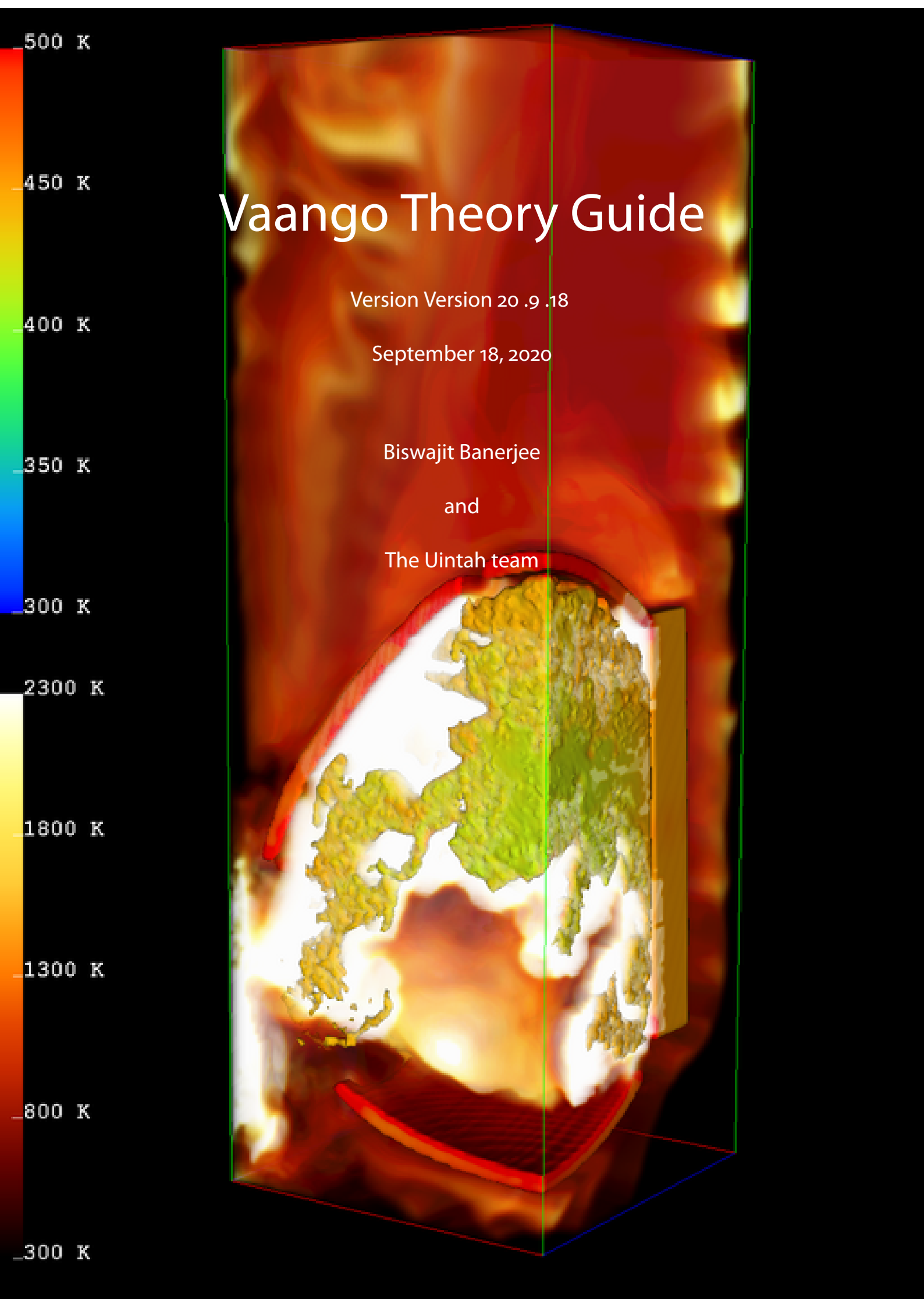


Vaango Theory Guide

Version Version 20 .9 .18

September 18, 2020

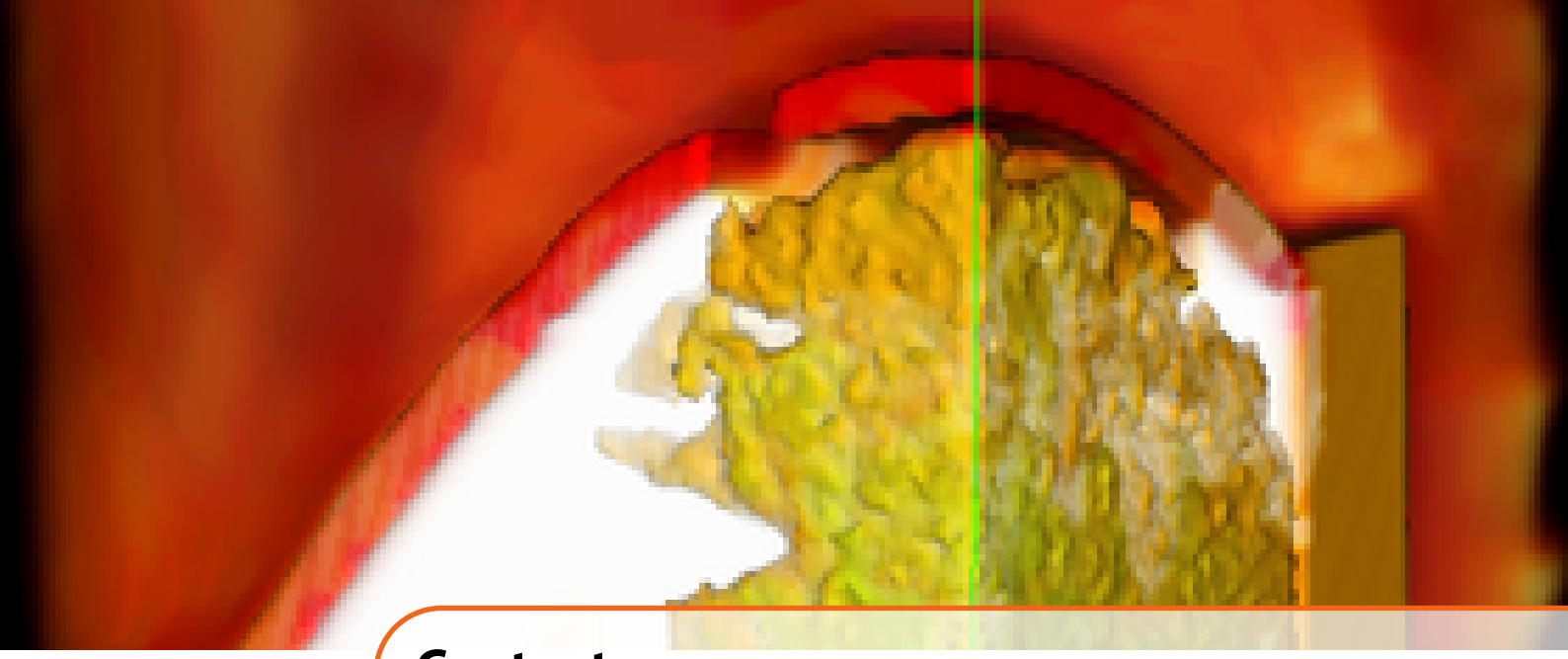
Biswajit Banerjee

and

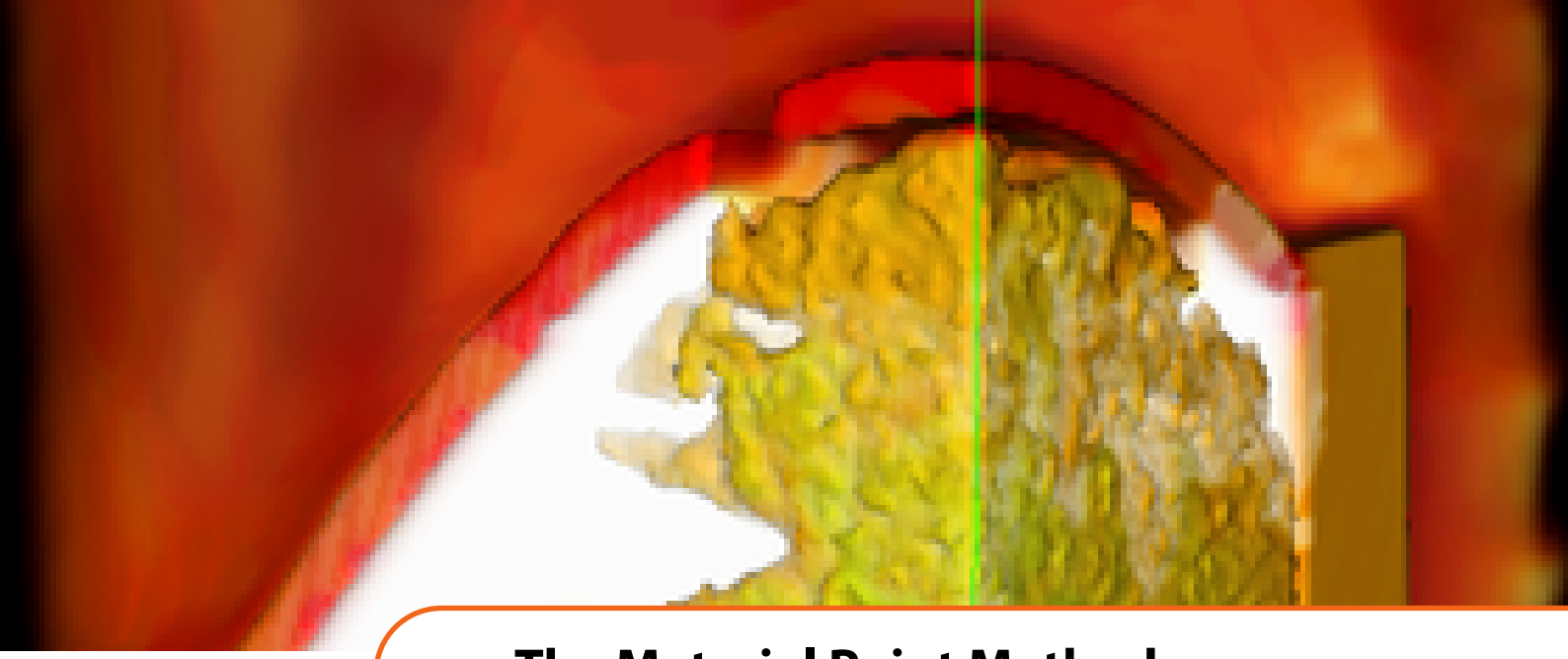
The Uintah team

Copyright © 2015-2020 Biswajit Banerjee

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 subscript 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

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

where $\mathbb{A}$ 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

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

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 $\mathbb{S}$ 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

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

A more compact matrix notation is used in [12]:

$$\boldsymbol{f}_p = \mathbb{S}\boldsymbol{f}_g \tag{1.24}$$

where $\boldsymbol{f}_p$ is a particle-based quantity matrix that has size $N_p \times 1$ for scalars, $N_p \times 3$ for vectors, and $N_p \times 6$ for symmetric second-order tensors. The matrix $\boldsymbol{f}_g$ are the corresponding grid quantities that have sizes $N_g \times 1$ for scalars, $N_g \times 3$ for vectors and $N_f \times 6$ for symmetric 2-tensors. The $\mathbb{S}$ matrix has size $N_p \times N_g$ with components $\left\langle S_{gp}\right\rangle$.

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

$$B_{gp} = V_p \left\langle S_{gp}\right\rangle \quad\Longrightarrow\quad \int_{\Omega_p} S_g(\mathbf{x})w(\mathbf{x})\,d\Omega = V_p \frac{\int_{\Omega_p} S_g(\mathbf{x})\omega_p(\mathbf{x})\,d\Omega}{\int_{\Omega_p} \omega_p(\mathbf{x})\,d\Omega} \tag{1.25}$$

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

$$\mathbf{f}_g = \sum_p \psi_{gp}\mathbf{f}_p \quad \text{where} \quad \psi_{gp} = \frac{V_p \left\langle S_{gp}\right\rangle}{\sum_p V_p \left\langle S_{gp}\right\rangle}\,,\quad \sum_p \psi_{gp} = 1\,. \tag{1.26}$$

The matrix notation used for the above relation in [12] is

$$\boldsymbol{f}_g = \mathbb{S}^+\boldsymbol{f}_p \tag{1.27}$$

where $\mathbb{S}^+$ is a $N_g \times N_p$ matrix.

1.3.1 Traditional MPM

If we wish to recover the traditional 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

$$\bar{S}_{gp} := \left\langle S_{gp}\right\rangle = \frac{\int_{\Omega_p} S_g(\mathbf{x})\,V_p\delta(\mathbf{x}-\mathbf{x}_p)\,d\Omega}{\int_{\Omega_p} V_p\delta(\mathbf{x}-\mathbf{x}_p)\,d\Omega} = S_g(\mathbf{x}_p)\,. \tag{1.28}$$

The gradient of the interpolation function evaluated at the particle is

$$\bar{\mathbf{G}}_{gp} = \left\langle \nabla S_{gp}\right\rangle = \frac{\int_{\Omega_p} \nabla S_g(\mathbf{x})\,V_p\delta(\mathbf{x}-\mathbf{x}_p)\,d\Omega}{\int_{\Omega_p} V_p\delta(\mathbf{x}-\mathbf{x}_p)\,d\Omega} = \nabla S_g(\mathbf{x}_p)\,. \tag{1.29}$$

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

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

The gradient of the interpolation function is

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

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:

$$\bar{S}_{gp} := \langle S_{gp} \rangle = \begin{cases} \frac{1}{V_p^\star} \int_{\Omega_p^\star} S_g^\star(\mathbf{x})\, d\Omega & \text{for } \mathbf{x} \in \Omega_p^\star \\ 0 & \text{otherwise.} \end{cases} \tag{1.32}$$

The gradient of the interpolation function is

$$\bar{\mathbf{G}}_{gp} = \langle \nabla S_{gp} \rangle = \begin{cases} \frac{1}{V_p^\star} \int_{\Omega_p^\star} \nabla S_g^\star(\mathbf{x})\, d\Omega & \text{for } \mathbf{x} \in \Omega_p^\star \\ 0 & \text{otherwise.} \end{cases} \tag{1.33}$$

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))

$$\mathbf{f}_p = \sum_g \mathbf{f}_g \bar{S}_{gp}\,. \tag{1.34}$$

In matrix form (see (1.24))

$$\boldsymbol{f}_p = \mathbb{S} \boldsymbol{f}_g\,. \tag{1.35}$$

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,

$$A_g = \int_\Omega \rho(\mathbf{x}) S_g(\mathbf{x})\, d\Omega = m_g\,, \quad B_{gp} = \int_\Omega \rho(\mathbf{x}) S_g(\mathbf{x}) \chi_p(\mathbf{x})\, d\Omega \tag{1.36}$$

and

$$\boxed{m_g \mathbf{f}_g = \sum_p \mathbf{f}_p \int_\Omega \rho(x) S_g(x) \chi_p(\mathbf{x})\, d\Omega = \sum_p \mathbf{f}_p m_p \bar{S}_{gp}\,.} \tag{1.37}$$

where m_g is the grid node mass and m_P is the particle mass. In matrix form equation 1.37 can be written as (see (1.27))

$$\boldsymbol{f}_g = \mathbb{S}^+ \boldsymbol{f}_p \quad \text{where} \quad \mathbb{S}^+ := \boldsymbol{m}_g^{-1} \mathbb{S}^T \boldsymbol{m}_p \tag{1.38}$$

where $\boldsymbol{m}_g$ is a $N_g \times N_g$ diagonal matrix that is invertible as long as $m_g \neq 0$, $\boldsymbol{m}_p$ is a $N_p \times N_p$ diagonal matrix, and $\mathbb{S}$ is a $N_p \times N_g$ matrix.

On the other hand, if $w(\mathbf{x}) = 1$, we have

$$A_g = \int_\Omega S_g(\mathbf{x})\, d\Omega = V_g\,, \quad B_{gp} = \int_\Omega S_g(\mathbf{x})\chi_p(\mathbf{x})\, d\Omega \tag{1.39}$$

and

$$\boxed{V_g \mathbf{f}_g = \sum_p \mathbf{f}_p \int_\Omega S_g(x)\chi_p(\mathbf{x})\, d\Omega = \sum_p \mathbf{f}_p V_p \bar{S}_{gp}\,.} \tag{1.40}$$

where V_g is the grid node volume and V_p is the particle volume. In matrix form equation (1.40) can be written as

$$\boldsymbol{f}_g = \mathbb{S}_V^+ \boldsymbol{f}_p \quad \text{where} \quad \mathbb{S}_V^+ := \boldsymbol{V}_g^{-1}\mathbb{S}^T \boldsymbol{V}_p \tag{1.41}$$

where $\boldsymbol{V}_g$ is a $N_g \times N_g$ diagonal matrix that is invertible as long as $V_g \neq 0$, and $\boldsymbol{V}_p$ is a $N_p \times N_p$ diagonal matrix.

In traditional MPM , the velocity ($\mathbf{v}$) is projected using mass weighting as per (1.37)), i.e.,

$$m_g \mathbf{v}_g = \sum_p m_p \mathbf{v}_p \bar{S}_{gp} \quad \text{or} \quad \boldsymbol{v}_g = \mathbb{S}^+ \boldsymbol{v}_p\,. \tag{1.42}$$

This implies that the mass density (ρ) and the momentum *per unit volume* ($\mathbf{P} = \rho\mathbf{v}$) are projected to grid nodes using the volume-weighted approach in (1.40):

$$\begin{aligned} V_g \rho_g &= \sum_p V_p \rho_p \bar{S}_{gp} \quad \text{or} \quad \boldsymbol{m}_g = \mathbb{S}^T \boldsymbol{m}_p \\ V_g \mathbf{P}_g &= \sum_p V_p \mathbf{P}_p \bar{S}_{gp} \quad \text{or} \quad \boldsymbol{p}_g = \mathbb{S}^T \boldsymbol{p}_p \end{aligned} \tag{1.43}$$

where $\boldsymbol{p}$ is a matrix of *total* momentum. 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

$$\int_{\Gamma_t} \bar{\mathbf{t}} \cdot \mathbf{w}\, d\Gamma - \int_\Omega \boldsymbol{\sigma} : \nabla \mathbf{w}\, d\Omega + \int_\Omega \rho\, \mathbf{w} \cdot \mathbf{b}\, d\Omega = \int_\Omega \rho\, \mathbf{w} \cdot \dot{\mathbf{v}}\, d\Omega\,. \tag{1.44}$$

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 (1.26) that

$$\mathbf{f}(\mathbf{x}) = \sum_g \mathbf{f}_g S_g(\mathbf{x}) \quad \text{and} \quad \mathbf{f}_g = \sum_p \psi_{gp} \mathbf{f}_p\,, \quad \psi_{gp} = \frac{V_p \langle S_{gp} \rangle}{\sum_p V_p \langle S_{gp} \rangle}\,, \quad \sum_p \psi_{gp} = 1\,. \tag{1.45}$$

Therefore, we can write

$$\mathbf{f}(\mathbf{x}) = \sum_g \sum_p \psi_{gp} \mathbf{f}_p S_g(\mathbf{x}) = \sum_p \mathbf{f}_p \sum_g \psi_{gp} S_g(\mathbf{x}) = \sum_p \mathbf{f}_p Y_p(\mathbf{x}) \tag{1.46}$$

where the particle basis functions , Y_p, are defined as

$$Y_p(\mathbf{x}) := \sum_g \psi_{gp} S_g(\mathbf{x}) = \frac{V_p \sum_g \langle S_{gp} \rangle S_g(\mathbf{x})}{\sum_p V_p \langle S_{gp} \rangle} \tag{1.47}$$

If we compare (1.46) with the particle representation in (1.5):

$$\mathbf{f}(\mathbf{x}) = \sum_p \mathbf{f}_p \chi_p(\mathbf{x}) \tag{1.48}$$

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

$$\int_\Omega w(\mathbf{x}) \chi_p(\mathbf{x}) = \int_\Omega w(\mathbf{x}) Y_p(\mathbf{x}) \tag{1.49}$$

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 (1.44) into a sum of integrals over particles using the particle basic functions, Y_p:

$$\begin{aligned} &\int_{\Gamma_t} \bar{\mathbf{t}}(\mathbf{x}) \cdot \mathbf{w}(\mathbf{x}) \, d\Gamma - \sum_p \int_{\Omega_p} Y_p(\mathbf{x}) \boldsymbol{\sigma}_p : \nabla \mathbf{w} \, d\Omega + \sum_p \int_{\Omega_p} Y_p(\mathbf{x}) \rho_p \mathbf{w}(\mathbf{x}) \cdot \mathbf{b}_p \, d\Omega \\ &\quad = \sum_p \int_{\Omega_p} Y_p(\mathbf{x}) \rho_p \mathbf{w}(\mathbf{x}) \cdot \dot{\mathbf{v}}(\mathbf{x}) \, d\Omega \, . \end{aligned} \tag{1.50}$$

