}$ 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.

$$\left\{ \begin{array}{c} \Psi_{(1,j+1/2)} \\ \Psi_{(2,j+1/2)} \end{array} \right\} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \left\{ \begin{array}{c} v_j \\ v_{j+1} \end{array} \right\} \tag{3.5.5.3}$$

To conserve momentum, Ψ is advected with the transport masses.

$$\Psi^+_{(m,j+1/2)} = \frac{M^-_{j+1/2}\Psi^-_{(m,j+1/2)} + \Psi_{(m,j-1)}\tilde{f}_{j-1} - \Psi_{(m,j+1)}\tilde{f}_{j+1}}{M^+_{j+1/2}} \tag{3.5.5.4}$$

$$v_j = \frac{1}{2M_j} \left[M_{j+1/2} \quad M_{j-1/2} \right] \left\{ \begin{array}{c} \Psi_{(1,j+1/2)} \\ \Psi_{(2,j-1/2)} \end{array} \right\} \tag{3.5.5.5}$$

3.5.6 Dispersion errors. A von Neumann analysis [137] 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 ξ is the wave number.

$$\begin{aligned} &\frac{\partial \phi}{\partial t} + c\frac{\partial \phi}{\partial x} = 0 \\ &\phi(x,t) = e^{i(\omega t - \xi x)} \end{aligned} \tag{3.5.6.1}$$

For Eq. (3.5.6.1), the dispersion equation is $\omega = c\xi$, but for discrete approximations of the equation and for general hyperbolic equations, the relation is $\omega = \omega(\xi)$. The phase velocity, c_p, and the group speed, c_g, are defined by Eq. (3.5.6.2).

$$\begin{aligned} c_p &= \frac{\omega(\xi)}{\xi} \\ c_g &= \frac{\partial \omega(\xi)}{\partial \xi} \end{aligned} \tag{3.5.6.2}$$

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 Eq. (3.5.6.3) results in a complex dispersion equation, Eq. (3.5.6.4), where $\mathcal{P}$ is a complex polynomial.

$$\phi_j^{n+1} = \phi_j^n + \mathcal{F}(c, h, J, \ldots, \phi_{j-1}^n, \phi_j^n, \phi_{j+1}^n, \ldots) \tag{3.5.6.3}$$

$$\begin{aligned} e^{i\omega h} &= 1 + \mathcal{P}(e^{i\xi J}) \\ \mathcal{P}(e^{i\xi J}) &= \sum_j \beta_j e^{i\xi j J} \text{ for the appropriate range of } j \end{aligned} \tag{3.5.6.4}$$

The dispersion equation has the general form given in Eq. (3.5.6.5), where $\mathcal{P}_r$ and $\mathcal{P}_i$ denote the real and imaginary parts of $\mathcal{P}$, respectively.

$$\omega h = \tan^{-1}\left(\frac{\mathcal{P}_i}{1 + \mathcal{P}_r}\right) \tag{3.5.6.5}$$

Recognizing that the relations in the above equations are periodic in ωh and ξJ, the normalized frequency and wave number are defined to simplify the notation.

$$\overline{\omega} = \omega h \qquad \overline{\xi} = \xi J \tag{3.5.6.6}$$

The von Neumann analysis of the SALE algorithm proceeds by first calculating the increment in the cell momentum.

$$\begin{aligned} P^n_{j+1/2} &= \tfrac{1}{2}(v^n_j + v^n_{j+1}) \\ &= \tfrac{1}{2}(1+e^{-i\overline{\xi}})v^n_j \\ \Delta P^{n+1}_{j+1/2} &= P^{n+1}_{j+1/2} - P^n_{j+1/2} \\ &= \tfrac{1}{2}(1+e^{-i\overline{\xi}})\mathcal{P}v^n_j \end{aligned} \tag{3.5.6.7}$$

The velocity is updated from the changes in the cell momentum.

$$\begin{aligned} v^{n+1}_j &= v^n_j + \tfrac{1}{2}(\Delta P_{j+1/2} + \Delta P_{j-1/2}) \\ e^{i\overline{\omega}}v^n_j &= \tfrac{1}{4}(1+e^{i\overline{\xi}})(1+e^{-i\overline{\xi}})\mathcal{P}v^n_j \\ &= \tfrac{1}{2}(1+\cos(\overline{\xi}))\mathcal{P}v^n_j \end{aligned} \tag{3.5.6.8}$$

The dispersion relation for the SALE advection algorithm is given by Eq. (3.5.6.9).

$$\overline{\omega} = \tan^{-1}\left(\frac{\frac{1}{2}(1+\cos(\overline{\xi}))\mathcal{P}_i}{1+\frac{1}{2}(1+\cos(\overline{\xi}))\mathcal{P}_r}\right) \tag{3.5.6.12}$$

By comparing Eq. (3.5.6.5) and Eq. (3.5.6.12), the effect of the SALE momentum advection algorithm on the dispersion is to introduce a factor $\mathcal{C}$, equal to $\frac{1}{2}(1+\cos(\overline{\xi}))$, into the spatial part of the advection stencil. For small values of $\overline{\xi}$, $\mathcal{C}$ is close to one, and the dispersion characteristics are not changed, but when $\overline{\xi}$ 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 $\mathcal{C}$ has the form $\frac{1}{4}[1+\cos(\overline{\xi})+\cos(\overline{\eta})+\cos(\overline{\xi})\cos(\overline{\eta})]$.

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 inversion error and the dispersion problem are linked. Algorithms that fall into the same general class as the SHALE and HIS algorithms will, therefore, not have dispersion problems [133].

The dispersion error in the SALE algorithm is illustrated in Fig. 24, which shows the result of advecting a square pulse with the van Leer MUSCL algorithm. The high frequency waves remain at the original position of the pulse and the lower frequency trailing oscillations show that that the dispersion error is substantial. Since the error can be interpreted as an inversion error, an exact inverse, Eq. (3.5.613), was derived for this problem.

$$v_{j+1} = 2P_{j+1/2} - v_j \tag{3.5.6.13}$$

While the dispersion error is eliminated, there are oscillations along the top of the pulse due to the nonlinearity of the MUSCL algorithm. When a linear advection method is used, the solutions obtained by the SALE algorithm and the staggered mesh algorithm are identical.

The loss of monotonicity with the SHALE algorithm is illustrated in Fig. 25. This problem can be eliminated by using special limiters [135]. On a uniform mesh, the HIS and staggered mesh algorithms give identical solutions. A nonuniform mesh, with the mesh spacing given by Eq. (3.5.6.14), is used in comparing the staggered mesh and HIS algorithms in Fig. 26. The two solutions are virtually identical and demonstrate that they alter the underlying advection algorithms the least.

$$\triangle x_i = 1.0 + 0.8\sin(\frac{2\pi}{60}(i-20)) \tag{3.5.6.14}$$

3.5.7 Two-dimensional momentum advection algorithms. The momentum advection algorithms discussed in the previous sections are extended to two dimensions in a straight forward manner, but the expressions can become quite lengthy.

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 Eq. (3.5.7.1).

$$\tilde{f}_{(k+1/2,\ell)} = \tfrac{1}{4}(\tilde{f}_{(k+1,\ell-1/2)} + \tilde{f}_{(k+1,\ell+1/2)} + \tilde{f}_{(k,\ell-1/2)} + \tilde{f}_{(k,\ell+1/2)}) \tag{3.5.7.1}$$

The SALE advection algorithm calculates the average momentum of the element from the four velocities at the nodes and distributes one fourth of the change in momentum to each node.

$$\begin{aligned} p_{(k+1/2,\ell+1/2)} =& \tfrac{1}{4}\rho_{(k+1/2,\ell+1/2)}(v_{(k,\ell)} + v_{(k+1,\ell)} + v_{(k+1,\ell+1)} + v_{(k,\ell+1)}) \\ P_{(k+1/2,\ell+1/2)} =& \tfrac{1}{4}M_{(k+1/2,\ell+1/2)}(v_{(k,\ell)} + v_{(k+1,\ell)} + v_{(k+1,\ell+1)} + v_{(k,\ell+1)}) \end{aligned} \tag{3.5.7.2}$$

$$\begin{aligned} v^+_{(k,\ell)} = \frac{M^-_{(k,\ell)}}{M^+_{(k,\ell)}} v^-_{(k,\ell)} + \frac{1}{4M^+_{(k,\ell)}} [&\triangle P_{(k-1/2,\ell-1/2)} + \triangle P_{(k+1/2,\ell-1/2)} \\ &+\triangle P_{(k+1/2,\ell+1/2)} + \triangle P_{(k-1/2,\ell+1/2)}] \end{aligned}$$

The original SHALE algorithm [136] advected three quantities: the average element momentum and the momentum gradients in the two coordinate directions. It was independently discovered by Margolin [135] and Benson [133] that a fourth momentum component, which corresponds to a cross derivative or hourglass momentum term, is necessary in order to eliminate the dispersion problems completely. While three terms eliminate the dispersion problems in the coordinate directions, extra dispersion is introduced in the diagonal directions.

$$\begin{aligned} P(x,y) =& P_{(k+1/2,\ell+1/2)} + S^x_{(k+1/2,\ell+1/2)}(x - x_{(k+1/2,\ell+1/2)}) \\ &+ S^y_{(k+1/2,\ell+1/2)}(y - y_{(k+1/2,\ell+1/2)}) \\ &+ S^{xy}_{(k+1/2,\ell+1/2)}(x - x_{(k+1/2,\ell+1/2)})(y - y_{(k+1/2,\ell+1/2)}) \end{aligned} \tag{3.5.7.3}$$

For the incremental update formula, which involves sixteen terms from the four surrounding elements, the interested reader is referred to Ref. [133].

The HIS algorithm is also readily extended to two dimensions. The defining matrices are given in Eq. (3.5.7.4) and Eq. (3.5.7.5).

$$\left\{ \begin{matrix} \Psi_{(1,k+1/2,\ell+1/2)} \\ \Psi_{(2,k+1/2,\ell+1/2)} \\ \Psi_{(3,k+1/2,\ell+1/2)} \\ \Psi_{(4,k+1/2,\ell+1/2)} \end{matrix} \right\} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \left\{ \begin{matrix} v_{(k,\ell)} \\ v_{(k+1,\ell)} \\ v_{(k+1,\ell+1)} \\ v_{(k,\ell+1)} \end{matrix} \right\} \tag{3.5.7.4}$$

$$v_{(k\ell)} = \frac{1}{4M_{(k,\ell)}} \begin{bmatrix} M_{(k+1/2,\ell+1/2)} \\ M_{(k+1/2,\ell-1/2)} \\ M_{(k-1/2,\ell-1/2)} \\ M_{(k+1/2,\ell-1/2)} \end{bmatrix}^t \left\{ \begin{matrix} \Psi_{(1,k+1/2,\ell+1/2)} \\ \Psi_{(2,k+1/2,\ell-1/2)} \\ \Psi_{(3,k-1/2,\ell-1/2)} \\ \Psi_{(4,k+1/2,\ell-1/2)} \end{matrix} \right\} \tag{3.5.7.5}$$

Despite the simplicity of this algorithm, it works well even on distorted meshes (see Fig. 27a). In contrast, the dispersion errors of the SALE algorithm are readily apparent even on a uniform mesh (Fig. 27b).

3.5.8 Momentum advection with edge-centered velocities. The staggered mesh approach seems to be favored for codes with edge-centered velocities. Since the advection is performed in alternating sweeps, it is sufficient to consider the advection of $\dot{x}_1$ and $\dot{x}_2$ in the x_1 direction. The advection of $\dot{x}_1$ involves no more than the one-dimensional algorithm discussed in section 3.5.2. To advect $\dot{x}_2$ in the x_1 direction, the transport masses must be defined.

Considering $\dot{x}_2$ centered at $(k+1/2,\ell)$, its associated mass is $\frac{1}{2}(M_{(k+1/2,\ell+1/2)} + M_{(k+1/2,\ell-1/2)})$. Invoking the consistency condition, the masses at (k,ℓ) and $(k+1,\ell)$ are calculated using Eq. (3.5.8.1).

$$\begin{aligned} \tilde{f}_{(k,\ell)} &= \tfrac{1}{2}(\tilde{f}_{(k,\ell+1/2)} + \tilde{f}_{(k,\ell-1/2)}) \\ \tilde{f}_{(k+1,\ell)} &= \tfrac{1}{2}(\tilde{f}_{(k+1,\ell+1/2)} + \tilde{f}_{(k+1,\ell-1/2)}) \end{aligned} \tag{3.5.8.2}$$

3.5.9 Momentum advection for Godunov methods. One of the advantages of Godunov methods is the velocity is centered with all the other solution variables at the element centroid. The momentum is advected like any other variable in a conservative manner. While this section is trivial in some sense (no new algorithms), it does point out a nontrivial advantage of the Godunov methods.

3.6 ADVECTING ENERGY

Either the internal energy, e, or the total energy, $e + \frac{1}{2}u_i u_i$, can be advected. The internal energy is used in the evaluation of the equation of state, and most codes advect it like any other conserved solution variable (keeping in mind that it must be converted to energy per unit volume if transport volumes are used). While the internal energy is conserved during advection, the momentum advection does not conserve the kinetic energy, with the consequence that the total energy usually decreases with time. The decrease in kinetic energy corresponds to the diffusion of the velocity through the mesh. An algorithm that preserved both total momentum and kinetic energy would probably not be monotonic.

An alternative strategy, which is used in CAVEAT [3], CTH [63], and PISCES [2], is to advect the total energy and calculate the internal energy in an element as the difference between the total energy and the kinetic energy. When the total energy is advected, anomalous pressures can result [138]. The internal energy, calculated as the difference of two monotonic functions, is not necessarily monotonic. Note that this is the same reason that the SHALE momentum advection algorithm is not monotonic. Another problem with advecting the total energy is the unrealistic heating of materials [3], [63]. As the momentum is diffused by the advection algorithm, the kinetic energy goes down, resulting in an increase in the internal energy.

Advecting the internal energy is not a complete solution. McGlaun [63] has found in CTH that the pressure and temperature oscillations can occur in problems involving large compressions. To eliminate this problem, he makes the internal energy a linear function of the density, forcing the density and internal energy to fall on a line defined in the state space of the adjacent elements.

If the total energy is not conserved, then shocks can weaken and other problems can occur. Some codes selectively convert losses in the kinetic energy into internal energy to avoid this problem. The selectivity is important because it reduces the problems that occur when the total energy is advected. McGlaun [63] uses the analogy of an inelastic collision, and converts the kinetic energy discrepancy into internal energy when the momentum transported into an element has the opposite sign of the element momentum. He reports that this gives the best results when compared to either always turning the kinetic energy discrepancy into internal energy or advecting the kinetic energy and calculating the velocity from it and the sign of the previous momentum. Christensen [51] converts a user-specified fraction of the kinetic energy discrepancy to internal energy only if the element is being shocked (the shock viscosity is on) and the kinetic energy discrepancy will increase the internal energy. This algorithm reduces the loss of total energy at shock fronts (while preventing shock overheating), but it does not attempt to enforce a global conservation of total energy.

3.7 ADVECTING HISTORY VARIABLES

Most nonlinear constitutive models for the strength of materials use a set of history variables to represent the current state of the material. For instance, J_2 plasticity models typically include the plastic strain, ϵ_p, and the backstress, α_{ij}. The deviatoric stress is constrained to lie on or interior to a yield surface that is defined in terms of these history variables. There is no guarantee that the deviatoric stress satisfies the yield surface constraint after advection.

The simplest strategy is to let the constitutive model resolve the discrepancy during the next Lagrangian step. This is the approach followed in the DYNA2D ALE calculations [103]. The material transport during any particular step is so low that the inconsistency between the yield stress and the current stress state is small. The Taylor impact test was used in a series of numerical experiments and no significant difference was found between doing nothing and projecting the stress on to the yield surface.

Predebon, Anderson, and Walker [110] have modified CSQ to return the stress to the yield surface after the advection and to update the plastic strain. The stress is first projected onto the yield surface. Their algorithm then back calculates the stress state at the previous time step based on the updated values of the velocity and the stress. The plastic strain is updated to enforce the consistency condition by calculating the plastic strain rates from the total strain rate and the stress rate. In their Taylor bar impact calculations they found that less than one percent of the total plastic strain was caused by their algorithm, indicating that, like DYNA2D, the stresses in the CSQ calculations were never very far outside the yield surface.

In PISCES [2], Hancock updates the stress deviators after their advection with the remapped strain rates instead of updating them after the Lagrange step. This eliminates the problem of any inconsistency between the stress and the yield surface by construction. Most of the problems solved with the Eulerian code in PISCES are dominated by the pressure, and Hancock reduces his storage and computational costs by advecting the stress deviator with a first order upwind advection algorithm.

