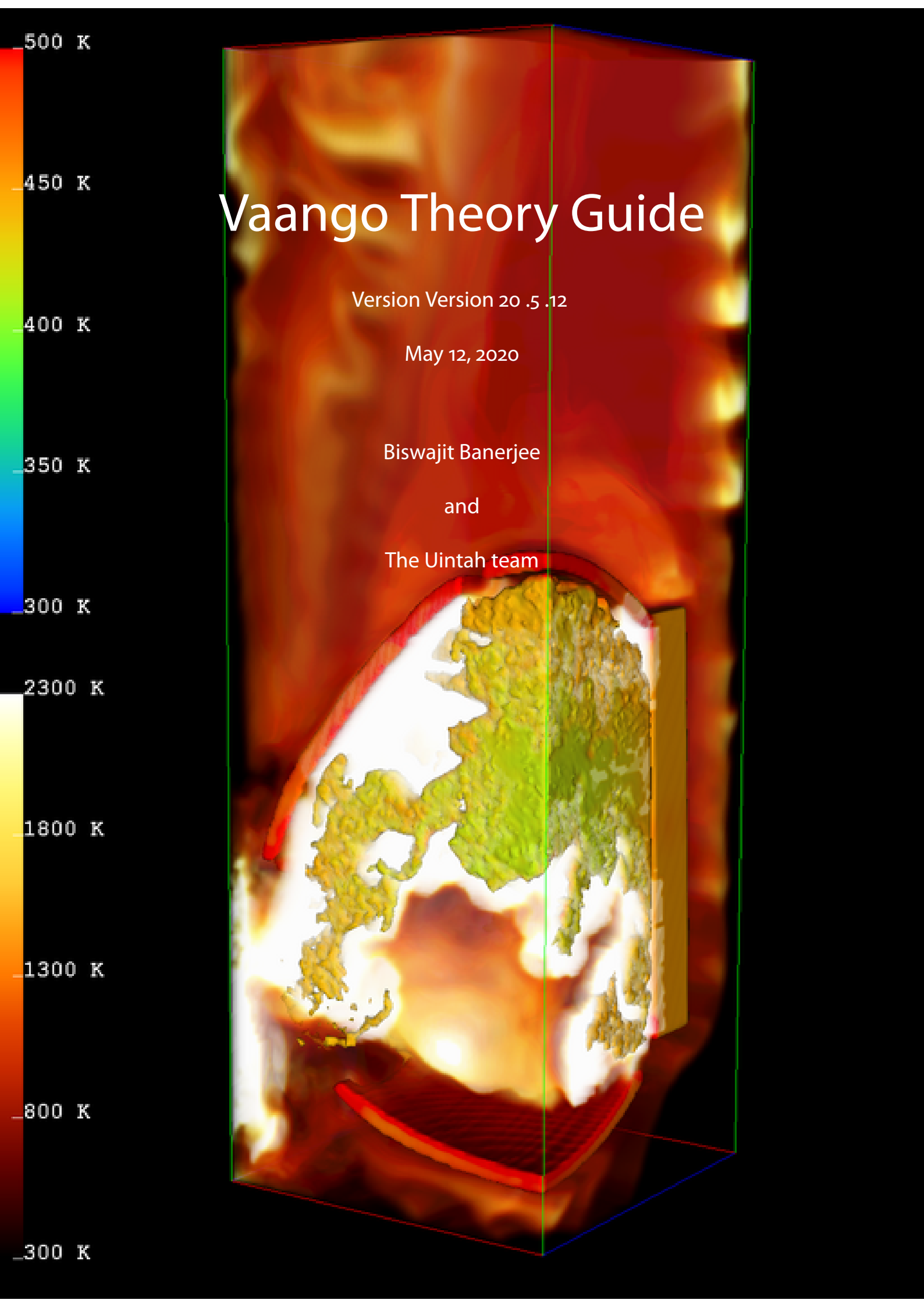


Vaango Theory Guide

Version Version 20 .5 .12

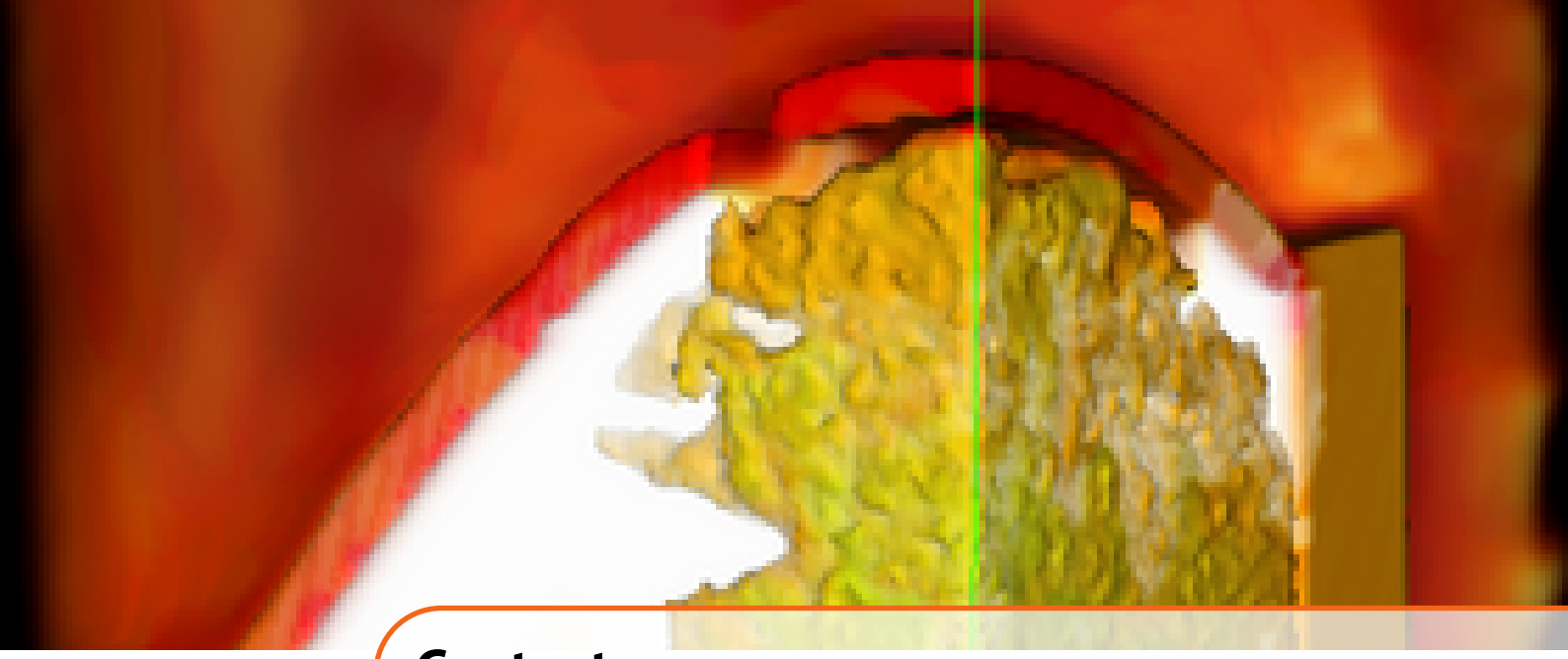
May 12, 2020

Biswajit Banerjee

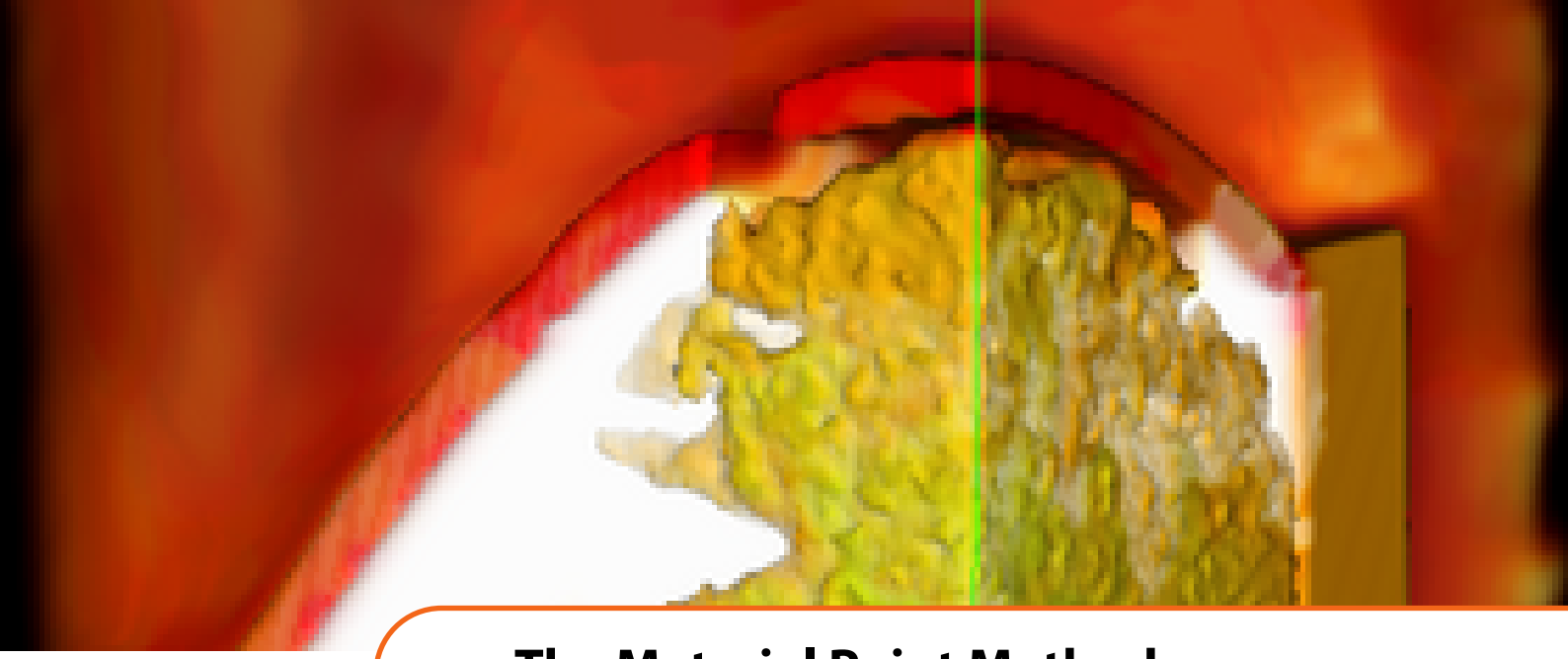
and

The Uintah team

The contents of this manual can and will change significantly over time. Please make sure that all the information is up to date.



Contents



1 — The Material Point Method

1.1 Introduction

The MPM component solves the momentum equations

$$\nabla \cdot \boldsymbol{\sigma} + \rho \mathbf{b} = \rho \dot{\mathbf{v}} \tag{1.1}$$

using an updated Lagrangian formulation. The momentum solve for solid materials is complicated by the fact that the equations need material constitutive models for closure. These material constitutive models vary significantly between materials and contribute a large fraction of the computational cost of a simulation.

The material point method (MPM) was described by Sulsky et al. [1, 2] as an extension to the FLIP (Fluid-Implicit Particle) method of Brackbill [3], which itself is an extension of the particle-in-cell (PIC) method of Harlow [4].

> Interestingly, the name "material point method" first appeared in the literature two years later in a description of an axisymmetric form of the method [5].

In both FLIP and MPM , the basic idea is the same: objects are discretized into particles, or material points, each of which contains all state data for the small region of material that it represents. Particles do not interact with each other directly, rather the particle information is accumulated to a background grid, where the equations of motion (1.1) are integrated forward in time. This time advanced solution is then used to update the particle state. Particle state data includes the position, mass, volume, velocity, stress, state of deformation of that material, and a number of time-dependent internal material variables.

> MPM differs from other "mesh-free" particle methods in that, while each object is primarily represented by a collection of particles, a computational mesh is also an important part of the calculation. This mesh reduces the computational cost of searching for neighboring particles.

MPM usually uses a regular structured grid as a computational mesh. While this grid, in principle, deforms as the material that it is representing deforms, at the end of each timestep, it is reset to its original undeformed position, in effect providing a new computational grid for each timestep. The use of a regular structured grid for each time step has a number of computational advantages. Computation of spatial gradients is simplified. Mesh entanglement, which can plague fully Lagrangian techniques, such as the Finite Element Method (FEM), is avoided.

MPM has also been successful in solving problems involving contact between colliding objects, having an

advantage over FEM in that the use of the regular grid eliminates the need for doing costly searches for contact surfaces[6].

In addition to the advantages that MPM brings, as with any numerical technique, it has its own set of shortcomings. It is computationally more expensive than a comparable FEM code. Accuracy for MPM is typically lower than FEM, and errors associated with particles moving around the computational grid can introduce non-physical oscillations into the solution. Finally, numerical difficulties can still arise in simulations involving large deformation that will prematurely terminate the simulation. The severity of all of these issues (except for the expense) has been significantly reduced with the introduction of the Generalized Interpolation Material Point Method, or GIMP [7]. Newer developments such as the CPDI MPM method [8] have also been incorporated into the explicit time integrated MPM in Vaango. Implementation of other approaches along the line of CPDI such as CPDI2 [9] and CPTI [10] is also being considered for future versions.

In addition, MPM can be incorporated with a multi-material CFD algorithm as the structural component in a fluid-structure interaction formulation. This capability was first demonstrated in the CFDLIB codes from Los Alamos by Bryan Kashiwa and co-workers[11]. There, as in the MPMICE component, MPM serves as the Lagrangian description of the solid material in a multimaterial CFD code. Certain elements of the solution procedure are based in the Eulerian CFD algorithm, including intermaterial heat and momentum transfer as well as satisfaction of a multimaterial equation of state. The use of a Lagrangian method such as MPM to advance the solution of the solid material eliminates the diffusion typically associated with Eulerian methods.

1.2 Weak form of the momentum equation

To derive the weak form of the momentum equation (1.1), we multiply the momentum equation with a vector-valued weighting function ($\mathbf{w}$) and integrate over the domain (Ω). The weighting function ($\mathbf{w}$) satisfies velocity boundary conditions on the parts of the boundary where velocities are prescribed. Then,

$$\int_\Omega \mathbf{w}\cdot\left[\nabla\cdot\boldsymbol{\sigma}+\rho\,\mathbf{b}\right]\,d\Omega=\int_\Omega \rho\,\mathbf{w}\cdot\dot{\mathbf{v}}\,d\Omega\,. \tag{1.2}$$

Using the identity $\mathbf{v}\cdot(\nabla\cdot\boldsymbol{S})=\nabla\cdot(\boldsymbol{S}^T\cdot\mathbf{v})-\boldsymbol{S}:\nabla\mathbf{v}$, where $\boldsymbol{S}$ is a second-order tensor valued field and $\mathbf{v}$ is a vector valued field, we have

$$\int_\Omega \left\{\nabla\cdot(\boldsymbol{\sigma}^T\cdot\mathbf{w})-\boldsymbol{\sigma}:\nabla\mathbf{w}+\rho\,\mathbf{w}\cdot\mathbf{b}\right\}\,d\Omega=\int_\Omega \rho\,\mathbf{w}\cdot\dot{\mathbf{v}}\,d\Omega\,.$$

Application the divergence theorem to the divergence of the weighted stress leads to

$$\int_\Gamma \mathbf{n}\cdot(\boldsymbol{\sigma}^T\cdot\mathbf{w})\,d\Gamma+\int_\Omega\left\{-\boldsymbol{\sigma}:\nabla\mathbf{w}+\rho\,\mathbf{w}\cdot\mathbf{b}\right\}\,d\Omega=\int_\Omega \rho\,\mathbf{w}\cdot\dot{\mathbf{v}}\,d\Omega$$

where $\mathbf{n}$ is the outward normal to the surface Γ. Rearranging,

$$\int_\Gamma(\boldsymbol{\sigma}\cdot\mathbf{n})\cdot\mathbf{w}\,d\Gamma-\int_\Omega\boldsymbol{\sigma}:\nabla\mathbf{w}\,d\Omega+\int_\Omega\rho\,\mathbf{w}\cdot\mathbf{b}\,d\Omega=\int_\Omega\rho\,\mathbf{w}\cdot\dot{\mathbf{v}}\,d\Omega\,. \tag{1.3}$$

If the applied surface traction is $\bar{\mathbf{t}}:=\boldsymbol{\sigma}\cdot\mathbf{n}$, since $\mathbf{w}$ is zero on the part of the boundary where velocities/displacements are specified, we get the weak form

$$\int_{\Gamma_t}\bar{\mathbf{t}}\cdot\mathbf{w}\,d\Gamma-\int_\Omega\boldsymbol{\sigma}:\nabla\mathbf{w}\,d\Omega+\int_\Omega\rho\,\mathbf{w}\cdot\mathbf{b}\,d\Omega=\int_\Omega\rho\,\mathbf{w}\cdot\dot{\mathbf{v}}\,d\Omega\,. \tag{1.4}$$

1.3 Information transfer from particles to grid and back

The goal of MPM is to find a unique function $f(\mathbf{x}, t)$ that satisfies the governing equations (1.1) for a given initial set of objects and a set of initial and boundary conditions.

An important underlying assumption in MPM is that continuum field quantities have two equivalent representations – a grid representation and a particle representation. For instance, the representation of a vector field $\mathbf{f}$ can be both

$$\boxed{\mathbf{f}(\mathbf{x}) = \sum_g \mathbf{f}(\mathbf{x}_g) S_g(\mathbf{x}) = \sum_g \mathbf{f}_g S_g(\mathbf{x})} \quad \text{and} \quad \boxed{\mathbf{f}(\mathbf{x}) = \sum_p \mathbf{f}(\mathbf{x}_p) \chi_p(\mathbf{x}) = \sum_p \mathbf{f}_p \chi_p(\mathbf{x})} \tag{1.5}$$

where the subscript g indicates a grid nodal quantity and the subscropt p indicates a particle quantity. A particle centroid is at the location $\mathbf{x}_p$ while a grid node is at $\mathbf{x}_g$. The functions S_g are interpolation functions (also called shape functions) that take values from the grid nodes to points in the computational domain. On the other hand, the functions χ_p are particle characteristic functions. We assume that both these representations are partitions of unity. In the above we have ignored time-dependence for simplicity.

The MPM algorithm is particle-centered. We start with information on particles and then project that information to the grid nodes for the solution of (1.1). After the equations have been solved, the information on the grid can be interpolated back to the particles in preparation for the next timestep. The projection operation from particles to the grid is not as obvious as the interpolation from the grid back to particles and requires some explanation.

Ideally we would like the two representations in (1.5) to produce identical results. However, due to approximation errors, they usually do not. Let $\mathbf{e}(\mathbf{x})$ be the error. Then we can pose a least-squares error minimization problem as

$$\text{Find } \mathbf{f}_g \text{ that minimizes } E = \int_\Omega w(\mathbf{x}) \, \|\mathbf{e}(\mathbf{x})\|^2 \, d\Omega \text{ where } \int_\Omega w(\mathbf{x}) \, d\Omega = 1 \,. \tag{1.6}$$

The domain of integration is the volume Ω and $w(\mathbf{x})$ is a weighting function. Then the minimum of the functional E can be found using

$$\frac{\partial E}{\partial \mathbf{f}_g} = \mathbf{o} \quad \Longrightarrow \quad \int_\Omega w(\mathbf{x}) \left[\frac{\partial \mathbf{e}}{\partial \mathbf{f}_g} \cdot \mathbf{e}(\mathbf{x}) + \mathbf{e}(\mathbf{x}) \cdot \frac{\partial \mathbf{e}}{\partial \mathbf{f}_g} \right] d\Omega = \mathbf{o} \tag{1.7}$$

From (1.5),

$$\frac{\partial \mathbf{e}}{\partial \mathbf{f}_g} = \frac{\partial}{\partial \mathbf{f}_g} \left[\sum_{g'} \mathbf{f}_{g'} S_{g'}(\mathbf{x}) - \sum_p \mathbf{f}_p \chi_p(\mathbf{x}) \right] = \sum_{g'} \frac{\partial \mathbf{f}_{g'}}{\partial \mathbf{f}_g} S_{g'}(\mathbf{x}) = S_g(\mathbf{x}) \, \boldsymbol{I} \,. \tag{1.8}$$

Therefore,

$$\int_\Omega w(\mathbf{x}) S_g(\mathbf{x}) \mathbf{e}(\mathbf{x}) \, d\Omega = \mathbf{o} \quad \Longrightarrow \quad \int_\Omega w(\mathbf{x}) S_g(\mathbf{x}) \left[\sum_{g'} \mathbf{f}_{g'} S_{g'}(\mathbf{x}) - \sum_p \mathbf{f}_p \chi_p(\mathbf{x}) \right] d\Omega = \mathbf{o} \,. \tag{1.9}$$

If we note that the particle characteristic function (χ_p) is required to be zero outside the domain of particle p and 1 inside, rearrangement of the above equation leads to

$$\sum_g \mathbf{f}_g \int_\Omega w(\mathbf{x}) S_{g'}(\mathbf{x}) S_g(\mathbf{x}) \, d\Omega = \sum_p \mathbf{f}_p \int_{\Omega_p} w(\mathbf{x}) S_{g'}(\mathbf{x}) \, d\Omega \,. \tag{1.10}$$

Define

$$\boxed{A_{g'g} := \int_\Omega w(\mathbf{x}) S_{g'}(\mathbf{x}) S_g(\mathbf{x}) \, d\Omega \quad \text{and} \quad B_{g'p} := \int_{\Omega_p} w(\mathbf{x}) S_{g'}(\mathbf{x}) \, d\Omega \,.} \tag{1.11}$$

Equation (1.10) can now be expressed as

$$\sum_g A_{g'g}\mathbf{f}_g = \sum_p B_{g'p}\mathbf{f}_p\,. \tag{1.12}$$

Inverting the relation, we have

$$\mathbf{f}_g = [\mathbf{A}^{-1}]_{gg'}\sum_p B_{g'p}\mathbf{f}_p = \sum_p [\mathbf{A}^{-1}]_{gg'}\,B_{g'p}\mathbf{f}_p \tag{1.13}$$

where $\mathbf{A}$ is the matrix representation of $A_{g'g}$ in (1.12). We can rewrite the above equation as

$$\boxed{\mathbf{f}_g = \sum_p \psi_{gp}\mathbf{f}_p} \tag{1.14}$$

where

$$\boxed{\psi_{gp} := [\mathbf{A}^{-1}]_{gg'}\,B_{g'p}\,.} \tag{1.15}$$

The map in equation (1.14) can be used to project particle quantities to grid nodes. However, some simplification is needed to avoid the need to invert a large matrix.

We can remove the need to invert $\mathbf{S}$ if we diagonalize $S_{g'g}$ using a lumped approximation. In that case

$$A_{g'} = \sum_g A_{g'g} = \int_\Omega w(\mathbf{x})S_{g'}(\mathbf{x})\sum_g S_g(\mathbf{x})\,d\Omega = \int_\Omega w(\mathbf{x})S_{g'}(\mathbf{x})\,d\Omega \tag{1.16}$$

where we have used the partition of unity property of the grid nodal interpolation function. Now note that the integral over the domain Ω can be split into a sum of integrals over particles.

The particle-to-grid projection operations in equations (1.14) and (1.15) can then be expressed as

$$\mathbf{f}_g = \sum_p \psi_{gp}\mathbf{f}_p \quad \text{where} \quad \psi_{gp} = \frac{B_{gp}}{A_g}\,,\;\; A_g = \sum_p B_{gp}\,,\;\; B_{gp} = \int_{\Omega_p} w(\mathbf{x})S_g(\mathbf{x})\,d\Omega\,. \tag{1.17}$$

Going back to (1.5), recall that we had assumed that $\mathbf{f}_p$ was the value of the function $\mathbf{f}(\mathbf{x})$ at the particle centroid, $\mathbf{x}_p$. However, this requirement is not necessary for the development of the projection from particles to the grid. We may, alternatively, define $\mathbf{f}_p$ as

$$\mathbf{f}_p = \frac{1}{W_p}\int_{\Omega_p}\mathbf{f}(\mathbf{x})\omega_p(\mathbf{x})\,d\Omega\,,\quad W_p := \int_{\Omega_p}\omega_p(\mathbf{x})\,d\Omega \tag{1.18}$$

where Ω_p is the particle domain and $\omega_p(\mathbf{x})$ is a weighting function. Also recall that the grid interpolation function has the form

$$\mathbf{f}(\mathbf{x}) = \sum_g \mathbf{f}_g S_g(\mathbf{x})\,. \tag{1.19}$$

Therefore, we can compute the value of a quantity at a particle using the grid interpolation functions by substituting (1.19) into (1.18) to get

$$\mathbf{f}_p = \frac{1}{W_p}\int_{\Omega_p}\left[\sum_g \mathbf{f}_g S_g(\mathbf{x})\right]\omega_p(\mathbf{x})\,d\Omega = \sum_g \mathbf{f}_g\left[\frac{1}{W_p}\int_{\Omega_p} S_g(\mathbf{x})\omega_p(\mathbf{x})\,d\Omega\right]. \tag{1.20}$$

Since the particle domain Ω_p is never known exactly and we would like to avoid determining that domain, we approximate the above equation as

$$\mathbf{f}_p \approx \sum_g \mathbf{f}_g\left[\frac{1}{W_p^\star}\int_{\Omega_p^\star} S_g(\mathbf{x})\omega_p^\star(\mathbf{x})\,d\Omega\right],\quad W_p^\star := \int_{\Omega_p^\star}\omega_p^\star(\mathbf{x})\,d\Omega \tag{1.21}$$

where the alternative weight function $\omega_p^\star(\mathbf{x})$ is defined as

$$\omega_p^\star(\mathbf{x}) := \omega_p(\mathbf{x})\,\chi_p^\star(\mathbf{x})\,d\Omega\,. \tag{1.22}$$

The function $\chi_p^\star(\mathbf{x})$ is called the particle averaging function since it is is not identical to the particle characteristic function $\chi_p(\mathbf{x})$. All that is needed is that the function have compact support in a neighborhood $\Omega_p^\star$ containing particle p.

The grid-to-particle interpolation function can then be expressed as

$$\mathbf{f}_p \approx \sum_g \mathbf{f}_g \left\langle S_{gp}\right\rangle \quad \text{where} \quad \left\langle S_{gp}\right\rangle := \frac{\int_{\Omega_p^\star} S_g(\mathbf{x})\omega_p^\star(\mathbf{x})\,d\Omega}{\int_{\Omega_p^\star} \omega_p^\star(\mathbf{x})\,d\Omega}\,,\quad \omega_p^\star(\mathbf{x}) := \omega_p(\mathbf{x})\,\chi_p^\star(\mathbf{x})\,d\Omega\,. \tag{1.23}$$

We can now make some special assumptions about the weight functions in (1.17) to reduce the projection operation to that use in tradition MPM approaches. Let us assume that

$$B_{gp} = V_p\left\langle S_{gp}\right\rangle \quad\Longrightarrow\quad \int_{\Omega_p} S_g(\mathbf{x})w(\mathbf{x})\,d\Omega = V_p\frac{\int_{\Omega_p} S_g(\mathbf{x})\omega_p(\mathbf{x})\,d\Omega}{\int_{\Omega_p}\omega_p(\mathbf{x})\,d\Omega} \tag{1.24}$$

With that assumption, the particle-to-grid projection operations in equations (1.17) become

$$\mathbf{f}_g = \sum_p \psi_{gp}\mathbf{f}_p \quad \text{where} \quad \psi_{gp} = \frac{V_p\left\langle S_{gp}\right\rangle}{\sum_p V_p\left\langle S_{gp}\right\rangle}\,,\quad \sum_p \psi_{gp} = 1\,. \tag{1.25}$$

1.3.1 Classical MPM

If we wish to recover the classical MPM formulation [2], take $\omega_p(\mathbf{x}) = 1$ and $\chi_p^\star(\mathbf{x}) = V_p\delta(\mathbf{x}-\mathbf{x}_p)$ where V_p is the volume of $\Omega_p^\star = \Omega_p$ and $\delta(\mathbf{x})$ is the Dirac delta function, we have

$$\bar{S}_{gp} := \left\langle S_{gp}\right\rangle = \frac{\int_{\Omega_p} S_g(\mathbf{x})V_p\delta(\mathbf{x}-\mathbf{x}_p)\,d\Omega}{\int_{\Omega_p} V_p\delta(\mathbf{x}-\mathbf{x}_p)\,d\Omega} = S_g(\mathbf{x}_p)\,. \tag{1.26}$$

The gradient of the interpolation function evaluated at the particle is

$$\bar{\mathbf{G}}_{gp} = \left\langle \nabla S_{gp}\right\rangle = \frac{\int_{\Omega_p} \nabla S_g(\mathbf{x})V_p\delta(\mathbf{x}-\mathbf{x}_p)\,d\Omega}{\int_{\Omega_p} V_p\delta(\mathbf{x}-\mathbf{x}_p)\,d\Omega} = \nabla S_g(\mathbf{x}_p)\,. \tag{1.27}$$

1.3.2 GIMP

To recover the GIMP formulation [7], we take $\omega_p(\mathbf{x}) = 1$ and the square pulse function $\chi_p^\star(\mathbf{x}) = 1$ for $\mathbf{x}\in\Omega_p^\star$ and $\chi_p^\star(\mathbf{x}) = 0$ otherwise. The particle domain $\Omega_p^\star$ is assumed to be a rectangular parallelepiped. Then

$$\bar{S}_{gp} := \left\langle S_{gp}\right\rangle = \begin{cases} \frac{1}{V_p^\star}\int_{\Omega_p^\star} S_g(\mathbf{x})\,d\Omega & \text{for } \mathbf{x}\in\Omega_p^\star \\ 0 & \text{otherwise.}\end{cases} \tag{1.28}$$

The gradient of the interpolation function is

$$\bar{\mathbf{G}}_{gp} = \left\langle \nabla S_{gp}\right\rangle = \begin{cases} \frac{1}{V_p^\star}\int_{\Omega_p^\star} \nabla S_g(\mathbf{x})\,d\Omega & \text{for } \mathbf{x}\in\Omega_p^\star \\ 0 & \text{otherwise.}\end{cases} \tag{1.29}$$

1.3.3 CPDI

For the CPDI formulation [8], we take $\omega_p^\star(\mathbf{x}) = 1$ and the particle domain $\Omega_p^\star$ is assumed to be a general parallelepiped that deforms based on the particle deformation gradient. The expression for ϕ_{gp} is similar to that for GIMP except that a modified shape function is used for interpolation:

$$\bar{S}_{gp} := \langle S_{gp} \rangle = \begin{cases} \frac{1}{V_p^\star} \int_{\Omega_p^\star} S_g^\star(\mathbf{x})\, d\Omega & \text{for } \mathbf{x} \in \Omega_p^\star \\ 0 & \text{otherwise.} \end{cases} \tag{1.30}$$

The gradient of the interpolation function is

$$\bar{\mathbf{G}}_{gp} = \langle \nabla S_{gp} \rangle = \begin{cases} \frac{1}{V_p^\star} \int_{\Omega_p^\star} \nabla S_g^\star(\mathbf{x})\, d\Omega & \text{for } \mathbf{x} \in \Omega_p^\star \\ 0 & \text{otherwise.} \end{cases} \tag{1.31}$$

1.3.4 Transfer to and from grid

For the interpolation from grid nodes to particles, the above relations indicate a general relation (see (1.23))

$$\mathbf{f}_p = \sum_g \mathbf{f}_g \bar{S}_{gp}\,. \tag{1.32}$$

For the particle-to-grid projection (see (1.17)), consider the case where $w(\mathbf{x}) = \rho(\mathbf{x})$ where ρ is the mass density. Then,

$$A_g = \int_\Omega \rho(\mathbf{x}) S_g(\mathbf{x})\, d\Omega = m_g\,, \quad B_{gp} = \int_\Omega \rho(\mathbf{x}) S_g(\mathbf{x}) \chi_p(\mathbf{x})\, d\Omega \tag{1.33}$$

and

$$\boxed{m_g \mathbf{f}_g = \sum_p \mathbf{f}_p \int_\Omega \rho(x) S_g(x) \chi_p(\mathbf{x})\, d\Omega = \sum_p \mathbf{f}_p m_p \bar{S}_{gp}\,.} \tag{1.34}$$

where m_g is the grid node mass and m_P is the particle mass. On the other hand, if $w(\mathbf{x}) = 1$, we have

$$A_g = \int_\Omega S_g(\mathbf{x})\, d\Omega = V_g\,, \quad B_{gp} = \int_\Omega S_g(\mathbf{x}) \chi_p(\mathbf{x})\, d\Omega \tag{1.35}$$

and

$$\boxed{V_g \mathbf{f}_g = \sum_p \mathbf{f}_p \int_\Omega S_g(x) \chi_p(\mathbf{x})\, d\Omega = \sum_p \mathbf{f}_p V_p \bar{S}_{gp}\,.} \tag{1.36}$$

where V_g is the grid node volume and V_p is the particle volume.

In MPM , the momentum per unit volume ($\mathbf{p} = \rho \mathbf{v}$), where ρ is the density and $\mathbf{v}$ is the velocity, is projected to grid nodes using the volume-weighted approach in (1.36):

$$V_g \mathbf{p}_g = \sum_p \mathbf{p}_p V_p \bar{S}_{gp}\,. \tag{1.37}$$

This implies that the velocity is projected using mass weighting (equation 1.34). Further details of the actual projection operators used in Vaango are discussed next.

1.4 MPM discretization of the weak form

The weak form of the momentum equation is

$$\int_{\Gamma_t} \bar{\mathbf{t}} \cdot \mathbf{w}\, d\Gamma - \int_{\Omega} \boldsymbol{\sigma} : \nabla \mathbf{w}\, d\Omega + \int_{\Omega} \rho\, \mathbf{w} \cdot \mathbf{b}\, d\Omega = \int_{\Omega} \rho\, \mathbf{w} \cdot \dot{\mathbf{v}}\, d\Omega\,. \tag{1.38}$$

To discretize the weak form we can use either of the assumed description of field variables shown in (1.5). The grid node-based discretization is used in finite elements while MPM uses the particle-based discretization but also a grid-based approximation.

Recall from (1.5) and (1.25) that

$$\mathbf{f}(\mathbf{x}) = \sum_g \mathbf{f}_g S_g(\mathbf{x}) \quad \text{and} \quad \mathbf{f}_g = \sum_p \psi_{gp} \mathbf{f}_p\,, \;\; \psi_{gp} = \frac{V_p \left\langle S_{gp} \right\rangle}{\sum_p V_p \left\langle S_{gp} \right\rangle}\,, \;\; \sum_p \psi_{gp} = 1\,. \tag{1.39}$$

Therefore, we can write

$$\mathbf{f}(\mathbf{x}) = \sum_g \sum_p \psi_{gp} \mathbf{f}_p S_g(\mathbf{x}) = \sum_p \mathbf{f}_p \sum_g \psi_{gp} S_g(\mathbf{x}) = \sum_p \mathbf{f}_p Y_p(\mathbf{x}) \tag{1.40}$$

where the particle basis functions, Y_p, are defined as

$$Y_p(\mathbf{x}) := \sum_g \psi_{gp} S_g(\mathbf{x}) = \frac{V_p \sum_g \left\langle S_{gp} \right\rangle S_g(\mathbf{x})}{\sum_p V_p \left\langle S_{gp} \right\rangle} \tag{1.41}$$

If we compare (1.40) with the particle representation in (1.5):

$$\mathbf{f}(\mathbf{x}) = \sum_p \mathbf{f}_p \chi_p(\mathbf{x}) \tag{1.42}$$

we see that for the grid-based and particle-based representations to be both accurate representations of the field we need the χ_p and Y_p values to be related by

$$\int_{\Omega} w(\mathbf{x}) \chi_p(\mathbf{x}) = \int_{\Omega} w(\mathbf{x}) Y_p(\mathbf{x}) \tag{1.43}$$

because they cannot be point-wise identical unless the particle characteristic functions satisfy the Kronecker property exactly. We will use the Y_p particle basis functions to discretize the momentum equation.

The first step in the MPM discretization is to convert the integrals over Ω in (1.38) into a sum of integrals over particles using the particle basic functions, Y_p:

$$\begin{aligned} &\int_{\Gamma_t} \bar{\mathbf{t}}(\mathbf{x}) \cdot \mathbf{w}(\mathbf{x})\, d\Gamma - \sum_p \int_{\Omega_p} Y_p(\mathbf{x}) \boldsymbol{\sigma}_p : \nabla \mathbf{w}\, d\Omega + \sum_p \int_{\Omega_p} Y_p(\mathbf{x}) \rho_p \mathbf{w}(\mathbf{x}) \cdot \mathbf{b}_p\, d\Omega \\ &\qquad = \sum_p \int_{\Omega_p} Y_p(\mathbf{x}) \rho_p \mathbf{w}(\mathbf{x}) \cdot \dot{\mathbf{v}}(\mathbf{x})\, d\Omega\,. \end{aligned} \tag{1.44}$$

The weighting function, the velocity, and the material time derivative of $\mathbf{v}$ are approximated as (see [2]):

$$\mathbf{w}(\mathbf{x}) = \sum_g \mathbf{w}_g S_g(\mathbf{x})\,, \;\; \mathbf{v}(\mathbf{x}) = \sum_h \mathbf{v}_h S_h(\mathbf{x})\,, \;\; \dot{\mathbf{v}}(\mathbf{x}) \approx \sum_h \dot{\mathbf{v}}_h S_h(\mathbf{x})\,. \tag{1.45}$$

Plugging these into the left hand side of (1.44) we get

$$\begin{aligned} \text{LHS} = &\int_{\Gamma_t} \bar{\mathbf{t}}(\mathbf{x}) \cdot \left[\sum_g \mathbf{w}_g S_g(\mathbf{x}) \right] d\Gamma - \sum_p \int_{\Omega_p} Y_p(\mathbf{x}) \boldsymbol{\sigma}_p : \left[\sum_g \mathbf{w}_g \otimes \nabla S_g \right] d\Omega \\ &+ \sum_p \int_{\Omega_p} Y_p(\mathbf{x}) \rho_p \left[\sum_g \mathbf{w}_g S_g(\mathbf{x}) \right] \cdot \mathbf{b}_p\, d\Omega \end{aligned} \tag{1.46}$$

Rearranging,

$$\text{LHS} = \sum_g \mathbf{w}_g \cdot \left[\int_{\Gamma_t} \bar{\mathbf{t}}(\mathbf{x}) S_g(\mathbf{x}) \, d\Gamma - \sum_p \int_{\Omega_p} Y_p(\mathbf{x}) \boldsymbol{\sigma}_p \cdot \nabla S_g \, d\Omega + \sum_p \int_{\Omega_p} \rho_p Y_p(\mathbf{x}) S_g(\mathbf{x}) \mathbf{b}_p \, d\Omega \right] \tag{1.47}$$

Similarly, the right hand side of (1.44) can be written as

$$\text{RHS} = \sum_p \int_{\Omega_p} Y_p(\mathbf{x}) \rho_p \left[\sum_g \mathbf{w}_g S_g(\mathbf{x}) \right] \cdot \left[\sum_h \dot{\mathbf{v}}_h S_h(\mathbf{x}) \right] d\Omega \,. \tag{1.48}$$

Rearrangement leads to

$$\text{RHS} = \sum_g \mathbf{w}_g \cdot \sum_h \left[\sum_p \int_{\Omega_p} \rho_p Y_p(\mathbf{x}) S_g(\mathbf{x}) S_h(\mathbf{x}) \dot{\mathbf{v}}_h \, d\Omega \right] . \tag{1.49}$$

Combining the left and right hand sides and invoking the arbitrariness of $\mathbf{w}_g$, for N_g grid points we get equations for $g = 1, 2, \ldots, N_g$:

$$\int_{\Gamma_t} \bar{\mathbf{t}}(\mathbf{x}) S_g(\mathbf{x}) \, d\Gamma - \sum_p \int_{\Omega_p} Y_p(\mathbf{x}) \boldsymbol{\sigma}_p \cdot \nabla S_g \, d\Omega + \sum_p \int_{\Omega_p} \rho_p Y_p(\mathbf{x}) S_g(\mathbf{x}) \mathbf{b}_p \, d\Omega = \sum_h \left[\sum_p \int_{\Omega_p} \rho_p Y_p(\mathbf{x}) S_g(\mathbf{x}) S_h(\mathbf{x}) \dot{\mathbf{v}}_h \, d\Omega \right] . \tag{1.50}$$

We can simplify the above equations further by taking the particle variables outside the integral by assuming they are constant over a particle domain:

$$\int_{\Gamma_t} \bar{\mathbf{t}}(\mathbf{x}) S_g(\mathbf{x}) \, d\Gamma - \sum_p \boldsymbol{\sigma}_p \cdot \left[\int_{\Omega_p} Y_p(\mathbf{x}) \nabla S_g \, d\Omega \right] + \sum_p \rho_p \mathbf{b}_p \left[\int_{\Omega_p} Y_p(\mathbf{x}) S_g(\mathbf{x}) \, d\Omega \right] = \sum_h \sum_p \rho_p \left[\int_{\Omega_p} Y_p(\mathbf{x}) S_g(\mathbf{x}) S_h(\mathbf{x}) \, d\Omega \right] \dot{\mathbf{v}}_h \,. \tag{1.51}$$

Recalling that Y_p has the same effect as χ_p when integrated over a particle volume, we can write

$$\langle S_{gp} \rangle := \frac{1}{V_p} \int_{\Omega_p} Y_p(\mathbf{x}) \, S_g(\mathbf{x}) \, d\Omega \,. \tag{1.52}$$

Then (1.51) can be expressed as

$$\int_{\Gamma_t} \bar{\mathbf{t}}(\mathbf{x}) S_g(\mathbf{x}) \, d\Gamma - \sum_p V_p \boldsymbol{\sigma}_p \cdot \langle \nabla S_{gp} \rangle + \sum_p V_p \rho_p \mathbf{b}_p \langle S_{gp} \rangle = \sum_h \sum_p \rho_p \left[\int_{\Omega_p} Y_p(\mathbf{x}) S_g(\mathbf{x}) S_h(\mathbf{x}) \, d\Omega \right] \dot{\mathbf{v}}_h \,. \tag{1.53}$$

Define the mass matrix ($\boldsymbol{M}$), the internal force vector ($\mathbf{f}_g^{\text{int}}$), the body force vector ($\mathbf{f}_g^{\text{body}}$), and the external force vector ($\mathbf{f}_g^{\text{ext}}$) at grid node g as

$$\begin{aligned} M_{gh} &:= \sum_p \rho_p \int_{\Omega_p} Y_p(\mathbf{x})\, S_g(\mathbf{x})\, S_h(\mathbf{x})\, d\Omega \\ \mathbf{f}_g^{\text{int}} &:= \sum_p V_p \boldsymbol{\sigma}_p \cdot \langle \nabla S_{gp} \rangle \\ \mathbf{f}_g^{\text{body}} &:= \sum_p m_p \mathbf{b}_p \, \langle S_{gp} \rangle \\ \mathbf{f}_g^{\text{ext}} &:= \int_{\Gamma_t} \bar{\mathbf{t}}(\mathbf{x})\, S_g(\mathbf{x})\, d\Gamma \,. \end{aligned} \tag{1.54}$$

Then, from (1.53) we get the semi-discrete system of equations

$$\sum_h M_{gh} \dot{\mathbf{v}}_h = \mathbf{f}_g^{\text{ext}} - \mathbf{f}_g^{\text{int}} + \mathbf{f}_g^{\text{body}} \; ; \quad g = 1 \ldots n_g \tag{1.55}$$

The mass matrix is typically lumped such that

$$\begin{aligned} m_g = \sum_h M_{gh} &= \sum_p \rho_p \int_{\Omega_p} Y_p(\mathbf{x})\, S_g(\mathbf{x}) \left[\sum_h S_h(\mathbf{x}) \right] d\Omega = \sum_p \rho_p \int_{\Omega_p} Y_p(\mathbf{x})\, S_g(\mathbf{x})\, d\Omega \\ &= \sum_p \rho_p V_p \langle S_{gp} \rangle = \sum_p m_p \langle S_{gp} \rangle \,. \end{aligned} \tag{1.56}$$

In that case the semi-discrete system of equations simplifes to

$$m_g \dot{\mathbf{v}}_g = \mathbf{f}_g^{\text{ext}} - \mathbf{f}_g^{\text{int}} + \mathbf{f}_g^{\text{body}} \; ; \quad g = 1 \ldots n_g \tag{1.57}$$

The external force at grid nodes is more difficult to estimate and is typically computed from particle values using

$$\mathbf{f}_g^{\text{ext}} = \sum_p \mathbf{f}_p^{\text{ext}} \langle S_{gp} \rangle \,. \tag{1.58}$$

1.5 Algorithm Description

The interested reader should consult [1, 2] for the development of the discrete equations in MPM discussed in this section, and [7] for the development of the equations for the GIMP method. These end up being very similar, the differences in how the two developments affect implementation will be described in Section 1.6.

In solving a structural mechanics problem with MPM , one begins by discretizing the object of interest into a suitable number of particles, or "material points".

What constitutes a suitable number is something of an open question, but it is typically advisable to use at least two particles in each computational cell in each direction, i.e. 4 particles per cell (PPC) in 2-D, 8 PPC in 3-D.

In choosing the resolution of the computational grid, similar considerations apply as for any computational method (trade-off between time to solution and accuracy, use of resolution studies to ensure convergence in results, etc.).) Each of these particles will carry, minimally, the following variables:

- position - $\mathbf{x}_p$
- mass - m_p
- volume - V_p

- velocity - $\mathbf{v}_p$
- stress - $\boldsymbol{\sigma}_p$
- deformation gradient - $\boldsymbol{F}_p$

The description that follows is a recipe for advancing each of these variables from the current (discrete) time t_n to the subsequent time t_{n+1}. Note that particle mass, m_p, typically remains constant throughout a simulation unless solid phase reaction models are utilized, a feature that is not present in VAANGO MPM . (Such models are available in MPMICE , see Section 12.) It is also important to point out that the algorithm for advancing the timestep is based on the so-called Update Stress Last (USL) algorithm.

> The superiority of this approach over the Update Stress First (USF) approach was clearly demonstrated by Wallstedt and Guilkey [12]. USF was the formulation used in Uintah until mid-2008.

The discrete momentum equation that results from the weak form is given as:

$$\mathbf{Ma} = \mathbf{f}^{\text{ext}} - \mathbf{f}^{\text{int}} + \mathbf{f}^{\text{body}} \tag{1.59}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{a}$ is the acceleration vector, $\mathbf{f}^{\text{ext}}$ is the external force vector (sum of the body forces and tractions), and $\mathbf{f}^{\text{int}}$ is the internal force vector resulting from the divergence of the material stresses. The construction of each of these quantities, which are based at the nodes of the computational grid, will be described below.

The solution begins by projecting the particle state to the nodes of the computational grid, to form the mass matrix $\mathbf{M}$ and to find the nodal external forces $\mathbf{f}^{\text{ext}}$, and velocities, $\mathbf{v}$. In practice, a lumped mass matrix is used to avoid the need to invert a system of equations to solve Eq. (1.59) for acceleration. These quantities are calculated at individual nodes by the following equations, where the $\sum_p$ represents a summation over all particles:

$$m_g = \sum_p m_p \bar{S}_{gp} \,, \quad \mathbf{v}_g = \frac{\sum_p m_p \mathbf{v}_p \bar{S}_{gp}}{m_g} \,, \quad \mathbf{f}_g^{\text{ext}} = \sum_p \mathbf{f}_p^{\text{ext}} \bar{S}_{gp} \tag{1.60}$$

and g refers to individual nodes of the grid, m_p is the particle mass, $\mathbf{v}_p$ is the particle velocity, and $\mathbf{f}_p^{\text{ext}}$ is the external force on the particle. The external forces that start on the particles typically the result of tractions, the application of which is discussed in the VAANGO User manual. $\bar{S}_{gp} = \langle S_{gp} \rangle$ is the shape function of the g-th node evaluated at the particle p as discussed in the section 1.3 equation (1.23). The functional form of the shape functions differs between MPM , GIMP , and CPDI . Further details of the difference are given in Section 1.6.

Following the operations in Eq. 1.60, $\mathbf{f}^{\text{int}}$ is still required in order to solve for acceleration at the nodes. This is computed at the nodes as a volume integral of the divergence of the stress on the particles, specifically:

$$\mathbf{f}_g^{\text{int}} = \sum_p V_p \boldsymbol{\sigma}_p \bar{\mathbf{G}}_{gp} \tag{1.61}$$

where $\bar{\mathbf{G}}_{gp}$ is the gradient of the shape function of the g-th node evaluated at the particle p, and $\boldsymbol{\sigma}_p$ and V_p are the time t_n values of particle stress and volume respectively.

Equation (1.59) can then be solved for $\mathbf{a}$.

$$\mathbf{a}_g = \frac{\mathbf{f}_g^{\text{ext}} - \mathbf{f}_g^{\text{int}} + \mathbf{f}_g^{\text{body}}}{m_g} \tag{1.62}$$

In the explicit version of MPM implemented in VAANGO , a forward Euler method is used for the time integration:

$$\mathbf{v}_g^L = \mathbf{v}_g + \mathbf{a}_g \Delta t \quad \text{where} \quad \Delta t = t_{n+1} - t_n \,. \tag{1.63}$$

The time advanced grid velocity, $\mathbf{v}_g^L$ is used to compute a velocity gradient at each particle according to:

$$\nabla \mathbf{v}_p = \sum_g \mathbf{v}_g^L \overline{\mathbf{G}}_{gp} \,. \tag{1.64}$$

This velocity gradient is used to update the particle's deformation gradient, volume and stress. First, an incremental deformation gradient is computed using the velocity gradient:

$$\Delta \boldsymbol{F}_p^{n+1} = (\boldsymbol{I} + \nabla \mathbf{v}_p \Delta t) \tag{1.65}$$

Particle volume and deformation gradient are updated by:

$$V_p^{n+1} = \det(\Delta \boldsymbol{F}_p^{n+1}) V_p^n \,, \quad \boldsymbol{F}_p^{n+1} = \Delta \boldsymbol{F}_p^{n+1} \cdot \boldsymbol{F}_p^n \,. \tag{1.66}$$

Finally, the velocity gradient, and/or the deformation gradient are provided to a constitutive model, which outputs a time advanced stress at the particles.

At this point in the timestep, the particle position and velocity are explicitly updated by:

$$\begin{aligned} \mathbf{v}_p(t + \Delta t) &= \mathbf{v}_p(t) + \sum_g \overline{S}_{gp} \mathbf{a}_g \Delta t \\ \mathbf{x}_p(t + \Delta t) &= \mathbf{x}_p(t) + \sum_g \overline{S}_{gp} \mathbf{v}_g^L \Delta t \end{aligned} \tag{1.67}$$

This completes one timestep, in that the update of all six of the variables enumerated above (with the exception of mass, which is assumed to remain constant) has been accomplished. Conceptually, one can imagine that, since an acceleration and velocity were computed at the grid, and an interval of time has passed, the grid nodes also experienced a displacement. This displacement also moved the particles in an isoparametric fashion. In practice, particle motion is accomplished by Equation 1.67, and the grid never deforms. So, while the MPM literature will often refer to resetting the grid to its original configuration, in fact, this isn't necessary as the grid nodes never leave that configuration. Regardless, at this point, one is ready to advance to the next timestep.

The algorithm described above is the core of the Vaango MPM implementation. However, it neglects a number of important considerations. The first is kinematic boundary conditions on the grid for velocity and acceleration. Next, is the use of advanced contact algorithms. By default, MPM enforces no-slip, no-interpenetration contact. This feature is extremely useful, but it also means that two bodies initially in "contact" (meaning that they both contain particles whose data are accumulated to common nodes) behave as if they are a single body. To enable multi-field simulations with frictional contact, or to impose displacement based boundary conditions, e.g. a rigid piston, additional steps must be taken. These steps implement contact formulations such as that described by Bardenhagen, et al.[13]. The *use* of the contact algorithms is described briefly in this manual, but the reader will be referred to the relevant literature for their development. Lastly, heat conduction is also available in the explicit MPM code, although it may be neglected via a run time option in the input file. Explicit MPM is typically used for high-rate simulations in which heat conduction is negligible.

1.6 Shape functions for MPM, GIMP, and CPDI

In both MPM and GIMP , the basic idea is the same: objects are discretized into particles, or material points, each of which contains all state data for the small region of material that it represents. In MPM , these particles are spatially Dirac delta functions, meaning that the material that each represents is assumed to exist at a single point in space, namely the position of the particle. Interactions between the particles and the grid take place using weighting functions, also known as shape functions or interpolation functions. These are typically, but not necessarily, linear, bilinear or trilinear in one, two and three dimensions, respectively.

Bardenhagen and Kober [7] generalized the development that gives rise to MPM , and suggested that MPM may be thought of as a subset of their "Generalized Interpolation Material Point" (GIMP) method. As discussed in Section 1.3, in the family of GIMP methods one chooses a characteristic function χ_p to represent the particles and a shape function S_g as a basis of support on the computational nodes. An effective shape function $\bar{S}_{gp}$ is found by the convolution of χ_p and S_g which is written as:

$$\bar{S}_{gp}(\mathbf{x}_p) = \frac{1}{V_p} \int_{\Omega_p \cap \Omega} \chi_p(\mathbf{x} - \mathbf{x}_p) S_g(\mathbf{x}) \, \mathrm{d}\mathbf{x}. \tag{1.68}$$

While the user has significant latitude in choosing these two functions, in practice, the choice of S_g is usually given (in one-dimension) as,

$$S_g(x) = \begin{cases} 1 + (x - x_g)/h & -h < x - x_g \leq 0 \\ 1 - (x - x_g)/h & 0 < x - x_g \leq h \\ 0 & \text{otherwise,} \end{cases} \tag{1.69}$$

where x_g is the vertex location, and h is the cell width, assumed to be constant in this formulation, although this is not a general restriction on the method. Multi-dimensional versions are constructed by forming tensor products of the one-dimensional version in the orthogonal directions. In three dimensions,

$$S_g^{\alpha}(r, s, t) = \frac{1}{8}(1 + r\, r_\alpha)(1 + s\, s_\alpha)(1 + t\, t_\alpha) \tag{1.70}$$

and $r, s, t \in [-1, 1]$ are the natural coordinates of the support domain. A plot of the basis function in two-dimensions is shown in Figure 1.1.

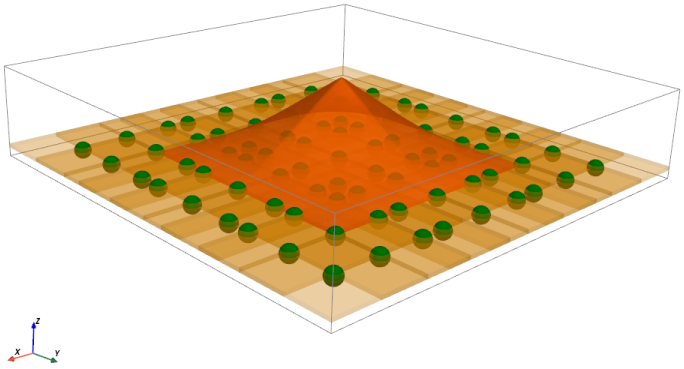


Figure 1.1: Linear grid node shape functions for 2D traditional MPM .

1.6.1 MPM

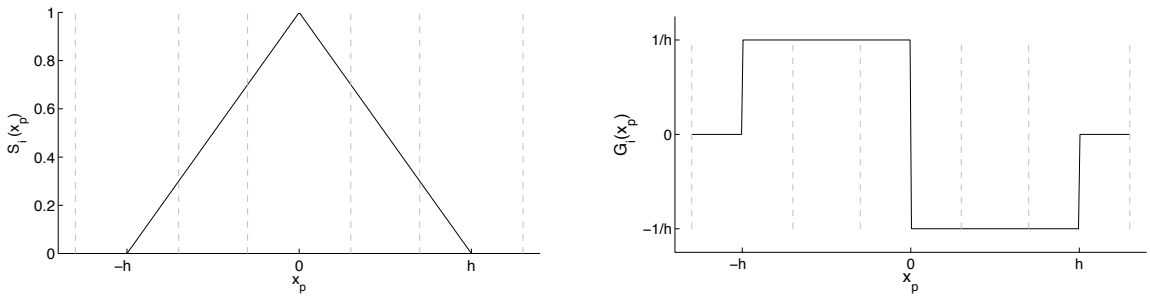
When the choice of characteristic function is the Dirac delta,

$$\chi_p(\mathbf{x}) = \delta(\mathbf{x} - \mathbf{x}_p) V_p, \tag{1.71}$$

where $\mathbf{x}_p$ is the particle position, and V_p is the particle volume, then traditional MPM is recovered. In that case, the effective shape function is still that given by Equation (1.69). Its gradient is given by:

$$G_g(x) = \begin{cases} 1/h & -h < x - x_g \leq 0 \\ -1/h & 0 < x - x_g \leq h \\ 0 & \text{otherwise,} \end{cases} \tag{1.72}$$

Plots of Equations 1.69 and 1.72 are shown below. The discontinuity in the gradient gives rise to poor accuracy and stability properties.



(a) Effective shape function when using traditional MPM . (b) Gradient of the effective shape function when using traditional MPM .

1.6.2 GIMP

Typically, when an analyst indicates that they are "using GIMP " this implies use of the linear grid basis function given in Eq. 1.69 and a "top-hat" characteristic function, given by (in one-dimension),

$$\chi_p(x) = H(x - (x_p - l_p)) - H(x - (x_p + l_p)), \tag{1.73}$$

where $H(x)$ is the Heaviside function ($H(x) = 0$ if $x < 0$ and $H(x) = 1$ if $x \geq 0$) and l_p is the half-length of the particle. When the convolution indicated in Eq. 1.68 is carried out using the expressions in Eqns. 1.69 and 1.73, a closed form for the effective shape function can be written as:

$$\bar{S}_{gp}\left(x_p\right) = \begin{cases} \frac{\left(h+l_p+\left(x_p-x_g\right)\right)^2}{4hl_p} & -h - l_p < x_p - x_g \leq -h + l_p \\ 1 + \frac{\left(x_p-x_g\right)}{h} & -h + l_p < x_p - x_g \leq -l_p \\ 1 - \frac{\left(x_p-x_g\right)^2+l_p^2}{2hl_p} & -l_p < x_p - x_g \leq l_p \\ 1 - \frac{\left(x_p-x_g\right)}{h} & l_p < x_p - x_g \leq h - l_p \\ \frac{\left(h+l_p-\left(x_p-x_g\right)\right)^2}{4hl_p} & h - l_p < x_p - x_g \leq h + l_p \\ 0 & \text{otherwise,} \end{cases} \tag{1.74}$$

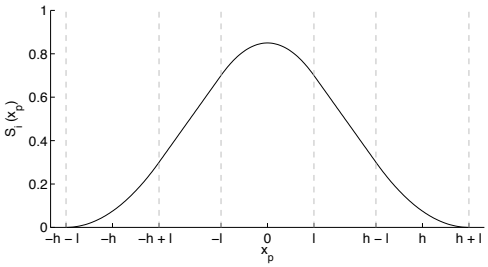
The gradient of which is:

$$\bar{G}_{gp}(x_p) = \begin{cases} \frac{h+l_p+\left(x_p-x_g\right)}{2hl_p} & -h - l_p < x_p - x_g \leq -h + l_p \\ \frac{1}{h} & -h + l_p < x_p - x_g \leq -l_p \\ -\frac{\left(x_p-x_g\right)}{hl_p} & -l_p < x_p - x_g \leq l_p \\ -\frac{1}{h} & l_p < x_p - x_g \leq h - l_p \\ -\frac{h+l_p-\left(x_p-x_g\right)}{2hl_p} & h - l_p < x_p - x_g \leq h + l_p \\ 0 & \text{otherwise,} \end{cases} \tag{1.75}$$

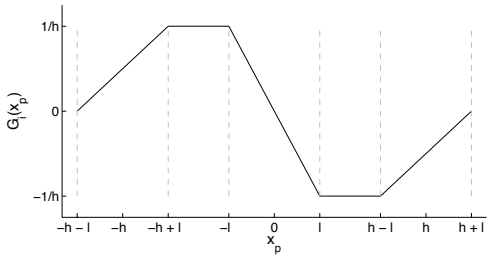
Plots of Equations 1.74 and 1.75 are shown in Figure 1.3. The continuous nature of the gradients are largely responsible for the improved robustness and accuracy of GIMP over MPM .

1.6.3 UGIMP and cpGIMP

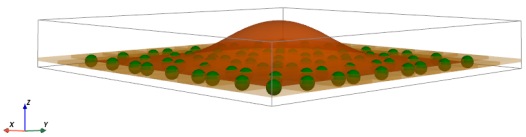
The GIMP effective shape functions in (1.74) are valid only for particle sizes that are smaller than the grid spacing. In Figure 1.4 we see that discontinuities appear in the effective shape function for particles for which $l_p > 0.5h$.



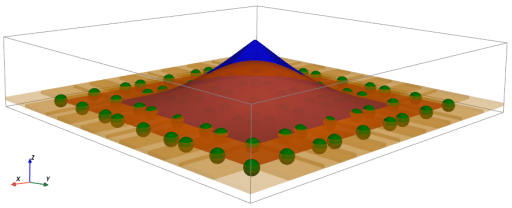
(a) One-dimensional shape function.



(b) Gradient of the one-dimensional shape function.



(c) Two-dimensional shape function.



(d) Two-dimensional GIMP compared to MPM (blue).

Figure 1.3: GIMP effective shape functions and their gradients.

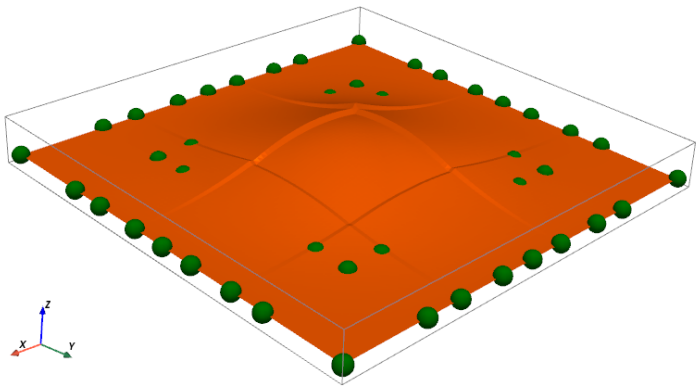


Figure 1.4: Two-dimensional GIMP effective shape functions ($\bar{S}_{gp}$) for $l_p = 0.7h$.

There is one further consideration in defining the effective shape function, and that is whether or not the size (length in 1-D) of the particle is kept fixed (denoted as UGIMP here) or is allowed to evolve due to material deformations ("Finite GIMP" or "Contiguous GIMP" and cpGIMP here). In one-dimensional simulations, evolution of the particle (half-)length is straightforward,

$$l_p^n = \boldsymbol{F}_p^n l_p^o, \tag{1.76}$$

where $\boldsymbol{F}_p^n$ is the deformation gradient at time n. A similar approach is used in CPDI.

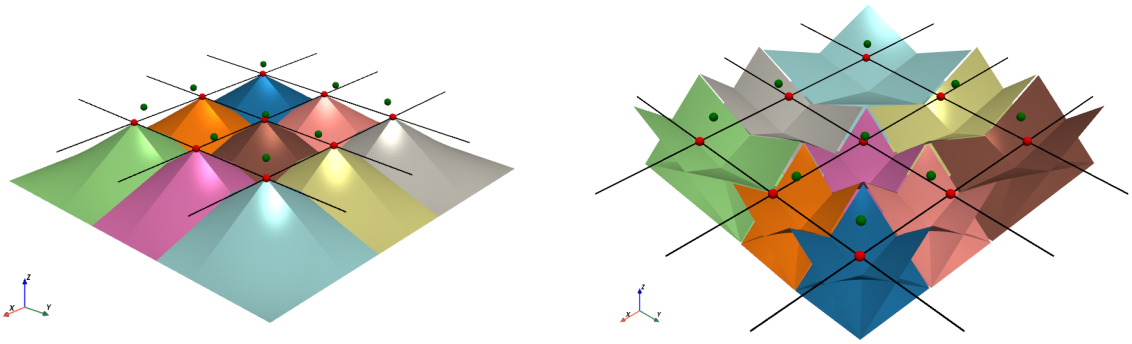
In multi-dimensional simulations, a similar approach can be used, assuming an initially rectangular or cuboid particle, to find the current particle shape. The difficulty arises in evaluating Eq. (1.68) for these general shapes. One approach, apparently effective, has been to create a cuboid that circumscribes the deformed particle shape [14]. Alternatively, one can assume that the particle size remains constant (insofar as it applies to the effective shape function evaluations only).

1.6.4 CPDI

The CPDI formulation [8] is a more recent method for calculating the quantities

$$\langle S_{gp} \rangle = \frac{1}{V_p} \int_{\Omega_p} Y_p(\mathbf{x})\, \tilde{S}_g(\mathbf{x})\, d\Omega \quad \text{and} \quad \langle \nabla S_{gp} \rangle = \frac{1}{V_p} \int_{\Omega_p} Y_p(\mathbf{x})\, \nabla \tilde{S}_g(\mathbf{x})\, d\Omega \tag{1.77}$$

where Y_p are the particle basis functions and $\tilde{S}_g$ are approximate grid basis functions. Figure 1.5 shows examples of two-dimensional grid and particle basis functions that are used in CPDI.