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

$$\mathbf{w}(\mathbf{x}) = \sum_g \mathbf{w}_g S_g(\mathbf{x}) \, , \quad \mathbf{v}(\mathbf{x}) = \sum_h \mathbf{v}_h S_h(\mathbf{x}) \, , \quad \dot{\mathbf{v}}(\mathbf{x}) \approx \sum_h \dot{\mathbf{v}}_h S_h(\mathbf{x}) \, . \tag{1.51}$$

Plugging these into the left hand side of (1.50) we get

$$\begin{aligned} \text{LHS} = &\int_{\Gamma_t} \bar{\mathbf{t}}(\mathbf{x}) \cdot \left[\sum_g \mathbf{w}_g S_g(\mathbf{x}) \right] d\Gamma - \sum_p \int_{\Omega_p} Y_p(\mathbf{x}) \boldsymbol{\sigma}_p : \left[\sum_g \mathbf{w}_g \otimes \nabla S_g \right] d\Omega \\ &+ \sum_p \int_{\Omega_p} Y_p(\mathbf{x}) \rho_p \left[\sum_g \mathbf{w}_g S_g(\mathbf{x}) \right] \cdot \mathbf{b}_p \, d\Omega \end{aligned} \tag{1.52}$$

Rearranging,

$$\begin{aligned} \text{LHS} = \sum_g \mathbf{w}_g \cdot \Bigg[&\int_{\Gamma_t} \bar{\mathbf{t}}(\mathbf{x}) S_g(\mathbf{x}) \, d\Gamma - \sum_p \int_{\Omega_p} Y_p(\mathbf{x}) \boldsymbol{\sigma}_p \cdot \nabla S_g \, d\Omega \\ &+ \sum_p \int_{\Omega_p} \rho_p Y_p(\mathbf{x}) S_g(\mathbf{x}) \mathbf{b}_p \, d\Omega \Bigg] \end{aligned} \tag{1.53}$$

Similarly, the right hand side of (1.50) can be written as

$$\text{RHS} = \sum_p \int_{\Omega_p} Y_p(\mathbf{x}) \rho_p \left[\sum_g \mathbf{w}_g S_g(\mathbf{x}) \right] \cdot \left[\sum_h \dot{\mathbf{v}}_h S_h(\mathbf{x}) \right] d\Omega \, . \tag{1.54}$$

Rearrangement leads to

$$\text{RHS} = \sum_g \mathbf{w}_g \cdot \sum_h \left[\sum_p \int_{\Omega_p} \rho_p Y_p(\mathbf{x}) S_g(\mathbf{x}) S_h(\mathbf{x}) \dot{\mathbf{v}}_h \, d\Omega \right]. \tag{1.55}$$

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

$$\begin{aligned} \int_{\Gamma_t} \bar{\mathbf{t}}(\mathbf{x}) S_g(\mathbf{x}) \, d\Gamma - \sum_p \int_{\Omega_p} Y_p(\mathbf{x}) \boldsymbol{\sigma}_p \cdot \nabla S_g \, d\Omega + \sum_p \int_{\Omega_p} \rho_p Y_p(\mathbf{x}) S_g(\mathbf{x}) \mathbf{b}_p \, d\Omega \\ = \sum_h \left[\sum_p \int_{\Omega_p} \rho_p Y_p(\mathbf{x}) S_g(\mathbf{x}) S_h(\mathbf{x}) \dot{\mathbf{v}}_h \, d\Omega \right]. \end{aligned} \tag{1.56}$$

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

$$\begin{aligned} \int_{\Gamma_t} \bar{\mathbf{t}}(\mathbf{x}) S_g(\mathbf{x}) \, d\Gamma - \sum_p \boldsymbol{\sigma}_p \cdot \left[\int_{\Omega_p} Y_p(\mathbf{x}) \nabla S_g \, d\Omega \right] + \sum_p \rho_p \mathbf{b}_p \left[\int_{\Omega_p} Y_p(\mathbf{x}) S_g(\mathbf{x}) \, d\Omega \right] \\ = \sum_h \sum_p \rho_p \left[\int_{\Omega_p} Y_p(\mathbf{x}) S_g(\mathbf{x}) S_h(\mathbf{x}) \, d\Omega \right] \dot{\mathbf{v}}_h . \end{aligned} \tag{1.57}$$

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

$$\langle S_{gp} \rangle := \frac{1}{V_p} \int_{\Omega_p} Y_p(\mathbf{x}) \, S_g(\mathbf{x}) \, d\Omega . \tag{1.58}$$

Then (1.57) can be expressed as

$$\begin{aligned} \int_{\Gamma_t} \bar{\mathbf{t}}(\mathbf{x}) S_g(\mathbf{x}) \, d\Gamma - \sum_p V_p \boldsymbol{\sigma}_p \cdot \langle \nabla S_{gp} \rangle + \sum_p V_p \rho_p \mathbf{b}_p \langle S_{gp} \rangle \\ = \sum_h \sum_p \rho_p \left[\int_{\Omega_p} Y_p(\mathbf{x}) S_g(\mathbf{x}) S_h(\mathbf{x}) \, d\Omega \right] \dot{\mathbf{v}}_h . \end{aligned} \tag{1.59}$$

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

$$\begin{aligned} M_{gh} &:= \sum_p \rho_p \int_{\Omega_p} Y_p(\mathbf{x}) \, S_g(\mathbf{x}) \, S_h(\mathbf{x}) \, d\Omega \\ \mathbf{f}_g^{\text{int}} &:= \sum_p V_p \boldsymbol{\sigma}_p \cdot \langle \nabla S_{gp} \rangle \\ \mathbf{f}_g^{\text{body}} &:= \sum_p m_p \mathbf{b}_p \, \langle S_{gp} \rangle \\ \mathbf{f}_g^{\text{ext}} &:= \int_{\Gamma_t} \bar{\mathbf{t}}(\mathbf{x}) \, S_g(\mathbf{x}) \, d\Gamma . \end{aligned} \tag{1.60}$$

Then, from (1.59) we get the semi-discrete system of equations

$$\sum_h M_{gh} \dot{\mathbf{v}}_h = \mathbf{f}_g^{\text{ext}} - \mathbf{f}_g^{\text{int}} + \mathbf{f}_g^{\text{body}} \; ; \quad g = 1 \ldots N_g \tag{1.61}$$

The mass matrix is typically lumped such that

$$\begin{aligned} m_g = \sum_h M_{gh} = \sum_p \rho_p \int_{\Omega_p} Y_p(\mathbf{x}) \, S_g(\mathbf{x}) \left[\sum_h S_h(\mathbf{x}) \right] d\Omega = \sum_p \rho_p \int_{\Omega_p} Y_p(\mathbf{x}) \, S_g(\mathbf{x}) \, d\Omega \\ = \sum_p \rho_p V_p \langle S_{gp} \rangle = \sum_p m_p \langle S_{gp} \rangle . \end{aligned} \tag{1.62}$$

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

$$m_g \dot{\mathbf{v}}_g = \mathbf{f}_g^{\text{ext}} - \mathbf{f}_g^{\text{int}} + \mathbf{f}_g^{\text{body}} \; ; \quad g = 1 \ldots N_g \tag{1.63}$$

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

$$\mathbf{f}_g^{\text{ext}} = \sum_p \mathbf{f}_p^{\text{ext}} \left\langle S_{gp} \right\rangle \, . \tag{1.64}$$

1.4.1 Damping

Two types of artificial damping are implemented in Vaango . The first approach modifies the acceleration in (1.63) such that

$$m_g \dot{\mathbf{v}}_g = \mathbf{f}_g^{\text{ext}} - \mathbf{f}_g^{\text{int}} + \mathbf{f}_g^{\text{body}} - \alpha_d \mathbf{v}_g \; ; \quad g = 1 \ldots N_g \tag{1.65}$$

where α_d is a damping coefficient.

The second approach uses Richtmyer-von Neumann artificial viscosity to damp out large oscillations in high strain-rate simulations. Vaango uses a three-dimensional form of the Richtmyer-von Neumann artificial viscosity ([13, 14], p.29). The viscosity factor takes the form

$$q = C_0 \, \rho \, l \sqrt{\frac{K}{\rho}} \, |\text{tr}\boldsymbol{d}| + C_1 \, \rho \, l^2 \, (\text{tr}\boldsymbol{d})^2 \tag{1.66}$$

where C_0 and C_1 are constants, ρ is the mass density, K is the bulk modulus, $\boldsymbol{d}$ is the rate of deformation tensor, and l is a characteristic length (usually the grid cell size). Typical values of the coefficients are $C_0 = 0.2$ and $C_1 = 2.0$.

The factor q is used to decrease the particle stress:

$$\boldsymbol{\sigma}_p = \boldsymbol{\sigma}_p - q\boldsymbol{I} \tag{1.67}$$

before it is projected to grid nodes for internal force calculations.

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 25.) 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 [15]. USF was the formulation used in Uintah until mid-2008.

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

$$\mathbf{Ma} = \mathbf{f}^{\text{ext}} - \mathbf{f}^{\text{int}} + \mathbf{f}^{\text{body}} \tag{1.68}$$

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. (1.68) for acceleration. These quantities are calculated at individual nodes by the following equations, where the $\sum_p$ represents a summation over all particles:

$$m_g = \sum_p m_p \bar{S}_{gp} \,, \quad \mathbf{v}_g = \frac{\sum_p m_p \mathbf{v}_p \bar{S}_{gp}}{m_g} \,, \quad \mathbf{f}_g^{\text{ext}} = \sum_p \mathbf{f}_p^{\text{ext}} \bar{S}_{gp} \tag{1.69}$$

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. 1.69, $\mathbf{f}^{\text{int}}$ 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:

$$\mathbf{f}_g^{\text{int}} = \sum_p V_p \boldsymbol{\sigma}_p \bar{\mathbf{G}}_{gp} \tag{1.70}$$

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

$$\mathbf{a}_g = \frac{\mathbf{f}_g^{\text{ext}} - \mathbf{f}_g^{\text{int}} + \mathbf{f}_g^{\text{body}}}{m_g} \tag{1.71}$$

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

$$\mathbf{v}_g^L = \mathbf{v}_g + \mathbf{a}_g \Delta t \quad \text{where} \quad \Delta t = t_{n+1} - t_n \,. \tag{1.72}$$

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

$$\nabla \mathbf{v}_p = \sum_g \mathbf{v}_g^L \bar{\mathbf{G}}_{gp} \,. \tag{1.73}$$

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:

$$\Delta \boldsymbol{F}_p^{n+1} = (\boldsymbol{I} + \nabla \mathbf{v}_p \Delta t) \tag{1.74}$$

Particle volume and deformation gradient are updated by:

$$V_p^{n+1} = \det(\Delta \boldsymbol{F}_p^{n+1}) V_p^n \,, \quad \boldsymbol{F}_p^{n+1} = \Delta \boldsymbol{F}_p^{n+1} \cdot \boldsymbol{F}_p^n \,. \tag{1.75}$$

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:

$$\begin{aligned} \mathbf{v}_p(t + \Delta t) &= \mathbf{v}_p(t) + \sum_g \bar{S}_{gp} \mathbf{a}_g \Delta t \\ \mathbf{x}_p(t + \Delta t) &= \mathbf{x}_p(t) + \sum_g \bar{S}_{gp} \mathbf{v}_g^L \Delta t \end{aligned} \tag{1.76}$$

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 1.76, 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.[16]. 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.

1.5.1 Deformation gradient computation

The deformation gradient computation involves the solution of a first-order differential equation:

$$\dot{\boldsymbol{F}} = \boldsymbol{l} \cdot \boldsymbol{F} = \nabla \mathbf{v} \cdot \boldsymbol{F} \tag{1.77}$$

which, for constant $\boldsymbol{l}$ and initial condition $\boldsymbol{F} = \boldsymbol{F}_0$, has the exact solution

$$\boldsymbol{F}(t) = \exp(t\, \boldsymbol{l}) \cdot \boldsymbol{F}_0 = \exp(t\, \nabla \mathbf{v}) \cdot \boldsymbol{F}_0 \,. \tag{1.78}$$

Expanded in series form, and considering only the time step Δt with initial condition $\boldsymbol{F} = \boldsymbol{F}_p^n$, we have

$$\boldsymbol{F}_p(t) = \left[\boldsymbol{I} + \Delta t \nabla \mathbf{v}_p + \frac{1}{2!}(\Delta t \nabla \mathbf{v}_p)^2 + \frac{1}{3!}(\Delta t \nabla \mathbf{v}_p)^3 + \dots\right] \cdot \boldsymbol{F}_p^n \,. \tag{1.79}$$

The approach in (1.74) is a first-order approximation of the Taylor series expansion for the deformation gradient:

$$\boldsymbol{F}_p^{n+1} = (\boldsymbol{I} + \nabla \mathbf{v}_p \Delta t)\boldsymbol{F}_p^n \,. \tag{1.80}$$

This is the most commonly used method of computing the deformation gradient. VAANGO also allows for an alternative estimate of the deformation gradient by subcycling after dividing Δt into k smaller increments:

$$\boldsymbol{F}_p^{n+1} = \left[\prod_k (\boldsymbol{I} + \nabla \mathbf{v}_p \Delta t_k)\right] \boldsymbol{F}_p^n \,. \tag{1.81}$$

Alternatively, multiple terms of the expansion in (1.79) can be evaluated by choosing an appropriate flag in the input file.

Finally, VAANGO also provides an option to compute the matrix exponential using the Cayley-Hamilton theorem . However, all these approaches assume that the velocity gradient remains constant over a time step.

Pressure stabilization

A pressure stabilization step may be required during the computation of deformation gradients of materials that are nearly incompressible. The algorithm involves computing a the particle volumes inside each grid cell (ignoring volume that may extend outside cell boundaries):

$$V_{co} = \sum_{p \in c} \frac{m_p}{\rho_o} \,, \quad V_c = \sum_{p \in c} V_p \tag{1.82}$$

where V_{co} is an estimate of the initial volume in a cell and V_c is the current volume in the cell. The initial density is ρ_o. An estimate of the volume change is computed using

$$J_c = \frac{V_c}{V_{co}} \,. \tag{1.83}$$

A correction is applied to the particle deformation gradient using

$$\boldsymbol{F}_p \leftarrow \left(\frac{J_c}{\det(\boldsymbol{F}_p)}\right)^{1/3} \boldsymbol{F}_p \,. \tag{1.84}$$

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:

$$\bar{S}_{gp}(\mathbf{x}_p) = \frac{1}{V_p} \int_{\Omega_p \cap \Omega} \chi_p(\mathbf{x} - \mathbf{x}_p) S_g(\mathbf{x}) \, \mathrm{d}\mathbf{x}. \tag{1.85}$$

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,

$$S_g(x) = \begin{cases} 1 + (x - x_g)/h & -h < x - x_g \leq 0 \\ 1 - (x - x_g)/h & 0 < x - x_g \leq h \\ 0 & \text{otherwise,} \end{cases} \tag{1.86}$$

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,

$$S_g^{\alpha}(r, s, t) = \frac{1}{8}(1 + r\, r_\alpha)(1 + s\, s_\alpha)(1 + t\, t_\alpha) \tag{1.87}$$

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.

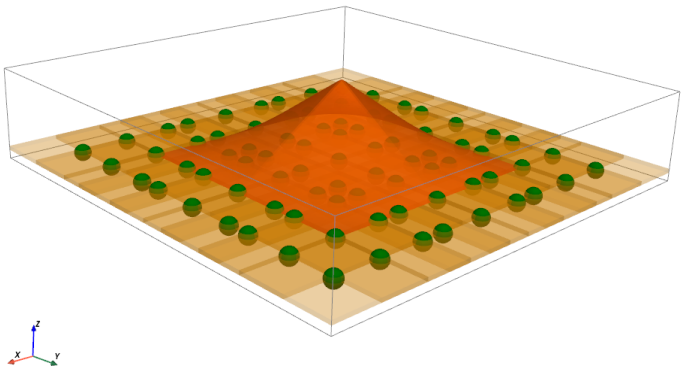


Figure 1.1: Linear grid node shape functions for 2D traditional MPM .

1.6.1 MPM

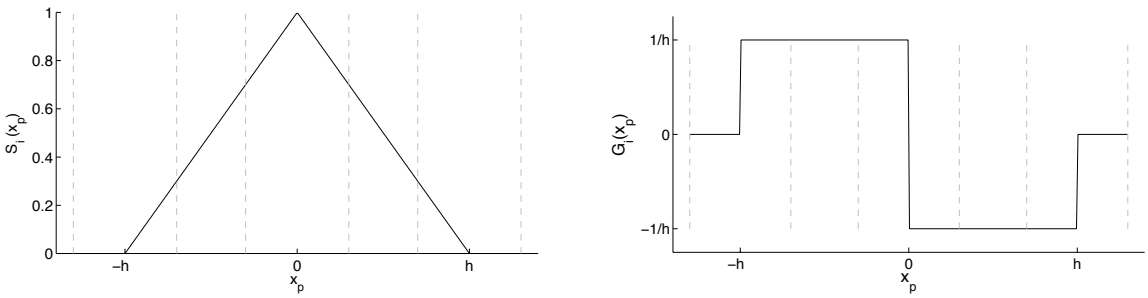
When the choice of characteristic function is the Dirac delta,

$$\chi_p(\mathbf{x}) = \delta(\mathbf{x} - \mathbf{x}_p) V_p, \tag{1.88}$$

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 (1.86). Its gradient is given by:

$$G_g(x) = \begin{cases} 1/h & -h < x - x_g \leq 0 \\ -1/h & 0 < x - x_g \leq h \\ 0 & \text{otherwise,} \end{cases} \tag{1.89}$$

Plots of Equations 1.86 and 1.89 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. 1.86 and a "top-hat" characteristic function, given by (in one-dimension),

$$\chi_p(x) = H(x - (x_p - l_p)) - H(x - (x_p + l_p)), \tag{1.90}$$

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. 1.85 is carried out using the expressions in Eqns. 1.86 and 1.90, a closed form for the effective shape function can be written as:

$$\bar{S}_{gp}\left(x_p\right) = \begin{cases} \frac{\left(h+l_p+\left(x_p-x_g\right)\right)^2}{4hl_p} & -h - l_p < x_p - x_g \leq -h + l_p \\ 1 + \frac{\left(x_p-x_g\right)}{h} & -h + l_p < x_p - x_g \leq -l_p \\ 1 - \frac{\left(x_p-x_g\right)^2+l_p^2}{2hl_p} & -l_p < x_p - x_g \leq l_p \\ 1 - \frac{\left(x_p-x_g\right)}{h} & l_p < x_p - x_g \leq h - l_p \\ \frac{\left(h+l_p-\left(x_p-x_g\right)\right)^2}{4hl_p} & h - l_p < x_p - x_g \leq h + l_p \\ 0 & \text{otherwise,} \end{cases} \tag{1.91}$$

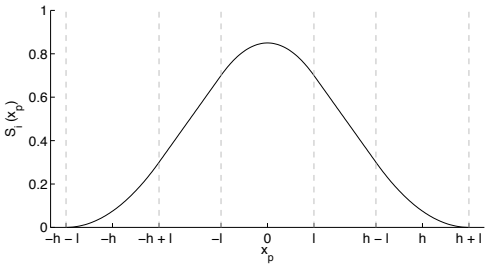
The gradient of which is:

$$\bar{G}_{gp}(x_p) = \begin{cases} \frac{h+l_p+\left(x_p-x_g\right)}{2hl_p} & -h - l_p < x_p - x_g \leq -h + l_p \\ \frac{1}{h} & -h + l_p < x_p - x_g \leq -l_p \\ -\frac{\left(x_p-x_g\right)}{hl_p} & -l_p < x_p - x_g \leq l_p \\ -\frac{1}{h} & l_p < x_p - x_g \leq h - l_p \\ -\frac{h+l_p-\left(x_p-x_g\right)}{2hl_p} & h - l_p < x_p - x_g \leq h + l_p \\ 0 & \text{otherwise,} \end{cases} \tag{1.92}$$

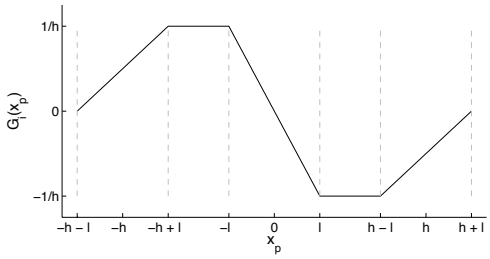
Plots of Equations 1.91 and 1.92 are shown in Figure 1.3. The continuous nature of the gradients are largely responsible for the improved robustness and accuracy of GIMP over MPM .