All the algorithms discussed in this section are particular to J_2 plasticity. The same ideas could be applied to other constitutive models such as the cap model for soils [140]. Many models do not have a yield surface so there is no consistency problem and their history variables are advected in a conservative manner. The fact that the test calculations, using both DYNA2D and CSQ, showed only small differences between the stress and the yield surface after advection suggests that the errors introduced by advecting the history variables is usually small for calculations involving low to moderate velocities.

3.8 INTERFACE TRACKING

Eulerian codes allow several materials within a single element. Elements with more than one material are referred to as “mixed elements.” The amount of each material in an element must be known, and rules must be established to determine how each material is advected. These rules, either implicitly or explicitly, track the interface between materials. A review of the different tracking methods is given in Ref. [141] and [142]. Most of the methods are based on tracking representations of the surface with marker particles or by deriving the surface definitions from the volume fractions of the different materials.

The interface tracking problem is largely geometrical because there is only one velocity field for all the materials within an element. This implies that slip is impossible on material interfaces within a mixed element, i.e., there are no slide lines in mixed elements. In problems involving the flow of gases next to solids, a boundary layer with a nonzero shear stress occurs in the solids, and the velocity profile of the gas is altered because of the artificial sticking condition within the mixed elements.

3.8.1 Donor cell methods. The volume fraction methods are the most popular approach for tracking the material interfaces in current production Eulerian hydrocodes. Associated with each element is a list of the materials in the element and the fraction of the total volume they each occupy. The sum of the volume fractions, V_i, must equal one, where, by definition, the empty region of an element is associated with a "void material" that has the volume fraction V_0. The volume of material i, $V_i V$, is $\overline{V}_i$.

The simplest algorithms do not explicitly construct material interfaces. The fraction of each material in a transport volume between elements is determined by an averaging procedure. While these algorithms are efficient, they tend to smear the material interfaces. The smearing can be characterized by the number of mixed elements that separate two materials. The best algorithms can resolve a material layer with a volume fraction less than one inside a single element, while the simplest methods may have several mixed elements separating two pure materials. Aside from the decreased accuracy associated with an inaccurate representation of the material interfaces, the cost of the calculation increases. Mixed elements require more calculations than pure elements because of the extra complexity that even the simplest mixture theories introduce. In addition, most codes do not vectorize the mixed element calculations, which increases their relative cost by a factor of five or more. Complicated interface tracking algorithms can increase the overall speed of a code if they minimize the number of mixed elements.

Simple volume fraction algorithms are used in PISCES [2], TOIL [143], and the original versions of CSQ [62] and JOY [100]. The descriptions for the VOF (Volume of Fluid) [145] interface algorithms that are in the literature almost universally use volume coordinates and reduce the element to a unit square. The VOF approach is illustrated with the PISCES algorithm, which is a generalization of the TOIL algorithm. The upstream and downstream volume fractions are denoted by V_i^u and V_i^d, respectively, and the volume transported between the elements, which is to be calculated, is f_i. Similarly, the upstream and downstream element volumes are V^u and V^d. The first stage of the algorithm considers only the materials that the upstream and downstream elements have in common. Each of the common materials is advected in proportion to the downstream volume fraction unless that would advect an amount greater than the volume fraction in the upstream element. The summation in Eq. (3.8.1.1) is restricted to the common materials.

$$f_i^1 = \min(V_i^u V^u, f V_i^u / \sum_j V_j^d) \tag{3.8.1.1}$$

If some of the materials are exhausted during the evaluation of Eq. (3.8.1.1), then the total of the f_i will be less than f. To make up the difference, Eq. (3.8.1.1) is applied again with the summation restricted over the remaining common materials, where $\tilde{f}$ is the remaining transport volume.

$$f_i^2 = \min(V_i^u V^u, \tilde{f} V_i^u / \sum_j V_j^d) \tag{3.8.1.2}$$

This equation is repeated until the total transport volume is accounted for or until there are no remaining untransported common materials. The volume associated with each common material is the sum of the increments calculated in Eq. (3.8.1.2) and Eq. (3.8.1.3).

$$\begin{aligned} f_i &= \sum_k f_i^k \\ \tilde{f} &= f - \sum_i f_i \end{aligned} \tag{3.8.1.3}$$

In the second stage, the remaining materials in the upstream element are used to fill the remaining volume, where the summation is restricted to the remaining materials in Eq. (3.8.1.4).

$$f_i = \tilde{f} V_i^u / \sum_j V_j^d \tag{3.8.1.4}$$

The original version of JOY [100] employed a similar strategy. JOY used an average value of the volume fraction to determine the value in the flux of the common materials.

$$f_i = \min(V_i^u V^u, \tfrac{1}{2}(V_i^u + V_i^d) f) \tag{3.8.1.5}$$

When the common materials are depleted, materials are chosen to fill the remaining volume fraction based on how close they were to the common materials during the original definition of the problem. For example, if material 2 is a common material and it is depleted from the upstream element, then the algorithm tries to advect the material from the upstream element that was originally on the boundary of material 2. If that material is not present in the element, then it searches backward material boundary by material boundary to find the closest material.

The next step up in complexity is to reconstruct the material interfaces within an element based on which adjacent elements contain common materials. In this approach, only two materials are considered at a time: the material of current interest, and the “other” material, which combines all the remaining materials in the element. An example of this approach is used in BBC [146], the precursor to the SLIC [147] algorithm. Although the basic ideas behind this algorithm are quite simple, the algorithm is quite complicated: almost half of the description of BBC is devoted to described the reconstruction of the interface geometry from the volume fractions.

The four adjacent elements are searched for the material under consideration and the result is either 0 for no and 1 for yes, resulting in a 4 bit code for the 30 possible configurations. The thirty configurations are reduced to eight types, three of which have subcases, which are finally divided into three classes. The first class assumes the materials occur in horizontal layers, the second class assumes vertical layers, and the third class contains the exceptions, most of which are materials concentrated at the corners or midsides of elements. The thickness of the layers is proportional to the volume fraction of the material. When a material is concentrated in a corner or an edge, it is assumed to have the same aspect ratio as the element.

Based on the geometry of the reconstructed interface for each material and the geometry of the transport volume (i.e., the volume and which edge), the amount of each material transported is calculated. Since each material is considered independently, it is possible that the materials overlap and that the sum of the individual material contributions do not equal the actual volume to be transported. The volume for the last material is chosen to be the difference between the total transport volume and the contributions from all the other materials. The last volume, however, must be positive and less than the amount of the last material in the upstream element. Additional steps are performed if these last two conditions are not satisfied. If the amount of the material is too large, then the individual volumes are scaled back equally.

$$f_i = f_i(\sum_k f/f_k) \tag{3.8.1.6}$$

The volume scaling is limited by the amount of each material available in the upstream element. Additional scaling is performed to adjust the volumes of the different materials that are not exhausted, and if the scaling does not work, then the actual volume transported between elements is reduced.

3.8.2 The SLIC algorithm. Probably the best first order interface tracking algorithm is SLIC (Simple Line Interface Calculation) [147], which was developed by Noh and Woodward. It is an alternating direction method that constructs the fluid interfaces out of straight lines that are parallel or perpendicular to the advection direction. In a manner similar to the BBC algorithm, the interfaces are completely determined by testing whether or not adjacent elements contain the various materials. A major advantage of the SLIC algorithm over the previous algorithms is it determines the volume of each material in a single pass.

Following Noh and Woodward, the fluid occupation numbers for material i , IL_i, and IR_i, are 0 if material i is absent and 1 if it is present in the elements to the left and right of the current element respectively. Materials having the same combination of (IL_i, IR_i) out of the four possible combinations are considered to be members of the same fluid group and are treated equally. The six possible configurations of the four possible fluid groups are shown in Fig. 28 from Ref. [147], where the sizes of the component rectangles are determined by their fractional volumes. During the calculation of the volumes, the materials in a particular fluid group are considered to be perfectly mixed. When only two materials are in an element, one of the materials has the occupation numbers (0,1) or (1,0) as shown in case b in Fig. 28. Regardless of the occupation numbers of the other fluid, the interface is defined as in case b to prevent excessive diffusion. Note that the interfaces are constructed based on information in one direction and that the interfaces will change depending on the current advection direction.

The SLIC algorithm was enhanced by Chorin [99]. His first modification was the introduction of a corner algorithm which uses information from both mesh directions. To illustrate his algorithm, consider the case where element $(i+1/2, j+1/2)$, $(i+1/2, j-1/2)$, and $(i-1/2, j+1/2)$ have a common material k. This material occupies the lower left hand corner of the element in both SLIC and in Chorin's extension of SLIC. Chorin defines the shape of the rectangle in element $(i+1/2, j+1/2)$, having a horizontal length a and a vertical height b, in terms of the volume fractions from the adjacent elements.

$$\begin{aligned} ab &= V_{k(i+1/2,j+1/2)} \\ \frac{b}{a} &= \frac{V_{k(i-1/2,j+1/2)}}{V_{k(i+1/2,j-1/2)}} \end{aligned} \tag{3.8.2.1}$$

If Eq. (3.8.2.1) predicts a value for a or b that is greater than the element dimensions, then a or b is limited to its maximum value and the other dimension is determined by conservation.

When there are only two materials, and one of them has the occupation numbers (0,0), it is located in the center of the element. Unless the Courant number is sufficiently large, the material remains trapped in the element forever. Chorin bases his positioning on the Glimm construction [149], where the volume centroid, V_k^c, of material k is located between $-\frac{1}{2}$ and $+\frac{1}{2}$ in volume coordinates according to a parameter θ. A suitable sequence of values for θ can be calculated in a number of ways, but θ must be equidistributed on the interval [0,1].

$$V_k^c = (1 - V_{k(i+1/2,j+1/2)})\theta - \tfrac{1}{2}(V_{k(i+1/2,j+1/2)}) \qquad (3.8.2.2)$$

3.8.3 Higher order VOF methods. The most sophisticated of today's VOF methods represent the interfaces between materials as arbitrarily oriented straight line segments within each element. The interfaces between the elements are not, however, necessarily continuous. Among the programs that have taken this approach are CAVEAT [138], KRAKEN [102], SOLA-VOF [145], PELE [150], MESA [101], and CTH [63]. The major differences between the algorithms are the calculation of the slope and the ordering of the materials for the slope calculations.

The calculation of the slopes is generally performed by one of two methods. KRAKEN and PELE use the volume fractions of a material from two adjacent elements, see Fig. 29. A line is drawn through the two elements such that the volume beneath the line equals the material volume in each element. Each element edge, therefore, has its own set of interfaces that are common to two elements, and that set of interfaces is used regardless of the direction of the transport across the edge. Referring to Fig. 29, the two elements are ordered such that the one on the right has the larger volume fraction, and variables are subscripted S or L for the small and large volume fraction elements respectively. Volume coordinates are used: the height of the elements is one and the volume widths are X_S and X_L. The interface is defined by the slope of the line, s, and the intercept on the common edge, b. Letting f be positive if it is transporting material from the left to the right element, the volume of the current material advected, f_i, is given by Eq. (3.8.3.1).

$$f_i = bf - \tfrac{1}{2}sf^2 \qquad (3.8.3.1)$$

There are four different configurations that can occur depending on the relative volume fractions, see Fig. 29, and the interface definition is not necessarily unique. To avoid smearing the interface, the definition with the steepest slope is chosen and the cases in Fig. 29 are ordered with the steepest slope first.

For the first case, the slope and intercept are given by Eq. (3.8.3.2) and they must satisfy the three constraints in Eq. (3.8.3.3).

$$\begin{aligned} a &= \sqrt{\frac{X_S V_S}{X_L(1-V_L)}} \\ b &= \frac{a}{1+a} \\ s &= \frac{b^2}{2X_S V_S} \end{aligned} \tag{3.8.3.2}$$

$$\begin{aligned} & X_L(1-V_L) - \frac{(1-b)^2}{2s} = 0 \\ & \frac{(1-b)}{s} \leq X_L \\ & \frac{2X_S V_S}{b} \leq X_S \end{aligned} \tag{3.8.3.3}$$

The second case is defined by Eq. (3.8.3.4) and Eq. (3.8.3.5).

$$\begin{aligned} b &= \frac{2}{X_L}\left\{\sqrt{X_S^2 V_S^2 + X_S V_S X_L V_L} - X_S V_S\right\} \\ s &= \frac{b^2}{2X_S V_S} \end{aligned} \tag{3.8.3.4}$$

$$\begin{aligned} & Y_L \equiv b + sX_L \ \ : b \leq Y_L \leq 1 \\ & \frac{2X_S V_S}{b} \leq X_S \end{aligned} \tag{3.8.3.5}$$

Case 3 is similar to case 2, but with the materials switched. By switching the materials, Eq. (3.8.3.6) and Eq. (3.8.3.7) are readily derived from Eq. (3.8.3.4) and Eq. (3.8.3.5) respectively.

$$\begin{aligned} b &= 1 - \frac{2}{X_S}\left\{\sqrt{X_L^2(1-V_L)^2 + X_S(1-V_S)X_L(1-V_L)} - X_L(1-V_L)\right\} \\ s &= \frac{(1-b)^2}{2X_L V_L} \end{aligned} \tag{3.8.3.6}$$

$$\begin{aligned} &Y_S \equiv b - sX_S \ \ : 0 \leq Y_S \leq b \\ &\frac{2(1-V_L)X_L}{(1-b)} \leq X_L \end{aligned} \tag{3.8.3.7}$$

The fourth case is defined by Eq. (3.8.3.8) and Eq. (3.8.3.9).

$$\begin{aligned} s &= \frac{2(V_L - V_S)}{X_L + X_S} \\ b &= \frac{V_S X_L + V_L X_S}{X_L + X_S} \end{aligned} \tag{3.8.3.8}$$

$$\begin{aligned} b - sX_S \geq 0 \\ b + sX_L \leq 1 \end{aligned} \tag{3.8.3.9}$$

A different strategy for finding the slopes was proposed by David Youngs [151] in two dimensions. Youngs later created a similar method for three-dimensional calculations [152] that is very similar to the one currently used in CTH [63]. Referring to Fig. 29, side fractions, $b_{(i,j+1/2)}$, are calculated for each edge of an element using the two adjacent elements and Eq. (3.8.3.2) through (3.8.3.9). Note that the element is normalized to a unit square. The edge fractions are differenced to calculate an interface slope.

$$s_{(i+1/2,j+1/2)} = \frac{b_{(i+1,j+1/2)} - b_{(i,j+1/2)}}{b_{(i+1/2,j+1)} - b_{(i+1/2,j)}} \tag{3.8.3.10}$$

The position of the interface is adjusted to match the volume fractions of the two materials. If the actual material interface is a straight line, then the calculations of the side fractions is exact, and the exact material interface is calculated.

Since the concept of a side fraction is not readily generalized to three dimensions, Youngs calculates the normal to the material interface by differencing the volume fractions in the surrounding elements, and adjusts the interface position to match the volumes of the two materials. For a uniform mesh, Youngs uses Eq. (3.8.3.11) to difference the slope with α set to 2.

$$\begin{aligned} \frac{\partial f}{\partial x_1} &= \frac{f_E - f_W}{2\triangle x_1} \\ \frac{\partial f}{\partial x_2} &= \frac{f_N - f_S}{2\triangle x_2} \end{aligned}$$

$$
\begin{aligned}
f_E &= \frac{f_{(i+3/2,j+3/2)} + \alpha f_{(i+3/2,j+1/2)} + f_{(i+3/2,j-1/2)}}{2+\alpha} \\
f_W &= \frac{f_{(i-1/2,j+3/2)} + \alpha f_{(i-1/2,j+1/2)} + f_{(i-1/2,j-1/2)}}{2+\alpha} \\
f_N &= \frac{f_{(i-1/2,j+3/2)} + \alpha f_{(i+1/2,j+3/2)} + f_{(i+3/2,j+3/2)}}{2+\alpha} \\
f_S &= \frac{f_{(i-1/2,j-1/2)} + \alpha f_{(i+1/2,j-1/2)} + f_{(i+3/2,j-1/2)}}{2+\alpha}
\end{aligned}
\tag{3.8.3.11}
$$

Youngs' algorithm was extended by Johnson [172] to nonorthogonal meshes and axisymmetric geometries in CAVEAT. The differences in Eq. (3.8.3.11) are evaluated with the logical coordinates, L_i, and then transformed to the physical coordinate system by inverting Eq. (3.8.3.12)

$$
\left\{ \begin{matrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{matrix} \right\} = \begin{bmatrix} \frac{\partial L_1}{\partial x_1} & \frac{\partial L_2}{\partial x_1} \\ \frac{\partial L_1}{\partial x_2} & \frac{\partial L_2}{\partial x_2} \end{bmatrix} \left\{ \begin{matrix} \frac{\partial f}{\partial L_1} \\ \frac{\partial f}{\partial L_2} \end{matrix} \right\}
\tag{3.8.3.12}
$$

While the location of the interface for planar problems can be calculated explicitly, the location in axisymmetric problems is described by a cubic equation which is solved by Newton iteration.