(a) CPDI grid basis functions. (b) CPDI particle basis functions.

Figure 1.5: Two-dimensional CPDI grid and particle basis functions.

In the reference state, the domain Ω_{po} for particle p is assumed to be a parallelepiped spanned by the three vectors $\boldsymbol{r}_p^{io}$, $i = 1, 2, 3$ with origin at the centroid. In the deformed state, these vectors become $\boldsymbol{r}_p^i = \boldsymbol{F}_p \cdot \boldsymbol{r}_p^{io}$ where $\boldsymbol{F}_p$ is the deformation gradient. The corners of the deformed parallelepiped are used in CPDI to create the grid basis functions:

$$\tilde{S}_g(\mathbf{x}) = \sum_{\alpha=1}^{8} N_p^\alpha(\mathbf{x}) S_g(\mathbf{x}_p^\alpha) \tag{1.78}$$

where α are the indices of the vertices of the particle parallelepiped,

$$N_p^\alpha(r,s,t) = \frac{1}{8}(1 + r\, r_\alpha)(1 + s\, s_\alpha)(1 + t\, t_\alpha) \tag{1.79}$$

and r, s, t the natural coordinates of the parallelepiped that range from -1 to 1. The functions $S_g(\mathbf{x})$ are typically chosen to be the hat functions of classical MPM. We can compute the quantities in (1.77) as follows.

Let

$$\mathbf{s}_p^0 = \begin{bmatrix} \boldsymbol{r}_p^{10} & \boldsymbol{r}_p^{20} & \boldsymbol{r}_p^{30} \end{bmatrix} \quad \text{and} \quad \mathbf{s}_p = \begin{bmatrix} \boldsymbol{r}_p^{1} & \boldsymbol{r}_p^{2} & \boldsymbol{r}_p^{3} \end{bmatrix} \quad \text{and} \quad \mathbf{F}_p = \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix}. \tag{1.80}$$

Then $\mathbf{s}_p = \mathbf{F}_p \mathbf{s}_p^0$. Let

$$\mathbf{S}_{gp} = \begin{bmatrix} S_g(\mathbf{x}_p^1) & S_g(\mathbf{x}_p^2) & S_g(\mathbf{x}_p^3) & S_g(\mathbf{x}_p^4) & S_g(\mathbf{x}_p^5) & S_g(\mathbf{x}_p^6) & S_g(\mathbf{x}_p^7) & S_g(\mathbf{x}_p^8) \end{bmatrix}. \tag{1.81}$$

Also, let

$$\mathbf{R} = \begin{bmatrix} -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{1.82}$$

Then,

$$\bar{S}_{gp} = \langle S_{gp} \rangle = \text{mean}(\mathbf{S}_{gp}) \quad \text{and} \quad \bar{\boldsymbol{G}}_{gp} = \langle \nabla S_{gp} \rangle = \frac{1}{8} \mathbf{s}_p^{-T} \mathbf{R}^T \mathbf{S}_{gp}^T. \tag{1.83}$$

1.7 Contact algorithms

The default behavior of MPM is to handle interactions between objects using velocities on the background grid. However, beyond some simple situations, contact requires the application of contact laws. In the Vaango implementation of friction contact, Coulomb friction is assumed. Alternative types of contact, such as adhesive contact. could also be implemented by changing the contact law.

The purpose of the various contact algorithms in Vaango is to correct the grid velocities such that a particular set of contact assumptions are satisfied. The two main algorithms are friction_bard , which is based on [13], and friction_LR , which is described in [15].

Let $m_p, \mathbf{v}_p, \mathbf{p}_p$ be the mass, velocity, and momentum of particle p. Also, let $m_g, \mathbf{v}_g, \mathbf{p}_g$ be the mass, velocity, and momentum at a grid point g due to nearby particles in the region of influence.

Let the interpolation from grid nodes to particles be given by

$$\mathbf{a}_p = \sum_g S_g(\mathbf{x}_p) \mathbf{a}_g \tag{1.84}$$

where $\mathbf{a}_p$ is a particle quantity, $\mathbf{a}_g$ is a grid quantity, and $S_g(\mathbf{x}_p)$ is a scalar interpolation function evaluated at the particle position $\mathbf{x}_p$. The quantities $\mathbf{a}_p$ and $\mathbf{a}_g$ are expressed as column vectors. For scalar quantities, the column vectors are of size 1×1. For vector quantities the size is 3×1, and for second-order symmetric tensors the vector has dimensions 6×1. For example, for a typical 3D vector,

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}_p = \sum_g S_g(\mathbf{x}_p) \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}_g. \tag{1.85}$$

The interpolation from grid to particles can then be expressed in matrix form as

$$\mathbf{a}_p = \boldsymbol{A}_g \cdot \mathbf{s}_{gp} \tag{1.86}$$

where, if S_g is the number of contributing grid nodes,

$$\boldsymbol{A}_g = \begin{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}_1 & \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}_2 & \cdots & \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}_{N_g} \end{bmatrix}, \quad \mathbf{s}_{gp} = \begin{bmatrix} S_1(\mathbf{x}_p) \\ S_2(\mathbf{x}_p) \\ \vdots \\ S_{N_g}(\mathbf{x}_p) \end{bmatrix}. \tag{1.87}$$

Because the number of particles can differ from the number of grid nodes, the projection of quantities from particles to the grid requires a different operator.

An important underlying assumption in MPM is that continuum field quantities have two equivalent representations – a grid representation and a particle representation. For instance, the representation of a vector field $\mathbf{a}$ can be both of

$$\mathbf{a}(\mathbf{x}) = \sum_g \mathbf{a}(\mathbf{x}_g) S_g(\mathbf{x}_g) \quad \text{and} \quad \mathbf{a}(\mathbf{x}) = \sum_p \mathbf{a}(\mathbf{x}_p) \chi_p(\mathbf{x}_p) \,. \tag{1.88}$$

Let that projection operation from particles to grid nodes be

$$\mathbf{a}_g = \sum_p T_p(\mathbf{x}_g) \mathbf{a}_p \tag{1.89}$$

1.8 Pseudocode of MPM algorithm in Vaango

The momentum equation is solved using the MPM algorithm while forward Euler time-stepping is use to integrate time derivatives. The pseudocode of the overall algorithm is given below. The main quantities of interest are:

- $t_{\max}$: The maximum time until which the simulation is to run.
- $t, \Delta t$: The current time ($t = t_n$) and the time step.
- $\boldsymbol{h}_g$: The grid spacing vector.
- m_p : The particle mass.
- V_p^n, V_p^{n+1} : The particle volume at $t = t_n$ and $t = t_{n+1}$.
- $\mathbf{x}_p^n, \mathbf{x}_p^{n+1}$: The particle position at $t = t_n$ and $t = t_{n+1}$.
- $\mathbf{u}_p^n, \mathbf{u}_p^{n+1}$: The particle displacement at $t = t_n$ and $t = t_{n+1}$.
- $\mathbf{v}_p^n, \mathbf{v}_p^{n+1}$: The particle velocity at $t = t_n$ and $t = t_{n+1}$.
- $\boldsymbol{\sigma}_p^n, \boldsymbol{\sigma}_p^{n+1}$: The particle Cauchy stress at time $t = t_n$ and $t = t_{n+1}$.
- $\boldsymbol{F}_p^n, \boldsymbol{F}_p^{n+1}$: The particle deformation gradient at time $t = t_n$ and $t = t_{n+1}$.

1.8.1 Initialization

An outline of the initialization process is described below. Specific details have been discussed in earlier reports. The new quantities introduced in this section are

- n_p : The number of particles used to discretize a body.
- $\mathbf{b}_p^n, \mathbf{b}_p^{n+1}$: The particle body force acceleration at $t = t_n$ and $t = t_{n+1}$.
- D_p^n, D_p^{n+1} : The particle damage parameter at $t = t_n$ and $t = t_{n+1}$.
- $\mathbf{f}_p^{\text{ext},n}, \mathbf{f}_p^{\text{ext},n+1}$: The particle external force at $t = t_n$ and $t = t_{n+1}$.

Algorithm 1 Initialization

Require: `xmlProblemSpec`, `defGradComputer`, `constitutiveModel`, `damageModel`, `particleBC`, `mpmFlags` `materialList`,

1: **procedure** INITIALIZE
2: **for** `matl` **in** `materialList` **do**
3: n_p[matl], $\mathbf{x}_p^o$[matl], $\mathbf{u}_p^o$[matl], m_p[matl], V_p^o[matl], $\mathbf{v}_p^o$[matl], $\mathbf{b}_p^o$[matl], $\mathbf{f}_p^{\text{ext},o}$[matl] ← matl.CREATEPARTICLES()
4: $\boldsymbol{F}_p^o$[matl] ← defGradComputer.INITIALIZE(matl)
5: $\boldsymbol{\sigma}_p^o$[matl] ← constitutiveModel.INITIALIZE(matl)
6: D_p^o[matl] ← damageModel.INITIALIZE(matl)

```
    end for
    if mpmFlags.initializeStressWithBodyForce = TRUE then
        b_p^o ← INITIALIZEBODYFORCE()
        σ_p^o, F_p^o ← INITIALIZESTRESSANDDEFGRADFROMBODYFORCE()
    end if
    if mpmFlags.applyParticleBCs = TRUE then
        f_p^{ext,o} ← particleBC.INITIALIZEPRESSUREBCS()
    end if
    return n_p, x_p^o, u_p^o, m_p, V_p^o, v_p^o, b_p^o, f_p^{ext,o}, F_p^o, σ_p^o, D_p^o
end procedure
```

1.8.2 Time advance

The operations performed during a timestep are shown in the pseudocode below.

Algorithm 2 The MPM time advance algorithm

```
procedure TIMEADVANCE(h_g, x_p^n, u_p^n, m_p, V_p^n, v_p^n, f_p^{ext,n}, d_p^n)
    b_p^n ← COMPUTEPARTICLEBODYFORCE()                ▷Compute the body force term
    f_p^{ext,n+1} ← APPLYEXTERNALLOADS()              ▷Apply external loads to the particles
    m_g, V_g, v_g, b_g, f_g^ext ← INTERPOLATEPARTICLESTOGRID()   ▷Interpolate particle data to the grid
    EXCHANGEMOMENTUMINTERPOLATED()                    ▷Exchange momentum between bodies on grid.
        ↪ Not discussed in this report.
    f_g^int, σ_g, v_g ← COMPUTEINTERNALFORCE()        ▷Compute the internal force at the grid nodes
    v_g^⋆, a_g ← COMPUTEANDINTEGRATEACCELERATION()    ▷Compute the grid velocity
        ↪ and grid acceleration
    EXCHANGEMOMENTUMINTEGRATED()                      ▷Exchange momentum between bodies on grid
        ↪ using integrated values. Not discussed in this report.
    v_g^⋆, a_g ← SETGRIDBOUNDARYCONDITIONS()          ▷Update the grid velocity and grid
        ↪ acceleration using the BCs
    l_p^n, F_p^{n+1}, V_p^{n+1} ← COMPUTEDEFORMATIONGRADIENT()   ▷Compute the velocity gradient
        ↪ and the deformation gradient
    σ_p^{n+1}, η_p^{n+1} ← COMPUTESTRESSTENSOR()      ▷Compute the updated stress and
        ↪ internal variables (if any)
    σ_p^{n+1}, η_p^{n+1}, χ_p^{n+1}, D_p^{n+1} ← COMPUTEBASICDAMAGE()   ▷Compute the damage parameter
        ↪ and update the stress and internal variables
    χ_p^{n+1}, D_p^{n+1} ← UPDATEEROSIONPARAMETER()   ▷Update the indicator variable that is used
        ↪ to delete particles at the end of a time step
    V_p^{n+1}, u_p^{n+1}, v_p^{n+1}, x_p^{n+1}, m_p, h_p^{n+1} ← INTERPOLATETOPARTICLESANDUPDATE()   ▷Update the
        ↪ particle variables after interpolating grid quantities to particles
end procedure
```

The algorithms used for the above operations are discussed next.

Computing the body force

The body force consists of a gravitational term and, optionally, centrifugal and coriolis terms that are needed for simulations inside a rotating frame such as a centrifuge.

Algorithm 3 Computing the body force on particles

```
Require: x_p^n, v_p^n, materialList, particleList, mpmFlags
procedure COMPUTEPARTICLEBODYFORCE
```

```
    for matl in materialList do
        if mpmFlags.rotatingCoordSystem = TRUE then
            g ← mpmFlags.gravityAcceleration
            b_p^n[matl] ← g
        else
            for part in particleList do
                g ← mpmFlags.gravityAcceleration
                x_rc ← mpmFlags.coordRotationCenter
                z_r ← mpmFlags.coordRotationAxis
                w ← mpmFlags.coordRotationSpeed
                ω ← w z_r                                   ▷Compute angular velocity vector
                a_corolis ← 2ω × v_p^n[matl, part]          ▷Compute Coriolis acceleration
                r ← x_p^n[matl, part] − x_rc
                a_centrifugal ← ω × ω × r                   ▷Compute the centrifugal body force acceleration
                b_p^n[matl,part] ← g − a_centrifugal − a_coriolis   ▷Compute the body force acceleration
            end for
        end if
    end for
    return b_p^n
end procedure
```

Applying external loads

Note that the updated deformation gradient has not been computed yet at this stage and the particle force is applied based on the deformation gradient at the beginning of the timestep. The new quantities introduced in this section are:

- $\boldsymbol{h}_p^n$: The particle size matrix at time $t = t_n$.

Algorithm 4 Applying external loads to particles

```
Require: t_{n+1}, x_p^n, h_p^n, u_p^n, f_p^{ext,n}, F_p^n, materialList, particleList, mpmFlags, particleBC
procedure APPLYEXTERNALLOADS
    f_p ← 0
    if mpmFlags.useLoadCurves = TRUE then
        f_p ← particleBC.COMPUTEFORCEPERPARTICLE(t^{n+1})   ▷Compute the force per particle
            ↪ due to the applied pressure
    end if
    for matl in materialList do
        if mpmFlags.useLoadCurves = TRUE then
            for part in particleList do
                f_p^{ext,n+1}[matl,part] ← particleBC.GETFORCEVECTOR(t_{n+1}, x_p^n, h_p^n, u_p^n,
                    ↪ f_p, F_p^n)                           ▷Compute the applied force vector at each particle
            end for
        else
            f_p^{ext,n+1}[matl] ← f_p^{ext,n}[matl]
        end if
    end for
    return f_p^{ext,n+1}
end procedure
```

Interpolating particles to grid

The grid quantities computed during this procedure and not stored for the next timestep except for the purpose of visualization. The new quantities introduced in this section are

- m_g : The mass at a grid node.
- V_g : The volume at a grid node.
- $\mathbf{v}_g$: The velocity at a grid node.
- $\mathbf{f}_g^{\text{ext}}$: The external force at a grid node.
- $\mathbf{b}_g$: The body force at a grid node.

Algorithm 5 Interpolating particle data to background grid

Require: $m_p, V_p^n, \mathbf{x}_p^n, \boldsymbol{h}_p^n, \mathbf{b}_p^n, \mathbf{f}_p^{\text{ext},n+1}, \boldsymbol{F}_p^n$, materialList, particleList, gridNodeList mpmFlags, particleBC

procedure INTERPOLATEPARTICLESTOGRID
  interpolator ← CREATEINTERPOLATOR(mpmFlags) ▷*Create the interpolator and find number of grid nodes that can affect a particle*
  for matl **in** materialList **do**
    for part **in** particleList **do**
      n_{gp}, S_{gp} ← interpolator.FINDCELLSANDWEIGHTS($\mathbf{x}_p^n$, $\boldsymbol{h}_p^n$, $\boldsymbol{F}_p^n$) ▷*Find the node indices of the cells affecting the particle and the interpolation weights*
      $\mathbf{p}_p \leftarrow m_p$[matl][part] $\mathbf{v}_p^n$[matl][part] ▷*Compute particle momentum*
      for node **in** n_{gp} **do**
        m_g[matl][node] ← m_g[matl][node] + m_p[matl][part] S_{gp}[node]
        V_g[matl][node] ← V_g[matl][node] + V_p^n[matl][part] S_{gp}[node]
        $\mathbf{v}_g$[matl][node] ← $\mathbf{v}_g$[matl][node] + $\mathbf{p}_p$ S_{gp}[node]
        $\mathbf{f}_g^{\text{ext}}$[matl][node] ← $\mathbf{f}_g^{\text{ext}}$[matl][node] + $\mathbf{f}_p^{\text{ext},n+1}$[matl][part] S_{gp}[node]
        $\mathbf{b}_g$[node] ← $\mathbf{b}_g$[node] + m_p[matl][part] $\mathbf{b}_p^n$[matl][part] S_{gp}[node]
      end for
    end for
    for node **in** gridNodeList **do**
      $\mathbf{v}_g$[matl][node] ← $\mathbf{v}_g$[matl][node]/m_g[matl][node]
    end for
    $\mathbf{v}_g$[matl] ← APPLYSYMMETRYVELOCITYBC($\mathbf{v}_g$[matl]) ▷*Apply any symmetry velocity BCs that may be applicable*
  end for
  return m_g, V_g, $\mathbf{v}_g$, $\mathbf{b}_g$, $\mathbf{f}_g^{\text{ext}}$
end procedure

Exchanging momentum using interpolated grid values

The exchange of momentum is carried out using a contact model. Details can be found in the Uintah Developers Manual.

Computing the internal force

This procedure computes the internal force at the grid nodes. The new quantities introduced in this section are

- n_{gp} : The number of grid nodes that are used to interpolate from particle to grid.
- S_{gp} : The nodal interpolation function evaluated at a particle
- $\boldsymbol{G}_{gp}$: The gradient of the nodal interpolation function evaluated at a particle
- $\boldsymbol{\sigma}_v$: A volume weighted grid node stress.
- $\mathbf{f}_g^{\text{int}}$: The internal force at a grid node.

Algorithm 6 Computing the internal force

Require: $\boldsymbol{h}_g$, V_g, V_p^n, $\mathbf{x}_p^n$, $\boldsymbol{h}_p^n$, $\boldsymbol{\sigma}_p^n$, $\boldsymbol{F}_p^n$, materialList, particleList, gridNodeList mpmFlags

procedure COMPUTEINTERNALFORCE
  interpolator ← CREATEINTERPOLATOR(mpmFlags) ▷*Create the interpolator and find number of grid nodes that can affect a particle*
  for matl **in** materialList **do**
    for part **in** particleList **do**
      n_{gp}, S_{gp}, $\mathbf{G}_{gp}$ ← interpolator.FINDCELLSANDWEIGHTSANDSHAPEDERVATIVES($\mathbf{x}_p^n$, $\boldsymbol{h}_p^n$, $\boldsymbol{F}_p^n$) ▷*Find the node indices of the cells affecting the particle and the interpolation weights and gradients*
      $\boldsymbol{\sigma}_v \leftarrow V_p$[matl][part] $\boldsymbol{\sigma}_p^n$[matl][part]
      for node **in** n_{gp} **do**
        $\mathbf{f}_g^{\text{int}}$[matl][node] ← $\mathbf{f}_g^{\text{int}}$[matl][node] - ($\mathbf{G}_{gp}$[node]/$\boldsymbol{h}_g$) ·$\boldsymbol{\sigma}_p^n$[matl][part] V_p^n[part]
        $\boldsymbol{\sigma}_g$[matl][node] ← $\boldsymbol{\sigma}_g$[matl][node] + $\boldsymbol{\sigma}_v$ S_{gp}[node]
      end for
    end for
    for node **in** gridNodeList **do**
      $\boldsymbol{\sigma}_g$[matl][node] ← $\boldsymbol{\sigma}_g$[matl][node]/V_g[matl][node]
    end for
    $\mathbf{v}_g$[matl] ← APPLYSYMMETRYTRACTIONBC() ▷*Apply any symmetry tractions BCs that may be applicable*
  end for
  return $\mathbf{f}_g^{\text{int}}$, $\boldsymbol{\sigma}_g$, $\mathbf{v}_g$
end procedure

Computing and integrating the acceleration

This procedure computes the accelerations at the grid nodes and integrates the grid accelerations using forward Euler to compute grid velocities. The new quantities introduced in this section are

- $\mathbf{a}_g$: The grid accelerations.
- $\mathbf{v}_g^\star$: The integrated grid velocities.

Algorithm 7 Computing and integrating the acceleration

Require: Δt, m_g, $\mathbf{f}_g^{\text{int}}$, $\mathbf{f}_g^{\text{ext}}$, $\mathbf{b}_g$, $\mathbf{v}_g$, materialList, gridNodeList, mpmFlags

procedure COMPUTEANDINTEGRATEACCELERATION
  for matl **in** materialList **do**
    for node **in** gridNodeList **do**
      $\mathbf{a}_g$[matl][node] ← ($\mathbf{f}_g^{\text{int}}$[matl][node] + $\mathbf{f}_g^{\text{ext}}$[matl][node] + $\mathbf{b}_g$[matl][node])/m_g[matl][node]
      $\mathbf{v}_g^\star$ ← $\mathbf{v}_g$[matl][node] + $\mathbf{a}_g$[matl][node] $*\Delta t$
    end for
  end for
  return $\mathbf{v}_g^\star$, $\mathbf{a}_g$
end procedure

Exchanging momentum using integrated grid values

The exchange of momentum is carried out using a contact model. Details can be found in the Uintah Developers Manual.

Setting grid boundary conditions

Algorithm 8 Setting grid boundary conditions

Require: Δt, $\mathbf{a}_g$, $\mathbf{v}_g^\star$, $\mathbf{v}_g$, materialList, gridNodeList, mpmFlags
1: **procedure** SETGRIDBOUNDARYCONDITIONS
2: **for** matl **in** materialList **do**
3: $\mathbf{v}_g^\star$[matl] ← APPLYSYMMETRYVELOCITYBC($\mathbf{v}_g^\star$[matl])
4: **for** node **in** gridNodeList **do**
5: $\mathbf{a}_g$[matl][node] ← ($\mathbf{v}_g^\star$[matl][node] - $\mathbf{v}_g$[matl][node]) /Δt
6: **end for**
7: **end for**
8: **return** $\mathbf{v}_g^\star$, $\mathbf{a}_g$
9: **end procedure**

Computing the deformation gradient

The velocity gradient is computed using the integrated grid velocities and then used to compute the deformation gradient. The new quantities introduced in this section are

- $\Delta \boldsymbol{F}_p^n$: The increment of the particle deformation gradient.
- $\boldsymbol{l}_p^{n+1}$: The particle velocity gradient.
- ρ_0 : The initial mass density of the material.

Algorithm 9 Computing the velocity gradient and deformation gradient

Require: Δt, $\mathbf{x}_p^n$, m_p, V_p^n, $\boldsymbol{h}_p^n$, $\mathbf{v}_p^n$, $\boldsymbol{l}_p^n$, $\boldsymbol{F}_p^n$, $\boldsymbol{h}_g$, $\mathbf{v}_g$, $\mathbf{v}_g^\star$, ρ_0 materialList, gridNodeList, mpmFlags, velGradComputer
1: **procedure** COMPUTEDEFORMATIONGRADIENT
2: interpolator ← CREATEINTERPOLATOR(mpmFlags)
3: **for** matl **in** materialList **do**
4: **for** part **in** particleList **do**
5: $\boldsymbol{l}_p^{n+1}$[matl,part] ← velGradComputer.COMPUTEVELGRAD(interpolator, $\boldsymbol{h}_g$, $\mathbf{x}_p^n$[matl,part], ↪ $\boldsymbol{h}_p^n$[matl,part], $\boldsymbol{F}_p^n$[matl,part], $\mathbf{v}_g^\star$[matl]) ▷*Compute the velocity gradient*
6: $\boldsymbol{F}_p^{n+1}$[matl,part], $\Delta \boldsymbol{F}_p^{n+1}$ ← COMPUTEDEFORMATIONGRADIENTFROMVELOCITY($\boldsymbol{l}_p^n$[matl,part], ↪ $\boldsymbol{l}_p^{n+1}$[matl,part], $\boldsymbol{F}_p^n$[matl,part]) ▷*Compute the deformation gradient*
7: V_p^{n+1}[matl,part] ← m_p[matl,part]/ρ_0 ∗ det($\boldsymbol{F}_p^{n+1}$[matl,part])
8: **end for**
9: **end for**
10: **return** $\boldsymbol{l}_p^{n+1}$, $\boldsymbol{F}_p^{n+1}$, V_p^{n+1}
11: **end procedure**

Algorithm 10 Computing the deformation gradient using the velocity gradient

Require: Δt, $\boldsymbol{l}_p^{n+1}$, $\boldsymbol{F}_p^n$, mpmFlags
1: **procedure** COMPUTEDEFORMATIONGRADIENTFROMVELOCITY
2: **if** mpmFlags.defGradAlgorithm = "first_order" **then**
3: $\boldsymbol{F}_p^{n+1}$, $\Delta \boldsymbol{F}_p^{n+1}$ ← SERIESUPDATECONSTANTVELGRAD(numTerms = 1, Δt, $\boldsymbol{l}_p^{n+1}$, $\boldsymbol{F}_p^n$)
4: **else if** mpmFlags.defGradAlgorithm = "subcycle" **then**
5: $\boldsymbol{F}_p^{n+1}$, $\Delta \boldsymbol{F}_p^{n+1}$ ← SUBCYCLEUPDATECONSTANTVELGRAD(Δt, $\boldsymbol{l}_p^{n+1}$, $\boldsymbol{F}_p^n$)
6: **else if** mpmFlags.defGradAlgorithm = "taylor_series" **then**
7: $\boldsymbol{F}_p^{n+1}$, $\Delta \boldsymbol{F}_p^{n+1}$ ← SERIESUPDATECONSTANTVELGRAD(numTerms = mpmFlags.numTaylorSeriesTerms, Δt, $\boldsymbol{l}_p^{n+1}$, $\boldsymbol{F}_p^n$)
8: **else**
9: $\boldsymbol{F}_p^{n+1}$, $\Delta \boldsymbol{F}_p^{n+1}$ ← CAYLEYUPDATECONSTANTVELGRAD(Δt, $\boldsymbol{l}_p^{n+1}$, $\boldsymbol{F}_p^n$)

10: **end if**
11: **return** $\boldsymbol{F}_p^{n+1}, \Delta\boldsymbol{F}_p^{n+1}$
12: **end procedure**

Computing the stress tensor

The stress tensor is compute by individual constitutive models. Details of the Arena partially saturated model are given later. The new quantities introduced in this section are

- $\boldsymbol{\eta}_p^n, \boldsymbol{\eta}_p^{n+1}$: The internal variables needed by the constitutive model.

Algorithm 11 Computing the stress tensor

Require: $\Delta t, \mathbf{x}_p^n, m_p, V_p^{n+1}, \boldsymbol{h}_p^n, \boldsymbol{l}_p^{n+1}, \boldsymbol{F}_p^{n+1}, \boldsymbol{\sigma}_p^n, \boldsymbol{\eta}_p^n, \rho_o$, `materialList`, `mpmFlags`, `constitutiveModel`
1: **procedure** COMPUTESTRESSTENSOR
2: **for** `matl` **in** `materialList` **do**
3: $\boldsymbol{\sigma}^{n+1}, \boldsymbol{\eta}_p^{n+1} \leftarrow$ `constitutiveModel[matl]`.COMPUTESTRESSTENSOR($\Delta t, \mathbf{x}_p^n, m_p, V_p^{n+1}, \boldsymbol{h}_p^n,$ $\boldsymbol{l}_p^{n+1}, \boldsymbol{F}_p^{n+1}, \boldsymbol{\sigma}_p^n, \boldsymbol{\eta}_p^n, \rho_o$, `mpmFlags`) ▷*Update the stress and any internal variables needed by the constitutive model*
4: **end for**
5: **return** $\boldsymbol{\sigma}_p^{n+1}, \boldsymbol{\eta}_p^{n+1}$
6: **end procedure**

Computing the basic damage parameter

The damage parameter is updated and the particle stress is modified in this procedure. The new quantities introduced in this section are

- $\varepsilon_p^{f,n}, \varepsilon_p^{f,n+1}$: The particle strain to failure at $t = T_n$ and $t = T_{n+1}$.
- χ_p^n, χ_p^{n+1} : An indicator function that identifies whether a particle has failed completely.
- $t_p^{\chi,n}, t_p^{\chi,n+1}$: The time to failure of a particle.
- D_p^n, D_p^{n+1} : A particle damage parameter that can be used to modify the stress.

Algorithm 12 Computing the damage parameter

Require: $t^{n+1}, V_p^{n+1}, \boldsymbol{F}_p^{n+1}, \boldsymbol{\sigma}_p^{n+1}, D_p^n, \varepsilon_p^{f,n}, \chi_p^n, t_p^{\chi,n}$, `materialList`, `mpmFlags`
1: **procedure** COMPUTEDAMAGE
2: **for** `matl` **in** `materialList` **do**
3: **for** `part` **in** `particleList` **do**
4: **if** `brittleDamage` = TRUE **then**
5: $\boldsymbol{\sigma}_p^{n+1}, \varepsilon_p^{f,n+1}, \chi_p^{n+1}, t_p^{\chi,n+1}, D_p^{n+1} \leftarrow$ UPDATEDAMAGEANDMODIFYSTRESS($V_p^{n+1}, \boldsymbol{F}_p^{n+1},$ $\boldsymbol{\sigma}_p^{n+1}, D_p^n, \varepsilon_p^{f,n}, \chi_p^n, t_p^{\chi,n}$) ▷*Update the damage parameters and stress*
6: **else**
7: $\boldsymbol{\sigma}_p^{n+1}, \varepsilon_p^{f,n+1}, \chi_p^{n+1}, t_p^{\chi,n+1} \leftarrow$ UPDATEFAILEDPARTICLESANDMODIFYSTRESS($V_p^{n+1}, \boldsymbol{F}_p^{n+1},$ $\boldsymbol{\sigma}_p^{n+1}, \varepsilon_p^{f,n}, \chi_p^n, t_p^{\chi,n}, t^{n+1}$) ▷*Update the failed particles and stress*
8: **end if**
9: **end for**
10: **end for**
11: **return** $\boldsymbol{\sigma}_p^{n+1}, \varepsilon_p^{f,n+1}, \chi_p^{n+1}, t_p^{\chi,n+1}, D_p^{n+1}$
12: **end procedure**

Updating the particle erosion parameter

The particle failure indicator function is updated in this procedure and used later for particle deletion if needed.

Algorithm 13 Updating the particle erosion parameter

Require: D_p^n, χ_p^n `materialList`, `mpmFlags`, `constitutiveModel`

procedure UPDATEEROSIONPARAMETER
  for `matl` **in** `materialList` **do**
    for `part` **in** `particleList` **do**
      if `matl`.doBasicDamage = TRUE **then**
        $\chi_p^{n+1} \leftarrow$ `damageModel`.GETLOCALIZATIONPARAMETER() ▷*Just get the indicator parameter for particles that will be eroded.*
      else
        $\chi_p^{n+1}, D_p^{n+1} \leftarrow$ `constitutiveModel`[`matl`].GETDAMAGEPARAMETER(χ_p^n, D_p^n) ▷*Update the damage parameter in the constitutive model.*
      end if
    end for
  end for
  return χ_p^{n+1}, D_p^{n+1}
end procedure

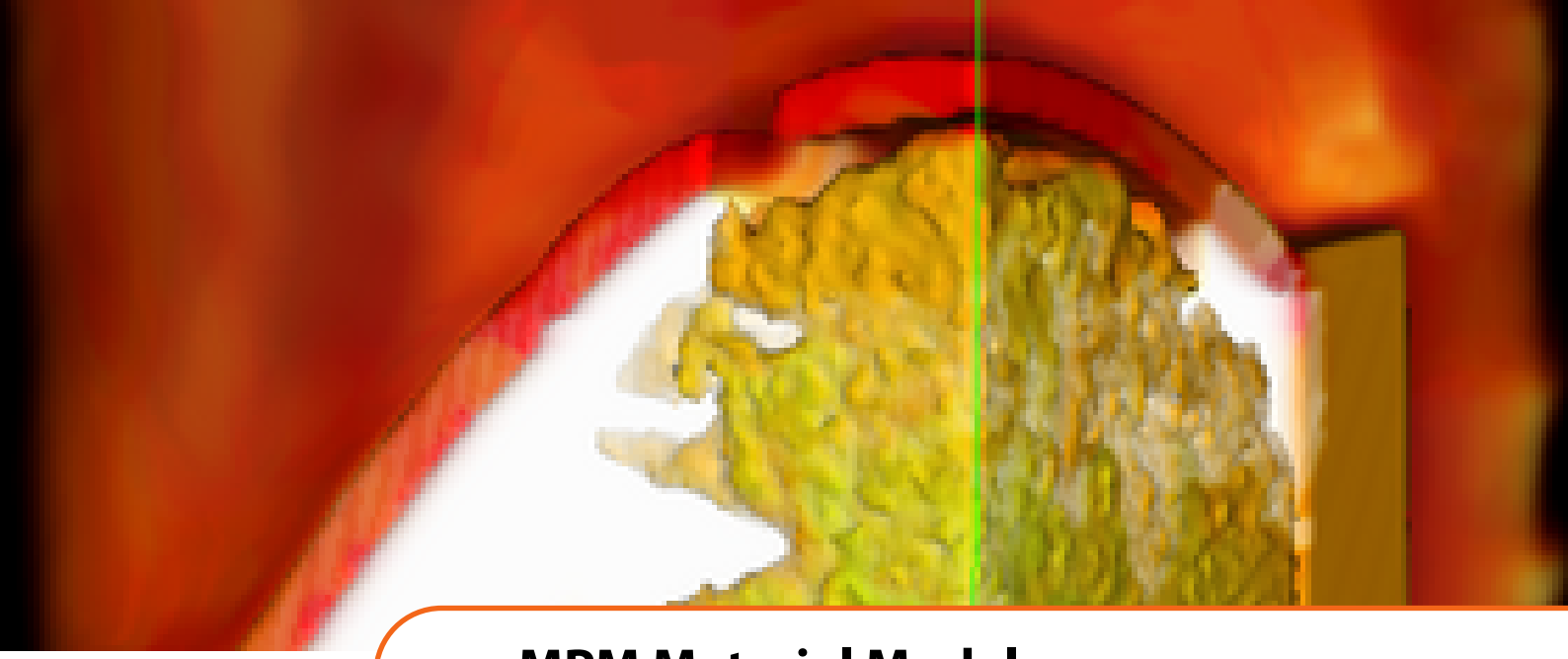
Interpolating back to the particles and update

This is the final step at which the particle velocities and positions are updated and the grid is reset. Particle that are to be removed are dealt with in a subsequent `relocation` step.

Algorithm 14 Interpolating back to the particles and position update

Require: Δt, $\mathbf{a}_g$, $\mathbf{v}_g^\star$, $\mathbf{x}_p^n$, $\mathbf{v}_p^n$, $\mathbf{u}_p^n$, $\boldsymbol{h}_p^n$, χ_p^{n+1}, $\boldsymbol{F}_p^{n+1}$, V_p^{n+1}, `materialList`, `particleList`, `gridNodeList`, `mpmFlags`

procedure INTERPOLATETOPARTICLESANDUPDATE
  `interpolator` $\leftarrow$ CREATEINTERPOLATOR(`mpmFlags`)
  for `matl` **in** `materialList` **do**
    $\boldsymbol{h}_p^{n+1} \leftarrow \boldsymbol{h}_p^n$
    for `part` **in** `particleList` **do**
      $n_{gp}, S_{gp} \leftarrow$ `interpolator`.FINDCELLSANDWEIGHTS($\mathbf{x}_p^n$, $\boldsymbol{h}_p^{n+1}$, $\boldsymbol{F}_p^{n+1}$)
      $\mathbf{v} \leftarrow \mathbf{0}$, $\mathbf{a} \leftarrow \mathbf{0}$,
      for `node` **in** `gridNodeList` **do**
        $\mathbf{v} \leftarrow \mathbf{v} + \mathbf{v}_g^\star$[`node`] $*$ S_{gp}[`node`] ▷*Update particle velocity*
        $\mathbf{a} \leftarrow \mathbf{a} + \mathbf{a}_g$[`node`] $*$ S_{gp}[`node`] ▷*Update particle acceleration*
      end for
      $\mathbf{x}_p^{n+1} \leftarrow \mathbf{x}_p^n + \mathbf{v} * \Delta t$ ▷*Update position*
      $\mathbf{u}_p^{n+1} \leftarrow \mathbf{u}_p^n + \mathbf{v} * \Delta t$ ▷*Update displacement*
      $\mathbf{v}_p^{n+1} \leftarrow \mathbf{v}_p^n + \mathbf{a} * \Delta t$ ▷*Update velocity*
    end for
  end for
  DELETEROGUEPARTICLES() ▷*Delete particles that are to be eroded.*
  return V_p^{n+1}, $\mathbf{u}_p^{n+1}$, $\mathbf{v}_p^{n+1}$, $\mathbf{x}_p^{n+1}$, m_p, $\boldsymbol{h}_p^{n+1}$
end procedure



2 — MPM Material Models

The MPM material models implemented in Vaango we originally chosen for three purposes:

- To verify the accuracy of the material point method (MPM) and to validate the coupling between the computational fluid dynamics code (ICE) and MPM.
- To model the elastic-plastic deformation of metals and the consequent damage in the regimes of both high and low strain rates and high and low temperatures.
- To model polymer bonded explosives under various strain rates and temperatures.
- To model the explosive deformation of rocks and soils.

In this chapter we discuss some general features of the MPM material models. Individual models are complex and are discussed in separate chapters.

2.0.1 Material models for the validation of MPM

The models that have been implemented for the verification of MPM are:

- Isotropic hypoelastic model using the Jaumann rate of stress.
  1. MPM predictions have been compared with exact results for thick cylinders under internal pressure for small strains, three-point beam bending, etc.
  2. MPM predictions for the strain/stress contours for a set of disks in contact have been found to match experimental results.
- Isotropic hyperelastic material models for Mooney-Rivlin rubber and a modified compressible Neo-Hookean material. Isotropic strain hardening plasticity for the Neo-Hookean material.
  1. A billet compression problem has been simulated using MPM and the results have been found to closely match finite element simulations.
  2. MPM simulations for a thick cylinder under internal pressure with plastic deformation (perfect plasticity) compare well with the exact solution.

2.0.2 Material models for metals

The material models for metals are used to determine the state of stress for an applied deformation rate and deformation gradient at each material point. The strain rates can vary from 10^{-3}/s to 10^{6}/s and temperatures in the container can vary from 250 K to 1000 K. For example, in an explosion inside a container, plasticity dominates the deformation of the container during the expansion of the explosive gases inside. At high strain rates the volumetric response of the container is best obtained using an equation of state.

After the plastic strain in the container has reached a threshold value a damage/erosion model is required to rupture the container.

Two plasticity models with strain rate and temperature dependency are the Johnson-Cook and the Mechanical Threshold Stress (MTS) models. The volumetric response is calculated using a modified Mie-Gruneisen equation of state. A damage model that ties in well with the Johnson-Cook plasticity model is the Johnson-Cook damage model. The erosion algorithm either removes the contribution of the mass of the material point or forces the material point to undergo no tension or shear under further loading.

The stress update at each material point is performed using either of the two methods discussed below.

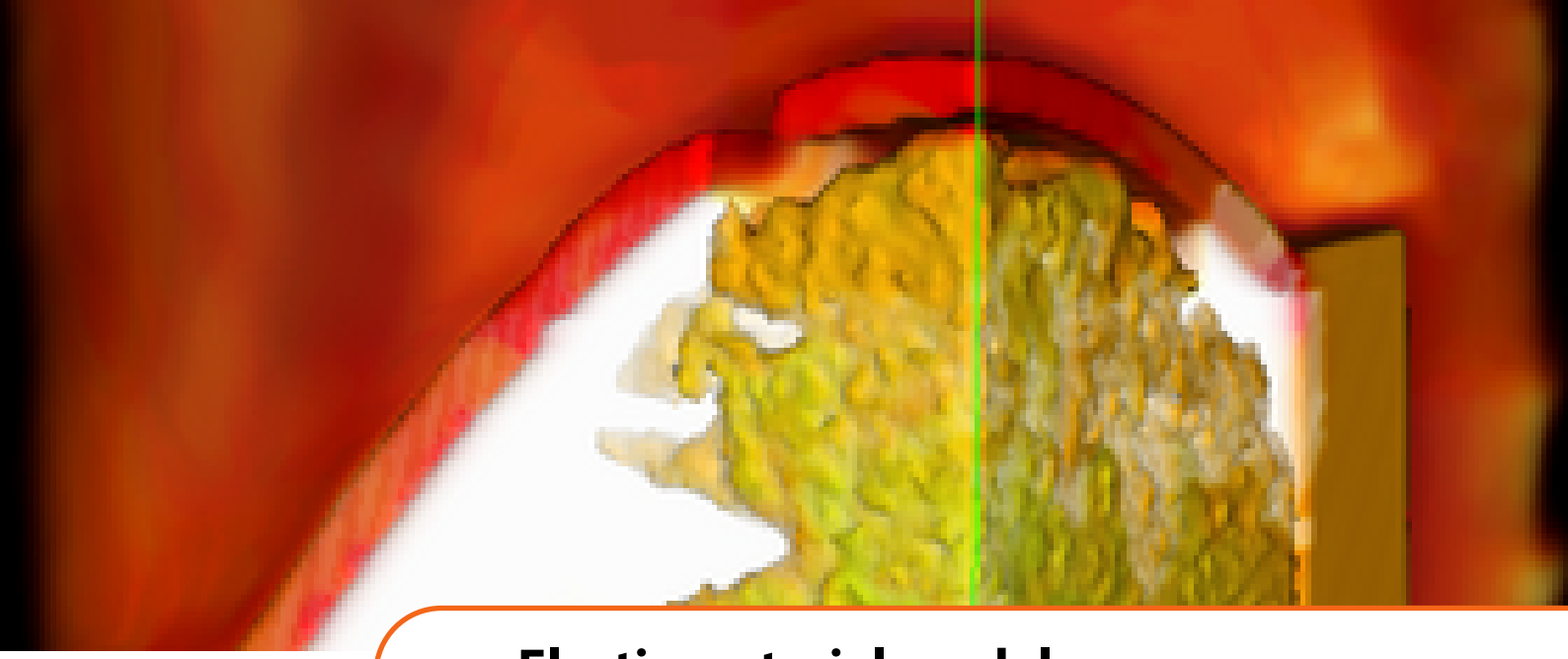
- Isotropic Hypoelastic-plastic material model using an additive decomposition of the rate of deformation tensor.
  1. The rate of deformation tensor a material point is calculated using the grid velocities.
  2. An incremental update of the left stretch and the rate of rotation tensors is calculated.
  3. The stress and the rate of deformation are rotated into the material coordinates.
  4. A trial elastic deviatoric stress state is calculated.
  5. The flow stress is calculated using the plasticity model and compared with the vonMises yield condition.
  6. If the stress state is elastic, an update of the stress is computed using the Mie-Gruneisen equation of state or the isotropic hypoelastic constitutive equation.
  7. If the stress state is plastic, all the strain rate is considered to the plastic and an elastic correction along with a radial return step move the stress state to the yield surface. The hydrostatic part of the stress is calculated using the equation of state or the hypoelastic constitutive equation.
  8. A scalar damage parameter is calculated and used to determine whether material points are to be eroded or not.
  9. Stresses and deformation rates are rotated back to the laboratory coordinates.
- Isotropic Hyperelastic-plastic material model using a multiplicative decomposition of the deformation gradient.
  1. The velocity gradient at a material point is calculated using the grid velocities.
  2. An incremental update of the deformation gradient and the left Cauchy-Green tensor is calculated.
  3. A trial elastic deviatoric stress state is calculated assuming a compressible Neo-Hookean elastic model.
  4. The flow stress is calculated using the plasticity model and compared with the vonMises yield condition.
  5. If the stress state is elastic, an update of the stress is computed using the Mie-Gruneisen equation of state or the compressible Neo-Hookean constitutive equation.
  6. If the stress state is plastic, all the strain rate is considered to the plastic and an elastic correction along with a radial return step move the stress state to the yield surface. The hydrostatic part of the stress state is calculated using the Mie-Gruneisen equation of state or the Neo-Hookean model.
  7. A scalar damage parameter is calculated and used to determine whether material points are to be eroded or not.

The implementations have been tested against Taylor impact test data for 4340 steel and HY 100 steel as well as one-dimensional problems which have been compared with experimental stress-strain data.

2.0.3 Material models for the explosive

The explosive is modeled using the ViscoSCRAM constitutive model. Since large deformations or strains are not expected in the explosive, a small strain formulation has been implemented into Vaango. The model consists of five generalized Maxwell elements arranged in parallel, crack growth, friction at the

crack interfaces and heating due to friction and reactions at the crack surfaces. The implementation has been verified with experimental data and found to be accurate.



3 — Elastic material models

3.1 Hyperelastic Material Models

The subject of modeling the response of materials to deformation is a subject that has filled numerous textbooks. Therefore, rather than attempt to condense these volumes, here the reader will be simply be given a simple material response model. Other more complex material response models can be interchanged in the framework discussed above quite readily.

The author has come to prefer a class of models known as hyperelastic models. What this means is that the stress response of these materials is derived from a strain energy function. A strain energy function gives a relationship between the state of deformation that a material is in, and the amount of stored strain energy that this material has. This is akin to the familiar relationship for the stored energy in a spring, $W = \frac{1}{2}kdx^2$ where k is the spring constant, and dx is the distance that the spring has been compressed or extended.

One such strain energy function is given by:

$$W = \frac{\lambda}{4}(J^2 - 1) - (\frac{\lambda}{2} + \mu)\ln J + \frac{\mu}{2}\mathrm{tr}(\mathbf{F}^T\mathbf{F} - 3) \tag{3.1}$$

from which the following relationship for the stress can be derived:

$$\boldsymbol{\sigma} = \frac{\lambda}{2}(J - \frac{1}{J})\mathbf{I} + \mu(\mathbf{F}\mathbf{F}^T) - \mathbf{I}) \tag{3.2}$$

where λ and μ are material constants, while J and $\mathbf{F}$ describe the state of deformation. These will be defined shortly.

In the Algorithm section, the calculation of the velocity gradient, $\nabla\mathbf{v}_p$ is given in Equation 1.64. Starting from there, we can then compute an increment in the deformation gradient, $\mathbf{F}(dt)$ by:

$$\mathbf{F}(dt) = \nabla\mathbf{v}_p dt + \mathbf{I}. \tag{3.3}$$

This increment in the deformation gradient can then be used to compute a new total deformation gradient using:

$$\mathbf{F}(t + dt) = \mathbf{F}(dt)\mathbf{F}(t). \tag{3.4}$$

Note that the initial (t=0) deformation gradient is simply the identity, i.e. $\mathbf{F}(0) = \mathbf{I}$. Now with the deformation gradient, one can compute J by:

$$J = \det(\mathbf{F}(t + dt)). \tag{3.5}$$

Note that J represents the volumetric part of the deformation. Specifically, it is the ratio of the current volume of an element of material to it's original volume. Similarly, we can define an increment in J as:

$$J_{inc} = \det(\mathbf{F}(dt)) \tag{3.6}$$

which is the ratio of the current volume of an element of material to it's volume at the previous timestep. Thus we can write:

$$v_p(t + dt) = J_{inc} v_p(t). \tag{3.7}$$

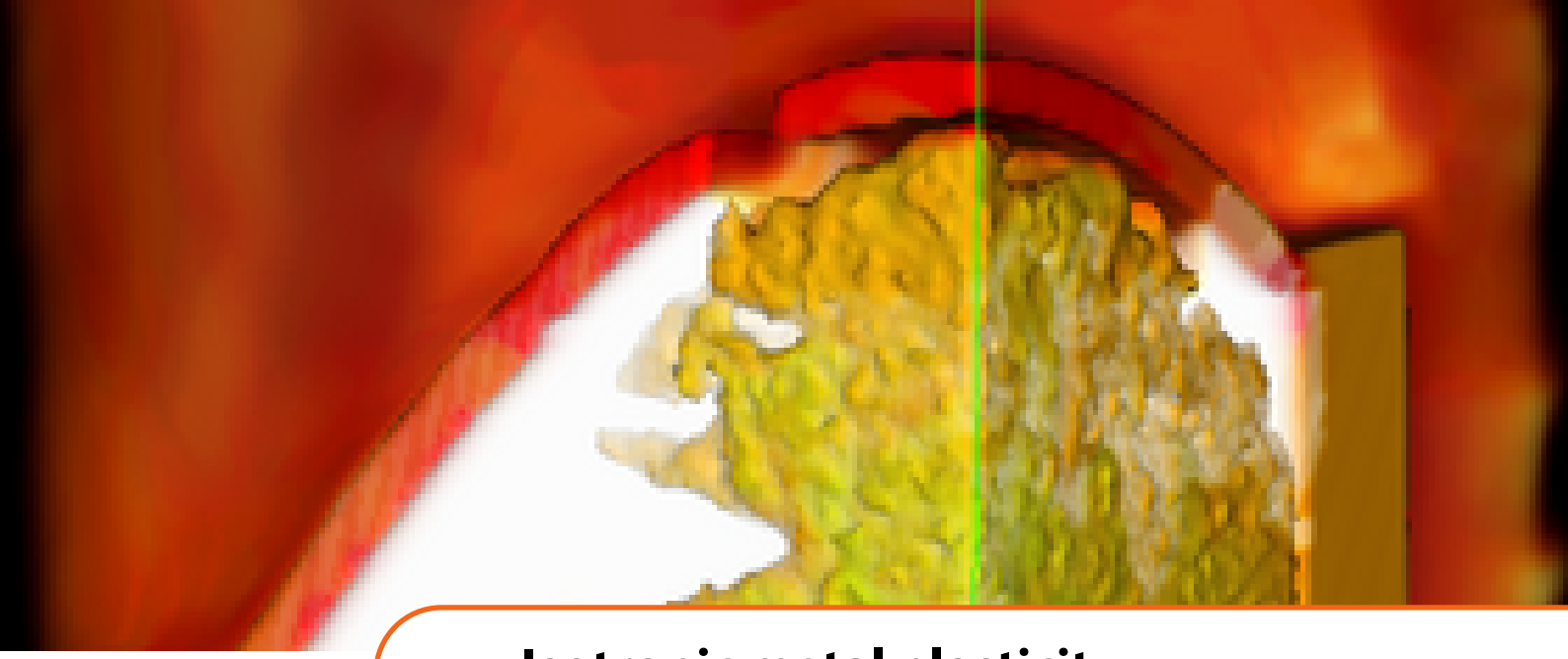
Elastic material properties are frequently given in terms of bulk and shear moduli, or κ and μ. The shear is sometimes denoted by G. The shear modulus μ appears in Equation 3.2 above. λ can be computed from κ and μ by:

$$\lambda = \kappa - \frac{2}{3}\mu. \tag{3.8}$$

Lastly, based on material properties λ and μ, a material wavespeed can be computed:

$$c^2 = (\lambda + 3\mu)\frac{m_p}{v_p}. \tag{3.9}$$

This wavespeed can be used in computing the timestep size as described above.



4 — Isotropic metal plasticity

4.1 Introduction

This document deals with some features of Vaango and some material models for solids that have been implemented in the Vaango Computational Framework (for use with the Material Point Method). The approach taken has been to separate the stress-strain relations from the numerical stress update algorithms as far as possible. A moderate rotation/small strain hypoelastic-plastic stress update algorithm is discussed and the manner in which plasticity flow rules, damage models and equations of state fit into the stress update algorithms are shown.

4.2 Stress Update Algorithms

The hypoelastic-plastic stress update is based on an additive decomposition of the rate of deformation tensor into elastic and plastic parts while the hyperelastic-plastic stress update is based on a multiplicative decomposition of the elastic and plastic deformation gradients. Incompressibility is assumed for plastic deformations. The volumetric response is therefore determined either by a bulk modulus and the trace of the rate of deformation tensor or using an equation of state. The deviatoric response is determined either by an elastic constitutive equation or using a plastic flow rule in combination with a yield condition.

The material models that can be varied in these stress update approaches are (not all are applicable to both hypo- and hyperelastic formulations nor is the list exhaustive):

1. The elasticity model, for example,
   - Isotropic linear elastic model.
   - Anisotropic linear elastic models.
   - Isotropic nonlinear elastic models.
   - Anisotropic nonlinear elastic models.
2. Isotropic hardening or Kinematic hardening using a back stress evolution rule, for example,
   - Ziegler evolution rule .
3. The flow rule and hardening/softening law, for example,
   - Perfect plasticity/power law hardening plasticity.
   - Johnson-Cook plasticity .
   - Mechanical Threshold Stress (MTS) plasticity .
   - Anand plasticity .
4. The yield condition, for example,

- von Mises yield condition.
- Drucker-Prager yield condition.
- Mohr-Coulomb yield condition.

5. A continuum or nonlocal damage model with damage evolution given by, for example,
   - Johnson-Cook damage model.
   - Gurson-Needleman-Tvergaard model.
   - Sandia damage model.
6. An equation of state to determine the pressure (or volumetric response), for example,
   - Mie-Gruneisen equation of state.

The currently implemented stress update algorithms in the Vaango Computational Framework do not allow for arbitrary elasticity models, kinematic hardening, arbitrary yield conditions and continuum or nonlocal damage (however the a damage parameter is updated and used in the erosion algorithm). The models that can be varied are the flow rule models, damage variable evolution models and the equation of state models.

Note that there are no checks in the Unitah Computational Framework to prevent users from mixing and matching inappropriate models.

This section describes the current implementation of the hypoelastic- plastic model. The stress update algorithm is a slightly modified version of the approach taken by Nemat-Nasser et al. (1991,1992) [16, 17], Wang (1994) [18], Maudlin (1996) [19], and Zocher et al. (2000) [20].

4.2.1 Simplified theory for hypoelastic-plasticity

A simplified version of the theory behind the stress update algorithm (in the contex of von Mises plasticity) is given below.

Following [19], the rotated spatial rate of deformation tensor ($\mathbf{d}$) is decomposed into an elastic part ($\mathbf{d}^e$) and a plastic part ($\mathbf{d}^p$)

$$\mathbf{d} = \mathbf{d}^e + \mathbf{d}^p \tag{4.1}$$

If we assume plastic incompressibility ($\mathrm{tr}(\mathbf{d}^p) = 0$), we get

$$\boldsymbol{\eta} = \boldsymbol{\eta}^e + \boldsymbol{\eta}^p \tag{4.2}$$

where $\boldsymbol{\eta}$, $\boldsymbol{\eta}^e$, and $\boldsymbol{\eta}^p$ are the deviatoric parts of $\mathbf{d}$, $\mathbf{d}^e$, and $\mathbf{d}^p$, respectively. For isotropic materials, the hypoelastic constitutive equation for deviatoric stress is

$$\dot{\mathbf{s}} = 2\mu(\boldsymbol{\eta} - \boldsymbol{\eta}^p) \tag{4.3}$$

where $\mathbf{s}$ is the deviatoric part of the stress tensor and μ is the shear modulus. We assume that the flow stress obeys the Huber-von Mises yield condition

$$f := \sqrt{\frac{3}{2}}\,\|\mathbf{s}\| - \sigma_y \le 0 \quad \text{or,} \quad F := \frac{3}{2}\mathbf{s} : \mathbf{s} - \sigma_y^2 \le 0 \tag{4.4}$$

where σ_y is the flow stress. Assuming an associated flow rule, and noting that $\mathbf{d}^p = \boldsymbol{\eta}^p$, we have

$$\boldsymbol{\eta}^p = \mathbf{d}^p = \lambda\frac{\partial f}{\partial \boldsymbol{\sigma}} = \Lambda\frac{\partial F}{\partial \boldsymbol{\sigma}} = 3\Lambda\mathbf{s} \tag{4.5}$$

where $\boldsymbol{\sigma}$ is the stress. Let $\mathbf{u}$ be a tensor proportional to the plastic straining direction, and define γ as

$$\mathbf{u} = \sqrt{3}\frac{\mathbf{s}}{\|\mathbf{s}\|}; \quad \gamma := \sqrt{3}\Lambda\,\|\mathbf{s}\| \quad \Longrightarrow \gamma\mathbf{u} = 3\Lambda\mathbf{s} \tag{4.6}$$

Therefore, we have

$$\boldsymbol{\eta}^p = \gamma \mathbf{u}; \quad \dot{\mathbf{s}} = 2\mu(\boldsymbol{\eta} - \gamma \mathbf{u}) \tag{4.7}$$

From the consistency condition, if we assume that the deviatoric stress remains constant over a timestep, we get

$$\gamma = \frac{\mathbf{s} : \boldsymbol{\eta}}{\mathbf{s} : \mathbf{u}} \tag{4.8}$$

which provides an initial estimate of the plastic strain-rate. To obtain a semi-implicit update of the stress using equation (4.7), we define

$$\tau^2 := \frac{3}{2}\mathbf{s} : \mathbf{s} = \sigma_y^2 \tag{4.9}$$

Taking a time derivative of equation (4.9) gives us

$$\sqrt{2}\dot{\tau} = \sqrt{3}\frac{\mathbf{s} : \dot{\mathbf{s}}}{\|\mathbf{s}\|} \tag{4.10}$$

Plugging equation (4.10) into equation $(4.7)_2$ we get

$$\dot{\tau} = \sqrt{2}\mu(\mathbf{u} : \boldsymbol{\eta} - \gamma \mathbf{u} : \mathbf{u}) = \sqrt{2}\mu(d - 3\gamma) \tag{4.11}$$

where $d = \mathbf{u} : \boldsymbol{\eta}$. If the initial estimate of the plastic strain-rate is that all of the deviatoric strain-rate is plastic, then we get an approximation to γ, and the corresponding error (γ_{er}) given by

$$\gamma_{\text{approx}} = \frac{d}{3}; \quad \gamma_{\text{er}} = \gamma_{\text{approx}} - \gamma = \frac{d}{3} - \gamma \tag{4.12}$$

The incremental form of the above equation is

$$\Delta\gamma = \frac{d^* \Delta t}{3} - \Delta\gamma_{\text{er}} \tag{4.13}$$

Integrating equation (4.11) from time t_n to time $t_{n+1} = t_n + \Delta t$, and using equation (4.13) we get

$$\tau_{n+1} = \tau_n + \sqrt{2}\mu(d^* \Delta t - 3\Delta\gamma) = \tau_n + 3\sqrt{2}\mu\Delta\gamma_{\text{er}} \tag{4.14}$$

where d^* is the average value of d over the timestep. Solving for $\Delta\gamma_{\text{er}}$ gives

$$\Delta\gamma_{\text{er}} = \frac{\tau_{n+1} - \tau_n}{3\sqrt{2}\mu} = \frac{\sqrt{2}\sigma_y - \sqrt{3}\,\|\mathbf{s}_n\|}{6\mu} \tag{4.15}$$

The direction of the total strain-rate ($\mathbf{u}^\eta$) and the direction of the plastic strain-rate ($\mathbf{u}^s$) are given by

$$\mathbf{u}^\eta = \frac{\boldsymbol{\eta}}{\|\boldsymbol{\eta}\|}; \quad \mathbf{u}^s = \frac{\mathbf{s}}{\|\mathbf{s}\|} \tag{4.16}$$

Let θ be the fraction of the time increment that sees elastic straining. Then

$$\theta = \frac{d^* - 3\gamma_n}{d^*} \tag{4.17}$$

where $\gamma_n = d_n/3$ is the value of γ at the beginning of the timestep. We also assume that

$$d^* = \sqrt{3}\boldsymbol{\eta} : \left[(1-\theta)\mathbf{u}^\eta + \frac{\theta}{2}(\mathbf{u}^\eta + \mathbf{u}^s)\right] \tag{4.18}$$

Plugging equation (4.17) into equation (4.18) we get a quadratic equation that can be solved for d^* as follows

$$\frac{2}{\sqrt{3}}(d^*)^2 - (\boldsymbol{\eta} : \mathbf{u}^s + \|\boldsymbol{\eta}\|)d^* + 3\gamma_n(\boldsymbol{\eta} : \mathbf{u}^s - \|\boldsymbol{\eta}\|) = 0 \tag{4.19}$$

The real positive root of the above quadratic equation is taken as the estimate for d. The value of $\Delta\gamma$ can now be calculated using equations (4.13) and (4.15). A semi-implicit estimate of the deviatoric stress can be obtained at this stage by integrating equation $(4.7)_2$

$$\tilde{\mathbf{s}}_{n+1} = \mathbf{s}_n + 2\mu\left(\eta\Delta t - \sqrt{3}\Delta\gamma\frac{\tilde{\mathbf{s}}_{n+1}}{\|\mathbf{s}_{n+1}\|}\right) \tag{4.20}$$

$$= \mathbf{s}_n + 2\mu\left(\eta\Delta t - \frac{3}{\sqrt{2}}\Delta\gamma\frac{\tilde{\mathbf{s}}_{n+1}}{\sigma_y}\right) \tag{4.21}$$

Solving for $\tilde{\mathbf{s}}_{n+1}$, we get

$$\tilde{\mathbf{s}}_{n+1} = \frac{\mathbf{s}^{\text{trial}}_{n+1}}{1 + 3\sqrt{2}\mu\dfrac{\Delta\gamma}{\sigma_y}} \tag{4.22}$$

where $\mathbf{s}^{\text{trial}}_{n+1} = \mathbf{s}_n + 2\mu\Delta t\boldsymbol{\eta}$. A final radial return adjustment is used to move the stress to the yield surface

$$\mathbf{s}_{n+1} = \sqrt{\frac{2}{3}}\sigma_y\frac{\tilde{\mathbf{s}}_{n+1}}{\|\tilde{\mathbf{s}}_{n+1}\|} \tag{4.23}$$

A pathological situation arises if $\gamma_n = \mathbf{u}_n : \boldsymbol{\eta}_n$ is less than or equal to zero or $\Delta\gamma_{\text{er}} \geq \frac{d^*}{3}\Delta t$. This can occur is the rate of plastic deformation is small compared to the rate of elastic deformation or if the timestep size is too small (see [17]). In such situations, we use a locally implicit stress update that uses Newton iterations (as discussed in [21], page 124) to compute $\tilde{\mathbf{s}}$.

4.3 Models

Below are some of the strain-rate, strain, and temperature dependent models for metals that are implemented in Vaango.

4.3.1 Equation of State Models

The elastic-plastic stress update assumes that the volumetric part of the Cauchy stress can be calculated using an equation of state. There are three equations of state that are implemented in Vaango. These are

1. A default hypoelastic equation of state.
2. A neo-Hookean equation of state.
3. A Mie-Gruneisen type equation of state.

Default hypoelastic equation of state

In this case we assume that the stress rate is given by

$$\dot{\boldsymbol{\sigma}} = \lambda\,\text{tr}(\mathbf{d}^e)\,\mathbf{1} + 2\,\mu\,\mathbf{d}^e \tag{4.24}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress, $\mathbf{d}^e$ is the elastic part of the rate of deformation, and λ, μ are constants.

If $\boldsymbol{\eta}^e$ is the deviatoric part of $\mathbf{d}^e$ then we can write

$$\dot{\boldsymbol{\sigma}} = \left(\lambda + \frac{2}{3}\,\mu\right)\text{tr}(\mathbf{d}^e)\,\mathbf{1} + 2\,\mu\,\boldsymbol{\eta}^e = \kappa\,\text{tr}(\mathbf{d}^e)\,\mathbf{1} + 2\,\mu\,\boldsymbol{\eta}^e\,. \tag{4.25}$$

If we split $\boldsymbol{\sigma}$ into a volumetric and a deviatoric part, i.e., $\boldsymbol{\sigma} = p\,\mathbf{1} + \mathbf{s}$ and take the time derivative to get $\dot{\boldsymbol{\sigma}} = \dot{p}\,\mathbf{1} + \dot{\mathbf{s}}$ then

$$\dot{p} = \kappa\,\mathrm{tr}(\mathbf{d}^e)\,. \tag{4.26}$$

In addition we assume that $\mathbf{d} = \mathbf{d}^e + \mathbf{d}^p$. If we also assume that the plastic volume change is negligible, we can then write that

$$\dot{p} = \kappa\,\mathrm{tr}(\mathbf{d})\,. \tag{4.27}$$

This is the equation that is used to calculate the pressure p in the default hypoelastic equation of state, i.e.,

$$\boxed{p_{n+1} = p_n + \kappa\,\mathrm{tr}(\mathbf{d}_{n+1})\,\Delta t\,.} \tag{4.28}$$

To get the derivative of p with respect to J, where $J = \det(\boldsymbol{F})$, we note that

$$\dot{p} = \frac{\partial p}{\partial J}\,\dot{J} = \frac{\partial p}{\partial J}\,J\,\mathrm{tr}(\mathbf{d})\,. \tag{4.29}$$

Therefore,

$$\boxed{\frac{\partial p}{\partial J} = \frac{\kappa}{J}\,.} \tag{4.30}$$

This model is invoked in Vaango using

```
<equation_of_state type="default_hypo">
</equation_of_state>
```

The code is in `.../MPM/ConstitutiveModel/PlasticityModels/DefaultHypoElasticEOS.cc`.

Default hyperelastic equation of state

In this model the pressure is computed using the relation

$$p = \tfrac{1}{2}\,\kappa\left(J^e - \frac{1}{J^e}\right) \tag{4.31}$$

where κ is the bulk modulus and J^e is determinant of the elastic part of the deformation gradient.