1.6.3 UGIMP and cpGIMP

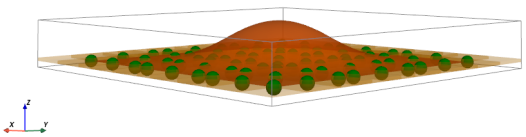
The GIMP effective shape functions in (1.91) are valid only for particle sizes that are smaller than the grid spacing. In Figure 1.4 we see that discontinuities appear in the effective shape function for particles for which $l_p > 0.5h$.



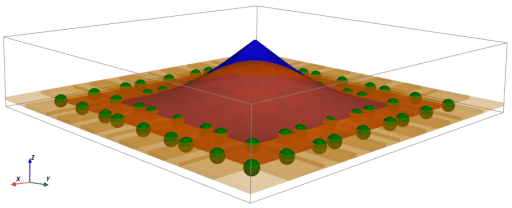
(a) One-dimensional shape function.



(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.

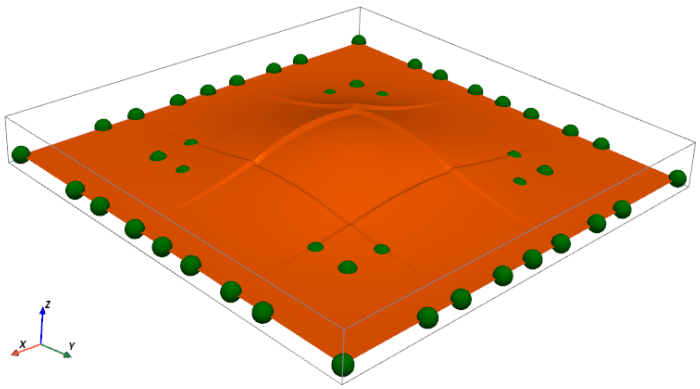


Figure 1.4: Two-dimensional GIMP effective shape functions ($\bar{S}_{gp}$) for $l_p = 0.7h$.

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,

$$l_p^n = \boldsymbol{F}_p^n l_p^o, \tag{1.93}$$

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

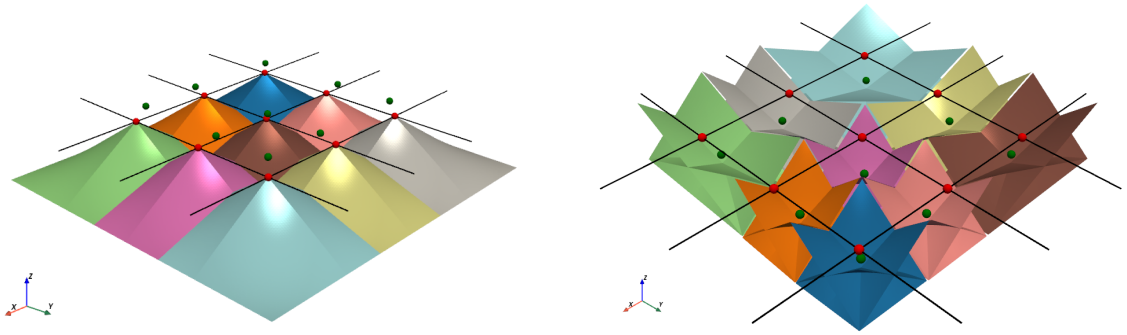
In multi-dimensional simulations, a similar approach can be used, assuming an initially rectangular or cuboid particle, to find the current particle shape. The difficulty arises in evaluating Eq. (1.85) for these general shapes. One approach, apparently effective, has been to create a cuboid that circumscribes the deformed particle shape [17]. 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

$$\langle S_{gp} \rangle = \frac{1}{V_p} \int_{\Omega_p} Y_p(\mathbf{x})\, \tilde{S}_g(\mathbf{x})\, d\Omega \quad \text{and} \quad \langle \nabla S_{gp} \rangle = \frac{1}{V_p} \int_{\Omega_p} Y_p(\mathbf{x})\, \nabla \tilde{S}_g(\mathbf{x})\, d\Omega \tag{1.94}$$

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 ($S_g(\mathbf{x})$). (b) CPDI particle basis functions ($N_p(\mathbf{x})$).

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

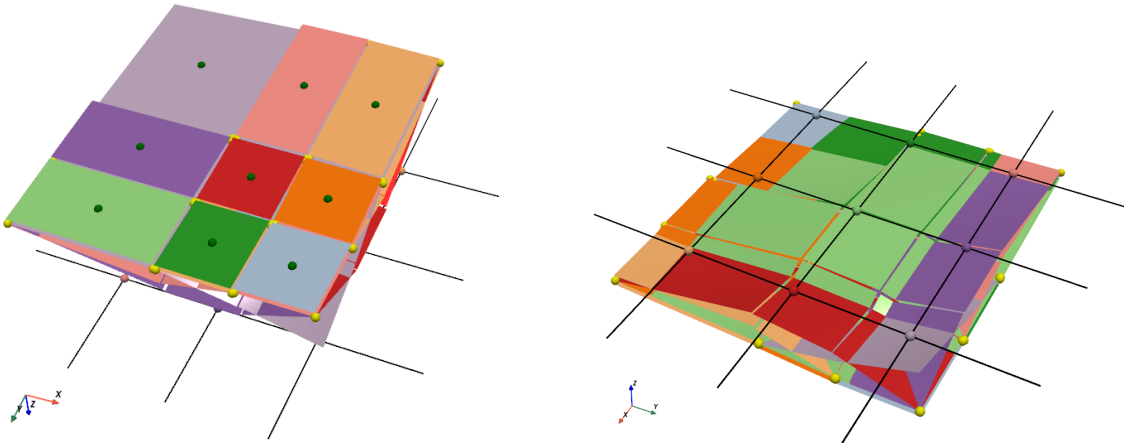
In the reference state, the domain Ω_{po} for particle p is assumed to be a parallelepiped spanned by the three vectors $\boldsymbol{r}_p^{io}$, $i = 1, 2, 3$ with origin at the centroid. In the deformed state, these vectors become $\boldsymbol{r}_p^i = \boldsymbol{F}_p \cdot \boldsymbol{r}_p^{io}$ where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

$$\tilde{S}_g(\mathbf{x}) = \sum_{\alpha=1}^{8} N_p^\alpha(\mathbf{x}) S_g(\mathbf{x}_p^\alpha) \quad \text{on } \Omega_p \tag{1.95}$$

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

$$N_p^\alpha(r,s,t) = \frac{1}{8}(1 + r\, r_\alpha)(1 + s\, s_\alpha)(1 + t\, t_\alpha) \tag{1.96}$$

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. Figure 1.6 shows the particle domains and the effective grid shape functions produced by the CPDI relation in (1.95).



(a) Particle domains Ω_p.

(b) Effective grid basis functions ($\tilde{S}_g(\mathbf{x})$).

Figure 1.6: Two-dimensional CPDI particle domains and effective grid basis functions.

We can compute the quantities in (1.94) as follows.

Let

$$\mathbf{s}_p^0 = \begin{bmatrix} \boldsymbol{r}_p^{10} & \boldsymbol{r}_p^{20} & \boldsymbol{r}_p^{30} \end{bmatrix} \quad \text{and} \quad \mathbf{s}_p = \begin{bmatrix} \boldsymbol{r}_p^{1} & \boldsymbol{r}_p^{2} & \boldsymbol{r}_p^{3} \end{bmatrix} \quad \text{and} \quad \mathbf{F}_p = \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix}. \tag{1.97}$$

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

$$\mathbf{S}_{gp} = \begin{bmatrix} S_g(\mathbf{x}_p^1) & S_g(\mathbf{x}_p^2) & S_g(\mathbf{x}_p^3) & S_g(\mathbf{x}_p^4) & S_g(\mathbf{x}_p^5) & S_g(\mathbf{x}_p^6) & S_g(\mathbf{x}_p^7) & S_g(\mathbf{x}_p^8) \end{bmatrix}. \tag{1.98}$$

Also, let

$$\mathbf{R} = \begin{bmatrix} -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{1.99}$$

Then,

$$\bar{S}_{gp} = \langle S_{gp} \rangle = \text{mean}(\mathbf{S}_{gp}) \quad \text{and} \quad \bar{\boldsymbol{G}}_{gp} = \langle \nabla S_{gp} \rangle = \frac{1}{8} \mathbf{s}_p^{-T} \mathbf{R}^T \mathbf{S}_{gp}^T. \tag{1.100}$$

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. Many of these algorithms require the computaion of surface normals.

1.7.1 Definitions

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. Consider N_α objects

that can potentially be in contact and index then by the superscript α. Then, from (1.43), we have

$$m_g^\alpha = V_g^\alpha \rho_g^\alpha = \sum_p V_p^\alpha \rho_p^\alpha \bar{S}_{gp} = \sum_p m_p^\alpha \bar{S}_{gp} \quad \text{and} \quad \mathbf{p}_g^\alpha = \sum_p \mathbf{p}_p^\alpha \bar{S}_{gp} \,. \tag{1.101}$$

In matrix notation,

$$\boldsymbol{m}_g^\alpha = \mathbb{S}^T \boldsymbol{m}_p^\alpha \quad \text{and} \quad \boldsymbol{p}_g^\alpha = \mathbb{S}^T \boldsymbol{p}_p^\alpha \,. \tag{1.102}$$

Similarly, from (1.42), we have

$$m_g^\alpha \mathbf{v}_g^\alpha = \sum_p m_p^\alpha \mathbf{v}_p^\alpha \bar{S}_{gp} \quad \Longrightarrow \quad \mathbf{v}_g^\alpha = \frac{1}{m_g^\alpha} \sum_p m_p^\alpha \mathbf{v}_p^\alpha \bar{S}_{gp} \,. \tag{1.103}$$

In matrix form,

$$\boldsymbol{v}_g^\alpha = \mathbb{S}_\alpha^+ \boldsymbol{v}_p^\alpha \quad \text{where} \quad \mathbb{S}_\alpha^+ = \left(\boldsymbol{m}_g^\alpha\right)^{-1} \mathbb{S}^T \boldsymbol{m}_p^\alpha \,. \tag{1.104}$$

Based on a local conservation of momentum, we define a center-of-mass velocity, $\mathbf{v}_g^{\text{cm}}$, at grid node g for all the contacting objects:

$$\mathbf{v}_g^{\text{cm}} = \frac{\sum_\alpha m_g^\alpha \mathbf{v}_g^\alpha}{\sum_\alpha m_g^\alpha} \,. \tag{1.105}$$

We also define an effective grid mass, m_g^{eff}, as

$$\frac{1}{m_g^{\text{eff}}} = \sum_\alpha \frac{1}{m_g^\alpha} \,. \tag{1.106}$$

1.7.2 Computing surface normals and tractions

Surface normals are typically estimated from the gradient of mass at grid nodes. If m_p is the mass of particle p, then the normal at grid node g due to the particles in its region of influence is

$$\mathbf{n}_g = \sum_p m_p \nabla \bar{S}_{gp} \,. \tag{1.107}$$

Normals are converted to unit vectors before they ar used in Vaango computations.

Surface tractions at the nodes are computed by projecting particle stresses ($\boldsymbol{\sigma}_p$) to grid nodes:

$$\bar{\mathbf{t}}_g = \mathbf{n}_g \cdot \left(\sum_p \boldsymbol{\sigma}_p \bar{S}_{gp} \right) \,. \tag{1.108}$$

1.7.3 Basic contact algorithm

The most basic contact algroithm in Vaango is called "single-velocity contact" The center-of-mass velocity is computed using (1.105). Upon contact detection, grid nodes that participate are assigned this velocity:

$$\mathbf{v}_g = \mathbf{v}_g^{\text{cm}} \,. \tag{1.109}$$

1.7.4 Contact with a specified master

A slightly more complex algorithm is the "master"-based contact which is called "specified-velocity contact" in Vaango . In this model, a selected master material is assigned velocities, $\mathbf{v}_g^m = \mathbf{v}^m(t)$, where m is the index of the master material. The grid node velocities of the materials are then adjusted according to

$$\mathbf{v}_g \leftarrow \mathbf{v}_g - \left[\mathbf{n}_g^m \cdot (\mathbf{v}_g - \mathbf{v}_g^m) \right] \mathbf{n}_g^m \tag{1.110}$$

where $\mathbf{n}_g^m$ is the normal for the master material computed using (1.107). This type of contact is useful for imposing boundary conditions on objects.

1.7.5 Frictional contact algorithms

The two main frictional contact algorithms are friction_bard, which is based on [16], and friction_LR, which is described in [12].

Bardenhagen et al. algorithm

In the algorithm developed in [16], a contact interface is defined as the set of nodes for which individual grid velocities associated with each object differ from the center of mass velocity:

$$\mathbf{v}_g^\alpha - \mathbf{v}_g^{\text{cm}} \neq 0\,. \tag{1.111}$$

Once this condition is identified, the surface normal $\mathbf{n}_g^\alpha$ is computed from the mass distribution around node g, and the surface normal traction $\mathbf{t}_g^\alpha$ is computed from the stresses in surrounding material points.

The contact condition is

$$(\mathbf{v}_g^\alpha - \mathbf{v}_g^{\text{cm}}) \cdot \mathbf{n}_g^\alpha > 0 \quad \text{and} \quad \mathbf{t}_g^\alpha \cdot \mathbf{n}_g^\alpha < 0\,. \tag{1.112}$$

This condition indicates compressive stress at node g. If this condition is not satisfied, the objects are assumed to have separated.

To enforce (1.112), the grid node velocities are adjusted such that momentum is conserved, i.e.,

$$\Delta(v_n)_g^\alpha = \Delta\mathbf{v}_g^\alpha \cdot \mathbf{n}_g^\alpha \quad \text{and} \quad \Delta(v_t)_g^\alpha = \Delta\mathbf{v}_g^\alpha \cdot \left[\mathbf{n}_g^\alpha \times \frac{\Delta\mathbf{v}_g^\alpha \times \mathbf{n}_g^\alpha}{\left\|\Delta\mathbf{v}_g^\alpha \times \mathbf{n}_g^\alpha\right\|}\right] \tag{1.113}$$

where

$$\Delta\mathbf{v}_g^\alpha := \mathbf{v}_g^\alpha - \mathbf{v}_g^{\text{cm}}\,. \tag{1.114}$$

Normal contact is enforced by adjusting material velocities by $\Delta(v_n)_g^\alpha$. The tangential contact is enforced using Coulomb friction with the tangential velocity determined using $\mu\Delta(v_n)_g^\alpha$ where μ is the friction coefficient. If $\Delta(v_t)_g^\alpha < \mu\Delta(v_n)_g^\alpha$, the no-slip condition is enforced. Otherwise, the tangential components of the nodal velocities are updated with a reduced friction coefficient

$$\mu_{\text{red}} = \min\left(\mu, \frac{|\Delta(v_t)_g^\alpha|}{|\Delta(v_n)_g^\alpha|}\right)\,. \tag{1.115}$$

Returning to the problem of computing object outward normals at a grid point, the traditional approach is to compute volume gradients using the set of particles influencing a node:

$$\mathbf{g}_g^\alpha = \sum_{p^\alpha} \overline{\mathbf{G}}_{gp} V_p \tag{1.116}$$

where the gradients $\overline{\mathbf{G}}_{gp}$ are as defined in (1.29), (1.31), and (1.33). The normal to an object is calculated using

$$\mathbf{n}^\alpha = \frac{\mathbf{g}_g^\alpha}{\left\|\mathbf{g}_g^\alpha\right\|}\,. \tag{1.117}$$

For multiple objects, an average gradient can be computed for better accuracy.

Nairn et al. algorithm

The more recent algorithm by [12] uses a logistic regression step to determine contact. The underlying approach is similar to that used in [16]. Since the approach is at its simplest when only two objects are involved at a grid point, we will describe only that case below. Most situations with contact between multiple objects see [12].

Let the two objects be indexed by α and β. Let $\mathbf{p}_g^{\alpha o}$ and $\mathbf{p}_g^{\beta o}$ be the particle momenta projected to the grid. We would like to compute the momentum correction $\Delta\mathbf{p}$ so that momentum is conserved after contact. Let the corrected momenta be

$$\mathbf{p}_g^{\alpha} = \mathbf{p}_g^{\alpha o} + \Delta\mathbf{p} \quad \text{and} \quad \mathbf{p}_g^{\beta} = \mathbf{p}_g^{\beta o} - \Delta\mathbf{p}\,. \tag{1.118}$$

If we restrict relative motion between objects at a grid point to the tangent plane, and let $\hat{\mathbf{t}}$ be the direction of relative motion, then

$$\mathbf{v}_g^{\beta} - \mathbf{v}_g^{\alpha} = k\hat{\mathbf{t}} \quad \Longrightarrow \quad \frac{\mathbf{p}_g^{\beta}}{m_g^{\beta}} - \frac{\mathbf{p}_g^{\alpha}}{m_g^{\alpha}} = k\hat{\mathbf{t}} \quad \Longrightarrow \quad m_g^{\alpha}\mathbf{p}_g^{\beta} - m_g^{\beta}\mathbf{p}_g^{\alpha} = m_g^{\alpha}m_g^{\beta}k\hat{\mathbf{t}}\,. \tag{1.119}$$

From the definition of the center-of-mass velocity in (1.105) and the effective grid mass (1.106), we have

$$\mathbf{v}_g^{\mathrm{cm}} = \frac{m_g^{\alpha}\mathbf{v}_g^{\alpha} + m_g^{\beta}\mathbf{v}_g^{\beta}}{m_g^{\alpha} + m_g^{\beta}} = \frac{\mathbf{p}_g^{\alpha} + \mathbf{p}_g^{\beta}}{m_g^{\alpha} + m_g^{\beta}} \quad \text{and} \quad m_g^{\mathrm{eff}} = \frac{m_g^{\alpha}m_g^{\beta}}{m_g^{\alpha} + m_g^{\beta}}\,. \tag{1.120}$$

Therefore,

$$\mathbf{p}_g^{\alpha} + \mathbf{p}_g^{\beta} = \frac{\mathbf{v}_g^{\mathrm{cm}} m_g^{\alpha} m_g^{\beta}}{m_g^{\mathrm{eff}}}\,. \tag{1.121}$$

Solving for $\mathbf{p}_g^{\alpha}, \mathbf{p}_g^{\beta}$ from equations (1.119) and (1.121), we have

$$\mathbf{p}_g^{\alpha} = m_g^{\alpha}\mathbf{v}_g^{\mathrm{cm}} - m_g^{\mathrm{eff}}k\hat{\mathbf{t}} \quad \text{and} \quad \mathbf{p}_g^{\beta} = m_g^{\beta}\mathbf{v}_g^{\mathrm{cm}} + m_g^{\mathrm{eff}}k\hat{\mathbf{t}}\,. \tag{1.122}$$

Therefore, from (1.118),

$$\Delta\mathbf{p} = \mathbf{p}_g^{\alpha} - \mathbf{p}_g^{\alpha o} = m_g^{\alpha}\mathbf{v}_g^{\mathrm{cm}} - m_g^{\mathrm{eff}}k\hat{\mathbf{t}} - m_g^{\alpha}\mathbf{v}_g^{\alpha o} = m_g^{\alpha}(\mathbf{v}_g^{\mathrm{cm}} - \mathbf{v}_g^{\alpha o}) - m_g^{\mathrm{eff}}k\hat{\mathbf{t}}\,. \tag{1.123}$$

The quantity

$$\Delta\mathbf{p}^{o} := m_g^{\alpha}(\mathbf{v}_g^{\mathrm{cm}} - \mathbf{v}_g^{\alpha o}) \tag{1.124}$$

is the initial change of momentum before tangential correction. Since the contact force ($\mathbf{f}^c$) is given by the rate of change of momentum due to contact, we have

$$\mathbf{f}^{co} = \frac{\Delta\mathbf{p}^{o}}{\Delta t} \quad \text{and} \quad \mathbf{f}^{c} = \frac{\Delta\mathbf{p}}{\Delta t} = \mathbf{f}^{co} - \frac{m_g^{\mathrm{eff}}k\hat{\mathbf{t}}}{\Delta t}\,. \tag{1.125}$$

where Δt is the timestep size. The contact compressive traction is found from the normal component of the contact force needed to prevent interpenetration:

$$T_n^c = -\frac{1}{A^c}\,\mathbf{f}^{co}\cdot\mathbf{n} \tag{1.126}$$

where A^c is the contact area. The tangential contact traction can be found using a contact law:

$$T_t^c = \frac{1}{A^c}\,\mathbf{f}^{c}\cdot\hat{\mathbf{t}} = \frac{1}{A^c}\left[\mathbf{f}^{co}\cdot\hat{\mathbf{t}} - \frac{m_g^{\mathrm{eff}}k}{\Delta t}\right]. \tag{1.127}$$

Therefore, if $T_t^c = T_t^c(T_n^c)$ is a contact law,

$$k = \frac{\Delta t}{m_g^{\text{eff}}}\left[\mathbf{f}^{\text{co}} \cdot \hat{\mathbf{t}} - A^c T_t^c(T_n^c)\right]. \tag{1.128}$$

We can compute k using the contact law $T_t^c(T_n^c)$ and then adjust $\mathbf{v}_g^\alpha$ and $\mathbf{v}_g^\beta$ using (1.118). Note that this process is identical to that used in the Bardenhagen at al. algorithm. The main difficulty is in finding where contact has occurred and the quantities $\mathbf{n}$, $\hat{\mathbf{t}}$, and A^c.