Somewhat simpler difference expressions were proposed earlier by Hirt and Nichols for SOLA-VOF [145]. For a rectangular mesh, the height of a particular material above the bottom edge of a row of elements is defined by Eq. (3.8.3.12), then the slope is calculated with Eq. (3.8.3.13), where $\triangle y$ and $\triangle z$ are the element dimensions.

$$
\begin{aligned}
Z_{i+1/2} =& f_{(i+1/2,j-1/2)} \triangle z_{(i+1/2,j-1/2)} + \\
& f_{(i+1/2,j+1/2)} \triangle z_{(i+1/2,j+1/2)} + \\
& f_{(i+1/2,j+3/2)} \triangle z_{(i+1/2,j+3/2)}
\end{aligned}
\tag{3.8.3.12}
$$

$$
\frac{\partial Z}{\partial y}\Big|_{(i+1/2,j+1/2)} = \frac{2(Z_{(i+3/2)} - Z_{(i-1/2)})}{\triangle y_{(i-1/2,j+1/2)} + 2\triangle y_{(i+1/2,j+1/2)} + \triangle y_{(i+3/2,j+1/2)}}
\tag{3.8.3.13}
$$

If the material interface is determined to be almost vertical, the slope $\partial Y/\partial z$ is calculated using an expression similar to Eq. (3.8.3.13). As in Youngs' algorithm, once the slope is calculated, its position is adjusted to match the volume fractions in element $(i+1/2, j+1/2)$.

McGlaun and Thompson [63] use a contouring algorithm to calculate the material interface. At each element corner, a volume fraction is calculated by averaging the volume fraction from the four surrounding elements. The family of possible material interfaces is the set of straight contour lines through the element parameterized by the contour level. The particular contour line that is used for the interface is the one that matches the material volumes.

3.8.4 Handling more than two materials. The VOF algorithms which explicitly calculate interfaces, with the exception of SLIC, are described in terms of two materials. Extending the algorithms to handling more materials is nontrivial. With three materials, Y intersections can occur and the material interfaces calculated by the VOF algorithms can intersect. Only the SLIC algorithm has the topological flexibility to represent Y or T intersections, while the other algorithms simply layer the materials.

Two strategies are used in calculating multiple material interfaces. The first is to calculate the interface between each material and the mixture of all other materials in a completely independent manner. The second strategy is more complicated to explain, but only a little more difficult to implement. The materials in the element are ordered in some manner, with the precise ordering being dependent on the code. The interface between the first material and the remaining materials is determined. The volume fractions of the first and second material are added to generate a mixed material, and the interface between the mixture and the remaining materials is calculated. The layer of the second material is defined by the two interfaces. In a similar manner, the layer for material i is defined by the previous interface and the interface between the mixture of materials 1 through i and the remaining materials. Youngs requires the user to specify the order of the materials in the layers as input [151], while McGlaun orders the materials by decreasing volume fraction in the downstream element. While building the interfaces layer by layer does not eliminate the problem of intersecting interfaces, it does minimize the problem and layers of material less than an element wide can be tracked accurately.

When two interfaces intersect, the interfaces can either be adjusted to eliminate the intersection or the fluxes can, if necessary, be scaled during the advection. The first strategy is fairly straight forward to implement when the interfaces are built up layer by layer. The second strategy follows the same rules used in section (3.8.2). Presumably the interfaces have been calculated using one of the higher order schemes of the previous section, and any alteration of the interfaces corresponds to adding extra diffusion. If the interfaces intersect outside of the volume being advected, then no problem will occur and interfaces and fluxes should remain unchanged. When the interfaces intersect inside the volume flux, the sum of the partial volume fluxes for each material will not equal the total volume flux.

3.8.5 Lagrangian marker particles. One way to represent material interfaces that is not used very much anymore in hydrocodes is Lagrangian marker particles. The particles are initially placed at the interface between materials. Their positions are updated during the Lagrangian step, and the current interface is defined by interpolating between the particles. Hermitian interpolation is sometimes used instead of linear interpolation to reduce the noise in the solution [141]. Particles are added and deleted dynamically to maintain an accurate interface. While this approach is superficially attractive (e.g., no diffusion across the material interfaces, the exact location of the interface is known), it has a number of difficulties. One difficulty is the topology is fixed by the Lagrangian nature of the lines. Complicated logic must be added to handle the changes in the topology that occur in penetration calculations.

Another problem is updating the position of the marker particles. Consider, for example, a particle in the interior of an element that is on the interface between steel and air. When the element is compressed, most of the compression is taken up by the air. An isoparametric interpolation of the velocity at the particle would compress the steel as much as the air, resulting in erroneously high stresses within the steel and incorrect volume fluxes during the advection. To eliminate this problem, some programs such as HELP [95], use a density weighted velocity average over an area surrounding the particle. The area is generally comparable to the area of an element.

Errors in the Lagrangian velocity of the marker particles can lead to the interfaces becoming tangled. Surface tension algorithms can prevent kinks and contact algorithms can prevent interpenetration of the interfaces. Each additional fix, however, reduces the attractiveness of the marker particles.

3.9 COUPLED EULERIAN LAGRANGIAN CALCULATIONS

In some cases it is desirable to represent one part of a problem with a Lagrangian mesh and another part with an Eulerian mesh. Examples of this type of calculation include underwater explosions, where the fluid is Eulerian and the hull is Lagrangian, and low velocity penetration calculations, where the penetrator is Lagrangian and the target is Eulerian. The Lagrangian mesh moves through the Eulerian mesh, and some mechanism is necessary to couple the their responses. Note that this is a different problem than the typical fluid-structure interaction problems described in the engineering literature. The latter case is usually treated by putting a slideline between the fluid and the structure, and the fluid is treated with a simplified ALE formulation. In explicit codes, the general strategy for coupled Eulerian Lagrangian calculations is to apply pressure boundary conditions on the Lagrangian mesh from the Eulerian mesh, and to treat the Lagrangian mesh boundaries as velocity boundary conditions in the Eulerian mesh [2], [150], [9].

3.9.1 The pressure boundary conditions. The Lagrangian boundary overlaps the Eulerian mesh as shown in Fig. 32. Each boundary segment is considered independently and subdivided into a set of subsegments, which each subsegment being defined as the portion of the segment that divides a particular Eulerian element. The force contribution from a particular Eulerian element to a segment is the normal component of the stress acting at the subsegment centroid times the area of the subsegment. The superscript k refers to the kth element contributing to the force acting on the segment, A^k is the area of the kth subsegment, and n is the normal to the boundary segment, and half the force on the segment is distributed to the two Lagrangian nodes defining it.

$$F_i^{seg} = n_\ell \sigma_{\ell j}^k n_j A^k n_i \tag{3.9.1.1}$$

$$F_{\alpha i} = \tfrac{1}{2} n_\ell \sigma_{\ell j}^k n_j A^k n_i N_\alpha(x_k) \tag{3.9.1.2}$$

While the area weighting works well in practice, Noh [9] has pointed out that the boundary is, in general, a contact discontinuity, and that the pressure and the normal component of the velocity are continuous across it. A better treatment might calculate the pressure as a density weighted average of the Eulerian and Lagrangian pressures.

Usually the deviatoric contribution is ignored, and $n_\ell \sigma_{\ell j}^k n_j$ simplifies to $-P^k$. Many codes, such as PELE [150] and CEL [9], assume that the pressure is constant within each Eulerian element, while other codes, such as PISCES [2], use a linear pressure approximation based on the van Leer MUSCL algorithm. In Eq. (3.9.1.3), a subscript c indicates the value at the area centroid of the element. The area centroid is calculated based on the area of the Eulerian element that is not covered by the Lagrangian mesh, see Fig. 32.

$$P(y,z) = P_c + \frac{\partial P}{\partial y}(y - y_c) + \frac{\partial P}{\partial z}(z - z_c) \tag{3.9.1.3}$$

Once the contributions to the boundary forces from the Eulerian mesh are calculated, the Lagrangian mesh is updated in the usual manner.

3.9.2 The velocity boundary conditions. The geometrical quantities associated with the Eulerian element cannot be calculated in the standard manner when it is partially overlapped by a Lagrangian mesh. Instead of the standard difference stencil for the average strain rate, an integral difference is performed over the arbitrarily shaped polygon defined by the intersection of the Eulerian element and the Lagrangian mesh, see Eq. (2.5.3.1), in CEL and PISCES. In PELE [150], the velocities at the covered edges of an element are calculated by requiring that the normal velocity of the Lagrangian boundary equal the fluid velocity in the same direction. After the edge velocities are calculated, the strain rates are calculated in the usual manner. The element volume is calculated by subtracting the Lagrangian volume from the volume of the Eulerian element. Once these quantities have been established, the stress and pressure are updated. The velocities of the uncovered parts of the element are updated in the normal manner.

When the Lagrangian boundary is uncovering an element, initial conditions must be established for the velocity, stress, and history variables. PISCES [2] blends and unblends small, partially covered elements with their adjacent elements to maintain a large time step. When an element is being uncovered, its uncovered area is initially very small and it is blended with its neighbor. When it is large enough not to control the time step, it is unblended. Its initial values are evaluated at its centroid by using a monotonic Taylor expansion about the centroid of the blended elements. CEL [9] also uses blending for the element-centered quantities, but the node-centered velocities are handled in a different manner. The velocity of a covered node in the direction direction of the normal to the boundary is interpolated from the boundary while the tangential velocity is governed by the Eulerian difference equations. As mentioned previously, PELE determines the velocities at the covered edges of an element by requiring that the normal velocity of the boundary and the normal component of the fluid velocity match.

3.10 REMAPPING AND REZONING

One alternative to Eulerian and ALE formulations, which is referred to as “rezoning,” is to allow a Lagrangian calculation to run until the mesh becomes unacceptably distorted, create a new, undistorted mesh, map the solution from the undistorted mesh, and then continue the Lagrangian calculation. The boundary between rezoning and the other formulations is not sharply defined. ALE codes can run in a purely Lagrangian manner or remap the solutions only regionally on a periodic basis. Rezoning algorithms are usually defined as mapping algorithms that operate on two arbitrary meshes while advection algorithms are restricted to topologically identical meshes and have a Courant limit on their stability.

The generality of rezoning algorithms makes them computationally expensive in comparison to advection algorithms, and generally restricts them to first or second order accuracy. As a consequence, they are used infrequently during an otherwise Lagrangian calculation. The simplest rezoning algorithms are based on interpolation and are not conservative, while more expensive algorithms map arbitrary linear approximations from one mesh to another in a conservative manner. The mathematical basis for rezoning algorithms is simple, but their implementation is complicated due to the tedious geometrical calculations associated with handling an arbitrary mesh, e.g., whether or not a point lies within a particular element.

3.10.1 Interpolation. The cheapest and most diffusive method for rezoning is interpolation. An interpolation function is constructed for the distorted mesh and new values are calculated by evaluating the interpolation function at the centroids of the elements in the new mesh. The definition of the interpolation function is not unique, but the most common approach is to use the bilinear shape functions at the nodes to interpolate both the node-centered and element-centered quantities.

The construction of the values of ϕ, an element-centered variable, at the nodes is frequently based on the minimization procedure developed by Zienkiewicz, et al., [157]. Denoting the node-centered values with Greek subscripts, and the element values with upper-case subscripts, the least squares fit is found by minimizing the functional Π.

$$\Pi = \tfrac{1}{2}\sum_I \int_{\Omega_I} (\phi_\alpha N_\alpha - \phi_I)^2 d\Omega \tag{3.10.1.1}$$

Taking the derivative of Π with respect to each ϕ_α generates a set of linear equations.

$$\left[\sum_I \int_{\Omega_I} N_\alpha N_\beta d\Omega\right]\phi_\beta d\Omega = \sum_I \int_{\Omega_I} N_\alpha \phi_I = f_\alpha \tag{3.10.1.2}$$

The term within the square brackets has the same form as the consistent mass matrix, defined in Eq. (2.4.1.6), and it is denoted $\mathcal{M}^c_{\alpha\beta}$. A lumped matrix, $\mathcal{M}^\ell_{\alpha\beta}$ can also be defined in the same manner as for a mass matrix. Using the lumped matrix, values for ϕ_α can calculated in a trivial manner.

$$\phi_\alpha = \frac{f_\alpha}{\mathcal{M}^\ell_{\alpha\alpha}} \quad \text{no sum on } \alpha \tag{3.10.1.3}$$

The values defined by Eq. (3.10.1.3) provide a starting vector for an iterative solution to the fully coupled equations. The superscript on the ϕ refers to the iteration number

$$\phi^{i+1}_\alpha = \frac{f_\alpha - (\mathcal{M}^c_{\alpha\beta} - \mathcal{M}^\ell_{\alpha\beta})\phi^i_\beta}{\mathcal{M}^\ell_{\alpha\alpha}} \quad \text{no sum on } \alpha \tag{3.10.1.4}$$

When the convergence criteria, ϵ, is relatively loose ($\sim 10^{-2}$), approximately five iterations are needed, while tight tolerances ($\sim 10^{-4}$) require approximately forty iterations [18].

$$\frac{||\phi^{i+1}_\alpha - \phi^i_\alpha||}{||\phi^{i+1}_\alpha||} \leq \epsilon \tag{3.10.1.5}$$

Hallquist [18] reports that the difference between the original element-centered values of ϕ and values calculated from the interpolation function average under five percent. Before the new value of ϕ is calculated for an element, the nodes are checked to see if they have been moved during the smoothing of the mesh. If they have not been changed, then the value of ϕ remains unchanged, which reduces the computational cost and avoids introducing any interpolation error into that element.

The new element-centered values in DYNA2D [18] are calculated by integrating the interpolation function with Gaussian quadrature (usually 2×2) and dividing it by the area of the new element.

$$\phi_I = \frac{\int_{\Omega_I} \phi d\Omega}{\Omega_I} \tag{3.10.1.6}$$

The two problems with this approach are the solution variables are not conserved, and the nodal values defined by Eq. (3.10.1.2) may not be monotonic. In practice, the conservation error is usually only a few percent. It can be reduced by increasing the number of Gauss points used in the quadrature, but the order of the integration should not be increased since the interpolation function is only bilinear. In areas where there are steep gradients, first order integration should be used with a large number of integration points. The lack of monotonicity can be easily demonstrated by considering two elements of equal size in one dimension. Letting the left element have a value of ϕ equal to one and the right element, zero, the nodal values for ϕ are 3/2, 1/2, and -1/2 from left to right. In regions around strong shocks, the initial iteration values of ϕ_α defined by Eq. (3.10.1.3) may be preferable to the converged values since the initial values do define a monotonic, if diffusive, interpolation function.

The geometrical challenge associated with this algorithm is calculating the location in the original mesh of a point in the new mesh. As a first step, the element in the original mesh that contains the point must be determined. The algorithm outlined here is used in the DYNA2D rezoner, and it also contains the basic search strategy for locating a slave node on a master surface in three dimensions [75].

To check whether or not an element contains a particular point is computationally expensive, and therefore a simple strategy is used to reduce the number elements that must be checked. Before the rezone is performed, a list of the elements surrounding each node is generated. The list is easily and efficiently generated from the connectivity data. The closest node in the original mesh to the specified point is determined either through a brute force search or a nested bucket sort [64], the latter being much more efficient. The list of elements surrounding the closest node provides an initial list of elements that could contain the specified point. Unless the mesh is unusually distorted, the initial list is sufficient, but the search list can be recursively expanded by considering all the elements surrounding all the nodes that are used to define the elements in the initial (or previous) search list.

The specified point usually lies within an element if Eq. (3.10.1.7) is satisfied. The vector $\vec{s}$ is the vector from the closest node to the specified point, and vectors $\vec{v}_1$ and $\vec{v}_2$ radiate outward from the closest node along the edges of the element, see Fig. 33.

$$(\vec{v}_1 \times \vec{s}) \cdot (\vec{s} \times \vec{c}_2) > 0 \tag{3.10.1.7}$$

After passing this test, the isoparametric coordinates of the point are calculated. If they fall outside the range of [-1,+1], the specified point doesn't lie within the element and the element search must be continued. There are other tests that can be performed to determine whether or not the point is within the element before attempting to calculate its isoparametric coordinates. Milgram [158] provides a concise review of the methods for solving this problem generalized to arbitrary polygons and introduces a method of his own.

The isoparametric coordinates are calculated by eliminating ξ_1 in terms of ξ_2 and solving the resulting quadratic equation. When a quadrilateral element is degenerated to obtain a triangular element, the isoparametric coordinates are degenerate at the two condensed nodes. A logical check must be performed for triangular elements to handle this case.