We can also compute

$$\frac{dp}{dJ} = \tfrac{1}{2}\,\kappa\left(1 + \frac{1}{(J^e)^2}\right)\,. \tag{4.32}$$

This model is invoked in Vaango using

```
<equation_of_state type="default_hyper">
</equation_of_state>
```

The code is in `.../MPM/ConstitutiveModel/PlasticityModels/HyperElasticEOS.cc`. If an EOS is not specified then this model is the **default**.

Mie-Gruneisen equation of state

The pressure (p) is calculated using a Mie-Grüneisen equation of state of the form ([20, 22])

$$\boxed{p_{n+1} = -\frac{\rho_0 C_0^2 (1 - J^e_{n+1})[1 - \Gamma_0(1 - J^e_{n+1})/2]}{[1 - S_\alpha(1 - J^e_{n+1})]^2} - \Gamma_0 e_{n+1}\,; \quad J^e := \det \boldsymbol{F}^e} \tag{4.33}$$

where C_0 is the bulk speed of sound, ρ_0 is the initial mass density, Γ_0 is the Grüneisen's gamma at the reference state, $S_\alpha = dU_s/dU_p$ is a linear Hugoniot slope coefficient, U_s is the shock wave velocity, U_p is the particle velocity, and e is the internal energy density (per unit reference volume), $\boldsymbol{F}^e$ is the elastic part of the deformation gradient. For isochoric plasticity,

$$J^e = J = \det(\boldsymbol{F}) = \frac{\rho_0}{\rho}\,.$$

The internal energy is computed using

$$E = \frac{1}{V_0}\int C_v dT \approx \frac{C_v(T - T_0)}{V_0} \tag{4.34}$$

where $V_0 = 1/\rho_0$ is the reference specific volume at temperature $T = T_0$, and C_v is the specific heat at constant volume.

Also,

$$\boxed{\frac{\partial p}{\partial J^e} = \frac{\rho_0 C_0^2 \left[1 + (S_\alpha - \Gamma_0)\,(1 - J^e)\right]}{\left[1 - S_\alpha\,(1 - J^e)\right]^3} - \Gamma_0 \frac{\partial e}{\partial J^e}.} \tag{4.35}$$

We neglect the $\frac{\partial e}{\partial J^e}$ term in our calculations.

This model is invoked in Vaango using

```
<equation_of_state type="mie_gruneisen">
  <C_0>5386</C_0>
  <Gamma_0>1.99</Gamma_0>
  <S_alpha>1.339</S_alpha>
</equation_of_state>
```

The code is in `.../MPM/ConstitutiveModel/PlasticityModels/MieGruneisenEOS.cc`.

4.3.2 Melting Temperature

Default model

The default model is to use a constant melting temperature. This model is invoked using

```
<melting_temp_model type="constant_Tm">
</melting_temp_model>
```

SCG melt model

We use a pressure dependent relation to determine the melting temperature (T_m). The Steinberg-Cochran-Guinan (SCG) melt model ([23]) has been used for our simulations of copper. This model is based on a modified Lindemann law and has the form

$$T_m(\rho) = T_{m0} \exp\left[2a\left(1 - \frac{1}{\eta}\right)\right]\eta^{2(\Gamma_0 - a - 1/3)}; \quad \eta = \frac{\rho}{\rho_0} \tag{4.36}$$

where T_{m0} is the melt temperature at $\eta = 1$, a is the coefficient of the first order volume correction to Grüneisen's gamma (Γ_0).

This model is invoked with

```
<melting_temp_model type="scg_Tm">
  <T_m0> 2310.0 </T_m0>
  <Gamma_0> 3.0 </Gamma_0>
  <a> 1.67 </a>
</melting_temp_model>
```

BPS melt model

An alternative melting relation that is based on dislocation-mediated phase transitions - the Burakovsky-Preston-Silbar (BPS) model ([24]) can also be used. This model has been used to determine the melt temperature for 4340 steel. The BPS model has the form

$$T_m(p) = T_m(0)\left[\frac{1}{\eta} + \frac{1}{\eta^{4/3}}\frac{\mu_0'}{\mu_0}p\right]; \quad \eta = \left(1 + \frac{K_0'}{K_0}p\right)^{1/K_0'} \tag{4.37}$$

$$T_m(0) = \frac{\kappa\lambda\mu_0 v_{WS}}{8\pi\ln(z-1)\,k_b}\ln\left(\frac{\alpha^2}{4\,b^2\rho_c(T_m)}\right) \tag{4.38}$$

where p is the pressure, $\eta = \rho/\rho_0$ is the compression, μ_0 is the shear modulus at room temperature and zero pressure, $\mu_0' = \partial\mu/\partial p$ is the derivative of the shear modulus at zero pressure, K_0 is the bulk modulus at room temperature and zero pressure, $K_0' = \partial K/\partial p$ is the derivative of the bulk modulus at zero pressure, κ is a constant, $\lambda = b^3/v_{WS}$ where b is the magnitude of the Burgers' vector, v_{WS} is the Wigner-Seitz volume, z is the coordination number, α is a constant, $\rho_c(T_m)$ is the critical density of dislocations, and k_b is the Boltzmann constant.

This model is invoked with

```
<melting_temp_model type="bps_Tm">
  <B0> 137e9 </B0>
  <dB_dp0> 5.48 <dB_dp0>
  <G0> 47.7e9 <G0>
  <dG_dp0> 1.4 <dG_dp0>
  <kappa> 1.25 <kappa>
  <z> 12 <z>
  <b2rhoTm> 0.64 <b2rhoTm>
  <alpha> 2.9 <alpha>
  <lambda> 1.41 <lambda>
  <a> 3.6147e-9<a>
  <v_ws_a3_factor> 1/4 <v_ws_a3_factor>
  <Boltzmann_Constant> <Boltzmann_Constant>
</melting_temp_model>
```

4.3.3 Shear Modulus

Three models for the shear modulus (μ) have been tested in our simulations. The first has been associated with the Mechanical Threshold Stress (MTS) model and we call it the MTS shear model. The second is the model used by Steinberg-Cochran-Guinan and we call it the SCG shear model while the third is a model developed by Nadal and Le Poac that we call the NP shear model.

Default model

The default model gives a constant shear modulus. The model is invoked using

```
<shear_modulus_model type="constant_shear">
</shear_modulus_model>
```

MTS Shear Modulus Model

The simplest model is of the form suggested by [25] ([26])

$$\mu(T) = \mu_o - \frac{D}{exp(T_o/T) - 1} \tag{4.39}$$

where μ_o is the shear modulus at oK, and D, T_o are material constants.

The model is invoked using

```
<shear_modulus_model type="mts_shear">
  <mu_0>28.0e9</mu_0>
  <D>4.50e9</D>
  <T_0>294</T_0>
</shear_modulus_model>
```

SCG Shear Modulus Model

The Steinberg-Cochran-Guinan (SCG) shear modulus model ([20, 23]) is pressure dependent and has the form

$$\mu(p, T) = \mu_o + \frac{\partial \mu}{\partial p}\frac{p}{\eta^{1/3}} + \frac{\partial \mu}{\partial T}(T - 300); \quad \eta = \rho/\rho_o \tag{4.40}$$

where, μ_o is the shear modulus at the reference state(T = 300 K, p = 0, η = 1), p is the pressure, and T is the temperature. When the temperature is above T_m, the shear modulus is instantaneously set to zero in this model.

The model is invoked using

```
<shear_modulus_model type="scg_shear">
  <mu_0> 81.8e9 </mu_0>
  <A> 20.6e-12 </A>
  <B> 0.16e-3 </B>
</shear_modulus_model>
```

NP Shear Modulus Model

A modified version of the SCG model has been developed by [27] that attempts to capture the sudden drop in the shear modulus close to the melting temperature in a smooth manner. The Nadal-LePoac (NP) shear modulus model has the form

$$\mu(p, T) = \frac{1}{\mathcal{J}(\hat{T})}\left[\left(\mu_o + \frac{\partial \mu}{\partial p}\frac{p}{\eta^{1/3}}\right)(1 - \hat{T}) + \frac{\rho}{Cm}\, k_b\, T\right]; \quad C := \frac{(6\pi^2)^{2/3}}{3} f^2 \tag{4.41}$$

where

$$\mathcal{J}(\hat{T}) := 1 + \exp\left[-\frac{1 + 1/\zeta}{1 + \zeta/(1 - \hat{T})}\right] \quad \text{for} \quad \hat{T} := \frac{T}{T_m} \in [0, 1 + \zeta], \tag{4.42}$$

μ_o is the shear modulus at 0 K and ambient pressure, ζ is a material parameter, k_b is the Boltzmann constant, m is the atomic mass, and f is the Lindemann constant.

The model is invoked using

```
<shear_modulus_model type="np_shear">
  <mu_0>26.5e9</mu_0>
  <zeta>0.04</zeta>
  <slope_mu_p_over_mu0>65.0e-12</slope_mu_p_over_mu0>
  <C> 0.047 </C>
  <m> 26.98 </m>
</shear_modulus_model>
```

PTW Shear model

The PTW shear model is a simplified version of the SCG shear model. The inputs can be found in `.../MPM/ConstitutiveModel/PlasticityModel/PTWShear.h`.

4.3.4 Flow Stress

We have explored five temperature and strain rate dependent models that can be used to compute the flow stress:

1. the Johnson-Cook (JC) model
2. the Steinberg-Cochran-Guinan-Lund (SCG) model.
3. the Zerilli-Armstrong (ZA) model.
4. the Mechanical Threshold Stress (MTS) model.
5. the Preston-Tonks-Wallace (PTW) model.

JC Flow Stress Model

The Johnson-Cook (JC) model ([28]) is purely empirical and gives the following relation for the flow stress (σ_y)

$$\sigma_y(\varepsilon_p, \dot{\varepsilon}_p, T) = \left[A + B(\varepsilon_p)^n\right]\left[1 + C\ln(\dot{\varepsilon}_p^*)\right]\left[1 - (T^*)^m\right] \tag{4.43}$$

where ε_p is the equivalent plastic strain, $\dot{\varepsilon}_p$ is the plastic strain rate, A, B, C, n, m are material constants,

$$\dot{\varepsilon}_p^* = \frac{\dot{\varepsilon}_p}{\dot{\varepsilon}_{p0}}; \quad T^* = \frac{(T - T_0)}{(T_m - T_0)}, \tag{4.44}$$

$\dot{\varepsilon}_{p0}$ is a user defined plastic strain rate, T_0 is a reference temperature, and T_m is the melt temperature. For conditions where $T^* < 0$, we assume that $m = 1$.

The inputs for this model are

```
<plasticity_model type="johnson_cook">
  <A>792.0e6</A>
  <B>510.0e6</B>
  <C>0.014</C>
  <n>0.26</n>
  <m>1.03</m>
  <T_r>298.0</T_r>
  <T_m>1793.0</T_m>
  <epdot_0>1.0</epdot_0>
</plasticity_model>
```

SCG Flow Stress Model

The Steinberg-Cochran-Guinan-Lund (SCG) model is a semi-empirical model that was developed by [23] for high strain rate situations and extended to low strain rates and bcc materials by [29]. The flow stress in this model is given by

$$\sigma_y(\varepsilon_p, \dot{\varepsilon}_p, T) = \left[\sigma_a f(\varepsilon_p) + \sigma_t(\dot{\varepsilon}_p, T)\right]\frac{\mu(p, T)}{\mu_0} \tag{4.45}$$

where σ_a is the athermal component of the flow stress, $f(\varepsilon_p)$ is a function that represents strain hardening, σ_t is the thermally activated component of the flow stress, $\mu(p, T)$ is the shear modulus, and μ_0 is the shear modulus at standard temperature and pressure. The strain hardening function has the form

$$f(\varepsilon_p) = \left[1 + \beta(\varepsilon_p + \varepsilon_{pi})\right]^n; \quad \sigma_a f(\varepsilon_p) \le \sigma_{\max} \tag{4.46}$$

where β, n are work hardening parameters, and ε_{pi} is the initial equivalent plastic strain. The thermal component σ_t is computed using a bisection algorithm from the following equation (based on the work of [30])

$$\dot{\varepsilon}_p = \left[\frac{1}{C_1}\exp\left[\frac{2U_k}{k_b\,T}\left(1-\frac{\sigma_t}{\sigma_p}\right)^2\right]+\frac{C_2}{\sigma_t}\right]^{-1}; \quad \sigma_t \le \sigma_p \tag{4.47}$$

where $2U_k$ is the energy to form a kink-pair in a dislocation segment of length L_d, k_b is the Boltzmann constant, σ_p is the Peierls stress. The constants C_1, C_2 are given by the relations

$$C_1 := \frac{\rho_d L_d a b^2 \nu}{2w^2}; \quad C_2 := \frac{D}{\rho_d b^2} \tag{4.48}$$

where ρ_d is the dislocation density, L_d is the length of a dislocation segment, a is the distance between Peierls valleys, b is the magnitude of the Burgers' vector, ν is the Debye frequency, w is the width of a kink loop, and D is the drag coefficient.

The inputs for this model are of the form

```
<plasticity_model type="steinberg_cochran_guinan">
  <mu_0> 81.8e9 </mu_0>
  <sigma_0> 1.15e9 </sigma_0>
  <Y_max> 0.25e9 </Y_max>
  <beta> 2.0 </beta>
  <n> 0.50 </n>
  <A> 20.6e-12 </A>
  <B> 0.16e-3 </B>
  <T_m0> 2310.0 </T_m0>
  <Gamma_0> 3.0 </Gamma_0>
  <a> 1.67 </a>
  <epsilon_p0> 0.0 </epsilon_p0>
</plasticity_model>
```

ZA Flow Stress Model

The Zerilli-Armstrong (ZA) model ([31–33]) is based on simplified dislocation mechanics. The general form of the equation for the flow stress is

$$\sigma_y(\varepsilon_p, \dot{\varepsilon}_p, T) = \sigma_a + B\exp(-\beta(\dot{\varepsilon}_p)T) + B_0\sqrt{\varepsilon_p}\exp(-\alpha(\dot{\varepsilon}_p)T) \tag{4.49}$$

where σ_a is the athermal component of the flow stress given by

$$\sigma_a := \sigma_g + \frac{k_h}{\sqrt{l}} + K\varepsilon_p^n, \tag{4.50}$$

σ_g is the contribution due to solutes and initial dislocation density, k_h is the microstructural stress intensity, l is the average grain diameter, K is zero for fcc materials, B, B_0 are material constants. The functional forms of the exponents α and β are

$$\alpha = \alpha_0 - \alpha_1\ln(\dot{\varepsilon}_p); \quad \beta = \beta_0 - \beta_1\ln(\dot{\varepsilon}_p); \tag{4.51}$$

where $\alpha_0, \alpha_1, \beta_0, \beta_1$ are material parameters that depend on the type of material (fcc, bcc, hcp, alloys). The Zerilli-Armstrong model has been modified by [34] for better performance at high temperatures. However, we have not used the modified equations in our computations.

The inputs for this model are of the form

```
<bcc_or_fcc> fcc </bcc_or_fcc>
<c2> </c2>
<c3> </c3>
<c4> </c4>
<n> </n>
```

MTS Flow Stress Model

The Mechanical Threshold Stress (MTS) model ([35–37]) gives the following form for the flow stress

$$\sigma_y(\varepsilon_p, \dot{\varepsilon}_p, T) = \sigma_a + (S_i\sigma_i + S_e\sigma_e)\frac{\mu(p,T)}{\mu_0} \tag{4.52}$$

where σ_a is the athermal component of mechanical threshold stress, μ_0 is the shear modulus at 0 K and ambient pressure, σ_i is the component of the flow stress due to intrinsic barriers to thermally activated dislocation motion and dislocation-dislocation interactions, σ_e is the component of the flow stress due to microstructural evolution with increasing deformation (strain hardening), (S_i, S_e) are temperature and strain rate dependent scaling factors. The scaling factors take the Arrhenius form

$$S_i = \left[1 - \left(\frac{k_b\ T}{g_{0i}b^3\mu(p,T)}\ln\frac{\dot{\varepsilon}_{poi}}{\dot{\varepsilon}_p}\right)^{1/q_i}\right]^{1/p_i} \tag{4.53}$$

$$S_e = \left[1 - \left(\frac{k_b\ T}{g_{0e}b^3\mu(p,T)}\ln\frac{\dot{\varepsilon}_{poe}}{\dot{\varepsilon}_p}\right)^{1/q_e}\right]^{1/p_e} \tag{4.54}$$

where k_b is the Boltzmann constant, b is the magnitude of the Burgers' vector, (g_{0i}, g_{0e}) are normalized activation energies, $(\dot{\varepsilon}_{poi}, \dot{\varepsilon}_{poe})$ are constant reference strain rates, and (q_i, p_i, q_e, p_e) are constants. The strain hardening component of the mechanical threshold stress (σ_e) is given by a modified Voce law

$$\frac{d\sigma_e}{d\varepsilon_p} = \theta(\sigma_e) \tag{4.55}$$

where

$$\theta(\sigma_e) = \theta_0[1 - F(\sigma_e)] + \theta_{IV}F(\sigma_e) \tag{4.56}$$

$$\theta_0 = a_0 + a_1\ln\dot{\varepsilon}_p + a_2\sqrt{\dot{\varepsilon}_p} - a_3T \tag{4.57}$$

$$F(\sigma_e) = \frac{\tanh\left(\alpha\frac{\sigma_e}{\sigma_{es}}\right)}{\tanh(\alpha)} \tag{4.58}$$

$$\ln\left(\frac{\sigma_{es}}{\sigma_{0es}}\right) = \left(\frac{kT}{g_{0es}b^3\mu(p,T)}\right)\ln\left(\frac{\dot{\varepsilon}_p}{\dot{\varepsilon}_{poes}}\right) \tag{4.59}$$

and θ_0 is the hardening due to dislocation accumulation, θ_{IV} is the contribution due to stage-IV hardening, $(a_0, a_1, a_2, a_3, \alpha)$ are constants, σ_{es} is the stress at zero strain hardening rate, σ_{0es} is the saturation threshold stress for deformation at 0 K, g_{0es} is a constant, and $\dot{\varepsilon}_{poes}$ is the maximum strain rate. Note that the maximum strain rate is usually limited to about 10^7/s.

The inputs for this model are of the form

```
<plasticity_model type="mts_model">
  <sigma_a>363.7e6</sigma_a>
  <mu_0>28.0e9</mu_0>
  <D>4.50e9</D>
  <T_0>294</T_0>
  <koverbcubed>0.823e6</koverbcubed>
  <g_0i>0.0</g_0i>
  <g_0e>0.71</g_0e>
  <edot_0i>0.0</edot_0i>
  <edot_0e>2.79e9</edot_0e>
  <p_i>0.0</p_i>
  <q_i>0.0</q_i>
  <p_e>1.0</p_e>
  <q_e>2.0</q_e>
  <sigma_i>0.0</sigma_i>
```

```
    <a_0>211.8e6</a_0>
    <a_1>0.0</a_1>
    <a_2>0.0</a_2>
    <a_3>0.0</a_3>
    <theta_IV>0.0</theta_IV>
    <alpha>2</alpha>
    <edot_es0>3.42e8</edot_es0>
    <g_0es>0.15</g_0es>
    <sigma_es0>1679.3e6</sigma_es0>
  </plasticity_model>
```

PTW Flow Stress Model

The Preston-Tonks-Wallace (PTW) model ([38]) attempts to provide a model for the flow stress for extreme strain rates (up to 10^{11}/s) and temperatures up to melt. The flow stress is given by

$$\sigma_y(\varepsilon_p, \dot{\varepsilon}_p, T) = \begin{cases} 2\left[\tau_s + \alpha \ln\left[1 - \varphi \exp\left(-\beta - \dfrac{\theta \varepsilon_p}{\alpha \varphi}\right)\right]\right]\mu(p,T) & \text{thermal regime} \\ 2\tau_s \mu(p,T) & \text{shock regime} \end{cases} \tag{4.60}$$

with

$$\alpha := \frac{s_0 - \tau_y}{d}; \quad \beta := \frac{\tau_s - \tau_y}{\alpha}; \quad \varphi := \exp(\beta) - 1 \tag{4.61}$$

where τ_s is a normalized work-hardening saturation stress, s_0 is the value of τ_s at 0K, τ_y is a normalized yield stress, θ is the hardening constant in the Voce hardening law, and d is a dimensionless material parameter that modifies the Voce hardening law. The saturation stress and the yield stress are given by

$$\tau_s = \max\left\{s_0 - (s_0 - s_\infty)\operatorname{erf}\left[\kappa \hat{T} \ln\left(\frac{\gamma\dot{\xi}}{\dot{\varepsilon}_p}\right)\right], s_0\left(\frac{\dot{\varepsilon}_p}{\gamma\dot{\xi}}\right)^{s_1}\right\} \tag{4.62}$$

$$\tau_y = \max\left\{y_0 - (y_0 - y_\infty)\operatorname{erf}\left[\kappa \hat{T} \ln\left(\frac{\gamma\dot{\xi}}{\dot{\varepsilon}_p}\right)\right], \min\left\{y_1\left(\frac{\dot{\varepsilon}_p}{\gamma\dot{\xi}}\right)^{y_2}, s_0\left(\frac{\dot{\varepsilon}_p}{\gamma\dot{\xi}}\right)^{s_1}\right\}\right\} \tag{4.63}$$

where s_∞ is the value of τ_s close to the melt temperature, (y_0, y_∞) are the values of τ_y at 0K and close to melt, respectively, (κ, γ) are material constants, $\hat{T} = T/T_m$, (s_1, y_1, y_2) are material parameters for the high strain rate regime, and

$$\dot{\xi} = \frac{1}{2}\left(\frac{4\pi\rho}{3M}\right)^{1/3}\left(\frac{\mu(p,T)}{\rho}\right)^{1/2} \tag{4.64}$$

where ρ is the density, and M is the atomic mass.

The inputs for this model are of the form

```
  <plasticity_model type="preston_tonks_wallace">
    <theta> 0.025 </theta>
    <p> 2.0 </p>
    <s0> 0.0085 </s0>
    <sinf> 0.00055 </sinf>
    <kappa> 0.11 </kappa>
    <gamma> 0.00001 </gamma>
    <y0> 0.0001 </y0>
    <yinf> 0.0001 </yinf>
    <y1> 0.094 </y1>
    <y2> 0.575 </y2>
    <beta> 0.25 </beta>
    <M> 63.54 </M>
    <G0> 518e8 </G0>
    <alpha> 0.20 </alpha>
    <alphap> 0.20 </alphap>
  </plasticity_model>
```

4.3.5 Adiabatic Heating and Specific Heat

A part of the plastic work done is converted into heat and used to update the temperature of a particle. The increase in temperature (ΔT) due to an increment in plastic strain ($\Delta\epsilon_p$) is given by the equation

$$\Delta T = \frac{\chi\sigma_y}{\rho C_p}\Delta\epsilon_p \tag{4.65}$$

where χ is the Taylor-Quinney coefficient, and C_p is the specific heat. The value of the Taylor-Quinney coefficient is taken to be 0.9 in all our simulations (see [39] for more details on the variation of χ with strain and strain rate).

The Taylor-Quinney coefficient is taken as input in the ElasticPlastic model using the tags

```
<taylor_quinney_coeff> 0.9 </taylor_quinney_coeff>
```

Default specific heat model

The default model returns a constant specific heat and is invoked using

```
<specific_heat_model type="constant_Cp">
</specific_heat_model>
```

Specific heat model for copper

The specific heat model for copper is of the form

$$Cp = \begin{cases} A_0\, T^3 - B_0\, T^2 + C_0\, T - D_0 & \text{if } T < T_0 \\ A_1\, T + B_1 & \text{if } T \geq T_0\,. \end{cases} \tag{4.66}$$

The model is invoked using

```
<specific_heat_model type = "copper_Cp"> </specific_heat_model>
```

Specific heat model for steel

A relation for the dependence of C_p upon temperature is used for the steel ([40]).

$$C_p = \begin{cases} A_1 + B_1\, t + C_1\, |t|^{-\alpha} & \text{if } T < T_c \\ A_2 + B_2\, t + C_2\, t^{-\alpha^{'}} & \text{if } T > T_c \end{cases} \tag{4.67}$$

$$t = \frac{T}{T_c} - 1 \tag{4.68}$$

where T_c is the critical temperature at which the phase transformation from the α to the γ phase takes place, and $A_1, A_2, B_1, B_2, \alpha, \alpha^{'}$ are constants.

The model is invoked using

```
<specific_heat_model type = "steel_Cp"> </specific_heat_model>
```

The heat generated at a material point is conducted away at the end of a time step using the transient heat equation. The effect of conduction on material point temperature is negligible (but non-zero) for the high strain-rate problems simulated using Vaango.

4.3.6 Adding new models

In the parallel implementation of the stress update algorithm, sockets have been added to allow for the incorporation of a variety of plasticity, damage, yield, and bifurcation models without requiring any change in the stress update code. The algorithm is shown in Algorithm 4.1. The equation of state, plasticity model, yield condition, damage model, and the stability criterion are all polymorphic objects created using a factory idiom in C++ ([41]).

Addition of a new model requires the following steps (the example below is only for the flow stress model but the same idea applies to other models) :

1. Creation of a new class that encapsulates the plasticity model. The template for this class can be copied from the existing plasticity models. The data that is unique to the new model are specified in the form of
   - A structure containing the constants for the plasticity model.
   - Particle variables that specify the variables that evolve in the plasticity model.
2. The implementation of the plasticity model involves the following steps.
   - Reading the input file for the model constants in the constructor.
   - Adding the variables that evolve in the plasticity model appropriately to the task graph.
   - Adding the appropriate flow stress calculation method.
3. The `PlasticityModelFactory` is then modified so that it recognizes the added plasticity model.

4.3.7 Damage Models and Failure

Only the Johnson-Cook damage evolution rule has been added to the DamageModelFactory so far. The damage model framework is designed to be similar to the plasticity model framework. New models can be added using the approach described in the previous section.

A particle is tagged as "failed" when its temperature is greater than the melting point of the material at the applied pressure. An additional condition for failure is when the porosity of a particle increases beyond a critical limit and the strain exceeds the fracture strain of the material. Another condition for failure is when a material bifurcation condition such as the Drucker stability postulate is satisfied. Upon failure, a particle is either removed from the computation by setting the stress to zero or is converted into a material with a different velocity field which interacts with the remaining particles via contact. Either approach leads to the simulation of a newly created surface. More details of the approach can be found in [42–44].

4.3.8 Yield conditions

When failure is to be simulated we can use the Gurson-Tvergaard-Needleman yield condition instead of the von Mises condition.

The von Mises yield condition

The von Mises yield condition is the default and is invoked using the tags

```
<yield_condition type="vonMises">
</yield_condition>
```

The Gurson-Tvergaard-Needleman (GTN) yield condition

The Gurson-Tvergaard-Needleman (GTN) yield condition [45, 46] depends on porosity. An associated flow rule is used to determine the plastic rate parameter in either case. The GTN yield condition can be written as

$$\Phi = \left(\frac{\sigma_{eq}}{\sigma_f}\right)^2 + 2q_1 f_* \cosh\left(q_2 \frac{Tr(\sigma)}{2\sigma_f}\right) - (1 + q_3 f_*^2) = 0 \tag{4.69}$$

Table 4.1: Stress Update Algorithm

Persistent:Initial moduli, temperature, porosity,
scalar damage, equation of state, plasticity model,
yield condition, stability criterion, damage model
Temporary:Particle state at time t
Output: Particle state at time $t + \Delta t$

For *all the patches in the domain*
Read the particle data and initialize updated data storage
For *all the particles in the patch*
Compute the velocity gradient and the rate of deformation tensor
Compute the deformation gradient and the rotation tensor
Rotate the Cauchy stress and the rate of deformation tensor
to the material configuration
Compute the current shear modulus and melting temperature
Compute the pressure using the equation of state,
update the hydrostatic stress, and
compute the trial deviatoric stress
Compute the flow stress using the plasticity model
Evaluate the yield function
If *particle is elastic*
Update the elastic deviatoric stress from the trial stress
Rotate the stress back to laboratory coordinates
Update the particle state
Else
Compute the elastic-plastic deviatoric stress
Compute updated porosity, scalar damage, and
temperature increase due to plastic work
Compute elastic-plastic tangent modulus and evaluate stability condition
Rotate the stress back to laboratory coordinates
Update the particle state
End If
If *Temperature* > *Melt Temperature* or *Porosity* > *Critical Porosity* or *Unstable*
Tag particle as failed
End If
Convert failed particles into a material with a different velocity field
End For
End For

where q_1, q_2, q_3 are material constants and f_* is the porosity (damage) function given by

$$f* = \begin{cases} f & \text{for } f \le f_c, \\ f_c + k(f - f_c) & \text{for } f > f_c \end{cases} \tag{4.70}$$

where k is a constant and f is the porosity (void volume fraction). The flow stress in the matrix material is computed using either of the two plasticity models discussed earlier. Note that the flow stress in the matrix material also remains on the undamaged matrix yield surface and uses an associated flow rule.

This yield condition is invoked using

```
<yield_condition type="gurson">
  <q1> 1.5 </q1>
  <q2> 1.0 </q2>
  <q3> 2.25 </q3>
  <k> 4.0 </k>
  <f_c> 0.05 </f_c>
</yield_condition>
```

4.3.9 Porosity model

The evolution of porosity is calculated as the sum of the rate of growth and the rate of nucleation [47]. The rate of growth of porosity and the void nucleation rate are given by the following equations [48]

$$\dot{f} = \dot{f}_{\text{nucl}} + \dot{f}_{\text{grow}} \tag{4.71}$$

$$\dot{f}_{\text{grow}} = (1 - f)\text{Tr}(\boldsymbol{D}_p) \tag{4.72}$$

$$\dot{f}_{\text{nucl}} = \frac{f_n}{(s_n\sqrt{2\pi})} \exp\left[-\tfrac{1}{2}\frac{(\epsilon_p - \epsilon_n)^2}{s_n^2}\right]\dot{\epsilon}_p \tag{4.73}$$

where $\boldsymbol{D}_p$ is the rate of plastic deformation tensor, f_n is the volume fraction of void nucleating particles , ϵ_n is the mean of the distribution of nucleation strains, and s_n is the standard deviation of the distribution.

The inputs tags for porosity are of the form

```
<evolve_porosity> true </evolve_porosity>
<initial_mean_porosity>          0.005 </initial_mean_porosity>
<initial_std_porosity>           0.001 </initial_std_porosity>
<critical_porosity>              0.3   </critical_porosity>
<frac_nucleation>                0.1   </frac_nucleation>
<meanstrain_nucleation>          0.3   </meanstrain_nucleation>
<stddevstrain_nucleation>        0.1   </stddevstrain_nucleation>
<initial_porosity_distrib>       gauss </initial_porosity_distrib>
```

4.3.10 Damage model

After the stress state has been determined on the basis of the yield condition and the associated flow rule, a scalar damage state in each material point can be calculated using the Johnson-Cook model [49]. The Johnson-Cook model has an explicit dependence on temperature, plastic strain, ans strain rate.

The damage evolution rule for the Johnson-Cook damage model can be written as

$$\dot{D} = \frac{\dot{\epsilon_p}}{\epsilon_p^f}; \quad \epsilon_p^f = \left[D_1 + D_2 \exp\left(\frac{D_3}{3}\sigma^*\right)\right]\left[1 + D_4 \ln(\dot{\epsilon_p}^*)\right]\left[1 + D_5 T^*\right] \; ; \quad \sigma^* = \frac{\text{Tr}(\boldsymbol{\sigma})}{\sigma_{eq}}; \tag{4.74}$$

where D is the damage variable which has a value of 0 for virgin material and a value of 1 at fracture, ϵ_p^f is the fracture strain, D_1, D_2, D_3, D_4, D_5 are constants, $\boldsymbol{\sigma}$ is the Cauchy stress, and T^* is the scaled temperature as in the Johnson-Cook plasticity model.

The input tags for the damage model are :

```
<damage_model type="johnson_cook">
  <D1>0.05</D1>
  <D2>3.44</D2>
  <D3>-2.12</D3>
  <D4>0.002</D4>
  <D5>0.61</D5>
</damage_model>
```

An initial damage distribution can be created using the following tags

```
<evolve_damage>                 true  </evolve_damage>
<initial_mean_scalar_damage>    0.005  </initial_mean_scalar_damage>
<initial_std_scalar_damage>     0.001 </initial_std_scalar_damage>
<critical_scalar_damage>        1.0   </critical_scalar_damage>
<initial_scalar_damage_distrib> gauss </initial_scalar_damage_distrib>
```

4.3.11 Erosion algorithm

Under normal conditions, the heat generated at a material point is conducted away at the end of a time step using the heat equation. If special adiabatic conditions apply (such as in impact problems), the heat is accumulated at a material point and is not conducted to the surrounding particles. This localized heating can be used to determine whether a material point has melted.

The determination of whether a particle has failed can be made on the basis of either or all of the following conditions:

- The particle temperature exceeds the melting temperature.
- The TEPLA-F fracture condition [50] is satisfied. This condition can be written as

$$(f/f_c)^2 + (\epsilon_p/\epsilon_p^f)^2 = 1 \tag{4.75}$$

  where f is the current porosity, f_c is the maximum allowable porosity, ϵ_p is the current plastic strain, and ϵ_p^f is the plastic strain at fracture.
- An alternative to ad-hoc damage criteria is to use the concept of bifurcation to determine whether a particle has failed or not. Two stability criteria have been explored in this paper - the Drucker stability postulate [51] and the loss of hyperbolicity criterion (using the determinant of the acoustic tensor) [52, 53].

The simplest criterion that can be used is the Drucker stability postulate [51] which states that time rate of change of the rate of work done by a material cannot be negative. Therefore, the material is assumed to become unstable (and a particle fails) when

$$\dot{\boldsymbol{\sigma}} : \boldsymbol{D}^p \leq 0 \tag{4.76}$$

Another stability criterion that is less restrictive is the acoustic tensor criterion which states that the material loses stability if the determinant of the acoustic tensor changes sign [52, 53]. Determination of the acoustic tensor requires a search for a normal vector around the material point and is therefore computationally expensive. A simplification of this criterion is a check which assumes that the direction of instability lies in the plane of the maximum and minimum principal stress [54]. In this approach, we assume that the strain is localized in a band with normal $\mathbf{n}$, and the magnitude of the velocity difference across the band is $\mathbf{g}$. Then the bifurcation condition leads to the relation

$$R_{ij}g_j = 0\,; \quad R_{ij} = M_{ikjl}n_k n_l + M_{ilkj}n_k n_l - \sigma_{ik}n_j n_k \tag{4.77}$$

where M_{ijkl} are the components of the co-rotational tangent modulus tensor and σ_{ij} are the components of the co-rotational stress tensor. If $\det(R_{ij}) \leq 0$, then g_j can be arbitrary and there is a possibility of strain localization. If this condition for loss of hyperbolicity is met, then a particle deforms in an unstable

manner and failure can be assumed to have occurred at that particle. We use a combination of these criteria to simulate failure.

Since the material in the container may unload locally after fracture, the hypoelastic-plastic stress update may not work accurately under certain circumstances. An improvement would be to use a hyperelastic-plastic stress update algorithm. Also, the plasticity models are temperature dependent. Hence there is the issue of severe mesh dependence due to change of the governing equations from hyperbolic to elliptic in the softening regime [55–57]. Viscoplastic stress update models or nonlocal/gradient plasticity models [58, 59] can be used to eliminate some of these effects and are currently under investigation.

The tags used to control the erosion algorithm are in two places. In the `<MPM> </MPM>` section the following flags can be set

```
<erosion algorithm = "ZeroStress"/>
<create_new_particles>           false      </create_new_particles>
<manual_new_material>            false      </manual_new_material>
```

If the erosion algorithm is `"none"` then no particle failure is done.

In the `<constitutive_model type="elastic_plastic">` section, the following flags can be set

```
<evolve_porosity>               true  </evolve_porosity>
<evolve_damage>                 true  </evolve_damage>
<do_melting>                    true  </do_melting>
<useModifiedEOS>                true  </useModifiedEOS>
<check_TEPLA_failure_criterion> true  </check_TEPLA_failure_criterion>
<check_max_stress_failure>      false </check_max_stress_failure>
<critical_stress>              12.0e9 </critical_stress>
```

4.4 Implementation

The elastic response is assumed to be isotropic. The material constants that are taken as input for the elastic response are the bulk and shear modulus. The flow rule is determined from the input and the appropriate plasticity model is created using the `PlasticityModelFactory` class. The damage evolution rule is determined from the input and a damage model is created using the `DamageModelFactory` class. The equation of state that is used to determine the pressure is also determined from the input. The equation of state model is created using the `MPMEquationOfStateFactory` class.

In addition, a damage evolution variable (D) is stored at each time step (this need not be the case and will be transfered to the damage models in the future). The left stretch and rotation are updated incrementally at each time step (instead of performing a polar decomposition) and the rotation tensor is used to rotate the Cauchy stress and rate of deformation to the material coordinates at each time step (instead of using a objective stress rate formulation).

Any evolution variables for the plasticity model, damage model or the equation of state are specified in the class that encapsulates the particular model.

The flow stress is calculated from the plasticity model using a function call of the form

```
double flowStress = d_plasticity->computeFlowStress(tensorEta, tensorS,
                                                    pTemperature[idx],
                                                    delT, d_tol, matl, idx);
```

A number of plasticity models can be evaluated using the inputs in the `computeFlowStress` call. The variable `d_plasticity` is polymorphic and can represent any of the plasticity models that can be created by the plasticity model factory. The plastic evolution variables are updated using a polymorphic function along the lines of `computeFlowStress`.

The equation of state is used to calculate the hydrostatic stress using a function call of the form

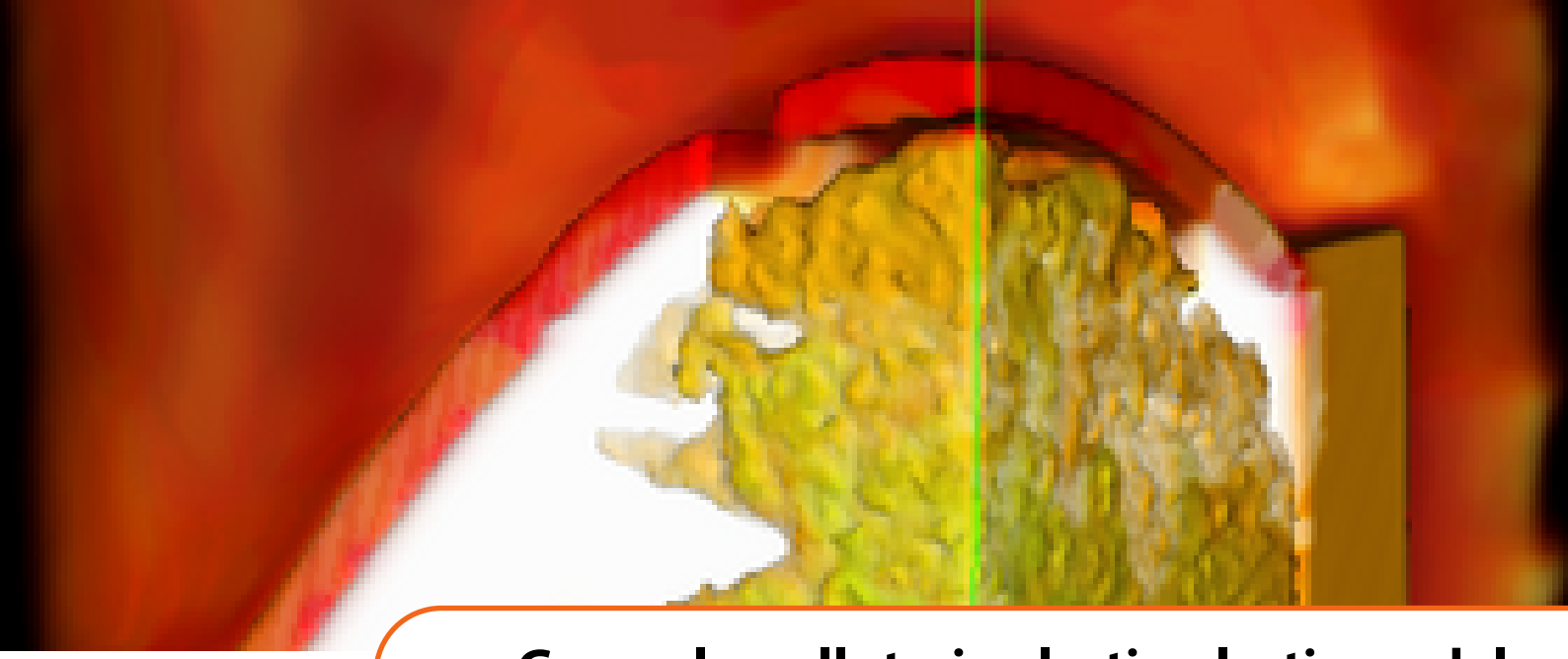
```
Matrix3 tensorHy = d_eos->computePressure(matl, bulk, shear,
```

```
                                                tensorF_new, tensorD,
                                                tensorP, pTemperature[idx],
                                                rho_cur, delT);
```

Similarly, the damage model is called using a function of the type

```
    double damage = d_damage->computeScalarDamage(tensorEta, tensorS,
                                                  pTemperature[idx],
                                                  delT, matl, d_tol,
                                                  pDamage[idx]);
```

Therefore, the plasticity, damage and equation of state models are easily be inserted into any other type of stress update algorithm without any change being needed in them as can be seen in the hyperelastic-plastic stress update algorithm discussed below.



5 — General small strain elastic-plastic model

5.1 Preamble

Let $\boldsymbol{F}$ be the deformation gradient, $\boldsymbol{\sigma}$ be the Cauchy stress, and $\boldsymbol{d}$ be the rate of deformation tensor. We first decompose the deformation gradient into a stretch and a rotations using $\boldsymbol{F} = \boldsymbol{R} \cdot \boldsymbol{U}$. The rotation $\boldsymbol{R}$ is then used to rotate the stress and the rate of deformation into the material configuration to give us

$$\hat{\boldsymbol{\sigma}} = \boldsymbol{R}^T \cdot \boldsymbol{\sigma} \cdot \boldsymbol{R} \,; \quad \hat{\mathbf{d}} = \boldsymbol{R}^T \cdot \mathbf{d} \cdot \boldsymbol{R} \tag{5.1}$$

This is equivalent to using a Green-Naghdi objective stress rate. In the following all equations are with respect to the hatted quantities and we drop the hats for convenience.

5.2 The model

5.2.1 Elastic relation

Let us split the Cauchy stress into a volumetric and a deviatoric part

$$\boldsymbol{\sigma} = p\,\boldsymbol{1} + \mathbf{s} \,; \quad p = \tfrac{1}{3}\,\mathrm{tr}(\boldsymbol{\sigma}) \,. \tag{5.2}$$

Taking the time derivative gives us

$$\dot{\boldsymbol{\sigma}} = \dot{p}\,\boldsymbol{1} + \dot{\mathbf{s}} \,. \tag{5.3}$$

We assume that the elastic response of the material is isotropic. The constitutive relation for a hypoelastic material of grade o can be expressed as

$$\dot{\boldsymbol{\sigma}} = \left[\lambda\,\mathrm{tr}(\mathbf{d}^e) - 3\,\kappa\,\alpha\,\frac{d}{dt}(T - T_o)\right]\boldsymbol{1} + 2\,\mu\,\mathbf{d}^e \,; \quad \mathbf{d} = \mathbf{d}^e + \mathbf{d}^p \tag{5.4}$$

where $\mathbf{d}^e$, $\mathbf{d}^p$ are the elastic and plastic parts of the rate of deformation tensor, λ, μ are the Lame constants, κ is the bulk modulus, α is the coefficient of thermal expansion, T_o is the reference temperature, and T is the current temperature. If we split $\mathbf{d}^e$ into volumetric and deviatoric parts as

$$\mathbf{d}^e = \tfrac{1}{3}\,\mathrm{tr}(\mathbf{d}^e)\,\boldsymbol{1} + \boldsymbol{\eta}^e \tag{5.5}$$

we can write

$$\dot{\boldsymbol{\sigma}} = \left[\left(\lambda + \frac{2}{3}\mu\right)\,\mathrm{tr}(\mathbf{d}^e) - 3\,\kappa\,\alpha\,\frac{d}{dt}(T - T_0)\right]\boldsymbol{1} + 2\,\mu\,\boldsymbol{\eta}^e = \kappa\left[\mathrm{tr}(\mathbf{d}^e) - 3\,\alpha\,\frac{d}{dt}(T - T_0)\right]\boldsymbol{1} + 2\,\mu\,\boldsymbol{\eta}^e \quad (5.6)$$

Therefore, we have

$$\dot{\mathbf{s}} = 2\,\mu\,\boldsymbol{\eta}^e\,. \quad (5.7)$$

and

$$\dot{p} = \kappa\left[\mathrm{tr}(\mathbf{d}^e) - 3\,\alpha\,\frac{d}{dt}(T - T_0)\right]. \quad (5.8)$$

We will use a standard elastic-plastic stress update algorithm to integrate the rate equation for the deviatoric stress. However, we will assume that the volumetric part of the Cauchy stress can be computed using an equation of state. Then the final Cauchy stress will be given by

$$\boldsymbol{\sigma} = \left[p(J) - 3\,J\,\frac{dp(J)}{dJ}\,\alpha\,(T - T_0)\right]\boldsymbol{1} + \mathbf{s}\,;\quad J = \det(\boldsymbol{F})\,. \quad (5.9)$$

(Note that we assume that the plastic part of the deformation is volume preserving. This is not true for Gurson type models and will lead to a small error in the computed value of $\boldsymbol{\sigma}$.)

5.2.2 Flow rule

We assume that the flow rule is given by

$$\mathbf{d}^p = \dot{\gamma}\,\boldsymbol{r} \quad (5.10)$$

We can split $\mathbf{d}^p$ into a trace part and a trace free part, i.e.,

$$\mathbf{d}^p = \tfrac{1}{3}\,\mathrm{tr}(\mathbf{d}^p)\,\boldsymbol{1} + \boldsymbol{\eta}^p \quad (5.11)$$

Then, using the flow rule, we have

$$\mathbf{d}^p = \tfrac{1}{3}\,\dot{\gamma}\,\mathrm{tr}(\boldsymbol{r})\,\boldsymbol{1} + \boldsymbol{\eta}^p\,. \quad (5.12)$$

Therefore we can write the flow rule as

$$\boldsymbol{\eta}^p = \dot{\gamma}\,\left(-\tfrac{1}{3}\,\mathrm{tr}(\boldsymbol{r})\,\boldsymbol{1} + \boldsymbol{r}\right). \quad (5.13)$$

Note that

$$\mathbf{d} = \mathbf{d}^e + \mathbf{d}^p \quad\Longrightarrow\quad \mathrm{tr}(\mathbf{d}) = \mathrm{tr}(\mathbf{d}^e) + \mathrm{tr}(\mathbf{d}^p)\,. \quad (5.14)$$

Also,

$$\mathbf{d} = \mathbf{d}^e + \mathbf{d}^p \quad\Longrightarrow\quad \tfrac{1}{3}\,\mathrm{tr}(\mathbf{d})\,\boldsymbol{1} + \boldsymbol{\eta} = \tfrac{1}{3}\,\mathrm{tr}(\mathbf{d}^e)\,\boldsymbol{1} + \boldsymbol{\eta}^e + \tfrac{1}{3}\,\mathrm{tr}(\mathbf{d}^p)\,\boldsymbol{1} + \boldsymbol{\eta}^p\,. \quad (5.15)$$

Therefore,

$$\boldsymbol{\eta} = \boldsymbol{\eta}^e + \boldsymbol{\eta}^p\,. \quad (5.16)$$

5.2.3 Isotropic and Kinematic hardening and porosity evolution rules

We assume that the strain rate, temperature, and porosity can be fixed at the beginning of a timestep and consider only the evolution of plastic strain and the back stress while calculating the current stress.

We assume that the plastic strain evolves according to the relation

$$\dot{\varepsilon^p} = \dot{\gamma}\, h^{\alpha} \tag{5.17}$$

We also assume that the back stress evolves according to the relation

$$\dot{\hat{\boldsymbol{\beta}}} = \dot{\gamma}\, \boldsymbol{h}^{\beta} \tag{5.18}$$

where $\hat{\boldsymbol{\beta}}$ is the back stress. If $\boldsymbol{\beta}$ is the deviatoric part of $\hat{\boldsymbol{\beta}}$, then we can write

$$\dot{\boldsymbol{\beta}} = \dot{\gamma}\, \mathrm{dev}(\boldsymbol{h}^{\beta})\,. \tag{5.19}$$

The porosity ϕ is assumed to evolve according to the relation

$$\dot{\phi} = \dot{\gamma}\, h^{\phi}\,. \tag{5.20}$$

5.2.4 Yield condition

The yield condition is assumed to be of the form

$$f(\mathbf{s}, \boldsymbol{\beta}, \varepsilon^p, \phi, \dot{\varepsilon}, T, \dots) = f(\boldsymbol{\xi}, \varepsilon^p, \phi, \dot{\varepsilon}, T, \dots) = 0 \tag{5.21}$$

where $\boldsymbol{\xi} = \mathbf{s} - \boldsymbol{\beta}$ and $\boldsymbol{\beta}$ is the deviatoric part of $\hat{\boldsymbol{\beta}}$. The Kuhn-Tucker loading-unloading conditions are

$$\dot{\gamma} \geq 0\,;\quad f \leq 0\,;\quad \dot{\gamma}\, f = 0 \tag{5.22}$$

and the consistency condition is $\dot{f} = 0$.

5.2.5 Temperature increase due to plastic dissipation

The temperature increase due to plastic dissipation is assume to be given by the rate equation

$$\dot{T} = \frac{\chi}{\rho\, C_p}\, \sigma_y\, \dot{\varepsilon^p}\,. \tag{5.23}$$

The temperature is updated using

$$T_{n+1} = T_n + \frac{\chi_{n+1}\, \Delta t}{\rho_{n+1}\, C_p}\, \sigma_y^{n+1}\, \dot{\varepsilon^p}_{n+1}\,. \tag{5.24}$$

5.2.6 Continuum elastic-plastic tangent modulus

To determine whether the material has undergone a loss of stability we need to compute the acosutic tensor which needs the computation of the continuum elastic-plastic tangent modulus.

To do that recall that

$$\boldsymbol{\sigma} = p\, \boldsymbol{1} + \mathbf{s} \quad \Longrightarrow \quad \dot{\boldsymbol{\sigma}} = \dot{p}\, \boldsymbol{1}\,. \tag{5.25}$$

We assume that

$$\dot{p} = J\, \frac{\partial p}{\partial J}\, \mathrm{tr}(\mathbf{d}) \qquad \text{and} \quad \dot{\mathbf{s}} = 2\, \mu\, \boldsymbol{\eta}^e\,. \tag{5.26}$$

Now, the consistency condition requires that

$$\dot{f}(\mathbf{s}, \boldsymbol{\beta}, \varepsilon^p, \phi, \dot{\varepsilon}, T, \dots) = 0 \,. \tag{5.27}$$

Keeping $\dot{\varepsilon}$ and T fixed over the time interval, we can use the chain rule to get

$$\dot{f} = \frac{\partial f}{\partial \mathbf{s}} : \dot{\mathbf{s}} + \frac{\partial f}{\partial \boldsymbol{\beta}} : \dot{\boldsymbol{\beta}} + \frac{\partial f}{\partial \varepsilon^p}\, \dot{\varepsilon^p} + \frac{\partial f}{\partial \phi}\, \dot{\phi} = 0 \,. \tag{5.28}$$

The needed rate equations are

$$\begin{aligned} \dot{\mathbf{s}} &= 2\,\mu\,\boldsymbol{\eta}^e = 2\,\mu\,(\boldsymbol{\eta} - \boldsymbol{\eta}^p) = 2\,\mu\,[\boldsymbol{\eta} - \dot{\gamma}\,\mathrm{dev}(\boldsymbol{r})] \\ \dot{\boldsymbol{\beta}} &= \dot{\gamma}\,\mathrm{dev}(\boldsymbol{h}^\beta) \\ \dot{\varepsilon^p} &= \dot{\gamma}\,h^\alpha \\ \dot{\phi} &= \dot{\gamma}\,h^\phi \end{aligned} \tag{5.29}$$

Plugging these into the expression for $\dot{f}$ gives

$$2\,\mu\,\frac{\partial f}{\partial \mathbf{s}} : [\boldsymbol{\eta} - \dot{\gamma}\,\mathrm{dev}(\boldsymbol{r})] + \dot{\gamma}\,\frac{\partial f}{\partial \boldsymbol{\beta}} : \mathrm{dev}(\boldsymbol{h}^\beta) + \dot{\gamma}\,\frac{\partial f}{\partial \varepsilon^p}\,h^\alpha + \dot{\gamma}\,\frac{\partial f}{\partial \phi}\,h^\phi = 0 \tag{5.30}$$

or,

$$\dot{\gamma} = \frac{2\,\mu\,\dfrac{\partial f}{\partial \mathbf{s}} : \boldsymbol{\eta}}{2\,\mu\,\dfrac{\partial f}{\partial \mathbf{s}} : \mathrm{dev}(\boldsymbol{r}) - \dfrac{\partial f}{\partial \boldsymbol{\beta}} : \mathrm{dev}(\boldsymbol{h}^\beta) - \dfrac{\partial f}{\partial \varepsilon^p}\,h^\alpha - \dfrac{\partial f}{\partial \phi}\,h^\phi} \,. \tag{5.31}$$

Plugging this expression for $\dot{\gamma}$ into the equation for $\dot{\mathbf{s}}$, we get

$$\dot{\mathbf{s}} = 2\,\mu\left[\boldsymbol{\eta} - \left(\frac{2\,\mu\,\dfrac{\partial f}{\partial \mathbf{s}} : \boldsymbol{\eta}}{2\,\mu\,\dfrac{\partial f}{\partial \mathbf{s}} : \mathrm{dev}(\boldsymbol{r}) - \dfrac{\partial f}{\partial \boldsymbol{\beta}} : \mathrm{dev}(\boldsymbol{h}^\beta) - \dfrac{\partial f}{\partial \varepsilon^p}\,h^\alpha - \dfrac{\partial f}{\partial \phi}\,h^\phi}\right)\mathrm{dev}(\boldsymbol{r})\right] \,. \tag{5.32}$$

At this stage, note that a symmetric $\boldsymbol{\sigma}$ implies a symmetric $\mathbf{s}$ and hence a symmetric $\boldsymbol{\eta}$. Also we assume that $\boldsymbol{r}$ is symmetric (and hence $\mathrm{dev}(\boldsymbol{r})$), which is true if the flow rule is associated. Then we can write

$$\boldsymbol{\eta} = \mathbf{I}^{4s} : \boldsymbol{\eta} \qquad \text{and} \qquad \mathrm{dev}(\boldsymbol{r}) = \mathbf{I}^{4s} : \mathrm{dev}(\boldsymbol{r}) \tag{5.33}$$

where $\mathbf{I}^{4s}$ is the fourth-order symmetric identity tensor. Also note that if $\boldsymbol{A}$, $\boldsymbol{C}$, $\boldsymbol{D}$ are second order tensors and $\mathbf{B}$ is a fourth order tensor, then

$$(\boldsymbol{A} : \mathbf{B} : \boldsymbol{C})\,(\mathbf{B} : \boldsymbol{D}) \equiv A_{ij}\,B_{ijkl}\,C_{kl}\,B_{mnpq}\,D_{pq} = (B_{mnpq}\,D_{pq})\,(A_{ij}\,B_{ijkl})\,C_{kl} \equiv [(\mathbf{B} : \boldsymbol{D}) \otimes (\boldsymbol{A} : \mathbf{B})] : \boldsymbol{C} \,. \tag{5.34}$$

Therefore we have

$$\dot{\mathbf{s}} = 2\,\mu\left[\mathbf{I}^{4s} : \boldsymbol{\eta} - \left(\frac{2\,\mu\,[\mathbf{I}^{4s} : \mathrm{dev}(\boldsymbol{r})] \otimes [\dfrac{\partial f}{\partial \mathbf{s}} : \mathbf{I}^{4s}]}{2\,\mu\,\dfrac{\partial f}{\partial \mathbf{s}} : \mathrm{dev}(\boldsymbol{r}) - \dfrac{\partial f}{\partial \boldsymbol{\beta}} : \mathrm{dev}(\boldsymbol{h}^\beta) - \dfrac{\partial f}{\partial \varepsilon^p}\,h^\alpha - \dfrac{\partial f}{\partial \phi}\,h^\phi}\right) : \boldsymbol{\eta}\right] \,. \tag{5.35}$$

Also,

$$\mathbf{I}^{4s} : \mathrm{dev}(\boldsymbol{r}) = \mathrm{dev}(\boldsymbol{r}) \qquad \text{and} \qquad \frac{\partial f}{\partial \mathbf{s}} : \mathbf{I}^{4s} = \frac{\partial f}{\partial \mathbf{s}} \,. \tag{5.36}$$

Hence we can write

$$\dot{\mathbf{s}} = 2\,\mu\left[\mathbf{I}^{4s} - \left(\frac{2\,\mu\,\mathrm{dev}(\boldsymbol{r})\otimes\frac{\partial f}{\partial \mathbf{s}}}{2\,\mu\,\frac{\partial f}{\partial \mathbf{s}}:\mathrm{dev}(\boldsymbol{r}) - \frac{\partial f}{\partial \boldsymbol{\beta}}:\mathrm{dev}(\boldsymbol{h}^{\beta}) - \frac{\partial f}{\partial \varepsilon^{p}}\,h^{\alpha} - \frac{\partial f}{\partial \phi}\,h^{\phi}}\right)\right]:\boldsymbol{\eta} \tag{5.37}$$

or,

$$\dot{\mathbf{s}} = \mathbf{B}^{ep}:\boldsymbol{\eta} = \mathbf{B}^{ep}:\left[\mathbf{d} - \tfrac{1}{3}\,\mathrm{tr}(\mathbf{d})\,\boldsymbol{1}\right] \tag{5.38}$$

where

$$\mathbf{B}^{ep} := 2\,\mu\left[\mathbf{I}^{4s} - \left(\frac{2\,\mu\,\mathrm{dev}(\boldsymbol{r})\otimes\frac{\partial f}{\partial \mathbf{s}}}{2\,\mu\,\frac{\partial f}{\partial \mathbf{s}}:\mathrm{dev}(\boldsymbol{r}) - \frac{\partial f}{\partial \boldsymbol{\beta}}:\mathrm{dev}(\boldsymbol{h}^{\beta}) - \frac{\partial f}{\partial \varepsilon^{p}}\,h^{\alpha} - \frac{\partial f}{\partial \phi}\,h^{\phi}}\right)\right]. \tag{5.39}$$

Adding in the volumetric component gives

$$\begin{aligned}
\dot{\boldsymbol{\sigma}} &= \dot{p}\,\boldsymbol{1} + \dot{\mathbf{s}} \\
&= J\,\frac{\partial p}{\partial J}\,\mathrm{tr}(\mathbf{d})\,\boldsymbol{1} + \mathbf{B}^{ep}:\left[\mathbf{d} - \tfrac{1}{3}\,\mathrm{tr}(\mathbf{d})\,\boldsymbol{1}\right] \\
&= \left[3\,J\,\frac{\partial p}{\partial J}\,\boldsymbol{1} - \mathbf{B}^{ep}:\boldsymbol{1}\right]\frac{\mathbf{d}:\boldsymbol{1}}{3} + \mathbf{B}^{ep}:\mathbf{d} \\
&= J\,\frac{\partial p}{\partial J}\,(\boldsymbol{1}\otimes\boldsymbol{1}):\mathbf{d} - \tfrac{1}{3}\left[\mathbf{B}^{ep}:(\boldsymbol{1}\otimes\boldsymbol{1})\right]:\mathbf{d} + \mathbf{B}^{ep}:\mathbf{d}\,.
\end{aligned} \tag{5.40}$$

Therefore,

$$\dot{\boldsymbol{\sigma}} = \left[J\,\frac{\partial p}{\partial J}\,(\boldsymbol{1}\otimes\boldsymbol{1}) - \tfrac{1}{3}\left[\mathbf{B}^{ep}:(\boldsymbol{1}\otimes\boldsymbol{1})\right] + \mathbf{B}^{ep}\right]:\mathbf{d} = \mathbf{C}^{ep}:\mathbf{d}\,. \tag{5.41}$$

The quantity $\mathbf{C}^{ep}$ is the continuum elastic-plastic tangent modulus. We also use the continuum elastic-plastic tangent modulus in the implicit version of the code. However, for improved accuracy and faster convergence, an algorithmically consistent tangent modulus should be used instead. That tangent modulus can be calculated in the usual manner and is left for development and implementation as an additional feature in the future.

5.3 Stress update

A standard return algorithm is used to compute the updated Cauchy stress. Recall that the rate equation for the deviatoric stress is given by

$$\dot{\mathbf{s}} = 2\,\mu\,\boldsymbol{\eta}^{e}\,. \tag{5.42}$$

Integrating the rate equation using a Backward Euler scheme gives

$$\mathbf{s}_{n+1} - \mathbf{s}_{n} = 2\,\mu\,\Delta\,t\,\boldsymbol{\eta}^{e}_{n+1} = 2\,\mu\,\Delta\,t\,(\boldsymbol{\eta}_{n+1} - \boldsymbol{\eta}^{p}_{n+1}) \tag{5.43}$$

Now, from the flow rule, we have

$$\boldsymbol{\eta}^{p} = \dot{\gamma}\,\left(\boldsymbol{r} - \tfrac{1}{3}\,\mathrm{tr}(\boldsymbol{r})\,\boldsymbol{1}\right)\,. \tag{5.44}$$

Define the deviatoric part of $\boldsymbol{r}$ as

$$\mathrm{dev}(\boldsymbol{r}) := \boldsymbol{r} - \tfrac{1}{3}\,\mathrm{tr}(\boldsymbol{r})\,\boldsymbol{1}\,. \tag{5.45}$$

Therefore,

$$\mathbf{s}_{n+1} - \mathbf{s}_n = 2\,\mu\,\Delta\,t\,\boldsymbol{\eta}_{n+1} - 2\,\mu\,\Delta\gamma_{n+1}\,\mathrm{dev}(\boldsymbol{r}_{n+1})\,. \tag{5.46}$$

where $\Delta\gamma := \dot{\gamma}\,\Delta t$. Define the trial stress

$$\mathbf{s}^{\mathrm{trial}} := \mathbf{s}_n + 2\,\mu\,\Delta\,t\,\boldsymbol{\eta}_{n+1}\,. \tag{5.47}$$

Then

$$\mathbf{s}_{n+1} = \mathbf{s}^{\mathrm{trial}} - 2\,\mu\,\Delta\gamma_{n+1}\,\mathrm{dev}(\boldsymbol{r}_{n+1})\,. \tag{5.48}$$

Also recall that the back stress is given by

$$\dot{\boldsymbol{\beta}} = \dot{\gamma}\,\mathrm{dev}\boldsymbol{h}^{\beta} \tag{5.49}$$

The evolution equation for the back stress can be integrated to get

$$\boldsymbol{\beta}_{n+1} - \boldsymbol{\beta}_n = \Delta\gamma_{n+1}\,\mathrm{dev}(\boldsymbol{h})^{\beta}_{n+1}\,. \tag{5.50}$$

Now,

$$\boldsymbol{\xi}_{n+1} = \mathbf{s}_{n+1} - \boldsymbol{\beta}_{n+1}\,. \tag{5.51}$$

Plugging in the expressions for $\mathbf{s}_{n+1}$ and $\boldsymbol{\beta}_{n+1}$, we get

$$\boldsymbol{\xi}_{n+1} = \mathbf{s}^{\mathrm{trial}} - 2\,\mu\,\Delta\gamma_{n+1}\,\mathrm{dev}(\boldsymbol{r}_{n+1}) - \boldsymbol{\beta}_n - \Delta\gamma_{n+1}\,\mathrm{dev}(\boldsymbol{h})^{\beta}_{n+1}\,. \tag{5.52}$$

Define

$$\boldsymbol{\xi}^{\mathrm{trial}} := \mathbf{s}^{\mathrm{trial}} - \boldsymbol{\beta}_n\,. \tag{5.53}$$

Then

$$\boldsymbol{\xi}_{n+1} = \boldsymbol{\xi}^{\mathrm{trial}} - \Delta\gamma_{n+1}(2\,\mu\,\mathrm{dev}(\boldsymbol{r}_{n+1}) + \mathrm{dev}(\boldsymbol{h})^{\beta}_{n+1})\,. \tag{5.54}$$

Similarly, the evolution of the plastic strain is given by

$$\varepsilon^p_{n+1} = \varepsilon^p_n + \Delta\gamma_{n+1}\,h^{\alpha}_{n+1} \tag{5.55}$$

and the porosity evolves as

$$\phi_{n+1} = \phi_n + \Delta\gamma_{n+1}\,h^{\phi}_{n+1}\,. \tag{5.56}$$

The yield condition is discretized as

$$f(\mathbf{s}_{n+1}, \boldsymbol{\beta}_{n+1}, \varepsilon^p_{n+1}, \phi_{n+1}, \dot{\varepsilon}_{n+1}, T_{n+1}, \dots) = f(\boldsymbol{\xi}_{n+1}, \varepsilon^p_{n+1}, \phi_{n+1}, \dot{\varepsilon}_{n+1}, T_{n+1}, \dots) = 0\,. \tag{5.57}$$

Important: We assume that the derivatives with respect to $\dot{\varepsilon}$ and T are small enough to be neglected.