The basic contact identification condition used in this approach, and in the Bardenhagen et al. approach, is

$$(\mathbf{v}_g^\beta - \mathbf{v}_g^\alpha) \cdot \frac{\mathbf{g}_g^\alpha - \mathbf{g}_g^\beta}{\left\|\mathbf{g}_g^\alpha - \mathbf{g}_g^\beta\right\|} < 0 \quad \text{and} \quad T_n^c > 0 \tag{1.129}$$

where $\mathbf{g}_g$ is the volume gradient defined in (1.116). This condition is a variation of (1.112) and is a necessary, but not sufficient, condition to detect whether the two objects are approaching each other and in contact. However, $T_n^c > 0$ even when the objects are not touching and a separation condition is needed to correctly identify contact.

The logistic regression approach developed in [12] attempts to identify contact without having to rely purely on grid information. This approach requires a set of particles (point-cloud) in the neighborhood of a grid point that satisfies (1.129). The aim of this technique is to identify a plane within the point-cloud that best separates the two objects. The normal to this plane is the contact normal $\mathbf{n}_g$. The logistic function penalizes points as a function of their distance from a preferred separation plane.

The logistic regression method for separation detection is described next. Let $\boldsymbol{x}_p$ be the homogeneous coordinate representation of a particle position, i.e., $\mathbf{X}_p = (\mathbf{x}_p, 1) =: (X_1, X_2, X_3, X_4)$, where $\mathbf{x}_p = (x_p^1, x_p^2, x_p^3)$ is the particle position. Let $\mathbf{N}$ be the corresponding normal vector of the separation plane, i.e., $\mathbf{N} = (\mathbf{n}, N_4)$ where $\mathbf{n} = (n_1, n_2, n_3) =: (N_1, N_2, N_3)$ is the normal to the separation plane and n_4 is an offset. The equation of the desired separation plane is

$$\mathbf{X} \cdot \mathbf{N} = 0 \tag{1.130}$$

where $\mathbf{X}$ is the vector of particle positions. Let there be P particles in the point-cloud consisting of points from objects α and β. Define a particle label c_p as:

$$c_p = \begin{cases} -1 & \text{for particles in object } \alpha \\ 1 & \text{for particles in object } \beta. \end{cases} \tag{1.131}$$

The objective function that has to be minimized is the error

$$E = \sum_{p=1}^{P} w_p \left[\mathcal{L}(\mathbf{X}_p, \mathbf{N}) - c_p\right]^2 + \sum_{j=1}^{4} \lambda_j^2 N_j^2 \tag{1.132}$$

where w_p are weights, λ_j^2 are penalty factors that help regularize the error function and $\mathcal{L}$ is the logistic function given by

$$\mathcal{L}(\mathbf{X}, \mathbf{N}) = \frac{2}{1 + \exp(-\mathbf{X} \cdot \mathbf{N})} - 1. \tag{1.133}$$

The minimum of E is achieved when $\frac{\partial E}{\partial \mathbf{N}} = \mathbf{0}$ and $\frac{\partial E}{\partial \lambda} = 0$. From the first requirement

$$\frac{\partial E}{\partial N_i} = 2 \sum_{p=1}^{P} w_p \left[\mathcal{L}(\mathbf{X}_p, \mathbf{N}) - c_p\right] \frac{\partial \mathcal{L}}{\partial N_i} + 2 \sum_{j=1}^{4} \lambda_j^2 N_j \frac{\partial N_j}{\partial N_i} = 0 \tag{1.134}$$

where

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial N_i} &= 2\frac{\partial}{\partial N_i}\left[1+\exp(-X_m N_m)\right]^{-1} = -2\left[1+\exp(-X_m N_m)\right]^{-2}\frac{\partial}{\partial N_i}\exp(-X_m N_m)\\ &= 2\left[1+\exp(-X_m N_m)\right]^{-2}\exp(-X_m N_m)X_m\frac{\partial N_m}{\partial N_i}\\ &= 2\left[1+\exp(-X_m N_m)\right]^{-2}\exp(-X_m N_m)X_i\,.\end{aligned} \tag{1.135}$$

Define

$$\theta_p := -\mathbf{X}_p\cdot\mathbf{N} \quad \text{and} \quad \phi_p := 1+\exp(\theta_p)\,. \tag{1.136}$$

Then, in vector form,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{N}} = \frac{2\exp(\theta_p)}{\phi_p^2}\mathbf{X} \quad \text{and} \quad \mathcal{L}(\mathbf{X}_p,\mathbf{N}) = \frac{2}{\phi_p}-1\,. \tag{1.137}$$

Returning to (1.134), we can write

$$\frac{\partial E}{\partial N_i} = 2\sum_{p=1}^{P} w_p\left[\mathcal{L}(\mathbf{X}_p,\mathbf{N})-c_p\right]\frac{\partial \mathcal{L}}{\partial N_i}+2\lambda_i^2 N_i = 0\,. \tag{1.138}$$

Similarly,

$$\frac{\partial E}{\partial \lambda_i} = 2\sum_{j=1}^{4}\lambda_j\frac{\partial \lambda_j}{\partial \lambda_i}N_j^2 = \lambda_i N_i^2 = 0 \tag{1.139}$$

Then, in vector form, the system of equations needed to solve for $\mathbf{N}$ and $\boldsymbol{\lambda}$ is

$$\boxed{\begin{aligned}&\sum_{p=1}^{P} w_p\left[\mathcal{L}(\mathbf{X}_p,\mathbf{N})-c_p\right]\frac{\partial \mathcal{L}}{\partial \mathbf{N}}+(\boldsymbol{\lambda}\odot\boldsymbol{\lambda})\odot\mathbf{N}=0 \quad \text{and}\\ &\boldsymbol{\lambda}\odot(\mathbf{N}\odot\mathbf{N})=0\end{aligned}} \tag{1.140}$$

where

$$\mathbf{N}=(N_1,N_2,N_3,N_4)\,,\ \boldsymbol{\lambda}=(\lambda_1,\lambda_2,\lambda_3,\lambda_4)\,,\ \mathbf{a}\odot\mathbf{b}=(a_1b_1,a_2b_2,a_3b_3,a_4b_4)\,. \tag{1.141}$$

The second set of equations in (1.140) suggest that the solution will improve as $\boldsymbol{\lambda}\to 0$. Given a vector $\boldsymbol{\lambda}$, the first equation in (1.140) can be solved for $\mathbf{N}$ using Newton's method. Define

$$\mathbf{Y}(\mathbf{N}) := \sum_{p=1}^{P} w_p\left[\mathcal{L}(\mathbf{X}_p,\mathbf{N})-c_p\right]\frac{\partial \mathcal{L}}{\partial \mathbf{N}}+(\boldsymbol{\lambda}\odot\boldsymbol{\lambda})\odot\mathbf{N}\,. \tag{1.142}$$

Then, with $\mathbf{I}$ denoting the 4×4 identity matrix,

$$\frac{\partial \mathbf{Y}}{\partial \mathbf{N}} = \sum_{p=1}^{P} w_p\left[\frac{\partial \mathcal{L}}{\partial \mathbf{N}}\otimes\frac{\partial \mathcal{L}}{\partial \mathbf{N}}+\left[\mathcal{L}(\mathbf{X}_p,\mathbf{N})-c_p\right]\frac{\partial^2 \mathcal{L}}{\partial \mathbf{N}^2}\right]+(\boldsymbol{\lambda}\odot\boldsymbol{\lambda})\mathbf{I}\,. \tag{1.143}$$

Then Newton's method gives the iterative rule

$$\mathbf{N}^{k+1} = \mathbf{N}^k - \left[\frac{\partial \mathbf{Y}}{\partial \mathbf{N}}\right]^{-1}_{\mathbf{N}^k}\cdot\mathbf{Y}(\mathbf{N}^k) \tag{1.144}$$

Given appropriate starting values, this method with converge to the solution except in situations where $\mathbf{X}\cdot\mathbf{N}<0$. It is preferable to normalize $\mathbf{N}$ in those situations where the exponential becomes too large.

Once the $\mathbf{N}$ vector has been found, the unit normal to the separation plane is determine by normalizing $\mathbf{n}=(N_1,N_2,N_3)$. Contact occurs at particle p if

$$\min_{p\in\beta}\left(\mathbf{x}_p\cdot\frac{\mathbf{n}}{\|\mathbf{n}\|}-R_p\right)-\max_{p\in\alpha}\left(\mathbf{x}_p\cdot\frac{\mathbf{n}}{\|\mathbf{n}\|}+R_p\right)<0 \tag{1.145}$$

where R_p is the distance from the centroid of particle p to its deformed edge along $\mathbf{n}$.

1.8 Implicit time integration

Recall from equation (1.61) that the MPM discretized momentum equations can be written as a semi-algebraic system

$$\sum_h M_{gh}\dot{\mathbf{v}}_h = \mathbf{f}_g^{\text{ext}} - \mathbf{f}_g^{\text{int}} + \mathbf{f}_g^{\text{body}} \; ; \quad g = 1 \dots N_g \tag{1.146}$$

where 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

$$\begin{aligned}
M_{gh} &:= \sum_p \rho_p \int_{\Omega_p} Y_p(\mathbf{x})\, S_g(\mathbf{x})\, S_h(\mathbf{x})\, d\Omega \\
\mathbf{f}_g^{\text{int}} &:= \sum_p V_p \boldsymbol{\sigma}_p \cdot \langle \nabla S_{gp} \rangle \\
\mathbf{f}_g^{\text{body}} &:= \sum_p m_p \mathbf{b}_p \, \langle S_{gp} \rangle \\
\mathbf{f}_g^{\text{ext}} &:= \int_{\Gamma_t} \bar{\mathbf{t}}(\mathbf{x})\, S_g(\mathbf{x})\, d\Gamma \, .
\end{aligned} \tag{1.147}$$

While the MPM background grid is reset after each time increment, MPM does not require it to be reset during each iteration of an implicit integration process. Therefore, during a time step, we can carry a grid displacement variable $\mathbf{u}$ that can be used to compute grid accelerations $\mathbf{a}_g = \dot{\mathbf{v}}_g$ and discarded at the end of a time step.

Let us express (1.146) in matrix form as

$$\boldsymbol{M}_g \boldsymbol{a}_g = \boldsymbol{f}_g^{\text{ext}} - \boldsymbol{f}_g^{\text{int}} + \boldsymbol{f}_g^{\text{body}} \tag{1.148}$$

Define the residual as

$$\boldsymbol{r}(\boldsymbol{u}_g^{n+1}, t_{n+1}) = \boldsymbol{M}_g \boldsymbol{a}_g^{n+1} - \boldsymbol{f}^{\text{ext}}(\boldsymbol{u}_g^{n+1}, t_{n+1}) + \boldsymbol{f}^{\text{int}}(\boldsymbol{u}_g^{n+1}, t_{n+1}) - \boldsymbol{f}^{\text{body}}(\boldsymbol{u}_g^{n+1}, t_{n+1}) = \mathbf{0}. \tag{1.149}$$

where the superscipt $n+1$ indicates quantities at time t_{n+1} and $\boldsymbol{u}_g$ is the $N_g \times 3$ matrix of grid node displacements. We use a Newmark-β method to integrate the acceleration:

$$\begin{aligned}
\boldsymbol{u}_g^{n+1} &= \boldsymbol{u}^\star + \beta \boldsymbol{a}_g^{n+1} (\Delta t)^2 \\
\boldsymbol{v}_g^{n+1} &= \boldsymbol{v}^\star + \gamma \boldsymbol{a}_g^{n+1} \Delta t
\end{aligned} \tag{1.150}$$

where

$$\begin{aligned}
\boldsymbol{u}^\star &= \boldsymbol{u}_g^n + \boldsymbol{v}_g^n \Delta t + \tfrac{1}{2}(1 - 2\beta) \boldsymbol{a}_g^n (\Delta t)^2 \\
\boldsymbol{v}^\star &= \boldsymbol{v}_g^n + (1 - \gamma) \boldsymbol{a}_g^n \Delta t \, .
\end{aligned} \tag{1.151}$$

1.8.1 Newton's method

In the Vaango implementation, the residual is expressed in terms of the displacement. We are required to do this because tangents needed in Newton's method are easier to compute when forces can be expressed in the form

$$\boldsymbol{f} = \boldsymbol{K} \cdot \boldsymbol{u} \tag{1.152}$$

where $\boldsymbol{K}$ is the stiffness matrix. If we were to use the velocity as the primary variable, as in explicit MPM, we would need rates of the forces instead:

$$\dot{\boldsymbol{f}} = \boldsymbol{K} \cdot \boldsymbol{v} + \dot{\boldsymbol{K}} \cdot \boldsymbol{u} \, . \tag{1.153}$$

The extra term involving the rate of change of the stiffness matrix complicates the process and we avoid it in VAANGO .

Then, using (1.149) and (1.150), we have

$$\boldsymbol{r}(\boldsymbol{u}_g^{n+1}, t_{n+1}) = \frac{1}{\beta \Delta t^2} \boldsymbol{M}_g (\boldsymbol{u}_g^{n+1} - \boldsymbol{u}^\star) - \boldsymbol{f}^{\text{ext}}(\boldsymbol{u}_g^{n+1}, t_{n+1}) + \boldsymbol{f}^{\text{int}}(\boldsymbol{u}_g^{n+1}, t_{n+1}) - \boldsymbol{f}^{\text{body}}(\boldsymbol{u}_g^{n+1}, t_{n+1}) = \mathbf{0}. \quad (1.154)$$

The problem then reduces to finding the solution $\boldsymbol{u}_g^{n+1}$ of the nonlinear system of equations (1.154). Newton's method is used in VAANGO with the starting value of $\boldsymbol{u}_g^{n+1} = \boldsymbol{u}_g^\star$. Dropping the subscript g temporarily for convenience, and denoting the current Newton iteration by the subscript k, we can linearize the residual at $\boldsymbol{u}_k^{n+1}$ using a Taylor expansion:

$$\mathbf{0} = \boldsymbol{r}(\boldsymbol{u}_{k+1}^{n+1}, t_{n+1}) = \boldsymbol{r}(\boldsymbol{u}_k^{n+1}, t_{n+1}) + \frac{\partial \boldsymbol{r}(\boldsymbol{u}_k^{n+1}, t_{n+1})}{\partial \boldsymbol{u}} (\boldsymbol{u}_{k+1}^{n+1} - \boldsymbol{u}_k^{n+1}) . \quad (1.155)$$

Rearranging the above equation,

$$\Delta \boldsymbol{u} = \boldsymbol{u}_{k+1}^{n+1} - \boldsymbol{u}_k^{n+1} = -\left[\frac{\partial \boldsymbol{r}(\boldsymbol{u}_k^{n+1}, t_{n+1})}{\partial \boldsymbol{u}} \right]^{-1} \boldsymbol{r}(\boldsymbol{u}_k^{n+1}, t_{n+1}) = -\boldsymbol{K}^{-1} \boldsymbol{r}(\boldsymbol{u}_k^{n+1}, t_{n+1}) . \quad (1.156)$$

This iterative process is continued until $\Delta \boldsymbol{u}$ is smaller than a given tolerance. The tangent matrix $\boldsymbol{K}$, of size $N_g \times N_g$, is

$$\boldsymbol{K} = \frac{\partial \boldsymbol{r}(\boldsymbol{u}_k^{n+1}, t_{n+1})}{\partial \boldsymbol{u}} . \quad (1.157)$$

This matrix is decomposed and evaluated separately for the internal and external forces, i.e.,

$$\boldsymbol{K} = \frac{\partial \boldsymbol{r}(\boldsymbol{u}_k^{n+1}, t_{n+1})}{\partial \boldsymbol{u}} = \frac{1}{\beta \Delta t^2} \boldsymbol{M}_g - \frac{\partial}{\partial \boldsymbol{u}} \left[\boldsymbol{f}^{\text{ext}}(\boldsymbol{u}_g^{n+1}, t_{n+1}) \right] + \frac{\partial}{\partial \boldsymbol{u}} \left[\boldsymbol{f}^{\text{int}}(\boldsymbol{u}_g^{n+1}, t_{n+1}) \right] - \frac{\partial}{\partial \boldsymbol{u}} \left[\boldsymbol{f}^{\text{body}}(\boldsymbol{u}_g^{n+1}, t_{n+1}) \right] . \quad (1.158)$$

Alternatively,

$$\boldsymbol{K} = \frac{1}{\beta \Delta t^2} \boldsymbol{M}_g - \boldsymbol{K}^{\text{ext}}(\boldsymbol{u}_g^{n+1}, t_{n+1}) + \boldsymbol{K}^{\text{int}}(\boldsymbol{u}_g^{n+1}, t_{n+1}) - \boldsymbol{K}^{\text{body}}(\boldsymbol{u}_g^{n+1}, t_{n+1}) . \quad (1.159)$$

1.8.2 Tangent stiffness matrix

The contribution to the tangent matrix ($\boldsymbol{K}$) from the internal forces is called the tangent stiffness matrix ($\boldsymbol{K}^{\text{int}}$). Since an updated Lagrangian formulation is used in MPM , we can compute the tangent stiffness using the configuration at time t_n as the reference configuration.

Recall from (1.147) that for explicit MPM we used

$$\mathbf{f}_g^{\text{int}} = \sum_p V_p \boldsymbol{\sigma}_p \cdot \langle \nabla S_{gp} \rangle \quad \text{where} \quad \langle S_{gp} \rangle := \frac{1}{V_p} \int_{\Omega_p} Y_p(\mathbf{x})\, S_g(\mathbf{x})\, d\Omega . \quad (1.160)$$

For the computation of the tangent matrix, it is preferable to start from the weak form of the momentum equation (1.4):

$$I = \int_\Omega \boldsymbol{\sigma} : \nabla \mathbf{w}\, d\Omega \quad (1.161)$$

which leads to integral form of equation (1.160) (see (1.56)):

$$\mathbf{f}_g^{\text{int}} = \sum_p \int_{\Omega_p} Y_p(\mathbf{x}) \boldsymbol{\sigma}_p \cdot \nabla S_g\, d\Omega . \quad (1.162)$$

Also, since we are typically working with rates of stress in the constitutive models, it is preferable to express all quantities in terms of stress rates that are objective. It is easier to work with the Lagrangian PK-1 stress ($\boldsymbol{P}$) at the beginning of the timestep rather than the spatial Cauchy stress ($\boldsymbol{\sigma}$).