3.10.2 The completely conservative rezone. Conservation is guaranteed by construction if the new element-centered (or momentum control volume) values of ϕ are defined by Eq. (3.10.1.6) with a conservative interpolation function and the integrals are evaluated exactly. This approach is used by Dukowicz [159], [160] and Hancock [2], but their implementations differ considerably in detail.

Hancock uses a constant value of ϕ over each zone and calculates the volume contributed from each of the old elements that overlaps a particular new element in the mesh. Denoting the values before and after the rezone by a superscript $-$ and $+$ respectively, and the volume from old element J overlapping new element I as V_J^I, the discrete version of Eq. (3.10.1.6) is given by Eq. (3.10.2.1). Note that the sum of V_J^I equals the volume of I, V^I.

$$\phi_I^+ = \frac{\sum_J V_J^I \phi_J^-}{\sum_J V_J^I} \tag{3.10.2.1}$$

The algorithm, which he attributes to Thorn and Holdridge [161], for calculating the overlap is the same one Hancock uses for calculating the volumes of the Eulerian elements in the coupled Eulerian-Lagrangian calculations. Both elements are decomposed into four quadrilaterals, as shown in Fig. 34, by projecting their four edges onto the y axis. The quadrilateral defined by the third and fourth nodes of element I is denoted $V_{4,3}^I$, and the others are denoted in a similar manner. The signed union of the four quadrilaterals equals the area of the associated element. Assuming that the nodes are numbered in a counterclockwise manner, the sign of the quadrilateral $V_{i+1,i}^I$ is the same as the sign of $y_i - y_{i+1}$ so that the upper edges of the quadrilateral always contribute a positive volumes, and the lower, negative volumes.

$$V^I = \bigcup_{i=1}^{4} \mathrm{sgn}(y_i - y_{i+1}) V_{i+1,i}^I \tag{3.10.2.2}$$

The overlap volume is the intersection of V^I and V^J.

$$V_J^I = V_I^J = V^I \bigcap V^J = \bigcup_{i,j=1}^{4} \mathrm{sgn}\big((y_i - y_{i+1})(y_j - y_{j+1})\big) V_{i+1,i}^I \bigcap V_{j+1,j}^J \tag{3.10.2.3}$$

The problem, therefore, is reduced to calculating the intersection of $V_{i+1,i}^I \cap V_{j+1,j}^J$. The two cases of the edges intersecting or not intersecting are identified and handled separately. The components of vector along the edge of element I are $\triangle y^I$ and $\triangle z^I$, and similarly for element J. If $\mathrm{sgn}\big((y_i - y_{i+1})(y_j - y_{j+1})\big)$ is zero, then at least one of the edges is parallel to the z axis and the volume is zero. The left and right boundaries of the overlap are calculated according to Eq. (3.10.2.4).

$$\begin{aligned} y_L &= \max[\min(y_i^I, y_{i+1}^I), \min(y_j^J, y_{j+1}^J)] \\ y_R &= \min[\max(y_i^I, y_{i+1}^I), \max(y_j^J, y_{j+1}^J)] \end{aligned} \tag{3.10.2.4}$$

The values of z at the left and right boundaries of the overlap are the minimums of the two edge values at y_L and y_R because the intersection of the two volumes is being calculated.

$$\begin{aligned} z_L &= \min(m^I(x_L - x_i^I) + z_i^I, m^J(x_L - x_j^J) + z_j^J) \\ z_R &= \min(m^I(x_R - x_i^I) + z_i^I, m^J(x_R - x_j^J) + z_j^J) \\ m^I &= \frac{\triangle z^I}{\triangle y^I} \\ m^J &= \frac{\triangle z^J}{\triangle y^J} \end{aligned} \tag{3.10.2.5}$$

The intersection of the edges does not occur in the region of the overlap if the slopes, m^I and m^J, are equal, or if the intersection occurs outside of $[y_L, y_R]$. The intersection of the lines defined by the edge segments is calculated using Eq. (3.10.2.6).

$$\begin{aligned} y^M &= \frac{(z_j^J - z_i^I + m^I y_i^I - m^J y_j^J)}{m^I - m^J} \\ z^M &= m^I(y^M - y_i^I) + z_i^I \end{aligned} \tag{3.10.2.6}$$

For the nonintersecting case, the volumes are given by Eq. (3.10.2.7), while the volumes for the intersecting case are given by Eq. (3.10.2.8). In the axisymmetric case, the volumes are for a wedge subtending one radian.

$$\begin{aligned} V_{i+1,i}^I \cap V_{j+1,j}^J &= \tfrac{1}{2}(y_R - y_L)(z_R + z_L) \quad \text{planar} \\ &= \tfrac{1}{6}(y_R - y_L)(z_R^2 + z_R z_L + z_L^2) \quad \text{axisymmetric} \end{aligned} \tag{3.10.2.7}$$

$$\begin{aligned} &V_{i+1,i}^I \cap V_{j+1,j}^J \\ &\quad = \tfrac{1}{2}\Big\{(y_M - y_L)(z_M + z_L) + \end{aligned}$$

$$(y_R - y_M)(z_R + z_M)\Big\} \quad \text{planar} \tag{3.10.2.8}$$

$$= \tfrac{1}{6}\Big\{(y_M - y_L)(z_M^2 + z_M z_L + z_L^2) +$$

$$(y_R - y_M)(z_R^2 + z_R z_M + z_M^2)\Big\} \quad \text{axisymmetric}$$

Dukowicz [159] assumed a constant ϕ within an element in his original paper, but he has extended his method to a piecewise linear approximation of ϕ [160]. His method is currently implemented in CAVEAT [3] and CAVEAT-GT [91]. Instead of performing a volume integral, Dukowicz uses the divergence theorem to convert the volume integral to a boundary integral. The reduced dimensionality greatly reduces the complexity of the problem and a careful structuring of the calculations makes this method very efficient.

A flux, F, is defined such that Eq. (3.10.2.9) holds, with the result that the volume and boundary integrals of Eq. (3.10.2.10) are equal by the divergence theorem.

$$\frac{\partial F_i}{\partial x_i} = \phi(x) \tag{3.10.2.9}$$

$$\int_\Omega \phi d\Omega = \oint_\Gamma F_i n_i d\Gamma \tag{3.10.2.10}$$

The flux is not uniquely defined by Eq. (3.10.2.9), and Dukowicz discusses a number of alternatives. In his original paper, he chose a flux parallel to the i lines in a logically regular mesh. In two dimensions, the divergence theorem is written as Eq. (3.10.2.11) where the y and z components of the flux are P and Q.

$$\int_\Omega \left(\frac{\partial P}{\partial y} + \frac{\partial Q}{\partial z}\right) dydz = \oint_\Gamma Pdy - Qdx \tag{3.10.2.11}$$

After transforming to the logical mesh coordinate system, the element average for ϕ is given by Eq. (3.10.2.12).

$$\begin{gathered} \phi_{(i+1/2,j+1/2)} = \frac{1}{V_{(i+1/2,j+1/2)}} \oint_\Gamma Pdj \\ \frac{\partial P}{\partial i} = \phi(i,j)\frac{\partial(y,z)}{\partial(i,j)} \end{gathered} \tag{3.10.2.12}$$

The function P is evaluated by integrating $\partial P/\partial i$ with j constant, requiring only a single pass through the old mesh. For axisymmetric problems, $\partial P/\partial i$ is set equal to

$2\pi\phi(i,j)\partial(y,z)/\partial(i,j)$. The integral appearing in Eq. (3.10.2.12) is evaluated by making a sweep along the i lines and a sweep along the j lines of the new mesh. Since each mesh line is a boundary for two rows of elements, the contribution from each line is added to one row and subtracted from the other, which greatly enhances the efficiency of the algorithm. This strategy also eliminates the need for the search strategy discussed in the section on interpolation, with a result that for logically regular meshes, this method is probably faster than interpolation.

An extension of the sweeping method to arbitrary meshes composed of triangular elements is used in CAVEAT-GT [195]. The key to the algorithm is a unique "Z" data structure that is associated with each edge, see Fig. 35. The body of the "Z" is designated the "direct" segment while the two arms are "indirect" segments. A path through the mesh is generated by using the downwind indirect segment as the index into the next "Z" structure. Each segment is covered twice, once as a direct segment, and once as an indirect. Starting at the boundaries, paths are generated through the mesh, with flags set after a segment is traversed as a direct and an indirect segment. When both flags have been set for all the boundary segments, all lines starting and ending on the boundaries have been traversed. For a logically regular mesh of quadrilaterals or hexagons, all lines would have been traversed at this point. A counter which is equal to the number of segments that have been traversed as either direct or indirect segments is checked. If it is equal to twice the number of segments, then all lines have been traversed for an arbitrary mesh, otherwise a starting segment is found in the interior of the mesh and segments are traversed until the loop closes. The counter is updated and checked. Interior loops are generated in this manner until the counter equals twice the number of segments.

When a piecewise linear approximation is used for ϕ, the flux F is generally discontinuous across the old element boundaries. The contributions to ϕ from the jumps in F across the old elements is accounted for by making an additional pass through the old mesh. Although this increases the computational cost, the linear approximation is worthwhile for problems involving steep gradients.

3.10.3 The PIC (Particle in Cell) method. Christensen [50] is currently using this method to conservatively remap problems in two and three dimensions. The old mesh is mapped sequentially element by element to the new mesh, where $\int \phi d\Omega$ is initially set to zero for each new element. For a particular old element, a linear, monotonic approximation of ϕ is calculated using the van Leer MUSCL algorithm. The element is subdivided by a logically regular mesh n subelements on a side, where n ranges between ten and twenty. The contribution of each subelement is $\phi(\xi)J(\xi)\triangle\xi_1\triangle\xi_2$, and it is attributed to the new element that contains the centroid of the subelement. After the old mesh contributions are distributed to the new mesh, the value of ϕ is given by Eq. (3.10.3.1), where δV_α is the volume of subelement α.

$$\phi = \frac{\phi_\alpha \delta V_\alpha}{V} \tag{3.10.3.1}$$

Note that the value used for the volume in the denominator is the volume of the new element, which will be slightly different than the sum of the subelement volumes. The new element volume is used to guarantee the conservation of ϕ between the old and new meshes. If the difference between the element volume and the sum of the subelement volumes is significantly different, the sum of the subelement volumes is used to avoid oscillations in the values of ϕ.

Some particle methods combine particle and continuum advection strategies [196]. Pure cells are advected with one of the conventional advection algorithms discussed in the previous sections, while the advection of the mixed cells is treated with the particle formulation. This approach is particularly attractive when new materials are created during the calculation, e.g., the formation of void material by a crack opening [197]. The Lagrangian nature of the particle transport overcomes the difficulties associated with transporting isolated material fragments. This is particularly important in problems involving localized material failure and impact debris.

3.10.4 Moving the mesh. During a rezone, the old mesh can be replaced by an entirely new mesh. A complete survey of mesh generation methods is beyond the scope of this paper, and the interested reader is referred to the review paper by Thompson [163], and [164]. Adaptive methods, where the mesh is changed locally based on some local error indicator, are being used extensively in gas dynamics codes, e.g., [124], [125], [130] and in solid mechanics, e.g., [121], but are currently not in production multimaterial hydrocodes. There are, however, some experimental hydrocodes, such as CAVEAT-GT [91], which can completely regenerate their mesh, but the user must specify through a weighting function how the mesh should be distributed.

Most ALE codes have adopted mesh relaxation procedures based on elliptic mesh generation methods. The coordinates of the nodes are calculated by minimizing a functional, I, which usually has elliptic Euler-Lagrange equations. Jacobi iteration is commonly used instead of a direct equation solver, resulting in a difference stencil that can be applied in a vectorized manner over the interior mesh. A mesh generator typically iterates hundreds of times to calculate the mesh, while ALE codes make only a few sweeps through the mesh each time step. The displacement increment calculated by the Jacobi iteration is limited by Courant restrictions associated with the advection algorithms.

One of the oldest relaxation stencils was developed by Winslow [109], [148]. He inverted Laplace's equation and derived a stencil so that the mesh forms lines of equal potential on a logically regular mesh. The mesh directions are ϕ and φ instead of ξ_i to avoid putting subscripts on the subscripts in the following equations. The inverted form of Laplace's equation is given by Eq. (3.10.4.1), with the coefficients given by Eq. (3.10.4.2). The differencing for quadrilateral elements is given by Eq. (3.10.4.3) and stencils for a wide variety of triangular mesh configurations have been derived [156].

$$\begin{aligned} \alpha y_{\phi\phi} - 2\beta y_{\phi\varphi} + \gamma y_{\varphi\varphi} = 0 \\ \alpha z_{\phi\phi} - 2\beta z_{\phi\varphi} + \gamma z_{\varphi} = 0 \end{aligned} \tag{3.10.4.1}$$

$$\begin{aligned} \alpha &= y_{\varphi}^2 + z_{\varphi}^2 \\ \beta &= y_{\phi} y_{\varphi} + z_{\phi} z_{\varphi} \\ \gamma &= y_{\phi}^2 + z_{\phi}^2 \end{aligned} \tag{3.10.4.2}$$

The difference stencils for z are derived by substituting z for y in Eq. (3.10.4.3).

$$\begin{aligned} y_{\phi} &= \tfrac{1}{2}(y_{(i,j+1)} - y_{(i,j-1)}) \\ y_{\varphi} &= \tfrac{1}{2}(y_{(i+1,j)} - y_{(i-1,j)}) \\ y_{\phi\phi} &= y_{(i,j+1)} - 2y_{(i,j)} + y_{(i,j-1)} \\ y_{\varphi\varphi} &= y_{(i+1,j)} - 2y_{(i,j)} + y_{(i-1,j)} \\ y_{\phi\varphi} &= \tfrac{1}{4}(y_{(i+1,j+1)} - y_{(i-1,j+1)} + y_{(i-1,j-1)} - y_{(i+1,j-1)}) \end{aligned} \tag{3.10.4.3}$$

After substituting Eq. (3.10.4.2) and Eq.(3.10.4.3) into Eq. (3.10.4.1), the coordinates of the center node are calculated.

$$
\begin{aligned}
y_{(i,j)} &= \frac{1}{2(\alpha+\gamma)}\Big(\alpha(y_{(i,j+1)}+y_{(i,j-1)})+\gamma(y_{(i+1,j)}+y_{(i-1,j)}) \\
&\qquad -\frac{\beta}{2}(y_{(i+1,j+1)}-y_{(i-1,j+1)}+y_{(i-1,j-1)}-y_{(i+1,j-1)})\Big) \\
z_{(i,j)} &= \frac{1}{2(\alpha+\gamma)}\Big(\alpha(z_{(i,j+1)}+z_{(i,j-1)})+\gamma(z_{(i+1,j)}+z_{(i-1,j)}) \\
&\qquad -\frac{\beta}{2}(z_{(i+1,j+1)}-z_{(i-1,j+1)}+z_{(i-1,j-1)}-z_{(i+1,j-1)})\Big)
\end{aligned}
\tag{3.10.4.4}
$$

The basic idea behind the Winslow algorithm has been extended in many ways, including by Winslow [148], Brackbill and Saltzman [144], and Giannakopoulos and Engel [139]. Winslow's variable diffusion method is used in CAVEAT [138], where $w(y,z)$ is a positive function.

$$
I=\int_\Omega \frac{1}{w}\left\{\frac{\partial\phi}{\partial x_i}\frac{\partial\phi}{\partial x_i}+\frac{\partial\varphi}{\partial x_i}\frac{\partial\varphi}{\partial x_i}\right\}d\Omega \tag{3.10.4.5}
$$

When w is constant, the original Winslow algorithm is recovered. The mesh is concentrated in areas where the logarithmic gradient, $\nabla w/w$, is largest. To concentrate the mesh in a region, a large value of w can be assigned in the neighborhood of the feature. A smoothing filter is applied to w to eliminate any high frequency components before it is used in Eq. (3.10.4.5).