5.3.1 Newton iterations

We now have the following equations that have to be solved for $\Delta\gamma_{n+1}$:

$$\begin{aligned}
\boldsymbol{\xi}_{n+1} &= \boldsymbol{\xi}^{\text{trial}} - \Delta\gamma_{n+1}(2\,\mu\,\text{dev}(\boldsymbol{r}_{n+1}) + \text{dev}(\boldsymbol{h})^{\beta}_{n+1}) \\
\varepsilon^{p}_{n+1} &= \varepsilon^{p}_{n} + \Delta\gamma_{n+1}\, h^{\alpha}_{n+1} \\
\phi_{n+1} &= \phi_{n} + \Delta\gamma_{n+1}\, h^{\phi}_{n+1} \\
f(\boldsymbol{\xi}_{n+1}, \varepsilon^{p}_{n+1}, \phi_{n+1}, \dot{\varepsilon}_{n+1}, T_{n+1}, \dots) &= 0\,.
\end{aligned} \tag{5.58}$$

Recall that if $g(\Delta\gamma) = 0$ is a nonlinear equation that we have to solve for $\Delta\gamma$, an iterative Newton method can be expressed as

$$\Delta\gamma^{(k+1)} = \Delta\gamma^{(k)} - \left[\frac{dg}{d\Delta\gamma}\right]^{-1}_{(k)} g^{(k)}\,. \tag{5.59}$$

Define

$$\delta\gamma := \Delta\gamma^{(k+1)} - \Delta\gamma^{(k)}\,. \tag{5.60}$$

Then, the iterative scheme can be written as

$$g^{(k)} + \left[\frac{dg}{d\Delta\gamma}\right]^{(k)} \delta\gamma = 0\,. \tag{5.61}$$

In our case we have

$$\begin{aligned}
\mathbf{a}(\Delta\gamma) &= 0 = -\boldsymbol{\xi} + \boldsymbol{\xi}^{\text{trial}} - \Delta\gamma(2\,\mu\,\text{dev}(\boldsymbol{r}) + \text{dev}(\boldsymbol{h})^{\beta}) \\
b(\Delta\gamma) &= 0 = -\varepsilon^{p} + \varepsilon^{p}_{n} + \Delta\gamma\, h^{\alpha} \\
c(\Delta\gamma) &= 0 = -\phi + \phi_{n} + \Delta\gamma\, h^{\phi} \\
f(\Delta\gamma) &= 0 = f(\boldsymbol{\xi}, \varepsilon^{p}, \phi, \dot{\varepsilon}, T, \dots)
\end{aligned} \tag{5.62}$$

Therefore,

$$\begin{aligned}
\frac{d\mathbf{a}}{d\Delta\gamma} &= -\frac{\partial\boldsymbol{\xi}}{\partial\Delta\gamma} - (2\,\mu\,\text{dev}(\boldsymbol{r}) + \text{dev}(\boldsymbol{h})^{\beta}) - \Delta\gamma\left(2\,\mu\,\frac{\partial\text{dev}(\boldsymbol{r})}{\partial\Delta\gamma} + \frac{\partial\text{dev}(\boldsymbol{h})^{\beta}}{\partial\Delta\gamma}\right) \\
&= -\frac{\partial\boldsymbol{\xi}}{\partial\Delta\gamma} - (2\,\mu\,\text{dev}(\boldsymbol{r}) + \text{dev}(\boldsymbol{h})^{\beta}) - \Delta\gamma\left(2\,\mu\,\frac{\partial\text{dev}(\boldsymbol{r})}{\partial\boldsymbol{\xi}} : \frac{\partial\boldsymbol{\xi}}{\partial\Delta\gamma} + 2\,\mu\,\frac{\partial\text{dev}(\boldsymbol{r})}{\partial\varepsilon^{p}}\,\frac{\partial\varepsilon^{p}}{\partial\Delta\gamma} + 2\,\mu\,\frac{\partial\text{dev}(\boldsymbol{r})}{\partial\phi}\,\frac{\partial\phi}{\partial\Delta\gamma} + \right. \\
&\qquad \left. \frac{\partial\text{dev}(\boldsymbol{h})^{\beta}}{\partial\boldsymbol{\xi}} : \frac{\partial\boldsymbol{\xi}}{\partial\Delta\gamma} + \frac{\partial\text{dev}(\boldsymbol{h})^{\beta}}{\partial\varepsilon^{p}}\,\frac{\partial\varepsilon^{p}}{\partial\Delta\gamma} + \frac{\partial\text{dev}(\boldsymbol{h})^{\beta}}{\partial\phi}\,\frac{\partial\phi}{\partial\Delta\gamma}\right) \\
\frac{db}{d\Delta\gamma} &= -\frac{\partial\varepsilon^{p}}{\partial\Delta\gamma} + h^{\alpha} + \Delta\gamma\left(\frac{\partial h^{\alpha}}{\partial\boldsymbol{\xi}} : \frac{\partial\boldsymbol{\xi}}{\partial\Delta\gamma} + \frac{\partial h^{\alpha}}{\partial\varepsilon^{p}}\,\frac{\partial\varepsilon^{p}}{\partial\Delta\gamma} + \frac{\partial h^{\alpha}}{\partial\phi}\,\frac{\partial\phi}{\partial\Delta\gamma}\right) \\
\frac{dc}{d\Delta\gamma} &= -\frac{\partial\phi}{\partial\Delta\gamma} + h^{\phi} + \Delta\gamma\left(\frac{\partial h^{\phi}}{\partial\boldsymbol{\xi}} : \frac{\partial\boldsymbol{\xi}}{\partial\Delta\gamma} + \frac{\partial h^{\phi}}{\partial\varepsilon^{p}}\,\frac{\partial\varepsilon^{p}}{\partial\Delta\gamma} + \frac{\partial h^{\phi}}{\partial\phi}\,\frac{\partial\phi}{\partial\Delta\gamma}\right) \\
\frac{df}{d\Delta\gamma} &= \frac{\partial f}{\partial\boldsymbol{\xi}} : \frac{\partial\boldsymbol{\xi}}{\partial\Delta\gamma} + \frac{\partial f}{\partial\varepsilon^{p}}\,\frac{\partial\varepsilon^{p}}{\partial\Delta\gamma} + \frac{\partial f}{\partial\phi}\,\frac{\partial\phi}{\partial\Delta\gamma}\,.
\end{aligned} \tag{5.63}$$

Now, define

$$\Delta\boldsymbol{\xi} := \frac{\partial\boldsymbol{\xi}}{\partial\Delta\gamma}\,\delta\gamma\,;\quad \Delta\varepsilon^{p} := \frac{\partial\varepsilon^{p}}{\partial\Delta\gamma}\,\delta\gamma\,;\quad \Delta\phi := \frac{\partial\phi}{\partial\Delta\gamma}\,\delta\gamma\,. \tag{5.64}$$

Then

$$\begin{aligned}
&\mathbf{a}^{(k)} - \Delta\boldsymbol{\xi} - [2\,\mu\,\mathrm{dev}(\boldsymbol{r}^{(k)}) + \mathrm{dev}(\boldsymbol{h})^{\beta(k)}]\,\delta\gamma \\
&\qquad - 2\,\mu\,\Delta\gamma\left(\frac{\partial \mathrm{dev}(\boldsymbol{r}^{(k)})}{\partial\boldsymbol{\xi}} : \Delta\boldsymbol{\xi} + \frac{\partial \mathrm{dev}(\boldsymbol{r}^{(k)})}{\partial\varepsilon^p}\,\Delta\varepsilon^p + \frac{\partial \mathrm{dev}(\boldsymbol{r}^{(k)})}{\partial\phi}\,\Delta\phi\right) \\
&\qquad - \Delta\gamma\left(\frac{\partial \mathrm{dev}(\boldsymbol{h})^{\beta(k)}}{\partial\boldsymbol{\xi}} : \Delta\boldsymbol{\xi} + \frac{\partial \mathrm{dev}(\boldsymbol{h})^{\beta(k)}}{\partial\varepsilon^p}\,\Delta\varepsilon^p + \frac{\partial \mathrm{dev}(\boldsymbol{h})^{\beta(k)}}{\partial\phi}\,\Delta\phi\right) = 0 \\
&b^{(k)} - \Delta\varepsilon^p + h^\alpha\,\delta\gamma + \Delta\gamma\left(\frac{\partial h^{\alpha(k)}}{\partial\boldsymbol{\xi}} : \Delta\boldsymbol{\xi} + \frac{\partial h^{\alpha(k)}}{\partial\varepsilon^p}\,\Delta\varepsilon^p + \frac{\partial h^{\alpha(k)}}{\partial\phi}\,\Delta\phi\right) = 0 \\
&c^{(k)} - \Delta\phi + h^\phi\,\delta\gamma + \Delta\gamma\left(\frac{\partial h^{\phi(k)}}{\partial\boldsymbol{\xi}} : \Delta\boldsymbol{\xi} + \frac{\partial h^{\phi(k)}}{\partial\varepsilon^p}\,\Delta\varepsilon^p + \frac{\partial h^{\phi(k)}}{\partial\phi}\,\Delta\phi\right) = 0 \\
&f^{(k)} + \frac{\partial f^{(k)}}{\partial\boldsymbol{\xi}} : \Delta\boldsymbol{\xi} + \frac{\partial f^{(k)}}{\partial\varepsilon^p}\,\Delta\varepsilon^p + \frac{\partial f^{(k)}}{\partial\phi}\,\Delta\phi = 0
\end{aligned} \tag{5.65}$$

Because the derivatives of $\boldsymbol{r}^{(k)}, \boldsymbol{h}^{\alpha(k)}, \boldsymbol{h}^{\beta(k)}, \boldsymbol{h}^{\phi(k)}$ with respect to $\boldsymbol{\xi}, \varepsilon^p, \phi$ may be difficult to calculate, we instead use a semi-implicit scheme in our implementation where the quantities $\boldsymbol{r}$, h^α, $\boldsymbol{h}^\beta$, and h^ϕ are evaluated at t_n. Then the problematic derivatives disappear and we are left with

$$\begin{aligned}
\mathbf{a}^{(k)} - \Delta\boldsymbol{\xi} - [2\,\mu\,\mathrm{dev}(\boldsymbol{r}_n) + \mathrm{dev}(\boldsymbol{h})_n^\beta]\,\delta\gamma &= 0 \\
b^{(k)} - \Delta\varepsilon^p + h_n^\alpha\,\delta\gamma &= 0 \\
c^{(k)} - \Delta\phi + h_n^\phi\,\delta\gamma &= 0 \\
f^{(k)} + \frac{\partial f^{(k)}}{\partial\boldsymbol{\xi}} : \Delta\boldsymbol{\xi} + \frac{\partial f^{(k)}}{\partial\varepsilon^p}\,\Delta\varepsilon^p + \frac{\partial f^{(k)}}{\partial\phi}\,\Delta\phi &= 0
\end{aligned} \tag{5.66}$$

We now force $\mathbf{a}^{(k)}$, $b^{(k)}$, and $c^{(k)}$ to be zero at all times, leading to the expressions

$$\begin{aligned}
\Delta\boldsymbol{\xi} &= -[2\,\mu\,\mathrm{dev}(\boldsymbol{r}_n) + \mathrm{dev}(\boldsymbol{h})_n^\beta]\,\delta\gamma \\
\Delta\varepsilon^p &= h_n^\alpha\,\delta\gamma \\
\Delta\phi &= h_n^\phi\,\delta\gamma \\
f^{(k)} + \frac{\partial f^{(k)}}{\partial\boldsymbol{\xi}} : \Delta\boldsymbol{\xi} + \frac{\partial f^{(k)}}{\partial\varepsilon^p}\,\Delta\varepsilon^p + \frac{\partial f^{(k)}}{\partial\phi}\,\Delta\phi &= 0
\end{aligned} \tag{5.67}$$

Plugging the expressions for $\Delta\boldsymbol{\xi}$, $\Delta\varepsilon^p$, $\Delta\phi$ from the first three equations into the fourth gives us

$$f^{(k)} - \frac{\partial f^{(k)}}{\partial\boldsymbol{\xi}} : [2\,\mu\,\mathrm{dev}(\boldsymbol{r}_n) + \mathrm{dev}(\boldsymbol{h})_n^\beta]\,\delta\gamma + h_n^\alpha\,\frac{\partial f^{(k)}}{\partial\varepsilon^p}\,\delta\gamma + h_n^\phi\,\frac{\partial f^{(k)}}{\partial\phi}\,\delta\gamma = 0 \tag{5.68}$$

or

$$\Delta\gamma^{(k+1)} - \Delta\gamma^{(k)} = \delta\gamma = \frac{f^{(k)}}{\frac{\partial f^{(k)}}{\partial\boldsymbol{\xi}} : [2\,\mu\,\mathrm{dev}(\boldsymbol{r}_n) + \mathrm{dev}(\boldsymbol{h}_n^\beta)] - h_n^\alpha\,\frac{\partial f^{(k)}}{\partial\varepsilon^p} - h_n^\phi\,\frac{\partial f^{(k)}}{\partial\phi}}. \tag{5.69}$$

5.3.2 Algorithm

The following stress update algorithm is used for each (plastic) time step:

1. Initialize:

$$k = 0\,;\quad (\varepsilon^p)^{(k)} = \varepsilon_n^p\,;\quad \phi^{(k)} = \phi_n\,;\quad \boldsymbol{\beta}^{(k)} = \boldsymbol{\beta}_n\,;\quad \Delta\gamma^{(k)} = 0\,;\quad \boldsymbol{\xi}^{(k)} = \boldsymbol{\xi}^{\mathrm{trial}}\,. \tag{5.70}$$

2. Check yield condition:

$$f^{(k)} := f(\boldsymbol{\xi}^{(k)}, (\varepsilon^p)^{(k)}, \phi^{(k)}, \dot{\varepsilon}_n, T_n, \dots) \tag{5.71}$$

If $f^{(k)} <$ tolerance then go to step 5 else go to step 3.

3. Compute updated $\delta\gamma^{(k)}$ using

$$\delta\gamma^{(k)} = \frac{f^{(k)}}{\dfrac{\partial f^{(k)}}{\partial \boldsymbol{\xi}} : [2\,\mu\,\mathrm{dev}(\boldsymbol{r}_n) + \mathrm{dev}(\boldsymbol{h}_n^{\beta})] - h_n^{\alpha}\,\dfrac{\partial f^{(k)}}{\partial \varepsilon^p} - h_n^{\phi}\,\dfrac{\partial f^{(k)}}{\partial \phi}}\,. \tag{5.72}$$

Compute

$$\begin{aligned} \Delta\boldsymbol{\xi}^{(k)} &= -[2\,\mu\,\mathrm{dev}(\boldsymbol{r}_n) + \mathrm{dev}(\boldsymbol{h}_n^{\beta})]\,\delta\gamma^{(k)} \\ (\Delta\varepsilon^p)^{(k)} &= h_n^{\alpha}\,\delta\gamma^{(k)} \\ \Delta\phi^{(k)} &= h_n^{\phi}\,\delta\gamma^{(k)} \end{aligned} \tag{5.73}$$

4. Update variables:

$$\begin{aligned} (\varepsilon^p)^{(k+1)} &= (\varepsilon^p)^{(k)} + (\Delta\varepsilon^p)^{(k)} \\ \phi^{(k+1)} &= \phi^{(k)} + \Delta\phi^{(k)} \\ \boldsymbol{\xi}^{(k+1)} &= \boldsymbol{\xi}^{(k)} + \Delta\boldsymbol{\xi}^{(k)} \\ \Delta\gamma^{(k+1)} &= \Delta\gamma^{(k)} + \delta\gamma^{(k)} \end{aligned} \tag{5.74}$$

Set $k \leftarrow k+1$ and go to step 2.

5. Update and calculate back stress and the deviatoric part of Cauchy stress:

$$\varepsilon^p_{n+1} = (\varepsilon^p)^{(k)}\;;\quad \phi_{n+1} = \phi^{(k)}\;;\quad \boldsymbol{\xi}_{n+1} = \boldsymbol{\xi}^{(k)}\;;\quad \Delta\gamma_{n+1} = \Delta\gamma^{(k)} \tag{5.75}$$

and

$$\begin{aligned} \hat{\boldsymbol{\beta}}_{n+1} &= \hat{\boldsymbol{\beta}}_n + \Delta\gamma_{n+1}\,\boldsymbol{h}^{\beta}(\boldsymbol{\xi}_{n+1}, \varepsilon^p_{n+1}, \phi_{n+1}) \\ \boldsymbol{\beta}_{n+1} &= \hat{\boldsymbol{\beta}}_{n+1} - \tfrac{1}{3}\,\mathrm{tr}(\hat{\boldsymbol{\beta}}_{n+1})\,\boldsymbol{1} \\ \mathbf{s}_{n+1} &= \boldsymbol{\xi}_{n+1} + \boldsymbol{\beta}_{n+1} \end{aligned} \tag{5.76}$$

6. Update the temperature and the Cauchy stress

$$\begin{aligned} T_{n+1} &= T_n + \frac{\chi_{n+1}\,\Delta t}{\rho_{n+1}\,C_p}\,\sigma_y^{n+1}\,\dot{\varepsilon^p}_{n+1} = T_n + \frac{\chi_{n+1}\,\Delta\gamma_{n+1}}{\rho_{n+1}\,C_p}\,\sigma_y^{n+1}\,h^{\alpha}_{n+1} \\ p_{n+1} &= p(J_{n+1}) \\ \kappa_{n+1} &= J_{n+1}\left[\frac{dp(J)}{dJ}\right]_{n+1} \\ \boldsymbol{\sigma}_{n+1} &= [p_{n+1} - 3\,\kappa_{n+1}\,\alpha\,(T_{n+1} - T_0)]\,\boldsymbol{1} + \mathbf{s}_{n+1} \end{aligned} \tag{5.77}$$

5.4 Examples

Let us now look at a few examples.

5.4.1 Example 1

Consider the case of J_2 plasticity with the yield condition

$$f := \sqrt{\frac{3}{2}}\,\|\mathbf{s}-\boldsymbol{\beta}\| - \sigma_y(\varepsilon^p, \dot{\varepsilon}, T, \dots) = \sqrt{\frac{3}{2}}\,\|\boldsymbol{\xi}\| - \sigma_y(\varepsilon^p, \dot{\varepsilon}, T, \dots) \le 0 \tag{5.78}$$

where $\|\boldsymbol{\xi}\| = \sqrt{\boldsymbol{\xi} : \boldsymbol{\xi}}$. Assume the associated flow rule

$$\mathbf{d}^p = \dot{\gamma}\,\boldsymbol{r} = \dot{\gamma}\,\frac{\partial f}{\partial \boldsymbol{\sigma}} = \dot{\gamma}\,\frac{\partial f}{\partial \boldsymbol{\xi}}\,. \tag{5.79}$$

Then

$$\boldsymbol{r} = \frac{\partial f}{\partial \boldsymbol{\xi}} = \sqrt{\frac{3}{2}}\,\frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|} \tag{5.80}$$

and

$$\mathbf{d}^p = \sqrt{\frac{3}{2}}\,\dot{\gamma}\,\frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|}\,;\quad \|\mathbf{d}^p\| = \sqrt{\frac{3}{2}}\,\dot{\gamma}\,. \tag{5.81}$$

The evolution of the equivalent plastic strain is given by

$$\dot{\varepsilon^p} = \dot{\gamma}\,h^\alpha = \sqrt{\frac{2}{3}}\,\|\mathbf{d}^p\| = \dot{\gamma}\,. \tag{5.82}$$

This definition is consistent with the definition of equivalent plastic strain

$$\varepsilon^p = \int_0^t \dot{\varepsilon}^p\,d\tau = \int_0^t \sqrt{\frac{2}{3}}\,\|\mathbf{d}^p\|\,d\tau\,. \tag{5.83}$$

The evolution of porosity is given by (there is no evolution of porosity)

$$\dot{\phi} = \dot{\gamma}\,h^\phi = 0 \tag{5.84}$$

The evolution of the back stress is given by the Prager kinematic hardening rule

$$\dot{\hat{\boldsymbol{\beta}}} = \dot{\gamma}\,\boldsymbol{h}^\beta = \frac{2}{3}\,H'\,\mathbf{d}^p \tag{5.85}$$

where $\hat{\boldsymbol{\beta}}$ is the back stress and H' is a constant hardening modulus. Also, the trace of $\mathbf{d}^p$ is

$$\mathrm{tr}(\mathbf{d}^p) = \sqrt{\frac{3}{2}}\,\dot{\gamma}\,\frac{\mathrm{tr}(\boldsymbol{\xi})}{\|\boldsymbol{\xi}\|}\,. \tag{5.86}$$

Since $\boldsymbol{\xi}$ is deviatoric, $\mathrm{tr}(\boldsymbol{\xi}) = 0$ and hence $\mathbf{d}^p = \boldsymbol{\eta}^p$. Hence, $\hat{\boldsymbol{\beta}} = \boldsymbol{\beta}$ (where $\boldsymbol{\beta}$ is the deviatoric part of $\hat{\boldsymbol{\beta}}$), and

$$\dot{\boldsymbol{\beta}} = \sqrt{\frac{2}{3}}\,H'\,\dot{\gamma}\,\frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|}\,. \tag{5.87}$$

These relation imply that

$$\boxed{\begin{aligned} \boldsymbol{r} &= \sqrt{\frac{3}{2}}\,\frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|} \\ h^\alpha &= 1 \\ h^\phi &= 0 \\ \boldsymbol{h}^\beta &= \sqrt{\frac{2}{3}}\,H'\,\frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|}\,. \end{aligned}} \tag{5.88}$$

We also need some derivatives of the yield function. These are

$$\begin{aligned}\frac{\partial f}{\partial \boldsymbol{\xi}} &= \boldsymbol{r}\\ \frac{\partial f}{\partial \varepsilon^p} &= -\frac{\partial \sigma_y}{\partial \varepsilon^p}\\ \frac{\partial f}{\partial \phi} &= 0\,.\end{aligned} \tag{5.89}$$

Let us change the kinematic hardening model and use the Armstrong-Frederick model instead, i.e.,

$$\dot{\boldsymbol{\beta}} = \dot{\gamma}\,\boldsymbol{h}^{\beta} = \frac{2}{3}\,H_1\,\mathbf{d}^p - H_2\,\boldsymbol{\beta}\,\left\|\mathbf{d}^p\right\|\,. \tag{5.90}$$

Since

$$\mathbf{d}^p = \sqrt{\frac{3}{2}}\,\dot{\gamma}\,\frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|} \tag{5.91}$$

we have

$$\left\|\mathbf{d}^p\right\| = \sqrt{\frac{3}{2}}\,\dot{\gamma}\,\frac{\|\boldsymbol{\xi}\|}{\|\boldsymbol{\xi}\|} = \sqrt{\frac{3}{2}}\,\dot{\gamma}\,. \tag{5.92}$$

Therefore,

$$\dot{\boldsymbol{\beta}} = \sqrt{\frac{2}{3}}\,H_1\,\dot{\gamma}\,\frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|} - \sqrt{\frac{3}{2}}\,H_2\,\dot{\gamma}\,\boldsymbol{\beta}\,. \tag{5.93}$$

Hence we have

$$\boxed{\boldsymbol{h}^{\beta} = \sqrt{\frac{2}{3}}\,H_1\,\frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|} - \sqrt{\frac{3}{2}}\,H_2\,\boldsymbol{\beta}\,.} \tag{5.94}$$

5.4.2 Example 2

Let us now consider a Gurson type yield condition with kinematic hardening. In this case the yield condition can be written as

$$f := \frac{3\,\boldsymbol{\xi}:\boldsymbol{\xi}}{2\,\sigma_y^2} + 2\,q_1\,\phi^*\,\cosh\left(\frac{q_2\,\mathrm{tr}(\boldsymbol{\sigma})}{2\,\sigma_y}\right) - \left[1 + q_3\,(\phi^*)^2\right] \tag{5.95}$$

where ϕ is the porosity and

$$\phi^* = \begin{cases}\phi & \text{for } \phi \le \phi_c\\ \phi_c - \dfrac{\phi_u^* - \phi_c}{\phi_f - \phi_c}\,(\phi - \phi_c) & \text{for } \phi > \phi_c\end{cases} \tag{5.96}$$

Final fracture occurs for $\phi = \phi_f$ or when $\phi_u^* = 1/q_1$.

Let us use an associated flow rule

$$\mathbf{d}^p = \dot{\gamma}\,\boldsymbol{r} = \dot{\gamma}\,\frac{\partial f}{\partial \boldsymbol{\sigma}}\,. \tag{5.97}$$

Then

$$\boldsymbol{r} = \frac{\partial f}{\partial \boldsymbol{\sigma}} = \frac{3\,\boldsymbol{\xi}}{\sigma_y^2} + \frac{q_1\,q_2\,\phi^*}{\sigma_y}\,\sinh\left(\frac{q_2\,\mathrm{tr}(\boldsymbol{\sigma})}{2\,\sigma_y}\right)\boldsymbol{1}\,. \tag{5.98}$$

In this case

$$\operatorname{tr}(\boldsymbol{r}) = \frac{3\, q_1\, q_2\, \phi^*}{\sigma_y}\, \sinh\left(\frac{q_2\, \operatorname{tr}(\boldsymbol{\sigma})}{2\, \sigma_y}\right) \neq 0 \tag{5.99}$$

Therefore,

$$\mathbf{d}^p \neq \boldsymbol{\eta}^p \,. \tag{5.100}$$

For the evolution equation for the plastic strain we use

$$(\boldsymbol{\sigma} - \hat{\boldsymbol{\beta}}) : \mathbf{d}^p = (1-\phi)\, \sigma_y\, \dot{\varepsilon}^p \tag{5.101}$$

where $\dot{\varepsilon}^p$ is the effective plastic strain rate in the matrix material. Hence,

$$\dot{\varepsilon}^p = \dot{\gamma}\, h^\alpha = \dot{\gamma}\, \frac{(\boldsymbol{\sigma} - \hat{\boldsymbol{\beta}}) : \boldsymbol{r}}{(1-\phi)\, \sigma_y} \,. \tag{5.102}$$

The evolution equation for the porosity is given by

$$\dot{\phi} = (1-\phi)\, \operatorname{tr}(\mathbf{d}^p) + A\, \dot{\varepsilon^p} \tag{5.103}$$

where

$$A = \frac{f_n}{s_n \sqrt{2\pi}} \exp[-1/2(\varepsilon^p - \varepsilon_n)^2 / s_n^2] \tag{5.104}$$

and f_n is the volume fraction of void nucleating particles, ε_n is the mean of the normal distribution of nucleation strains, and s_n is the standard deviation of the distribution.

Therefore,

$$\dot{\phi} = \dot{\gamma}\, h^\phi = \dot{\gamma} \left[(1-\phi)\, \operatorname{tr}(\boldsymbol{r}) + A\, \frac{(\boldsymbol{\sigma} - \hat{\boldsymbol{\beta}}) : \boldsymbol{r}}{(1-\phi)\, \sigma_y} \right] . \tag{5.105}$$

If the evolution of the back stress is given by the Prager kinematic hardening rule

$$\dot{\hat{\boldsymbol{\beta}}} = \dot{\gamma}\, \boldsymbol{h}^\beta = \frac{2}{3}\, H'\, \mathbf{d}^p \tag{5.106}$$

where $\hat{\boldsymbol{\beta}}$ is the back stress, then

$$\dot{\hat{\boldsymbol{\beta}}} = \frac{2}{3}\, H'\, \dot{\gamma}\, \boldsymbol{r} \,. \tag{5.107}$$

Alternatively, if we use the Armstrong-Frederick model, then

$$\dot{\hat{\boldsymbol{\beta}}} = \dot{\gamma}\, \boldsymbol{h}^\beta = \frac{2}{3}\, H_1\, \mathbf{d}^p - H_2\, \hat{\boldsymbol{\beta}}\, \left\| \mathbf{d}^p \right\| \,. \tag{5.108}$$

Plugging in the expression for $\mathbf{d}^p$, we have

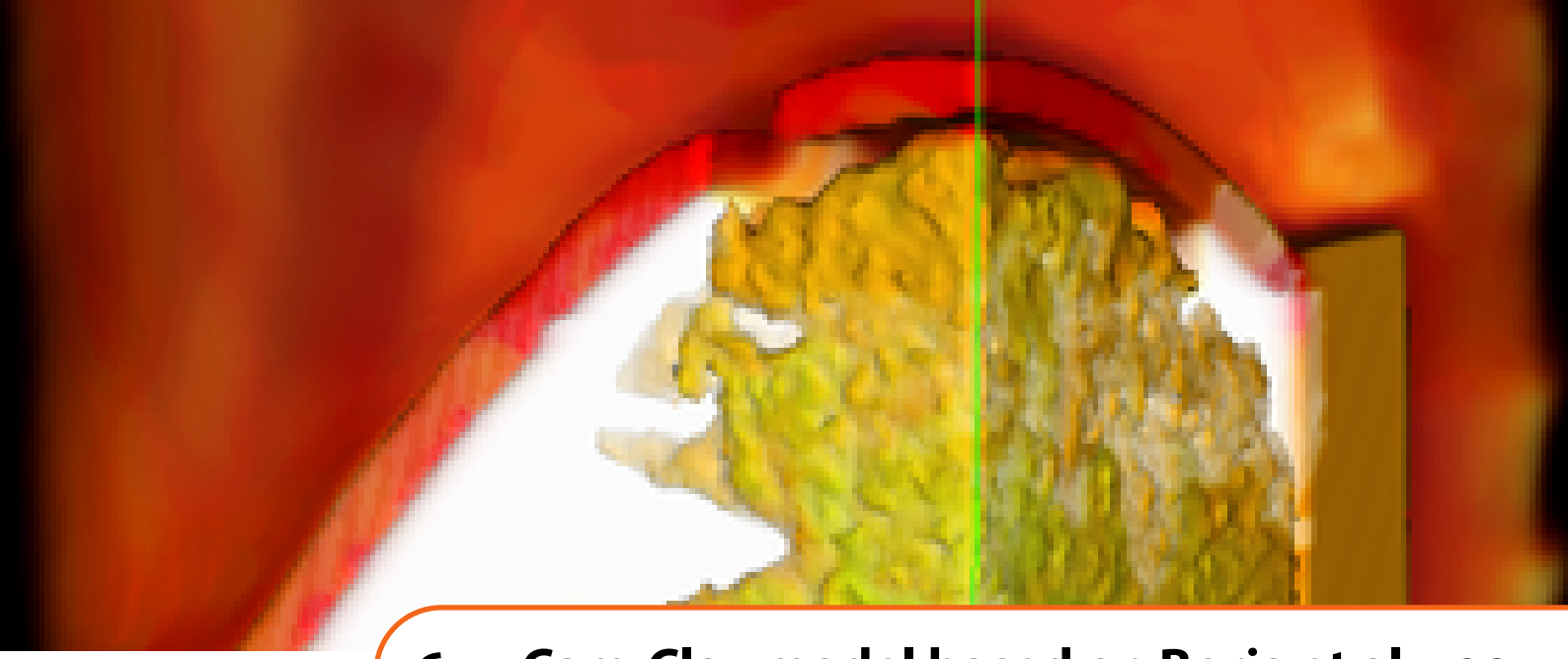
$$\dot{\hat{\boldsymbol{\beta}}} = \dot{\gamma} \left[\frac{2}{3}\, H_1\, \boldsymbol{r} - H_2\, \hat{\boldsymbol{\beta}}\, \|\boldsymbol{r}\| \right] . \tag{5.109}$$

Therefore, for this model,

$$\boxed{\begin{aligned} \boldsymbol{r} &= \frac{3\,\boldsymbol{\xi}}{\sigma_y^2} + \frac{q_1\,q_2\,\phi^*}{\sigma_y}\,\sinh\left(\frac{q_2\,\mathrm{tr}(\boldsymbol{\sigma})}{2\,\sigma_y}\right)\boldsymbol{1} \\ h^\alpha &= \frac{(\boldsymbol{\sigma}-\boldsymbol{\beta}):\boldsymbol{r}}{(1-\phi)\,\sigma_y} \\ h^\phi &= (1-\phi)\,\mathrm{tr}(\boldsymbol{r}) + A\,\frac{(\boldsymbol{\sigma}-\hat{\boldsymbol{\beta}}):\boldsymbol{r}}{(1-\phi)\,\sigma_y} \\ \boldsymbol{h}^\beta &= \frac{2}{3}\,H_1\,\boldsymbol{r} - H_2\,\hat{\boldsymbol{\beta}}\;\|\boldsymbol{r}\| \end{aligned}} \tag{5.110}$$

The other derivatives of the yield function that we need are

$$\begin{aligned} \frac{\partial f}{\partial \boldsymbol{\xi}} &= \frac{3\,\boldsymbol{\xi}}{\sigma_y^2} \\ \frac{\partial f}{\partial \varepsilon^p} &= \frac{\partial f}{\partial \sigma_y}\,\frac{\partial \sigma_y}{\partial \varepsilon^p} = -\left[\frac{3\,\boldsymbol{\xi}:\boldsymbol{\xi}}{\sigma_y^3} + \frac{q_1\,q_2\,\phi^*\,\mathrm{tr}(\boldsymbol{\sigma})}{\sigma_y^2}\,\sinh\left(\frac{q_2\,\mathrm{tr}(\boldsymbol{\sigma})}{2\,\sigma_y}\right)\right]\frac{\partial \sigma_y}{\partial \varepsilon^p} \\ \frac{\partial f}{\partial \phi} &= 2\,q_1\,\frac{d\phi^*}{d\phi}\,\cosh\left(\frac{q_2\,\mathrm{tr}(\sigma)}{2\,\sigma_y}\right) - 2\,q_3\,\phi^*\,\frac{d\phi^*}{d\phi}\,. \end{aligned} \tag{5.111}$$



6 — Cam-Clay model based on Borja et al. 1997

6.1 Introduction

The Cam-clay plasticity model and its modified ellipsoidal version [60–63] is widely considered to be an accurate model for the prediction of the compressive and shear behavior of clays. The Borja model [64–67] extends the original Cam-clay model to large deformations and uses a hyperelastic model and large strain elastic-plastic kinematics.

The Borja Cam-clay model and its implementation in Vaango are discussed in this chapter.

6.2 Quantities that are needed in a Vaango implementation

The implementation of a hyperelastic-plastic model in Vaango typically (but not always) involves the following:

1. an elasticity model factory that creates an elasticity model that provides the simulation with a pressure and a deviatoric stress for a given (elastic) deformation gradient.
2. a plasticity model factory that creates:
   (a) a yield condition factory that compute the yield function for a given stress and internal variable state,
   (b) a flow rule factory that gives the value of the plastic potential for a given state of stress/internal variables. The flow rule factory and yield condition factory are typically assumed to be identical (i.e., plastic flow is associated),
   (c) an internal variable factory that is used to update internal variables and compute hardening moduli.

The models returned by the various factories for Borja cam-clay are discussed below.

6.2.1 Elasticity

The elastic strain energy density in Borja's model has the form

$$W(\varepsilon_v^e, \varepsilon_s^e) = W_{\text{vol}}(\varepsilon_v^e) + W_{\text{dev}}(\varepsilon_v^e, Ve_s^e)$$

where

$$W_{\text{vol}}(\varepsilon_v^e) = -p_o\tilde{\kappa}\,\exp\left(-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right)$$
$$W_{\text{dev}}(\varepsilon_v^e, \varepsilon_s^e) = \tfrac{3}{2}\,\mu\,(\varepsilon_s^e)^2$$

where ε_{vo}^e is the volumetric strain corresponding to a mean normal compressive stress p_o (tension positive), $\tilde{\kappa}$ is the elastic compressibility index, and the shear modulus is given by

$$\mu = \mu_o + \frac{\alpha}{\tilde{\kappa}}\,W_{\text{vol}}(\varepsilon_v^e) = \mu_o - \alpha p_o\,\exp\left(-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right) = \mu_o - \mu_{\text{vol}}\,.$$

The parameter α determines the extent of coupling between the volumetric and deviatoric responses. For consistency with isotropic elasticity, Rebecca Brannon suggests that $\alpha = 0$ (citation?).

The stress invariants p and q are defined as

$$p = \frac{\partial W}{\partial \varepsilon_v^e} = p_o\left[1 + \tfrac{3}{2}\,\frac{\alpha}{\tilde{\kappa}}\,(\varepsilon_s^e)^2\right]\exp\left(-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right) = p_o\,\beta\,\exp\left(-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right)$$
$$q = \frac{\partial W}{\partial \varepsilon_s^e} = 3\left[\mu_o - \alpha p_o \exp\left(-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right)\right]\varepsilon_s^e = 3\mu\,\varepsilon_s^e\,.$$

The derivatives of the stress invariants are

$$\frac{\partial p}{\partial \varepsilon_v^e} = -\frac{p_o}{\tilde{\kappa}}\left[1 + \tfrac{3}{2}\,\frac{\alpha}{\tilde{\kappa}}\,(\varepsilon_s^e)^2\right]\exp\left(-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right) = -\frac{p}{\tilde{\kappa}}$$
$$\frac{\partial p}{\partial \varepsilon_s^e} = \frac{\partial q}{\partial \varepsilon_v^e} = \frac{3\alpha p_o \varepsilon_s^e}{\tilde{\kappa}}\,\exp\left(-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right) = \frac{3\alpha p}{\beta\tilde{\kappa}}\,\varepsilon_s^e = \frac{3\mu_{\text{vol}}}{\tilde{\kappa}}\,\varepsilon_s^e$$
$$\frac{\partial q}{\partial \varepsilon_s^e} = 3\left[\mu_o - \alpha p_o\,\exp\left(-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right)\right] = 3\mu\,.$$

6.2.2 Plasticity

For plasticity we use a Cam-Clay yield function of the form

$$f = \left(\frac{q}{M}\right)^2 + p(p - p_c)$$

where M is the slope of the critical state line and the consolidation pressure p_c is an internal variable that evolves according to

$$\frac{1}{p_c}\,\frac{dp_c}{dt} = \frac{1}{\tilde{\lambda} - \tilde{\kappa}}\,\frac{d\varepsilon_v^p}{dt}\,.$$

The derivatives of f that are of interest are

$$\frac{\partial f}{\partial p} = 2p - p_c$$
$$\frac{\partial f}{\partial q} = \frac{2q}{M^2}\,.$$

If we integrate the equation for p_c from t_n to t_{n+1}, we can show that

$$(p_c)_{n+1} = (p_c)_n \exp\left[\frac{(\varepsilon_v^e)_{\text{trial}} - (\varepsilon_v^e)_{n+1}}{\tilde{\lambda} - \tilde{\kappa}}\right].$$

The derivative of p_c that is of interest is

$$\frac{\partial p_c}{\partial (\varepsilon_v^e)_{n+1}} = -\frac{(p_c)_n}{\tilde{\lambda} - \tilde{\kappa}}\,\exp\left[\frac{(\varepsilon_v^e)_{\text{trial}} - (\varepsilon_v^e)_{n+1}}{\tilde{\lambda} - \tilde{\kappa}}\right].$$

6.3 Stress update based Rich Reguiero's notes

The volumetric and deviatoric components of the elastic strain $\boldsymbol{\epsilon}^e$ are defined as follows:

$$\boldsymbol{e}^e = \boldsymbol{\epsilon}^e - \tfrac{1}{3}\varepsilon_v^e\,\boldsymbol{1} = \boldsymbol{\epsilon}^e - \tfrac{1}{3}\mathrm{tr}(\boldsymbol{\epsilon}^e)\,\boldsymbol{1} \quad \text{and} \quad \varepsilon_s^e = \sqrt{\tfrac{2}{3}}\,\|\boldsymbol{e}^e\| = \sqrt{\tfrac{2}{3}}\sqrt{\boldsymbol{e}^e : \boldsymbol{e}^e}\,.$$

The stress tensor is decomposed into a volumetric and a deviatoric component

$$\boldsymbol{\sigma} = p\,\boldsymbol{1} + \sqrt{\tfrac{2}{3}}\,q\,\boldsymbol{n} \quad \text{with} \quad \boldsymbol{n} = \frac{\boldsymbol{e}^e}{\|\boldsymbol{e}^e\|} = \sqrt{\tfrac{2}{3}}\,\frac{\boldsymbol{e}^e}{\varepsilon_s^e}\,.$$

The models used to determine p and q are

$$p = p_o\beta\exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right] \quad \text{with} \quad \beta = 1 + \tfrac{3}{2}\,\frac{\alpha}{\tilde{\kappa}}\,(\varepsilon_s^e)^2$$
$$q = 3\mu\varepsilon_s^e\,.$$

The strains are updated using

$$\boldsymbol{\epsilon}^e = \boldsymbol{\epsilon}^e_{\text{trial}} - \Delta\gamma\,\frac{\partial f}{\partial \boldsymbol{\sigma}} \quad \text{where} \quad \boldsymbol{\epsilon}^e_{\text{trial}} = \boldsymbol{\epsilon}^e_n + \Delta\boldsymbol{\epsilon} = \boldsymbol{\epsilon}^e_n + (\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_n)\,.$$

Remark 1: The interface with MPMICE, among other things in Vaango, requires the computation of the quantity dp/dJ. Since J does not appear in the above equation we proceed as explained below.

$$J = \det(\boldsymbol{F}) = \det(\boldsymbol{1} + \nabla_o\mathbf{u}) = \det(\boldsymbol{1} + \boldsymbol{\epsilon})$$
$$= 1 + \mathrm{tr}\boldsymbol{\epsilon} + \tfrac{1}{2}\left[(\mathrm{tr}\boldsymbol{\epsilon})^2 - \mathrm{tr}(\boldsymbol{\epsilon}^2)\right] + \det(\boldsymbol{\epsilon})\,. \quad = 1 + \varepsilon_v + \tfrac{1}{2}\left[\varepsilon_v^2 - \mathrm{tr}(\boldsymbol{\epsilon}^2)\right] + \det(\boldsymbol{\epsilon})\,.$$

Also,

$$J = \frac{\rho_o}{\rho} = \frac{V}{V_o} \quad \text{and} \quad \varepsilon_v = \frac{V - V_o}{V_o} = \frac{V}{V_o} - 1 = J - 1\,.$$

We use the relation $J = 1 + \varepsilon_v$ while keeping in mind that this is *true only for infinitesimal strains and plastic incompressibility* for which ε_v^2, $\mathrm{tr}(\boldsymbol{\epsilon}^2)$, and $\det(\boldsymbol{\epsilon})$ are zero. Under these conditions

$$\frac{\partial p}{\partial J} = \frac{\partial p}{\partial \varepsilon_v}\,\frac{\partial \varepsilon_v}{\partial J} = \frac{\partial p}{\partial \varepsilon_v} \quad \text{and} \quad \frac{\partial p}{\partial \rho} = \frac{\partial p}{\partial \varepsilon_v}\,\frac{\partial \varepsilon_v}{\partial J}\,\frac{\partial J}{\partial \rho} = -\frac{J}{\rho}\,\frac{\partial p}{\partial \varepsilon_v}\,.$$

Remark 2: MPMICE also needs the density at a given pressure. For the Borja model, with $\varepsilon_v = J - 1 = \rho_o/\rho - 1$, we have

$$\rho = \rho_o\left[1 + \varepsilon_{vo} + \tilde{\kappa}\ln\left(\frac{p}{p_o\beta}\right)\right]^{-1}\,.$$

Remark 3: The quantity q is related to the deviatoric part of the Cauchy stress, $\boldsymbol{s}$ as follows:

$$q = \sqrt{3J_2} \quad \text{where} \quad J_2 = \tfrac{1}{2}\,\boldsymbol{s} : \boldsymbol{s}\,.$$

The shear modulus relates the deviatoric stress $\boldsymbol{s}$ to the deviatoric strain $\boldsymbol{e}^e$. We assume a relation of the form

$$\boldsymbol{s} = 2\mu\boldsymbol{e}^e\,.$$

Note that the above relation assumes a linear elastic type behavior. Then we get the Borja shear model:

$$q = \sqrt{\tfrac{3}{2}\,\boldsymbol{s} : \boldsymbol{s}} = \sqrt{\tfrac{3}{2}}\,(2\mu)\,\sqrt{\boldsymbol{e}^e : \boldsymbol{e}^e} = \sqrt{\tfrac{3}{2}}\,(2\mu)\,\sqrt{\tfrac{3}{2}}\,\varepsilon_s^e = 3\mu\varepsilon_s^e\,.$$

6.3.1 Elastic-plastic stress update

For elasto-plasticity we start with a yield function of the form

$$f = \left(\frac{q}{M}\right)^2 + p(p - p_c) \leq 0 \quad \text{where} \quad \frac{1}{p_c}\frac{dp_c}{dt} = \frac{1}{\tilde{\lambda} - \tilde{\kappa}}\frac{d\varepsilon_v^p}{dt}.$$

Integrating the ODE for p_c with the initial condition $p_c(t_n) = (p_c)_n$, at $t = t_{n+1}$,

$$(p_c)_{n+1} = (p_c)_n \exp\left[\frac{(\varepsilon_v^p)_{n+1} - (\varepsilon_v^p)_n}{\tilde{\lambda} - \tilde{\kappa}}\right].$$

From the additive decomposition of the strain into elastic and plastic parts, and if the elastic trial strain is defined as

$$(\varepsilon_v^e)_{\text{trial}} := (\varepsilon_v^e)_n + \Delta\varepsilon_v$$

we have

$$\varepsilon_v^p = \varepsilon_v - \varepsilon_v^e \quad \Longrightarrow \quad (\varepsilon_v^p)_{n+1} - (\varepsilon_v^p)_n = (\varepsilon_v)_{n+1} - (\varepsilon_v^e)_{n+1} - (\varepsilon_v)_n + (\varepsilon_v^e)_n = \Delta\varepsilon_v + (\varepsilon_v^e)_n - (\varepsilon_v^e)_{n+1} = (\varepsilon_v^e)_{\text{trial}} - (\varepsilon_v^e)_{n+1}.$$

Therefore we can write

$$(p_c)_{n+1} = (p_c)_n \exp\left[\frac{(\varepsilon_v^e)_{\text{trial}} - (\varepsilon_v^e)_{n+1}}{\tilde{\lambda} - \tilde{\kappa}}\right].$$

The flow rule is assumed to be given by

$$\frac{\partial \boldsymbol{\epsilon}^p}{\partial t} = \gamma \frac{\partial f}{\partial \boldsymbol{\sigma}}.$$

Integration of the PDE with backward Euler gives

$$\boldsymbol{\epsilon}_{n+1}^p = \boldsymbol{\epsilon}_n^p + \Delta t\, \gamma_{n+1} \left[\frac{\partial f}{\partial \boldsymbol{\sigma}}\right]_{n+1} = \boldsymbol{\epsilon}_n^p + \Delta\gamma \left[\frac{\partial f}{\partial \boldsymbol{\sigma}}\right]_{n+1}.$$

This equation can be expressed in terms of the trial elastic strain as follows.

$$\boldsymbol{\epsilon}_{n+1} - \boldsymbol{\epsilon}_{n+1}^e = \boldsymbol{\epsilon}_n - \boldsymbol{\epsilon}_n^e + \Delta\gamma \left[\frac{\partial f}{\partial \boldsymbol{\sigma}}\right]_{n+1}$$

or

$$\boldsymbol{\epsilon}_{n+1}^e = \Delta\boldsymbol{\epsilon} + \boldsymbol{\epsilon}_n^e - \Delta\gamma \left[\frac{\partial f}{\partial \boldsymbol{\sigma}}\right]_{n+1} = \boldsymbol{\epsilon}_{\text{trial}}^e - \Delta\gamma \left[\frac{\partial f}{\partial \boldsymbol{\sigma}}\right]_{n+1}.$$

In terms of the volumetric and deviatoric components

$$(\varepsilon_v^e)_{n+1} = \text{tr}(\boldsymbol{\epsilon}_{n+1}^e) = \text{tr}(\boldsymbol{\epsilon}_{\text{trial}}^e) - \Delta\gamma\, \text{tr}\left[\frac{\partial f}{\partial \boldsymbol{\sigma}}\right]_{n+1} = (\varepsilon_v^e)_{\text{trial}} - \Delta\gamma\, \text{tr}\left[\frac{\partial f}{\partial \boldsymbol{\sigma}}\right]_{n+1}$$

and

$$\boldsymbol{e}_{n+1}^e = \boldsymbol{e}_{\text{trial}}^e - \Delta\gamma \left[\left(\frac{\partial f}{\partial \boldsymbol{\sigma}}\right)_{n+1} - \tfrac{1}{3}\,\text{tr}\left(\frac{\partial f}{\partial \boldsymbol{\sigma}}\right)_{n+1} \boldsymbol{1}\right].$$

With $\boldsymbol{s} = \boldsymbol{\sigma} - p\boldsymbol{1}$, we have

$$\frac{\partial f}{\partial \boldsymbol{\sigma}} = \frac{\partial f}{\partial \boldsymbol{s}} : \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{\sigma}} + \frac{\partial f}{\partial p}\frac{\partial p}{\partial \boldsymbol{\sigma}} = \frac{\partial f}{\partial \boldsymbol{s}} : \left[\mathbf{I}^{(s)} - \tfrac{1}{3}\boldsymbol{1} \otimes \boldsymbol{1}\right] + \frac{\partial f}{\partial p}\boldsymbol{1} = \frac{\partial f}{\partial \boldsymbol{s}} - \tfrac{1}{3}\,\text{tr}\left[\frac{\partial f}{\partial \boldsymbol{s}}\right]\boldsymbol{1} + \frac{\partial f}{\partial p}\boldsymbol{1}$$

and

$$\tfrac{1}{3}\,\mathrm{tr}\left[\frac{\partial f}{\partial \boldsymbol{\sigma}}\right]\boldsymbol{1} = \tfrac{1}{3}\left(\mathrm{tr}\left[\frac{\partial f}{\partial \boldsymbol{s}}\right] - \mathrm{tr}\left[\frac{\partial f}{\partial \boldsymbol{s}}\right] + 3\frac{\partial f}{\partial p}\right)\boldsymbol{1} = \frac{\partial f}{\partial p}\,\boldsymbol{1}\,.$$

Remark 4: Note that, because $\boldsymbol{\sigma} = \boldsymbol{\sigma}(p, q, p_c)$ the chain rule should contain a contribution from p_c:

$$\frac{\partial f}{\partial \boldsymbol{\sigma}} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial \boldsymbol{\sigma}} + \frac{\partial f}{\partial p}\frac{\partial p}{\partial \boldsymbol{\sigma}} + \frac{\partial f}{\partial p_c}\frac{\partial p_c}{\partial \boldsymbol{\sigma}}\,.$$

However, the Borja implementation does not consider that extra term. Also note that for the present model

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(p(\varepsilon_v^e, \varepsilon_s^e, \varepsilon_v^p, \varepsilon_s^p), \boldsymbol{s}(\varepsilon_v^e, \varepsilon_s^e, \varepsilon_v^p, \varepsilon_s^p), p_c(\varepsilon_v^p))$$

Therefore, for situations where $\mathrm{tr}(\partial f/\partial \boldsymbol{s}) = \mathbf{0}$, we have

$$\frac{\partial f}{\partial \boldsymbol{\sigma}} - \tfrac{1}{3}\,\mathrm{tr}\left[\frac{\partial f}{\partial \boldsymbol{\sigma}}\right]\boldsymbol{1} = \frac{\partial f}{\partial \boldsymbol{s}} - \tfrac{1}{3}\,\mathrm{tr}\left[\frac{\partial f}{\partial \boldsymbol{s}}\right]\boldsymbol{1} = \frac{\partial f}{\partial \boldsymbol{s}}\,.$$

The deviatoric strain update can be written as

$$\boldsymbol{e}^e_{n+1} = \boldsymbol{e}^e_{\text{trial}} - \Delta\gamma\left(\frac{\partial f}{\partial \boldsymbol{s}}\right)_{n+1}$$

and the shear invariant update is

$$(\varepsilon_s^e)_{n+1} = \sqrt{\tfrac{2}{3}}\,\sqrt{\boldsymbol{e}^e_{n+1} : \boldsymbol{e}^e_{n+1}} = \sqrt{\tfrac{2}{3}}\,\sqrt{\boldsymbol{e}^e_{\text{trial}} : \boldsymbol{e}^e_{\text{trial}} - 2\Delta\gamma\left[\frac{\partial f}{\partial \boldsymbol{s}}\right]_{n+1} : \boldsymbol{e}^e_{\text{trial}} + (\Delta\gamma)^2\left[\frac{\partial f}{\partial \boldsymbol{s}}\right]_{n+1} : \left[\frac{\partial f}{\partial \boldsymbol{s}}\right]_{n+1}}$$

The derivative of f can be found using the chain rule (for smooth f):

$$\frac{\partial f}{\partial \boldsymbol{\sigma}} = \frac{\partial f}{\partial p}\frac{\partial p}{\partial \boldsymbol{\sigma}} + \frac{\partial f}{\partial q}\frac{\partial q}{\partial \boldsymbol{\sigma}} = (2p - p_c)\frac{\partial p}{\partial \boldsymbol{\sigma}} + \frac{2q}{M^2}\frac{\partial q}{\partial \boldsymbol{\sigma}}\,.$$

Now, with $p = 1/3\,\mathrm{tr}(\boldsymbol{\sigma})$ and $q = \sqrt{3/2\,\mathbf{s} : \mathbf{s}}$, we have

$$\frac{\partial p}{\partial \boldsymbol{\sigma}} = \frac{\partial}{\partial \boldsymbol{\sigma}}\left[\tfrac{1}{3}\,\mathrm{tr}(\boldsymbol{\sigma})\right] = \tfrac{1}{3}\,\boldsymbol{1}$$
$$\frac{\partial q}{\partial \boldsymbol{\sigma}} = \frac{\partial}{\partial \boldsymbol{\sigma}}\left[\sqrt{\tfrac{3}{2}\,\boldsymbol{s} : \boldsymbol{s}}\right] = \sqrt{\tfrac{3}{2}}\,\frac{1}{\sqrt{\boldsymbol{s} : \boldsymbol{s}}}\,\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{\sigma}} : \boldsymbol{s} = \sqrt{\tfrac{3}{2}}\,\frac{1}{\|\boldsymbol{s}\|}\left[\mathbf{I}^{(s)} - \tfrac{1}{3}\boldsymbol{1}\otimes\boldsymbol{1}\right] : \boldsymbol{s} = \sqrt{\tfrac{3}{2}}\,\frac{\boldsymbol{s}}{\|\boldsymbol{s}\|}\,.$$

Therefore,

$$\frac{\partial f}{\partial \boldsymbol{\sigma}} = \frac{2p - p_c}{3}\,\boldsymbol{1} + \sqrt{\tfrac{3}{2}}\,\frac{2q}{M^2}\,\frac{\boldsymbol{s}}{\|\boldsymbol{s}\|}\,.$$

Recall that

$$\boldsymbol{\sigma} = p\,\boldsymbol{1} + \sqrt{\tfrac{2}{3}}\,q\,\boldsymbol{n} = p\,\boldsymbol{1} + \boldsymbol{s}\,.$$

Therefore,

$$\boldsymbol{s} = \sqrt{\tfrac{2}{3}}\,q\,\boldsymbol{n} \quad \text{and} \quad \|\boldsymbol{s}\| = \sqrt{\boldsymbol{s} : \boldsymbol{s}} = \sqrt{\tfrac{2}{3}\,q^2\,\boldsymbol{n} : \boldsymbol{n}} = \sqrt{\tfrac{2}{3}\,q^2\,\frac{\boldsymbol{e}^e : \boldsymbol{e}^e}{\|\boldsymbol{e}^e\|^2}} = \sqrt{\tfrac{2}{3}\,q^2} = \sqrt{\tfrac{2}{3}}\,q\,.$$

So we can write

$$\frac{\partial f}{\partial \boldsymbol{\sigma}} = \frac{2p - p_c}{3}\,\boldsymbol{1} + \sqrt{\tfrac{3}{2}}\,\frac{2q}{M^2}\,\boldsymbol{n}\,. \tag{6.1}$$

Using the above relation we have

$$\frac{\partial f}{\partial p} = \tfrac{1}{3}\,\mathrm{tr}\left[\frac{\partial f}{\partial \boldsymbol{\sigma}}\right] = 2p - p_c \quad \text{and} \quad \frac{\partial f}{\partial \boldsymbol{s}} = \frac{\partial f}{\partial \boldsymbol{\sigma}} - \frac{\partial f}{\partial p}\boldsymbol{1} = \sqrt{\tfrac{3}{2}}\,\frac{2q}{M^2}\,\boldsymbol{n}\,.$$

The strain updates can now be written as

$$\begin{aligned}
(\varepsilon_v^e)_{n+1} &= (\varepsilon_v^e)_{\text{trial}} - \Delta\gamma\,[2p_{n+1} - (p_c)_{n+1}] \\
\boldsymbol{e}^e_{n+1} &= \boldsymbol{e}^e_{\text{trial}} - \sqrt{\tfrac{3}{2}}\,\Delta\gamma\left(\frac{2q_{n+1}}{M^2_{n+1}}\right)\boldsymbol{n}_{n+1} \\
(\varepsilon_s^e)_{n+1} &= \sqrt{\tfrac{2}{3}}\sqrt{\boldsymbol{e}^e_{\text{trial}} : \boldsymbol{e}^e_{\text{trial}} - \sqrt{6}\,(\Delta\gamma)^2\left(\frac{2q_{n+1}}{M^2_{n+1}}\right)\boldsymbol{n}_{n+1} : \boldsymbol{e}^e_{\text{trial}} + \tfrac{3}{2}\,(\Delta\gamma)^4\left(\frac{2q_{n+1}}{M^2_{n+1}}\right)^2}\,.
\end{aligned}$$

From the second equation above,

$$\boldsymbol{n}_{n+1} : \boldsymbol{e}^e_{\text{trial}} = \boldsymbol{n}_{n+1} : \boldsymbol{e}^e_{n+1} + \sqrt{\tfrac{3}{2}}\,\Delta\gamma\left(\frac{2q_{n+1}}{M^2_{n+1}}\right)\boldsymbol{n}_{n+1} : \boldsymbol{n}_{n+1} = \frac{\boldsymbol{e}^e_{n+1} : \boldsymbol{e}^e_{n+1}}{\|\boldsymbol{e}^e_{n+1}\|} + \sqrt{\tfrac{3}{2}}\,\Delta\gamma\left(\frac{2q_{n+1}}{M^2_{n+1}}\right) = \|\boldsymbol{e}^e_{n+1}\| + \sqrt{\tfrac{3}{2}}\,\Delta\gamma\left(\frac{2q_{n+1}}{M^2_{n+1}}\right).$$

Also notice that

$$\boldsymbol{e}^e_{\text{trial}} : \boldsymbol{e}^e_{\text{trial}} = \boldsymbol{e}^e_{n+1} : \boldsymbol{e}^e_{n+1} + 2\sqrt{\tfrac{3}{2}}\,\Delta\gamma\left(\frac{2q_{n+1}}{M^2_{n+1}}\right)\boldsymbol{e}^e_{n+1} : \boldsymbol{n}_{n+1} + \left[\sqrt{\tfrac{3}{2}}\,\Delta\gamma\left(\frac{2q_{n+1}}{M^2_{n+1}}\right)\right]^2$$

or,

$$\|\boldsymbol{e}^e_{\text{trial}}\|^2 = \left[\|\boldsymbol{e}^e_{n+1}\| + \sqrt{\tfrac{3}{2}}\,\Delta\gamma\left(\frac{2q_{n+1}}{M^2_{n+1}}\right)\right]^2.$$

Therefore,

$$\boldsymbol{n}_{n+1} : \boldsymbol{e}^e_{\text{trial}} = \|\boldsymbol{e}^e_{\text{trial}}\|$$

and we have

$$(\varepsilon_s^e)_{n+1} = \sqrt{\tfrac{2}{3}}\sqrt{\|\boldsymbol{e}^e_{\text{trial}}\|^2 - \sqrt{6}\,(\Delta\gamma)^2\left(\frac{2q_{n+1}}{M^2_{n+1}}\right)\|\boldsymbol{e}^e_{\text{trial}}\| + \tfrac{3}{2}\,(\Delta\gamma)^4\left(\frac{2q_{n+1}}{M^2_{n+1}}\right)^2} = \sqrt{\tfrac{2}{3}}\,\|\boldsymbol{e}^e_{\text{trial}}\| - \Delta\gamma\left(\frac{2q_{n+1}}{M^2_{n+1}}\right).$$

The elastic strain can therefore be updated using

$$\begin{aligned}
(\varepsilon_v^e)_{n+1} &= (\varepsilon_v^e)_{\text{trial}} - \Delta\gamma\,[2p_{n+1} - (p_c)_{n+1}] \\
(\varepsilon_s^e)_{n+1} &= (\varepsilon_s^e)_{\text{trial}} - \Delta\gamma\left(\frac{2q_{n+1}}{M^2_{n+1}}\right).
\end{aligned}$$

The consistency condition is needed to close the above equations

$$f = \left(\frac{q_{n+1}}{M}\right)^2 + p_{n+1}[p_{n+1} - (p_c)_{n+1}] = 0\,.$$

The unknowns are $(\varepsilon_v^e)_{n+1}$, $(\varepsilon_s^e)_{n+1}$ and $\Delta\gamma$. Note that we can express the three equations as

$$\begin{aligned}
(\varepsilon_v^e)_{n+1} &= (\varepsilon_v^e)_{\text{trial}} - \Delta\gamma\left[\frac{\partial f}{\partial p}\right]_{n+1} \\
(\varepsilon_s^e)_{n+1} &= (\varepsilon_s^e)_{\text{trial}} - \Delta\gamma\left[\frac{\partial f}{\partial q}\right]_{n+1} \\
f_{n+1} &= 0\,.
\end{aligned} \tag{6.2}$$

6.3.2 Newton iterations

The three nonlinear equations in the three unknowns can be solved using Newton iterations for smooth yield functions. Let us define the residual as

$$\underline{\underline{r}}(\underline{\underline{x}}) = \begin{bmatrix} (\varepsilon_v^e)_{n+1} - (\varepsilon_v^e)_{\text{trial}} + \Delta\gamma \left[\dfrac{\partial f}{\partial p}\right]_{n+1} \\ (\varepsilon_s^e)_{n+1} - (\varepsilon_s^e)_{\text{trial}} + \Delta\gamma \left[\dfrac{\partial f}{\partial q}\right]_{n+1} \\ f_{n+1} \end{bmatrix} =: \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} \quad \text{where} \quad \underline{\underline{x}} = \begin{bmatrix} (\varepsilon_v^e)_{n+1} \\ (\varepsilon_s^e)_{n+1} \\ f_{n+1} \end{bmatrix} =: \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}.$$

The Newton root finding algorithm is :

Require: $\underline{\underline{x}}^0$

$k \leftarrow 0$

while $\underline{\underline{r}}(\underline{\underline{x}}^k) \neq 0$ **do**

$$\underline{\underline{x}}^{k+1} \Leftarrow \underline{\underline{x}}^k - \left[\left(\frac{\partial \underline{\underline{r}}}{\partial \underline{\underline{x}}}\right)^{-1}\right]_{\underline{\underline{x}}^k} \cdot \underline{\underline{r}}(\underline{\underline{x}}^k)$$

$k \leftarrow k + 1$

end while

To code the algorithm we have to find the derivatives of the residual with respect to the primary variables. Let's do the terms one by one. For the first row,

$$\frac{\partial r_1}{\partial x_1} = \frac{\partial}{\partial \varepsilon_v^e}\left[\varepsilon_v^e - (\varepsilon_v^e)_{\text{trial}} + \Delta\gamma\,(2p - p_c)\right] = 1 + \Delta\gamma\left(2\frac{\partial p}{\partial \varepsilon_v^e} - \frac{\partial p_c}{\partial \varepsilon_v^e}\right)$$
$$\frac{\partial r_1}{\partial x_2} = \frac{\partial}{\partial \varepsilon_s^e}\left[\varepsilon_v^e - (\varepsilon_v^e)_{\text{trial}} + \Delta\gamma\,(2p - p_c)\right] = 2\Delta\gamma\frac{\partial p}{\partial \varepsilon_s^e}$$
$$\frac{\partial r_1}{\partial x_3} = \frac{\partial}{\partial \Delta\gamma}\left[\varepsilon_v^e - (\varepsilon_v^e)_{\text{trial}} + \Delta\gamma\,(2p - p_c)\right] = 2p - p_c = \frac{\partial f}{\partial p}$$

where

$$\frac{\partial p}{\partial \varepsilon_v^e} = -\frac{p_o\,\beta}{\tilde{\kappa}}\exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right] = \frac{p}{\tilde{\kappa}} \quad , \quad \frac{\partial p_c}{\partial \varepsilon_v^e} = \frac{(p_c)_n}{\tilde{\kappa} - \tilde{\lambda}}\exp\left[\frac{\varepsilon_v^e - (\varepsilon_v^e)_{\text{trial}}}{\tilde{\kappa} - \tilde{\lambda}}\right] \quad \text{and}$$
$$\frac{\partial p}{\partial \varepsilon_s^e} = \frac{3\,p_o\,\alpha\,\varepsilon_s^e}{\tilde{\kappa}}\exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right].$$

For the second row,

$$\frac{\partial r_2}{\partial x_1} = \frac{\partial}{\partial \varepsilon_v^e}\left[\varepsilon_s^e - (\varepsilon_s^e)_{\text{trial}} + \Delta\gamma\,\frac{2q}{M^2}\right] = \frac{2\Delta\gamma}{M^2}\frac{\partial q}{\partial \varepsilon_v^e}$$
$$\frac{\partial r_2}{\partial x_2} = \frac{\partial}{\partial \varepsilon_s^e}\left[\varepsilon_s^e - (\varepsilon_s^e)_{\text{trial}} + \Delta\gamma\,\frac{2q}{M^2}\right] = 1 + \frac{2\Delta\gamma}{M^2}\frac{\partial q}{\partial \varepsilon_s^e}$$
$$\frac{\partial r_2}{\partial x_3} = \frac{\partial}{\partial \Delta\gamma}\left[\varepsilon_s^e - (\varepsilon_s^e)_{\text{trial}} + \Delta\gamma\,\frac{2q}{M^2}\right] = \frac{2q}{M^2} = \frac{\partial f}{\partial q}$$

where

$$\frac{\partial q}{\partial \varepsilon_v^e} = -\frac{3p_o\,\alpha\,\varepsilon_s^e}{\tilde{\kappa}}\exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right] = \frac{\partial p}{\partial \varepsilon_s^e} \quad \text{and} \quad \frac{\partial q}{\partial \varepsilon_s^e} = 3\mu_o + 3p_o\,\alpha\exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right] = 3\mu\,.$$

For the third row,

$$\frac{\partial r_3}{\partial x_1} = \frac{\partial}{\partial \varepsilon_v^e}\left[\frac{q^2}{M^2} + p\,(p - p_c)\right] = \frac{2q}{M^2}\,\frac{\partial q}{\partial \varepsilon_v^e} + (2p - p_c)\,\frac{\partial p}{\partial \varepsilon_v^e} - p\,\frac{\partial p_c}{\partial \varepsilon_v^e} = \frac{\partial f}{\partial q}\,\frac{\partial q}{\partial \varepsilon_v^e} + \frac{\partial f}{\partial p}\,\frac{\partial p}{\partial \varepsilon_v^e} - p\,\frac{\partial p_c}{\partial \varepsilon_v^e}$$
$$\frac{\partial r_3}{\partial x_2} = \frac{\partial}{\partial \varepsilon_s^e}\left[\frac{q^2}{M^2} + p\,(p - p_c)\right] = \frac{2q}{M^2}\,\frac{\partial q}{\partial \varepsilon_s^e} + (2p - p_c)\,\frac{\partial p}{\partial \varepsilon_s^e} = \frac{\partial f}{\partial q}\,\frac{\partial q}{\partial \varepsilon_s^e} + \frac{\partial f}{\partial p}\,\frac{\partial p}{\partial \varepsilon_s^e}$$
$$\frac{\partial r_3}{\partial x_3} = \frac{\partial}{\partial \Delta\gamma}\left[\frac{q^2}{M^2} + p\,(p - p_c)\right] = 0\,.$$

We have to invert a matrix in the Newton iteration process. Let us see whether we can make this quicker to do. The Jacobian matrix has the form

$$\frac{\partial \underline{\underline{r}}}{\partial \underline{\underline{x}}} = \begin{bmatrix} \frac{\partial r_1}{\partial x_1} & \frac{\partial r_1}{\partial x_2} & \frac{\partial r_1}{\partial x_3} \\ \frac{\partial r_2}{\partial x_1} & \frac{\partial r_2}{\partial x_2} & \frac{\partial r_2}{\partial x_3} \\ \frac{\partial r_3}{\partial x_1} & \frac{\partial r_3}{\partial x_2} & \frac{\partial r_3}{\partial x_3} \end{bmatrix} = \begin{bmatrix} \underline{\underline{A}} & \underline{\underline{B}} \\ \underline{\underline{C}} & 0 \end{bmatrix}$$

where

$$\underline{\underline{A}} = \begin{bmatrix} \frac{\partial r_1}{\partial x_1} & \frac{\partial r_1}{\partial x_2} \\ \frac{\partial r_2}{\partial x_1} & \frac{\partial r_2}{\partial x_2} \end{bmatrix}, \quad \underline{\underline{B}} = \begin{bmatrix} \frac{\partial r_1}{\partial x_3} \\ \frac{\partial r_2}{\partial x_3} \end{bmatrix}, \quad \text{and} \quad \underline{\underline{C}} = \begin{bmatrix} \frac{\partial r_3}{\partial x_1} & \frac{\partial r_3}{\partial x_2} \end{bmatrix}.$$

We can also break up the $\underline{\underline{x}}$ and $\underline{\underline{r}}$ matrices:

$$\Delta\underline{\underline{x}} = \underline{\underline{x}}^{k+1} - \underline{\underline{x}}^{k} = \begin{bmatrix} \Delta\underline{\underline{x}}^{vs} \\ \Delta x_3 \end{bmatrix}, \quad \underline{\underline{r}} = \begin{bmatrix} \underline{\underline{r}}^{vs} \\ r_3 \end{bmatrix} \quad \text{where} \quad \underline{\underline{r}}^{vs} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} \quad \text{and} \quad \Delta\underline{\underline{x}}^{vs} = \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix}.$$

Then

$$\begin{bmatrix} \Delta\underline{\underline{x}}^{vs} \\ \Delta x_3 \end{bmatrix} = -\begin{bmatrix} \underline{\underline{A}} & \underline{\underline{B}} \\ \underline{\underline{C}} & 0 \end{bmatrix}^{-1} \begin{bmatrix} \underline{\underline{r}}^{vs} \\ r_3 \end{bmatrix} \quad \Longrightarrow \quad \begin{bmatrix} \underline{\underline{A}} & \underline{\underline{B}} \\ \underline{\underline{C}} & 0 \end{bmatrix} \begin{bmatrix} \Delta\underline{\underline{x}}^{vs} \\ \Delta x_3 \end{bmatrix} = -\begin{bmatrix} \underline{\underline{r}}^{vs} \\ r_3 \end{bmatrix}$$

or

$$\underline{\underline{A}}\,\Delta\underline{\underline{x}}^{vs} + \underline{\underline{B}}\,\Delta x_3 = -\underline{\underline{r}}^{vs} \quad \text{and} \quad \underline{\underline{C}}\,\Delta\underline{\underline{x}}^{vs} = -r_3\,.$$

From the first equation above,

$$\Delta\underline{\underline{x}}^{vs} = -\underline{\underline{A}}^{-1}\,\underline{\underline{r}}^{vs} - \underline{\underline{A}}^{-1}\,\underline{\underline{B}}\,\Delta x_3\,.$$

Plugging in the second equation gives

$$r_3 = \underline{\underline{C}}\,\underline{\underline{A}}^{-1}\,\underline{\underline{r}}^{vs} + \underline{\underline{C}}\,\underline{\underline{A}}^{-1}\,\underline{\underline{B}}\,\Delta x_3\,.$$

Rearranging,

$$\Delta x_3 = x_3^{k+1} - x_3^{k} = \frac{-\underline{\underline{C}}\,\underline{\underline{A}}^{-1}\,\underline{\underline{r}}^{vs} + r_3}{\underline{\underline{C}}\,\underline{\underline{A}}^{-1}\,\underline{\underline{B}}}\,.$$

Using the above result,

$$\Delta\underline{\underline{x}}^{vs} = -\underline{\underline{A}}^{-1}\,\underline{\underline{r}}^{vs} - \underline{\underline{A}}^{-1}\,\underline{\underline{B}}\left(\frac{-\underline{\underline{C}}\,\underline{\underline{A}}^{-1}\,\underline{\underline{r}}^{vs} + r_3}{\underline{\underline{C}}\,\underline{\underline{A}}^{-1}\,\underline{\underline{B}}}\right).$$

We therefore have to invert only a 2×2 matrix.