To convert from the spatial description (1.161) to a Lagrangian material description, observe that

$$\underset{n+1}{\nabla}\mathbf{w} = \frac{\partial \mathbf{w}}{\partial \mathbf{x}^{n+1}} = \frac{\partial \mathbf{w}}{\partial \mathbf{x}^{n}} \cdot \frac{\partial \mathbf{x}^{n}}{\partial \mathbf{x}^{n+1}} = \frac{\partial \mathbf{w}}{\partial \mathbf{x}^{n}} \cdot (\Delta \boldsymbol{F}_n^{n+1})^{-1} = \frac{\partial \mathbf{w}}{\partial \mathbf{x}^{n}} \cdot \frac{\Delta \boldsymbol{F}_c^T}{J_n^{n+1}} = \underset{n}{\nabla}\mathbf{w} \cdot \frac{\Delta \boldsymbol{F}_c^T}{J_n^{n+1}} \tag{1.163}$$

where, with $\boldsymbol{F}$ as the deformation gradient,

$$\boldsymbol{F}^{n+1} = \Delta \boldsymbol{F}_n^{n+1} \boldsymbol{F}^n \,, \quad J_n^{n+1} = \det(\Delta \boldsymbol{F}_n^{n+1}) \,, \quad \Delta \boldsymbol{F}_c = \text{cofactor}(\Delta \boldsymbol{F}_n^{n+1}) \,. \tag{1.164}$$

Therefore,

$$\begin{aligned} I &= \int_{\Omega^{n+1}} \boldsymbol{\sigma}^{n+1} : \underset{n+1}{\nabla}\mathbf{w} \, d\Omega^{n+1} = \int_{\Omega^n} \boldsymbol{\sigma}^{n+1} : \underset{n+1}{\nabla}\mathbf{w} \, J_n^{n+1} \, d\Omega^n \\ &= \int_{\Omega^n} \boldsymbol{\sigma}^{n+1} : \left(\underset{n}{\nabla}\mathbf{w} \cdot \Delta \boldsymbol{F}_c^T \right) d\Omega^n = \int_{\Omega^n} (\boldsymbol{\sigma}^{n+1} \cdot \Delta \boldsymbol{F}_c) : \underset{n}{\nabla}\mathbf{w} \, d\Omega^n \\ &= \int_{\Omega^n} \boldsymbol{P}^n : \underset{n}{\nabla}\mathbf{w} \, d\Omega^n \end{aligned} \tag{1.165}$$

where $\boldsymbol{P}^n$ is the first Piola-Kirchhoff stress. Following the same process as used to derive (1.56), we get

$$\mathbf{f}_g^{\text{int}} = \sum_p \int_{\Omega_p^n} Y_p(\mathbf{x}^n) \boldsymbol{P}_p^n \cdot \underset{n}{\nabla} S_g \, d\Omega^n \,. \tag{1.166}$$

Taking the material time derivative of (1.166), we have

$$\dot{\mathbf{f}}_g^{\text{int}} = \sum_p \int_{\Omega_p^n} Y_p(\mathbf{x}^n) \dot{\boldsymbol{P}}_p^n \cdot \underset{n}{\nabla} S_g \, d\Omega^n \,. \tag{1.167}$$

Since the rate of the first Piola-Kirchhoff stress is not objective, it is easier to work with the rate of the second Piola-Kirchhoff stress ($\boldsymbol{S}$):

$$\boldsymbol{P} = \boldsymbol{F} \cdot \boldsymbol{S} \quad \Longrightarrow \quad \dot{\boldsymbol{P}} = \dot{\boldsymbol{F}} \cdot \boldsymbol{S} + \boldsymbol{F} \cdot \dot{\boldsymbol{S}} \,. \tag{1.168}$$

Substitution of (1.168) into (1.167) gives

$$\dot{\mathbf{f}}_g^{\text{int}} = \sum_p \int_{\Omega_p^n} Y_p(\mathbf{x}^n) \left[\dot{\boldsymbol{F}}_p^{n+1} \cdot \boldsymbol{S}_p^n + \boldsymbol{F}_p^{n+1} \cdot \dot{\boldsymbol{S}}_p^n \right] \cdot \underset{n}{\nabla} S_g \, d\Omega^n \,. \tag{1.169}$$

Separating out the two components, we have

$$\dot{\mathbf{f}}_g^{\text{int}} = \sum_p \int_{\Omega_p^n} Y_p(\mathbf{x}^n) \dot{\boldsymbol{F}}_p^{n+1} \cdot \boldsymbol{S}_p^n \cdot \underset{n}{\nabla} S_g \, d\Omega^n + \sum_p \int_{\Omega_p^n} Y_p(\mathbf{x}^n) \boldsymbol{F}_p^{n+1} \cdot \dot{\boldsymbol{S}}_p^n \cdot \underset{n}{\nabla} S_g \, d\Omega^n \,. \tag{1.170}$$

The rate of the internal force can then be expressed as

$$\dot{\mathbf{f}}_g^{\text{int}} = \dot{\mathbf{f}}_g^{\text{geo}} + \dot{\mathbf{f}}_g^{\text{mat}} \tag{1.171}$$

where the geometric and material rates of the internal forces are defined as

$$\begin{aligned} \dot{\mathbf{f}}_g^{\text{geo}} &:= \sum_p \int_{\Omega_p^n} Y_p(\mathbf{x}^n) \dot{\boldsymbol{F}}_p^{n+1} \cdot \boldsymbol{S}_p^n \cdot \underset{n}{\nabla} S_g \, d\Omega^n \\ \dot{\mathbf{f}}_g^{\text{mat}} &:= \sum_p \int_{\Omega_p^n} Y_p(\mathbf{x}^n) \boldsymbol{F}_p^{n+1} \cdot \dot{\boldsymbol{S}}_p^n \cdot \underset{n}{\nabla} S_g \, d\Omega^n \,. \end{aligned} \tag{1.172}$$

We can now use the constitutive relation between the second Piola-Kirchhoff stress and the Green strain ($\boldsymbol{E}$), the expression for the Green strain in terms of the deformation gradient, the relationship between the velocity gradient ($\boldsymbol{l}$) and the rate of change of the deformation gradient, and the definition of the rate-of-deformation ($\boldsymbol{d}$)

$$\dot{\boldsymbol{S}} = \mathcal{C} : \dot{\boldsymbol{E}} \,, \quad \boldsymbol{E} = \tfrac{1}{2}\left(\boldsymbol{F}^T \cdot \boldsymbol{F} - \boldsymbol{I}\right) \,, \quad \dot{\boldsymbol{F}} = \boldsymbol{l} \cdot \boldsymbol{F}, \quad \text{and} \quad \boldsymbol{d} = \tfrac{1}{2}(\boldsymbol{l} + \boldsymbol{l}^T) \tag{1.173}$$

to write the material and geometric rates of the internal force in (1.172) as

$$\begin{aligned}
\dot{\mathbf{f}}_g^{\text{mat}} &= \sum_p \int_{\Omega_p^n} Y_p(\mathbf{x}^n) \boldsymbol{F}_p^{n+1} \cdot \left[\mathcal{C}_p^n : \dot{\boldsymbol{E}}_p^n\right] \cdot \underset{n}{\nabla} S_g \, d\Omega^n \\
&= \tfrac{1}{2} \sum_p \int_{\Omega_p^n} Y_p(\mathbf{x}^n) \boldsymbol{F}_p^{n+1} \cdot \left[\mathcal{C}_p^n : \left((\dot{\boldsymbol{F}}_p^{n+1})^T \cdot \boldsymbol{F}_p^{n+1} + (\boldsymbol{F}_p^{n+1})^T \cdot \dot{\boldsymbol{F}}_p^{n+1}\right)\right] \cdot \underset{n}{\nabla} S_g \, d\Omega^n \\
&= \tfrac{1}{2} \sum_p \int_{\Omega_p^n} Y_p(\mathbf{x}^n) \boldsymbol{F}_p^{n+1} \cdot \left[\mathcal{C}_p^n : \left((\boldsymbol{F}_p^{n+1})^T \cdot (\boldsymbol{l}_p^{n+1})^T \cdot \boldsymbol{F}_p^{n+1} + (\boldsymbol{F}_p^{n+1})^T \cdot \boldsymbol{l}_p^{n+1} \cdot \boldsymbol{F}_p^{n+1}\right)\right] \cdot \underset{n}{\nabla} S_g \, d\Omega^n \\
&= \sum_p \int_{\Omega_p^n} Y_p(\mathbf{x}^n) \boldsymbol{F}_p^{n+1} \cdot \left[\mathcal{C}_p^n : \left((\boldsymbol{F}_p^{n+1})^T \cdot \boldsymbol{d}_p^{n+1} \cdot \boldsymbol{F}_p^{n+1}\right)\right] \cdot \underset{n}{\nabla} S_g \, d\Omega^n \\
\dot{\mathbf{f}}_g^{\text{geo}} &:= \sum_p \int_{\Omega_p^n} Y_p(\mathbf{x}^n) \left[(\boldsymbol{F}_p^{n+1})^T \cdot (\boldsymbol{l}_p^{n+1})^T\right] \cdot \boldsymbol{S}_p^n \cdot \underset{n}{\nabla} S_g \, d\Omega^n \,.
\end{aligned} \tag{1.174}$$

Recall the interpolation of the velocity from the grid nodes (h) to particles (p) can be computed using

$$\mathbf{v}_p(\mathbf{x}^{n+1}) = \sum_h \mathbf{v}_h^{n+1} S_h(\mathbf{x}^{n+1}) \tag{1.175}$$

Therefore,

$$\begin{aligned}
\boldsymbol{l}_p^{n+1} &= \underset{n+1}{\nabla} \mathbf{v}_p^{n+1}(\mathbf{x}^{n+1}) = \sum_h \mathbf{v}_h^{n+1} \otimes \underset{n+1}{\nabla} S_h \\
\boldsymbol{d}_p^{n+1} &= \tfrac{1}{2} \sum_h \left[\mathbf{v}_h^{n+1} \otimes \underset{n+1}{\nabla} S_h + \underset{n+1}{\nabla} S_h \otimes \mathbf{v}_h^{n+1}\right] .
\end{aligned} \tag{1.176}$$

Using (1.176) in (1.174), we have

$$\begin{aligned}
\dot{\mathbf{f}}_g^{\text{mat}} &= \tfrac{1}{2} \sum_h \sum_p \int_{\Omega_p^n} Y_p(\mathbf{x}^n) \boldsymbol{F}_p^{n+1} \cdot \left[\mathcal{C}_p^n : \left((\boldsymbol{F}_p^{n+1})^T \cdot \left[\underset{n+1}{\nabla} S_h \otimes \mathbf{v}_h^{n+1} + \mathbf{v}_h^{n+1} \otimes \underset{n+1}{\nabla} S_h\right] \cdot \boldsymbol{F}_p^{n+1}\right)\right] \cdot \underset{n}{\nabla} S_g \, d\Omega^n \\
\dot{\mathbf{f}}_g^{\text{geo}} &= \sum_h \sum_p \int_{\Omega_p^n} Y_p(\mathbf{x}^n) \left[(\boldsymbol{F}_p^{n+1})^T \cdot \left(\underset{n+1}{\nabla} S_h \otimes \mathbf{v}_h^{n+1}\right)\right] \cdot \boldsymbol{S}_p^n \cdot \underset{n}{\nabla} S_g \, d\Omega^n \,.
\end{aligned} \tag{1.177}$$

Since both the second Piola-Kirchhoff stress and the Green strain are symmetric, the tensor $\mathcal{C}$ has the symmetries $C_{ijkl} = C_{jikl} = C_{jilk}$. For hyperelastic materials we have the additional symmetry $C_{ijkl} = C_{klij}$. We can take advantage of these symmetries to simplify the above expressions. The first term in the expression for the rate of the material internal force contains an expression of the form

$$\mathbf{A} := \boldsymbol{F} \cdot \left[\mathcal{C} : \left(\boldsymbol{F}^T \cdot \left[\tilde{\mathbf{G}} \otimes \mathbf{v}\right] \cdot \boldsymbol{F}\right)\right] \cdot \mathbf{G} =: \boldsymbol{\alpha} \cdot \mathbf{v} \tag{1.178}$$

while the second term contains

$$\mathbf{B} := \boldsymbol{F} \cdot \left[\mathcal{C} : \left(\boldsymbol{F}^T \cdot \left[\mathbf{v} \otimes \tilde{\mathbf{G}}\right] \cdot \boldsymbol{F}\right)\right] \cdot \mathbf{G} =: \boldsymbol{\beta} \cdot \mathbf{v} \tag{1.179}$$

where

$$\mathbf{G} = \mathbf{G}_g := \underset{n}{\nabla} S_g \quad \text{and} \quad \tilde{\mathbf{G}} = \mathbf{G}_h := \underset{n+1}{\nabla} S_h \,. \tag{1.180}$$

In index notation,

$$\begin{aligned} A_r &= F_{ri} C_{ijk\ell} F^T_{km} \tilde{G}_m v_n F_{n\ell} G_j = F_{ri} C_{ijk\ell} F_{mk} \tilde{G}_m F_{n\ell} G_j v_n = G_j (\boldsymbol{F} \cdot \mathcal{C} \cdot \boldsymbol{F}^T)_{rjkn} (\tilde{\mathbf{G}} \cdot \boldsymbol{F})_k v_n \\ &= G_j (\tilde{\mathbf{G}} \cdot \boldsymbol{F})_k (\boldsymbol{F} \cdot \mathcal{C} \cdot \boldsymbol{F}^T)_{rjkn} v_n =: \alpha_{rn} v_n \\ B_r &= F_{ri} C_{ijk\ell} F^T_{kn} v_n \tilde{G}_m F_{m\ell} G_j = F_{ri} C_{ijk\ell} F_{nk} \tilde{G}_m F_{m\ell} G_j v_n = G_j (\boldsymbol{F} \cdot \mathcal{C} \cdot \boldsymbol{F}^T)_{rj\ell n} (\tilde{\mathbf{G}} \cdot \boldsymbol{F})_\ell v_n \\ &= G_j (\tilde{\mathbf{G}} \cdot \boldsymbol{F})_\ell (\boldsymbol{F} \cdot \mathcal{C} \cdot \boldsymbol{F}^T)_{rj\ell n} v_n =: \beta_{rn} v_n = \alpha_{rn} v_n = A_r \end{aligned} \tag{1.181}$$

Similarly for the geometrically nonlinear component, we have

$$\mathbf{C} := \left[\boldsymbol{F}^T \cdot \left(\tilde{\mathbf{G}} \otimes \mathbf{v} \right) \right] \cdot \boldsymbol{S} \cdot \mathbf{G} =: \boldsymbol{\gamma} \cdot \mathbf{v} \,. \tag{1.182}$$

In index notation

$$C_r = F_{ri} \tilde{G}_i v_n S_{nk} G_k = (\tilde{\mathbf{G}} \cdot \boldsymbol{F}^T)_r (\mathbf{G} \cdot \boldsymbol{S})_n v_n =: \gamma_{rn} v_n \,. \tag{1.183}$$

We can now express (1.177) as

$$\begin{aligned} \dot{\mathbf{f}}^{\text{mat}}_g &= \sum_h \left[\sum_p \int_{\Omega^n_p} Y_p(\mathbf{x}^n)\, \boldsymbol{\alpha}\, d\Omega^n \right] \cdot \mathbf{v}^{n+1}_h \\ \dot{\mathbf{f}}^{\text{geo}}_g &= \sum_h \left[\sum_p \int_{\Omega^n_p} Y_p(\mathbf{x}^n)\, \boldsymbol{\gamma}\, d\Omega^n \right] \cdot \mathbf{v}^{n+1}_h \end{aligned} \tag{1.184}$$

where

$$\begin{aligned} [\boldsymbol{\alpha}]_{i\ell} &= G_j \left[\tilde{\mathbf{G}} \cdot \boldsymbol{F}^{n+1}_p \right]_k \left[\boldsymbol{F}^{n+1}_p \cdot \mathcal{C}^n_p \cdot \left(\boldsymbol{F}^{n+1}_p \right)^T \right]_{ijk\ell} \\ [\boldsymbol{\gamma}]_{i\ell} &= \left[\tilde{\mathbf{G}} \cdot \left(\boldsymbol{F}^{n+1}_p \right)^T \right]_i \left[\mathbf{G} \cdot \boldsymbol{S}^n_p \right]_\ell \end{aligned} \tag{1.185}$$

Using

$$\dot{\mathbf{f}} = \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \cdot \mathbf{v} \tag{1.186}$$

where $\mathbf{u}$ is the displacement, we notice from (1.184) that

$$\begin{aligned} (\boldsymbol{K}^{\text{mat}})_{gh} &= \frac{\partial \mathbf{f}^{\text{mat}}_g}{\partial \mathbf{u}^{n+1}_h} = \sum_p \int_{\Omega^n_p} Y_p(\mathbf{x}^n)\, \boldsymbol{\alpha}\, d\Omega^n \\ (\boldsymbol{K}^{\text{geo}})_{gh} &= \frac{\partial \mathbf{f}^{\text{geo}}_g}{\partial \mathbf{u}^{n+1}_h} = \sum_p \int_{\Omega^n_p} Y_p(\mathbf{x}^n)\, \boldsymbol{\gamma}\, d\Omega^n \,. \end{aligned} \tag{1.187}$$

If we now set the current configuration as the reference configuration (see [18], section 6.1.3), we have

$$\boldsymbol{F}^{n+1}_p = \boldsymbol{I} \,, \quad \boldsymbol{S}^n_p = \boldsymbol{\sigma}^{n+1}_p \,, \quad \mathbf{x}^n = \mathbf{x}^{n+1} \,, \quad d\Omega_n = d\Omega_{n+1} \,, \quad \mathbf{G} = \tilde{\mathbf{G}} \,, \quad \mathcal{C}^n_p = (\mathcal{C}^\sigma)^{n+1}_p \,. \tag{1.188}$$

Therefore,

$$(\boldsymbol{K}^{\text{int}})_{gh}(\boldsymbol{u}^{n+1}_g, t_{n+1}) = (\boldsymbol{K}^{\text{mat}})_{gh} + (\boldsymbol{K}^{\text{geo}})_{gh} \tag{1.189}$$

where

$$\begin{aligned} (\boldsymbol{K}^{\text{mat}})_{gh} &= \sum_p \int_{\Omega^{n+1}_p} Y_p(\mathbf{x}^{n+1})\, \tilde{\mathbf{G}}_g \cdot (\mathcal{C}^\sigma)^{n+1}_p \cdot \tilde{\mathbf{G}}_h \, d\Omega^{n+1} \\ (\boldsymbol{K}^{\text{geo}})_{gh} &= \sum_p \int_{\Omega^{n+1}_p} Y_p(\mathbf{x}^{n+1})\, (\tilde{\mathbf{G}}_h \otimes \tilde{\mathbf{G}}_g) \cdot \boldsymbol{\sigma}^{n+1}_p \, d\Omega^{n+1} \,. \end{aligned} \tag{1.190}$$

An efficient way of converting these relations to Voigt form is possible only in the case where the grid basis functions are trilinear. For GIMP and CPDI basis functions, the problem becomes more complex and have not been implemented in Vaango .

1.8.3 External force stiffness matrix

Recall from (1.60) that the external force is given by

$$\mathbf{f}_g^{\text{ext}} := \int_{\Gamma_t} \bar{\mathbf{t}}(\mathbf{x})\, S_g(\mathbf{x})\, d\Gamma\,. \tag{1.191}$$

To find the contribution to the stiffness matrix from the external force, note that

$$\dot{\mathbf{f}}_g^{\text{ext}} := \int_{\Gamma_t} \left[\dot{\bar{\mathbf{t}}}(\mathbf{x})\, S_g(\mathbf{x}) + \bar{\mathbf{t}}(\mathbf{x})\, (\nabla S_g \cdot \mathbf{v}_g)\right]\, d\Gamma = \int_{\Gamma_t} \left[\dot{\bar{\mathbf{t}}}(\mathbf{x})\, S_g(\mathbf{x}) + (\bar{\mathbf{t}}(\mathbf{x}) \otimes \tilde{\mathbf{G}}_g) \cdot \mathbf{v}_g\right]\, d\Gamma \tag{1.192}$$

We make the simplifying assumption that

$$\dot{\bar{\mathbf{t}}}(\mathbf{x}) = \tilde{t}(\mathbf{x})\mathbf{v}_g \tag{1.193}$$

to get

$$\dot{\mathbf{f}}_g^{\text{ext}} := \int_{\Gamma_t} \left[\tilde{t}(\mathbf{x})\, S_g(\mathbf{x})\boldsymbol{I} + \bar{\mathbf{t}}(\mathbf{x}) \otimes \tilde{\mathbf{G}}_g\right] \cdot \mathbf{v}_g\, d\Gamma \tag{1.194}$$

Therefore,

$$(\boldsymbol{K}^{\text{ext}})_{gh} = \int_{\Gamma_t} \left[\tilde{t}(\mathbf{x})\, S_g(\mathbf{x})\boldsymbol{I} + \bar{\mathbf{t}}(\mathbf{x}) \otimes \tilde{\mathbf{G}}_g\right] \delta_{gh}\, d\Gamma \tag{1.195}$$

1.8.4 Body force stiffness matrix

The body force is given by

$$\mathbf{f}_g^{\text{ext}} := \sum_p \int_{\Omega_p} \rho_p Y_p(\mathbf{x}) S_g(\mathbf{x}) \mathbf{b}_p\, d\Omega \tag{1.196}$$

In Vaango we assume that the body force does not vary with deformation. Therefore,

$$(\boldsymbol{K}^{\text{body}})_{gh} = 0\,. \tag{1.197}$$

1.9 Pseudocode of explicit 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 $t = t_{n+1}$.

1.9.1 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`,

procedure INITIALIZE
  for `matl` **in** `materialList` **do**
    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()
    $\boldsymbol{F}_p^o$[matl] ← defGradComputer.INITIALIZE(matl)
    $\boldsymbol{\sigma}_p^o$[matl] ← constitutiveModel.INITIALIZE(matl)
    D_p^o[matl] ← damageModel.INITIALIZE(matl)
  end for
  if mpmFlags.initializeStressWithBodyForce = TRUE **then**
    $\mathbf{b}_p^o$ ← INITIALIZEBODYFORCE()
    $\boldsymbol{\sigma}_p^o, \boldsymbol{F}_p^o$ ← INITIALIZESTRESSANDDEFGRADFROMBODYFORCE()
  end if
  if mpmFlags.applyParticleBCs = TRUE **then**
    $\mathbf{f}_p^{\text{ext},o}$ ← particleBC.INITIALIZEPRESSUREBCS()
  end if
  return $n_p, \mathbf{x}_p^o, \mathbf{u}_p^o, m_p, V_p^o, \mathbf{v}_p^o, \mathbf{b}_p^o, \mathbf{f}_p^{\text{ext},o}, \boldsymbol{F}_p^o, \boldsymbol{\sigma}_p^o, D_p^o$
end procedure