In CAVEAT-GT [91], there are two rezoning algorithms, one being nearly Lagrangian, and the other, a complete rezone. The nearly Lagrangian rezone attempts to preserve the Lagrangian element volumes and a smooth mesh by filtering out the vorticity in the mesh motion. The functional is given by Eq. (3.10.4.6), where the increment in the mesh displacement and the volume are denoted as δx and δV respectively. A diagonally scaled conjugate gradient algorithm is used to solve the linearized equations generated from Eq. (3.10.4.6).

$$
I=\int_\Omega \left\{\left(\frac{\partial\delta x_i}{\partial x_i}\right)^2+2\delta x_j\frac{\partial\delta V/V}{\partial x_j}\right\}d\Omega \tag{3.10.4.6}
$$

In the complete rezone, the functional is a linear combination of two functionals. The first term wants to make the elements circular since it wants to make the ratio of the circumference C_i, to the area, A_i, of an element stationary. Triangular elements become equilateral, and quadrilaterals (which are not used in CAVEAT-GT) would become square. The second term is the sum of the squares of the lengths, D_i, of the element edges and its purpose is to smooth the distribution of the areas over the mesh. A relative weight, α, is used as a scaling coefficient.

$$I = \sum_{i=1}^{elements} \frac{C_i}{A_i} + \alpha \sum_{j=1}^{edges} D_j^2 \tag{3.10.4.8}$$

The boundary nodes are moved before the interior mesh is smoothed. Usually the nodes are equally spaced or their locations are calculated by projecting the locations of the first row of interior nodes onto the boundary. If a row of ghost elements surrounds the mesh, the interior mesh relaxation algorithm can be applied to the boundary nodes with the final positions of the nodes determined by a projection onto the boundary. Unless the nodes are moved tangentially to the boundary, volume is not conserved. For multimaterial ALE formulations, this does not present a problem, but simplified ALE formulations do not allow mixed elements. Instead of adjusting the values of the solution variables to enforce the conservation of mass, energy, etc, the loss of conservation, which is usually small, is tolerated in preference to introducing spurious oscillations in the solution variables.

3.11 MIXTURE THEORY

There are two essential issues associated with mixed elements:1) the force a mixture generates at a node during the Lagrangian step, and 2) the thermodynamic state within a mixed element. Since the forces are related to the state within the element, it is quite common to see both issues buried in the handling of the element thermodynamics within a calculation. The biggest obstacle to an accurate solution within a mixed element is the lack of information about the velocity distribution within the element. When the compressibility of the materials in a mixed element differ greatly, the standard assumption of a bilinear or linear velocity field over the element is inaccurate. Some assumptions about the behavior of the mixture must be introduced.

In hydrocodes, the subelement thermodynamics is treated with mixture theory. Each material has its own set of properties and history variables, and each material is assumed to occupy some fraction of the total volume of the element. Thermodynamic conditions are imposed to define a system of equations, which are generally nonlinear, with the volume fractions as the unknowns. There are three different sets of conditions that are frequently used [63]: 1) all the materials have the same pressure, but all the other variables may differ, 2) all the materials have the same pressure and temperature, and 3) the volumetric strain rate is equal for all the materials. These conditions may be imposed in either an exact or an approximate manner.

The first two sets of conditions assume pressure equilibrium. All researchers realize that this is a physically unrealistic assumption. A wave would cross the element many times during the length of time necessary for the materials to equilibrate. The Courant limit on the time step size, however, prevents a wave from crossing even the smallest element during a single time step. When material models incorporating tensile failure mechanisms are in an element that is partially a void, the failure may never occur because the void cannot support tension.

Thermal equilibrium is also a questionable assumption. It implies an infinite thermal conductivity within the element (thermal equilibrium applies only within an element; adjacent elements may have different temperatures). Gases that are in elements with solid materials may be artificially heated or cooled to an extreme degree depending on the situation.

The assumption of equal strain rates is clearly incorrect. In an element containing a mixture of steel and air, the air, being highly compressible, will absorb most of the volumetric strain. By assuming equal strain rates, however, the steel is forced to accept the same amount of strain as the air, and artificially high stresses are calculated in it. Despite the error in the assumption, McGlaun [63] and Christensen [51] report that solutions it gives are surprisingly good. This approach is also used in CAVEAT [138] and MESA [101].

Once the thermdynamics assumptions are established, their effect on the nodal forces can be determined. Based on virtual work, the relation between the mean stress, $\bar{\sigma}$, and mean pressure, $\bar{P}$, which are used to generate the nodal forces, and the individual stresses for each material is given by Eq. (3.11.1). The deviatoric stress and pressure in material k are σ'^k_{ij} and P^k respectively, and the fraction of the ij strain rate in material k is α^k_{ij}.

$$\begin{aligned}\bar{\sigma}'_{ij} &= \sum_k \sigma'^k_{ij}\alpha^k_{ij}V_k \quad \text{no sum on } ij \\ \bar{P}_{ij} &= \sum_k P^k\alpha^k_{jj}V_k \\ \alpha^k_{ij}V_k &= 1\end{aligned} \tag{3.11.1}$$

When the pressures are equal, then the mean pressure equals the individual material pressures. On the other hand, if α is 1.0 in all cases (equal strain rates), then the mean quantities are simply the volume-weighted average of the individual stresses. The consistency between the volume-weighted stress in the Lagrangian treatment and the equal strain rate thermodynamics established by Eq. (3.9.1) is probably the major reason that the equal strain rates assumption works as well as it does. An interesting application of mixture theory arises in elements that are designed for problems involving shearbanding. Belytshcko and his students use two materials elements with shear bands [197], [198], with one material unloading elastically and the other experiencing strain softening.

While the thermodynamics of the mixture theory is simple, the resulting equations are highly nonlinear and quite difficult to solve. The remainder of this section is devoted primarily to the numerical methods that different codes use to solve the mixture equations. When only two materials are present, the algorithms discussed below can be simplified by eliminating one of the volume fractions. See Anderson, O'Donoguhe, and Skerhut [153] for a very complete description of the simplifications and the application of mixture theory to composite materials.

3.11.1 The basic equations. The pressure equilibrium condition is expressed in either of two ways. The first equates the pressure in pairs of materials.

$$P_i(\rho_i, e_i) - P_{i+1}(\rho_{i+1}, e_{i+1}) = 0 \quad \text{for } i = 1, N-1 \tag{3.11.1.1}$$

The second expression equates the pressure in each material to the volume weighted pressure in the element.

$$\begin{aligned} &P_i(\rho_i, e_i) - P_T = 0 \\ &P_T = P_i V_i \end{aligned} \tag{3.11.1.2}$$

The pressure terms in Eq. (3.11.1.1) and Eq. (3.11.1.2) may include the shock viscosity for finite element and finite difference formulations, but not for Godunov formulations.

The pressure can be reduced to a function of only the volume fraction by expressing the density and the energy in terms of the volume fraction. The specific form for the energy function in Eq. (3.11.1.3) naturally depends on how the energy equation is being integrated. Some methods assume that the internal energy remains fixed in each material during the solution, while others account for the PdV work from the volume changes due to equilibrating the pressures.

$$\begin{aligned}\rho_i &= \frac{M_i}{V_i V} \\ e_i &= e(e_i, P_i^{n+1}, P_i^n, V_i^{n+1}, V_i^n, V^{n+1}, V^n, M_i)\end{aligned} \tag{3.11.1.3}$$

The temperature can be calculated either from the equation of state, or, as is usually the case, by dividing the internal energy by the heat capacity, C, (which is usually assumed to be constant).

$$\frac{e_i}{C_i} = \frac{e_{i+1}}{C_{i+1}} \tag{3.11.1.4}$$

Most solution methods iterate on the $N-1$ pressure equations defined by Eq. (3.11.1.1) or Eq. (3.11.1.2) and the constraint on the volume fractions, Eq. (3.11.1.5), with the N volume fractions as the unknowns. One reason that the equal strain rate assumption is often preferred to the pressure equilibrium assumption is the solution of the equilibrium equations may result in volume changes larger than the initial volume [92]. The constraint equation therefore also requires that all the volume fractions are positive.

$$\sum_i |V_i| = 1 \tag{3.11.1.5}$$

3.11.2 Solving the mixture equations with Newton-Raphson. The discussion in this section is restricted to the case where the materials are in pressure equilibrium, but they are not in thermal equilibrium. The Newton-Raphson method is one of the most efficient numerical methods available to solve nonlinear algebraic equations when the equations are well behaved and the initial guess for the solution is good.

When Eq. (3.111.1.) and Eq. (3.11.1.5) are used, the structure of the Jacobian matrix is simple enough that an efficient, special purpose equation solver can be written for it. For the first $N-1$ equations, there is a diagonal entry, $\partial P_i/\partial V_i$, and a superdiagonal entry, $\partial P_{i+1}/\partial V_{i+1}$, and the last row is all 1s. An alternative approach is used in CSQ [62]. Instead of Eq. (3.11.1.1), a set of functions is defined.

$$F_i = P_i - P_N \quad \text{for } i = 1, N-1 \tag{3.11.2.1}$$

For each Newton-Raphson iteration, the increments in the volume fractions are calculated according to Eq. (3.11.2.2).

$$\begin{aligned} \triangle V_N &= \frac{\sum_{i=1}^{N-1} F_i / \frac{\partial P_i}{\partial V_i}}{\frac{\partial P_N}{\partial V_N} \sum_{k=1}^{N} 1/\frac{\partial P_k}{\partial V_k}} \\ \triangle V_i &= \frac{\frac{\partial P_N}{\partial V_N} \triangle V_N - F_i}{\frac{\partial P_i}{\partial V_i}} \end{aligned} \tag{3.11.2.2}$$

The Newton-Raphson iteration requires good $\partial P/\partial V$ data. Unfortunately, the quality of the derivative data from tabular equations of state is often not very good [92], [166]. A better better estimate for the derivatives is often obtained by using ρc^2, which is a better behaved quantity, even for tabular equations of state.

The default element thermodynamics in CSQ assumes that the materials are at the same temperature. The CSQ solution strategy for thermal equilibrium, which will be discussed in a later section, consists of two nested iteration loops. In the inner loop, the equilibrium pressure calculation, described by Eq. (3.11.2.1) and Eq. (3.11.2.2), assumes a fixed temperature.

Christensen [50] uses only a single iteration of Newton-Raphson towards $P + Q$ equilibrium. The viscosity term is linear in the volumetric strain rate and it is only used in calculating the new volume fraction. This procedure relaxes the pressures towards equilibrium rather than enforcing equilibrium exactly. Once the new volume fractions are calculated, the pressure in each material is evaluated and the element pressure, P_T, is evaluated according to Eq. (3.11.1.2).

An alternative strategy [2], [154] is based on Eq. (3.11.1.2). A set of functions is again defined, but this time the range is from 1 to N.

$$F_i = P_i - P_T \tag{3.11.2.3}$$

Using the Newton-Raphson algorithm, the change in the volume fraction is given by Eq. (3.11.2.4).

$$\triangle V_i = \frac{P_T - P_i}{\frac{\partial P_i}{\partial V_i}} \tag{3.11.2.5}$$

The expression for P_T is derived by summing both sides of Eq. (3.11.2.5) and using the fact that the sum over i of $V_i + \triangle V_i$ is one.

$$P_T = \frac{1 + \sum_{i=1}^{N} (P_i / \frac{\partial P_i}{\partial V_i} - V_i)}{\sum_{k=1}^{N} 1/\frac{\partial P_k}{\partial V_k}} \tag{3.11.2.6}$$

During the solution of the equilibrium equations the change in the volume fractions must be limited so that the volume fraction of every material is bounded between zero and one. Furthermore, the magnitude of the change in the volume fractions must be limited since the pressure tends to infinity as the volume of a material approaches zero. If the changes in the volume fractions are not limited, then a material may become overly compressed during a single iteration and destabilize the Newton-Raphson iteration. When a phase transition occurs, $\partial P/\partial V$ is zero and the iteration schemes described above will not work. A backup scheme, which is described later, is used instead of the Newton-Raphson method.

3.11.3 Integrating the partial volumes with time. Drumheller [155] numerically integrates Eq. (3.11.3.1) to solve for the new values of the volume fractions. The value of α is chosen so that the solution of Eq. (3.11.3.1) approximates the solution of Eq. (3.11.1.2). Care must be used to avoid choosing a value of α that results in a numerically stiff set of differential equations.

$$\dot{V}_i = \alpha(P_T + Q_T - P_i - Q_i) \tag{3.11.3.1}$$

Drumheller uses an Adams method to solve this set of equations. The densities and their time derivatives are approximated by linearly interpolating the densities between t^n and t^{n+1}. Although Drumheller does not discuss it in his paper, it appears that densities at t^{n+1} are calculated initially by assuming that the volume fractions remain constant over the time step.

3.11.4 Iteration for pressure and thermal equilibrium. Thermal equilibrium imposes a constraint on the energy distribution, Eq. (3.11.1.4). Following the practice in most hydrocodes, the temperature, T, is related to the internal energy by e_i/C_i, where C_i, the specific heat, is assumed to be constant. In addition, the sum of the energy increments for each material must equal the total work of the element performed over the time step. The increment in the specific internal energy is $\triangle E_i/M_i$.

$$\sum_{i=1}^{N} \triangle E_i = \triangle E = \int_\Omega \int_{t^n}^{t^{n+1}} \sigma_{k\ell}\dot{\epsilon}_{k\ell} dt d\Omega \tag{3.11.4.1}$$

The stress in Eq. (3.11.4.1) includes the pressure, the shock viscosity, and the deviatoric stress. For a Godunov method, $\triangle E$ is completely determined by boundary tractions and velocities of the element, and it is also known when Eq. (2.7.3.2) is used to update the energy since the element stress and the strain rate, centered at $t^{n+1/2}$, are known. In these two cases the energy distribution in the element is independent of the volume fractions and it can be solved for directly. The pressure equilibrium is then determined holding the specific energies constant. Following a procedure similar to the pressure equilibrium iteration starting at Eq. (3.11.2.3), the energy distribution can be calculated explicitly.

$$\begin{aligned}
&\frac{\triangle e_i}{C_i} - \frac{\triangle e_N}{C_N} = \frac{e_N}{C_N} - \frac{e_i}{C_i} = F_i \\
&\sum_{i=1}^{N-1} M_i \triangle e_i + M_N \triangle e_N = \triangle E \\
&\triangle e_i = C_i(F_i + \triangle e_N / C_N) \\
&\triangle e_N = \frac{\triangle E - \sum_{i=1}^{N-1} M_i C_i F_i}{M_N + \sum_{i=1}^{N-1} M_i C_i / C_N}
\end{aligned} \tag{3.11.4.2}$$

When the energy equation is integrated using the strategy in HEMP, see Eq. (2.7.2.1), then the element energy equation increment is unknown. For a Grüneisen equation of state, the energy increment can be expressed as a function of the volume, see Eq. (2.7.2.4). By linearizing and substituting Eq. (2.7.2.1) into Eq. (3.11.4.2), the total derivative for the pressure can be calculated for the Newton-Raphson iteration.

$$\frac{\mathrm{d}P}{\mathrm{d}V_i} = \frac{\partial P}{\partial \rho}\frac{\partial \rho}{\partial V_i} + \frac{\partial P}{\partial e}\frac{\partial e}{\partial V_i} \tag{3.11.4.3}$$

CSQ [62] uses two nested iterations to calculate the equilibrium temperature and pressure. The inner loop equilibrates the pressure for a given constant temperature, while the outer loop iterates on the temperature.

$$M_i[e_i(T^{n+1}) - e_i(T^n)] - \int_\Omega \int_{t^n}^{t^{n+1}} \sigma_{k\ell}(T)\dot{\epsilon}_{k\ell} dt \; d\Omega = 0 \tag{3.11.4.4}$$

The derivatives of the energy and pressure with respect to the temperature at a constant density must be evaluated to solve Eq. (3.11.4.4) by the Newton-Raphson method. Using the chain rule, the derivatives are evaluated provided that $\partial \rho_i / \partial T|_\rho$ is known. CSQ [62] evaluates these derivatives by solving a set of linear equations.

$$\begin{aligned}
\frac{\partial P_i}{\partial T}\bigg|_{\rho_i} + \frac{\partial P_i}{\partial \rho_i}\bigg|_T \frac{\partial \rho_i}{\partial T}\bigg|_\rho &= \frac{\partial P_N}{\partial T}\bigg|_{\rho_N} + \frac{\partial P_N}{\partial \rho_N}\bigg|_T \frac{\partial \rho_N}{\partial T}\bigg|_\rho \\
\frac{M_i}{\rho_i^2}\frac{\partial \rho_i}{\partial T}\bigg|_\rho &= 0
\end{aligned} \tag{3.11.4.5}$$

The second equation is derived by differentiating the conservation of volume, written as Eq. (3.11.4.6), with respect to temperature.

$$\sum_{i=1}^{N} \frac{M_i}{\rho_i} = V \tag{3.11.4.6}$$

Assuming that $\partial P_i/\partial \rho_i$ is positive for all the materials (i.e., no phase changes at the current pressure), the solution for Eq. (3.11.4.5) can be obtained directly.

$$\begin{aligned} \left.\frac{\partial \rho_i}{\partial T}\right|_{\rho} &= \frac{\displaystyle\sum_{k=1}^{N} \left[\left.\frac{\partial P_k}{\partial T}\right|_{\rho_k} - \left.\frac{\partial P_i}{\partial T}\right|_{\rho_i} \right] \frac{\gamma_k}{\beta_k}}{\displaystyle\beta_i \sum_{\ell=1}^{N} \frac{\gamma_\ell}{\beta_\ell}} \\ \beta_i &= \left.\frac{\partial P_i}{\partial \rho_i}\right|_{T} \\ \gamma_i &= \frac{M_i}{\rho_i^2} \end{aligned} \tag{3.11.4.7}$$

When a phase change occurs, at least one of the β_i are zero. The solution for all i such that β_i is positive is given by Eq. (3.11.4.8).

$$\left.\frac{\partial \rho_i}{\partial T}\right|_{\rho} = \frac{1}{\beta_i} \left[\left.\frac{\partial P}{\partial T}\right|_{\rho} - \left.\frac{\partial P_i}{\partial T}\right|_{\rho_i} \right] \tag{3.11.4.8}$$

The derivatives for the materials that have β_j equal to zero are not uniquely defined if there are more than one, and they are all set to the same value.

$$\begin{aligned} \left.\frac{\partial \rho_j}{\partial T}\right|_{\rho} &= -\frac{\displaystyle\sum_{i \in S_1} \gamma_i \left.\frac{\partial \rho_i}{\partial T}\right|_{\rho}}{\displaystyle\sum_{k \in S_2} \gamma_k} \quad \forall\, j \in S_2 \\ S_1 &= \{i | \beta_i \neq 0\} \\ S_2 &= \{i | \beta_i = 0\} \end{aligned} \tag{3.11.4.9}$$

If an element contains a void, the equilibrium pressure is zero and the volume of each material for a zero pressure can be calculated independently. A check must be performed to ensure that the sum of the volumes does not exceed the element volume. If the sum is too large, then the void is eliminated by the expansion of the materials. Some equations of state incorporate a pressure cutoff, P_{min}, to restrict the amount of tension the material can withstand. If the pressure exceeds the pressure cutoff, then the material has failed and a void is allowed to form. The pressure is usually allowed to decay towards zero over several time steps in order to avoid instabilities in the solution. The step nonlinearities associated with pressure cutoffs can prevent the Newton-Raphson method from converging to a solution.