6.3.3 Tangent calculation: elastic

We want to find the derivative of the stress with respect to the strain:

$$\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\epsilon}} = \boldsymbol{1} \otimes \frac{\partial p}{\partial \boldsymbol{\epsilon}} + \sqrt{\tfrac{2}{3}}\, \boldsymbol{n} \otimes \frac{\partial q}{\partial \boldsymbol{\epsilon}} + \sqrt{\tfrac{2}{3}}\, q\, \frac{\partial \boldsymbol{n}}{\partial \boldsymbol{\epsilon}} \,. \tag{6.3}$$

For the first term above,

$$\frac{\partial p}{\partial \boldsymbol{\epsilon}} = p_o \, \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right] \frac{\partial \beta}{\partial \boldsymbol{\epsilon}} - p_o \, \frac{\beta}{\tilde{\kappa}} \, \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right] \frac{\partial \varepsilon_v^e}{\partial \boldsymbol{\epsilon}} = p_o \, \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right] \left(\frac{\partial \beta}{\partial \boldsymbol{\epsilon}} - \frac{\beta}{\tilde{\kappa}} \, \frac{\partial \varepsilon_v^e}{\partial \boldsymbol{\epsilon}}\right) .$$

Now,

$$\frac{\partial \beta}{\partial \boldsymbol{\epsilon}} = \frac{3\alpha}{\tilde{\kappa}} \, \varepsilon_s^e \, \frac{\partial \varepsilon_s^e}{\partial \boldsymbol{\epsilon}} \,.$$

Therefore,

$$\frac{\partial p}{\partial \boldsymbol{\epsilon}} = \frac{p_o}{\tilde{\kappa}} \, \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right] \left(3\alpha \, \varepsilon_s^e \, \frac{\partial \varepsilon_s^e}{\partial \boldsymbol{\epsilon}} - \beta \, \frac{\partial \varepsilon_v^e}{\partial \boldsymbol{\epsilon}}\right) .$$

We now have to figure out the other derivatives in the above expression. First,

$$\frac{\partial \varepsilon_s^e}{\partial \boldsymbol{\epsilon}} = \sqrt{\tfrac{2}{3}} \, \frac{1}{\sqrt{\boldsymbol{e}^e : \boldsymbol{e}^e}} \, \frac{\partial \boldsymbol{e}^e}{\partial \boldsymbol{\epsilon}} : \boldsymbol{e}^e = \sqrt{\tfrac{2}{3}} \, \frac{1}{\|\boldsymbol{e}^e\|} \left(\frac{\partial \boldsymbol{e}^e}{\partial \boldsymbol{\epsilon}} - \tfrac{1}{3} \boldsymbol{1} \otimes \frac{\partial \varepsilon_v^e}{\partial \boldsymbol{\epsilon}}\right) : \boldsymbol{e}^e \,.$$

For the special situation where all the strain is elastic, $\boldsymbol{\epsilon} = \boldsymbol{\epsilon}^e$, and (see Wikipedia article on tensor derivatives)

$$\frac{\partial \boldsymbol{\epsilon}^e}{\partial \boldsymbol{\epsilon}} = \frac{\partial \boldsymbol{\epsilon}}{\partial \boldsymbol{\epsilon}} = \mathbf{I}^{(s)} \quad \text{and} \quad \frac{\partial \varepsilon_v^e}{\partial \boldsymbol{\epsilon}} = \frac{\partial \varepsilon_v}{\partial \boldsymbol{\epsilon}} = \boldsymbol{1} \,.$$

That gives us

$$\frac{\partial \varepsilon_s^e}{\partial \boldsymbol{\epsilon}} = \sqrt{\tfrac{2}{3}} \, \frac{1}{\|\boldsymbol{e}^e\|} \left(\mathbf{I}^{(s)} - \tfrac{1}{3} \boldsymbol{1} \otimes \boldsymbol{1}\right) : \boldsymbol{e}^e = \sqrt{\tfrac{2}{3}} \, \frac{1}{\|\boldsymbol{e}^e\|} \left[\boldsymbol{e}^e - \tfrac{1}{3} \mathrm{tr}(\boldsymbol{e}^e) \boldsymbol{1}\right] .$$

But $\mathrm{tr}(\boldsymbol{e}^e) = 0$ because this is the deviatoric part of the strain and we have

$$\boxed{\frac{\partial \varepsilon_s^e}{\partial \boldsymbol{\epsilon}} = \sqrt{\tfrac{2}{3}} \, \frac{\boldsymbol{e}^e}{\|\boldsymbol{e}^e\|} = \sqrt{\tfrac{2}{3}} \, \boldsymbol{n}} \quad \text{and} \quad \boxed{\frac{\partial \varepsilon_v^e}{\partial \boldsymbol{\epsilon}} = \boldsymbol{1} \,.}$$

Using these, we get

$$\frac{\partial p}{\partial \boldsymbol{\epsilon}} = \frac{p_o}{\tilde{\kappa}} \, \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right] \left(\sqrt{6}\, \alpha \, \varepsilon_s^e \, \boldsymbol{n} - \beta \, \boldsymbol{1}\right) . \tag{6.4}$$

The derivative of q with respect to $\boldsymbol{\epsilon}$ can be calculated in a similar way, i.e.,

$$\frac{\partial q}{\partial \boldsymbol{\epsilon}} = 3\mu \, \frac{\partial \varepsilon_s^e}{\partial \boldsymbol{\epsilon}} + 3\varepsilon_s^e \, \frac{\partial \mu}{\partial \boldsymbol{\epsilon}} = 3\mu \, \frac{\partial \varepsilon_s^e}{\partial \boldsymbol{\epsilon}} - 3 \frac{p_o}{\tilde{\kappa}} \, \alpha \, \varepsilon_s^e \, \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right] \frac{\partial \varepsilon_v^e}{\partial \boldsymbol{\epsilon}} \,.$$

Using the expressions in the boxes above,

$$\frac{\partial q}{\partial \boldsymbol{\epsilon}} = \sqrt{6}\, \mu \, \boldsymbol{n} - 3 \frac{p_o}{\tilde{\kappa}} \, \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right] \alpha \, \varepsilon_s^e \, \boldsymbol{1} \,. \tag{6.5}$$

Also,

$$\frac{\partial \boldsymbol{n}}{\partial \boldsymbol{\epsilon}} = \sqrt{\tfrac{2}{3}} \left[\frac{1}{\varepsilon_s^e} \frac{\partial \boldsymbol{e}^e}{\partial \boldsymbol{\epsilon}} - \frac{1}{(\varepsilon_s^e)^2} \boldsymbol{e}^e \otimes \frac{\partial \varepsilon_s^e}{\partial \boldsymbol{\epsilon}} \right] .$$

Using the previously derived expression, we have

$$\frac{\partial \boldsymbol{n}}{\partial \boldsymbol{\epsilon}} = \sqrt{\tfrac{2}{3}} \frac{1}{\varepsilon_s^e} \left[\mathbf{I}^{(s)} - \tfrac{1}{3} \boldsymbol{1} \otimes \boldsymbol{1} - \sqrt{\tfrac{2}{3}} \frac{1}{\varepsilon_s^e} \frac{\boldsymbol{e}^e \otimes \boldsymbol{e}^e}{\|\boldsymbol{e}^e\|} \right]$$

or

$$\frac{\partial \boldsymbol{n}}{\partial \boldsymbol{\epsilon}} = \sqrt{\tfrac{2}{3}} \frac{1}{\varepsilon_s^e} \left[\mathbf{I}^{(s)} - \tfrac{1}{3} \boldsymbol{1} \otimes \boldsymbol{1} - \boldsymbol{n} \otimes \boldsymbol{n} \right] . \tag{6.6}$$

Plugging the expressions for these derivatives in the original equation, we get

$$\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\epsilon}} = \frac{p_o}{\tilde{\kappa}} \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}} \right] \left(\sqrt{6}\, \alpha\, \varepsilon_s^e\, \boldsymbol{1} \otimes \boldsymbol{n} - \beta\, \boldsymbol{1} \otimes \boldsymbol{1} \right) + 2\mu\, \boldsymbol{n} \otimes \boldsymbol{n} - \sqrt{6} \frac{p_o}{\tilde{\kappa}} \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}} \right] \alpha\, \varepsilon_s^e\, \boldsymbol{n} \otimes \boldsymbol{1} + \tfrac{2}{3} \frac{q}{\varepsilon_s^e} \left[\mathbf{I}^{(s)} - \tfrac{1}{3} \boldsymbol{1} \otimes \boldsymbol{1} - \boldsymbol{n} \otimes \boldsymbol{n} \right] .$$

Reorganizing,

$$\boxed{\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\epsilon}} = \frac{\sqrt{6}\, p_o\, \alpha\, \varepsilon_s^e}{\tilde{\kappa}} \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}} \right] (\boldsymbol{1} \otimes \mathbf{n} + \mathbf{n} \otimes \boldsymbol{1}) - \left(\frac{p_o \beta}{\tilde{\kappa}} \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}} \right] + \tfrac{2}{9} \frac{q}{\varepsilon_s^e} \right) \boldsymbol{1} \otimes \boldsymbol{1} + 2 \left(\mu - \tfrac{1}{3} \frac{q}{\varepsilon_s^e} \right) \boldsymbol{n} \otimes \boldsymbol{n} + \tfrac{2}{3} \frac{q}{\varepsilon_s^e} \mathbf{I}^{(s)} .} \tag{6.7}$$

6.3.4 Tangent calculation: elastic-plastic

From the previous section recall that

$$\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\epsilon}} = \boldsymbol{1} \otimes \frac{\partial p}{\partial \boldsymbol{\epsilon}} + \sqrt{\tfrac{2}{3}}\, \boldsymbol{n} \otimes \frac{\partial q}{\partial \boldsymbol{\epsilon}} + \sqrt{\tfrac{2}{3}}\, q \frac{\partial \boldsymbol{n}}{\partial \boldsymbol{\epsilon}}$$

where

$$\frac{\partial p}{\partial \boldsymbol{\epsilon}} = \frac{p_o}{\tilde{\kappa}} \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}} \right] \left(3\alpha\, \varepsilon_s^e \frac{\partial \varepsilon_s^e}{\partial \boldsymbol{\epsilon}} - \beta \frac{\partial \varepsilon_v^e}{\partial \boldsymbol{\epsilon}} \right) , \qquad \frac{\partial q}{\partial \boldsymbol{\epsilon}} = 3\mu \frac{\partial \varepsilon_s^e}{\partial \boldsymbol{\epsilon}} - 3 \frac{p_o}{\tilde{\kappa}} \alpha\, \varepsilon_s^e \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}} \right] \frac{\partial \varepsilon_v^e}{\partial \boldsymbol{\epsilon}} \quad \text{and}$$

$$\frac{\partial \boldsymbol{n}}{\partial \boldsymbol{\epsilon}} = \sqrt{\tfrac{2}{3}} \left[\frac{1}{\varepsilon_s^e} \frac{\partial \boldsymbol{e}^e}{\partial \boldsymbol{\epsilon}} - \frac{1}{(\varepsilon_s^e)^2} \boldsymbol{e}^e \otimes \frac{\partial \varepsilon_s^e}{\partial \boldsymbol{\epsilon}} \right] .$$

The total strain is equal to the elastic strain for the purely elastic case and the tangent is relatively straightforward to calculate. For the elastic-plastic case we have

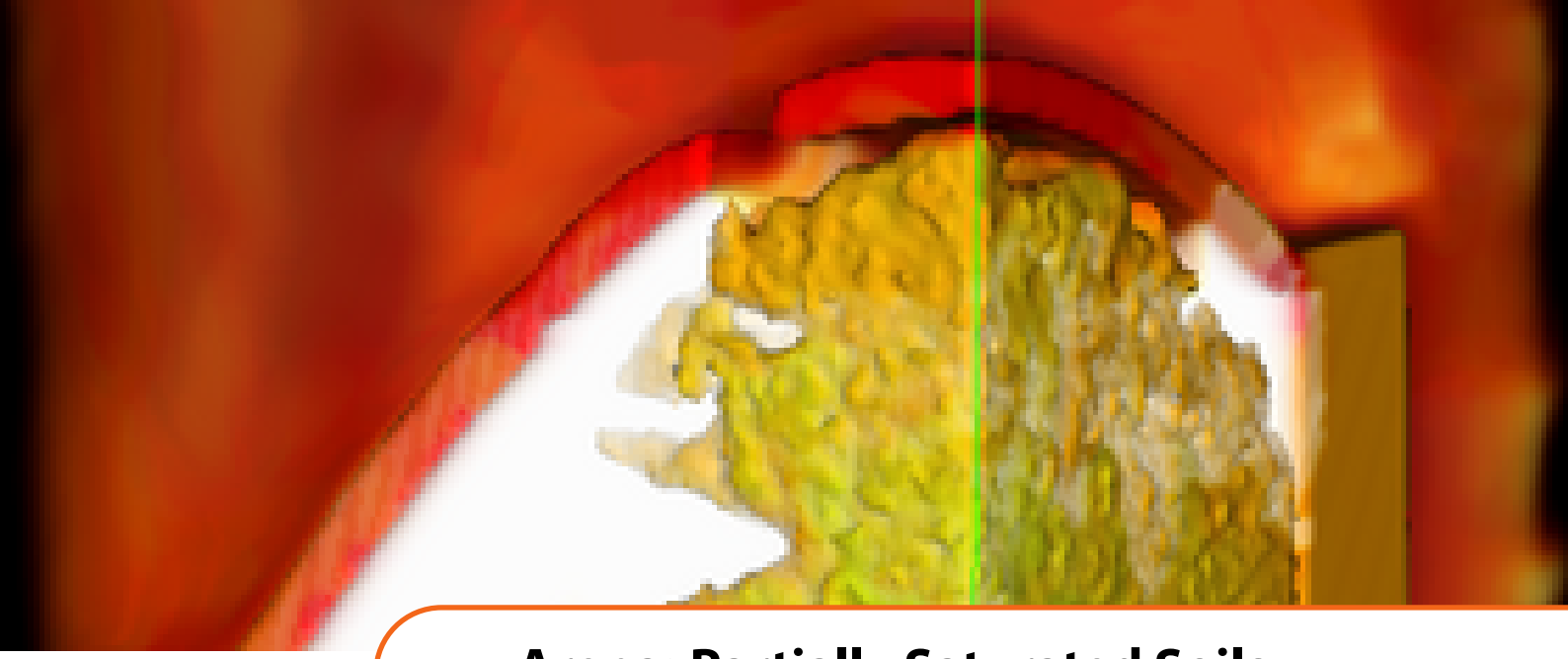
$$\boldsymbol{\epsilon}_{n+1}^e = \boldsymbol{\epsilon}_{\text{trial}}^e - \Delta\gamma \left[\frac{\partial f}{\partial \boldsymbol{\sigma}} \right]_{n+1} .$$

Dropping the subscript $n+1$ for convenience, we have

$$\frac{\partial \boldsymbol{\epsilon}^e}{\partial \boldsymbol{\epsilon}} = \frac{\partial \boldsymbol{\epsilon}_{\text{trial}}^e}{\partial \boldsymbol{\epsilon}} - \frac{\partial f}{\partial \boldsymbol{\sigma}} \otimes \frac{\partial \Delta\gamma}{\partial \boldsymbol{\epsilon}} - \Delta\gamma \frac{\partial}{\partial \boldsymbol{\epsilon}} \left[\frac{\partial f}{\partial \boldsymbol{\sigma}} \right] = \mathbf{I}^{(s)} - \left[\frac{2p - p_c}{3} \boldsymbol{1} + \sqrt{\tfrac{3}{2}} \frac{2q}{M^2} \boldsymbol{n} \right] \otimes \frac{\partial \Delta\gamma}{\partial \boldsymbol{\epsilon}} - \Delta\gamma \frac{\partial}{\partial \boldsymbol{\epsilon}} \left[\frac{2p - p_c}{3} \boldsymbol{1} + \sqrt{\tfrac{3}{2}} \frac{2q}{M^2} \boldsymbol{n} \right] .$$

6.4 Caveats

The Cam-Clay implementation in Vaango behaves reasonably for moderate strains but is known to fail to converge for high-rate applications that involve very large plastic strains.



7 — Arena: Partially Saturated Soils

For a more detailed description and a sample of the input file format, please see the manual in the ArenaSoil directory.

The convention used in Vaango is that tension is positive and compression is negative. To keep the notation simple we define, for any x,

$$\bar{x} := -x\,, \quad \dot{x} := \frac{\partial x}{\partial t}\,. \tag{7.1}$$

7.1 Elasticity

The elasticity model used by Arena has the form

$$\overset{\circ}{\boldsymbol{\sigma}}^{\text{eff}} = \dot{\boldsymbol{\sigma}} - \dot{\boldsymbol{\alpha}} = \mathbf{C}^{\text{e}}(\boldsymbol{\sigma}, \boldsymbol{\varepsilon}^{\text{p}}, \phi, S_w) : \overset{\circ}{\boldsymbol{\varepsilon}}^{\text{e}} - \dot{\lambda}\boldsymbol{Z} \tag{7.2}$$

where $\boldsymbol{\sigma}^{\text{eff}}$ is the effective stress, $\boldsymbol{\sigma}$ is the unrotated Cauchy stress, $\boldsymbol{\alpha}$ is the backstress, $\mathbf{C}^{\text{e}}$ is a tangent elastic modulus which depends on the stress (and also the plastic strain $\boldsymbol{\varepsilon}^{\text{p}}$, porosity ϕ, and water saturation S_w), the elastic strain is $\boldsymbol{\varepsilon}^{\text{e}}$, $\dot{\lambda}$ is the plastic flow rate, and $\boldsymbol{Z}$ in an elastic-plastic coupling tensor.

The model assumes that the tangent modulus tensor is isotropic and can be expressed as

$$\mathbf{C}^{\text{e}} = \left(K - \tfrac{2}{3}G\right)\boldsymbol{I}\otimes\boldsymbol{I} + 2G\,\mathbf{I} \tag{7.3}$$

where $K(\boldsymbol{\sigma}, \boldsymbol{\alpha}, \boldsymbol{\varepsilon}^{\text{p}}, \phi, S_w)$ is the bulk modulus, $G(\boldsymbol{\sigma}, \boldsymbol{\alpha}, \boldsymbol{\varepsilon}^{\text{p}}, \phi, S_w)$ is the shear modulus, $\boldsymbol{I}$ is rank-2 identity tensor, and $\mathbf{I}$ is the symmetric part of the rank-4 identity tensor.

If the effective stress is decomposed into volumetric and deviatoric parts:

$$\boldsymbol{\sigma}^{\text{eff}} = -\bar{p}\,\boldsymbol{I} + \mathbf{s}\,, \quad p := \tfrac{1}{3}\text{tr}(\boldsymbol{\sigma}^{\text{eff}})\,, \quad \mathbf{s} := \boldsymbol{\sigma}^{\text{eff}} - \tfrac{1}{3}\text{tr}(\boldsymbol{\sigma}^{\text{eff}})\boldsymbol{I} \tag{7.4}$$

and the elastic strain is also decomposed into volumetric and deviatoric parts

$$\boldsymbol{\varepsilon}^{\text{e}} = -\tfrac{1}{3}\overline{\varepsilon_v^{\text{e}}}\,\boldsymbol{I} + \boldsymbol{\gamma}^{\text{e}}\,, \quad \varepsilon_v^{\text{e}} := \text{tr}(\boldsymbol{\varepsilon}^{\text{e}})\,, \quad \boldsymbol{\gamma}^{\text{e}} := \boldsymbol{\varepsilon}^{\text{e}} - \tfrac{1}{3}\text{tr}(\boldsymbol{\varepsilon}^{\text{e}})\boldsymbol{I} \tag{7.5}$$

the elasticity model (without the coupling term), simplifies to

$$\dot{\bar{p}} = K(\boldsymbol{\sigma}, \alpha, \boldsymbol{\varepsilon}^{\text{p}}, \phi, S_w)\dot{\overline{\varepsilon_v^{\text{e}}}}\,, \quad \overset{\circ}{\mathbf{s}} = 2G(\boldsymbol{\sigma}, \alpha, \boldsymbol{\varepsilon}^{\text{p}}, \phi, S_w)\,\overset{\circ}{\boldsymbol{\gamma}}^{\text{e}}\,. \tag{7.6}$$

The partially saturated Arena model assumes the moduli depend only on

$$I_1 := \text{tr}(\boldsymbol{\sigma})\,, \quad \zeta = \text{tr}(\boldsymbol{\alpha})\,, \quad \varepsilon_v^{\text{p}} := \text{tr}(\boldsymbol{\varepsilon}^{\text{p}})\,, \quad \phi\,, \quad S_w\,. \tag{7.7}$$

7.1.1 Bulk modulus model: Solid matrix material

The pressure in the solid matrix is expressed as

$$\bar{p}_s = K_s \overline{\varepsilon_v^s} \; ; \quad \overline{\varepsilon_v^s} := \ln\left(\frac{V_{so}}{V_s}\right) \tag{7.8}$$

where $\bar{p}_s$ is the solid matrix pressure, K_s is the solid bulk modulus, $\overline{\varepsilon_v^s}$ is the volumetric strain, V_{so} is the initial volume of the solid, and V_s is the current volume of the solid. The solid bulk modulus is assumed to modeled by the Murnaghan equation:

$$K_s(\bar{p}_s) = K_{so} + n_s\,(\bar{p}_s - \bar{p}_{so}) \tag{7.9}$$

where K_{so} and n_s are material properties, and $\bar{p}_{so}$ is a reference pressure.

7.1.2 Bulk modulus model: Pore water

The equation of state of the pore water is

$$\bar{p}_w = K_w \overline{\varepsilon_v^w} + \bar{p}_o \; ; \quad \overline{\varepsilon_v^w} := \ln\left(\frac{V_{wo}}{V_w}\right) \tag{7.10}$$

where $\bar{p}_w$ is the water pressure, K_w is the water bulk modulus, V_{wo} is the initial volume of water, V_w is the current volume of water, $\bar{p}_o$ is the initial water pressure, and $\bar{\varepsilon}_v^w$ is the volumetric strain in the water. We use the isothermal Murnaghan bulk modulus model for water:

$$K_w(\bar{p}_w) = K_{wo} + n_w\,(\bar{p}_w - \bar{p}_{wo}) \tag{7.11}$$

where K_{wo} and n_w are material properties, and $\bar{p}_{wo}$ is a reference pressure.

7.1.3 Bulk modulus model: Pore air

The isentropic ideal gas equation of state for the pore air is

$$\bar{p}_a = \bar{p}_r\left[\exp(\gamma\,\overline{\varepsilon_v^a}) - 1\right] \; ; \quad \overline{\varepsilon_v^a} := \ln\left(\frac{V_{ao}}{V_a}\right) \tag{7.12}$$

where the quantities with subscript a represent quantities for the air model analogous to those for the water model in (7.10), $\bar{p}_r$ is a reference pressure (101325 Pa) and $\gamma = 1.4$. The bulk modulus of air (K_a) varies with the volumetric strain in the air:

$$K_a = \frac{d\bar{p}_a}{d\overline{\varepsilon_v^a}} = \gamma\,\bar{p}_r\,\exp(\gamma\,\overline{\varepsilon_v^a}) = \gamma\,(\bar{p}_a + \bar{p}_r)\,. \tag{7.13}$$

7.1.4 Bulk modulus model: Drained soil

The pressure model for drained soils has the form

$$\frac{\bar{p}^{\text{eff}}}{K_s(\bar{p}^{\text{eff}})} = b_0\,\overline{\varepsilon_v^e} + \frac{b_1(\overline{\varepsilon_v^e})^{b_4}}{b_2(\overline{\varepsilon_v^e})^{b_4} + b_3} \tag{7.14}$$

where the material parameters are $b_0 > 0, b_1 > 0, b_2 > 0, b_3 > 0, b_4 > 1$. Dependence on plastic strain can be added to the model if necessary.

The tangent bulk modulus is defined as

$$K_d(\bar{p}^{\text{eff}}) := \frac{d\bar{p}^{\text{eff}}}{d\overline{\varepsilon_v^e}}\,. \tag{7.15}$$

Then, using (7.14),

$$K_d(\bar{p}^{\text{eff}}) = \frac{[K_s(\bar{p}^{\text{eff}})]^2}{[K_s(\bar{p}^{\text{eff}}) - n_s\bar{p}^{\text{eff}}]}\left[b_0 + \frac{b_1 b_3 b_4 (\bar{\varepsilon}_v^e)^{b_4-1}}{\left[b_2(\bar{\varepsilon}_v^e)^{b_4} + b_3\right]^2}\right]. \tag{7.16}$$

To express (7.16) in closed-form in terms of $\bar{p}$ we have to eliminate $\bar{\varepsilon}_v^e$. But a closed form expression for the volumetric elastic strain cannot be derived from the pressure model. So we find an approximate form of (7.14) by assuming $b_0 \to 0$. This approximation is valid at moderate to large strains. Then, from (7.14) with $b_0 = 0$, we have

$$\bar{\varepsilon}_v^e \approx \left[\frac{b_3\bar{p}^{\text{eff}}}{b_1 K_s(\bar{p}^{\text{eff}}) - b_2\bar{p}^{\text{eff}}}\right]^{1/b_4} \tag{7.17}$$

and (7.16) can be expressed in terms of $\bar{p}$ as

$$K_d(\bar{p}^{\text{eff}}) = \frac{[K_s(\bar{p}^{\text{eff}})]^2}{[K_s(\bar{p}^{\text{eff}}) - n_s\bar{p}^{\text{eff}}]}\left[b_0 + \frac{b_1 b_3 b_4 \left(\frac{b_3\bar{p}^{\text{eff}}}{b_1 K_s(\bar{p}^{\text{eff}}) - b_2\bar{p}^{\text{eff}}}\right)^{1-1/b_4}}{\left[b_2\left(\frac{b_3\bar{p}^{\text{eff}}}{b_1 K_s(\bar{p}^{\text{eff}}) - b_2\bar{p}^{\text{eff}}}\right) + b_3\right]^2}\right]. \tag{7.18}$$

7.1.5 Bulk modulus model: Partially saturated soil

The pressure in the partially saturated soil ($\bar{p}$) is given by

$$\bar{p} = \int K(\bar{I}_1, \bar{\zeta}, \bar{\varepsilon}_v^p, \phi, S_w)\, \mathrm{d}\bar{\varepsilon}_v^e. \tag{7.19}$$

Note that

$$\bar{p}^{\text{eff}} = \tfrac{1}{3}(\bar{I}_1 - \bar{\zeta}). \tag{7.20}$$

The tangent bulk modulus of the partially saturated soil is found using a variation on the Grassman model for fully saturated rocks:

$$K(\bar{p}^{\text{eff}}, \bar{\varepsilon}_v^p, \phi, S_w) = K_d(\bar{p}^{\text{eff}}) + \frac{\left(1 - \frac{K_d(\bar{p}^{\text{eff}})}{K_s(\bar{p}^{\text{eff}})}\right)^2}{\frac{1}{K_s(\bar{p}^{\text{eff}})}\left(1 - \frac{K_d(\bar{p}^{\text{eff}})}{K_s(\bar{p}^{\text{eff}})}\right) + \phi\left(\frac{1}{K_f(\bar{\zeta})} - \frac{1}{K_s(\bar{p}^{\text{eff}})}\right)} \tag{7.21}$$

where K is the effective bulk modulus of the partially saturated soil, K_d is the bulk modulus of the drained soil, K_f is the bulk modulus of the pore fluid, and K_s is the bulk modulus of the solid grains. At partial saturation, we compute the pore fluid bulk modulus using a harmonic mean (lower bound) on the air and water bulk moduli (K_a, K_f):

$$\frac{1}{K_f(\bar{\zeta})} = \frac{S_w}{K_w(\bar{\zeta})} + \frac{1 - S_w}{K_a(\bar{\zeta})}. \tag{7.22}$$

7.1.6 Shear modulus model: Drained soil

The shear modulus is typically assumed to be constant. However, a variable shear modulus may be needed to fit experimental data and to prevent negative values of Poisson's ratio in the simulations. In those situations a variable Poisson's ratio (ν) is defined as

$$\nu = \nu_1 + \nu_2 \exp\left[-\frac{K_d(\bar{p}^{\text{eff}}, \bar{\varepsilon}_v^p, \phi, S_w)}{K_s(\bar{p}^{\text{eff}})}\right] \tag{7.23}$$

where ν_1 and ν_2 are material parameters. The shear modulus is computed using the Poisson's ratio and the drained bulk modulus:

$$G(\bar{p}^{\text{eff}}, \bar{\varepsilon}_v^p, \phi, S_w) = \frac{3K_d(\bar{p}^{\text{eff}}, \bar{\varepsilon}_v^p, \phi, S_w)(1 - 2\nu)}{2(1 + \nu)}. \tag{7.24}$$

7.2 Rate-independent plasticity

7.2.1 Yield function

The Arena yield function is

$$f = \sqrt{J_2} - F_f(\bar{I}_1, \zeta)\, F_c(\bar{I}_1, \bar{\zeta}, \overline{X}, \bar{\kappa}) = \sqrt{J_2} - F_f(\bar{p}^{\text{eff}})\, F_c(\bar{p}^{\text{eff}}, \overline{X}, \bar{\kappa}) \tag{7.25}$$

where

$$F_f(\bar{p}^{\text{eff}}) = a_1 - a_3 \exp[-3a_2\bar{p}^{\text{eff}})] + 3a_4\bar{p}^{\text{eff}} \tag{7.26}$$

and

$$F_c(\bar{p}^{\text{eff}}, \overline{X}, \bar{\kappa}) = \begin{cases} 1 & \text{for} \quad 3\bar{p}^{\text{eff}} \le \bar{\kappa} \\ \sqrt{1 - \left(\dfrac{3\bar{p}^{\text{eff}} - \bar{\kappa}}{\overline{X} - \bar{\kappa}}\right)^2} & \text{for} \quad 3\bar{p}^{\text{eff}} > \bar{\kappa}\,. \end{cases} \tag{7.27}$$

Here $\overline{X}$ is the hydrostatic compressive strength, $\bar{\kappa}$ is the branch point at which the cap function F_c starts decreasing until it reaches the hydrostatic strength point $(\overline{X}, 0)$, and

$$J_2 = \tfrac{1}{2}\mathbf{s} : \mathbf{s}\,. \tag{7.28}$$

Non-associativity is modeled using a parameter β that modifies $\sqrt{J_2}$ (see 7.8).

7.2.2 Hydrostatic compressive strength: Drained soil

The drained crush curve model is used to compute $\overline{X}$ and has the form

$$\overline{\varepsilon_v^p} - p_3 = \ln\left[1 - \frac{1 - \exp(-p_3)}{1 + \left(\frac{\overline{X}_d - p_0}{p_1}\right)^{p_2}}\right]. \tag{7.29}$$

where p_0, p_1, p_2, p_3 are model parameters and $\bar{\xi} = \overline{X} - \bar{p}_0$ where $\overline{X}$ is the hydrostatic compressive strength. The parameter p_3 is related to the initial porosity (ϕ_0) by $p_3 = -\ln(1 - \phi_0)$.

The drained hydrostatic compressive strength ($\overline{X}_d/3$) is found from the drained material crush curve using

$$\overline{X}_d(\overline{\varepsilon_v^p}, \phi_0) - p_0 = p_1\left[\frac{1 - \exp(-p_3)}{1 - \exp(-p_3 + \overline{\varepsilon_v^p})} - 1\right]^{1/p_2}, \quad p_3 := -\ln(1 - \phi_0)\,. \tag{7.30}$$

7.2.3 Hydrostatic compressive strength: Partially saturated soil

The elastic part of the volumetric strain at yield is defined in the model as

$$\varepsilon_v^{e,\text{yield}}(\overline{\varepsilon_v^p}) = \frac{\overline{X}_d(\overline{\varepsilon_v^p}, \phi_0)}{3\,K_d\left(\frac{1}{2}\frac{\overline{X}_d(\overline{\varepsilon_v^p}, \phi_0)}{3}\right)} \tag{7.31}$$

where X_d is found from the drained material crush curve.

The elastic volumetric strain at yield is assumed to be identical for drained and partially saturated materials. With this assumption, the compressive strength of a partially saturated sand is given by

$$\overline{X}(\overline{\varepsilon_v^p}) = 3K(\bar{p}^{\text{eff}}, \overline{\varepsilon_v^p}, \phi, S_w)\, \bar{\varepsilon}_v^{e,\text{yield}}(\overline{\varepsilon_v^p}) \tag{7.32}$$

where K is the bulk modulus of the partially saturated material.

7.2.4 Backstress: Pore pressure

The pore pressure as an isotropic backstress (ζ) that translates the Cauchy stress to the effective stress:

$$\boldsymbol{\sigma}^{\text{eff}} = \boldsymbol{\sigma} - \zeta \boldsymbol{I}\,, \quad \zeta := -\left[(1 - S_w)\bar{p}_a + S_w \bar{p}_w\right]. \tag{7.33}$$

In the elastically unloaded state (where the effective stress is zero) we assume that the pore pressure ($\bar{\zeta}$) is related to the volumetric plastic strain by

$$\exp(-\overline{\varepsilon_v^p}) = \phi_o\,(1 - S_o)\exp\left[-\frac{1}{\gamma}\ln\left(\frac{\bar{\zeta}}{\bar{p}_r} + 1\right)\right] + \phi_o\, S_o\, \exp\left(-\frac{\bar{\zeta} - \bar{p}_o}{K_w}\right) + (1 - \phi_o)\exp\left(-\frac{\bar{\zeta}}{K_s}\right). \tag{7.34}$$

This equation can be solved for $\bar{\zeta}(\overline{\varepsilon_v^p})$ using a root finding algorithm.

Alternatively, this equation can be converted into rate form and integrated using an explicit time stepping method if a Newton solve is too expensive or fails to converge:

$$\zeta = \int \frac{d\zeta}{d\varepsilon_v^p}\mathrm{d}\varepsilon_v^p\,. \tag{7.35}$$

where

$$\frac{d\zeta}{d\varepsilon_v^p} = \frac{\exp(-\overline{\varepsilon_v^p})}{\mathcal{B}}\,, \tag{7.36}$$

and

$$\mathcal{B} := \left[\frac{\phi_o\,(1 - S_o)}{\gamma(\bar{p}_r + \bar{\zeta})}\right]\exp\left[-\frac{1}{\gamma}\ln\left(\frac{\bar{\zeta}}{\bar{p}_r} + 1\right)\right] + \frac{\phi_o\, S_o}{K_w}\exp\left(\frac{\bar{p}_o - \bar{\zeta}}{K_w}\right) + \frac{1 - \phi_o}{K_s}\exp\left(-\frac{\bar{\zeta}}{K_s}\right). \tag{7.37}$$

7.3 Rate-dependent plasticity

7.4 Porosity and saturation

The total volumetric strain is given by

$$\exp(\varepsilon_v) = (1 - S_o)\phi_o \exp(\varepsilon_v^a) + S_o\phi_o \exp(\varepsilon_v^w) + (1 - \phi_o)\exp(\varepsilon_v^s) \tag{7.38}$$

where ϕ_o, S_o are the initial porosity and saturation, and

$$\varepsilon_v^w(\varepsilon_v) = -\frac{\bar{p}(\varepsilon_v) - \bar{p}_o}{K_w}\,, \quad \varepsilon_v^a(\varepsilon_v) = -\frac{1}{\gamma}\ln\left[1 + \frac{\bar{p}(\varepsilon_v)}{\bar{p}_r}\right]\,, \quad \varepsilon_v^s(\varepsilon_v) = -\frac{\bar{p}(\varepsilon_v)}{K_s}\,. \tag{7.39}$$

We can combine (7.38) and (7.39) to solve for $\bar{p}(\varepsilon_v)$ and then compute the volumetric strain in the air in terms of the total volumetric strain.

7.4.1 Saturation

The saturation function $S_w(\varepsilon_v)$, is given by

$$S_w(\varepsilon_v) = \frac{\mathcal{C}(\varepsilon_v)}{1 + \mathcal{C}(\varepsilon_v)}\,, \quad \mathcal{C}(\varepsilon_v) := \left(\frac{S_o}{1 - S_o}\right)\exp(\varepsilon_v^w)\exp(-\varepsilon_v^a)\,. \tag{7.40}$$

7.4.2 Porosity

The porosity evolution equation (in the elastically unloaded state) for partially saturated sand has the form

$$\phi(\varepsilon_v) = \phi_o\left(\frac{1 - S_o}{1 - S_w(\varepsilon_v)}\right)\left[\frac{\exp(\varepsilon_v^a)}{\exp(\varepsilon_v)}\right]. \tag{7.41}$$

7.5 Summary of partially saturated soil model

Summary 7.5.1 ***Bulk modulus model***

Drained soil:
The equation of state of the drained soil is

$$K_d = \frac{[K_s]^2}{[K_s - n_s \bar{p}^{\text{eff}}]}\left[b_0 + \frac{b_1 b_3 b_4 (\bar{\varepsilon}_v^e)^{b_4 - 1}}{\left[b_2 (\bar{\varepsilon}_v^e)^{b_4} + b_3\right]^2}\right], \quad \bar{\varepsilon}_v^e \approx \left[\frac{b_3 \bar{p}^{\text{eff}}}{b_1 K_s - b_2 \bar{p}^{\text{eff}}}\right]^{1/b_4}.$$

Partially saturated soil:
The bulk modulus model is

$$K = K_d + \frac{\left(1 - \frac{K_d}{K_s}\right)^2}{\frac{1}{K_s}\left(1 - \frac{K_d}{K_s}\right) + \phi\left(\frac{S_w}{K_w} + \frac{1 - S_w}{K_a} - \frac{1}{K_s}\right)}$$

where

$$K_s(\bar{p}) = K_{so} + n_s(\bar{p} - \bar{p}_{so}), \quad K_w(\bar{p}) = K_{wo} + n_w(\bar{p} - \bar{p}_{wo}), \quad K_a(\bar{p}) = \gamma(\bar{p} + \bar{p}_r)$$

Summary 7.5.2 ***Shear modulus model***

The shear modulus is either a constant (G_o) or determined using a variable Poisson's ratio (ν)

$$\nu = \nu_1 + \nu_2 \exp\left[-\frac{K_d(\bar{p}^{\text{eff}}, \bar{\varepsilon}_v^p, \phi, S_w)}{K_s(\bar{p}^{\text{eff}})}\right]$$

$$G(\bar{p}^{\text{eff}}, \bar{\varepsilon}_v^p, \phi, S_w) = \frac{3K_d(\bar{p}^{\text{eff}}, \bar{\varepsilon}_v^p, \phi, S_w)(1 - 2\nu)}{2(1 + \nu)}.$$

Summary 7.5.3 ***Yield function***

The Arena yield function is

$$f = \sqrt{J_2} - F_f(\bar{I}_1, \zeta)\, F_c(\bar{I}_1, \bar{\zeta}, \bar{X}, \bar{\kappa}) = \sqrt{J_2} - F_f(\bar{p}^{\text{eff}})\, F_c(\bar{p}^{\text{eff}}, \bar{X}, \bar{\kappa}) \tag{7.42}$$

where

$$F_f(\bar{p}^{\text{eff}}) = a_1 - a_3 \exp[-3a_2 \bar{p}^{\text{eff}})] + 3a_4 \bar{p}^{\text{eff}} \tag{7.43}$$

and

$$F_c(\bar{p}^{\text{eff}}, \bar{X}, \bar{\kappa}) = \begin{cases} 1 & \text{for} \quad 3\bar{p}^{\text{eff}} \le \bar{\kappa} \\ \sqrt{1 - \left(\dfrac{3\bar{p}^{\text{eff}} - \bar{\kappa}}{\bar{X} - \bar{\kappa}} \right)^2} & \text{for} \quad 3\bar{p}^{\text{eff}} > \bar{\kappa}\,. \end{cases} \tag{7.44}$$

Non-associativity is modeled using a parameter β that modifies $\sqrt{J_2}$.

Summary 7.5.4 ***Hydrostatic strength model***

Drained soil:

$$\bar{X}_d(\bar{\varepsilon}_v^p) - p_0 = p_1 \left[\frac{1 - \exp(-p_3)}{1 - \exp(-p_3 + \bar{\varepsilon}_v^p)} - 1 \right]^{1/p_2} \quad , \quad p_3 = -\ln(1 - \phi_0)\,.$$

Partially saturated soil:

$$\bar{X}(\bar{\varepsilon}_v^p) = 3K(\bar{I}_1, \bar{\varepsilon}_v^p, \phi, S_w)\, \bar{\varepsilon}_v^{e,\text{yield}}(\bar{\varepsilon}_v^p)$$

where

$$\varepsilon_v^{e,\text{yield}}(\bar{\varepsilon}_v^p) = \frac{\bar{X}_d(\bar{\varepsilon}_v^p)}{3\,K_d\left(\dfrac{\bar{X}_d(\bar{\varepsilon}_v^p)}{6}, \bar{\varepsilon}_v^p \right)}$$

Summary 7.5.5 ***Pore pressure model***

Solve $g(\bar{\zeta}, \overline{\varepsilon_v^p}) = 0$ for $\bar{\zeta}$.

$$g(\bar{\zeta}, \overline{\varepsilon_v^p}) = -\exp(-\overline{\varepsilon_v^p}) + \phi_0\,(1-S_0)\exp\left[-\frac{1}{\gamma}\ln\left(\frac{\bar{\zeta}}{\bar{p}_r}+1\right)\right] + \phi_0\,S_0\,\exp\left(-\frac{\bar{\zeta}-\bar{p}_0}{K_w}\right) + (1-\phi_0)\exp\left(-\frac{\bar{\zeta}}{K_s}\right).$$

Alternatively, integrate

$$\bar{\zeta} = \int \frac{d\bar{\zeta}}{d\overline{\varepsilon_v^p}}\,\mathrm{d}\overline{\varepsilon_v^p}\,.$$

where

$$\frac{d\bar{\zeta}}{d\overline{\varepsilon_v^p}} = \frac{\exp(-\overline{\varepsilon_v^p})}{\mathcal{B}},$$

and

$$\mathcal{B} := \left[\frac{\phi_0\,(1-S_0)}{\gamma(\bar{p}_r+\bar{\zeta})}\right]\exp\left[-\frac{1}{\gamma}\ln\left(\frac{\bar{\zeta}}{\bar{p}_r}+1\right)\right] + \frac{\phi_0\,S_0}{K_w}\exp\left(\frac{\bar{p}_0-\bar{\zeta}}{K_w}\right) + \frac{1-\phi_0}{K_s}\exp\left(-\frac{\bar{\zeta}}{K_s}\right).$$

Summary 7.5.6 ***Saturation and porosity evolution***

Saturation:

$$S_w(\varepsilon_v) = \frac{\mathcal{C}(\varepsilon_v)}{1+\mathcal{C}(\varepsilon_v)}\,,\quad \mathcal{C}(\varepsilon_v) := \left(\frac{S_0}{1-S_0}\right)\exp(\varepsilon_v^w)\exp(-\varepsilon_v^a)\,.$$

where ϕ_0, S_0 are the initial porosity and saturation, and

$$\varepsilon_v^w(\varepsilon_v) = -\frac{\bar{p}(\varepsilon_v)-\bar{p}_0}{K_w}\,,\quad \varepsilon_v^a(\varepsilon_v) = -\frac{1}{\gamma}\ln\left[1+\frac{\bar{p}(\varepsilon_v)}{\bar{p}_r}\right]\,,\quad \varepsilon_v^s(\varepsilon_v) = -\frac{\bar{p}(\varepsilon_v)}{K_s}\,.$$

Porosity:

$$\phi(\varepsilon_v) = \phi_0\left(\frac{1-S_0}{1-S_w(\varepsilon_v)}\right)\left[\frac{\exp(\varepsilon_v^a)}{\exp(\varepsilon_v)}\right]. \tag{7.45}$$

Note that

$$\exp(\varepsilon_v) = (1-S_0)\phi_0\exp(\varepsilon_v^a) + S_0\phi_0\exp(\varepsilon_v^w) + (1-\phi_0)\exp(\varepsilon_v^s)$$

7.6 Computing the stress and internal variables

The partially saturated soil model uses Michael Homel's "consistency bisection" algorithm to find the plastic strain direction and to update the internal state variables. A closest-point return algorithm in transformed stress space is used to project the trial stress state on to the yield surface. Because of the nonlinearities in the material models, it is easier to solve the problem by dividing the strain increment to substeps.

The partially saturated soil model treats the porosity (ϕ) and saturation (S_w) as internal variables in addition to the hydrostatic compressive strength (X), the isotropic backstress (ζ), and the plastic strain ($\boldsymbol{\varepsilon}^{\mathrm{p}}$) which are used by the fully saturated model.

The inputs to the rate-independent stress update algorithm for a single material point are:

- $\mathbf{d}^{\mathrm{n}}$: the rate of deformation at time $t = t_n$; defined as $\mathbf{d} := \frac{1}{2}(\boldsymbol{l} + \boldsymbol{l}^T)$ where $\boldsymbol{l} = \nabla \mathbf{v}$ and $\mathbf{v}$ is the velocity field.
- Δt : the time step
- $\boldsymbol{\sigma}^{\mathrm{n}}$: the unrotated Cauchy step at time $t = t_n$.
- ϕ^{n} : the porosity at time $t = t_n$.
- S_w^{n} : the saturation at time $t = t_n$.
- X^{n} : the hydrostatic compressive strength at time $t = t_n$.
- ζ^{n} : the trace of the backstress at time $t = t_n$.
- $\boldsymbol{\varepsilon}^{\mathrm{p,n}}$: the plastic strain at time $t = t_n$.

After the return algorithm has been exercised, the outputs from the algorithm are:

- $\boldsymbol{\sigma}^{\mathrm{n+1}}$: the unrotated Cauchy step at time $t = t_{n+1} = t_n + \Delta t$.
- $\phi^{\mathrm{n+1}}$: the porosity at time $t = t_{n+1}$.
- $S_w^{\mathrm{n+1}}$: the saturation at time $t = t_{n+1}$.
- $X^{\mathrm{n+1}}$: the hydrostatic compressive strength at time $t = t_{n+1}$.
- $\zeta^{\mathrm{n+1}}$: the trace of the backstress at time $t = t_{n+1}$.
- $\boldsymbol{\varepsilon}^{\mathrm{p,n+1}}$: the plastic strain at time $t = t_{n+1}$.

The update algorithm uses the standard predictor-corrector approach of hypoelastic-plasticity where a trial predictor stress is computed first and then a corrector return algorithm is used to locate the position of the correct stress on the yield surface. This approach requires that the trial stress ($\boldsymbol{\sigma}^{\mathrm{trial}}$) is computed using the relation

$$\boldsymbol{\sigma}^{\mathrm{trial}} = \boldsymbol{\sigma}^{\mathrm{n}} + \mathbf{C}^{\mathrm{e}} : (\mathbf{d}\,\Delta t) \tag{7.46}$$

where $\mathbf{C}^{\mathrm{e}}$ is an elastic modulus that is typically assumed to be constant over the time step Δt. Though this assumption suffices for nonlinear elastic materials if the rate of deformation is small or the timestep is small or both, for large $\mathbf{d}\Delta t$ significant errors can enter the calculation. The Vaango implementation assumes that $\mathbf{C}^{e}$ is the tangent modulus at the beginning of a timestep (or load substep).

Caveat:

The partially saturated soil model has been developed for an explicit dynamics code where tiemsteps are typically very small. Care should be exercised if the application domain requires timesteps to be large.

Remark:

Note that in the Kayenta model (which is the basis for Arenisca and Arena), the bulk modulus has a high pressure limit. This limit was used by Michael Homel in Arenisca3 and Arenisca4 to define conservative elastic properties during the stress and internal variable update. However, the bulk modulus model used by the partially saturated version of Arena does not have this limit. Therefore the trial stress for the partially saturated model is computed using an alternative approach that assumes that the elastic moduli are those at the beginning of the timestep (or load substep).

After the trial stress is computed, the timestep is subdivided into substeps based on the characteristic dimension of the yield surface relative to the magnitude of the trial stress increment ($\boldsymbol{\sigma}^{\mathrm{trial}} - \boldsymbol{\sigma}^{\mathrm{n}}$). The substep size is then recomputed by comparing the elastic properties at $\boldsymbol{\sigma}^{\mathrm{trial}}$ with those at $\boldsymbol{\sigma}^{\mathrm{n}}$ to make sure that the nonlinear elastic solution is accurate.

The pseudocode for the algorithm is given below.

Algorithm 15 The stress and internal variable update algorithm

procedure RATEINDEPENDENTPLASTICUPDATE($\mathbf{d}^{\mathrm{n}}$, Δt, $\boldsymbol{\sigma}^{\mathrm{n}}$, ϕ^{n}, S_w^{n}, X^{n}, ζ^{n}, $\boldsymbol{\varepsilon}^{\mathrm{p,n}}$)
  $K^{\mathrm{n}}, G^{\mathrm{n}} \leftarrow$ COMPUTEELASTICMODULI($\boldsymbol{\sigma}^{\mathrm{n}}$, $\boldsymbol{\varepsilon}^{\mathrm{p,n}}$, ϕ^{n}, S_w^{n}) ▷*Compute tangent bulk and shear modulus*
  $\boldsymbol{\sigma}^{\mathrm{trial}} \leftarrow$ COMPUTETRIALSTRESS($\boldsymbol{\sigma}^{\mathrm{n}}$, K^{n}, G^{n}, $\mathbf{d}^{\mathrm{n}}$, Δt) ▷*Compute trial stress*
  $n_{\mathrm{sub}} \leftarrow$ COMPUTESTEPDIVISIONS($\boldsymbol{\sigma}^{\mathrm{n}}$, $\boldsymbol{\varepsilon}^{\mathrm{p,n}}$, ϕ^{n}, S_w^{n}, $\boldsymbol{\sigma}^{\mathrm{trial}}$) ▷*Compute number of substeps*
  $\delta t \leftarrow \dfrac{\Delta t}{n_{\mathrm{sub}}}$ ▷*Substep timestep*
  $\boldsymbol{\sigma}^{\mathrm{old}} \leftarrow \boldsymbol{\sigma}^{\mathrm{n}}$, $\boldsymbol{\varepsilon}^{\mathrm{p,old}} \leftarrow \boldsymbol{\varepsilon}^{\mathrm{p,n}}$, $\phi^{\mathrm{old}} \leftarrow \phi^{\mathrm{n}}$, $S_w^{\mathrm{old}} \leftarrow S_w^{\mathrm{n}}$, $X^{\mathrm{old}} \leftarrow X^{\mathrm{n}}$, $\zeta^{\mathrm{old}} \leftarrow \zeta^{\mathrm{n}}$
  $\chi \leftarrow 1$, $t_{\mathrm{local}} \leftarrow 0.0$ ▷*Initialize substep multiplier and accumulated time increment*
  isSuccess $\leftarrow$ FALSE
  repeat
    isSuccess, $\boldsymbol{\sigma}^{\mathrm{new}}$, $\boldsymbol{\varepsilon}^{\mathrm{p,new}}$, ϕ^{new}, S_w^{new}, X^{new}, $\zeta^{\mathrm{new}} \leftarrow$ COMPUTESUBSTEP($\boldsymbol{\sigma}^{\mathrm{old}}$, $\boldsymbol{\varepsilon}^{\mathrm{p,old}}$, ϕ^{old}, S_w^{old}, X^{old}, ζ^{old}, $\mathbf{d}^{\mathrm{n}}$, δt) ▷*Compute updated stress and internal variable for the current substep*
    if isSuccess = TRUE **then**
      $t_{\mathrm{local}} \leftarrow t_{\mathrm{local}} + \delta t$
      $\boldsymbol{\sigma}^{\mathrm{old}} \leftarrow \boldsymbol{\sigma}^{\mathrm{new}}$, $\boldsymbol{\varepsilon}^{\mathrm{p,old}} \leftarrow \boldsymbol{\varepsilon}^{\mathrm{p,new}}$, $\phi^{\mathrm{old}} \leftarrow \phi^{\mathrm{new}}$, $S_w^{\mathrm{old}} \leftarrow S_w^{\mathrm{new}}$ $X^{\mathrm{old}} \leftarrow X^{\mathrm{new}}$, $\zeta^{\mathrm{old}} \leftarrow \zeta^{\mathrm{new}}$
    else
      $\chi \leftarrow 2\chi$
      $\delta t \leftarrow \delta t/2$ ▷*Halve the timestep*
      if $\chi >$ CHI_MAX **then**
        return IsSuccess, $\boldsymbol{\sigma}^{\mathrm{n}}$, ϕ^{n}, S_w^{n}, X^{n}, ζ^{n}, $\boldsymbol{\varepsilon}^{\mathrm{p,n}}$ ▷*Algorithm has failed to converge*
      end if
    end if
  until $t_{\mathrm{local}} \geq \Delta t$
  return IsSuccess, $\boldsymbol{\sigma}^{\mathrm{new}}$, ϕ^{new}, S_w^{new}, X^{new}, ζ^{new}, $\boldsymbol{\varepsilon}^{\mathrm{p,new}}$ ▷*Algorithm has converged*
end procedure

Algorithm 16 Computing the elastic moduli

procedure COMPUTEELASTICMODULI($\boldsymbol{\sigma}^{\mathrm{n}}$, $\boldsymbol{\varepsilon}^{\mathrm{p,n}}$, ϕ^{n}, S_w^{n})
  $K \leftarrow 0$, $G \leftarrow 0$
  $\bar{I}_1 \leftarrow -\mathrm{tr}(\boldsymbol{\sigma}^{\mathrm{n}})$, $\bar{\varepsilon}_v^{\mathrm{p}} \leftarrow -\mathrm{tr}(\boldsymbol{\varepsilon}^{\mathrm{p,n}})$
  if $S_w^{\mathrm{n}} > 0$ **then**
    $K, G \leftarrow$ COMPUTEPARTIALSATURATEDMODULI($\bar{I}_1$, $\bar{\varepsilon}_v^{\mathrm{p}}$, ϕ^{n}, S_w^{n})
  else
    $K, G \leftarrow$ COMPUTEDRAINEDMODULI($\bar{I}_1$, $\bar{\varepsilon}_v^{\mathrm{p}}$)
  end if
  return K, G
end procedure

Algorithm 17 Computing the partially saturated elastic moduli

Require: K_{so}, n_s, $\bar{p}_{so}$, K_{wo}, n_w, $\bar{p}_{wo}$, γ, $\bar{p}_r$

procedure COMPUTEPARTIALSATURATEDMODULI($\bar{I}_1$, $\bar{\varepsilon}_v^{\mathrm{p}}$, ϕ^{n}, S_w^{n})
  if $\bar{I}_1 > 0$ **then**
    $\bar{p} \leftarrow \bar{I}_1/3$
    $K_s \leftarrow K_{so} + n_s(\bar{p} - \bar{p}_{so})$
    $K_w \leftarrow K_{wo} + n_w(\bar{p} - \bar{p}_{wo})$
    $K_a \leftarrow \gamma(\bar{p} + \bar{p}_r)$
    $K_d, G \leftarrow$ COMPUTEDRAINEDMODULI($\bar{I}_1$, $\bar{\varepsilon}_v^{\mathrm{p}}$)
    $K_f \leftarrow 1.0/\left[S_w^{\mathrm{n}}/K_w + (1.0 - S_w^{\mathrm{n}})/K_a\right]$ ▷*Bulk modulus of air + water mixture*
    numer $\leftarrow (1.0 - K_d/K_s)^2$

denom $\leftarrow (1.0/K_s)\,(1.0 - K_d/K_s) + \phi^{\mathrm{n}}\,(1.0/K_f - 1.0/K_s)$
$K \leftarrow K_d + \text{numer}/\text{denom}$ ▷*Bulk modulus of partially saturated material (Biot-Grassman model)*
else
$K, G \leftarrow$ computeDrainedModuli$(\overline{I}_1, \overline{\varepsilon}_v^{\mathrm{p}})$
end if
return K, G
end procedure

Algorithm 18 Computing the drained elastic moduli

Require: K_{so}, n_s, $\overline{p}_{so}$, b_0, b_1, b_2, b_3, b_4, G_0, ν_1, ν_2
procedure computeDrainedModuli$(\overline{I}_1, \overline{\varepsilon}_v^{\mathrm{p}})$
if $\overline{I}_1 > 0$ **then**
$\overline{p} \leftarrow \overline{I}_1/3$
$K_s \leftarrow K_{so} + n_s(\overline{p} - \overline{p}_{so})$
$K_s^{\text{ratio}} \leftarrow K_s/(1.0 - n_s * \overline{p}/K_s)$
$\varepsilon_v^{\mathrm{e}} \leftarrow \text{pow}((b_3 * \overline{p})/(b_1 K_s - b_2 \overline{p}), (1.0/b_4))$;
$y \leftarrow \text{pow}(\varepsilon_v^{\mathrm{e}}, b_4)$
$z \leftarrow b_2 y + b_3$
$K \leftarrow K_s^{\text{ratio}}[b_0 + (1/\varepsilon_v^{\mathrm{e}}) b_1 b_3 b_4 y/z^2]$; ▷ *Compute compressive bulk modulus*
$\nu = \nu_1 + \nu_2 \exp(-K/K_s)$
$G \leftarrow G_0$
if $\nu > 0$ **then**
$G \leftarrow 1.5K\,(1.0 - 2.0\nu)/(1.0 + \nu)$ ▷*Update the shear modulus (if $nu_1, \nu_2 > 0$)*
end if
else
$K \leftarrow b_0 K_{so}$ ▷*Tensile bulk modulus = Bulk modulus at $p = 0$*
$G \leftarrow G_0$ ▷*Tensile shear modulus*
end if
return K, G
end procedure

Algorithm 19 Computing the trial stress

procedure computeTrialStress$(\boldsymbol{\sigma}^{\mathrm{n}}, K^{\mathrm{n}}, G^{\mathrm{n}}, \mathbf{d}^{\mathrm{n}}, \Delta t)$
$\Delta\boldsymbol{\varepsilon} \leftarrow \mathbf{d}^{\mathrm{n}}\,\Delta t$ ▷*Total strain increment*
$\Delta\boldsymbol{\varepsilon}^{\text{iso}} \leftarrow \frac{1}{3}\text{tr}(\Delta\boldsymbol{\varepsilon})\boldsymbol{I}$
$\Delta\boldsymbol{\varepsilon}^{\text{dev}} \leftarrow \Delta\boldsymbol{\varepsilon} - \Delta\boldsymbol{\varepsilon}^{\text{iso}}$
$\boldsymbol{\sigma}^{\text{trial}} \leftarrow \boldsymbol{\sigma}^{\mathrm{n}} + 3K^{\mathrm{n}}\Delta\boldsymbol{\varepsilon}^{\text{iso}} + 2G^{\mathrm{n}}\Delta\boldsymbol{\varepsilon}^{\text{dev}}$
return $\boldsymbol{\sigma}^{\text{trial}}$
end procedure

Algorithm 20 Computing the initial number of substeps

Require: $n^{\max}$, I_1^{peak}, STREN, $\epsilon \leftarrow 10^{-4}$
procedure computeStepDivisions$(\boldsymbol{\sigma}^{\mathrm{n}}, \boldsymbol{\varepsilon}^{\mathrm{p,n}}, \phi^{\mathrm{n}}, S_w^{\mathrm{n}}, \boldsymbol{\sigma}^{\text{trial}}, X^{\mathrm{n}})$
$K^{\mathrm{n}}, G^{\mathrm{n}} \leftarrow$ computeElasticModuli$(\boldsymbol{\sigma}^{\mathrm{n}}, \boldsymbol{\varepsilon}^{\mathrm{p,n}}, \phi^{\mathrm{n}}, S_w^{\mathrm{n}})$
$K^{\text{trial}}, G^{\text{trial}} \leftarrow$ computeElasticModuli$(\boldsymbol{\sigma}^{\text{trial}}, \boldsymbol{\varepsilon}^{\mathrm{p,n}}, \phi^{\mathrm{n}}, S_w^{\mathrm{n}})$
$n^{\text{bulk}} \leftarrow \lceil |K^{\mathrm{n}} - K^{\text{trial}}|/K^{\mathrm{n}} \rceil$ ▷*Compute change in bulk modulus*
$\Delta\boldsymbol{\sigma} \leftarrow \boldsymbol{\sigma}^{\text{trial}} - \boldsymbol{\sigma}^{\mathrm{n}}$
$L \leftarrow \frac{1}{2}(I_1^{\text{peak}} - X^{\mathrm{n}})$

```
    if STREN > 0.0 then
        L ← min(L, STREN)
    end if
    n^yield ← ⌈ϵ × ‖Δσ‖ / L⌉                ▷Compute trial stress increment relative to yield surface size
    n^sub ← max(n^bulk, n^yield)            ▷n^sub is the maximum of the two values
    if n^sub > n^max then
        n^sub ← −1
    else
        n^sub ← min(max(n^sub, 1), n^max)
    end if
    return n^sub
end procedure
```

Algorithm 21 Computing the stress and internal variable update for a substep

```
procedure computeSubstep(σ^old, ε^p,old, φ^old, S_w^old, X^old, ζ^old, d^n, δt)
    K^old, G^old ← computeElasticModuli(σ^old, ε^p,old, φ^old, S_w^old)   ▷Compute tangent bulk and shear modulus
    δε ← d^n δt                                                           ▷Compute strain increment
    σ^trial ← computeTrialStress(σ^old, K^old, G^old, d^n, Δt)            ▷Compute trial stress
    I_1^trial, √J_2^trial ← stressInvariants(σ^trial)                     ▷Compute invariants of the trial stress
    isElastic ← evalYieldCondition(I_1^trial, √J_2^trial, X^old, ζ^old, K^old, G^old, β)
    if isElastic = TRUE then
        σ^new ← σ^trial, ε^p,new ← ε^p,old, φ^new ← φ^old, S_w^new ← S_w^old, X^new ← X^old, ζ^new ← ζ^old
        isSuccess = TRUE
        return isSuccess, σ^new, ε^p,new φ^new, S_w^new, X^new, ζ^new
    end if
    σ^o, δε^p,o ← nonHardeningReturn(σ^old, σ^trial, δε, X^old, ζ^old, K^old, G^old, β, I_1^peak)   ▷Compute return to updated yield surface (no hardening)
    isSuccess, σ^new, ε^p,new, X^new, ζ^new, K^mid, G^mid ← consistencyBisection(ε^p,old, δε^p,o, ζ^old, σ^o, σ^trial, K^old, G^old, β, I_1^peak)
    if iSuccess = FALSE then
        return isSuccess, σ^old, ε^p,old, φ^old, S_w^old, X^old, ζ^old
    end if
    return isSuccess, σ^new, ε^p,new, φ^new, S_w^new, X^new, ζ^new
end procedure
```

7.7 The consistency bisection algorithm

7.7.1 Fixed (nonhardening) yield surface

Let the stress at the beginning of the load step be $\boldsymbol{\sigma}^{\text{old}}$ and let the trial stress be $\boldsymbol{\sigma}^{\text{trial}}$. Assume the yield surface is fixed and let the correct projection of the trial stress on to the fixed yield surface be $\boldsymbol{\sigma}^{\text{new,o}}$.