1.9.2 Time advance

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

Algorithm 2 The MPM time advance algorithm

procedure TIMEADVANCE($\boldsymbol{h}_g, x_p^n, u_p^n, m_p, V_p^n, \mathbf{v}_p^n, \mathbf{f}_p^{\text{ext},n}, \boldsymbol{d}_p^n$)
  $\mathbf{b}_p^n$ ← COMPUTEPARTICLEBODYFORCE() ▷*Compute the body force term*
  $\mathbf{f}_p^{\text{ext},n+1}$ ← APPLYEXTERNALLOADS() ▷*Apply external loads to the particles*
  $m_g, V_g, \mathbf{v}_g, \mathbf{b}_g, \mathbf{f}_g^{\text{ext}}$ ← INTERPOLATEPARTICLESTOGRID() ▷*Interpolate particle data to the grid*
  EXCHANGEMOMENTUMINTERPOLATED() ▷*Exchange momentum between bodies on grid. Not discussed in this report.*
  $\mathbf{f}_g^{\text{int}}, \boldsymbol{\sigma}_g, \mathbf{v}_g$ ← COMPUTEINTERNALFORCE() ▷*Compute the internal force at the grid nodes*
  $\mathbf{v}_g^\star, \mathbf{a}_g$ ← COMPUTEANDINTEGRATEACCELERATION() ▷*Compute the grid velocity and grid acceleration*
  EXCHANGEMOMENTUMINTEGRATED() ▷*Exchange momentum between bodies on grid using integrated values. Not discussed in this report.*
  $\mathbf{v}_g^\star, \mathbf{a}_g$ ← SETGRIDBOUNDARYCONDITIONS() ▷*Update the grid velocity and grid acceleration using the BCs*
  $\boldsymbol{l}_p^n, \boldsymbol{F}_p^{n+1}, V_p^{n+1}$ ← COMPUTEDEFORMATIONGRADIENT() ▷*Compute the velocity gradient and the deformation gradient*

11: $\boldsymbol{\sigma}_p^{n+1}, \boldsymbol{\eta}_p^{n+1} \leftarrow$ computeStressTensor() ▷*Compute the updated stress and internal variables (if any)*
12: $\boldsymbol{\sigma}_p^{n+1}, \boldsymbol{\eta}_p^{n+1}, \chi_p^{n+1}, D_p^{n+1} \leftarrow$ computeBasicDamage() ▷*Compute the damage parameter and update the stress and internal variables*
13: $\chi_p^{n+1}, D_p^{n+1} \leftarrow$ updateErosionParameter() ▷*Update the indicator variable that is used to delete particles at the end of a time step*
14: $V_p^{n+1}, \mathbf{u}_p^{n+1}, \mathbf{v}_p^{n+1}, \mathbf{x}_p^{n+1}, m_p, \boldsymbol{h}_p^{n+1} \leftarrow$ interpolateToParticlesAndUpdate() ▷*Update the particle variables after interpolating grid quantities to particles*
15: **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: $\mathbf{x}_p^n, \mathbf{v}_p^n$, materialList, particleList, mpmFlags
procedure computeParticleBodyForce
  for matl **in** materialList **do**
    if mpmFlags.rotatingCoordSystem = TRUE **then**
      $\mathbf{g} \leftarrow$ mpmFlags.gravityAcceleration
      $\mathbf{b}_p^n$[matl] $\leftarrow \mathbf{g}$
    else
      for part **in** particleList **do**
        $\mathbf{g} \leftarrow$ mpmFlags.gravityAcceleration
        $\mathbf{x}_{rc} \leftarrow$ mpmFlags.coordRotationCenter
        $\mathbf{z}_r \leftarrow$ mpmFlags.coordRotationAxis
        $w \leftarrow$ mpmFlags.coordRotationSpeed
        $\boldsymbol{\omega} \leftarrow w\mathbf{z}_r$ ▷*Compute angular velocity vector*
        $\mathbf{a}_{\text{corolis}} \leftarrow 2\boldsymbol{\omega} \times \mathbf{v}_p^n$[matl, part] ▷*Compute Coriolis acceleration*
        $\boldsymbol{r} \leftarrow \mathbf{x}_p^n$[matl, part] $- \mathbf{x}_{rc}$
        $\mathbf{a}_{\text{centrifugal}} \leftarrow \boldsymbol{\omega} \times \boldsymbol{\omega} \times \boldsymbol{r}$ ▷*Compute the centrifugal body force acceleration*
        $\mathbf{b}_p^n$[matl,part] $\leftarrow \mathbf{g} - \mathbf{a}_{\text{centrifugal}} - \mathbf{a}_{\text{coriolis}}$ ▷*Compute the body force acceleration*
      end for
    end if
  end for
  return $\mathbf{b}_p^n$
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: $t_{n+1}, \mathbf{x}_p^n, \boldsymbol{h}_p^n, \mathbf{u}_p^n, \mathbf{f}_p^{\text{ext},n}, \boldsymbol{F}_p^n$, materialList, particleList, mpmFlags, particleBC
procedure applyExternalLoads

$f_p \leftarrow 0$
if mpmFlags.useLoadCurves = TRUE **then**
$f_p \leftarrow$ 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**
$\mathbf{f}_p^{\text{ext},n+1}$[matl,part] $\leftarrow$ particleBC.getForceVector(t_{n+1}, $\mathbf{x}_p^n$, $\boldsymbol{h}_p^n$, $\mathbf{u}_p^n$, f_p, $\boldsymbol{F}_p^n$) ▷*Compute the applied force vector at each particle*
end for
else
$\mathbf{f}_p^{\text{ext},n+1}$[matl] $\leftarrow \mathbf{f}_p^{\text{ext},n}$[matl]
end if
end for
return $\mathbf{f}_p^{\text{ext},n+1}$
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 $\leftarrow$ 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}, $S_{gp} \leftarrow$ 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] $\leftarrow m_g$[matl][node] + m_p[matl][part] S_{gp}[node]
V_g[matl][node] $\leftarrow V_g$[matl][node] + V_p^n[matl][part] S_{gp}[node]
$\mathbf{v}_g$[matl][node] $\leftarrow \mathbf{v}_g$[matl][node] + $\mathbf{p}_p$ S_{gp}[node]
$\mathbf{f}_g^{\text{ext}}$[matl][node] $\leftarrow \mathbf{f}_g^{\text{ext}}$[matl][node] + $\mathbf{f}_p^{\text{ext},n+1}$[matl][part] S_{gp}[node]
$\mathbf{b}_g$[node] $\leftarrow \mathbf{b}_g$[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] $\leftarrow \mathbf{v}_g$[matl][node]/m_g[matl][node]
end for
$\mathbf{v}_g$[matl] $\leftarrow$ applySymmetryVelocityBC($\mathbf{v}_g$[matl]) ▷*Apply any symmetry velocity BCs that may be applicable*

```
    end for
    return m_g, V_g, v_g, b_g, f_g^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
- $\mathbf{G}_{gp}$: 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: h_g, V_g, V_p^n, x_p^n, h_p^n, σ_p^n, 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, G_gp ←
                ↪ interpolator.findCellsAndWeightsAndShapeDervatives(x_p^n, h_p^n, F_p^n)
                ↪                ▷Find the node indices of the cells affecting the particle and
                ↪ the interpolation weights and gradients
            σ_v ← V_p[matl][part] σ_p^n[matl][part]
            for node in n_gp do
                f_g^int[matl][node] ← f_g^int[matl][node] - (G_gp[node]/h_g) ·σ_p^n[matl][part] V_p^n[part]
                σ_g[matl][node] ← σ_g[matl][node] + σ_v S_gp[node]
            end for
        end for
        for node in gridNodeList do
            σ_g[matl][node] ← σ_g[matl][node]/V_g[matl][node]
        end for
        v_g[matl] ← applySymmetryTractionBC()        ▷Apply any symmetry tractions BCs
            ↪ that may be applicable
    end for
    return f_g^int, σ_g, 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] $\leftarrow$ ($\mathbf{f}_g^{\text{int}}$[matl][node] + $\mathbf{f}_g^{\text{ext}}$[matl][node] + $\mathbf{b}_g$[matl][node])/m_g[matl][node]
      $\mathbf{v}_g^\star \leftarrow \mathbf{v}_g$[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

procedure SETGRIDBOUNDARYCONDITIONS
  for matl **in** materialList **do**
    $\mathbf{v}_g^\star$[matl] $\leftarrow$ APPLYSYMMETRYVELOCITYBC($\mathbf{v}_g^\star$[matl])
    for node **in** gridNodeList **do**
      $\mathbf{a}_g$[matl][node] $\leftarrow$ ($\mathbf{v}_g^\star$[matl][node] - $\mathbf{v}_g$[matl][node]) /Δt
    end for
  end for
  return $\mathbf{v}_g^\star$, $\mathbf{a}_g$
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

procedure COMPUTEDEFORMATIONGRADIENT
  interpolator $\leftarrow$ CREATEINTERPOLATOR(mpmFlags)
  for matl **in** materialList **do**
    for part **in** particleList **do**
      $\boldsymbol{l}_p^{n+1}$[matl,part] $\leftarrow$ 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*
      $\boldsymbol{F}_p^{n+1}$[matl,part], $\Delta\boldsymbol{F}_p^{n+1}$ $\leftarrow$ COMPUTEDEFORMATIONGRADIENTFROMVELOCITY($\boldsymbol{l}_p^n$[matl,part], $\boldsymbol{l}_p^{n+1}$[matl,part], $\boldsymbol{F}_p^n$[matl,part]) ▷*Compute the deformation gradient*
      V_p^{n+1}[matl,part] $\leftarrow$ m_p[matl,part]/ρ_0 $*$ det($\boldsymbol{F}_p^{n+1}$[matl,part])

```
        end for
    end for
    return l_p^{n+1}, F_p^{n+1}, V_p^{n+1}
end procedure
```

Algorithm 10 Computing the deformation gradient using the velocity gradient

```
Require: Δt, l_p^{n+1}, F_p^n, mpmFlags
procedure computeDeformationGradientFromVelocity
    if mpmFlags.defGradAlgorithm = "first_order" then
        F_p^{n+1}, ΔF_p^{n+1} ← seriesUpdateConstantVelGrad(numTerms = 1, Δt, l_p^{n+1}, F_p^n)
    else if mpmFlags.defGradAlgorithm = "subcycle" then
        F_p^{n+1}, ΔF_p^{n+1} ← subcycleUpdateConstantVelGrad(Δt, l_p^{n+1}, F_p^n)
    else if mpmFlags.defGradAlgorithm = "taylor_series" then
        F_p^{n+1}, ΔF_p^{n+1} ← seriesUpdateConstantVelGrad(numTerms = mpmFlags.numTaylorSeriesTerms,
Δt, l_p^{n+1}, F_p^n)
    else
        F_p^{n+1}, ΔF_p^{n+1} ← cayleyUpdateConstantVelGrad(Δt, l_p^{n+1}, F_p^n)
    end if
    return F_p^{n+1}, ΔF_p^{n+1}
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: Δt, x_p^n, m_p, V_p^{n+1}, h_p^n, l_p^{n+1}, F_p^{n+1}, σ_p^n, η_p^n, ρ_o, materialList, mpmFlags, constitutiveModel
procedure computeStressTensor
    for matl in materialList do
        σ^{n+1}, η_p^{n+1} ← constitutiveModel[matl].computeStressTensor(Δt, x_p^n, m_p, V_p^{n+1}, h_p^n,
                ↪ l_p^{n+1}, F_p^{n+1}, σ_p^n, η_p^n, ρ_o, mpmFlags)        ▷Update the stress and any
                ↪ internal variables needed by the constitutive model
    end for
    return σ_p^{n+1}, η_p^{n+1}
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}, F_p^{n+1}, σ_p^{n+1}, D_p^n, ε_p^{f,n}, χ_p^n, t_p^{χ,n}, materialList, mpmFlags
```

procedure COMPUTEDAMAGE
  for `matl` **in** `materialList` **do**
    for `part` **in** `particleList` **do**
      if `brittleDamage` = TRUE **then**
        $\boldsymbol{\sigma}_p^{n+1}, \varepsilon_p^{f,n+1}, \chi_p^{n+1}, t_p^{\chi,n+1}, D_p^{n+1} \leftarrow$ UPDATEDAMAGEANDMODIFYSTRESS($V_p^{n+1}, \boldsymbol{F}_p^{n+1}$, $\hookrightarrow$ $\boldsymbol{\sigma}_p^{n+1}, D_p^n, \varepsilon_p^{f,n}, \chi_p^n, t_p^{\chi,n}$) ▷*Update the damage parameters and stress*
      else
        $\boldsymbol{\sigma}_p^{n+1}, \varepsilon_p^{f,n+1}, \chi_p^{n+1}, t_p^{\chi,n+1} \leftarrow$ UPDATEFAILEDPARTICLESANDMODIFYSTRESS($V_p^{n+1}, \boldsymbol{F}_p^{n+1}$, $\hookrightarrow$ $\boldsymbol{\sigma}_p^{n+1}, \varepsilon_p^{f,n}, \chi_p^n, t_p^{\chi,n}, t^{n+1}$) ▷*Update the failed particles and stress*
      end if
    end for
  end for
  return $\boldsymbol{\sigma}_p^{n+1}, \varepsilon_p^{f,n+1}, \chi_p^{n+1}, t_p^{\chi,n+1}, D_p^{n+1}$
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

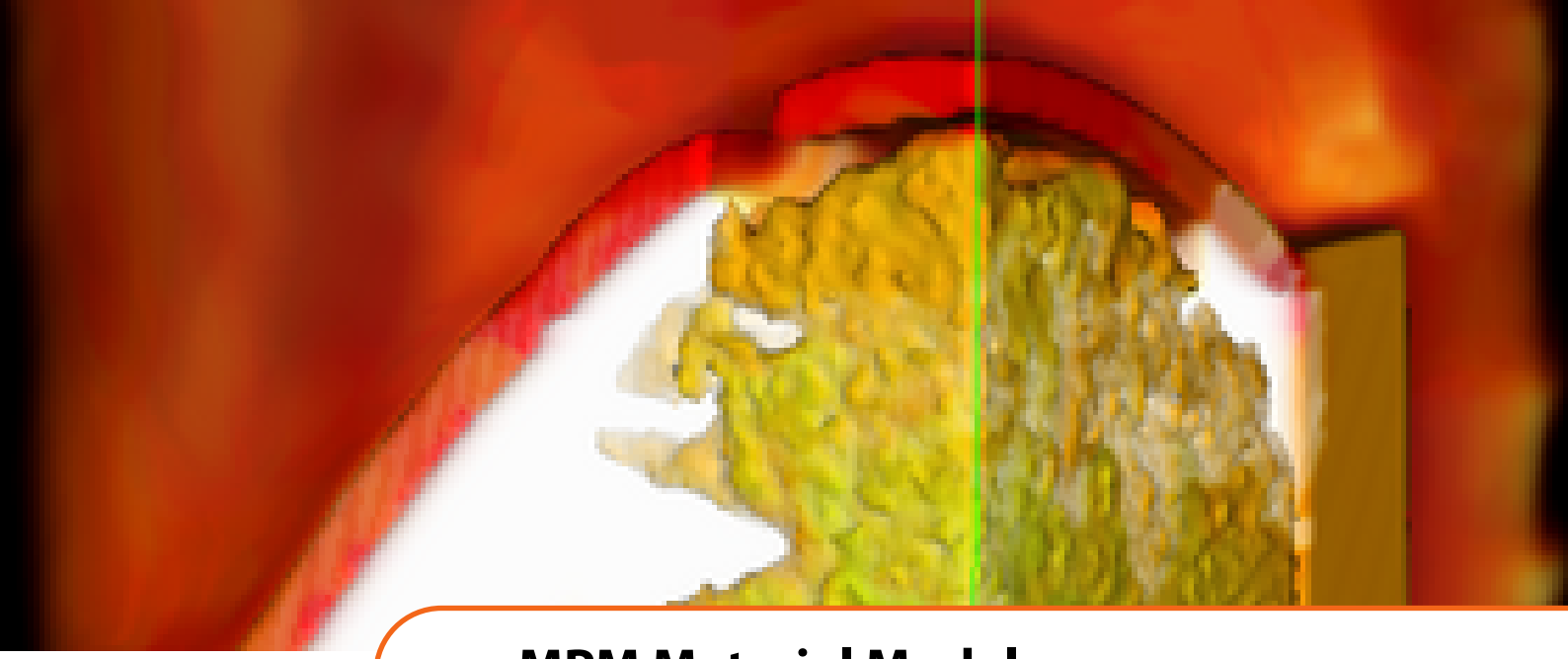
This is the final step at which the particle velocities and positions are updated and the grid is reset. Particle that are to be removed are dealt with in a subsequent `relocation` step.

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`$(\mathbf{x}_p^n, \boldsymbol{h}_p^{n+1}, \boldsymbol{F}_p^{n+1})$