3.11.5 The backup search. A failsafe method is usually incorporated in Eulerian codes for the cases when the Newton-Raphson method cannot find the equilibrium pressure. A binary search strategy is used in CSQ [62] and PISCES [2], while CTH [113] uses a combination of the secant method and the binary search. The PISCES binary search, which incorporates ideas from monotonicity analysis, appears to be a very efficient algorithm.

A binary search is useful only for solving single equations when the solution is known to lie within a specified interval. The problem is to determine a set of volume fractions such that each material is at the equilibrium pressure, P_T, subject to the constraint that the volume fractions sum to one.

$$\begin{aligned} P_i(V_i) &= P_T \\ \sum_{i=1}^{N} V_i &= 1 \end{aligned} \tag{3.11.5.1}$$

If the pressure falls below the cutoff pressure, then there is a void, the volume fraction of the void, V_0, is given by Eq. (3.11.5.2), and the pressure is zero.

$$V_0 = 1 - \sum_{i=1}^{N} V_i \tag{3.11.5.2}$$

The problem is reduced to a single equation by considering the test function F, the error in the volume fraction constraint (or void volume) as a function of pressure.

$$F = 1 - \sum_{i=1}^{N} V_i \tag{3.11.5.3}$$

The zero of Eq. (3.11.5.3) is the equilibrium pressure if there is no void, otherwise the value of F is the void volume, see Fig. 30. Once the equilibrium pressure is established, the volume fraction for each material is calculated by solving the first equation of Eq. (3.11.5.1) for each material independently.

The bounding interval for the pressures is defined by a lower pressure, P_L, and an upper pressure, P_U. The bounding interval is reduced every iteration and convergence in the pressure is defined when the interval is smaller than some specified value.

$$P_U - P_L \leq \max(\triangle P, \epsilon_2 |P_L|) \tag{3.11.5.4}$$

An alternative convergence criterion, based on the volume fractions, is necessary to guarantee convergence.

$$|F(P_T)| \leq \epsilon_1 \tag{3.11.5.5}$$

In the CSQ implementation, the volume fraction for a specified pressure is calculated for each material using Newton-Raphson. Hancock invokes a monotonicity assumption, Eq. (3.11.5.6), in PISCES to reduce the number of equation of state evaluations to one per binary search iteration, an enormous increase in efficiency.

$$V_i(P^+) \leq V_i(P^-) \quad \text{if} \quad P^+ \geq P^- \tag{3.11.5.6}$$

An arbitrary set of states can be used to bound F by invoking Eq. (3.11.5.6). For example, by letting P^- equal P_i, and P^+ equal the maximum P_i over all i, and summing the volume fractions gives a bound for the maximum P_i.

$$\begin{aligned} \sum_i V_i(P_i^{max}) &\leq \sum_i V_i(P_i) \\ F(P_i^{max}) = 1 - \sum_i V_i(P_i^{max}) &\geq 1 - \sum_i V_i(P_i) \end{aligned} \tag{3.11.5.7}$$

In a similar manner, a bound can be found for the minimum pressure.

$$F(P_i^{min}) = 1 - \sum_i V_i(P_i^{min}) \leq 1 - \sum_i V_i(P_i) \tag{3.11.5.8}$$

An ordinary binary search would pick a value of P_T, evaluate F, and then determine whether the pressure corresponds to a new upper or lower bound. Hancock keeps a history of the equation state evaluations for each material, ordered by pressure. A description of how the history is generated is deferred. For a specified P_T, a utility routine gathers a set of states with pressures below P_T and another set with pressures above P_T, see Fig. 31. If F evaluated with the lower pressures is positive, then the maximum of the lower pressures defines a new P^U by Eq. (3.11.5.7). Similarly, if F evaluated with the higher pressures is negative, then the minimum of the higher pressures defines a new lower bound, P^L. Note that the inequalities produce a sharper bound than the ordinary binary search, reducing the number of iterations required for a specified level of accuracy.

A single equation of state evaluation is made if the bracketing states do not result in a new upper or lower bound. The material with the greatest difference in its bracketing volume fractions is evaluated with a volume fraction midway between the bracketing values. Depending on whether the pressure is larger or smaller than the test pressure, the new equation of state evaluation is a new upper or lower bracketing state. The upper and lower bounds on the void fraction function F are again evaluated and checked. Once new upper and lower bounds are established, a new test pressure is calculated. When the upper and lower pressure bounds are widely separated, a geometric average is used, otherwise the new test pressure is the arithmetic average.

The backup search is invoked when the Newton-Raphson iteration appears to be diverging. During each Newton-Raphson iteration, new volume fractions are calculated corresponding to a value of P_T. A set of bracketing values of V_i for each material is found in the equation of state history tables based on the current value of P_T. If the new V_i falls outside the bracketed range for any material, then the binary search is used until a solution is found. There is no mechanism in PISCES for switching back to the Newton-Raphson method from the binary search.

To reduce the cost of evaluating the equations of state, a record of the results of each equation of state evaluation is kept in a doubly linked list that is sorted by pressure. When bracketing states for a particular pressure are requested, the table is searched. If a lower bound cannot be found, then the volume fraction is set to one and the pressure is set to the negative of some large number. Conversely, if an upper bound cannot be found, then the volume fraction is set to zero, and the pressure is set to a large number. The linked list is also used to generate interpolating functions for the equations of state since the search is performed over a single parameter. After each equation of state evaluation, the new state is stored unless the following four criteria are met:

1. Neither of the bracketing states corresponds to one of the artificial bounding states discussed above.
2. The volume fraction lies in the middle sixty percent of the interval. This criterion is included because the interpolation error is largest at the center of the interval.

3. The interpolation function, P_I, closely approximates the equation of state.

$$|P_I - P| \leq \max(\triangle P, \epsilon_2 |P|) \tag{3.11.5.9}$$

4. The bracketing states are not too far apart. This prevents a false indication of convergence.

An interval is considered to have "converged" when the criteria are satisfied and all future equation of state evaluations lying in the interval are replaced by a cubic spline (or a linear fit if the spline is not monotonic).

In CTH [113], the void function is solved using secant iteration, Eq. (3.11.5.10), with every third or fourth secant iteration followed by a single binary search. Note that each evaluation of $F(P)$ involves iteratively solving N nonlinear problems for the volume fractions.

$$P_T = P_L - \frac{P_U - P_L}{F(P_U) - F(P_L)} F(P_L) \tag{3.11.5.10}$$

The binary search improves the convergence rate for some problems where one of the initial bounds remains fixed during the iterations.

The relative efficiency of the binary search and the secant method are difficult to assess. Each iteration of the binary search reduces the error by a factor of two in theory, but the monotonicity assumptions invoked by Hancock give him sharper bounds. In addition, the bounding strategy does not require the accurate evaluation of F, only its bound, so that Hancock can reduce the number of equation of state evaluations by using the bracketing states and interpolation. The near quadratic convergence rate of the secant method is attractive, but each iteration requires the solution of N nonlinear subproblems.

3.11.6 Partitioning the deviatoric strain rate and the bulk viscosity. The deviatoric strain rate in each material is assumed to be the average strain rate of the element. This assumption permits a solid material to retain its strength when it is in a mixed element containing a gas or liquid. The deviatoric stress for each material is computed independently and a volume weighted mean stress is calculated for the element.

$$\sigma_{ij} = \sigma_{ij}^k V_k \tag{3.11.6.1}$$

The shock viscosity is usually constant throughout the element, and it is computed based on the element average for the density and sound speed. The increment in work for each material from its deviatoric stress and bulk viscosity is calculated and treated as a source term during the pressure equilibrium calculations. Anderson, O'Donoghue, and Skerhut [153] follow a slightly different path for their two material mixture theory. They assume that the strain rate in each material is the same as the element average strain rate, but the densities are different. Their final element viscosity has the same form as the volume weighted mean stress in Eq. (3.11.6.1).

3.11.7 Modifications for Godunov methods. The internal energy in a Godunov method is calculated as the difference between the total energy of the zone and its kinetic energy. The total energy is updated by a boundary integral involving the contact pressures and velocities. Unlike finite element and finite difference codes, which solve the pressure and energy equations simultaneously, the internal energy of the zone in a Godunov code is calculated independently of the zone pressure. The energy distribution of the zone after the pressure equilibrium must satisfy the conservation of energy equation, where E^{n+1} is the total internal energy of the zone.

$$M_i e_i^{n+1} = E^{n+1} \tag{3.11.7.1}$$

The energy E^{n+1} includes contributions associated with shock heating while the algorithms outlined in the previous sections for calculating the equilibrium pressure include shock heating only through the shock viscosity. Since Godunov methods do not include a shock viscosity, Eq. (3.11.7.1) is generally not satisfied. The difference, $\triangle E$, which is associated with the shock heating, must be distributed to the different materials.

$$\triangle E = E^{n+1} - M_i e_i^{n+1} \tag{3.11.7.2}$$

PISCES distributes the difference based on the relative compressibilities of the materials. The energy increment $\triangle e_i$ is the energy added to material i during the pressure equilibration for Eq. (3.11.7.2).

$$\begin{aligned} e_i^{n+1} &= e_i^n + \triangle e_i + \omega_i \frac{\triangle E}{M} \\ \omega_i &= \frac{M_i/\rho_i a_i^2}{\sum_{k=1}^N M_k/\rho_k a_k^2} \end{aligned} \tag{3.11.7.3}$$

Using the new values for the internal energies given by Eq. (3.11.7.3), a new equilibrium pressure is calculated using the strategy discussed in the previous section.

CAVEAT distributes the internal energy increment according to the mass fraction of each material [92]. Although this is a very simple algorithm, it works well and it is very robust.

ACKNOWLEDGMENTS

I have talked to many people while I was writing this paper, and I thank them for all their help. My special thanks to Steve Hancock, Hans Mair of NAVSWC, and Norm Johnson of Los Alamos National Laboratory for diligently proofreading my original draft. John Dukowicz, Frank Addessio, Bryan Kashiwa, and Ed Cort, all from Los Alamos National Laboratory, also provided valuable advice and comments. Randy Christensen, Bob Tipton, Richard Sharpe and John Trangenstein at Lawrence Livermore National Laboratory patiently answered my questions about their work. Despite their best efforts and mine, I am sure that no paper this long can be perfect. Since I typed every last word and equation, the responsibility for the errors and omissions are mine alone.

REFERENCES

1. Malvern, L. E., Introduction to the Mechanics of a Continuous Medium, Prentice-Hall, Englewood Cliffs, N. J., (1969).

2. Hancock, S., “ PISCES 2DELK Theoretical Manual”, Physics International, (1985).

3. Addessio, F. L., Carroll, D. E., Dukowicz, J. K., Harlow, F. H., Johnson, J. N., Kaskiwa, B. A., Maltrud, M. E., and Ruppel, H. M., “CAVEAT: A Computer Code for Fluid Dynamics Problems with Large Distortion and Internal Slip,” Los Alamos National Laboratory UC-32, (1988).

4. Noh, W. F., “Numerical Methods in Hydrodynamic Calculations,” Lawrence Livermore National Laboratory, UCRL-52112, (1976).

5. Courant, R., and Friedrichs, K. O., *Supersonic Flow and Shock Waves*, Springer-Verlag, New York, (1976).

6. Maenchen, G., Sack, S., “The TENSOR Code,” *Methods in Computational Physics*, Vol. 3, Academic Press, New York, (1964).

7. Wilkins, M., “Calculation of Elastic - Plastic Flow,” *Methods in Computational Physics, Vol. 3, Fundamental Methods in Hydrodynamics*, Academic Press, pp. 211-263, (1964).

8. Krieg, R. D., and Key, S. W., “Implementation of a Time Dependent Plasticity Theory into Structural Programs,” *Constitutive Equations in Viscoplasticity: Computational and Engineering Aspects*, ASME Vol. 20, pp. 125-137, (1976).

9. Noh, W. F., “CEL: A Time - Dependent, Two - Space - Dimensional, Coupled Eulerian - Lagrange Code,” *Methods in Computational Physics, Volume 3, Fundamental Methods in Hydrodynamics*, Academic Press, (1964), pp. 117-179.

10. Fritts, M. J., Crowley, W. P., and Trease, H., “The Free-Lagrange Method” Lecture Notes in Physics 238, Springer-Verlag, Berlin, (1983).

11. Godunov, S. K., “Finite Difference Method for Numerical Computation of Discontinuous Solutions of the Equations of Fluid Dynamics,” *Matematicheskii Sbornik*, Vol. 47, No. 3, pp. 271-306, (1959).

12. Hughes, T. J. R., The Finite Element Method, Linear, Static and Dynamic Finite Element Analysis, Prentice - Hall, (1987).

13. Zienkiewicz, O. C., The Finite Element Method, Third Edition, McGraw Hill, (1987).

14. Bathe, K. J., Finite Element Procedures in Engineering Analysis, Prentice-Hall, Englewood Cliffs, N. J., (1982).

15. Cook, R. D., Malkus, D. S., and Plesha, M. E., Concepts and Applications of Finite Element Analysis, Third Edition, John Wiley and Sons, New York, N. Y., (1989).

16. Johnson, C., Numerical Solutions of Partial Differential Equations by the Finite Element Method, Cambridge University Press, Cambridge, N. Y., (1987).

17. Rank, E., Katz, C., and Werner, H., "On the Importance of the Discrete Maximum Principle in Transient Analysis Using Finite Element Methods," *International Journal for Numerical Methods in Engineering*," Vol. 19, pp. 1771-1782, (1983).

18. Hallquist, J. O., DYNA3D Course Notes, Lawrence Livermore National Laboratory, UCID-19899, Rev. 2, (1987).

19. Benson, D. J., "Vectorizing the Right Hand Side Assembly in an Explicit Finite Element Program," *Computer Methods in Applied Mechanics and Engineering*, Vol. 73, pp. 147-152, (1989).

20. Johnson, G. R., and Stryk, R. A., User Instructions for the EPIC-2 Code, Air Force Armament Laboratory, AFATL-TR-86-51, (1986).

21. Nagtegaal, J. C., Parks, D. M., and Rice, J. R., "On Numerically Accurate Finite Element Solutions in the Fully Plastic Range," *Computer Methods in Applied Mechanics and Engineering*, Vol. 4, pp. 153-177, (1974).

22. Johnson, G. R., and Stryk, R. A., "Eroding Interface and Improved Tetrahedral Element Algorithms for High-Velocity Impact Computations in Three Dimensions," *Int. J. Impact. Engng.* Vol. 5, pp. 411-421, (1987).

23. Demuth, R. B., Margolin, L. G., Nichols, B. D., Adams, T. F., and Smith, B. W., "SHALE: A Computer Program for Solid Dynamics," Los Alamos National Laboratory, LA-10236, (1985).