The increment of stress for the load step ($\Delta\boldsymbol{\sigma}^{\text{o}}$) is related to the elastic strain increment ($\Delta\boldsymbol{\varepsilon}^{\text{e,o}}$)by

$$\Delta\boldsymbol{\sigma}^{\text{o}} = \boldsymbol{\sigma}^{\text{new,o}} - \boldsymbol{\sigma}^{\text{old}} = \mathbf{C} : \Delta\boldsymbol{\varepsilon}^{\text{e,o}} \tag{7.47}$$

where $\mathbf{C}$ is a constant elastic modulus tensor. The elastic modulus tensor can be assumed to be an average value of the nonlinear tangent modulus for the load step.

If we know $\mathbf{C}$, we can compute the elastic strain increment using

$$\Delta\boldsymbol{\varepsilon}^{\text{e,o}} = \mathbf{C}^{-1} : \Delta\boldsymbol{\sigma}^{\text{o}} \,. \tag{7.48}$$

For a strain driven update algorithm, the total strain increment $\Delta\boldsymbol{\varepsilon}$ is known. Assuming that the total strain increment can be additively decomposed into an elastic and a plastic part, we can find the plastic strain increment ($\Delta\boldsymbol{\varepsilon}^{\mathrm{p,o}}$) using

$$\Delta\boldsymbol{\varepsilon}^{\mathrm{p,o}} = \Delta\boldsymbol{\varepsilon} - \Delta\boldsymbol{\varepsilon}^{\mathrm{e,o}} \,. \tag{7.49}$$

7.7.2 Hardening yield surface

Now, if we allow the yield surface to harden, the distance between the trial stress point and its projection on to the yield surface decreases compared to that for a fixed yield surface. If $\Delta\boldsymbol{\varepsilon}^{\mathrm{p}}$ is the plastic strain increment for a hardening yield surface, we have

$$\Delta\boldsymbol{\varepsilon}^{\mathrm{p}} > \Delta\boldsymbol{\varepsilon}^{\mathrm{p,o}} \tag{7.50}$$

where the inequality can be evaluated using an appropriate Euclidean norm. Note that this distance is proportional to the consistency parameter $\dot{\lambda}$.

Fully saturated model

In the fully saturated version of the Arenisca model, the internal variables are the hydrostatic compressive strength (X) and the scalar isotropic backstress (ζ). These depend only on the **volumetric** plastic strain increment

$$\Delta\varepsilon_v^{\mathrm{p}} = \mathrm{tr}(\Delta\boldsymbol{\varepsilon}^{\mathrm{p}}) \,. \tag{7.51}$$

Because

$$\Delta\varepsilon_v^{\mathrm{p}} > \Delta\varepsilon_v^{\mathrm{p,o}} \tag{7.52}$$

we can define a parameter, $\eta \in (0, 1)$, such that

$$\eta := \frac{\Delta\varepsilon_v^{\mathrm{p}}}{\Delta\varepsilon_v^{\mathrm{p,o}}} \,. \tag{7.53}$$

Because the solution is bounded by the fixed yield surface, a bisection algorithm can be used to find the parameter η.

Partially saturated model

TODO

7.7.3 Bisection algorithm: Fully saturated

Algorithm 22 The consistency bisection algorithm for fully saturated materials

procedure CONSISTENCYBISECTION($\boldsymbol{\varepsilon}^{\mathrm{p,old}}$, $\delta\boldsymbol{\varepsilon}^{\mathrm{p,o}}$, ζ^{old}, $\boldsymbol{\sigma}^{\mathrm{o}}$, $\boldsymbol{\sigma}^{\mathrm{trial}}$, K^{old}, G^{old}, β, I_1^{peak})
  $\boldsymbol{\sigma}^{\mathrm{new}} \leftarrow \boldsymbol{\sigma}^{\mathrm{o}}$, $\delta\boldsymbol{\varepsilon}^{\mathrm{p}} \leftarrow \delta\boldsymbol{\varepsilon}^{\mathrm{p,o}}$
  $\varepsilon_v^{\mathrm{p,old}} \leftarrow \mathrm{tr}(\boldsymbol{\varepsilon}^{\mathrm{p,old}})$, $\delta\varepsilon_v^{\mathrm{p,old}} \leftarrow \mathrm{tr}(\delta\boldsymbol{\varepsilon}^{\mathrm{p,old}})$
  $i \leftarrow 1$
  $\eta^{\mathrm{in}} \leftarrow 0$, $\eta^{\mathrm{out}} \leftarrow 1$
  repeat
    $j \leftarrow 1$
    isElastic $\leftarrow$ TRUE
    while isElastic = TRUE **do**
      $\eta^{\mathrm{mid}} \leftarrow \frac{1}{2}(\eta^{\mathrm{out}} + \eta^{\mathrm{in}})$

$X^{\text{new}} \leftarrow$ computeHydrostaticStrength$(\varepsilon_v^{\text{p,old}} + \eta^{\text{mid}}\,\delta\varepsilon_v^{\text{p,o}})$ ▷*Update the hydrostatic compressive strength*

$\frac{\partial\zeta}{\partial\varepsilon_v^{\text{p}}} \leftarrow$ computeDerivativeOfBackstress(Arguments?)

$\zeta^{\text{new}} \leftarrow \zeta^{\text{old}} + \left(\frac{\partial\zeta}{\partial\varepsilon_v^{\text{p}}}\right) \times (\eta^{\text{mid}}\,\delta\varepsilon_v^{\text{p,o}})$ ▷*Update the isotropic backstress*

$I_1^{\text{trial}}, \sqrt{J_2^{\text{trial}}} \leftarrow$ stressInvariants$(\boldsymbol{\sigma}^{\text{trial}})$ ▷*Compute invariants of the trial stress*

isElastic $\leftarrow$ evalYieldCondition$(I_1^{\text{trial}}, \sqrt{J_2^{\text{trial}}}, X^{\text{new}}, \zeta^{\text{new}}, K^{\text{old}}, G^{\text{old}}, \beta)$

$\eta^{\text{out}} \leftarrow \eta^{\text{mid}}$ ▷*Too much plastic strain*

$j \leftarrow j + 1$

if $j \geq j^{\max}$ **then**

isSuccess $\leftarrow$ FALSE

return isSuccess

end if

end while

$\boldsymbol{\sigma}^{\text{mid}} \leftarrow \frac{1}{2}(\boldsymbol{\sigma}^{\text{old}} + \boldsymbol{\sigma}^{\text{new}})$

$\boldsymbol{\varepsilon}^{\text{p,mid}} \leftarrow \boldsymbol{\varepsilon}^{\text{p,old}} + \frac{1}{2}\eta^{\text{mid}}\,\delta\boldsymbol{\varepsilon}^{\text{p,o}}$

$K^{\text{mid}}, G^{\text{mid}} \leftarrow$ computeElasticModuli$(\boldsymbol{\sigma}^{\text{mid}}, \boldsymbol{\varepsilon}^{\text{p,mid}})$

$\boldsymbol{\sigma}^{\text{new}}, \delta\boldsymbol{\varepsilon}^{\text{p,new}} \leftarrow$ nonHardeningReturn$(\boldsymbol{\sigma}^{\text{old}}, \boldsymbol{\sigma}^{\text{trial}}, \delta\boldsymbol{\varepsilon}^{\text{new}}, X^{\text{new}}, \zeta^{\text{new}}, K^{\text{mid}}, G^{\text{mid}}, \beta, I_1^{\text{peak}})$ ▷*Compute return to updated yield surface (no hardening)*

if $\text{sign}(\text{tr}(\boldsymbol{\sigma}^{\text{trial}} - \boldsymbol{\sigma}^{\text{new}})) \neq \text{sign}(\text{tr}(\boldsymbol{\sigma}^{\text{trial}} - \boldsymbol{\sigma}^{\text{o}}))$ **or** $\|\delta\boldsymbol{\varepsilon}^{\text{p,new}}\|_2 > \eta^{\text{mid}}\,\|\delta\boldsymbol{\varepsilon}^{\text{p,o}}\|_2$ **then**

$\eta^{\text{out}} \leftarrow \eta^{\text{mid}}$ ▷*Too much plastic strain*

else

if $\|\delta\boldsymbol{\varepsilon}^{\text{p,new}}\|_2 < \eta^{\text{mid}}\,\|\delta\boldsymbol{\varepsilon}^{\text{p,o}}\|_2$ **then**

$\eta^{\text{in}} \leftarrow \eta^{\text{mid}}$ ▷*Too little plastic strain*

end if

end if

$i \leftarrow i + 1$

if $i \geq i^{\max}$ **then**

isSuccess $\leftarrow$ FALSE

return isSuccess

end if

until $\text{abs}(\|\delta\boldsymbol{\varepsilon}^{\text{p,new}}\|_2 - \eta^{\text{mid}}\,\|\delta\boldsymbol{\varepsilon}^{\text{p,o}}\|_2) <$ TOLERANCE

$\boldsymbol{\varepsilon}^{\text{p,new}} = \boldsymbol{\varepsilon}^{\text{p,old}} + \delta\boldsymbol{\varepsilon}^{\text{p,new}}$ ▷*Update the plastic strain*

$X^{\text{new}} \leftarrow$ computeHydrostaticStrength$(\text{tr}(\boldsymbol{\varepsilon}^{\text{p,new}}))$ ▷*Update the hydrostatic compressive strength*

$\frac{\partial\zeta}{\partial\varepsilon_v^{\text{p}}} \leftarrow$ computeDerivativeOfBackstress(Arguments?)

$\zeta^{\text{new}} \leftarrow \zeta^{\text{old}} + \left(\frac{\partial\zeta}{\partial\varepsilon_v^{\text{p}}}\right) \times (\text{tr}(\delta\boldsymbol{\varepsilon}^{\text{p,new}}))$ ▷*Update the isotropic backstress*

isSuccess $\leftarrow$ TRUE

return isSuccess, $\boldsymbol{\sigma}^{\text{new}}$, $\boldsymbol{\varepsilon}^{\text{p,new}}$, X^{new}, ζ^{new}, K^{mid}, G^{mid}

end procedure

7.8 The nonhardening return algorithm

Let the plastic flow direction be $\boldsymbol{M}$. Then

$$\dot{\boldsymbol{\varepsilon}}^{p} = \dot{\lambda}\boldsymbol{M}\,. \tag{7.54}$$

The nonhardening return algorithm uses a transformed space where the computation is carried out in special Lode coordinates (z', r') where

$$z' = z - \frac{\zeta}{\sqrt{3}}, \quad z := \frac{I_1}{\sqrt{3}} \quad \text{and} \quad r' = \sqrt{\frac{3K}{2G}}\, r, \quad r := \sqrt{2J_2}. \tag{7.55}$$

If the flow rule is non-associative, the yield surface parameter $\beta \neq 1$. In that case,

$$r' \leftarrow \beta r'. \tag{7.56}$$

The quantities needed by the non-hardening return algorithm are:

Require: as input

- $\boldsymbol{\sigma}^{\text{trial}}$ ▷*Trial stress*
- $\boldsymbol{\sigma}^{\text{old}}$ ▷*Stress at the start of the substep*
- $\delta\boldsymbol{\varepsilon}^{\text{new}}$ ▷*Increment of total strain*
- X^{old} ▷*Hydrostatic compressive strength*
- ζ^{old} ▷*Isotropic backstress (trace)*
- K^{old} ▷*Tangent bulk modulus*
- G^{old} ▷*Tangent shear modulus*
- I_1^{peak} ▷*The location of the yield surface vertex*
- β ▷*The yield surface non-associativity parameter*

The nonhardening return algorithm pseudocode is listed below:

Algorithm 23 Non-hardening return algorithm

procedure NONHARDENINGRETURN($\boldsymbol{\sigma}^{\text{old}}$, $\boldsymbol{\sigma}^{\text{trial}}$, $\delta\boldsymbol{\varepsilon}^{\text{new}}$, X^{old}, ζ^{old}, K^{old}, G^{old}, β, I_1^{peak})
  $I_1^{\text{trial}}, J_2^{\text{trial}} \leftarrow$ STRESSINVARIANTS($\boldsymbol{\sigma}^{\text{trial}}$) ▷*Compute invariants of the trial stress*
  $r^{\text{trial}} \leftarrow \beta\sqrt{2J_2^{\text{trial}}}$, $z^{\text{trial}} \leftarrow \dfrac{I_1^{\text{trial}}}{\sqrt{3}}$ ▷*Compute Lode coordinates of the trial stress*
  $(r')^{\text{trial}} \leftarrow r^{\text{trial}}\sqrt{\dfrac{3K^{\text{old}}}{2G^{\text{old}}}}$ ▷*Transform the trial r coordinate*
  $I_1^{\text{o}} \leftarrow \zeta^{\text{old}} + \frac{1}{2}(X^{\text{old}} + I_1^{\text{peak}})$, $J_2^{\text{o}} \leftarrow 0$ ▷*Compute interior point*
  $r^{\text{o}} \leftarrow \beta\sqrt{2J_2^{\text{o}}}$, $z^{\text{o}} \leftarrow \dfrac{I_1^{\text{o}}}{\sqrt{3}}$ ▷*Compute Lode coordinates of the interior point*
  $(r')^{\text{o}} \leftarrow r^{\text{o}}\sqrt{\dfrac{3K^{\text{old}}}{2G^{\text{old}}}}$ ▷*Transform the interior point r coordinate*
  $\theta \leftarrow 0$
  repeat
    $z^{\text{new}}, (r')^{\text{new}} \leftarrow$ APPLYBISECTIONALGORITHM(z^{o}, $(r')^{\text{o}}$, z^{trial}, $(r')^{\text{trial}}$, X^{old}, ζ^{old}, K^{old}, G^{old}, β) ▷*Find intersection point on the non-hardening yield surface*
    θ, z^{rot}, $(r')^{\text{rot}} \leftarrow$ FINDNEWINTERNALPOINT(z^{trial},$(r')^{\text{trial}}$, z^{new}, $(r')^{\text{new}}$, θ, X^{old}, ζ^{old}, K^{old}, G^{old}, β) ▷*Apply rotation algorithm to find new internal point*
    $(r')^{\text{o}} \leftarrow (r')^{\text{rot}}$, $z^{\text{o}} \leftarrow z^{\text{rot}}$
  until $\theta \leq$ TOLERANCE
  $I_1^{\text{new}} = \sqrt{3}\, z^{\text{new}}$, $\sqrt{J_2^{\text{new}}} = \sqrt{\dfrac{2G^{\text{old}}}{3K^{\text{old}}}}\dfrac{(r')^{\text{new}}}{\sqrt{2}\beta}$ ▷*Compute updated stress invariants*
  $\mathbf{s}^{\text{trial}} \leftarrow \boldsymbol{\sigma}^{\text{trial}} - \frac{1}{3} I_1^{\text{trial}} \boldsymbol{I}$ ▷*Compute deviatoric trial stress*
  $\boldsymbol{\sigma}^{\text{new}} = \frac{1}{3} I_1^{\text{new}} \boldsymbol{I} + \dfrac{\sqrt{J_2^{\text{new}}}}{\sqrt{J_2^{\text{trial}}}} \mathbf{s}^{\text{trial}}$ ▷*Compute updated stress*
  $\delta\boldsymbol{\varepsilon}^{\text{p,new}} = \delta\boldsymbol{\varepsilon} - \mathbf{C}^{-1} : (\boldsymbol{\sigma}^{\text{new}} - \boldsymbol{\sigma}^{\text{old}})$ ▷*Compute plastic strain increment*

return Outputs:
- $\boldsymbol{\sigma}^{\text{new}}$ ▷*Updated stress tensor*
- $\delta\boldsymbol{\varepsilon}^{\text{p,new}}$ ▷*Increment in plastic strain*

end procedure

Algorithm 24 Apply bisection algorithm to find point on yield surface.

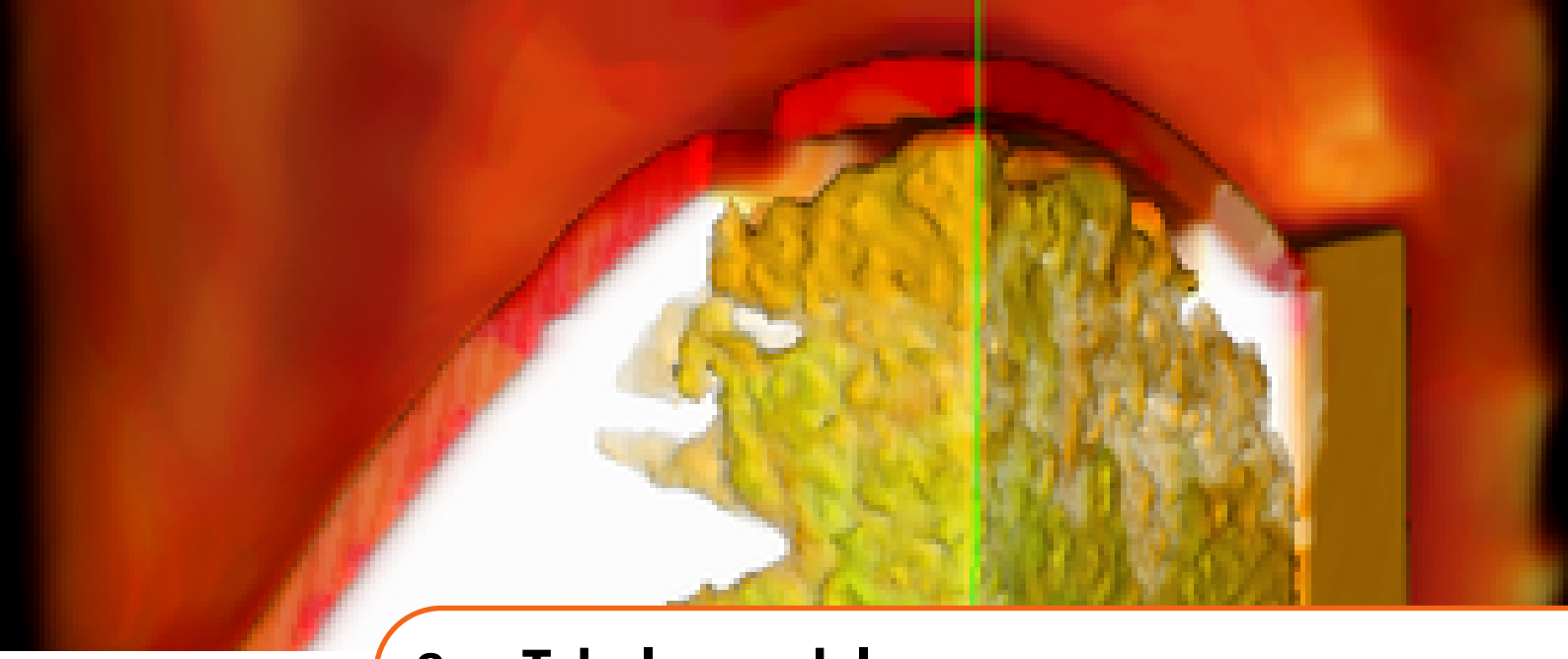
procedure APPLYBISECTIONALGORITHM(z^{o}, $(r')^{\text{o}}$, z^{trial}, $(r')^{\text{trial}}$, X^{old}, ζ^{old}, K^{old}, G^{old}, β)
  $\eta^{\text{in}} \leftarrow 0$, $\eta^{\text{out}} \leftarrow 1$
  while $\eta^{\text{out}} - \eta^{\text{in}} \geq \text{TOL}$ **do**
    $\eta^{\text{mid}} = \frac{1}{2}(\eta^{\text{in}} + \eta^{\text{out}})$
    $\begin{bmatrix} z^{\text{mid}} \\ (r')^{\text{mid}} \end{bmatrix} \leftarrow \eta^{\text{mid}} \begin{bmatrix} z^{\text{trial}} - z^{\text{o}} \\ (r')^{\text{trial}} - (r')^{\text{o}} \end{bmatrix} + \begin{bmatrix} z^{\text{o}} \\ (r')^{\text{o}} \end{bmatrix}$
    isElastic ← EVALYIELDCONDITION(z^{mid}, $(r')^{\text{mid}}$, X^{old}, ζ^{old}, K^{old}, G^{old}, β)
    if isElastic = TRUE **then**
      $\eta^{\text{in}} \leftarrow \eta^{\text{mid}}$
    else
      $\eta^{\text{out}} \leftarrow \eta^{\text{mid}}$
    end if
  end while
  $z^{\text{new}} \leftarrow z^{\text{mid}}$, $(r')^{\text{new}} \leftarrow (r')^{\text{mid}}$
  return z^{new}, $(r')^{\text{new}}$
end procedure

Algorithm 25 Rotation around trial state to find internal point inside yield surface

procedure FINDNEWINTERNALPOINT(z^{trial}, $(r')^{\text{trial}}$, z^{new}, $(r')^{\text{new}}$, θ, X^{old}, ζ^{old}, K^{old}, G^{old}, β)
  $n \leftarrow 0$
  repeat
    $n \leftarrow n + 1$
    $\theta \leftarrow (-1)^n \times \dfrac{\pi}{2} \times \left(\frac{1}{2}\right)^{\frac{\text{floor}(n)}{2}}$
    $[\text{Q}] \leftarrow \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$
    $\begin{bmatrix} z^{\text{rot}} \\ (r')^{\text{rot}} \end{bmatrix} \leftarrow [\text{Q}] \cdot \begin{bmatrix} z^{\text{new}} - z^{\text{trial}} \\ (r')^{\text{new}} - (r')^{\text{trial}} \end{bmatrix} + \begin{bmatrix} z^{\text{trial}} \\ (r')^{\text{trial}} \end{bmatrix}$
    isElastic ← EVALYIELDCONDITION(z^{rot}, $(r')^{\text{rot}}$, X^{old}, ζ^{old}, K^{old}, G^{old}, β)
  until isElastic = FALSE
  return θ, z^{rot}, $(r')^{\text{rot}}$
end procedure

Algorithm 26 Evaluate the yield condition

procedure EVALYIELDCONDITION(z^{new}, $(r')^{\text{new}}$, X^{old}, ζ^{old}, K^{old}, G^{old}, β)
  $I_1^{\text{new}} \leftarrow \sqrt{3}\, z^{\text{new}}$, $\sqrt{J_2^{\text{new}}} \leftarrow \sqrt{\dfrac{2G^{\text{old}}}{3K^{\text{old}}}} \times \dfrac{1}{\sqrt{2}\beta} \times (r')^{\text{new}}$ ▷*Transform back into stress space*
  isElastic ← EVALYIELDCONDITION(I_1^{new}, $\sqrt{J_2^{\text{new}}}$, X^{old}, ζ^{old}, K^{old}, G^{old}, β)
  return isElastic
end procedure



8 — Tabular models

At present we allow only three independent variables in VAANGO .

MPM tabular material data is often of the form shown in Figure 8.1. In this particular data set, we have three independent variables: the plastic strain (β), the saturation (α), and the strain (ε). Pressure (p) is the dependent variable. The data represents a function of the form $p = p(\varepsilon, \alpha, \beta)$. We are given an input point in the three-dimensional independent variable space, (ε_o, α_o, β_o), and we would like to find the corresponding value of the pressure, p_o.

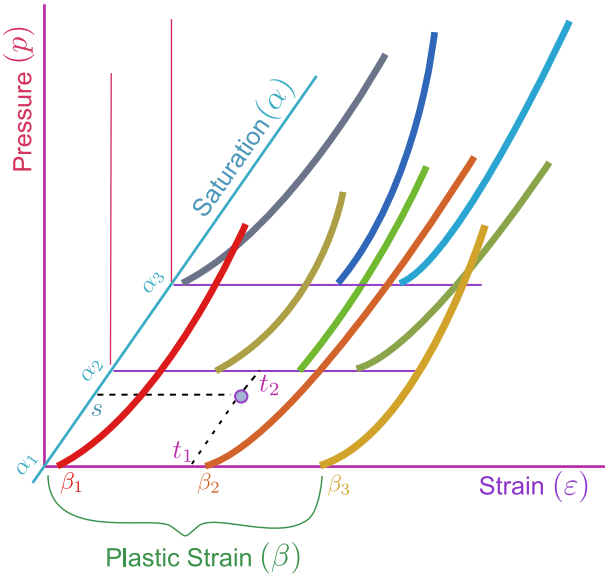


Figure 8.1: Schematic of tabular material data for MPM constitutive models. The circle in blue is the input data point for which we would like to find the pressure.

As we can see from the figure, the data are largely unstructured. However, there is some structure to the data. For instance, the data are provided for three values of saturation, $[\alpha_1, \alpha_2, \alpha_3]$. For each value of α, we have data for a few plastic strain values: $\alpha_1 : [\beta_{11}, \beta_{12}, \beta_{13}]$, $\alpha_2 : [\beta_{21}, \beta_{22}, \beta_{23}, \beta_{24}, \dots]$, and $\alpha_3 : [\beta_{31}, \beta_{32}, \dots]$. Finally, for each value of the plastic strain, we have a pressure-strain curve, for example, for $\alpha_1, \beta_{11} : [\varepsilon_{111}, \varepsilon_{112}, \varepsilon_{113}, \dots, \varepsilon_{11N}]$ and $[p_{111}, p_{112}, p_{113}, \dots, p_{11N}]$, or for $\alpha_3, \beta_{32} : [\varepsilon_{321}, \varepsilon_{322}, \varepsilon_{323}, \dots, \varepsilon_{32M}]$ and $[p_{321}, p_{321}, p_{321}, \dots, p_{32M}]$. Clearly, the data become quite complex as the number of dimensions is

increased.

8.1 Linear interpolation

> The procedure below assumes that the α values are sorted in ascending order. If $\alpha_0 \notin [\alpha_1, \alpha_N]$, VAANGO will throw an exception and exit. Also observe that at least two sets of data are needed for the interpolation procedure to work.

In this section we describe the process used in VAANGO to interpolate the data. For simplicity, we only consider two independent variables, the saturation (α) and the strain (ε) as shown in Figure 8.2.

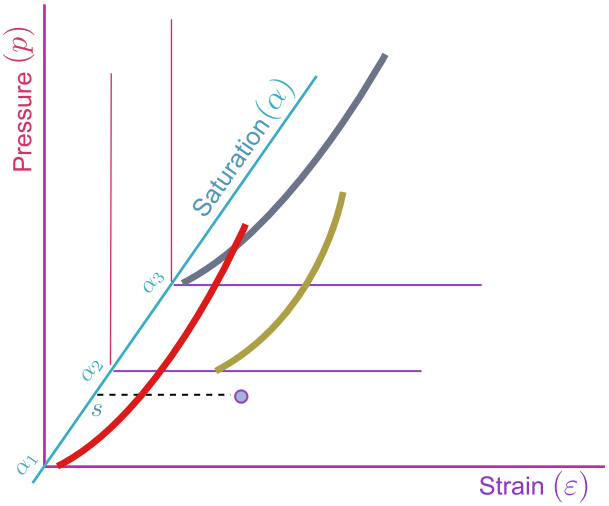


Figure 8.2: Schematic of a three variable table of material data. The circle in blue is the input point for which we would like to find the pressure.

The first step in the process is to find the pressure-strain data that are needed for the interpolation process. This can be accomplished by iterating through the αs and finding a value of the parameter $s \in [0, 1]$ where

$$s = \frac{\alpha_0 - \alpha_k}{\alpha_{k+1} - \alpha_k}, \quad k = 1, 2, \ldots, N-1 \tag{8.1}$$

where N is the number of values of α for which data area available.

Once the two curves needed for interpolation have been identified, the next step is to find the segments of the pressure-strain curves that correspond to the input variable ε_0. These segments are highlighted with thick lines in Figure 8.3. The two associated parameters t_1 and t_2 are calculated using

$$\begin{aligned} t_1 &= \frac{\varepsilon_0 - \varepsilon_{j,k}}{\varepsilon_{j,k+1} - \varepsilon_{j,k}}, \quad k = 1, 2, \ldots, M_j - 1 \\ t_2 &= \frac{\varepsilon_0 - \varepsilon_{j+1,k}}{\varepsilon_{j+1,k+1} - \varepsilon_{j+1,k}}, \quad k = 1, 2, \ldots, M_{j+1} - 1 \end{aligned} \tag{8.2}$$

where $\varepsilon_{j,k}$ is a point on the pressure-strain curve for saturation α_j, and M_j is the number of points on the curve.

We can now compute the pressures at these two points, using

$$\begin{aligned} p_1 &= (1 - t_1) p_{j,k} + t_1 p_{j,k+1} \\ p_2 &= (1 - t_2) p_{j+1,k} + t_2 p_{j+1,k+1} \end{aligned} \tag{8.3}$$

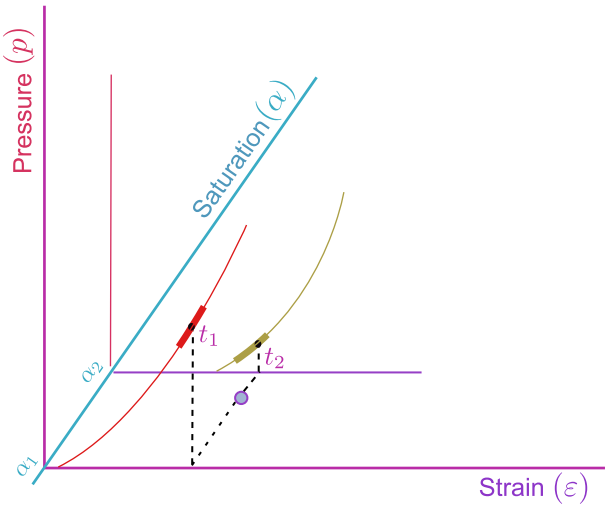


Figure 8.3: Second stage of interpolation of a three variable table of material data. The circle in blue is the input point for which we would like to find the pressure.

The final step of the process is to compute the interpolated pressure p_o using

$$p_o = (1 - s)p_1 + sp_2 \,. \tag{8.4}$$

A schematic of this operation is shown in Figure 8.4.

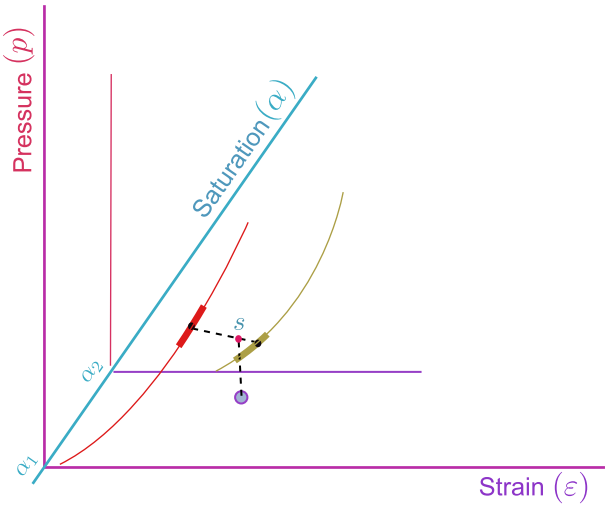


Figure 8.4: Final stage of interpolation of a three variable table of material data. The circle in blue is the input point and the red circle is the interpolated value.

8.2 The tabular equation of state

For the tabular equation of state, we assume that there is only one independent variable, the density ratio $\eta = \rho/\rho_o$ where ρ is the current mass density and ρ_o is its reference value. The dependent variable is the pressure, $\bar{p} = \bar{p}(\eta)$, which is positive in compression. A linear interpolation is done to compute the pressure for a given state of deformation.

The bulk modulus is computed using

$$K = \rho \left[\frac{p(\eta + \epsilon) - p(\eta - \epsilon)}{2\epsilon} \right] \tag{8.5}$$

The tolerance ϵ is hardcoded to 10^{-6} in VAANGO but may not be adequate for some problems.

8.3 The tabular plasticity model

The tabular plasticity model was designed for materials that have almost no tensile strength, and the inputs are expected in the **compression positive** convention. Note that the general convention used in the Vaango code is that **tension is positive and compression is negative**. Conversions are done internally in the code to make sure that signs are consistent.

The model uses isotropic elasticity, with a shear modulus that is either a constant (G_0) or determined using a Poisson's ratio (ν) from the tabular bulk modulus, $K(p)$:

$$G = \frac{3K(1-2\nu)}{2(1+\nu)} \tag{8.6}$$

This relation is activated if $\nu \in [-1.0, 0.5)$, otherwise the constant shear modulus is used.

The tangent bulk modulus is determined from a table of unloading curves (see Figure 8.5 of the mean stress, $\bar{p}$, as a function of the total Hencky volumetric strain, $\bar{\varepsilon}_p$. Each unloading curve is associated with a Hencky plastic volumetric strain ($\bar{\varepsilon}_v^p$). Additive decomposition of the volumetric strains is assumed. The plastic volumetric strain is subtracted from the total volumetric strain to compute the elastic volumetric strain ($\bar{\varepsilon}_v^e$). The data stored in the table is therefore of the form $\bar{p}(\bar{\varepsilon}_v^p, \bar{\varepsilon}_v^e)$ and the bulk modulus is computed, after interpolation, using the central difference scheme:

$$K(\bar{\varepsilon}_{v0}^p, \bar{\varepsilon}_{v0}^e) = \frac{p(\bar{\varepsilon}_{v0}^p, \bar{\varepsilon}_{v0}^e + \epsilon) - p(\bar{\varepsilon}_{v0}^p, \bar{\varepsilon}_{v0}^e - \epsilon)}{2\epsilon} \tag{8.7}$$

The tolerance ϵ is hardcoded to 10^{-6} in VAANGO and may not be adequate for some problems.

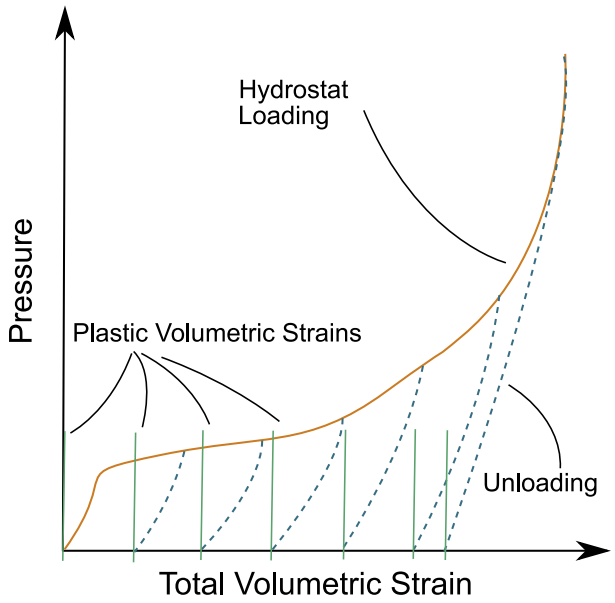
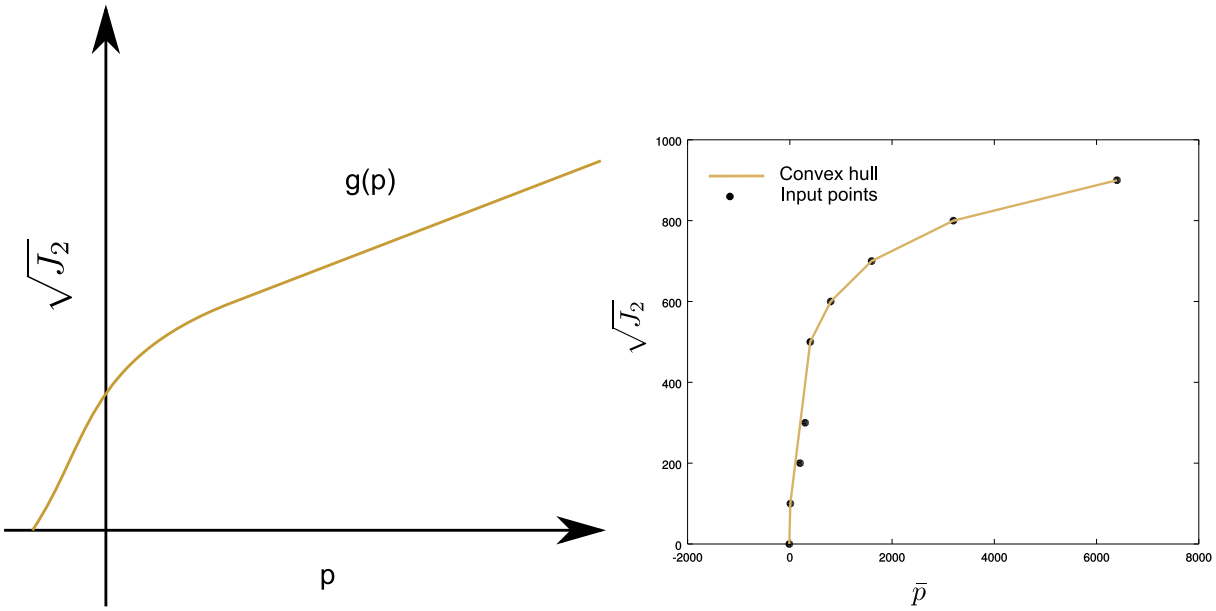


Figure 8.5: Unloading curves uses to determine the tangent bulk modulus for the tabular plasticity model at various plastic strain value.

The tabular yield condition has the form

$$f = \sqrt{J_2} - g(\bar{p}) = 0 \tag{8.8}$$

The function $g(\bar{p})$ is provided in tabular form and is depicted in Figure 8.6(a). To ensure convexity of the tabular data, a convex hull of the data points is computed first as shown in Figure 8.6(b).



(a) Yield function

(b) Computed hull of the input data

Figure 8.6: Yield function used by the tabular plasticity.

Linear interpolation is used to determine is a stress state is inside the yield surface. We also compute a normal to the yield surface using linear interpolation and a central difference scheme (similar to that used to compute the bulk modulus). However, the actual return algorithm uses a geometric closest point computation rather than the derivative of the yield function with respect to the stress. The approach is similar to that used in the ARENA material model.

If the number of points in the input table is equal to 2, the yield function is either a von Mises model or a linear Drucker-Prager model. In that can we find the closest point to the tabular data directly.

For tables with more than two input points, we fit a quadratic B-spline to the closest segment of the input tabular data and find the closest distance to that spline. Approximating, rather than interpolating, splines are used to retain the convexity of the yield function.

The B-splines are computed using

$$s_x = \mathbf{a} \cdot (\boldsymbol{M}_j \cdot \mathbf{p}_x) \,, \quad s_y = \mathbf{a} \cdot (\boldsymbol{M}_j \cdot \mathbf{p}_y) \tag{8.9}$$

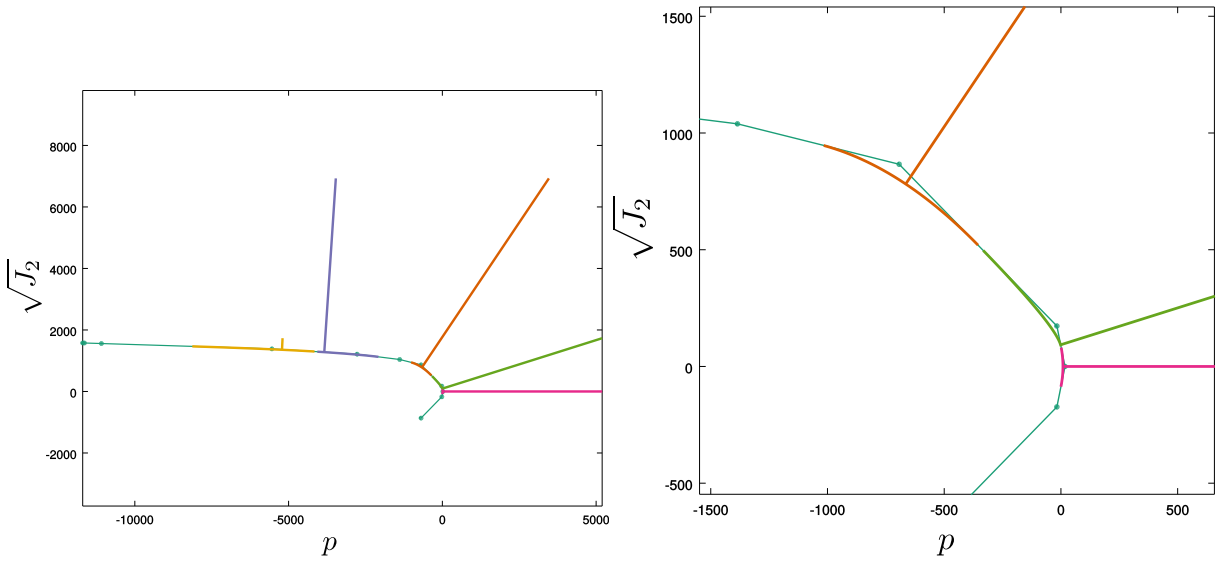
where $\mathbf{a} = (1, t, t^2)$, $t \in [0, 1]$ parameterizes each segment of the tabular data, $\mathbf{p}_x = (x_k, x_{k+1}, x_{k+2})$, $\mathbf{p}_y = (y_k, y_{k+1}, y_{k+2})$, and (x_k, y_k) are the input $0, \ldots, N-1$ tabular data points. The associated matrices that are used are:

$$\boldsymbol{M}_{j=0} = 0.5 \begin{bmatrix} 2 & 0 & 0 \\ -4 & 4 & 0 \\ 2 & -3 & 1 \end{bmatrix} \,, \quad \boldsymbol{M}_j = 0.5 \begin{bmatrix} 1 & 1 & 0 \\ -2 & 2 & 0 \\ 1 & -3 & 2 \end{bmatrix} \,, \quad \boldsymbol{M}_{j=N-1} = 0.5 \begin{bmatrix} 1 & 1 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{bmatrix} \tag{8.10}$$

Closest point projections of stress states outside the yield surface to fitted B-splines along the yield surface are shown in Figure 8.7.

8.4 Theory behind closest-point projection

The ideas behind the closest-point projection approach were made rigorous in the mid-to-late 1980s by a group of researchers influenced by developments in convex optimization.



(a) Closest points to yield function

(b) Zoomed view showing B-splines

Figure 8.7: Closest point projections to yield function used by the tabular plasticity.

8.4.1 Background

In nonlinear optimization, the method of Lagrange multipliers has been used since the mid 1800s to solve minimization problems with *equality* constraints. In 1950, this approach was generalized by Kuhn and Tucker to allow for *inequality* constraints. Later it was discovered that W. Karush from the University of Chicago had reached the same conclusions in his MSc thesis from 1939.

Primal form

The primal form of the optimization problem is

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{subject to} && g_i(\mathbf{x}) \le 0, \quad i = 1, \dots, m \\ &&& h_j(\mathbf{x}) = 0, \quad j = 1, \dots, p \end{aligned} \tag{8.11}$$

Note that there is no convexity requirement for this problem.

The Lagrangian

The Lagrangian ($\mathcal{L}$) associated with the primal form is just the weighted sum of the objective function f_0 and the constraint functions g_i and h_j. Thus

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f(\mathbf{x}) + \boldsymbol{\lambda} \cdot \mathbf{g}(\mathbf{x}) + \boldsymbol{\nu} \cdot \mathbf{h}(\mathbf{x}) \tag{8.12}$$

where

$$\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_m \end{bmatrix}, \quad \boldsymbol{\nu} = \begin{bmatrix} \nu_1 \\ \nu_2 \\ \vdots \\ \nu_p \end{bmatrix}, \quad \mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_p \end{bmatrix}. \tag{8.13}$$

The vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$ are called *Lagrange multiplier vectors* or, more frequently, the *dual variables* of the primal problem.

Dual function

The dual function ($\mathcal{F}(\boldsymbol{\lambda}, \boldsymbol{\nu})$) to the primal problem is defined as

$$\mathcal{F}(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\mathbf{x}} \left[f(\mathbf{x}) + \boldsymbol{\lambda} \cdot \mathbf{g}(\mathbf{x}) + \boldsymbol{\nu} \cdot \mathbf{h}(\mathbf{x}) \right] \tag{8.14}$$

Note that the dual function is the minimum of a family of affine functions (linear + a constant term) in $(\boldsymbol{\lambda}, \boldsymbol{\nu})$. This makes the dual problem concave. Note also that since the dual function is affine, it is bounded from below by $-\infty$ when the value of $\mathbf{x}$ is unbounded.

Simplified forms for $\mathcal{F}$ can be found for many problems, including problems that can be expressed as quadratic forms.

Dual form

Since the dual function is the largest lower bound on the Lagrangian, the *Lagrange dual form* of the primal minimization can be expressed as

$$\begin{aligned} &\text{maximize} \quad \mathcal{F}(\boldsymbol{\lambda}, \boldsymbol{\nu}) \\ &\text{subject to} \quad \boldsymbol{\lambda} \succeq \mathbf{o} \end{aligned} \tag{8.15}$$

We don't have any constraint on $\boldsymbol{\nu}$ because $\mathbf{h}(\mathbf{x}) = \mathbf{o}$.

Karush-Kuhn-Tucker optimality conditions

Let $\mathbf{x}^\star$ be the optimal solution for the primal problem and let $(\boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star)$ be the optimal solution of the dual problem. When these two solutions lead to a zero duality gap, i.e.,

$$f(\mathbf{x}^\star) = \mathcal{F}(\boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) \tag{8.16}$$

the Lagrangian at that optimal point is

$$\mathcal{L}(\mathbf{x}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) = f(\mathbf{x}^\star) + \boldsymbol{\lambda}^\star \cdot \mathbf{g}(\mathbf{x}^\star) + \boldsymbol{\nu}^\star \cdot \mathbf{h}(\mathbf{x}^\star) \tag{8.17}$$

Also, since $\boldsymbol{\lambda}^\star \geq \mathrm{o}$ and $\mathbf{h} = \mathrm{o}$,

$$f(\mathbf{x}^\star) = \mathcal{F}(\boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) = \inf_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) \leq \mathcal{L}(\mathbf{x}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) \leq f(\mathbf{x}^\star) \tag{8.18}$$

The only way for the above to be true is when

$$\boldsymbol{\lambda}^\star \cdot \mathbf{g}(\mathbf{x}^\star) = \mathrm{o} \quad \leftrightarrow \quad \lambda_i^\star g_i(\mathbf{x}^\star) = \mathrm{o}\,. \tag{8.19}$$

Also, since $\mathbf{x}^\star$ minimizes the Lagrangian, its gradient is zero at that point:

$$\frac{\partial}{\partial \mathbf{x}} \mathcal{L}(\mathbf{x}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) = \mathbf{o} = \frac{\partial f(\mathbf{x}^\star)}{\partial \mathbf{x}} + \boldsymbol{\lambda}^\star \cdot \frac{\partial \mathbf{g}(\mathbf{x}^\star)}{\partial \mathbf{x}} + \boldsymbol{\nu}^\star \cdot \frac{\partial \mathbf{h}(\mathbf{x}^\star)}{\partial \mathbf{x}} \tag{8.20}$$

These results, along with the original constraints of the primal and dual problems, are collected together into the *Karush-Kuhn-Tucker optimality conditions*:

$$\begin{aligned} &g_i(\mathbf{x}^\star) \leq \mathrm{o} && h_j(\mathbf{x}^\star) = \mathrm{o} \\ &\lambda_i^\star \geq \mathrm{o} && \lambda_i^\star g_i(\mathbf{x}^\star) = \mathrm{o} \\ &\frac{\partial f(\mathbf{x}^\star)}{\partial \mathbf{x}} + \boldsymbol{\lambda}^\star \cdot \frac{\partial \mathbf{g}(\mathbf{x}^\star)}{\partial \mathbf{x}} + \boldsymbol{\nu}^\star \cdot \frac{\partial \mathbf{h}(\mathbf{x}^\star)}{\partial \mathbf{x}} = \mathbf{o} \end{aligned} \tag{8.21}$$

8.4.2 Similarity with plasticity

The plastic loading-unloading conditions are similar to the Karush-Kush-Tucker optimality conditions in that we have

$$g(\boldsymbol{\sigma}) \le 0\,, \quad \dot{\lambda} \ge 0\,, \quad \dot{\lambda} g(\boldsymbol{\sigma}) = 0 \tag{8.22}$$

where $g(\boldsymbol{\sigma})$ is the yield surface constraining the values of $\boldsymbol{\sigma}$. We may also interpret the flow rule as the last Karush-Kuhn-Tucker condition:

$$-\dot{\boldsymbol{\varepsilon}}^p + \dot{\lambda}\frac{\partial g}{\partial \boldsymbol{\sigma}} = 0 \quad \text{where} \quad -\dot{\boldsymbol{\varepsilon}}^p =: \frac{\partial f}{\partial \boldsymbol{\sigma}} \tag{8.23}$$

and $f(\boldsymbol{\sigma})$ is the quantity that is minimized in the primal problem. We can interpret f as the negative of the maximum plastic dissipation, i.e.,

$$f(\boldsymbol{\sigma}) = -\boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}}^p\,. \tag{8.24}$$

If we use a first-order update approach, the discretized equations for perfect plasticity are

$$\begin{aligned}
&\boldsymbol{\sigma}_{n+1} = \mathbf{C} : (\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}^p_{n+1}) = \boldsymbol{\sigma}^{\text{trial}}_{n+1} - \mathbf{C} : (\boldsymbol{\varepsilon}^p_{n+1} - \boldsymbol{\varepsilon}^p_n)\\
&\boldsymbol{\varepsilon}^p_{n+1} = \boldsymbol{\varepsilon}^p_n + \Delta\lambda \left.\frac{\partial g}{\partial \boldsymbol{\sigma}}\right|_{\boldsymbol{\sigma}_n} \quad \text{or} \quad \boldsymbol{\varepsilon}^p_{n+1} = \boldsymbol{\varepsilon}^p_n + \Delta\lambda \left.\frac{\partial g}{\partial \boldsymbol{\sigma}}\right|_{\boldsymbol{\sigma}_{n+1}}\\
&\quad g(\boldsymbol{\sigma}_{n+1}) \le 0\,, \quad \Delta\lambda \ge 0\,, \quad \Delta\lambda g(\boldsymbol{\sigma}_{n+1}) = 0
\end{aligned} \tag{8.25}$$

> Note that if we interpret the flow rule as an optimality condition a backward Euler update is consistent with the Karush-Kuhn-Tucker conditions and a forward Euler update is ruled out.

8.4.3 Closest point return

Let $\boldsymbol{\sigma}^{\text{trial}}$ be the trial stress and let $g(\boldsymbol{\sigma}^{\text{trial}})$ be the value of the yield function at that state. Let $\boldsymbol{\sigma}_{n+1}$ be actual stress and let $g(\boldsymbol{\sigma}_{n+1}) = 0$ be the value of the yield function at the actual stress state.

Let us assume the actual stress state on the yield surface is at the closest distance from the trial stress. Then we can devise the primal minimization problem:

$$\begin{aligned}
&\text{minimize} \quad f(\boldsymbol{\sigma}) = \|\boldsymbol{\sigma}^{\text{trial}} - \boldsymbol{\sigma}\|^2\\
&\text{subject to} \quad g(\boldsymbol{\sigma}) \le 0
\end{aligned} \tag{8.26}$$

where

$$\|\boldsymbol{\sigma}\| = \sqrt{\boldsymbol{\sigma} : \boldsymbol{\sigma}} \tag{8.27}$$

The Lagrangian for this problem is

$$\mathcal{L}(\boldsymbol{\sigma}, \lambda) = f(\boldsymbol{\sigma}) + \Delta\lambda g(\boldsymbol{\sigma}) = \|\boldsymbol{\sigma}^{\text{trial}} - \boldsymbol{\sigma}\|^2 + \Delta\lambda g(\boldsymbol{\sigma}) \tag{8.28}$$

The Karush-Kuhn-Tucker conditions for this problem at the optimum value $\boldsymbol{\sigma}_{n+1}$ are

$$\begin{aligned}
&g(\boldsymbol{\sigma}_{n+1}) \le 0\,, \quad \Delta\lambda \ge 0\,, \quad \Delta\lambda g(\boldsymbol{\sigma}_{n+1}) = 0\\
&\frac{\partial f(\boldsymbol{\sigma}_{n+1})}{\partial \boldsymbol{\sigma}} + \Delta\lambda \frac{\partial g(\boldsymbol{\sigma}_{n+1})}{\partial \boldsymbol{\sigma}} = -2(\boldsymbol{\sigma}^{\text{trial}} - \boldsymbol{\sigma}_{n+1}) + \Delta\lambda \frac{\partial g(\boldsymbol{\sigma}_{n+1})}{\partial \boldsymbol{\sigma}} = \mathbf{0}
\end{aligned} \tag{8.29}$$

From the last condition we see that the closest distance using this criterion leads to a stress value of

$$\boldsymbol{\sigma}_{n+1} = \boldsymbol{\sigma}^{\text{trial}} - \tfrac{1}{2}\Delta\lambda \frac{\partial g(\boldsymbol{\sigma}_{n+1})}{\partial \boldsymbol{\sigma}} \tag{8.30}$$

But we have seen previously that the first-order stress update with backward Euler leads to

$$\boldsymbol{\sigma}_{n+1} = \boldsymbol{\sigma}^{\text{trial}} - \Delta\lambda \mathbf{C} : \frac{\partial g(\boldsymbol{\sigma}_{n+1})}{\partial \boldsymbol{\sigma}} \tag{8.31}$$

The similarity between the two indicates that we are on the right track, i.e., the actual stress is at the closest distance from the trial stress to the yield surface. But the correct closest distance is not in the standard standard stress space, but in a space where the norm to be minimized is given by

$$\|\boldsymbol{\sigma}\|_{\mathbf{C}^{-1}} = \sqrt{\boldsymbol{\sigma} : \mathbf{C}^{-1} : \boldsymbol{\sigma}} \tag{8.32}$$

This can be verified by repeating the above exercise with the new definition of the norm. More specifically, the correct updated stress is at the shortest distance from the trial stress to the yield surface in a 9-dimensional space that has the Euclidean distance measure

$$\|\boldsymbol{\sigma}\|_{\mathsf{C}^{-1}} = \sqrt{\boldsymbol{\sigma} : \mathsf{C}^{-1} : \boldsymbol{\sigma}} \tag{8.33}$$

where C is the stiffness tensor. We will explore some of the implications of this idea in this article.

Note that this particular closest-point interpretation applies only for *perfect plasticity* and only *associative* flow rules. For hardening plasticity, the space in which the actual stress is closest to the trial stress is different. For non-associative plasticity, it is unclear whether any closest-point approach can be rigorously justified.

8.4.4 Eigendecompositions in linear elasticity

The stiffness tensor for an isotropic elastic material is

$$\mathsf{C} = \lambda\, \mathbf{I} \otimes \mathbf{I} + 2\mu\, \mathsf{I}^s \tag{8.34}$$

where λ, μ are the Lamé elastic constants, $\mathbf{I}$ is the rank-2 identity tensor, and I^s is the symmetric rank-4 identity tensor. The inverse of C is the compliance tensor

$$\mathsf{C}^{-1} = \mathsf{S} = -\frac{\lambda}{2\mu(3\lambda + 2\mu)}\, \mathbf{I} \otimes \mathbf{I} + \frac{1}{2\mu}\, \mathsf{I}^s \tag{8.35}$$

Eigendecompositions of the stiffness and compliance tensors are defined via

$$\mathsf{C} : \mathbf{V} = \lambda \mathbf{V}\,, \quad \mathsf{S} : \mathbf{V} = \frac{1}{\lambda}\mathbf{V} \tag{8.36}$$

where λ are the eigenvalues (not to be confused with the Lamé modulus) and $\mathbf{V}$ are rank-2 tensors that form the eigenbasis. Because of the symmetries of the stiffness matrix, there are six or less unique eigenvalues and the corresponding eigentensors are orthogonal, i.e.,

$$\mathbf{V}_i : \mathbf{V}_i = 1 \quad \text{and} \quad \mathbf{V}_i : \mathbf{V}_j = 0\,. \tag{8.37}$$

The stiffness and compliance tensors may then be represented as:

$$\mathsf{C} = \sum_{i=1}^{m} \lambda_i \mathbf{V}_i \otimes \mathbf{V}_i\,, \quad \mathsf{S} = \sum_{i=1}^{m} \frac{1}{\lambda_i} \mathbf{V}_i \otimes \mathbf{V}_i \tag{8.38}$$

where m is the number of non-zero and distinct eigenvalues. Note also that, in this eigenbasis, the symmetric rank-4 identity tensor is

$$\mathsf{I}^s = \sum_{i=1}^{m} \mathbf{V}_i \otimes \mathbf{V}_i\,. \tag{8.39}$$

Eigenprojectors are defined as rank-4 tensors that have the property (for $i \neq j$)

$$\mathsf{P}_i : \mathbf{V}_i = \mathbf{V}_i \,, \quad \mathsf{P}_i : \mathbf{V}_j = \mathbf{o} \tag{8.40}$$

If we apply the eigenprojector to the rank-4 identity tensor, we get

$$\mathsf{P}_k = \mathsf{P}_k : \mathsf{I}^s = \sum_{i=1}^{m} \mathsf{P}_k : (\mathbf{V}_i \otimes \mathbf{V}_i) = \sum_{i=1}^{m} (\mathsf{P}_k : \mathbf{V}_i) \otimes \mathbf{V}_i = (\mathsf{P}_k : \mathbf{V}_k) \otimes \mathbf{V}_k = \mathbf{V}_k \otimes \mathbf{V}_k \tag{8.41}$$

Therefore we may also write the eigendecomposition in terms of the eigenprojectors

$$\mathsf{C} = \sum_{i=1}^{m} \lambda_i \mathsf{P}_i \,, \quad \mathsf{S} = \sum_{i=1}^{m} \frac{1}{\lambda_i} \mathsf{P}_i \,, \quad \mathsf{I}^s = \sum_{i=1}^{m} \mathsf{P}_i \,. \tag{8.42}$$

For isotropic materials, a small amount of algebra shows that there are two unique eigenvectors which lead to the decomposition

$$\mathsf{C} = \lambda_1 \mathsf{P}_1 + \lambda_2 \mathsf{P}_2 \quad \text{where} \quad \mathsf{S} = \frac{1}{\lambda_1} \mathsf{P}_1 + \frac{1}{\lambda_2} \mathsf{P}_2 \tag{8.43}$$

and

$$\mathsf{P}_1 = \tfrac{1}{3} \mathbf{I} \otimes \mathbf{I} \,, \quad \mathsf{P}_2 = \mathsf{I}^s - \mathsf{P}_1 \,. \tag{8.44}$$

We can now express the stiffness and compliance tensors in terms of these eigenprojections:

$$\begin{aligned} \mathsf{C} &= \left(\kappa - \tfrac{2}{3} \mu \right) \mathbf{I} \otimes \mathbf{I} + 2\mu \, \mathsf{I}^s \\ &= 3\kappa \left(\tfrac{1}{3} \mathbf{I} \otimes \mathbf{I} \right) + 2\mu \left(\mathsf{I}^s - \tfrac{1}{3} \mathbf{I} \otimes \mathbf{I} \right) \end{aligned} \tag{8.45}$$

where κ is the bulk modulus and μ is the shear modulus. Also,

$$\begin{aligned} \mathsf{S} &= \frac{1}{3} \left(\frac{1}{3\kappa} - \frac{1}{2\mu} \right) \mathbf{I} \otimes \mathbf{I} + \frac{1}{2\mu} \mathsf{I}^s \\ &= \frac{1}{3\kappa} \left(\tfrac{1}{3} \mathbf{I} \otimes \mathbf{I} \right) + \frac{1}{2\mu} \left(\mathsf{I}^s - \tfrac{1}{3} \mathbf{I} \otimes \mathbf{I} \right) \end{aligned} \tag{8.46}$$

Therefore, we can write

$$\mathsf{C} = 3\kappa \, \mathsf{P}^{\text{iso}} + 2\mu \, \mathsf{P}^{\text{symdev}} \quad \text{and} \quad \mathsf{S} = \frac{1}{3\kappa} \mathsf{P}^{\text{iso}} + \frac{1}{2\mu} \mathsf{P}^{\text{symdev}} \tag{8.47}$$

where $\mathsf{P}^{\text{iso}} = \mathsf{P}_1$ and $\mathsf{P}^{\text{symdev}} = \mathsf{P}_2$.