```
7:              v ← 0,   a ← 0,
8:              for node in gridNodeList do
9:                  v ← v + v_g^⋆[node] ∗ S_gp[node]          ▷Update particle velocity
10:                 a ← a + a_g[node] ∗ S_gp[node]            ▷Update particle acceleration
11:             end for
12:             x_p^{n+1} ← x_p^n + v ∗ Δt                    ▷Update position
13:             u_p^{n+1} ← u_p^n + v ∗ Δt                    ▷Update displacement
14:             v_p^{n+1} ← v_p^n + a ∗ Δt                    ▷Update velocity
15:         end for
16:     end for
17:     deleteRogueParticles()                                ▷Delete particles that are to be eroded.
18:     return V_p^{n+1}, u_p^{n+1}, v_p^{n+1}, x_p^{n+1}, m_p, h_p^{n+1}
19: end procedure
```



2 — MPM Material Models

In this chapter we discuss some general features of the MPM material models. Individual models are complex and are discussed in separate chapters. Notation and definitions that are used frequently are also elaborated upon here.

2.1 Notation and definitions

A primary assumption made in many of the material models in Vaango is that stresses and (moderate) strains can be additively decomposed into volumetric and deviatoric parts.

2.1.1 Volumetric-deviatoric decomposition

The volumetric-deviatoric decomposition of stress ($\boldsymbol{\sigma}$) is expressed as

$$\boldsymbol{\sigma} = p\boldsymbol{I} + \boldsymbol{s} \tag{2.1}$$

where the mean stress (p) and the deviatoric stress ($\boldsymbol{s}$) are given by

$$p = \tfrac{1}{3}\mathrm{tr}(\boldsymbol{\sigma}) = \tfrac{1}{3}\boldsymbol{\sigma} : \boldsymbol{I} = \tfrac{1}{3}(\sigma_{11} + \sigma_{22} + \sigma_{33}) \quad \text{and} \quad \boldsymbol{s} = \boldsymbol{\sigma} - p\boldsymbol{I} = \begin{bmatrix} \sigma_{11} - p & \sigma_{12} & \sigma_{13} \\ \sigma_{12} & \sigma_{22} - p & \sigma_{23} \\ \sigma_{13} & \sigma_{23} & \sigma_{33} - p \end{bmatrix}. \tag{2.2}$$

Similarly, the volumeric-deviatoric split of the strain ($\boldsymbol{\varepsilon}$) is expressed as

$$\boldsymbol{\varepsilon} = \tfrac{1}{3}\varepsilon_v \boldsymbol{I} + \boldsymbol{\varepsilon}_s \tag{2.3}$$

where the volumetric strain (ε_v) and the deviatoric strain ($\boldsymbol{\varepsilon}_s$) are defined as

$$\varepsilon_v = \mathrm{tr}(\boldsymbol{\varepsilon}) = \boldsymbol{\varepsilon} : \boldsymbol{I} = \varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33} \quad \text{and} \quad \boldsymbol{\varepsilon}_s = \boldsymbol{\varepsilon} - \tfrac{1}{3}\varepsilon_v \boldsymbol{I} = \begin{bmatrix} \varepsilon_{11} - \frac{1}{3}\varepsilon_v & \varepsilon_{12} & \varepsilon_{13} \\ \varepsilon_{12} & \varepsilon_{22} - \frac{1}{3}\varepsilon_v & \varepsilon_{23} \\ \varepsilon_{13} & \varepsilon_{23} & \varepsilon_{33} - \frac{1}{3}\varepsilon_v \end{bmatrix}. \tag{2.4}$$

2.1.2 Stress invariants

The principal invariants and principal deviatoric invariants of the stress are used in several models. Frequently used invariants are:

$$\begin{aligned}
I_1 &= \mathrm{tr}(\boldsymbol{\sigma}) = \sigma_{11} + \sigma_{22} + \sigma_{33} \\
I_2 &= \tfrac{1}{2}\left[\mathrm{tr}(\boldsymbol{\sigma})^2 - \mathrm{tr}(\boldsymbol{\sigma}^2)\right] = \sigma_{11}\sigma_{22} + \sigma_{22}\sigma_{33} + \sigma_{33}\sigma_{11} - \left(\sigma_{12}^2 + \sigma_{23}^2 + \sigma_{13}^2\right) \\
I_3 &= \det(\boldsymbol{\sigma}) = \sigma_{11}\sigma_{22}\sigma_{33} + 2\sigma_{12}\sigma_{23}\sigma_{13} - \sigma_{12}^2\sigma_{33} - \sigma_{23}^2\sigma_{11} - \sigma_{13}^2\sigma_{22} \\
J_2 &= \tfrac{1}{2}\boldsymbol{s} : \boldsymbol{s} = \tfrac{1}{6}\left[(\sigma_{11} - \sigma_{22})^2 + (\sigma_{22} - \sigma_{33})^2 + (\sigma_{33} - \sigma_{11})^2\right] + \left(\sigma_{12}^2 + \sigma_{23}^2 + \sigma_{31}^2\right) \\
J_3 &= \det(\boldsymbol{s}) = \tfrac{2}{27}I_1^3 - \tfrac{1}{3}I_1 I_2 + I_3
\end{aligned} \tag{2.5}$$

Alternatives to I_1, J_2 and j_3 are p, q, and θ, defined as

$$p := \frac{1}{3}I_1\,, \quad q := \sqrt{3J_2}\,, \quad \cos 3\theta := \left(\frac{r}{q}\right)^3 = \frac{3\sqrt{3}}{2}\frac{J_3}{J_2^{3/2}}\,, \quad r^3 = \frac{27}{2}J_3\,. \tag{2.6}$$

A geometric accurate view of the stress state and yield surfaces is obtained if the isomorphic cylindrical coordinates z, ρ, and θ are used instead, where

$$z := \frac{I_1}{\sqrt{3}} = \sqrt{3}p\,, \quad \rho := \sqrt{2J_2} = \sqrt{\frac{2}{3}}q\,, \quad \cos 3\theta := \frac{3\sqrt{3}}{2}\frac{J_3}{J_2^{3/2}}\,. \tag{2.7}$$

2.1.3 Effective stress and strain

The effective stress and strain (sometimes also referred to as the shear stress and shear strain in the code) are defined such that the product is equal to the plastic work done. These measures are strictly applicable only to J_2 plasticity models but have also been used elsewhere.

The effective stress is defined as

$$\sigma_{\mathrm{eff}} = q = \sqrt{3J_2} = \sqrt{\tfrac{3}{2}\boldsymbol{s} : \boldsymbol{s}} = \sqrt{\tfrac{1}{2}\left[(\sigma_{11} - \sigma_{22})^2 + (\sigma_{22} - \sigma_{33})^2 + (\sigma_{33} - \sigma_{11})^2\right] + 3\left(\sigma_{12}^2 + \sigma_{23}^2 + \sigma_{31}^2\right)}\,. \tag{2.8}$$

The effective strain is defined as

$$\varepsilon_{\mathrm{eff}} = \sqrt{\tfrac{2}{3}\boldsymbol{\varepsilon}_s : \boldsymbol{\varepsilon}_s} \tag{2.9}$$

so that

$$\sigma_{\mathrm{eff}}\varepsilon_{\mathrm{eff}} = \sqrt{(\boldsymbol{s} : \boldsymbol{s})(\boldsymbol{\varepsilon}_s : \boldsymbol{\varepsilon}_s)}\,. \tag{2.10}$$

From the definition of $\boldsymbol{\varepsilon}_s$ we see that

$$\begin{aligned}
\boldsymbol{\varepsilon}_s : \boldsymbol{\varepsilon}_s &= \boldsymbol{\varepsilon} : \boldsymbol{\varepsilon} - \tfrac{2}{3}\mathrm{tr}(\boldsymbol{\varepsilon})\boldsymbol{I} : \boldsymbol{\varepsilon} + \tfrac{1}{9}\left[\mathrm{tr}(\boldsymbol{\varepsilon})\right]^2 \boldsymbol{I} : \boldsymbol{I} = \boldsymbol{\varepsilon} : \boldsymbol{\varepsilon} - \tfrac{2}{3}\left[\mathrm{tr}(\boldsymbol{\varepsilon})\right]^2 + \tfrac{1}{3}\left[\mathrm{tr}(\boldsymbol{\varepsilon})\right]^2 = \boldsymbol{\varepsilon} : \boldsymbol{\varepsilon} - \tfrac{1}{3}\left[\mathrm{tr}(\boldsymbol{\varepsilon})\right]^2 \\
&= \varepsilon_{11}^2 + \varepsilon_{22}^2 + \varepsilon_{33}^2 + 2\varepsilon_{12}^2 + 2\varepsilon_{23}^2 + 2\varepsilon_{13}^2 - \tfrac{1}{3}\left[\varepsilon_{11}^2 + \varepsilon_{22}^2 + \varepsilon_{23}^2 + 2\varepsilon_{11}\varepsilon_{22} + 2\varepsilon_{22}\varepsilon_{33} + 2\varepsilon_{11}\varepsilon_{33}\right] \\
&= \tfrac{1}{3}\left[(\varepsilon_{11} - \varepsilon_{22})^2 + (\varepsilon_{22} - \varepsilon_{33})^2 + (\varepsilon_{33} - \varepsilon_{11})^2\right] + 2\left(\varepsilon_{12}^2 + \varepsilon_{23}^2 + \varepsilon_{13}^2\right)
\end{aligned} \tag{2.11}$$

Therefore,

$$\varepsilon_{\mathrm{eff}} = \sqrt{\tfrac{2}{3}\left[\tfrac{1}{3}\left[(\varepsilon_{11} - \varepsilon_{22})^2 + (\varepsilon_{22} - \varepsilon_{33})^2 + (\varepsilon_{33} - \varepsilon_{11})^2\right] + 2\left(\varepsilon_{12}^2 + \varepsilon_{23}^2 + \varepsilon_{13}^2\right)\right]} \tag{2.12}$$

For volume preserving plastic deformations, $\mathrm{tr}(\boldsymbol{\varepsilon}) = 0$, and we have

$$\varepsilon_{\mathrm{eff}} = \sqrt{\tfrac{2}{3}\left(\varepsilon_{11}^2 + \varepsilon_{22}^2 + \varepsilon_{33}^2\right) + \tfrac{4}{3}\left(\varepsilon_{12}^2 + \varepsilon_{23}^2 + \varepsilon_{13}^2\right)} = \sqrt{\tfrac{2}{3}\left(\varepsilon_{11}^2 + \varepsilon_{22}^2 + \varepsilon_{33}^2\right) + \tfrac{1}{3}\left(\gamma_{12}^2 + \gamma_{23}^2 + \gamma_{13}^2\right)}\,. \tag{2.13}$$

2.1.4 Equivalent strain rate and plastic strain

The equivalent strain rate is defined as

$$\dot{\varepsilon}^{\text{eq}} = \sqrt{\dot{\boldsymbol{\varepsilon}} : \dot{\boldsymbol{\varepsilon}}} \tag{2.14}$$

where $\dot{\boldsymbol{\varepsilon}}(t)$ is the strain rate tensor. The distortional equivalent strain rate is

$$\dot{\gamma}^{\text{eq}} = \sqrt{\tfrac{2}{3}\dot{\boldsymbol{\varepsilon}}_s : \dot{\boldsymbol{\varepsilon}}_s} \tag{2.15}$$

where $\dot{\boldsymbol{\varepsilon}}_s(t)$ is the deviatoric strain rate tensor.

For models where an equivalent plastic strain is computed, we define a scalar equivalent plastic strain rate as

$$\dot{\varepsilon}_p^{\text{eq}} = \sqrt{\dot{\boldsymbol{\varepsilon}}^p : \dot{\boldsymbol{\varepsilon}}^p} \tag{2.16}$$

where $\dot{\boldsymbol{\varepsilon}}^p(t)$ is the plastic strain rate tensor. The definition of the scalar equivalent plastic strain is

$$\varepsilon_p^{\text{eq}}(t) = \int_0^t \dot{\varepsilon}_p^{\text{eq}}(\tau) d\tau \,. \tag{2.17}$$

The corresponding distortional equivalent plastic strain rate and strain are defined as

$$\dot{\gamma}_p^{\text{eq}} = \sqrt{\tfrac{2}{3}\dot{\boldsymbol{\varepsilon}}_s^p : \dot{\boldsymbol{\varepsilon}}_s^p} \tag{2.18}$$

and

$$\gamma_p^{\text{eq}}(t) = \int_0^t \dot{\gamma}_p^{\text{eq}}(\tau) d\tau \,. \tag{2.19}$$

2.1.5 Velocity gradient, rate-of-deformation, deformation gradient

The velocity gradient is represented by $\boldsymbol{l}$ and the defromation gradient by $\boldsymbol{F}$. The rate-of-deformation is

$$\boldsymbol{d} = \tfrac{1}{2}(\boldsymbol{l} + \boldsymbol{l}^T) = \tfrac{1}{2}(\nabla \mathbf{v} + \nabla \mathbf{v}^T) \,. \tag{2.20}$$

2.1.6 Eigenvectors and coordinate transformations

For most situations, tensor components in VAANGO are expressed in terms of the basis vectors $\mathbf{e}_1 = (1,0,0)$, $\mathbf{e}_2 = (0,1,0)$, and $\mathbf{e}_3 = (0,0,1)$. However, in some situations tensor components have to be expressed in the eigenbasis of a second-order tensor. Let these eigenvectors be $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$. Then a vector $\mathbf{a}$ with components (a_1, a_2, a_2) in the original basis has components (a_1', a_2', a_3') in the eigenbasis. The two sets of components are related by

$$\begin{bmatrix} a_1' \\ a_2' \\ a_3' \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1 \cdot \mathbf{v}_1 & \mathbf{e}_2 \cdot \mathbf{v}_1 & \mathbf{e}_3 \cdot \mathbf{v}_1 \\ \mathbf{e}_1 \cdot \mathbf{v}_2 & \mathbf{e}_2 \cdot \mathbf{v}_2 & \mathbf{e}_3 \cdot \mathbf{v}_2 \\ \mathbf{e}_1 \cdot \mathbf{v}_3 & \mathbf{e}_2 \cdot \mathbf{v}_3 & \mathbf{e}_3 \cdot \mathbf{v}_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} v_{11} & v_{12} & v_{13} \\ v_{21} & v_{22} & v_{23} \\ v_{31} & v_{32} & v_{33} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \boldsymbol{Q} \cdot \mathbf{a} \,. \tag{2.21}$$

The matrix that is used for this coordinate transformation, $\boldsymbol{Q}$, is given by

$$\boldsymbol{Q}^T := \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix} \tag{2.22}$$

where $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$ are column vectors representing the components of the eigenvectors in the reference basis.

The above coordinate transformation for vectors can be written in index notation as

$$a_i' = Q_{ij} a_j \,. \tag{2.23}$$

For transformations of second-order tensors, we have

$$T'_{ij} = Q_{ip} Q_{jq} T_{pq} \,. \tag{2.24}$$

For fourth-order tensors, the transformation relation is

$$C'_{ijk\ell} = Q_{im} Q_{jn} Q_{kp} Q_{\ell q} C_{mnpq} \,. \tag{2.25}$$

If the second-order tensor $\boldsymbol{T}$ is symmetric, we can express it in Mandel notation as a six-dimensional vector $\hat{\mathbf{t}}$:

$$\hat{\mathbf{t}} = \begin{bmatrix} T_{11} & T_{22} & T_{33} & \sqrt{2}T_{23} & \sqrt{2}T_{31} & \sqrt{2}T_{12} \end{bmatrix}^T \tag{2.26}$$

Then the transformation matrix is a 6×6 matrix, $\hat{\boldsymbol{Q}}$, such that

$$\hat{t}'_i = \hat{Q}_{ij} \hat{t}_j \,. \tag{2.27}$$

The matrix $\hat{\boldsymbol{Q}}$ has components [19],

$$\hat{\boldsymbol{Q}} = \begin{bmatrix} Q_{11}^2 & Q_{12}^2 & Q_{13}^2 & \sqrt{2}Q_{12}Q_{13} & \sqrt{2}Q_{11}Q_{13} & \sqrt{2}Q_{11}Q_{12} \\ Q_{21}^2 & Q_{22}^2 & Q_{23}^2 & \sqrt{2}Q_{22}Q_{23} & \sqrt{2}Q_{21}Q_{23} & \sqrt{2}Q_{21}Q_{22} \\ Q_{31}^2 & Q_{32}^2 & Q_{33}^2 & \sqrt{2}Q_{32}Q_{33} & \sqrt{2}Q_{31}Q_{33} & \sqrt{2}Q_{31}Q_{32} \\ \sqrt{2}Q_{21}Q_{31} & \sqrt{2}Q_{22}Q_{32} & \sqrt{2}Q_{23}Q_{33} & Q_{22}Q_{33} + Q_{23}Q_{32} & Q_{21}Q_{33} + Q_{31}Q_{23} & Q_{21}Q_{32} + Q_{31}Q_{22} \\ \sqrt{2}Q_{11}Q_{31} & \sqrt{2}Q_{12}Q_{32} & \sqrt{2}Q_{13}Q_{33} & Q_{12}Q_{33} + Q_{32}Q_{13} & Q_{11}Q_{33} + Q_{13}Q_{31} & Q_{11}Q_{32} + Q_{31}Q_{12} \\ \sqrt{2}Q_{11}Q_{31} & \sqrt{2}Q_{12}Q_{22} & \sqrt{2}Q_{13}Q_{23} & Q_{12}Q_{23} + Q_{22}Q_{13} & Q_{11}Q_{23} + Q_{21}Q_{13} & Q_{11}Q_{22} + Q_{21}Q_{12} \end{bmatrix} \tag{2.28}$$

Similarly, the transformation relation for fourth-order tensors simplifies to

$$\hat{C}'_{ij} = \hat{Q}_{ip} \hat{Q}_{jq} \hat{C}_{pq} \tag{2.29}$$

where

$$\hat{\boldsymbol{C}} = \begin{bmatrix} C_{1111} & C_{1122} & C_{1133} & \sqrt{2}C_{1123} & \sqrt{2}C_{1131} & \sqrt{2}C_{1112} \\ C_{2211} & C_{2222} & C_{2233} & \sqrt{2}C_{2223} & \sqrt{2}C_{2231} & \sqrt{2}C_{2212} \\ C_{3311} & C_{3322} & C_{3333} & \sqrt{2}C_{3323} & \sqrt{2}C_{3331} & \sqrt{2}C_{3312} \\ \sqrt{2}C_{2311} & \sqrt{2}C_{2322} & \sqrt{2}C_{2333} & 2C_{2323} & 2C_{2331} & 2C_{2312} \\ \sqrt{2}C_{3111} & \sqrt{2}C_{3122} & \sqrt{2}C_{3133} & 2C_{3123} & 2C_{3131} & 2C_{3112} \\ \sqrt{2}C_{1211} & \sqrt{2}C_{1222} & \sqrt{2}C_{1233} & 2C_{1223} & 2C_{1231} & 2C_{1212} \end{bmatrix} \tag{2.30}$$

2.2 Material models available in Vaango

The MPM material models implemented in Vaango were originally chosen for the following 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 and polymers under various strain rates and temperatures.
- To model the deformation of biological tissues .
- To model the explosive deformation of rocks and soils .

As of Vaango Version 20 .9 .18 , the material models that have been implemented are:

1. Rigid material

2. Ideal gas material
3. Water material
4. Membrane material
5. Programmed burn material
6. Tabular equation of state
7. Murnaghan equation of state
8. JWL++ equation of state
9. Hypoelastic material
10. Hypoelastic material with manufactured solutions
11. Hypoelastic material implementation in FORTRAN
12. Polar-orthotropic hypoelastic material
13. Compressible neo-Hookean hyperelastic material
14. Compressible neo-Hookean hyperelastic material with manufactured solutions
15. Compressible neo-Hookean hyperelastic material with damage
16. Unified explicit/implicit compressible Neo-Hookean hyperelastic material with damage
17. Compressible Neo-Hookean hyperelastic-J_2 plastic material with damage
18. Compressible Mooney-Rivlin hyperelastic material
19. Compressible neo-Hookean material for shells
20. Transversely isotropic hyperelastic material
21. The p-α model for porous materials
22. Viscoelastic material written in FORTRAN for damping
23. Simplified Maxwell viscoelastic material
24. Visco-SCRAM model for viscoelastic materials with cracks
25. Visco-SCRAM hotspot model
26. Tabular plasticity model
27. Tabular plasticity model with cap