24. 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).

25. Amsden, A. A., and Ruppel, H. M. "SALE-3D: A Simplified ALE Computer Program for Calculating Three-Dimensional Fluid Flow," Los Alamos Scientific Laboratory, (Nov. 1981).

26. Amsden, A. A., and Hirt, C. W., “YAQUI: An Arbitrary Largrangian-Eulerian Computer Program for Fluid Flow at All Speeds,” Los Alamos Scientific Laboratory, LA-5100, (1973).

27. Hallquist, J. O., “NIKE2D – A Vectorized Implicit, Finite Deformation Finite Element Code for Analyzing the Static and Dynamic Response of 2-D Solids with Interactive Rezoning and Graphics,” Lawrence Livermore National Laboratory, UCID-19677, Rev. 1, (1986).

28. Truesdell, C., and Noll, W., The Non-Linear Field Theories of Mechanics, Handbuch Der Physik, Volume III/3, Springer-Verlag, Berlin, (1965).

29. Marsden, J. E., and Hughes, T. J. R., Mathematical Foundations of Elasticity, Prentice-Hall, Englewood Cliffs, N. J., (1983).

30. Billington, E. W., and Tate, A., The Physics of Deformation and Flow, McGraw-Hill, New York, N. Y., (1981).

31. Gurtin, M. E., An Introduction to Continuum Mechanics, Mathematics in Science and Engineering, Vol. 158, Academic Press, New York, (1981).

32. Johnson, G. C., Bammann, D. J., 1984, “A Discussion of Stress Rates in Finite Deformation Problems,” *Int. J. Solids Struct.*, Vol. 20, pp. 725-737.

33. Polukhin, P., Gorelik, S., and Vorontsov, V., Physical principles of Plastic Deformation, Mir Publishers, Moscow, U.S.S.R., (1983).

34. Taylor, L. M., and Flanagan, D. P., “PRONTO 2D A Two-Dimensional Transient Solid Dynamics Program”, Sandia National Laboratories, SAND86-0594, (1987).

35. Hughes, T. J. R., Liu, W. K., and Zimmerman, T. K., “Lagrangian-Eulerian Finite Element Formulation for Incompressible Viscous Flows,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 29, pp. 329-349, (1981).

36. Donea, J., "Arbitrary Lagrangian-Eulerian Finite Element Methods," Computational Methods for Transient Analysis, T. Belytschko and T. J. R. Hughes, ed., Elsevier Science Publishers, B. V., pp 473-516, (1983).

37. Bammann, D., personal communication, Sandia National Laboratories, (1983).

38. Hoger, A., and Carlson, D. E., "Determination of the Stretch and Rotation in the Polar Decomposition of the Deformation Gradient," *Quart. Appl. Math.*, Vol. 42, pp. 113-117, (1984).

39. Dienes, J. K., "On the Analysis of Rotation and Stress Rate in Deforming Bodies," *Acta Mechanica*, Vol. 32, pp. 217-232, (1979).

40. Hughes, T. J. R., and Winget, J., "Finite Rotation Effects in Numerical Integration of Rate Constitutive Equations Arising in Large-Deformation Analysis," International Journal for Numerical Methods in Engineering, Vol. 15, No. 12, pp. 1862-1867, (1980).

41. Margolin, L. G., "A Centered Artificial Viscosity for Cells with Large Aspect Ratio," Lawrence Livermore National Laboratory, UCRL-53882, (1988).

42. Benson, D. J., "A New Two-Dimensional Flux-Limited Shock Viscosity," *Computer Methods in Applied Mechanics and Engineering*, (in press).

43. Von Neumann, J., and Richtmyer, R. D., "A Method for the Numerical Calculation of Hydrodynamic Shocks," *Journal of Applied Physics*, Vol. 21, (1950).

44. Crowley, W. P., "Numerical Methods in Fluid Dynamics," Lawrence Livermore National Laboratory UCRL-51824, (1975).

45. Noh, W. F., "Errors for Calculations of Strong Shocks Using an Artificial Viscosity and an Artificial Heat Flux," Lawrence Livermore National Laboratory, UCRL-53669, (1985).

46. Wilkins, M. L., "Use of Artificial Viscosity in Multidimensional Fluid Dynamic Calculations," *Journal of Computational Physics*, Vol. 36, pp. 381-303, (1980).

47. Landshoff, R., “A Numerical Method for Treating Fluid Flow in the Presence of Shocks,” Los Alamos Scientific Laboratory LA-1930, (1955).

48. Richards, G. T., “Derivation of a Generalized Von Neumann Pseudo-Viscosity with Directional Properties,” Lawrence Radiation Laboratory UCRL-14244, (1965).

49. Kuropatenko, V. F., in *Difference Methods for Solutions of Problems of Mathematical Physics, I*, ed., Janenko, N. N., American Mathematical Society, Providence, RI, (1967).

50. Christensen, R., personal communication, Lawrence Livermore National Laboratory, (1989).

51. Christensen, R., personal communication, Lawrence Livermore National Laboratory, (1990).

52. 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).

53. Johnson, W. E., Anderson, C. E., “History and Application of Hydrocodes in Hypervelocity Impact,” *Int. J. Impact Engng.*, Vol. 5, pp. 423-439, (1987).

54. Cherry, J. T., Sack, S., Maenchen, G., and Kransky, V., “Two-Dimensional Stress-Induced Adiabatic Flow,” Lawrence Livermore National Laboratory, UCRL-50987, (1970).

55. Burton, D. E., Lettis, L. A., Bryan, J. B., and Frary, N. R., “Physics and Numerics of the TENSOR Code [Incomplete Preliminary Documentation],” Lawrence Livermore National Laboratory, UCID-19428, (1982).

56. Schulz, W. D., “Tensor Artificial Viscosity for Numerical Hydrodynamics,” *Journal of Mathematical Physics*, Vol. 5, No. 1, (1964).

57. Washizu, K., Variational Methods in Elasticity and Plasticity, Third Edition, Pergamon Press, Oxford, England, (1982).

58. Flanagan, D. P., and Belytschko, T., "A Uniform Strain Hexahedron and Quadrilateral and Orthogonal Hourglass Control," *Int. J. Numer. Meth. Eng.*, Vol. 17, pp. 679-706, (1981).

61. Belytschko, T., and Bachrach, W., "Efficient Implementation of Quadrilaterals with High Coarse-mesh Accuracy," *Computer Methods in Applied Mechanics and Engineering*, Vol. 54, pp. 279-301, (1986).

60. Margolin, L. G., and Pyun J. J., "A Method for Treating Hourglass Patterns," Proc. of the Fourth International Conference on Numerical Methods in Laminar and Turbulent Flow, Montreal, Quebec, Canada, July 6-10, 1987.

61. Kosloff, D., and Frazier, G., "Treatment of Hourglass Patterns in Low Order Finite Element Codes," *Numerical and Analytical Methods in Geomechanics*, Vol. 2, pp. 57-72, (1978).

62. Thompson, S. L., "CSQ – A Two Dimensional Hydrodynamic Program with Energy Flow and Material Strength," Sandia Laboratories, SAND74-0122, (1975).

63. 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).

64. Benson, D. J., and Hallquist, J. O., "A Single Surface Contact Algorithm for the Postbuckling Analysis of Shell Structures," *Computer Methods in Applied Mechanics and Engineering*, Vol. 78, pp. 141-163, (1990).

65. Chaudhary, A. B., and Bathe, K. J., "A Lagrange Multiplier Segment Procedure for Solution of Three Dimensional Contact Problems," U.S. Army Ballistic Research Laboratory, BRL-CR-544, (1985).

66. Kikuchi, N. and Oden, J. T., Contact Problems in Elasticity: A Study of Variational Inequalities and Finite Element Methods for a Class of Contact Problems in Elasticity, SIAM Studies Vol. 8, (1986).

67. Zhong, Z. H., On Contact-Impact Problems, Ph.D. thesis, Division of Solid Mechanics and Strength of Materials, Department of Mechanical Engineering, Linkoping University, Linkoping, Sweden, (1988).

68. Anderson, C. E., "An Overview of the Theory of Hydrocodes," *Int. J. Impact Engng.*, Vol. 5, pp. 33-59, (1987).

69. Belytschko, T., and Neal, M. O., "Contact-Impact by the Pinball Algorithm with Penalty, Projection, and Lagrangian Methods," Proc. Symp. on Computational Techniques for Impact, Penetration, and Performation of Solids, ed. L. E. Schwer, N. J. Salamon, and W. K. Liu, ASME Winter Annual Meeting, San Fransisco, CA, Dec. 1989, AMD-Vol. 103, (1989).

70. Simo, J. C., Wriggers, P., and Taylor, R. L., "A Perturbed Lagrangian Formulation for the Finite Element Solution of Contact Problems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 50, pp. 163-180, (1985).

71. Christensen, R., personal communication, Lawrence Livermore National Laboratory, (1990).

72. Belytschko, T., and Lin, J. I., "A New Interaction Algorithm with Erosion for EPIC-3," U.S. Army Ballistic Research Laboratory, BRL-CR-540, (1985).

73. Belytschko, T., personal communication, Northwestern University, (1990).

74. Nilsson, L., Zhong, Z. H., and Oldenburg, M., "Analysis of Shell Structures Subjected to Contacts-Impacts," Proc. Symp. on Analytical and Computational Models for Shells, ASME Winter Annual Meeting, San Fransisco, CA, (1989).

75. Hallquist, J. O., Goudreau, G. L., and Benson, D. J., "Sliding Interfaces with Contact-Impact in Large-Scale Lagrangian Computations," Computer Methods in Applied Mechanics and Engineering, Vol. 51, No. 1/3, (1985).

76. Bertholf, L. D., and Benzley, S. E., "TOODY II, A Computer Program for Two-Dimensional Wave Propagation," Sandia National Laboratories, SC-RR-68-41, Sandia National Laboratories, (1968).

77. Hughes, T. J. R., Taylor, R. L., Sackman, J. L., Curnier, A., and Kanoknukulchai, W., “A Finite Element Method for a Class of Contact-Impact Problems,” Computer Methods in Applied Mechanics and Engineering, Vol. 8, pp. 249-276, (1976).

78. Key, S. W., *HONDO - A Finite Element Computer Program for the Large Deformation Response of Axisymmetric Solids*, Sandia National Laboratory, Rept. 74-0039, (1974).

79. Johnson, G. R., Stryk, R. A., and Dodd, J. G., “Dynamic Lagrangian Computations for Solids, with Variable Nodal Connectivity for Severe Distortions,” *International Journal for Numerical Methods in Engineering*, Vol. 23, pp. 509-522, (1986).

80. Stecher, F. P., and Johnson, G. C., “Lagrangian Computations for Projectile Penetration into Thick Plates,” Computers in Engineering, Vol.2, ed. W. A. Gruver, ASME G00240, (1984).

81. Hicks, D. L., “Stability Analysis of WONDY for a Special Case of Maxwell’s Law,” *Mech. Comp.*, Vol. 32, p. 1123, (1978).

82. Hallquist, J. O., “User’s Manual for DYNA2D – An Explicit Two-Dimensional Hydrodynamic Finite Element Code with Interactive Rezoning”, Lawrence Livermore National Laboratory, UCID-18756 Rev. 2, (1984).

83. Hallquist, J. O., and Benson, D. J., “DYNA3D User’s Manual (Nonlinear Dynamic Analysis of Structures in Three Dimensions)” Lawrence Livermore National Laboratory, UCID-19592, Rev. 3, (1987).

84. Trangenstein, J. A., and Colella, P., “A Higher-Order Godunov Method for Modeling Finite Deformation in Elastic-Plastic Solids,” submitted to *Communications in Pure and Applied Mathematics*, (1989).

85. Zukas, J. A., Nicholas, T. Swift, H. F., Greszczuk, L. B., and Curran, D. R., Impact Dynamics, John Wiley and Sons, New York, (1982).

86. Colella, P., and Woodward, P., "The Piecewise Parabolic Method (PPM) for Gas Dynamical Simulations," *Journal of Computational Physics*, Vol. 54, pp. 174-201, (1984).

87. Dukowicz, J. K., "A General, Non-Iterative Riemann Solver for Godunov's Method," *Journal of Computational Physics*, Vol. 61, pp. 119-137, (1985).

88. Harten, A., "ENO Schemes with Subcell Resolution," *J. of Computational Physics*, Vol. 83, pp. 148-184, (1989).

89. Trangenstein, J. A., "A Second-Order Algorithm for the Dynamic Response of Soils," Lawrence Livermore National Laboratory preprint, (1990).

90. Colella, P., "Multidimensional Upwind Methods for Hyperbolic Conservation Laws," Lawrence Berkeley Laboratory, LBL-17023, May, 1984, published in *Journal of Computational Physics*, Vol. 87, pp. 171-209, (1990).

91. Dukowicz, J. K., Cline, M. C., and Addessio, F. L., "A General Topology Godunov Method," *J. of Computational Physics*, Vol. 82, pp. 29-63, (1989).

92. Johnson, N., personal communication, Los Alamos National Laboratory, (1990).

93. Roe, P. L., "Characteristic-Based Schemes for the Euler Equations," *Annual Review of Fluid Mechanics*, Vol. 18, pp. 337-365, (1986).

94. Hancock, S., "Numerical Studies of a One-Dimensional Lagrangian Godunov Scheme," Lecture notes from Mechanical Engineering 266, University of California, Berkeley, (1976).

95. Hageman, L. J., Wilkins, D. E., Sedgwick, R. T., and Waddell, J. L., HELP A Multimaterial Eulerian Program for Compressible Fluid and Elastic-Plastic Flows in Two Space Dimensions and Time, System, Science and Software, SSS-R-75-2654, (1975).

96. Hughes, T. J. R., and Tezduyar, T. E., “Finite Element Methods for First-Order Hyperbolic Systems with Particular Emphasis on the Compressible Euler Equations,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 45, pp. 217-284, (1984).

97. Sod, G. A., “A Survey of Several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws,” *Journal of Computational Physics*, Vol. 27, pp. 1-31, (1978).

98. Roache, P. J., *Computational Fluid Dynamics*, Hermosa Publishers, 1976.

99. Chorin, A. J., “Flame Advection and Propagation Algorithms,” *Journal of Computational Physics*, Vol. 35, pp. 1-11, (1980).

100. Couch, R., Albright, E., Alexander, N., “The JOY Computer Code,” Lawrence Livermore National Laboratory, UCID-19688, (1983).

101. Holian, K. S., Mandell, D. A., Adams, T. F., Addessio, F. L., Baumgardner, J. R., and Mosso, S. J., “MESA: A 3-D Computer Code for Armor/Anti-Armor Applications,” Proc. of the Supercomputing World Conference, San Francisco, California, (1989).

102. DeBar, R. B., “Fundamentals of the KRAKEN Code,” Lawrence Livermore Laboratory, UCIR-760, (1974).

103. Benson, D., “An Efficient, Accurate, Simple ALE Method for Nonlinear Finite Element Programs,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 72, pp. 305-350, (1989).

104. Sharp, R. W., and Barton, R. T., *HEMP Advection Model*, Lawrence Livermore National Laboratory, UCID-17809 Rev.1, (1981).

105. 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.

106. Woodward, P. R., and Colella, P., “The Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks,” Lawrence Livermore National Laboratory, UCRL-86952, (1982).

107. Chorin, A., Hughes, T. J. R., McCracken, M. F., and Marsden, J. E., “Product Formulas and Numerical Algorithms,” *Commun. Pure Appl. Math.* Vol 31, pp. 205-256 (1978).

108. Simo, J. C., and Ortiz, M., “A Unified Approach to Finite Deformation Elastoplastic Analysis Based on the Use of Hyperelastic Constitutive Equations,” *Computer Methods in Applied Mechanics and Engineering* Vol. 49, pp. 221-245, (1985).

109. Winslow, A. M.,“Equipotential Zoning of Two-Dimensional Meshes”, Lawrence Radiation Laboratory, UCRL-7312, (1963).

110. Predebon, W. W., Anderson, C. E., and Walker, J. D., “Inclusion of Evolutionary Damage Measures in Eulerian Wavecodes,” preprint (1990).

111. 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, (1982), pp. 199-259.

112. Johnson, C., Navert, U., and Pitkaranta, J., “Finite Element Methods for Linear Hyperbolic Problems,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 45, (1984), pp. 285-312.

113. McGlaun, personal communication, Sandia National Laboratories, (1990).

114. Book, D. L., and Boris, J. P., “Flux-corrected Transport I. SHASTA, A Transport Algorithm that Works,” *Journal of Computational Physics*, Vol. 11, pp. 38-69, (1973).