It is also worth noting that if

$$\mathsf{C}^{1/2} : \mathsf{C}^{1/2} := \mathsf{C} \quad \text{and} \quad \mathsf{S}^{1/2} : \mathsf{S}^{1/2} := \mathsf{S} \tag{8.48}$$

then, using the property that $\mathsf{P}_1 : \mathsf{P}_2 = 0$,

$$\mathsf{C}^{1/2} = \sqrt{\lambda_1} \mathsf{P}_1 + \sqrt{\lambda_2} \mathsf{P}_2 \quad \text{where} \quad \mathsf{S}^{1/2} = \frac{1}{\sqrt{\lambda_1}} \mathsf{P}_1 + \frac{1}{\sqrt{\lambda_2}} \mathsf{P}_2 \tag{8.49}$$

In that case, we have

$$\mathsf{C}^{1/2} = \sqrt{3\kappa} \, \mathsf{P}^{\text{iso}} + \sqrt{2\mu} \, \mathsf{P}^{\text{symdev}} \quad \text{and} \quad \mathsf{S}^{1/2} = \frac{1}{\sqrt{3\kappa}} \mathsf{P}^{\text{iso}} + \frac{1}{\sqrt{2\mu}} \mathsf{P}^{\text{symdev}} \tag{8.50}$$

8.4.5 The transformed space for isotropic linear elasticity

Details of the transformed space for isotropic linear elasticity were worked out by M. Homel in his 2014 PhD dissertation. We will follow his approach in this section.

The distance measure

$$\|\boldsymbol{\sigma}\|_{\mathsf{S}} = \sqrt{\boldsymbol{\sigma} : \mathsf{S} : \boldsymbol{\sigma}} \tag{8.51}$$

can be interpreted as a standard Euclidean distance measure in a transformed stress space by observing that

$$\begin{aligned} \|\boldsymbol{\sigma}\|_{\mathsf{S}} &= \sqrt{(\boldsymbol{\sigma} : \mathsf{S}^{1/2}) : (\mathsf{S}^{1/2} : \boldsymbol{\sigma})} \quad \text{where} \quad \mathsf{S}^{1/2} : \mathsf{S}^{1/2} := \mathsf{S} \\ &= \sqrt{(\mathsf{S}^{1/2} : \boldsymbol{\sigma}) : (\mathsf{S}^{1/2} : \boldsymbol{\sigma})} \quad \text{using the major symmetry of} \quad \mathsf{S} \\ &= \sqrt{\boldsymbol{\sigma}^\star : \boldsymbol{\sigma}^\star} = \|\boldsymbol{\sigma}^\star\| \end{aligned} \tag{8.52}$$

We would like to calculate the transformed stress tensor.

The Lode invariants and the Lode basis

The Lode basis (described by R. M. Brannon in 2009) is an alternative basis that can be used to decompose the stress tensor. Let us define the following deviatoric quantities:

$$\boldsymbol{s} = \operatorname{dev}(\boldsymbol{\sigma}) = \boldsymbol{\sigma} - \tfrac{1}{3}\operatorname{tr}(\boldsymbol{\sigma})\mathbf{I} \quad \text{and} \quad \boldsymbol{t} = \operatorname{dev}(\boldsymbol{s}\cdot\boldsymbol{s}) = \boldsymbol{s}\cdot\boldsymbol{s} - \tfrac{1}{3}\operatorname{tr}(\boldsymbol{s}\cdot\boldsymbol{s})\mathbf{I} \tag{8.53}$$

The quantity $\boldsymbol{t}$ is also called the *Hill tensor*.

The Lode invariants of a stress tensor are

$$z = \tfrac{1}{\sqrt{3}}\operatorname{tr}(\boldsymbol{\sigma})\,, \quad r = \|\boldsymbol{s}\|\,, \quad \sin 3\theta = 3\sqrt{6}\det\left(\frac{\boldsymbol{s}}{\|\boldsymbol{s}\|}\right) \tag{8.54}$$

These invariants are associated with an orthonormal set of unit tensors

$$\mathbf{E}_z = \tfrac{1}{\sqrt{3}}\mathbf{I}\,, \quad \mathbf{E}_r = \frac{\boldsymbol{s}}{\|\boldsymbol{s}\|}\,, \quad \mathbf{E}_\theta = \frac{\frac{\boldsymbol{t}}{\|\boldsymbol{t}\|} - \sin 3\theta\,\frac{\boldsymbol{s}}{\|\boldsymbol{s}\|}}{\cos 3\theta} \tag{8.55}$$

The stress can be expressed in terms of the Lode basis as

$$\boldsymbol{\sigma} = z\,\mathbf{E}_z + r\,\mathbf{E}_r\,. \tag{8.56}$$

The transformed stress tensor

We can now compute the transformed stress tensor:

$$\boldsymbol{\sigma}^\star = \mathsf{S}^{1/2} : \boldsymbol{\sigma} = \left[\frac{1}{\sqrt{3\kappa}}\mathsf{P}^{\text{iso}} + \frac{1}{\sqrt{2\mu}}\mathsf{P}^{\text{symdev}}\right] : (z\,\mathbf{E}_z + r\,\mathbf{E}_r)\,. \tag{8.57}$$

We can show that

$$\mathsf{P}^{\text{iso}} : \mathbf{E}_z = \mathbf{E}_z\,, \quad \mathsf{P}^{\text{iso}} : \mathbf{E}_r = \mathbf{o}\,, \quad \mathsf{P}^{\text{symdev}} : \mathbf{E}_z = \mathbf{o}\,, \quad \mathsf{P}^{\text{symdev}} : \mathbf{E}_r = \mathbf{E}_r \tag{8.58}$$

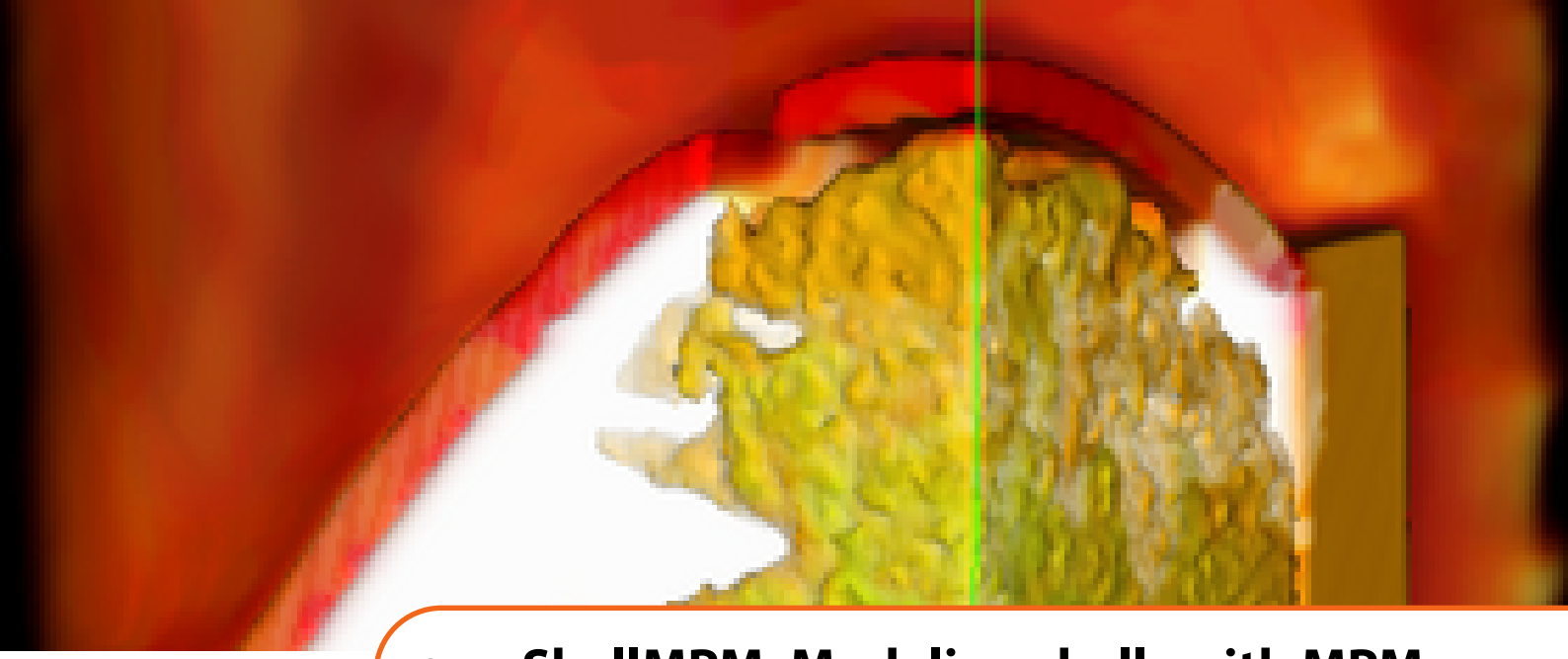
Therefore,

$$\boldsymbol{\sigma}^\star = \frac{z}{\sqrt{3\kappa}}\mathbf{E}_z + \frac{r}{\sqrt{2\mu}}\mathbf{E}_r \tag{8.59}$$

We can also show that the transformed stress vector remains geometrically unchanged (in the sense that angles are unchanged) if we express it as

$$\boldsymbol{\sigma}^{\star} = z\,\mathbf{E}_z + \sqrt{\frac{3\kappa}{2\mu}}\,r\,\mathbf{E}_r =: z\,\mathbf{E}_z + r'\,\mathbf{E}_r \tag{8.60}$$

So we have a straightforward way of computing stresses in the transformed space and use this idea in the geometrical closest point return algorithm.



9 — ShellMPM: Modeling shells with MPM

9.1 Shell theory

The continuum-based approach to shell theory has been chosen because of the relative ease of implementation of constitutive models in this approach compared to exact geometrical descriptions of the shell. In order to include transverse shear strains in the shell, a modified Reissner-Mindlin assum The major assumptions of the shell formulation are [68, 69]

1. The normal to the mid-surface of the shell remains straight but not necessarily normal. The direction of the initial normal is called the "fiber" direction and it is the evolution of the fiber that is tracked.
2. The stress normal to the mid-surface vanishes (plane stress)
3. The momentum due to the extension of the fiber and the momentum balance in the direction of the fiber are neglected.
4. The curvature of the shell at a material point is neglected.

The shell formulation is based on a plate formulation by Lewis et al. [69]. A discussion of the formulation follows.

The velocity field in the shell is given by

$$\mathbf{w}(\alpha,\beta) = \mathbf{u}(\alpha,\beta) + z\,\boldsymbol{\omega}(\alpha,\beta)\times\mathbf{n}(\alpha,\beta) + \dot{z}\,\mathbf{n}(\alpha,\beta) \tag{9.1}$$

where $\mathbf{w}$ is the velocity of a point in the shell, $\mathbf{u}$ is the velocity of the center of mass of the shell, $\mathbf{n}$ is the normal or director vector, $\boldsymbol{\omega}$ is the angular velocity of the director, (α,β) are orthogonal co-ordinates on the mid-surface of the shell, z is the perpendicular distance from the mid-surface of the shell, and $\dot{z}$ is the rate of change of the length of the shell director.

Since momentum balance is not enforced for the motion in the direction of the director $\mathbf{n}$, the terms involving $\dot{z}$ are dropped in constructing the equations of motion. These terms are also omitted in the deformation gradient calculation. However, the thickness change in the shell is not neglected in the computation of internal forces and moments. Equation (9.1) can therefore be written as

$$\mathbf{w}(\alpha,\beta) = \mathbf{u}(\alpha,\beta) + z\,\mathbf{r}(\alpha,\beta) \tag{9.2}$$

where $\mathbf{r}$, the rotation rate of $\mathbf{n}$, is a vector that is perpendicular to $\mathbf{n}$.

The velocity gradient tensor for $\mathbf{w}$ is used to compute the stresses in the shell. If the curvature of the shell

is neglected, i.e., the shell is piecewise plane, the velocity gradient tensor for $\mathbf{w}$ can be written as

$$\nabla \mathbf{w} = \left[\nabla^{(s)} \mathbf{u} + z \, \nabla^{(s)} \mathbf{r} \right] + \mathbf{r} \otimes \mathbf{n} \tag{9.3}$$

where $\mathbf{r} \otimes \mathbf{n}$ represents the dyadic product, and $\nabla^{(s)}$ is the in-surface gradient operator, defined as,

$$\nabla^{(s)} = [\nabla(\ )] \bullet \mathbf{I}^{(s)} \, . \tag{9.4}$$

The $\bullet$ represents a tensor inner product and $\mathbf{I}^{(s)}$ is the in-surface identity tensor (or the projection operator), defined as,

$$\mathbf{I}^{(s)} = \mathbf{I} - \mathbf{n} \otimes \mathbf{n}. \tag{9.5}$$

It should be noted that, for accuracy, the vector $\mathbf{n}$ should not deviate significantly from the actual normal to the surface (i.e., the transverse shear strains should be small).

The determination of the shell velocity tensor $\nabla \mathbf{w}$ requires the determination of the center of mass velocity $\mathbf{u}$ of the shell. This quantity is determined using the balance of linear momentum in the shell. The local three-dimensional equation of motion for the shell is, in the absence of body forces,

$$\nabla \bullet \boldsymbol{\sigma} = \rho \, \mathbf{a} \tag{9.6}$$

where **sigma** is the stress tensor, ρ is the density of the shell material, and $\mathbf{a}$ is the acceleration of the shell. The two-dimensional form of the linear momentum balance equation (9.6) with respect to the surface of the shell is given by

$$\nabla^{(s)} \bullet \langle \boldsymbol{\sigma} \rangle = \rho \, \mathbf{a} \, . \tag{9.7}$$

The acceleration of the material points in the shell are now due to the in-surface divergence of the average stress $\langle \boldsymbol{\sigma} \rangle$ in the shell, given by

$$\langle \boldsymbol{\sigma} \rangle := \frac{1}{h} \int_{-h^-}^{h^+} \boldsymbol{\sigma}(z) \, dz \tag{9.8}$$

where h^+ is the "thickness" of the shell (along the director) from the center of mass to the "top" of the shell, h^- is the thickness from the center of mass to the "bottom" of the shell, and $h = h^+ + h^-$. The point of departure from the formulation of Lewis et al. [69] is that instead of separate linear momentum balance laws for shell and non-shell materials, a single global momentum balance is used and the "plane stress" condition $\sigma_{zz} = 0$ is enforced in the shell stress update, where the subscript zz represents the direction of the shell director.

The shell director $\mathbf{n}$ and its rotation rate $\mathbf{r}$ also need to be known before the shell velocity gradient tensor $\nabla \mathbf{w}$ can be determined. These quantities are determined using an equation for the conservation of angular momentum [70], given by

$$\nabla^{(s)} \bullet \mathbf{M} - \mathbf{n} \bullet \langle \boldsymbol{\sigma} \rangle \bullet \mathbf{I}^{(s)} = \frac{1}{12} \rho \, h^2 \, \dot{\mathbf{r}} \tag{9.9}$$

where $\dot{\mathbf{r}}$ is the rotational acceleration of $\mathbf{n}$, ρ is the density of the shell material, and $\mathbf{M}$ is the average moment, defined as

$$\mathbf{M} := \mathbf{I}^{(s)} \bullet \left[\frac{1}{h} \int_{-h^-}^{h^+} \boldsymbol{\sigma}(z) \, z \, dz \right] \bullet \mathbf{I}^{(s)} \, . \tag{9.10}$$

The center-of-mass velocity $\mathbf{u}$, the director $\mathbf{n}$ and its rate of rotation $\mathbf{r}$ provide a means to obtain the velocity of material points on the shell. The shell is divided into a number of layers with discrete values of z and the layer-wise gradient of the shell velocity is used to compute the stress and deformation in each layer of the shell.

9.2 Shell Implementation for the Material Point Method

The shell description given in the previous section has been implemented such that the standard steps of the material point method [1] remain the same for all materials. Some additional steps are performed for shell materials. These steps are encapsulated within the shell constitutive model.

The steps involved for each time increment Δt are discussed below. The superscript n represents the value of the state variables at time $n\ \Delta t$ while the superscript $n+1$ represents the value at time $(n+1)\ \Delta t$. Note that Δt need not necessarily be constant. In the following, the subscript p is used to index material point variables while the subscript v is used to index grid vertex variables. The notation $\sum_p$ denotes summation over material points and $\sum_v$ denotes summation over grid vertices. Zeroth order interpolation functions associated with each material point are denoted by $S^{(0)}_{p,v}$ while first order interpolation functions are denoted by $S^{(1)}_{p,v}$.

9.2.1 Interpolate state data from material points to the grid.

The state variables are interpolated from the material points to the grid vertices using the contiguous generalized interpolation material point (GIMP) method [71]. In the GIMP method material points are defined by particle characteristic functions $\chi_p(\mathbf{x})$ which are required to be a partition of unity,

$$\sum_p \chi_p(\mathbf{x}) = 1 \quad \forall \quad \mathbf{x} \in \Omega \tag{9.11}$$

where $\mathbf{x}$ is the position of a point in the body Ω. A continuous representation of the property $f(\mathbf{x})$ is given by

$$f(\mathbf{x}) = \sum_p f_p\ \chi_p(\mathbf{x}) \tag{9.12}$$

where f_p is the value at a material point. Similarly, a continuous representation of the grid data is given by

$$g(\mathbf{x}) = \sum_v g_v\ S_v(\mathbf{x}) \tag{9.13}$$

where

$$\sum_v S_v(\mathbf{x}) = 1 \quad \forall \quad \mathbf{x} \in \Omega\ . \tag{9.14}$$

To interpolate particle data to the grid, the interpolation (or weighting functions) $S^{(1)}_{p,v}$ are used, which are defined as

$$S^{(1)}_{p,v} = \frac{1}{V_p} \int_{\Omega_p \cap \Omega} \chi_p(\mathbf{x})\ S_v(\mathbf{x})\ d\mathbf{x} \tag{9.15}$$

where V_p is the volume associated with a material point, Ω_p is the region of non-zero support for the material point, and

$$\sum_v S^{(1)}_{p,v} = 1 \quad \forall \quad \mathbf{x_p} \in \Omega_p\ . \tag{9.16}$$

The state variables that are interpolated to the grid in this step are the mass (m), momentum ($m\mathbf{u}$), volume (V), external forces ($\mathbf{f}^{\text{ext}}$), temperature (T), and specific volume (v) using relations of the form

$$m_v = \sum_p m_p\ S^{(1)}_{p,v}\ . \tag{9.17}$$

In our computations, bilinear hat functions S_v were used that lead to interpolation functions $S^{(1)}_{p,v}$ with non-zero support in adjacent grid cells and in the next nearest neighbor grid cells. Details of these functions can be found in reference [71].

For shell materials, an additional step is required to inhabit the grid vertices with the interpolated normal rotation rate from the particles. However, instead of interpolating the angular momentum, the quantity $\mathbf{p}_p = m_p \mathbf{r}_p$ is interpolated to the grid using the relation

$$\mathbf{p}_v = \sum_p \mathbf{p}_p \, S^{(1)}_{p,v} \, . \tag{9.18}$$

At the grid, the rotation rate is recovered using

$$\mathbf{r}_v = \mathbf{p}_v / m_v \tag{9.19}$$

This approximation is required because the moment of inertia contains h^2 terms which can be very small for thin shells. Floating point errors are magnified when m_p is multiplied by h^2. In addition, it is not desirable to interpolate the plate thickness to the grid.

9.2.2 Compute heat and momentum exchange due to contact.

In this step, any heat and momentum exchange between bodies inside the computational domain is performed through the grid. Details of contact algorithms used my the material point method can be found in references [1, 6, 13]. subsection Compute the stress tensor. The stress tensor computation follows the procedure for hyperelastic materials cited in reference [21]. However, some extra steps are required for shell materials. The stress update is performed using a forward Euler explicit time stepping procedure. The velocity gradient $\nabla \mathbf{w}$ at a material point is required for the stress update. This quantity is determined using equation (9.3). The velocity gradient of the center of mass of the shell ($\nabla \mathbf{u}$) is computed from the grid velocities using gradient weighting functions of the form

$$\nabla S^{(1)}_{p,v} = \frac{1}{V_p} \int_{\Omega_p \cap \Omega} \chi_p(\mathbf{x}) \, \nabla S_v(\mathbf{x}) \, d\mathbf{x} \tag{9.20}$$

so that

$$\nabla \mathbf{u}_p = \sum_v \mathbf{u}_v \, \nabla S^{(1)}_{p,v} \, . \tag{9.21}$$

The gradient of the rotation rate ($\nabla \mathbf{r}$) is also interpolated to the particles using the same procedure, i.e.,

$$\nabla \mathbf{r}_p = \sum_v \mathbf{r}_v \, \nabla S^{(1)}_{p,v} \, . \tag{9.22}$$

The next step is to calculate the in-surface gradients $\nabla^{(s)} \mathbf{u}_p$ and $\nabla^{(s)} \mathbf{r}_p$. These are calculated as

$$\nabla^{(s)} \mathbf{u}_p = \nabla \mathbf{u}_p \bullet \left(\mathbf{I} - \mathbf{n}^n_p \otimes \mathbf{n}^n_p \right) \tag{9.23}$$

$$\nabla^{(s)} \mathbf{r}_p = \nabla \mathbf{r}_p \bullet \left(\mathbf{I} - \mathbf{n}^n_p \otimes \mathbf{n}^n_p \right) \tag{9.24}$$

The superscript n represents the values at the end of the n-th time step. The shell is now divided into a number of layers with different values of z (these can be considered to be equivalent to Gauss points to be used in the integration over z). The number of layers depends on the requirements of the problem. Three layers are used to obtain the results that follow. The velocity gradient $\nabla \mathbf{w}_p$ is calculated for each of the layers using equation (9.3). For a shell with three layers (top, center and bottom), the velocity gradients are given by

$$\nabla \mathbf{w}^{\text{top}}_p = \left[\nabla^{(s)} \mathbf{u}_p + h^+ \, \nabla^{(s)} \mathbf{r}_p \right] + \mathbf{r}^n_p \otimes \mathbf{n}^n_p \tag{9.25}$$

$$\nabla \mathbf{w}^{\text{cen}}_p = \nabla^{(s)} \mathbf{u}_p + \mathbf{r}^n_p \otimes \mathbf{n}^n_p \tag{9.26}$$

$$\nabla \mathbf{w}^{\text{bot}}_p = \left[\nabla^{(s)} \mathbf{u}_p - h^- \, \nabla^{(s)} \mathbf{r}_p \right] + \mathbf{r}^n_p \otimes \mathbf{n}^n_p \tag{9.27}$$

The increment of deformation gradient ($\Delta\mathbf{F}$) in each layer is computed using

$$\Delta\mathbf{F}_p = \Delta t\ \nabla\mathbf{w}_p + \mathbf{I} \tag{9.28}$$

The total deformation gradient ($\mathbf{F}$) in each layer is updated using

$$\tilde{\mathbf{F}}_p^{n+1} = \Delta\mathbf{F}_p \bullet \mathbf{F}_p^n \tag{9.29}$$

where $\tilde{\mathbf{F}}_p^{n+1}$ is the intermediate updated deformation gradient prior to application of the "plane stress" condition.

The stress in the shell is computed using a stored energy function (W) of the form

$$W = \frac{1}{2}K\left[\frac{1}{2}(J^2 - 1) - \ln J\right] + \frac{1}{2}G\left[\mathrm{tr}(\bar{\mathbf{b}}) - 3\right] \tag{9.30}$$

where K is the bulk modulus, G is the shear modulus, J is the Jacobian ($J = \det\mathbf{F}$), and $\bar{\mathbf{b}}$ is the volume preserving part of the left Cauchy-Green strain tensor, defined as

$$\bar{\mathbf{b}} := J^{-\frac{2}{3}}\mathbf{F} \bullet \mathbf{F}^T \tag{9.31}$$

The Cauchy stress then has the form

$$\boldsymbol{\sigma} = \frac{1}{2}K\left(J - \frac{1}{J}\right)\mathbf{I} + \frac{G}{J}\left[\bar{\mathbf{b}} - \frac{1}{3}\mathrm{tr}(\bar{\mathbf{b}})\right]. \tag{9.32}$$

The "plane stress" condition in the thickness direction of the shell is applied at this stage using an iterative Newton method. To apply this condition, the deformation gradient tensor has to be rotated such that its (33) component is aligned with the (zz) direction of the shell. The rotation tensor is the one required to rotate the vector $\mathbf{e}_3 \equiv (0, 0, 1)$ to the direction $\mathbf{n}_p^n$ about the vector $\mathbf{e}_3 \times \mathbf{n}_p^n$. If θ is the angle of rotation and $\mathbf{a}$ is the unit vector along axis of rotation, the rotation tensor is given by (using the derivative of the Euler-Rodrigues formula)

$$\mathbf{R} = \cos\theta\,(\mathbf{I} - \mathbf{a}\otimes\mathbf{a}) + \mathbf{a}\otimes\mathbf{a} - \sin\theta\,\mathbf{A} \tag{9.33}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}. \tag{9.34}$$

The rotated deformation gradient in each layer is given by

$$\mathbf{F}_p^{\mathrm{rot}} = \mathbf{R} \bullet \tilde{\mathbf{F}}_p^{n+1} \bullet \mathbf{R}^T. \tag{9.35}$$

The updated stress ($\boldsymbol{\sigma}_p^{\mathrm{rot}}$) is calculated in this rotated coordinate system using equation (9.32). Thus,

$$\boldsymbol{\sigma}_p^{\mathrm{rot}} = \frac{1}{2}K\left(J_p^{\mathrm{rot}} - \frac{1}{J_p^{\mathrm{rot}}}\right)\mathbf{I} + \frac{G}{J_p^{\mathrm{rot}}}\left[\bar{\mathbf{b}}_p^{\mathrm{rot}} - \frac{1}{3}\mathrm{tr}(\bar{\mathbf{b}}_p^{\mathrm{rot}})\right]. \tag{9.36}$$

An iterative Newton method is used to determine the deformation gradient component F_{33} for which the stress component σ_{33} is zero. The "plane stress" deformation gradient is denoted $\overset{\circ}{\mathbf{F}}$ and the stress is denoted $\overset{\circ}{\boldsymbol{\sigma}}$.

At this stage, the updated thickness of the shell at a material point is calculated from the relations

$$h_{n+1}^{+} = h_0^{+}\int_0^1 \overset{\circ}{F}_{zz}(+z)\,dz \tag{9.37}$$

$$h_{n+1}^{-} = h_0^{-}\int_0^1 \overset{\circ}{F}_{zz}(-z)\,dz \tag{9.38}$$

where h_o^+ and h_o^- are the initial values, and h_{n+1}^+ and h_{n+1}^- are the updated values, of h^+ and h^-, respectively.

In the next step, the deformation gradient and stress values for all the layers at each material point are rotated back to the original coordinate system. The updated Cauchy stress and deformation gradient are

$$\mathbf{F}_p^{n+1} = \mathbf{R}^T \bullet \overset{\circ}{\mathbf{F}} \bullet \mathbf{R} \tag{9.39}$$

$$\boldsymbol{\sigma}_p^{n+1} = \mathbf{R}^T \bullet \overset{\circ}{\boldsymbol{\sigma}} \bullet \mathbf{R} \,. \tag{9.40}$$

The deformed volume of the shell is approximated using the Jacobian of the deformation gradient at the center of mass of the shell

$$V_p^{n+1} = V_p^o \, J_p^{n+1} \,. \tag{9.41}$$

9.2.3 Compute the internal force and moment.

The internal force for general materials is computed at the grid using the relation

$$\mathbf{f}_v^{\text{int}} = \sum_p \left[\boldsymbol{\sigma}_p^{n+1} \bullet \nabla S_{p,v}^{(1)} \right] V_p^{n+1} \tag{9.42}$$

For shell materials, this relation takes the form

$$\mathbf{f}_v^{\text{int}} = \sum_p \left[\left\langle \boldsymbol{\sigma}_p^{n+1} \right\rangle \bullet \nabla S_{p,v}^{(1)} \right] V_p^{n+1} \tag{9.43}$$

In addition to internal forces, the formulation for shell materials requires the computation of internal moments in order to solve for the rotational acceleration in the rotational inertia equation (9.9). To obtain the discretized form of equation (9.9), the equation is integrated over the volume of the shell leading to [69]

$$-\sum_p \left[\left(\mathbf{M}_p \bullet \nabla S_{p,v}^{(1)} \bullet \mathbf{I}^{(s)} \right) + \left(\mathbf{n}_p \bullet \left\langle \boldsymbol{\sigma}_p \right\rangle \bullet \mathbf{I}^{(s)} \right) S_{p,v}^{(o)} \right] V_p = \left(\frac{1}{12} \sum_p S_{p,v}^{(o)} \, m_p \, h_p^2 \right) \dot{\mathbf{r}}_v \,. \tag{9.44}$$

The average stress over the thickness of the shell is calculated using equation (9.8) and the average moment is calculated using equation (9.10). The trapezoidal rule is used in both cases. Thus,

$$\left\langle \boldsymbol{\sigma}_p^{n+1} \right\rangle = \frac{1}{h_{n+1}} \int_{-h_{n+1}^-}^{h_{n+1}^+} \boldsymbol{\sigma}_p^{n+1}(z) \, dz \tag{9.45}$$

$$\mathbf{M}_p^{n+1} = \mathbf{I}^{(s)} \bullet \left[\frac{1}{h_{n+1}} \int_{-h_{n+1}^-}^{h_{n+1}^+} \boldsymbol{\sigma}_p^{n+1}(z) \, z \, dz \right] \bullet \mathbf{I}^{(s)} \tag{9.46}$$

where

$$\mathbf{I}^{(s)} = \mathbf{I} - \mathbf{n}_p^n \otimes \mathbf{n}_p^n \tag{9.47}$$

These are required in the balance of rotational inertia that is used to compute the updated rotation rate and the updated director vector. The internal moment for the shell material points can therefore the calculated using

$$\mathbf{m}_v^{\text{int}} = \sum_p \left[\left(\mathbf{M}_p^{n+1} \bullet \nabla S_{p,v}^{(1)} \bullet \mathbf{I}^{(s)} \right) + \left(\mathbf{n}_p^n \bullet \left\langle \boldsymbol{\sigma}_p^{n+1} \right\rangle \bullet \mathbf{I}^{(s)} \right) S_{p,v}^{(o)} \right] V_p^{n+1} \tag{9.48}$$

In practice, only the first term of equation (9.48) is interpolated to the grid and back to the particles. The equation of motion for rotational inertia is solved on the particles.

9.2.4 Solve the equations of motion.

The equations of motion for linear momentum are solved on the grid so that the acceleration at the grid vertices can be determined. The relation that is used is

$$\dot{\mathbf{u}}_v = \frac{1}{m_v}\left(\mathbf{f}_v^{\text{ext}} - \mathbf{f}_v^{\text{int}}\right) \tag{9.49}$$

where $\mathbf{f}^{\text{ext}}$ are external forces.

The angular momentum equations are solved on the particles after interpolating the term

$$\tilde{\mathbf{m}}_v = \sum_p \left(\mathbf{M}_p^{n+1} \bullet \nabla S_{p,v}^{(1)} \bullet \mathbf{I}^{(s)}\right) \tag{9.50}$$

back to the material points to get $\tilde{\mathbf{m}}_p$. The rotational acceleration is calculated using

$$\dot{\mathbf{r}}_p = \left(\frac{12\ V_p}{m_p\ h_p^2}\right)\left[\mathbf{m}_p^{\text{ext}} - \tilde{\mathbf{m}}_p - \mathbf{n}_p \bullet \langle \boldsymbol{\sigma}_p \rangle \bullet \mathbf{I}^{(s)}\right] \tag{9.51}$$

9.2.5 Integrate the acceleration.

The linear acceleration in integrated using a forward Euler rule on the grid, giving the updated velocity on the grid as

$$\mathbf{u}_v^{n+1} = \mathbf{u}_v^n + \Delta t\ \dot{\mathbf{u}}_v \tag{9.52}$$

For the rotational acceleration, the same procedure is followed at each material point to obtain an intermediate increment

$$\Delta\tilde{\mathbf{r}}_p = \Delta t\ \dot{\mathbf{r}}_p \tag{9.53}$$

The factor $m_p\ h_p^2$ in the denominator of the right hand side of equation (9.51) makes the differential equation stiff. An accurate solution of the equation requires an implicit integration or extremely small time steps. Instead, an implicit correction is made to $\Delta\tilde{\mathbf{r}}_p$ by solving the equation [72]

$$\left[\mathbf{I} + \beta\left(\mathbf{I} - \mathbf{n}_p^n \otimes \mathbf{n}_p^n\right)\right]\Delta\overset{\circ}{\mathbf{r}}_p = \Delta\tilde{\mathbf{r}}_p \tag{9.54}$$

where $\Delta\overset{\circ}{\mathbf{r}}_p$ is the corrected value of $\Delta\tilde{\mathbf{r}}_p$ and

$$\beta = \frac{6\ E}{V_p\ m_p}\left(\frac{\Delta t}{h}\right)^2 \tag{9.55}$$

which uses the Young's modulus E of the shell material. The intermediate rotation rate is updated using the corrected increment. Thus,

$$\overset{\star}{\boldsymbol{r}}_p^{n+1} = \mathbf{r}_p^n + \Delta\overset{\circ}{\mathbf{r}}_p\ . \tag{9.56}$$

9.2.6 Update the shell director and rotate the rotation rate

At this stage, the shell director at each material point is updated. The incremental rotation tensor $\Delta\mathbf{R}$ is calculated using equation (9.33) with rotation angle $\theta = |r|\Delta t$ and axis of rotation

$$\mathbf{a} = \frac{\mathbf{n}_p^n \times \overset{\star}{\boldsymbol{r}}_p^{n+1}}{|\mathbf{n}_p^n \times \overset{\star}{\boldsymbol{r}}_p^{n+1}|}\ . \tag{9.57}$$

The updated director is

$$\mathbf{n}_p^{n+1} = \Delta\mathbf{R} \bullet \mathbf{n}_p^n\ . \tag{9.58}$$

In addition, the rate of rotation has to be rotated so that the direction is perpendicular to the director using,

$$\mathbf{r}_p^{n+1} = \Delta\mathbf{R} \bullet \overset{\star}{\boldsymbol{r}}_p^{n+1}\ . \tag{9.59}$$

9.2.7 Interpolate back to the material points and update the state variables.

In the final step, the state variables at the grid are interpolated back to the material points using relations of the form

$$\mathbf{u}_p^{n+1} = \sum_v \mathbf{u}_v^{n+1} S_{p,v}^{(1)} \tag{9.60}$$

9.3 Typical simulation results

Three tests of the shell formulation have been performed on different shell geometries - a plane shell, a cylindrical shell, and a spherical shell.

9.3.1 Punched Plane Shell

This problem involves the indentation of a plane, circular shell into a rigid cylindrical die of radius 8 cm. The shell is made of annealed copper with the properties and dimensions shown in Table 9.1.

Table 9.1: Circular plane shell properties and dimensions.

ρ_0 (kg/m^3)	K (GPa)	G (GPa)	Thickness (cm)	Radius (cm)	Velocity (m/s)
8930	136.35	45.45	0.3	8	100

Snapshots of the deformation of the shell are shown in Figure 9.1. Substantial deformation of the shell

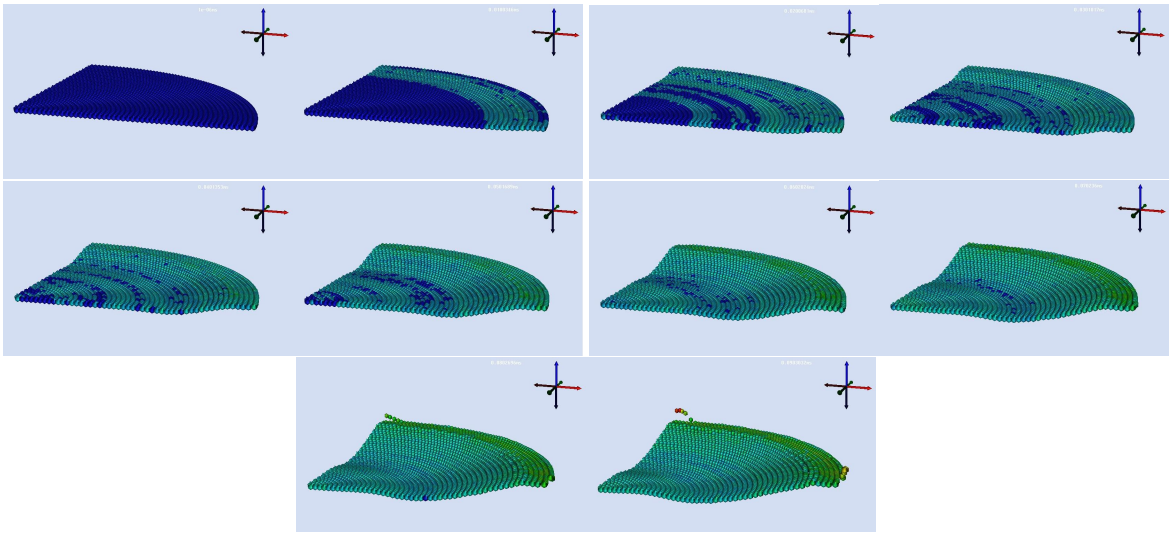


Figure 9.1: Deformation of punched circular plane shell.

occurs before particles at the edges tend to tear off. The tearing off of particles is due to the presence of large rotation rates ($\mathbf{r}$) which are due to the stiffness of the rotational acceleration equation (9.51). The implicit correction does not appear to be adequate beyond a certain point and a fully implicit shell formulation may be required for accurate simulation of extremely large deformations.

Particles in the figure have been colored using the equivalent stress at the center-of-mass of the shell. The stress distribution in the shell is quite uniform, though some artifacts in the form of rings appear. An

implicit formulation has been shown to remove such artifacts in the stress distribution in membranes [73]. Therefore, an implicit formulation may be useful for the shell formulation. Another possibility is that these artifacts may be due to membrane and shear locking, a known phenomenon in finite element formulations of shells based on a continuum approach [68, 74]. Such locking effects can be reduced using an addition hour glass control step [68] in the simulation.

9.3.2 Pinched Cylindrical Shell

The pinched cylindrical shell is one of the benchmark problems proposed by MacNeal and Harder [75]. The cylindrical shell that has been simulated in this work has dimensions similar to those used by Li et al. [76]. The shell is pinched by contact with two small rigid solid cylinders placed diametrically opposite each other and located at the midpoint of the axis of the cylinder. Each of the solid cylinders is 0.25 cm in radius, 0.5 cm in length, and moves toward the center of the pinched shell in a radial direction at 10 ms^{-1}. The material of the shell is annealed copper (properties are shown in Table 9.1). The cylindrical shell is 2.5 cm in radius, 5.0 cm long, and 0.05 cm thick.

Snapshots of the deformation of the pinched cylindrical shell are shown in Figure 9.2. The deformation

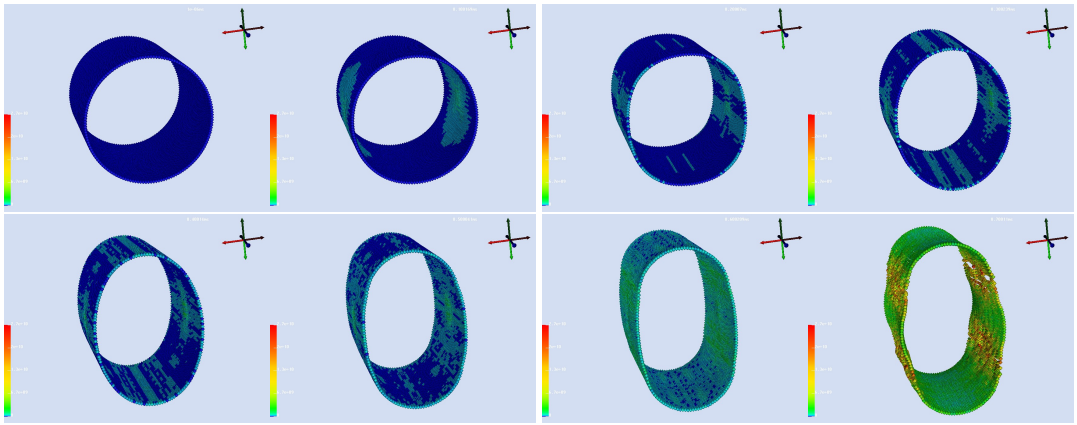


Figure 9.2: Deformation of pinched cylindrical shell.

of the shell proceeds uniformly for 60 ms. However, at this time the increments of rotation rate begin to increase rapidly at each time step, even though the velocity of the center-of-mass of the shell still remains stable. This effect can be attributed to the stiffness of the rotational inertia equation. The effect is that extremely large rotation rates are produced at 70 ms causing high velocities and eventual numerical fracture of the cylinder. The problem may be solved using an implicit shell formulation.

9.3.3 Inflating Spherical Shell

The inflating spherical shell problem is similar to that used to model lipid bilayers by Ayton et al. [77]. The shell is made of a soft rubbery material with a density of 10 kg m^{-3}, a bulk modulus of 60 KPa and a shear modulus of 30 KPa. The sphere has a radius of 0.5 m and is 1 cm thick. The spherical shell is pressurized by an initial internal pressure of 10 KPa. The pressure increases in proportion to the internal surface area as the sphere inflates.

The deformation of the shell with time is shown in Figure 9.3. The particles in the figure are colored on the basis of the equivalent stress. Though there is some difference between the values at different latitudes in the sphere, the equivalent stress is quite uniform in the shell. The variation can be reduced using the implicit material point method [78].

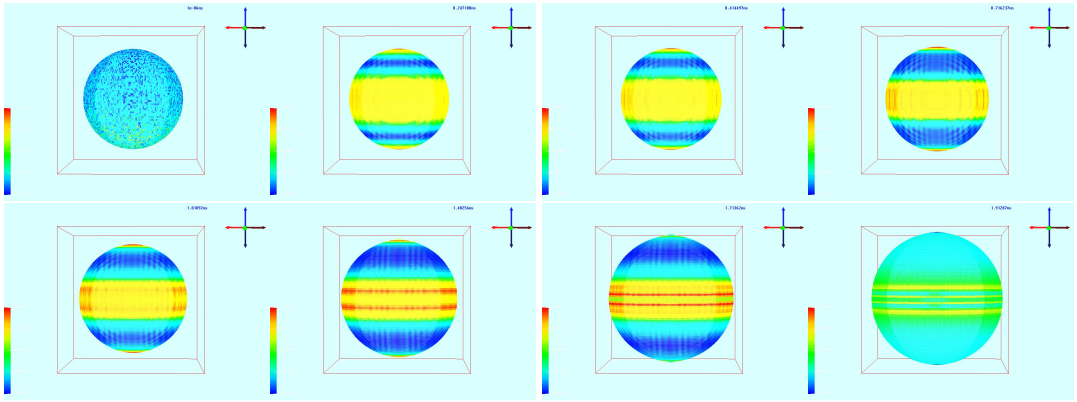


Figure 9.3: Deformation of inflating spherical shell.

9.4 Problems

A shell formulation has been developed and implemented for the explicit time stepping material point method based on the work of Lewis et al. [69]. Three different shell geometries and loading conditions have been tested. The results indicate that the stiff nature of the equation for rotational inertia may require the use of an implicit time stepping scheme for shell materials.

1. Shells and solids cannot interact easily.
2. Shell interpolations should be on a shell-based grid.

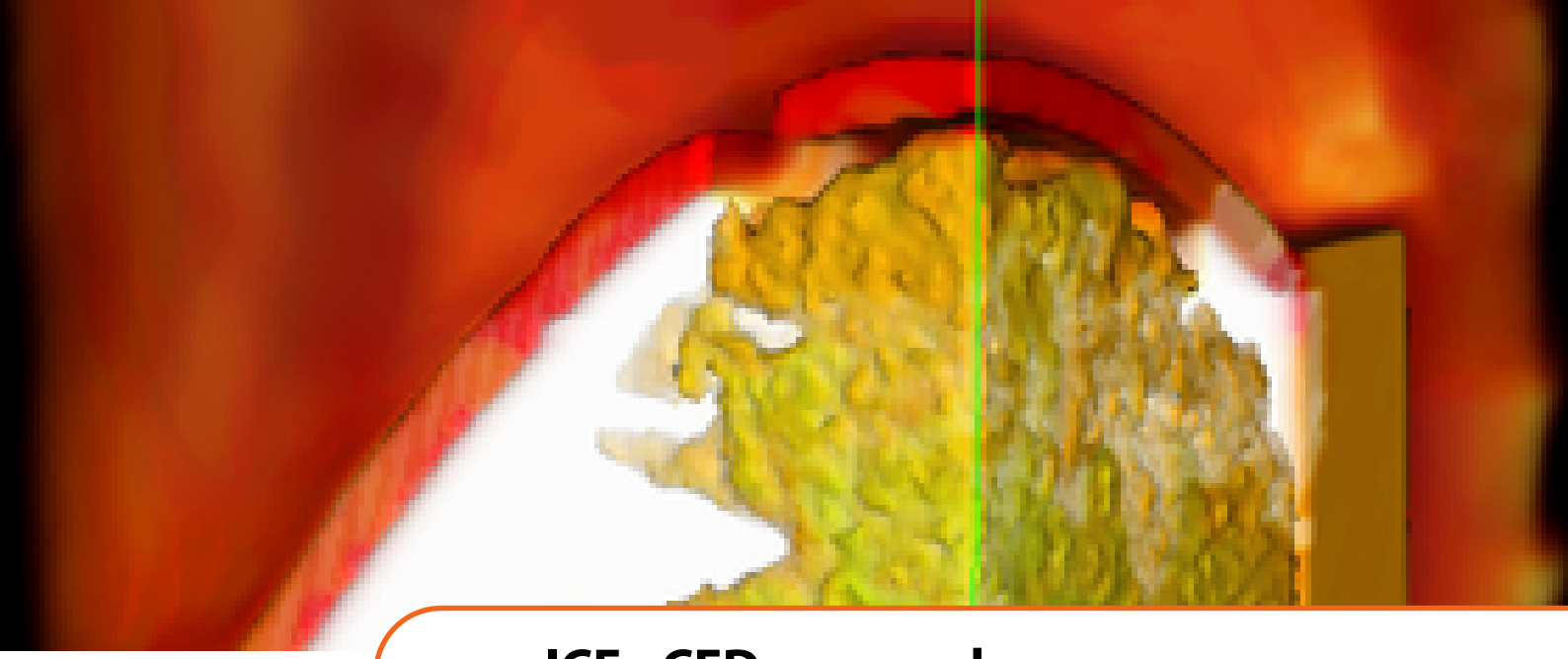
9.5 Alternative approaches

The approach discussed in the previous section suffers from the defect that particle data from the surface of the shell are projected on to grid points that are not on the surface of the shell.

This shortcoming has been addressed by Jiang et al. [79] by representing the shell with a surface mesh with quadrature points rather than with unstructured particles. Particles in a given simulation are represented either as:

1. standard MPM particles,
2. particles that represent Lagrangian mesh nodes, and
3. particles that represent Lagrangian mesh element quadrature points.

The particle state at time t_n includes the position $\mathbf{x}_p^n$, the velocity $\mathbf{v}_p^n$, the mass m_p, the volume V_p, the elastic deformation gradient $\boldsymbol{F}_p^{e,n}$, an affine velocity $\mathbf{c}_p^n$, and the material directions $\boldsymbol{D}_p$. Note that the deformation gradient is stored only in the standard MPM particles and at the shell mesh quadrature points. The material directions are stored only at the mesh quadrature points. These approaches are being explored in the research version of Vaango .



10 — ICE - CFD approach

10.1 Introduction

The ICE (Implicit Compressible Eulerian) code for fluid simulations in VAANGO uses a multi-material CFD approach designed to solve "full physics" simulations of dynamic fluid structure interactions involving large deformations and material transformations (e.g., phase change). "Full physics" refers to problems involving strong interactions between the fluid field and solid field temperatures and velocities, with a full Navier Stokes representation of fluid materials and the transient, nonlinear response of solid materials. These interactions may include chemical or physical transformation between the solid and fluid fields.

The theoretical and algorithmic basis for the multi-material CFD algorithm presented here is based on a body of work of several investigators at Los Alamos National Laboratory, primarily Bryan Kashiwa, Rick Rauenzahn and Matt Lewis. Several reports by these researchers are publicly available and are cited herein. It is largely through our personal interactions that we have been able to bring these ideas to bear on the simulations described herein.

An exposition of the governing equations is given in the next section, followed by an algorithmic description of the solution of those equations. This description is first done separately for the materials in the Eulerian and Lagrangian frames of reference, before details associated with the integrated approach are given.

10.1.1 Governing Equations

The governing multi-material model equations are stated and described, but not developed, here. Their development can be found in [11]. Here, our intent is to identify the quantities of interest, of which there are eight, as well as those equations (or closure models) which govern their behavior. Consider a collection of N materials, and let the subscript r signify one of the materials, such that $\mathrm{r} = 1, 2, 3, \ldots, N$. In an arbitary volume of space $V(\mathbf{x}, t)$, the averaged thermodynamic state of a material is given by the vector $[M_\mathrm{r}, \mathbf{u}_\mathrm{r}, e_\mathrm{r}, T_\mathrm{r}, v_\mathrm{r}, \theta_\mathrm{r}, \boldsymbol{\sigma}_\mathrm{r}, p]$, the elements of which are the r-material mass, velocity, internal energy, temperature, specific volume, volume fraction, stress, and the equilibration pressure. The r-material averaged density is $\rho_\mathrm{r} = M_\mathrm{r}/V$. The rate of change of the state in a volume moving with the velocity of r-material

is:

$$\frac{1}{V}\frac{D_r M_r}{Dt} = \sum_{i=1}^{N} \Gamma_{rs} \tag{10.1}$$

$$\frac{1}{V}\frac{D_r}{Dt}(M_r \mathbf{u}_r) = \theta_r \nabla \cdot \boldsymbol{\sigma} + \nabla \cdot \theta_r(\boldsymbol{\sigma}_r - \boldsymbol{\sigma}) + \rho_r \mathbf{g} + \sum_{i=1}^{N} \mathbf{f}_{rs} + \sum_{i=1}^{N} \mathbf{u}_{rs}^{+} \Gamma_{rs} \tag{10.2}$$

$$\frac{1}{V}\frac{D_r}{Dt}(M_r e_r) = -\rho_r p \frac{D_r v_r}{Dt} + \theta_r \boldsymbol{\tau}_r : \nabla \mathbf{u}_r - \nabla \cdot \mathbf{j}_r + \sum_{i=1}^{N} q_{rs} + \sum_{i=1}^{N} h_{rs}^{+} \Gamma_{rs} \tag{10.3}$$

Equations (10.1-10.3) are the averaged model equations for mass, momentum, and internal energy of r-material, in which $\boldsymbol{\sigma}$ is the mean mixture stress, taken here to be isotropic, so that $\boldsymbol{\sigma} = -p\boldsymbol{I}$ in terms of the hydrodynamic pressure p. The effects of turbulence have been explicitly omitted from these equations, and the subsequent solution, for the sake of simplicity. However, including the effects of turbulence is not precluded by either the model or the solution method used here.

In Eq. (10.2) the term $\sum_{s=1}^{N} \mathbf{f}_{rs}$ signifies a model for the momentum exchange among materials. This term results from the deviation of the r-field stress from the mean stress, averaged, and is typically modeled as a function of the relative velocity between materials at a point. (For a two material problem this term might look like $\mathbf{f}_{12} = K_{12}\theta_1\theta_2(\mathbf{u}_1 - \mathbf{u}_2)$ where the coefficient K_{12} determines the rate at which momentum is transferred between materials). Likewise, in Eq. (10.3), $\sum_{s=1}^{N} q_{rs}$ represents an exchange of heat energy among materials. For a two material problem $q_{12} = H_{12}\theta_1\theta_2(T_2 - T_1)$ where T_r is the r-material temperature and the coefficient H_{rs} is analogous to a convective heat transfer rate coefficient. The heat flux is $\mathbf{j}_r = -\rho_r b_r \nabla T_r$ where the thermal diffusion coefficient b_r includes both molecular and turbulent effects (when the turbulence is included).

In Eqs. (10.1-10.3) the term Γ_{rs} is the rate of mass conversion from s-material into r-material, for example, the burning of a solid or liquid reactant into gaseous products. The rate at which mass conversion occurs is governed by a reaction model. In Eqs. (10.2) and (10.3), the velocity $\mathbf{u}_{rs}^{+}$ and the enthalpy h_{rs}^{+} are those of the s-material that is converted into r-material. These are simply the mean values associated with the donor material.

The temperature T_r, specific volume v_r, volume fraction θ_r, and hydrodynamic pressure p are related to the r-material mass density, ρ_r, and specific internal energy, e_r, by way of equations of state. The four relations for the four quantites (T_r, v_r, θ_r, p) are:

$$e_r = e_r(v_r, T_r) \tag{10.4}$$

$$v_r = v_r(p, T_r) \tag{10.5}$$

and

$$\theta_r = \rho_r v_r \tag{10.6}$$

$$0 = 1 - \sum_{i=1}^{N} \rho_s v_s \tag{10.7}$$

Equations (10.4) and (10.5) are, respectively, the caloric and thermal equations of state. Equation (10.6) defines the volume fraction, θ, as the volume of r-material per total material volume, and with that definition, Equation (10.7), referred to as the multi-material equation of state, follows. It defines the unique value of the hydrodynamic pressure p that allows arbitrary masses of the multiple materials to identically fill the volume V. This pressure is called the "equilibration" pressure [80].

A closure relation is still needed for the material stress $\boldsymbol{\sigma}_r$. For a fluid $\boldsymbol{\sigma}_r = -p\boldsymbol{I} + \boldsymbol{\tau}_r$ where the deviatoric stress is well known for Newtonian fluids. For a solid, the material stress is the Cauchy stress. The Cauchy stress is computed using a solid constitutive model and may depend on the the rate of deformation, the

current state of deformation ($\boldsymbol{E}$), the temperature, and possibly a number of history variables. Such a relationship may be expressed as:

$$\boldsymbol{\sigma}_\mathrm{r} \equiv \boldsymbol{\sigma}_\mathrm{r}(\nabla \mathbf{u}_\mathrm{r}, \boldsymbol{E}_\mathrm{r}, T_\mathrm{r}, \ \ldots) \tag{10.8}$$

The approach described here imposes no restrictions on the types of constitutive relations that can be considered. More specific discussion of some of the models used in this work can be found in the section on ICE models.

Equations (10.1-10.8) form a set of eight equations for the eight-element state vector, $[M_\mathrm{r}, \mathbf{u}_\mathrm{r}, e_\mathrm{r}, T_\mathrm{r}, v_\mathrm{r}, \theta_\mathrm{r}, \boldsymbol{\sigma}_\mathrm{r}, p]$, for any arbitrary volume of space V moving with the r-material velocity. The approach described here uses the reference frame most suitable for a particular material type. As such, there is no guarantee that arbitrary volumes will remain coincident for materials described in different reference frames. This problem is addressed by treating the specific volume as a dynamic variable of the material state which is integrated forward in time from initial conditions. In so doing, at any time, the total volume associated with all of the materials is given by:

$$V_t = \sum_{r=1}^{N} M_\mathrm{r} v_\mathrm{r} \tag{10.9}$$

so the volume fraction is $\theta_\mathrm{r} = M_\mathrm{r} v_\mathrm{r} / V_t$ (which sums to one by definition). An evolution equation for the r-material specific volume, derived from the time variation of Eqs. (10.4-10.7), has been developed in [11]. It is stated here as:

$$\frac{1}{V}\frac{D_\mathrm{r}}{Dt}(M_\mathrm{r} v_\mathrm{r}) = f_\mathrm{r}^\theta \nabla \cdot \mathbf{u} + \left[v_\mathrm{r}\Gamma_\mathrm{r} - f_\mathrm{r}^\theta \sum_{i=1}^{N} v_\mathrm{s}\Gamma_\mathrm{s} \right] + \left[\theta_\mathrm{r}\beta_\mathrm{r}\frac{D_\mathrm{r} T_\mathrm{r}}{Dt} - f_\mathrm{r}^\theta \sum_{i=1}^{N} \theta_\mathrm{s}\beta_\mathrm{s}\frac{D_\mathrm{s} T_\mathrm{s}}{Dt} \right]. \tag{10.10}$$

where

$$f_\mathrm{r}^\theta = \frac{\theta_\mathrm{r}\kappa_\mathrm{r}}{\sum_{\mathrm{s}=1}^{N} \theta_\mathrm{s}\kappa_\mathrm{s}} \tag{10.11}$$

and κ_r is the r-material bulk compressibility.

The evaluation of the multi-material equation of state (Eq. (10.7)) is still required in order to determine an equilibrium pressure that results in a common value for the pressure, as well as specific volumes that fill the total volume identically.

A description of the means by which numerical solutions to the equations in Section 10.2 are found is presented next. This begins with separate, brief overviews of the methodologies used for the Eulerian and Lagrangian reference frames. The algorithmic details necesssary for integrating them to achieve a tightly coupled fluid-structure interaction capability is provided in Sec. 12.

10.2 Algorithm Description

The Eulerian method implemented here is a cell-centered, finite volume, multi-material version of the ICE (for Implicit, Continuous fluid, Eulerian) method [81] developed by Kashiwa and others at Los Alamos National Laboratory [82]. "Cell-centered" means that all elements of the state are colocated at the grid cell-center (in contrast to a staggered grid, in which velocity components may be centered at the faces of grid cells, for example). This colocation is particularly important in regions where a material mass is vanishing. By using the same control volume for mass and momentum it can be assured that as the material mass goes to zero, the mass and momentum also go to zero at the same rate, leaving a well-defined velocity. The technique is fully compressible, allowing wide generality in the types of problems that can be addressed.

Our use of the cell-centered ICE method employs time splitting: first, a Lagrangian step updates the state due to the physics of the conservation laws (i.e., right hand side of Eqs. 10.1-10.3); this is followed by an Eulerian step, in which the change due to advection is evaluated. For solution in the Eulerian frame, the method is well developed and described in [82].

In the mixed frame approach used here, a modification to the multi-material equation of state is needed. Equation (10.7) is unambiguous when all materials are fluids or in cases of a flow consisting of dispersed solid grains in a carrier fluid. However in fluid-structure problems the stress state of a submerged structure may be strongly directional, and the isotropic part of the stress has nothing to do with the hydrodynamic (equilibration) pressure p. The equilibrium that typically exists between a fluid and a solid is at the interface between the two materials: there the normal part of the traction equals the pressure exerted by the fluid on the solid over the interface. Because the orientation of the interface is not explicitly known at any point (it is effectively lost in the averaging) such an equilibrium cannot be computed.

The difficulty, and the modification that resolves it, can be understood by considering a solid material in tension coexisting with a gas. For solid materials, the equation of state is the bulk part of the constitutive response (that is, the isotropic part of the Cauchy stress versus specific volume and temperature). If one attempts to equate the isotropic part of the stress with the fluid pressure, there exist regions in pressure-volume space for which Eq. (10.7) has no physical solutions (because the gas pressure is only positive). This can be seen schematically in Fig. 10.1, which sketches equations of state for a gas and a solid, at an arbitrary temperature.

Recall that the isothermal compressiblity is the negative slope of the specific volume versus pressure. Embedded structures considered here are solids and, at low pressure, possess a much smaller compressibility than the gasses in which they are submerged. Nevertheless the variation of condensed phase specific volume can be important at very high pressures, where the compressibilities of the gas and condensed phase materials can become comparable (as in a detonation wave, for example). Because the speed of shock waves in materials is determined by their equations of state, obtaining accurate high pressure behavior is an important goal of our FSI studies.

To compensate for the lack of directional information for the embedded surfaces, we evaluate the solid phase equations of state in two parts. Above a specified postive threshold pressure (typically 1 atmosphere), the full equation of state is respected; below that threshold pressure, the solid phase pressure follows a polynomial chosen to be C^1 continuous at the threshold value and which approaches zero as the specific volume becomes large. The effect is to decouple the solid phase specific volume from the stress when the isotropic part of the stress falls below a threshold value. In regions of coexistence at states below the threshold pressure, p tends to behave according to the fluid equation of state (due to the greater compressibility) while in regions of pure condensed phase material p tends rapidly toward zero and the full material stress dominates the dynamics as it should.

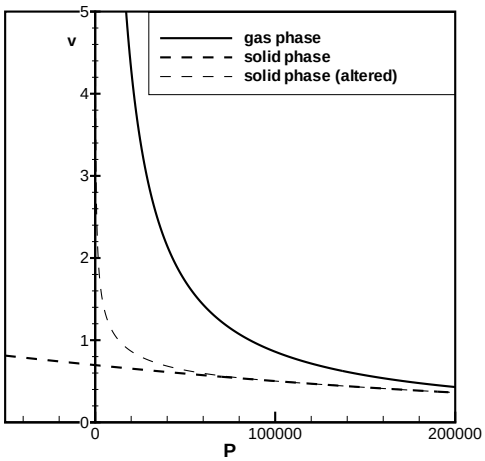
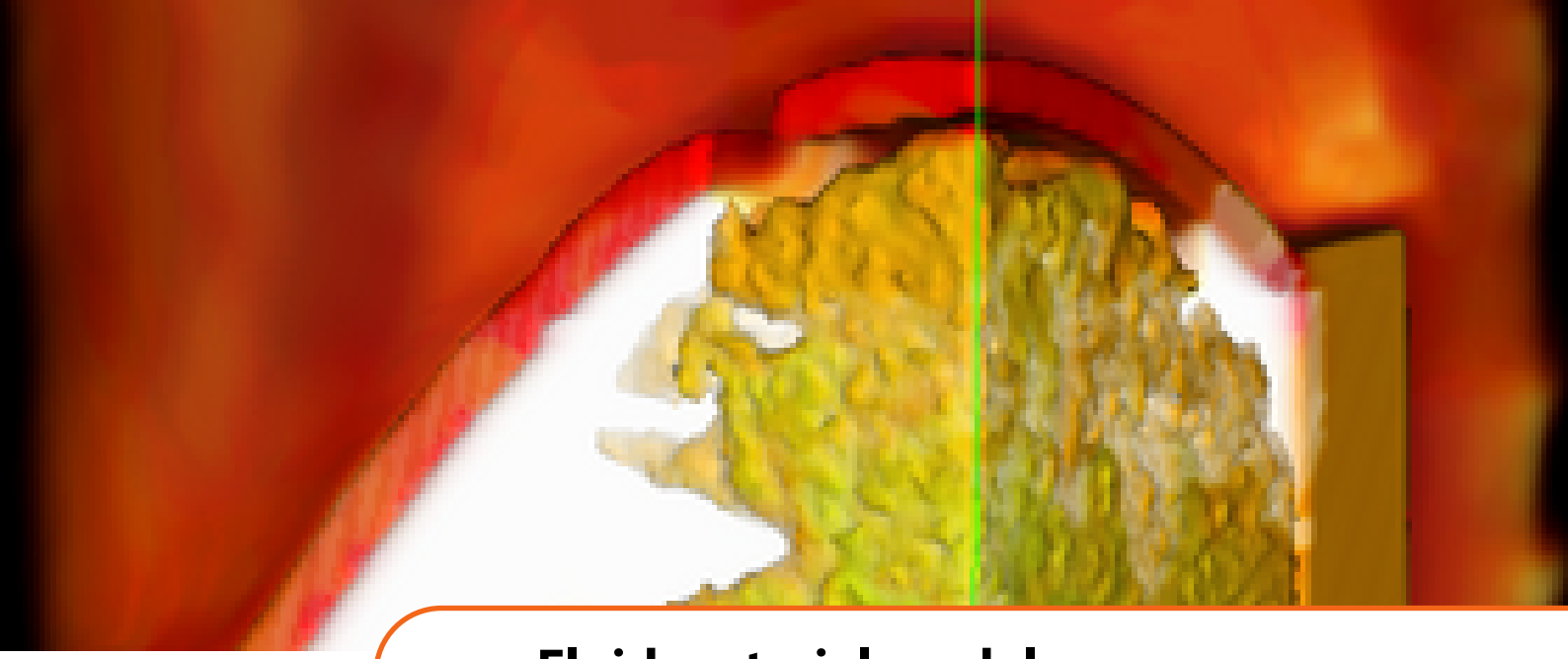


Figure 10.1: Specific volume *vs* pressure for a gas phase material and a solid phase material. Light dashed line reflects an altered solid phase equation of state to keep all materials in positive equilibration pressure space.



11 — Fluid material models

ICE use standard Newtonian fluid models for the simulation of fluids. However, since it was designed for shock-compression applications, we focus on some of the high energy material models that are used by ICE in Vaango .

11.1 High Energy Material Reaction Models

Two types of High Energy (HE) reaction models were considered here. The first is a model for detonation, in which the reaction front proceeds as a shock wave through the solid reactant, leaving highly pressurized product gases behind the shock. The second is a deflagration model, in which the reaction proceeds more slowly through the reactant in the form of a thermal burn. Each is described here.

11.1.1 The JWL++ Detonation Model

The detonation model used in two of the calculations discussed in Section 12.3 is a reactive flow model known as JWL++[83]. JWL++ consists of equations of state for the reactant and the products of reaction as well as a rate equation governing the transformation from product to reactant. In addition, the model consists of a "mixer" which is a rule for determining the pressure in a mixture of product and reactant, as found in a partially reacted cell. Because pressure equilibration among materials is already part of the multi-material CFD formulation described in Section 12.1, the mixer was not part of the current implementation. Lastly, two additional rules apply. The first is that reaction begins in a cell when the pressure in that cell exceeds 200 MPa. Finally, no more than 20% of the explosive in a cell is allowed to react in a given timestep.