115. van Leer, B., “Toward the Ultimate Conservative Difference Scheme. II. Monotonicity and Conservation Combined in a Second-Order Scheme,” *Journal of computatinal Physics*, Vol. 14, pp. 361-370, (1974).

116. Roe, P. L., “Some Contributions to the Modeling of Discontinuous Flows, ” Lecture Notes in Applied Mathematics 22, Spring-Verlag, New York, pp. 354-359, (1981).

117. Chakravarthy, S. R., and Osher, S., “Computing with High Resolution Upwind Schemes for Hyperbolic Equations,” Large-Scale Computations in Fluid Mechanics, Lectures in Applied Mathematics, Vol. 22, pp. 57-86, (1985).

118. Harten, A., “High Resolution Schemes for Hyperbolic Conservation Laws,” *Journal of Computational Physics*, vol. 49, pp. 357-393, (1983).

119. Sweby, P. K., “High Resolution Schemes Using Flux Limiters for Hyperbolic Conservation Laws,” *SIAM J. Numer. Anal.*, Vol. 21, No. 5, pp. 995-1011, (1984).

120. Osher, S., “Riemann Solvers, the Entropy Condition, and Difference Approximations,” *SIAM J. Numer. Anal.*, Vol. 21, No. 5, pp. 217-235, (1984).

121. Babuska, I., Zienkiewicz, O. C., Gago, J., and de Oliveira, E. R., eds., Accuracy Estimates and Adaptive Refinements in Finite Element Computations, John Wiley and Sons, New York, N. Y., (1986).

122. Dahlquist, G. Bjorck, A., Numerical Methods, Prentice-Hall, Englewood Cliffs, N. J., (1974).

123. Zalesak, S. T., “Fully Multidimensional Flux-Corrected Transport Algorithms for Fluids,” *Journal of Computational Physics*, vol. 31, pp. 335-362, (1979).

124. Lohner, R., “An Adaptive Finite Element Scheme for Transient Problems in CFD,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 61, pp. 323-338, (1987).

125. Oden, J. T., Devloo, P., and Strouboulis, T., “Adaptive Finite Element Methods for the Analysis of Inviscid Compressible Flow: Part I. Fast Refinement/ Unrefinement and Moving Mesh Methods for Unstructured Meshes,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 59, pp. 327-362, (1986).

126. Smolarkiewicz, P. K., “A Fully Multidimensional Positive Definite Advection Transport Algorithm with Small Implicit Diffusion,” JCP, vol. 54, pp. 325-362, (1984).

127. Margolin, L. G., and Smolarkiewicz, P. K., “Antidifuusive Velocities for Multipass Donor Cell Advection,” Lawrence Livermore National Laboratory, UCID-21866, (1989).

128. 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)

129. Bell, J. B., Dawson, C. N., and Shubin, G. R., “An Unsplit, Higher Order Godunov Method for Scalar Conservation Laws in Multiple Dimensions,” JCP, vol. 74, pp. 1-24, (1988).

130. Bell, J. B., Colella, P., Trangenstein, J. A., and Welcome, M., “Adaptive Mesh Refinement on Moving Quadrilateral Grids,” Proc. of the AIAA 9th Computational Fluid Dynamics Conference in Buffalo, New York, (1989).

131. Eidelman, S., Colella, P., and Shreeve, R. P. S., “Application of the Godunov Method and Higher Order Extensionas of the Govunov Method to Cascade Flow Modeling,” AIAA paper 83-1941-CP, AIAA 6th Computational Fluid Dynamics Conference, Danvers, MA., (1983).

132. Van Leer, B., “Multidimensional Explicit Difference Schemes for Hyperbolic Conservation Laws,” *Computing Methods in Applied Sciences and Engineering, VI*, (Elsevier Science, New York, 1984), p. 493.

133. Benson, D. J., “Momentum Advection on a Staggered Mesh,” submitted to *Journal of Computational Physics*, (in press).

134. Benson, D. J., “Vectorization Techniques for Explicit Arbitrary Lagrangian Eulerian Calculations,” submitted to *Computer Methods in Applied Mechanics and Engineering*, (1989)

135. Margolin, L. G., personal communication, (1989).

136. Margolin, L. G., and Beason, C. W., “Remapping on the Staggered Mesh,” Fifth Nuclear Code Developers’ Conference October 11 - 14, 1988, Boulder, CO.

137. Trefethen, L., N., “Group Velocity in Finite Difference Schemes,” SIAM Review, Vol. 24, No. 2, (1982).

138. Addessio, F. L., Baumgardner, J. R., Dukowicz, J. K., Johnson, N. L., Kashiwa, B. A., Rauenzahn, R. M., and Zemach, C. ,“CAVEAT: A Computer Code for Fluid Dynamics Problems with Large Distortion and Internal Slip, (revised edition, 1990),” Los Alamos National Laboratory, (1990).

139. Giannakopoulos, A. E., and Engel, A. J., “Directional Control in Grid Generation,” *J. of Computational Physics*, Vol. 74, pp. 422-439, (1988).

140. Sandler, I. S., and Rubin, D., “An Algorithm and a Modular Subroutine for the Cap Model,” *Int. J. Numer. Analy. Meth. Geomech.*, Vol. 3, pp. 173-186, (1979).

141. Hyman, J. M., “Numerical methods for Tracking Interfaces,” *Physica 12D*, pp. 396-407, (1984).

142. Oran, E. S., and Boris, J. P., Numerical Simulation of Reactive Flow, Elsevier Sciences Publishing Co., Inc., (1987).

143. Johnson, W. E., TOIL (A Two-Material Version of the OIL Code), General Atomic Report GAMD-8073, (1967).

144. Brackbill, J. U., and Saltzman, J. S., “Adaptive Zoning for Singular Problems in Two Dimensions,” *Journal of Computational Physics*, Vol. 46, No. 3, (1982).

145. Hirt, C. W., and Nichols, B. D., “Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries,” *Journal of Computational Physics*, Vol. 39, pp. 201-225, (1981).

146. Sutcliffe, W. G., “BBC Hydrodynamics,” Lawrence Livermore National Laboratory, UCIR-716, (1973).

147. Noh, W. F. and Woodward, P., "SLIC (Simple Line Interface Calculation)," Lecture Notes in Physics 59, Spring-Verlag, Berlin, (1976).

148. Winslow, A. M., and Barton, R. T., *Rescaling of Equipotential Smoothing*, Lawrence Livermore National Laboratory, UCID-19486, (August 1982).

149. Glimm, J., "Solutions in the Large for Nonlinear Hyperbolic Systems of Equations," *Commun. Pure Appl. Math.*, Vol. 18, pp. 697-715, (1965).

150. McMaster, W. H., "Computer Codes for Fluid-Structure Interactions," Proc. of the 1984 Pressure Vessel and Piping Conference, San Antonio, Texas, Lawrence Livermore National Laboratory UCRL-89724 preprint, (1984).

151. Youngs, D. L., "Time Dependent Multi-Material Flow with Large Fluid Distortion," Numerical Methods for Fluid Dynamics, ed. Morton, K. W., and Baines, M. J., pp. 273-285, (1982).

152. Youngs, D. L., "An Interface Tracking Method for a 3D Eulerian Hydrodynamics Code," AWRE Design Mathematics Division, AWRE/44/92/35, (1987).

153. Anderson, C. E., O'Donoghue, P. E., and Skerhut, D., "A Mixture Theory Approach for the Shock Response of Composite Materials," *Journal of Composite Materials*, in press (1990).

154. Harlow, F. H., and Amsden, A. A., "Flow of Interpenetrating Material Phases," *Journal of Computational Physics*, Vol. 18, pp. 440-464, (1975).

155. Drumheller, D. S., "Hypervelocity Impact of Mixtures," Proc. of the 1986 Hypervelocity Impact Symposium, San Antonio, Texas, Oct. 21-24, 1986, published in *International Journal of Impact Engineering* Vol. 5, No. 1-4, (1987).

156. Winslow, A. M., "Adaptive Mesh Zoning by the Equipotential Method," Lawrence Livermore National Laboratory, UCID-19062, (1981).

157. Zienkiewicz, O. C., Xi-Kui, L., and Nakazawa, S., “Iterative Solution of Mixed Problems and Stress Recovery Procedures,” *Comp. Appl. Num. Meths.*, Vol. 1, pp. 3-9, (1985)

158. Milgram, M. S., “Does a Point Lie Inside a Polygon,” *Journal of Computational Physics*, Vol. 84, pp. 134-144, (1989).

159. Dukowicz, J. K., “Conservative Rezoning (Remapping) for General Quadrilateral Meshes,” *Journal of Computational Physics*, Vol. 54, pp. 411-424, (1984).

160. Dukowicz, J. K., and Kodis, J. W., “Accurate Conservative Remapping (Rezoning) for Arbitrary Lagrangian-Eulerian Computations,” *SIAM J. Sci. Stat. Comput.*, Vol. 5, No. 3, pp. 305-321, (1987).

161. Thorn, B. J., and Holdridge, D. B., “The TOOREZ Lagrangian Rezoning Code,” Sandia National Laboratories, SLA-73-1057, (1973).

162. Evans, M. E., and Harlow, F. H., “The Particle-in-Cell Method for Hydrodynamic Calculations,” Los Alamos National Laboratory, LA-2139, (1957).

163. Thompson, J. F., Warsi, Z. U. A., and Mastin, C. W., “Boundary-Fitted Coordinate Systems for Numerical Solution of Partial Differential Equations — A Review,” *Journal of Computational Physics*, Vol. 47, (1982), pp. 1-108.

164. Thompson, J. F., Warsi, Z. U. A., and Mastin, C. W., Numerical Grid Generation Foundations and Applications, North Holland, New York, N. Y., (1985).

165. Trulio, J. G., and Trigger, K. R., “Numerical Solution of the One-Dimensional Lagrangian Hydrodynamic Equations,” Lawrence Radiation Laboratory, UCRL-6267, (1961).

166. Hancock, S., personal communication, (1991).

166. Burton, D. E., "Exact Conservation of Energy and Momentum in Staggered-Grid Hydrodynamics with Arbitrary Connectivity," Lawrence Livermore National Laboratory, UCRL-JC-104258 preprint, also *The Proceedings of the Next Free Lagrange Conference*, Jackson Lake Lodge, Wyoming, June 3-7,(1990).

167. Lax, P. D., "Weak Solutions of Nonlinear Hyperbolic Equations and Their Numerical Computation," *Communications on Pure and Applied Mathematics*, Vol. 7, pp. 159-193, (1954).

168. Harten, A., Hyman, J. M., and Lax, P. D., "On Finite-Difference Approximations and Entropy Conditions for Shocks," *Communications on Pure and Applied Mathematics*, Vol., 29, pp. 297-322, (1976).

169. Belytschko, T., and Bindeman, L. P., "Assumed Strain Stabilization of the 4-node Quadrilateral with 1-Point Quadrature for Nonlinear Problems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 88, pp. 311-340, (1991).

170. Simo, J. C., and Hughes, T. J. R., "On the Variational Foundations of Assumed Strain Methods," *Journal of Applied Mechanics*, Vol. 53, pp. 51-54, (1986).

171. Carpenter, N. J., Taylor, R. L., and Katona, M. G., "Lagrange Constraints for Transient Surface Contact," *International Journal for Numerical Methods in Engineering*, Vol. 32, pp. 103-128, (1991).

172. Johnson, N. J., Addessio, F. L., Baumgardner, J. R., Kashiwa, B. A., Rauenzahn, R. M., "CAVEAT: Current Extensions and Future Capabilities," Los Alamos National Laboratory, LA-??????-MS (draft), (1990)

173. Addessio, F. L., and Johnson, J. N., "CAVEAT Material Model," Los Alamos National Laboratory memorandum, August 7, 1985

174. Dukowicz, J. K., and Meltz, B. J. A., "Vorticity Error in Multidimensional Lagrangian Codes," Los Alamos National Laboratory, LA-UR-90-697 (revised) (1990)

175. Kashiwa, B. and Lee, W. H., "Comparisons Between the Cell-Centered and Staggered Mesh Lagrangian Hydrodynamics," Los Alamos National Laboratory, LA-UR-90-2452, submitted to The Next Free Lagrange Conference, June 3-7, 1990, Jackson Lake Lodge, Wyoming (1990).

176. Lax, P., and Wendroff, B., "Systems of Conservation Laws," *Communications in Pure and Applied Mathematics*, Vol. 13, pp. 217-237, (1960).

178. Davis, S. F., *SIAM J. Sci. Stat. Comput.*, Vol. 8, pp. 1-18, (1987)

179. Brackbill, J. U., and Monaghan, J. J., eds., Particle Methods in Fluid Dynamics and Plasma Physics, Proceedings of the Workshop on Particle Methods in Fluid Dynamics and Plasma Physics, Los Alamos, April 13-15, 1987, published in *Computer Physics Communications*, Vol. 48, No. 1, (1988).

180. Harlow, F. H., "PIC and Its Progeny," *Computer Physics Communications*, Vol. 48, No. 1, pp. 1-10, (1988).

181. Brackbill, J. U., Kothe, D. B., and Ruppel, H. M., "FLIP: A Low-Dissipation, Particle-in-Cell Method for Fluid Flow," *Computer Physics Communications*, Vol. 48, No. 1, pp. 25-38, (1988).

182. Benz, W., "Smooth Particle Hydrodynamics: A Review," Harvard-Smithsonian Center for Astrophysics Preprint Series No. 2884, (1989).

183. Ghosh, S., and Kikuchi, N., "An Arbitrary Lagrangian-Eulerian Finite Element Method for Large Deformation Analysis of Elastic-Viscoplastic Solids," *Computer Methods in Applied Mechanics and Engineering*, Vol. 86, pp. 127-188, (1991).

184. Liu, W. K., Belytschko, T., and Chang, H., "An Arbitrary Lagrangian-Eulerian Finite Element Method for Path-Dependent Materials," *Computer Methods in Applied Mechanics and Engineering*, Vol. 58, pp. 227-245, (1986).

185. P. Lasaint and P. A. Raviart, "On a Finite Element Method for Solving the Neutron Transport Equation," Mathematical Aspects of Finite Elements in Partial Differential Equations, ed. de Boor, C., Academic Press, New York, pp. 89-123, (1974).

186. G. Chavent, Cockburn, B., Cohen, G., Jaffre, J., "A Discontinuous Finite Element Method for Nonlinear Hyperbolic Equations," Innovative Numerical Methods in Engineering, Springer, Berlin, pp. 337-342, (1986).

187. Caussignac, P. and Touzani, R., "Solution of Three-Dimensional Boundary Layer Equations by a Discontinuous Finite Element Method, Part I: Numerical Analysis of a Linear Model Problem," *Computer Methods in Applied Mechanics and Engineering*, Vol. 78, pp. 249-271, (1990).

188. Hughes, T. J. R., Mallet, M., and Mizukami, M., "A New Finite Element Formulation for Computational Fluid Dynamics: II. Beyond SUPG," *Computer Methods In Applied Mechanics and Engineering*, Vol. 54, pp. 341-355, (1986).

189. Johnson, C., Navert, U., Pitkaranta, J., "Finite Element Methods for Linear Hyperbolic Problems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 45, pp. 285-312, (1984).

190. Shu, C. W., "Numerical Experiments on the Accuracy of ENO and Modified ENO Schemes," *Journal of Scientific Computing*, Vol. 5, No. 2, pp. 127-149, (1990).

191. Rogerson, A. M., and Meiburg, E., "A Numerical Study of the Convergence Properties of ENO Schemes, *Journal of Scientific Computing*, Vol. 5, No. 2, pp. 151-167, (1990).

192. Harten, A., and Osher, S. "Uniformly High-Order Accurate Nonoscillatory Schemes," *SIAM Journal of Numerical Analysis*, Vol. 24, pp. 279-309, (1987).

193. Huynh, H. T., "Second-Order Accurate Nonoscillatory Schemes for Scalar Conservation Laws," NASA Technical Memorandum 102010, (1989).

194. Huynh, H. T., "Accurate Monotone Cubic Interpolation," NASA Technical Memorandum 103789, (1991).

195. Lee, W. H., and Kwak, D., "PIC Method for a Two-Dimensional Elastic-Plastic-Hydro Code," *Computer Physics Communications*, Vol. 48, pp. 11-16, (1986).

196. Lee, W. H., and Painter, J. W., “Material Void Opening Computation Using Particle Method,” International Conference on Computational Physics, Beijing, China, (1988).

197. Belytschko, T., Fish, J., and Engelmann, B. E., “A Finite Element with Embedded Localization Zones, *Computer Methods in Applied Mechanics and Engineering*, Vol. 70, pp. 59-89, (1988).

198. Fish, J. and Belytschko, T., “Elements with Embedded Localization Zones for Large Deformation Problems,” *Computers and Structures*, Vol. 30, pp. 247-256, (1988).