The Murnaghan equation of state [84] used for the solid reactant material is given by:

$$p = \frac{1}{n\kappa}\left(\frac{1}{v^n} - 1\right) \tag{11.1}$$

where $v = \rho_0/\rho$, and n and κ are material dependent model parameters. Note that while the reactants are solid materials, they are assumed to not support deviatoric stress. Since a detonation propagates faster than shear waves, the strength in shear of the reactants can be neglected. Since it is not necessary to track the deformation history of a particular material element, in this case, the reactant material was tracked only in the Eulerian frame, *i.e.* not represented by particles within MPM.

The JWL C-term form is the equation of state used for products, and is given by:

$$P = A\exp(-R_1 v) + B\exp(-R_2 v) + \frac{C}{\rho_0 \kappa v^{n-1}} \tag{11.2}$$

where A, B, C, R_1, R_2, ρ_0 and κ are all material dependent model parameters.

The rate equation governing the transformation of reactant to product is given by:

$$\frac{dF}{dt} = G(p+q)^b(1-F) \tag{11.3}$$

where G is a rate constant, and b indicates the power dependence on pressure. q is an artificial viscosity, but was not included in the current implementation of the model. Lastly:

$$F = \frac{\rho_{\text{product}}}{\rho_{\text{reactant}} + \rho_{\text{product}}} \tag{11.4}$$

is the burn fraction in a cell. This can be differentiated and solved for a mass burn rate in terms of dF:

$$\Gamma = \frac{dF}{dt}\left(\rho_{\text{reactant}} + \rho_{\text{product}}\right) \tag{11.5}$$

11.1.2 Deflagration Model

The rate of thermal burning, or deflagration, of a monopropellant solid explosive is typically assumed to behave as:

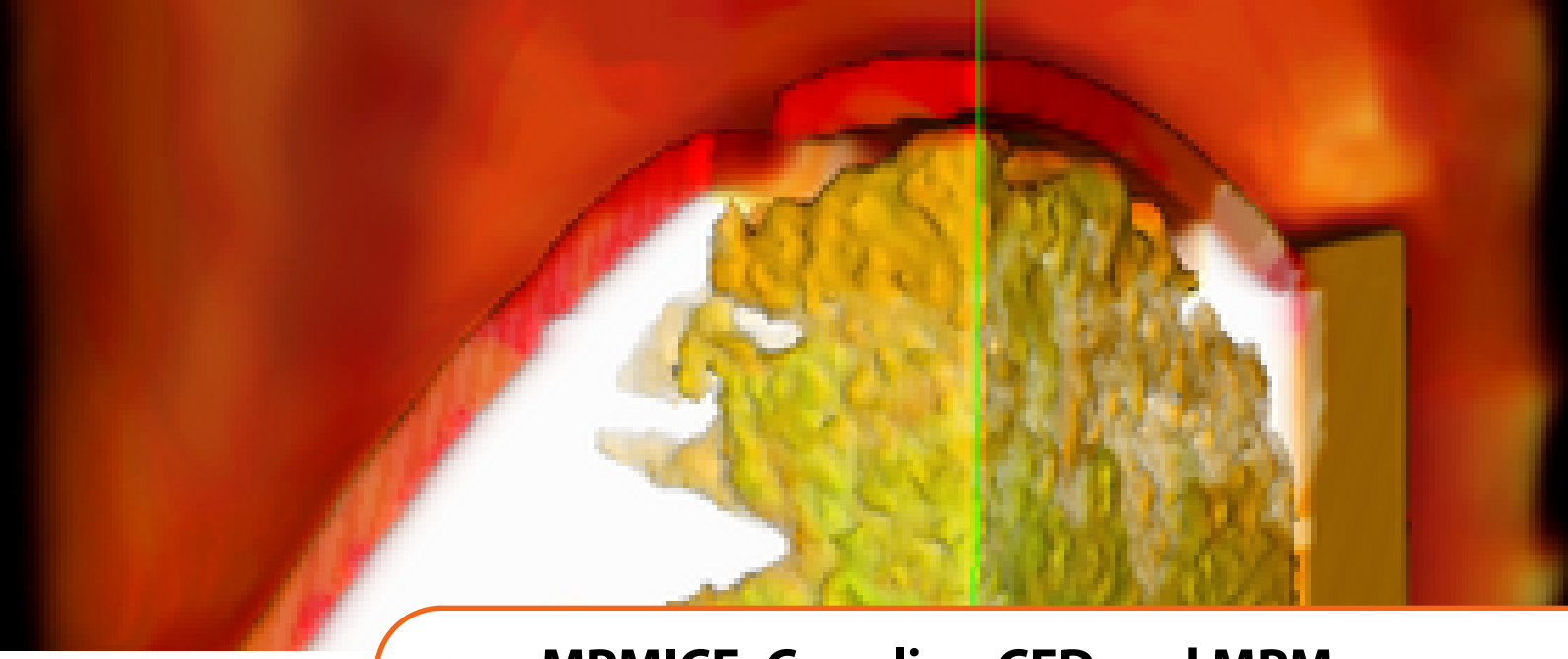
$$D = Ap^n \tag{11.6}$$

where D can be thought of as the velocity at which the burn front propagates through the reactant (with units of length/time) and p is the local pressure [85]. A and n are parameters that are empirically determined for particular explosives. Because deflagration is a surface phenomenon, our implementation requires the identification of the surface of the explosive. The surface is assumed to lie within those cells which have the highest gradient of mass density of the reactant material. Within each surface cell, an estimate of the surface area a is made based on the direction of the gradient, and the rate D above is converted to a mass burn rate by:

$$\Gamma = aD\rho_{\text{reactant}} \tag{11.7}$$

where ρ_{reactant} is the local density of the explosive. While the reaction rate is independent of temperature, initiation of the burn depends on reaching a threshold temperature at the surface.

Since the rate at which a deflagration propagates is much slower than the shear wave speed in the reactant, it is important to track its deformation as pressure builds up within the container. This deformation may lead to the formation of more surface area upon which the reaction can take place, and the change to the shape of the explosive can affect the eventual violence of the explosion. Because of this, for deflagration cases, the explosive is represented by particles in the Lagrangian frame. The stress response is usually treated by an implementation of ViscoSCRAM [86], which includes representation of the material's viscoelastic response, and considers effects of micro-crack growth within the granular composite material.



12 — MPMICE: Coupling CFD and MPM

Approaches to fluid structure interaction (FSI) problems are typically divided into two classes. "Separated" approaches treat individual materials as occupying distinct regions of space, with interactions occurring only at material interfaces. The details of those interactions vary between implementations, and are often a function of the degree, or "strength" of the coupling between the fluid and solid fields. Because of the separated nature of the materials, only one set of state variables is needed at any point in space, since only one material is allowed to exist at that point. "Averaged" model approaches allow **all** materials to exist at any point in space with some probability. Variables describing the material state vary continuously throughout the computational domain, thus, the state of every material is defined at every point in space. Distinct material interfaces are not defined, rather the interaction between materials is computed in an average sense, and, as such, interactions among materials may take place anywhere.

While both the separated model and averaged model approaches have their respective merits, the averaged model, when carried out on an Eulerian grid, allows arbitrary distortion of materials and material interfaces. However, these distortions can be catastrophic for the solid material, as the deformation history of the solid must be transported through the Eulerian grid. This transport can lead to non-physical stresses and the interface between materials is also subject to diffusion. The latter problem can be mitigated via surface tracking and the use of a single valued velocity field [87, 88], but this does not eliminate the problems of stress transport.

The approach described here uses the averaged model approach, and addresses the issue of stress transport by integrating the state of the solid field in the "material" frame of reference through use of the Material Point Method (MPM) [1, 2]. MPM is a particle method for solid mechanics that allows the solid field to undergo arbitrary distortion. Because the fluid state is integrated in the Eulerian frame, it can also undergo arbitrary distortion. MPM uses a computational "scratchpad" grid to advance the solution to the equations of motion, and by choosing to use the same grid used in the Eulerian frame of reference, interactions among the materials are facilitated on this common computational framework. By choosing to use an infinitely fast rate of momentum transfer between the materials, the single velocity field limit is obtained, and the interface between materials is limited to, at most, a few cells. Thus, in the differential limit, the separated model can be recovered. This means that with sufficient grid resolution, the accuracy of the separated model and the robustness of the averaged model can be enjoyed simultaneously.

An exposition of the governing equations of the CFD approach are given in Chapter 10 while those for MPM can be found in Chapter 1. Algorithmic description of the solution of those equations can also be found in those chapters, but a summary is provided here. The reader is encouraged to browse Section 12.3 to better appreciate the direction that the subsequent development is headed.

12.1 Numerical Implementation

A description of the means by which numerical solutions to the equations in the preceding section are found is presented next. This begins with separate, brief, overviews of the methodologies used for the Eulerian and Lagrangian reference frames. The algorithmic details necesssary for integrating them to achieve a tightly coupled fluid-structure interaction capability is provided in Sec. 12.1.3.

12.1.1 ICE Eulerian Multi-Material Method

The Eulerian method implemented here is a cell-centered, finite volume, multi-material version of the ICE (for Implicit, Continuous fluid, Eulerian) method [89] developed by Kashiwa and others at Los Alamos National Laboratory [90]. "Cell-centered" means that all elements of the state are colocated at the grid cell-center (in contrast to a staggered grid, in which velocity components may be centered at the faces of grid cells, for example). This colocation is particularly important in regions where a material mass is vanishing. By using the same control volume for mass and momentum it can be assured that as the material mass goes to zero, the mass and momentum also go to zero at the same rate, leaving a well defined velocity. The technique is fully compressible, allowing wide generality in the types of problems that can be efficiently computed.

Our use of the cell-centered ICE method employs time splitting: first, a Lagrangian step updates the state due to the physics of the conservation laws (i.e., right hand side of Eqs. 10.1-10.3); this is followed by an Eulerian step, in which the change due to advection is evaluated. For solution in the Eulerian frame, the method is well developed and described in [90].

In the mixed frame approach used here, a modification to the multi-material equation of state is needed. Equation 10.7 is unambiguous when all materials are fluids or in cases of a flow consisting of dispersed solid grains in a carrier fluid. However in fluid-structure problems the stress state of a submerged structure may be strongly directional, and the isotropic part of the stress has nothing to do with the hydrodynamic (equilibration) pressure p. The equilibrium that typically exists between a fluid and a solid is at the interface between the two materials: there the normal part of the traction equals the pressure exerted by the fluid on the solid over the interface. Because the orientation of the interface is not explicitly known at any point (it is effectively lost in the averaging) such an equilibrium cannot be computed.

The difficulty, and the modification that resolves it, can be understood by considering a solid material in tension coexisting with a gas. For solid materials, the equation of state is the bulk part of the constitutive response (that is, the isotropic part of the Cauchy stress versus specific volume and temperature). If one attempts to equate the isotropic part of the Cauchy stress with the fluid pressure, there exist regions in pressure-volume space for which Eq. 10.7 has no physical solutions (because the gas pressure is only positive). This can be seen schematically in Fig. 10.1, which sketches equations of state for a gas and a solid, at an arbitrary temperature.

Recall that the isothermal compressiblity is the negative slope of the specific volume versus pressure. Embedded structures considered here are solids and, at low pressure, possess a much smaller compressibility than the gasses in which they are submerged. Nevertheless the variation of condensed phase specific volume can be important at very high pressures, where the compressibilities of the gas and condensed phase materials can become comparable (as in a detonation wave, for example). Because the speed of shock waves in materials is determined by their equations of state, obtaining accurate high pressure behavior is an important goal of our FSI studies.

To compensate for the lack of directional information for the embedded surfaces, we evaluate the solid phase equations of state in two parts. Above a specified postive threshold pressure (typically 1 atmosphere), the full equation of state is respected; below that threshold pressure, the solid phase pressure follows a polynomial chosen to be C^1 continuous at the threshold value and which approaches zero as the specific volume becomes large. The effect is to decouple the solid phase specific volume from the stress when the isotropic part of the stress falls below a threshold value. In regions of coexistence at states be-

low the threshold pressure, p tends to behave according to the fluid equation of state (due to the greater compressibility) while in regions of pure condensed phase material p tends rapidly toward zero and the full material stress dominates the dynamics as it should.

12.1.2 The Material Point Method

Solid materials with history dependent constitutive relations are more conveniently treated in the Lagrangian frame. Here we briefly describe a particle method known as the Material Point Method (MPM) which is used to evolve the equations of motion for the solid phase materials. MPM is a powerful technique for computational solid mechanics, and has found favor in applications involving complex geometries [91], large deformations [92] and fracture [93], to name a few. After the description of MPM, its incorporation within the multi-material solution is described in Sec. 12.1.3.

Originally described by Sulsky, et al., [1, 2], MPM is a particle method for structural mechanics simulations. MPM is an extension to solid mechanics of FLIP [3], which is a particle-in-cell (PIC) method for fluid flow simulation. The method typically uses a cartesian grid as a computational scratchpad for computing spatial gradients. This same grid also functions as an updated Lagrangian grid that moves with the particles during advection and thus eliminates the diffusion problems associated with advection on an Eulerian grid. At the end of a timestep, the grid is reset to the original, regularly ordered, position. Details of the theory of MPM can be found in Chapter 1.

By describing and implementing MPM in an independent fashion, validation of the method itself as well as submodels (e.g., constitutive models and contact) is simplified. However, we emphasize that its use here is for selected material field description within the general multi-material formulation. This integration is described next.

12.1.3 Integration of MPM within the Eulerian Multi-Material Formulation

An important feature of this work is the ability to represent a material in either the Lagrangian or Eulerian frame. This allows treating specific phases in their traditionally preferred frame of reference. The Material Point Method, is used to time advance solid materials that are best described in a Lagrangian reference frame. By choosing the background grid used to update the solid materials to be the same grid used in the multi-material Eulerian description, all interactions among materials can be computed in the common framework, according to the momentum and heat exchange terms in Eqs 10.2-10.3. This results in a robust and tightly coupled solution for interacting materials with very different responses.

To illustrate how the integration is accomplished in an algorithmic fashion the explicit steps for advancing a fluid-structure interaction problem from time t to time $t + \Delta t$ are described below.

1. **Project particle state to grid:** A simulation timestep begins by interpolating the particle description of the solid to the grid. This starts with a projection of particle data to grid vertices, or nodes, as described in Eq. 1.60, and is followed by a subsequent projection from the nodes to the cell-centers. Since our work uses a uniform structured grid, each node has equal weight in its contribution to the cell-centered value. The exception to this is near computational boundaries where symmetric boundary conditions are used. The weight of those nodes on the boundary must be doubled in order to achieve the desired effect.
2. **Compute the equilibrium pressure:** While Eq. 10.7 and the surrounding discussion describes the basic process, one specific point warrants further explanation. In particular, the manner in which each material's volume fraction is computed is crucial. Because the solid and fluid materials are evolved in different frames of reference, the total volume of material in a cell is not necessarily equal to the volume of a computational cell. Material volume is tracked by evolving the specific volume for each material according to Eq. 10.10. The details of this are further described in step 11. With the materials' masses and specific volumes, material volume can be computed ($V_r = M_r v_r$) and summed to find the total material volume. The volume fraction θ_r is then computed as the volume

of r-material per total material volume. With this, the solution of Eq. 10.7 can be carried out at each cell using a Newton-Raphson technique[94], which results in new values for the equilibrium pressure, p_{eq}, volume fraction, θ_r and specific volume, v_r.

3. **Compute face-centered velocities, u_r^*, for the Eulerian advection:** At this point, fluxing velocities are computed at each cell face. The expression for this is based on a time advanced estimate for the cell-centered velocity. A full development can be found in [90] and [11] but here, only the result is given:

$$u_r^* = \frac{\rho_{r_L} u_{r_L} + \rho_{r_R} u_{r_R}}{\rho_{r_L} + \rho_{r_R}} - \left(\frac{2 v_{r_L} v_{r_R} \Delta t}{v_{r_L} + v_{r_R}}\right)\left(\frac{p_{\text{eq}_R} - p_{\text{eq}_L}}{\Delta x}\right) + g\Delta t \tag{12.1}$$

The first term above is a mass weighted average of the logically left and right cell-centered velocities, the second is a pressure gradient acceleration term, and the third is acceleration due to the component of gravity in the face normal direction. Not shown explicitly is the necessary momentum exchange at the face-centers. This is done on the faces in the same manner as described subsequently in step 10 for the cell-centered momentum exchange.

4. **Multiphase chemistry:** Compute sources of mass, momentum, energy and specific volume as a result of phase changing chemical reactions for each r-material, Γ_r, $u_r\Gamma_r$, $e_r\Gamma_r$, and $v_r\Gamma_r$. Specifics of the calculation of Γ_r are model dependent, and examples are given in Sec. 11.1.
Care must be taken to reduce the momentum, internal energy and volume of the reactant by an amount proportional to the mass consumed each timestep, so that those quantities are depleted at the same rate as the mass. When the reactant material is described by particles, decrementing the particle mass automatically decreases the momentum and internal energy of that particle by the appropriate amount. This mass, momentum, and internal energy is transferred to the product material's state, and the volume fraction for the reactant and product materials is recomputed.

5. **Compute an estimate of the time advanced pressure, p:** Based on the volume of material being added to (or subtracted from) a cell in a given timestep, an increment to the cell-centered pressure is computed using:

$$\Delta p = \Delta t \frac{\sum_{r=1}^{N} v_r \Gamma_r - \sum_{r=1}^{N} \nabla \cdot (\theta_r^* u_r^*)}{\sum_{r=1}^{N} \theta_r \kappa_r} \tag{12.2}$$

$$p = p_{\text{eq}} + \Delta p \tag{12.3}$$

where κ_r is the r-material bulk compressibility. The first term in the numerator of Eq. 12.2 represents the change in volume due to reaction, i.e., a given amount of mass would tend to occupy more volume in the gas phase than the solid phase, leading to an increase in pressure. The second term in the numerator represents the net change in volume of material in a cell due to flow into or out of the cell. The denominator is essentially the mean compressibility of the mixture of materials within that cell. This increment in pressure is added to the equilibrium pressure computed in step 2 and is the pressure used for the remainder of the current timestep. Again, the details leading to this equation can be found in [90].

6. **Face Centered Pressure p^*:** The calculation of p^* is discussed at length in [11]. For this work, it is computed using the updated pressure by:

$$p^* = \left(\frac{p_L}{\rho_L} + \frac{p_R}{\rho_R}\right) \Big/ \left(\frac{1}{\rho_L} + \frac{1}{\rho_R}\right) \tag{12.4}$$

where the subscripts L and R refer to the logically left and right cell-centered values, respectively, and ρ is the sum of all material's densities in that cell. This will be used subsequently for the computation of the pressure gradient, ∇p^*.

7. **Material Stresses:** For the solid, we calculate the velocity gradient at each particle based on the grid velocity (Eq. 1.64) for use in a constitutive model to compute particle stress. Fluid stresses are computed on cell faces based on cell-centered velocities.

8. **Accumulate sources of mass, momentum and energy at cell-centers:** These terms are of the form:

$$\Delta(m)_r = \Delta t\ V \sum_{s=1,s\neq r}^{N} \Gamma_s \tag{12.5}$$

$$\Delta(mu)_r = -\Delta t\ V \left[\theta_r \nabla p^* + \nabla\cdot\theta_r(\boldsymbol{\sigma}_r - \boldsymbol{\sigma}) + \sum_{s=1,s\neq r}^{N} u_s \Gamma_s \right] \tag{12.6}$$

$$\Delta(me)_r = -\Delta t\ V \left[f^{\theta r} p \sum_{s=1}^{N} \nabla\cdot(\theta_r^* u_r^*) + \sum_{s=1,s\neq r}^{N} e_s \Gamma_s \right] \tag{12.7}$$

Note that the only source of internal energy being considered here is that due to "flow work". This is required for the compressible flow formulation, but other terms, such as heat conduction are at times included.

9. **Compute Lagrangian phase quantities at cell-centers:** The increments in mass, momentum and energy computed above are added to their time t counterparts to get the Lagrangian values for these quantities. Note that here, some Lagrangian quantities are denoted by an $L-$ superscript. This indicates that all physical processes have been accounted for except for inter-material exchange of momentum and heat which is described in the following step.

$$(m)_r^L = (m)_r^t + \Delta(m)_r \tag{12.8}$$

$$(mu)_r^{L-} = (mu)_r^t + \Delta(mu)_r \tag{12.9}$$

$$(me)_r^{L-} = (me)_r^t + \Delta(me)_r \tag{12.10}$$

10. **Momentum and heat exchange:** The exchange of momentum and heat between materials is computed according to:

$$(mu)_r^L = (mu)_r^{L-} + \Delta t\ m_r \sum_{s=1}^{N} \theta_r\theta_s K_{rs}(u_s^L - u_r^L) \tag{12.11}$$

$$(me)_r^L = (me)_r^{L-} + \Delta t\ m_r\ c_{v_r} \sum_{s=1}^{N} \theta_r\theta_s H_{rs}\left(T_s^L - T_r^L\right) \tag{12.12}$$

These equations are solved in a pointwise implicit manner that allows arbitrarily large momentum transfer to take place between materials. Typically, in FSI solutions, very large (10^{15}) values of K are used, which results in driving contacting materials to the same velocity. Intermaterial heat exchange is usually modeled at a lower rate. Again, note that the same operation must be done following Step 3 above in the computation of the face-centered velocities.

11. **Specific volume evolution:** As discussed above in step 2, in order to correctly compute the equilibrium pressure and the volume fraction, it is necessary to keep an accurate accounting of the specific volume for each material. Here, we compute the evolution in specific volume due to the changes in temperature and pressure, as well as phase change, during the foregoing Lagrangian portion of the calculation, according to:

$$\Delta(mv)_r = \Delta t\ V \left[v_r\Gamma_r + f^{\theta r}\nabla\cdot\sum_{s=1}^{N}\theta_s^* u_s^* + \theta_r\beta_r\dot{T}_r - f^{\theta r}\sum_{s=1}^{N}\theta_s\beta_s\dot{T}_s \right] \tag{12.13}$$

$$(mv)_r^L = (mv)_r^n + \Delta(mv)_r \tag{12.14}$$

where β is the constant pressure thermal expansivity and $\dot{T} = \frac{T^L - T^t}{\Delta t}$ is the rate of change of each material's temperature during the Lagrangian phase of the computation.

12. **Advect Fluids:** For the fluid phase, use a suitable advection scheme, such as that described in [95], to transport mass, momentum, internal energy and specific volume. As this last item is an intensive quantity, it is converted to material volume for advection, and then reconstituted as specific volume for use in the subsequent timestep's equilibrium pressure calculation.
13. **Update Nodal Quantities for Solid Materials:** Those changes in solid material mass, momentum and internal energy that are computed at the cell-centers are interpolated to the nodes as field quantities, e.g., changes in momentum are expressed as accelerations, for use in Eq. 1.63.
14. **Advect Solids:** For the solid phase, interpolate the time advanced grid velocity and the correspond-

ing velocity increment (acceleration) back to the particles, and use these to advance the particle's position and velocity, according to Eqs. 1.67.

This completes one timestep. In the preceding, the user has a number of options in the implementation. The approach taken here was to develop a working MPM code and a separate working multi-material ICE code. In addition, some routines specific to the integration are required, for example, to transfer data from grid nodes to cell-centers. We note, however, that the fluid structure interaction methodology should not be looked at in the context of a "marriage" between an Eulerian CFD code and MPM. The underlying theory is a multi-material description that has the flexibility to incorporate different numerical descriptions for solid and fluid fields within the overarching solution process. To have flexibility in treating a widest range of problems, it was our desire that in the integration of the two algorithms, each of the components be able to function independently. As described here, this method is fully explicit in time. To make this implicit with respect to the propagation of pressure waves, a Poisson equation is solved in the calculation of Δp, which is in turn used to iteratively update the face-centered velocities [90].

12.2 Models

The governing equations given in Section 10.1.1 are incomplete without closure equations for quantities such as pressure, stress, and rate of exchange of mass between materials. Equations of state, constitutive models and reaction models provide the needed closure. Some ICE material models have been discussed in Chapter 11. Materials used by the MPM component are discussed elsewhere in the Vaango Theory Manual.

12.3 Numerical Results

The simulation results presented here are intended to serve two purposes, to validate the method presented above, and to demonstrate its capabilities. While results from some very basic validation tests can be found in [96], the validation tests presented here are targeted toward exploding energetic devices. Extensive experimental data have been collected for the first two cases, and these data are compared with simulation results.

The first test, detonation of a series of cylinders of explosive, validates both the general multi-material framework, including material transformation, as well as the detonation model itself. In the second test, a cylinder of explosive confined in a copper tube is detonated. There, the confidence gained from the first test is built upon and extended to include the interaction of the highly pressurized product gases with the confining copper cylinder. Wall velocity of the copper tube is compared with experimental measurements.

For the last case, a steel cylinder filled with PBX-9501 is heated to the critical temperature to commence a deflagration. The simulation continues through the rupture of the case when product gases are free to interact with the surrounding air. This simulation demonstrates a unique capability of this approach, in which initially separate fluid regions are allowed to interact following the failure of the steel container.

12.3.1 Rate Stick Simulations

A well known phenomenon of detonating solid high explosives is the so-called "size effect". The size effect refers to the change of the steady state detonation velocity of explosives, U_s with size R_0 [83]. In order to validate our implementation of the JWL++ detonation model within our multi-material framework, a parameter study was conducted for cylinders of Ammonium Nitrate Fuel Oil (ANFO-K1) with length of 10 cm and radii ranging from 4 mm to 20 mm. In addition, a one-dimensional simulation provided for the "infinite radius" case. In each of the finite radius cases, the cylinder was initially surrounded by air. Detonation was initiated by impacting the cylinder at 90 m/s against the boundary of the computational domain, at which a zero velocity Dirichlet boundary condition was imposed. This impact was sufficient

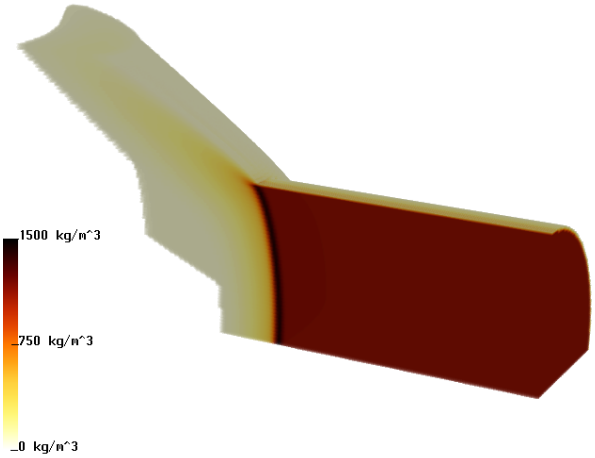


Figure 12.1: Unconfined 12 mm "rate-stick". The mass density of the reactant material is volume rendered, and shows evidence of the curvature of the reaction front, and the compression of the reactant just ahead of the reaction. Behind the detonation, most of the reactant material is consumed.

to raise the pressure within the cylinder to above the threshold for initiation of reaction. The detonation velocity was determined by comparing the arrival time of the detonation at two points along the cylinder, sufficiently into the far field that the detonation had reached a steady state.

Material properties for these cases included the following: The reactant was described by a Murnaghan equation of state with parameters $n = 7.4$, $\kappa = 3.9\times10^{11}$ Pa^{-}1 and $\rho_0 = 1160.0$ kg/m^3. The products of reaction were described by a JWL C-term form equation of state with parameters $A = 2.9867\times10^{11}$ Pa, $B = 4.11706\times10^{9}$ Pa, $C = 7.206147\times10^{8}$ Pa, $R_1 = 4.95$, $R_2 = 1.15$, $\omega = 0.35$ and $\rho_0 = 1160.0$ kg/m^3. The JWL++ parameters were taken as: $G = 3.5083\times10^{-7}$ s^{-1}Pa^{b}, $b = 1.3$, $\rho_0 = 1160.0$ kg/m^3. In all, this simulation included 3 materials; the reactant material, the products of reaction and the surrounding air.

Simulations were carried out on uniform meshes with cell sizes of 1.0 mm, 0.5 mm and 0.25 mm. A one-quarter symmetry was assumed in all cases. A qualitative representation is shown in Figure 12.1, which depicts a volume rendering of the density of the reactant as the detonation has progressed about halfway into the material for the 12 mm radius case at the finest resolution. The curvature of the burn front and the elevated density just ahead of it are evident in this view.

Figure 12.2 is a plot of detonation velocity *versus* the inverse of the sample radius. Experimental data are represented by open squares, while results of the simulations are shown with filled circles (h = 1.0 mm), filled diamonds (h = 0.5 mm) and filled triangles (h = 0.25 mm). Connecting lines for the numerical data are in place to guide the eyes of the reader. Evident from this plot is the convergence of detonation velocities with grid resolution, and the generally good agreement between experimental and computed detonation velocities at the finer grid resolutions, particularly at the larger radii, where both the experimental data and the model are considered more reliable.

Again, while this set of tests doesn't validate the full fluid-structure interaction approach, it does give credibility to the underlying multi-material formulation, including the pressure equilibration and the exchange of mass between materials, in this case as governed by the JWL++ detonation model, as well as momentum and energy.

12.3.2 Cylinder Test Simulation

The cylinder test is an experiment which is frequently used to calibrate equations of state for detonation products of reaction [97]. In this case, the test consists of an OFHC copper tube with an inner radius of 2.54 cm, an outer radius of 3.06 cm and a length of 35 cm. The tube is filled with QM-100, an Ammonium

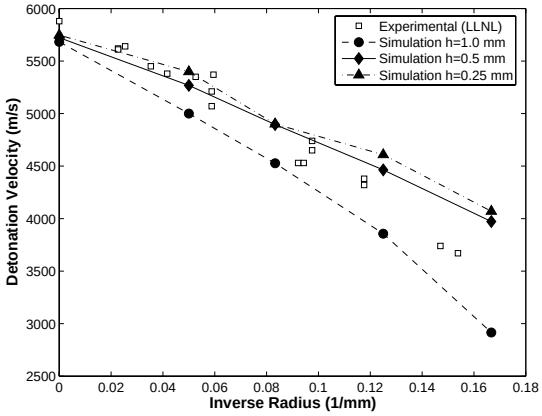


Figure 12.2: Detonation velocity vs. inverse radius. Experimental and numerical data are presented, and indicate good agreement of the model with experiment, as well as convergence of detonation velocity with grid resolution.

Nitrate emulsion and a detonation is initiated at one end of the tube. Measurements of the wall velocity wall are made at individual points along the length of the tube using Fabry-Perot interferometry or streak cameras.

A simulation of this configuration was performed and wall velocity data were collected at an axial location 25 cm from the point of initiation. The reactant was again described by a Murnaghan equation of state with parameters $n = 7.0$, $\kappa = 1.02\times10^{-9}$ Pa^{-1} and $\rho_0 = 1260.0$ kg/m^3. The products of reaction were described by a JWL C-term form equation of state with parameters $A = 4.8702\times10^{11}$ Pa, $B = 2.54887\times10^{9}$ Pa, $C = 5.06568\times10^{8}$ Pa, $R_1 = 5.0$, $R_2 = 1.0$, $\omega = 0.3$ and $\rho_0 = 1260.0$ kg/m^3. The JWL++ parameters were taken as: $G = 9.1\times10^{-5}$s^{-1} Pa, $b = 1.0$, $\rho_0 = 1260.0$ kg/m^3. The copper tube was modeled as an elastic-perfectly plastic material with a density of 8930.0 kg/m^3, bulk and shear moduli of 117.0 GPa and 43.8 GPa, respectively,and a yield stress of 70.0 MPa. The copper tube was surrounded by air. In all, 4 materials are present in this simulation, the reactant, the products of reaction, the copper tube, and the surrounding air.

Again, a one-quarter symmetry section of the full cylinder was modeled using a cell size of h = 0.5 mm and a total domain size of 35 cm × 6 cm × 6 cm. Zero gradient conditions described the exterior boundaries, which allowed material to exit the domain.

Figure 12.3 shows a snapshot of this test midway through the simulation, at t = 18.8 μs. The copper tube is depicted using an iso-surface of the cell-centered mass density (the two surfaces are the inner and outer walls of the tube) that is colored by velocity. A volume rendering of the pressure field is also present. Alternating bands of high and low velocity of the tube wall are evidently due to the reflection of the impulse provided by the shock between the inner and outer surfaces of the tube.

Velocity data was collected from those particles which were both initially at an axial location of 25 cm, and upon the exterior surface of the tube. The velocity from this collection of particles was averaged over the circumference and plotted vs. time in Figure 12.4. In addition, experimental results (LLNL, Shot No. K260-581) are also shown. Both datasets are time shifted to coincide with the arrival of the detonation. Good agreement is evident between the experimental and numerical data, further indicating the validity of the approach described here.

12.3.3 Fast Cookoff Simulation

Cookoff tests, generally speaking, refer to experiments in which energetic material is heated until it reaches ignition. The rate of heating typically differentiates these tests in to “fast” or “slow” cookoff. In slow cookoff tests, the temperature is usually increased very slowly, perhaps a few degrees per hour, so

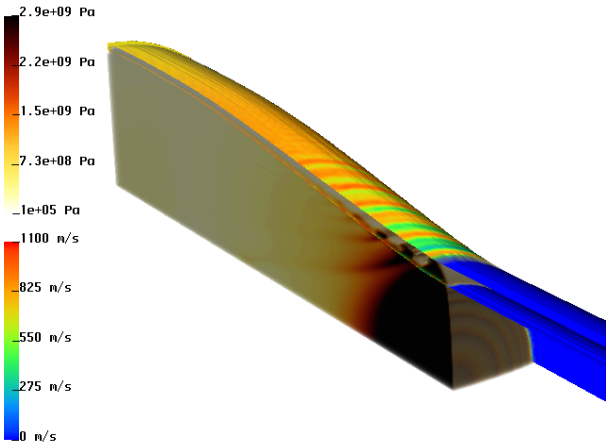


Figure 12.3: Copper cylinder test simulation. The walls of the copper tube are depicted as an isosurface of density of the copper material and are colored by velocity magnitude. Pressure is represented by a volume rendering, and indicates the progress of the detonation, as well as the interaction of the pressurized products of reaction with the confining walls.

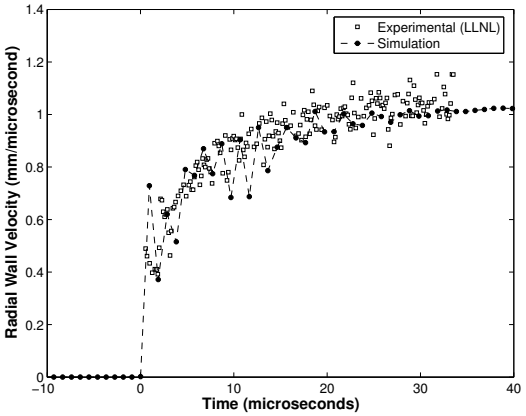


Figure 12.4: Copper cylinder test simulation. Experimental and computational velocities of the cylinder vs. time. Data was collected at a point 25 cm from the point of initiation of the detonation.

Table 12.1: Material constants for 4340 steel.

ρ (kg/m^3)	K (GPa)	μ (GPa)	T_o (K)	T_m (K)	C_o (m/s)	Γ_o	S_α		
7830.0	173.3	80.0	294.0	1793.0	3574	1.69	1.92		
A (MPa)	B (MPa)	C	n	m	D_1	D_2	D_3	D_4	D_5
792.0	510.0	0.014	0.26	1.03	0.05	3.44	-2.12	0.002	0.61

that the entire sample is able to equilibrate and is nearly isothermal when ignition occurs. In fast cookoff tests, heat is added to the system quickly, which is likely to lead to relatively local ignition at the surface of the sample. Fast cookoff is more likely to occur in an accident scenario, where ordinance may be subject to heating by a fire, as occurred on the USS Forrestal in 1967.

The scenario considered here consists of a cylindrical 4340 steel container with both inner diameter and length of 10.16 cm, and wall thickness of 0.635 cm, filled with PBX-9501. The temperature of the container was initialized to be 1^o K above the ignition temperature in the deflagration model for PBX-9501. In this way, the entire outer surface of the explosive is ignited simultaneously. This is, of course, somewhat unrealistic for an accident scenario, but rather is an idealization.

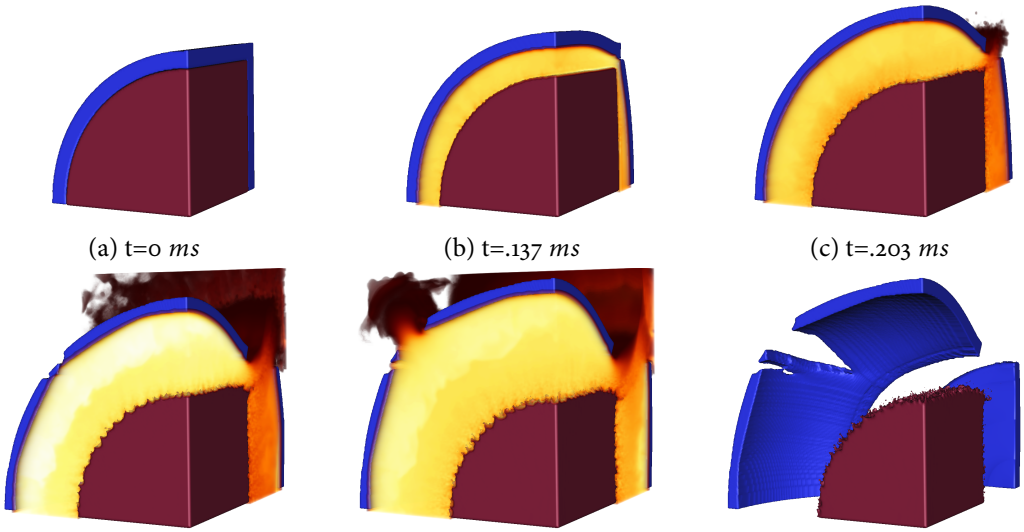
Mechanical properties for PBX 9501 were obtained from the literature [86], while the material constants used in the modeling of 4340 steel are shown in Table 12.1. A temperature-dependent specific heat model [98] was used to compute the internal energy and the rate of temperature increase in the material. We assumed an initial mean porosity of 0.005 with a standard deviation of 0.001. The critical porosity was 0.3. The mean strain at void nucleation was assumed to be 0.3 with a standard deviation of 0.1. The scalar damage variable was initialized with a mean of 0.005 and a standard deviation of 0.001.

Three planes of symmetry are assumed, which allows modeling only 1/8th of the total geometry. Each dimension of the computational domain was 9.0 cm discretized into 180 computational cells, for a grid spacing of h = 0.5 mm. Four materials were present, the steel container and the PBX-9501, each of which are treated in the Lagrangian frame of reference, as well as the air initially surrounding the container, and the products of reaction from the deflagration, both of which are represented in the Eulerian frame of reference. Neumann zero gradient boundary conditions are used on the exterior domain boundaries to allow material to flow out of the domain, as the explosion progressed.

Because of the size and complexity of this simulation, significant computational resources were required to obtain a solution. Namely, the simulation ran for about 48 hours on 600 processors of a Linux cluster at Lawrence Livermore National Laboratory, which resulted in 0.31 milliseconds of simulated time.

Results from this simulation are shown in Fig. 12.5. In each panel, the container and explosive are depicted by isosurfaces, blue and red, respectively. In Fig. 12.5b-12.5e, a volume rendering of the mass density of the product material of the reaction is also included. Fig. 12.5a shows the initial state of the geometry, while the remaining panels show the progression of the simulation at the times indicated in the captions. The last two panels depict the same time, with the product gas removed in the final panel, to more clearly show the state of the container at that time. Close comparison of the initial and final panels also reveals the reduction in size of the explosive pellet, due to the reaction. Product gas first begins to leave the container through a rupture where the side and end of the container meet(Fig. 12.5c), and ultimately also through a rupture in middle (Fig. 12.5e). The formation of these openings is governed by material localization.

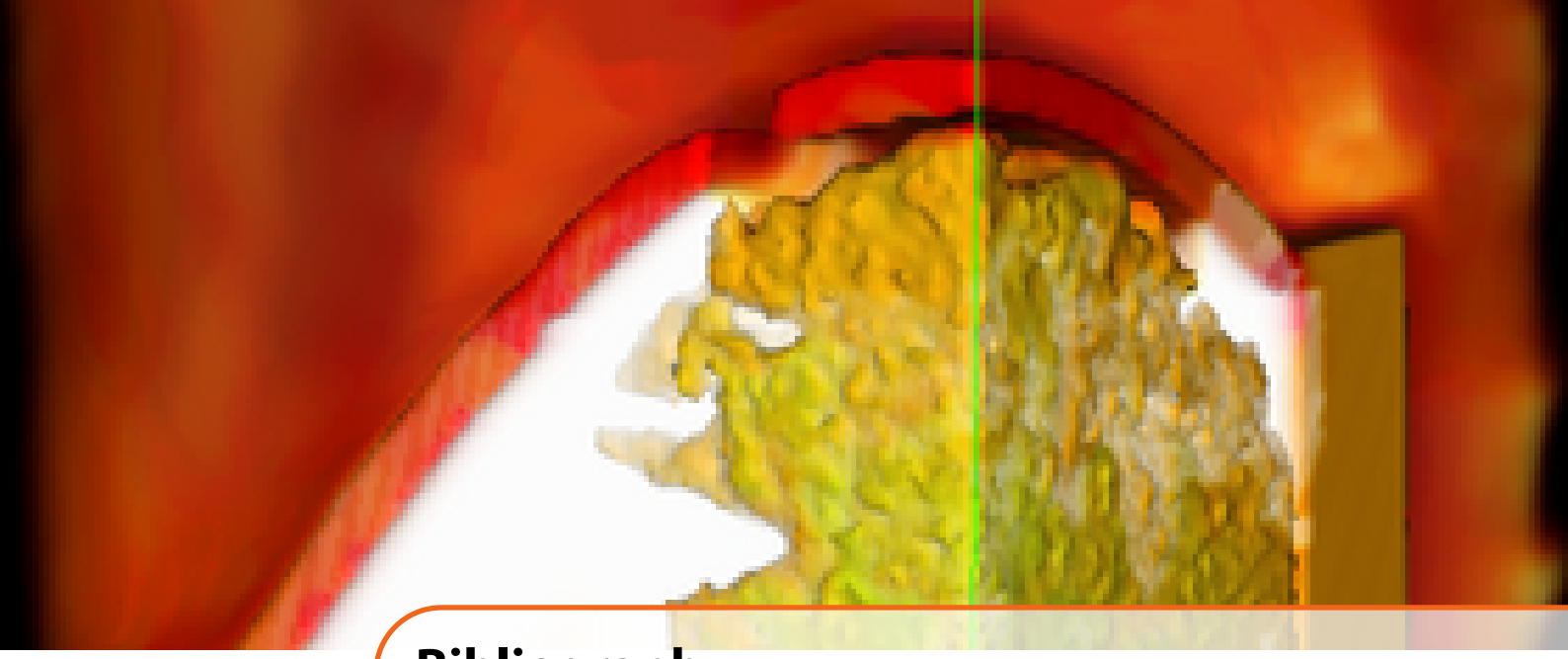
Since no surface tracking is required in this method, there is no requirement to track the creation of the new surfaces that occur due to material failure. Gas is free to escape through the openings simply because there is no longer anything in those computational cells to prevent it once the gap is sufficiently wide.



(a) t=0 *ms* (b) t=.137 *ms* (c) t=.203 *ms*

(d) t=.259 *ms* (e) t=.312 *ms* (f) t=.312 *ms*

Figure 12.5: Time series of a steel container (blue) filled with deflagrating plastic bonded explosive(red). A volume rendering of the mass density of the products of reaction is also shown, except in the final panel, where it is removed to more clearly show the regions where the container has failed.



Bibliography

[1] D. Sulsky, Z. Chen, and H. L. Schreyer. "A particle method for history dependent materials". In: *Comput. Methods Appl. Mech. Engrg.* 118 (1994), pages 179–196 (cited on pages 7, 15, 111, 112, 127, 129).

[2] D. Sulsky, S. Zhou, and H. L. Schreyer. "Application of a particle-in-cell method to solid mechanics". In: *Computer Physics Communications* 87 (1995), pages 236–252 (cited on pages 7, 11, 13, 15, 127, 129).

[3] J.U. Brackbill and H.M. Ruppel. "FLIP: A Low-Dissipation, Particle-In-Cell Method for Fluid Flows in Two Dimensions". In: *J. Comp. Phys.* 65 (1986), pages 314–343 (cited on pages 7, 129).

[4] F.H. Harlow. "The particle-in-cell computing method for fluid dynamics". In: *Methods Comput. Phys.* 3 (1963), pages 319–343 (cited on page 7).

[5] D. Sulsky and H.L. Schreyer. "Axisymmetric form of the material point method with applications to upsetting and Taylor impact problems". In: *Computer Methods in Applied Mechanics and Engineering* 139 (1996), pages 409–429 (cited on page 7).

[6] S.G. Bardenhagen, J.U. Brackbill, and D. Sulsky. "The material-point method for granular materials". In: *Comput. Methods Appl. Mech. Engrg.* 187 (2000), pages 529–541 (cited on pages 8, 112).

[7] S. G. Bardenhagen and E. M. Kober. "The Generalized Interpolation Material Point Method". In: *Comp. Model. Eng. Sci.* 5.6 (2004), pages 477–495 (cited on pages 8, 11, 15, 18).

[8] Alireza Sadeghirad, Rebecca M Brannon, and Jeff Burghardt. "A convected particle domain interpolation technique to extend applicability of the material point method for problems involving massive deformations". In: *International Journal for numerical methods in Engineering* 86.12 (2011), pages 1435–1456 (cited on pages 8, 12, 21).

[9] A Sadeghirad, RM Brannon, and JE Guilkey. "Second-order convected particle domain interpolation (CPDI2) with enrichment for weak discontinuities at material interfaces". In: *International Journal for numerical methods in Engineering* 95.11 (2013), pages 928–952 (cited on page 8).

[10] RB Leavy et al. "A convected-particle tetrahedron interpolation technique in the material-point method for the mesoscale modeling of ceramics". In: *Computational Mechanics* 64.3 (2019), pages 563–583 (cited on page 8).

[11] B.A. Kashiwa. *A multifield Model and Method for Fluid-Structure Interaction Dynamics*. Technical report LA-UR-01-1136. Los Alamos: Los Alamos National Laboratory, 2001 (cited on pages 8, 119, 121, 130).

[12] P. C. Wallstedt and J. E. Guilkey. "An evaluation of explicit time integration schemes for use with the generalized interpolation material point method". In: *J. Comp. Phys.* 227 (2008), pages 9628–9642 (cited on page 16).

[13] S.G. Bardenhagen et al. "An Improved Contact Algorithm for the Material Point Method and Application to Stress Propagation in Granular Material". In: *Computer Modeling in Engineering and Sciences* 2 (2001), pages 509–522 (cited on pages 17, 22, 112).

[14] J. Ma, H. Lu, and R. Komanduri. "Structured Mesh Refinement in Generalized Interpolation Material Point Method (GIMP) for Simulation of Dynamic Problems". In: *Computer Modeling in Engineering and Sciences* 12 (2006), pages 213–227 (cited on page 21).

[15] John A Nairn, Chad C Hammerquist, and Grant D Smith. "New material point method contact algorithms for improved accuracy, large-deformation problems, and proper null-space filtering". In: *Computer Methods in Applied Mechanics and Engineering* 362 (2020), page 112859 (cited on page 22).

[16] S. Nemat-Nasser. "Rate-independent finite-deformation elastoplasticity: a new explicit constitutive algorithm". In: *Mech. Mater.* 11 (1991), pages 235–249 (cited on page 38).

[17] S. Nemat-Nasser and D. -T. Chung. "An explicit constitutive algorithm for large-strain, large-strain-rate elastic-viscoplasticity". In: *Comput. Meth. Appl. Mech. Engrg* 95.2 (1992), pages 205–219 (cited on pages 38, 40).

[18] L. H. Wang and S. N. Atluri. "An analysis of an explicit algorithm and the radial return algorithm, and a proposed modification, in finite elasticity". In: *Computational Mechanics* 13 (1994), pages 380–389 (cited on page 38).

[19] P. J. Maudlin and S. K. Schiferl. "Computational anisotropic plasticity for high-rate forming applications". In: *Comput. Methods Appl. Mech. Engrg.* 131 (1996), pages 1–30 (cited on page 38).

[20] M. A. Zocher et al. "An evaluation of several hardening models using Taylor cylinder impact data". In: *Proc., European Congress on Computational Methods in Applied Sciences and Engineering*. ECCOMAS. Barcelona, Spain, 2000 (cited on pages 38, 42, 44).

[21] JC Simo and TJR Hughes. *Computational Inelasticity*. New York: Springer-Verlag, 1998 (cited on pages 40, 112).

[22] M. L. Wilkins. *Computer Simulation of Dynamic Phenomena*. Berlin: Springer-Verlag, 1999 (cited on page 42).

[23] D. J. Steinberg, S. G. Cochran, and M. W. Guinan. "A constitutive model for metals applicable at high-strain rate". In: *J. Appl. Phys.* 51.3 (1980), pages 1498–1504 (cited on pages 42, 44, 45).

[24] L. Burakovsky and D. L. Preston. "Analysis of dislocation mechanism for melting of elements". In: *Solid State Comm.* 115 (2000), pages 341–345 (cited on page 43).

[25] Y. P. Varshni. "Temperature dependence of the elastic constants". In: *Physical Rev. B* 2.10 (1970), pages 3952–3958 (cited on page 44).

[26] S. R. Chen and G. T. Gray. "Constitutive behavior of tantalum and tantalum-tungsten alloys". In: *Metall. Mater. Trans. A* 27A (1996), pages 2994–3006 (cited on page 44).

[27] M.-H. Nadal and P. Le Poac. "Continuous model for the shear modulus as a function of pressure and temperature up to the melting point: analysis and ultrasonic validation". In: *J. Appl. Phys.* 93.5 (2003), pages 2472–2480 (cited on page 44).

[28] G. R. Johnson and W. H. Cook. "A constitutive model and data for metals subjected to large strains, high strain rates and high temperatures". In: *Proc. 7th International Symposium on Ballistics*. 1983, pages 541–547 (cited on page 45).

[29] D. J. Steinberg and C. M. Lund. "A constitutive model for strain rates from 10^{-4} to 10^{6} s^{-1}". In: *J. Appl. Phys.* 65.4 (1989), pages 1528–1533 (cited on page 45).

[30] K. G. Hoge and A. K. Mukherjee. "The temperature and strain rate dependence of the flow stress of tantalum". In: *J. Mater. Sci.* 12 (1977), pages 1666–1672 (cited on page 46).

[31] F. J. Zerilli and R. W. Armstrong. "Dislocation-mechanics-based constitutive relations for material dynamics calculations". In: *J. Appl. Phys.* 61.5 (1987), pages 1816–1825 (cited on page 46).

[32] F. J. Zerilli and R. W. Armstrong. "Constitutive relations for the plastic deformation of metals". In: *High-Pressure Science and Technology - 1993*. American Institute of Physics. Colorado Springs, Colorado, 1993, pages 989–992 (cited on page 46).

[33] F. J. Zerilli. "Dislocation mechanics-based constitutive equations". In: *Metall. Mater. Trans. A* 35A (2004), pages 2547–2555 (cited on page 46).

[34] F. H. Abed and G. Z. Voyiadjis. "A consistent modified Zerilli-Armstrong flow stress model for bcc and fcc metals for elevated temperatures". In: *Acta Mechanica* 175 (2005), pages 1–18 (cited on page 46).

[35] P. S. Follansbee and U. F. Kocks. "A Constitutive Description of the Deformation of copper Based on the Use of the Mechanical Threshold Stress as an Internal State Variable". In: *Acta Metall.* 36 (1988), pages 82–93 (cited on page 47).

[36] D. M. Goto et al. "Anisotropy-corrected MTS constitutive strength modeling in HY-100 steel". In: *Scripta Mater.* 42 (2000), pages 1125–1131 (cited on page 47).

[37] U. F. Kocks. "Realistic constitutive relations for metal plasticity". In: *Materials Science and Engrg.* A317 (2001), pages 181–187 (cited on page 47).

[38] D. L. Preston, D. L. Tonks, and D. C. Wallace. "Model of plastic deformation for extreme loading conditions". In: *J. Appl. Phys.* 93.1 (2003), pages 211–220 (cited on page 48).

[39] G. Ravichandran et al. "On the conversion of plastic work into heat during high-strain-rate deformation". In: *Proc. , 12th APS Topical Conference on Shock Compression of Condensed Matter*. American Physical Society. 2001, pages 557–562 (cited on page 49).

[40] F. L. Lederman, M. B. Salamon, and L. W. Shacklette. "Experimental verification of scaling and test of the universality hypothesis from specific heat data". In: *Phys. Rev. B* 9.7 (1974), pages 2981–2988 (cited on page 49).

[41] J. O. Coplien. *Advanced C++ Programming Styles and Idioms*. Reading, MA: Addison-Wesley, 1992 (cited on page 50).

[42] B. Banerjee. "Material Point Method simulations of fragmenting cylinders". In: *Proc. 17th ASCE Engineering Mechanics Conference (EM2004)*. Newark, Delaware, 2004 (cited on page 50).

[43] B. Banerjee. *MPM Validation: Sphere-Cylinder Impact Tests: Energy Balance*. Technical report C-SAFE-CD-IR-04-001. University of Utah, USA: Center for the Simulation of Accidental Fires and Explosions, 2004. URL: http://www.csafe.utah.edu/documents/C-SAFE-CD-IR-04-001.pdf (cited on page 50).

[44] B. Banerjee. "Validation of UINTAH: Taylor impact and plasticity models". In: *Proc. 2005 Joint ASME/ASCE/SES Conference on Mechanics and Materials (McMat 2005)*. Baton Rouge, LA, 2005 (cited on page 50).

[45] A. L. Gurson. "Continuum theory of ductile rupture by void nucleation and growth: Part 1. Yield criteria and flow rules for porous ductile media". In: *ASME J. Engg. Mater. Tech.* 99 (1977), pages 2–15 (cited on page 50).

[46] V. Tvergaard and A. Needleman. "Analysis of the cup-cone fracture in a round tensile bar". In: *Acta Metall.* 32.1 (1984), pages 157–169 (cited on page 50).

[47] S. Ramaswamy and N. Aravas. "Finite element implementation of gradient plasticity models Part II: Gradient-dependent evolution equations". In: *Comput. Methods Appl. Mech. Engrg.* 163 (1998), pages 33–53 (cited on page 52).

[48] C. C. Chu and A. Needleman. "Void nucleation effects in biaxially stretched sheets". In: *ASME J. Engg. Mater. Tech.* 102 (1980), pages 249–256 (cited on page 52).

[49] G. R. Johnson and W. H. Cook. "Fracture characteristics of three metals subjected to various strains, strain rates, temperatures and pressures". In: *Int. J. Eng. Fract. Mech.* 21 (1985), pages 31–48 (cited on page 52).

[50] G. R. Johnson and T. J. Holmquist. "Evaluation of cylinder-impact test data for constitutive models". In: *J. Appl. Phys.* 64.8 (1988), pages 3901–3910 (cited on page 53).

[51] D. C. Drucker. "A definition of stable inelastic material". In: *J. Appl. Mech.* 26 (1959), pages 101–106 (cited on page 53).

[52] J. W. Rudnicki and J. R. Rice. "Conditions for the localization of deformation in pressure-sensitive dilatant materials". In: *J. Mech. Phys. Solids* 23 (1975), pages 371–394 (cited on page 53).

[53] P. Perzyna. "Constitutive modelling of dissipative solids for localization and fracture". In: *Localization and Fracture Phenomena in Inelastic Solids: CISM Courses and Lectures No. 386*. Edited by Perzyna P. New York: SpringerWien, 1998, pages 99–241 (cited on page 53).

[54] R. Becker. "Ring fragmentation predictions using the Gurson model with material stability conditions as failure criteria". In: *Int. J. Solids Struct.* 39 (2002), pages 3555–3580 (cited on page 53).

[55] R. Hill and J. W. Hutchinson. "Bifurcation phenomena in the plane tension test". In: *J. Mech. Phys. Solids* 23 (1975), pages 239–264 (cited on page 54).

[56] Z. P. Bazant and T. Belytschko. "Wave propagation in a strain-softening bar: Exact solution". In: *ASCE J. Engg. Mech* 111.3 (1985), pages 381–389 (cited on page 54).

[57] V. Tvergaard and A. Needleman. "Ductile failure modes in dynamically loaded notched bars". In: *Damage Mechanics in Engineering Materials: AMD 109/MD 24*. Edited by J. W. Ju, D. Krajcinovic, and H. L. Schreyer. New York, NY: American Society of Mechanical Engineers, 1990, pages 117–128 (cited on page 54).

[58] S. Ramaswamy and N. Aravas. "Finite element implementation of gradient plasticity models Part I: Gradient-dependent yield functions". In: *Comput. Methods Appl. Mech. Engrg.* 163 (1998), pages 11–32 (cited on page 54).

[59] S. Hao, W. K. Liu, and D. Qian. "Localization-induced band and cohesive model". In: *J. Appl. Mech.* 67 (2000), pages 803–812 (cited on page 54).

[60] K. H. Roscoe and A. N. Schofield. "Mechanical behaviour of an idealized wet clay". In: *presented at the Proc. 2nd Eur. Conf SMFE, Wiesbaden* 1 (1963), pages 47–54 (cited on page 71).

[61] K. H. Roscoe and J. B. Burland. *On the generalized stress-strain behavior of wet clay*. 1968 (cited on page 71).

[62] A. N. Schofield and P. Wroth. *Critical state soil mechanics*. McGraw-Hill, 1968 (cited on page 71).

[63] P. C. Wroth and G. T. Houlsby. *A Critical State Model for Predicting the Behaviour of Clays*. presented at the Application of Plasticity and Generalized Stress-strain in Geotechnical Engineering, 1986, pages 592–627 (cited on page 71).

[64] R. I. Borja and S. R. Lee. "Cam-Clay plasticity, Part 1: Implicit integration of elasto-plastic constitutive relations". In: *Computer Methods in Applied Mechanics and Engineering* 78.1 (Jan. 1990), pages 49–72 (cited on page 71).

[65] R. I. Borja. "Cam-Clay plasticity. II: Implicit integration of constitutive equation based on a nonlinear elastic stress predictor". In: *Computer Methods in Applied Mechanics and Engineering* 88.2 (1991), pages 225–240 (cited on page 71).

[66] R. I. Borja, C. Tamagnini, and A. Amorosi. "Coupling Plasticity and Energy-Conserving Elasticity Models for Clays". In: *Journal of Geotechnical and Geoenvironmental Engineering* 123.10 (Oct. 1997), pages 948–957 (cited on page 71).

[67] R. I. Borja and C. Tamagnini. "Cam-Clay plasticity part III: Extension of the infinitesimal model to include finite strains". In: *Computer Methods in Applied Mechanics and Engineering* 155 (Mar. 1998), pages 73–95 (cited on page 71).

[68] T. Belytschko, W. K. Liu, and B. Moran. *Nonlinear Finite Elements for Continua and Structures*. New York: John Wiley and Sons, Ltd., 2000 (cited on pages 109, 117).

[69] M. W. Lewis, B. A. Kashiwa, and R. M. Rauenzahn. "Hydrodynamic ram modeling with the immersed boundary method". In: *Proc. , ASME Pressure Vessels and Piping Conference*. American Society of Mechanical Engineers. San Diego, California, 1998 (cited on pages 109, 110, 114, 118).

[70] H. L. Schreyer. *Lecture Notes in Plate Theory*. Albuquerque: University of New Mexico, 1997 (cited on page 110).

[71] S. G. Bardenhagen, J. U. Brackbill, and D. Sulsky. "Numerical Study of Stress Distributions in Sheared Granular Material in Two Dimensions". In: *Phys. Rev. E* 62 (2000), pages 3882–3890 (cited on pages 111, 112).

[72] B. A. Kashiwa and M. W Lewis. *CFDLIB version 02. 1*. Los Alamos National Laboratories. Los Alamos, New Mexico, 2002 (cited on page 115).

[73] J. E. Guilkey and J. A. Weiss. "A general membrane formulation for use with the material point method". In preparation. 2002 (cited on page 117).

[74] A. Libai and J. G. Simmonds. *The Nonlinear Theory of Elastic Shells: 2nd Edition*. Cambridge, UK: Cambridge University Press, 1998 (cited on page 117).

[75] R. H. MacNeal and R. L. Harder. "A proposed standard set of problems to test finite element accuracy". In: *Finite Elements in Analysis and Design* 11 (1985), pages 3–20 (cited on page 117).

[76] S. Li, W. Hao, and W. K. Liu. "Numerical simulations of large deformation of thin shell structures using meshfree methods". In: *Computational Mechanics* 25.2-3 (2000), pages 102–116 (cited on page 117).

[77] G. Ayton et al. "Interfacing molecular dynamics and macro-scale simulations for lipid bilayer vesicles". In: *Biophysical Journal* 83 (2002), pages 1026–1038 (cited on page 117).

[78] J. E. Guilkey and J. A. Weiss. "Implicit time integration for the material point method: Quantitative and algorithmic comparisons with the finite element method". In: *Int. J. Numer. Meth. Engng.* 57.9 (2003), pages 1323–1338 (cited on page 117).

[79] C. Jiang, T. Gast, and J. Teran. "Anisotropic Elastoplasticity for Cloth, Knit and Hair Frictional Contact". In: *ACM Transactions on Graphics* 36.4 (2017), 152:1–152:14 (cited on page 118).

[80] B.A. Kashiwa and R.M. Rauenzahn. *A Multimaterial Formalism*. Technical report LA-UR-94-771. Los Alamos: Los Alamos National Laboratory, 1994 (cited on page 120).

[81] F.H. Harlow and A.A. Amsden. "Numerical Calculation of Almost Incompressible Flow". In: *J. Comp. Phys.* 3 (1968), pages 80–93 (cited on page 121).

[82] B.A. Kashiwa and R.M. Rauenzahn. *A Cell-Centered ICE Method for Multiphase Flow Simulations*. Technical report LA-UR-93-3922. Los Alamos: Los Alamos National Laboratory, 1994 (cited on pages 121, 122).

[83] P.C. Souers et al. "JWL++: A simple reactive flow code package for detonation". In: *Propellants, Explosives and Pyrotechnics* 25 (2000), pages 54–58 (cited on pages 125, 132).

[84] F. D. Murnaghan. "The compressibility of media under extreme pressures". In: *Proc. Natl. Acad. Sci.* 30 (1944), pages 244–247 (cited on page 125).

[85] S.F. Son et al. "Burn Rate Measurements of HMX, TATB, DHT, DAAF and BTATz". In: *Proceedings of the Combustion Institute* 28 (2000), pages 919–924 (cited on page 126).

[86] R.M. Hackett and J.G. Bennett. "Implicit finite element material model for energetic particulate composite materials". In: *International Journal for Numerical Methods in Engineering* 49 (2000), pages 1191–1209 (cited on pages 126, 136).

[87] D. J. Benson. "A Multi-Material Eulerian Formulation for the Efficient Solution of Impact and Penetration Problems". In: *Comput. Mech.* 15 (1995), pages 558–557 (cited on page 127).

[88] D. J. Benson. "Eulerian Finite Element methods for the Micromechanics of Heterogeneous materials: Dynamic Prioritization of material Interfaces". In: *Comput. Methods Appl. Mech. Engrg.* 151 (1998), pages 343–360 (cited on page 127).

[89] F.H. Harlow and A.A. Amsden. "Numerical Calculation of Almost Incompressible Flow". In: *J. Comp. Phys.* 3 (1968), pages 80–93 (cited on page 128).

[90] B.A. Kashiwa and R.M. Rauenzahn. *A Cell-Centered ICE Method for Multiphase Flow Simulations*. Technical report LA-UR-93-3922. Los Alamos: Los Alamos National Laboratory, 1994 (cited on pages 128, 130, 132).

[91] J.E. Guilkey, J.B. Hoying, and J.A. Weiss. "Modeling of multicellular constructs with the Material Point Method". In: *Journal of Biomechanics* 0 (2006), page 1 (cited on page 129).

[92] A.D. Brydon et al. "Simulation of the Densification of Real Open-Celled Foam Microstructures". In: *Journal of the Mechanics and Physics of Solids* 53 (2005), pages 2638–2660 (cited on page 129).

[93] Y. Guo and J.A. Nairn. "Calculation of J-Integral and Stress Intensity Factors using the Material Point Method". In: *Computer Modeling in Engineering and Sciences* 6 (2004), pages 295–308 (cited on page 129).

[94] F.H. Harlow and A.A. Amsden. "Flow of Interpenetrating Material Phases". In: *J. Comp. Phys.* 18 (1975), pages 440–464 (cited on page 130).

[95] W.B. VanderHeyden and B.A. Kashiwa. "Compatible Fluxes for van Leer Advection". In: *J. Comp. Phys.* 146 (1998), pages 1–28 (cited on page 131).

[96] J.E. Guilkey, T.B. Harman, and B. Banerjee. "An Eulerian-Lagrangian Approach for Simulating Explosions of Energetic Devices". In: *Computers and Structures* 85 (2007), pages 660–674 (cited on page 132).

[97] P.C. Souers et al. "Detonation Energies from the Cylinder Test and CHEETAH V3.0". In: *Propellants, Explosives and Pyrotechnics* 26 (2001), pages 180–190 (cited on page 133).

[98] D. M. Goto et al. "The mechanical threshold stress constitutive-strength model description of HY-100 steel". In: *Metallurgical and Materials Transactions A* 31A (2000), pages 1985–1996 (cited on page 136).