28. Hypoelastic J_2 plasticity model with damage for high-rates
29. Viscoplastic J_2 plasticity model
30. Mohr-Coulomb material
31. Drucker-Prager material with deformation induced elastic anisotropy
32. CAM-Clay model for soils
33. Nonlocal Drucker-Prager material
34. Arenisca material for rocks and soils
35. Arenisca3 material for rocks and soils
36. Arena material for partially saturated soils
37. Arena-mixture material for mixes of partially saturated sand and clay
38. Brannon's soil model
39. Soil foam model

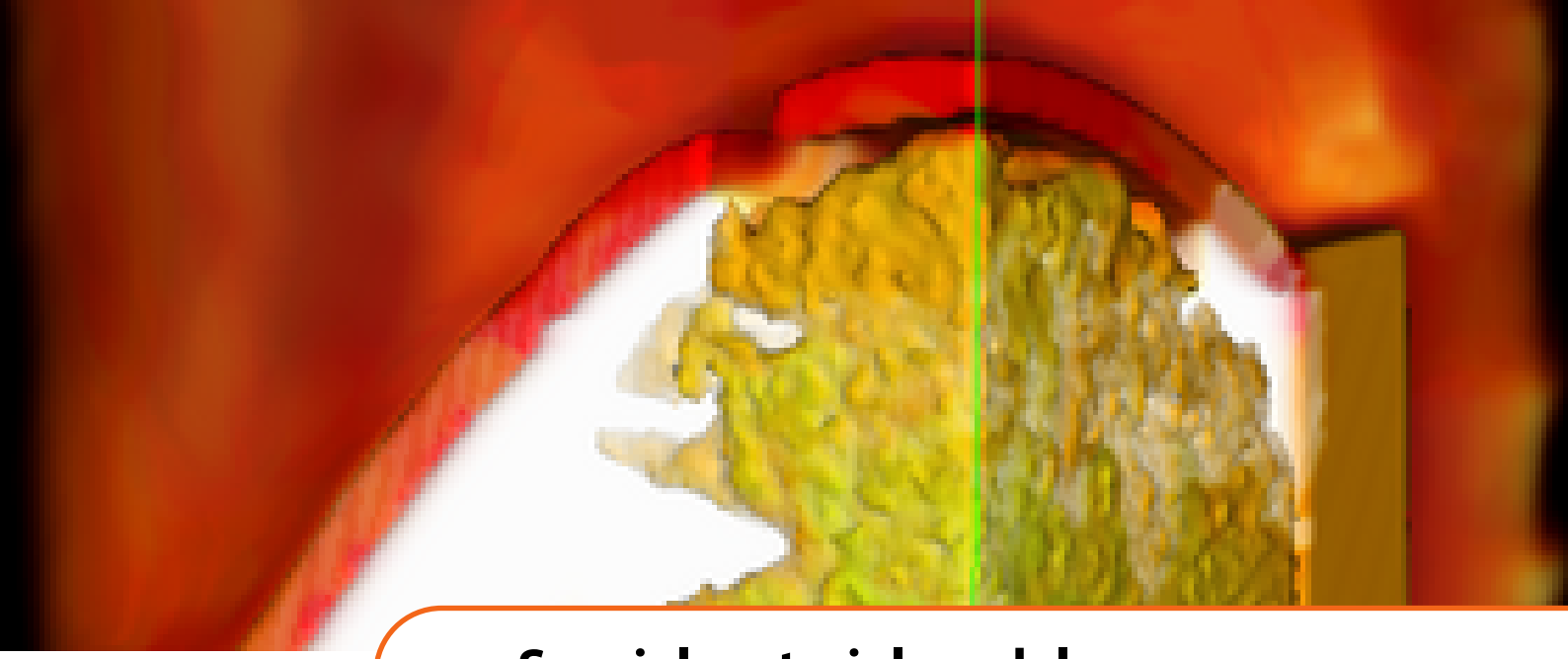
A small subset of these models also have implementations that can be used with Implicit MPM .

Some of these models can work with multiple sub-models such as elasticity model or yield condition. As of Vaango Version 20 .9 .18 , the implemented sub-models are:

1. Equations of state:
   (a) Pressure model for Air
   (b) Pressure model for Borja's CAMClay
   (c) Pressure model for Granite
   (d) Pressure model for hyperelastic materials
   (e) Pressure model for Hypoelastic materials
   (f) Pressure model for Mie-Gruneisen equation of state
   (g) Mie-Gruneisen energy-based equation of state for pressure
   (h) Pressure model for Water

2. Shear modulus models:
   (a) Constant shear modulus model
   (b) Shear modulus model for Borja's CAMClay
   (c) Shear modulus model by Nadal and LePoac
   (d) Mechanical Threshold Stress shear modulus model
   (e) Preston-Tonks-Wallace shear modulus model
   (f) Steinberg-Guinan shear modulus model
3. Combined elastic modulus models:
   (a) Constant elastic modulus model
   (b) Tabular elastic modulus model
   (c) Neural net elastic modulus model
   (d) Arena elastic modulus model
   (e) Arena mixture elastic modulus model
   (f) Arenisca elastic modulus model
4. Yield condition models:
   (a) Yield condition for Arena model
   (b) Yield condition for Arena mixture model
   (c) Yield condition for Arenisca3 model
   (d) Yield condition for CamClay model
   (e) Yield condition for Gurson model
   (f) Yield condition for Tabular plasticity with Cap
   (g) Yield condition for Tabular plasticity with
   (h) Yield condition for vonMises J_2 plasticity
   (i) Classic Mohr-Coulomb model
   (j) Sheng's Mohr-Coulomb model
5. Plastic flow stress models
   (a) Isotropic hardening plastic flow model
   (b) Johnson-Cook plastic flow model
   (c) Mechanical Threshold Stress plastic flow model
   (d) Preston-Tonks-Wallace plastic flow model
   (e) Steinberg-Guinan plastic flow model
   (f) SuvicI viscoplastic flow model
   (g) Zerilli-Armstrong metal plastic flow model
   (h) Zerilli-Armstrong polymer plastic flow model
6. Plastic internal variable models:
   (a) Arena internal variable model
   (b) Borja internal variable model
   (c) Brannan's soil model internal variable model
   (d) Tabular plasticity with cap internal variable model
7. Kinematic hardening models:
   (a) Prager kinematic hardening model
   (b) Armstrong-Frederick kinematic hardening model
   (c) Arena kinematic hardening model
8. Damage models
   (a) Becker's damage model
   (b) Drucker and Becker combined damage model
   (c) Drucker loss of stability model
   (d) Johnson-Cook damage model
   (e) Hancock-MacKenzie damage model
9. Melting model
   (a) Constant melting temperature model

   (b) Linear melting temperature model
   (c) BPS melting model
   (d) Steinberg-Guinan melting temperature model
10. Specific heat model
   (a) Constant specific heat
   (b) Cubic specific heat model
   (c) Copper specific heat model
   (d) Steel specific heat model



3 — Special material models

VAANGO contains a few material models that are designed for special problems. These are discussed in this chapter.

3.1 Rigid material

Applicable to: explicit and implicit MPM

This material model assumes that

$$\boldsymbol{\sigma}(\boldsymbol{F}, t) = \mathbf{o} \quad \text{and} \quad \boldsymbol{F}(t) = \boldsymbol{I}\,. \tag{3.1}$$

The model is a rough approximation of a rigid body as long as there is no contact between objects. Upon contact, the model should ideally transition into the form

$$\boldsymbol{\sigma}(\boldsymbol{F}, t) = \infty \quad \text{and} \quad \boldsymbol{F}(t) = \boldsymbol{I}\,. \tag{3.2}$$

This situation is approximated using the specified body contact algorithm which is applicable only in certain directions. Rigid materials were designed to act as rigid surfaces against which deformable objects could be compressed. The specified body contat algorithm can simulate the interaction of a single "master" rigid body with deformable objects.

3.2 Ideal gas material

Applicable to: explicit MPM only

The ideal gas material assumes that the stress at a particle is

$$\boldsymbol{\sigma}(\boldsymbol{F}, t) = \begin{cases} \bar{p}(\boldsymbol{F})\,\boldsymbol{I} & \text{for} \quad p \geq 0 \\ 0 & \text{for} \quad p < 0 \end{cases} \tag{3.3}$$

where $\bar{p} = -p$, and the pressure p is computed with an isentropic ideal gas equation of state:

$$p = p_{\text{ref}}\left[\exp(\gamma\,\overline{\varepsilon_v}) - 1\right] \;;\; \overline{\varepsilon_v} = \begin{cases} -\ln(J) & \text{for} \quad J < 1 \\ 0 & \text{for} \quad J \geq 1 \end{cases} \tag{3.4}$$

where $J = \det(\boldsymbol{F})$.

A rate of change of temperature (T) can also be computed by the model:

$$\frac{dT}{dt} = \frac{1}{\Delta t}\left(1 - \frac{J_{n+1}}{J_n}\right)\left(\frac{p}{\rho C_v}\right) \tag{3.5}$$

where $J_{n+1} = J(t_{n+1}, J_n = J(t_n)$, ρ is the mass density, and C_v is the constant volume specific heat.

3.3 Water material

Applicable to: explicit MPM only

This material models water [20], and assumes that the stress is given by

$$\boldsymbol{\sigma}(\boldsymbol{F}, t) = \bar{p}(\boldsymbol{F})\boldsymbol{I} + 2\mu\,\boldsymbol{\eta}. \tag{3.6}$$

where μ is a shear viscosity, $\boldsymbol{d}$ is the symmetric part of the velocity gradient,

$$\bar{p} = -p \quad \text{and} \quad \boldsymbol{\eta} = \boldsymbol{d} - \tfrac{1}{3}\mathrm{tr}(\boldsymbol{d})\boldsymbol{I}\,. \tag{3.7}$$

The pressure is given by:

$$p = \kappa\left[J^{-\gamma} - 1\right]\,, \quad J = \det(\boldsymbol{F}) \tag{3.8}$$

where κ the bulk modulus and γ is a model parameter. It has not been validated, but gives qualitatively reasonable behavior.

3.4 Murnaghan material

Applicable to: explicit MPM only

This material is based on the equation of state proposed in [21]. The stress is given by

$$\boldsymbol{\sigma}(\boldsymbol{F}, t) = \bar{p}(\boldsymbol{F})\boldsymbol{I} + 2\mu\,\boldsymbol{\eta}. \tag{3.9}$$

where μ is a shear viscosity, $\boldsymbol{d}$ is the symmetric part of the velocity gradient,

$$\bar{p} = -p \quad \text{and} \quad \boldsymbol{\eta} = \boldsymbol{d} - \tfrac{1}{3}\mathrm{tr}(\boldsymbol{d})\boldsymbol{I}\,. \tag{3.10}$$

The pressure is given by:

$$p = \frac{\kappa}{\kappa'}\left[J^{-\kappa'} - 1\right]\,, \quad J = \det(\boldsymbol{F}) \tag{3.11}$$

where κ the initial bulk modulus and $\kappa' = d\kappa/dp$ is a constant.

3.5 JWL++ material

Applicable to: explicit MPM only

The JWL++ material is a combination of the Murnaghan and JWL models along with a burn algorithm to convert from one to the other [22]. A small viscous component is added to the JWL model to stabilize behavior.

The stress is given by

$$\boldsymbol{\sigma}(\boldsymbol{F}, t) = \bar{p}(\boldsymbol{F})\boldsymbol{I} + 2\mu\,\boldsymbol{\eta}. \tag{3.12}$$

where μ is a shear viscosity, $\boldsymbol{d}$ is the symmetric part of the velocity gradient,

$$\bar{p} = -p \quad \text{and} \quad \boldsymbol{\eta} = \boldsymbol{d} - \tfrac{1}{3}\mathrm{tr}(\boldsymbol{d})\boldsymbol{I}\,. \tag{3.13}$$

The burn rate is computed as

$$\dot{f} = (1-f)G\,p^b \tag{3.14}$$

where f is the volume fraction of the reactant, G, b are fit parameters, and p is the pressure, computed using

$$p = (1-f)p_{\mathrm{m}} + f\,p_{\mathrm{jwl}}\,. \tag{3.15}$$

The Murnaghan pressure (p_{m}) is given by:

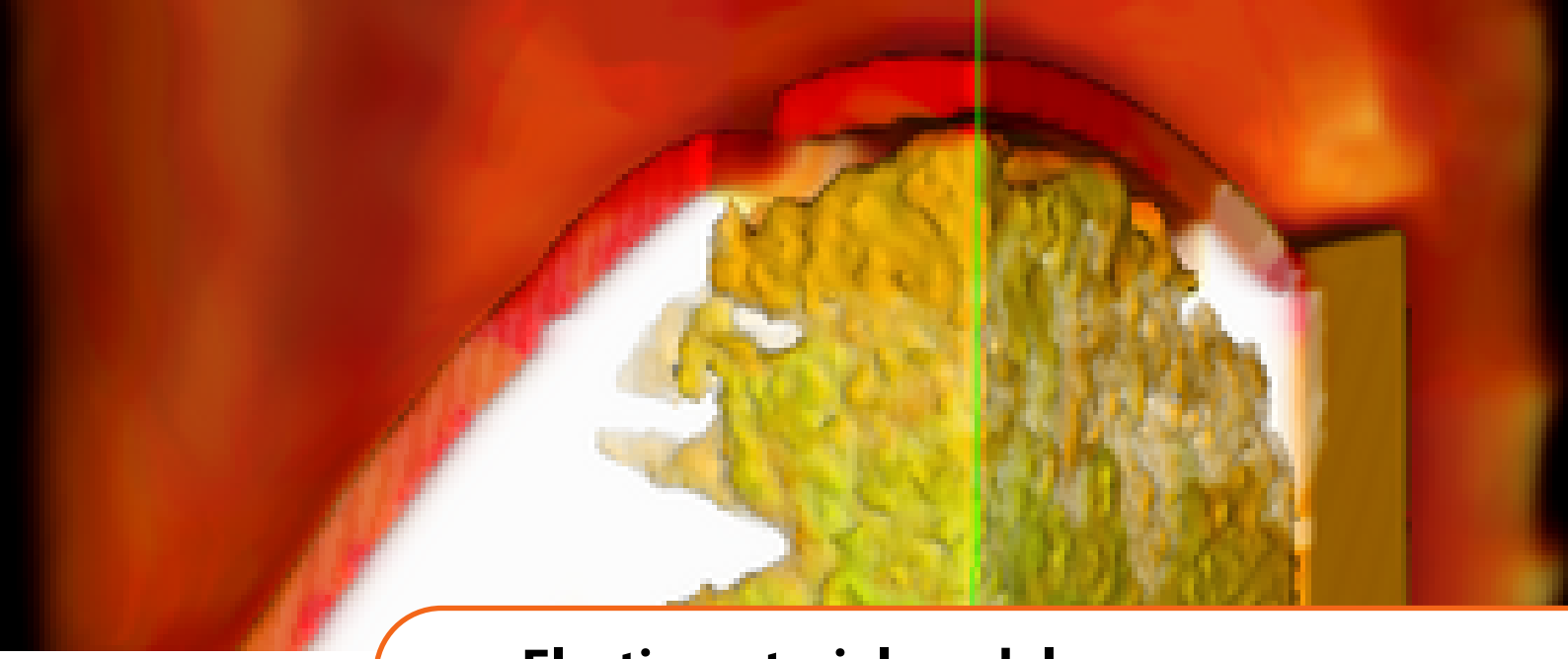
$$p_{\mathrm{m}} = \frac{1}{nK}\left[J^{-n} - 1\right]\,, \quad J = \det(\boldsymbol{F}) \tag{3.16}$$

where $K = 1/\kappa$, κ is the initial bulk modulus, and $n = \kappa' = d\kappa/dp$ is a constant.

The JWL pressure (p_{jwl}) is given by

$$p_{\mathrm{jwl}} = A\exp(-R_1 J) + B\exp(-R_2 J) + CJ^{-(1+\omega)} \tag{3.17}$$

where A, B, C, ω are fit parameters, R_1, R_2 are fit rate parameters, and $J = \det \boldsymbol{F}$.



4 — Elastic material models

4.1 Hypoelastic material

Applicable to: explicit and implicit MPM

Hypoelastic materials have stress-deformation relationships of the form

$$\dot{\boldsymbol{\sigma}}(\boldsymbol{F}) = \mathbb{C}(\boldsymbol{F}) : \boldsymbol{d}(\boldsymbol{F}) \tag{4.1}$$

where $\mathbb{C}$ is an elastic stiffness tensor and $\boldsymbol{d}$ is the symmetric part of the velocity gradient.

The base hypoelastic material implemented in Vaango is linear and isotropic:

$$\dot{\boldsymbol{\sigma}} = \left(\kappa - \tfrac{2}{3}\mu\right) \operatorname{tr}(\boldsymbol{d})\, \boldsymbol{I} + 2\mu \boldsymbol{d} \tag{4.2}$$

where μ is the shear modulus and κ is the bulk modulus.

To ensure frame indifference, both $\boldsymbol{\sigma}$ and $\boldsymbol{d}$ are unrotated using the beginning of the timestep deformation gradient polar decomposition before any constitutive relations are evaluated. The updated stress is rotated back using the deformation gradient decomposition at the end of the time step.

4.2 Hyperelastic Material Models

Several hyperelastic material models have been implemented in Vaango. Other models can be easily implemented using the available infrastructure. The general model has the form

$$\boldsymbol{\sigma} = \frac{1}{J} \frac{\partial W}{\partial \boldsymbol{F}} \cdot \boldsymbol{F}^T \tag{4.3}$$

where W is a strain energy function and $J = \det \boldsymbol{F}$. For isotropic hyperelastic functions that are expessed in terms of the invariants (I_1, I_2, J) of the right Cauchy-Green deformation ($\boldsymbol{C} = \boldsymbol{F}^T \cdot \boldsymbol{F}$), the Cauchy stress is given by

$$\boldsymbol{\sigma} = \frac{2}{J}\left[\frac{1}{J^{2/3}}\left(\frac{\partial W}{\partial \bar{I}_1} + \bar{I}_1 \frac{\partial W}{\partial \bar{I}_2}\right)\boldsymbol{B} - \frac{1}{J^{4/3}} \frac{\partial W}{\partial \bar{I}_2}\, \boldsymbol{B}\cdot\boldsymbol{B}\right] + \left[\frac{\partial W}{\partial J} - \frac{2}{3J}\left(\bar{I}_1 \frac{\partial W}{\partial \bar{I}_1} + 2\,\bar{I}_2 \frac{\partial W}{\partial \bar{I}_2}\right)\right]\boldsymbol{I} \tag{4.4}$$

where $\boldsymbol{B} = \boldsymbol{F} \cdot \boldsymbol{F}^T$, and

$$J = \det \boldsymbol{F}\,, \quad \bar{I}_1 = J^{-2/3} I_1\,, \quad \bar{I}_2 = J^{-4/3} I_2\,, \quad I_1 = \operatorname{tr} \boldsymbol{C}\,, \quad I_2 = \tfrac{1}{2}\left[(\operatorname{tr} \boldsymbol{C})^2 - \operatorname{tr}(\boldsymbol{C}\cdot\boldsymbol{C})\right] \tag{4.5}$$

Note that I_1 and I_2 are identical for $\boldsymbol{C}$ and $\boldsymbol{B}$. Alternatively,

$$\boldsymbol{\sigma} = \frac{2}{J}\left[\left(\frac{\partial W}{\partial I_1} + I_1\frac{\partial W}{\partial I_2}\right)\boldsymbol{B} - \frac{\partial W}{\partial I_2}\boldsymbol{B}\cdot\boldsymbol{B}\right] + 2J\frac{\partial W}{\partial I_3}\boldsymbol{I} \tag{4.6}$$

where $I_3 = J^2$.

The P-wave speed (c) needed to estimate the timestep can be computed using

$$c_i^2 = \frac{1}{\rho J}\frac{\partial^2 W}{\partial \lambda_i^2} = \frac{1}{\rho J}\left[\frac{\partial W}{\partial I_1}\frac{\partial^2 I_1}{\partial \lambda_i^2} + \frac{\partial W}{\partial I_2}\frac{\partial^2 I_2}{\partial \lambda_i^2} + \frac{\partial W}{\partial I_3}\frac{\partial^2 I_3}{\partial \lambda_i^2}\right] \tag{4.7}$$

where λ_i are the principal stretches, i.e., $I_1 = \sum_i \lambda_i^2$, $I_2 = \lambda_1^2\lambda_2^2 + \lambda_2^2\lambda_3^2 + \lambda_1^2\lambda_3^2$ and $I_3 = \lambda_1^2\lambda_2^2\lambda_3^2$.

4.2.1 Compressible neo-Hookean material

Applicable to: explicit and implicit MPM

The default strain energy function for the compressible neo-Hookean material model implemented in Vaango is ([23], p.307):

$$W = \frac{\kappa}{2}\left[\frac{1}{2}(J^2 - 1) - \ln J\right] + \frac{\mu}{2}\left[\bar{I}_1 - 3\right] \tag{4.8}$$

The Cauchy stress corresponding to this function is

$$\boldsymbol{\sigma} = \frac{\kappa}{2}\left(J - \frac{1}{J}\right)\boldsymbol{I} + \frac{\mu}{J}(\overline{\boldsymbol{B}} - \tfrac{1}{3}\bar{I}_1\boldsymbol{I}) \tag{4.9}$$

where $\overline{\boldsymbol{B}} = J^{-2/3}\boldsymbol{B} = J^{-2/3}\boldsymbol{F}\cdot\boldsymbol{F}^T$ and $J = \det \boldsymbol{F}$. Consistency with linear elasticity requires that $\kappa = K$ and $\mu = G$ where K and G are the linear elastic bulk and shear moduli, respectively.

Alternative expressions for the bulk modulus factor are allowed and defined in the equation-of-state sub-models.

4.2.2 Compressible Mooney-Rivlin material

Applicable to: explicit MPM only

The compressible Mooney-Rivlin material implemented in Vaango has the form

$$W = C_1(I_1 - 3) + C_2(I_2 - 3) + C_3\left(\frac{1}{I_3^2} - 1\right) + C_4(I_3 - 1)^2 \tag{4.10}$$

where C_1, C_2 and ν are parameters and

$$C_3 = \tfrac{1}{2}(C_1 + 2C_2)\,, \quad C_4 = \tfrac{1}{2}\left[\frac{C_1(5\nu - 2) + C_2(11\nu - 5)}{1 - 2\nu}\right]. \tag{4.11}$$

The corresponding Cauchy stress is

$$\boldsymbol{\sigma} = \frac{2}{J}\left[(C_1 + C_2 I_1)\boldsymbol{B} - C_2\boldsymbol{B}\cdot\boldsymbol{B} + J^2\left[-\frac{2C_3}{I_3^3} + 2C_4(I_3 - 1)\right]\right]. \tag{4.12}$$

4.2.3 Transversely isotropic hyperelastic material

Applicable to: explicit and implicit MPM

The transversely isotropic material model implemented in VAANGO is based on [24]. The model asssumes a stiffer, "fiber", direction denoted $\hat{\mathbf{f}}$ and isotropy ortogonal to that direction.

The strain energy density function for the model has the form

$$W = W_v + W_d \tag{4.13}$$

where W_v is the volumetric part and W_d is the deviatoric (volume preserving) part. The volumetric part of the strain energy is given by

$$W_v = \tfrac{1}{2}\kappa(\ln J)^2 \tag{4.14}$$

where κ is the bulk modulus and $J = \det \boldsymbol{F}$. The deviatoric part, W_d, is given by

$$W_d = \begin{cases} C_1(\overline{I}_1 - 3) + C_2(\overline{I}_2 - 3) + C_3\left[\exp\left(C_4(\bar{\lambda} - 1)\right) - 1\right] & \text{for} \quad \bar{\lambda} < \lambda^\star \\ C_1(\overline{I}_1 - 3) + C_2(\overline{I}_2 - 3) + C_5\bar{\lambda} + C_6 \ln \bar{\lambda} & \text{for} \quad \bar{\lambda} \geq \lambda^\star \end{cases} \tag{4.15}$$

where C_1, C_2, C_3, C_4, C_5, $\lambda^\star$ are model parameters, and

$$\begin{aligned} C_6 &= C_3\left[\exp\left(C_4(\lambda^\star - 1)\right) - 1\right] - C_5\lambda^\star \\ \bar{\lambda} &= \sqrt{\overline{I}_4}\,, \quad \overline{I}_4 = \hat{\mathbf{f}} \cdot (\overline{\boldsymbol{C}} \cdot \hat{\mathbf{f}})\,, \quad \overline{\boldsymbol{C}} = J^{-2/3}\boldsymbol{C}\,. \end{aligned} \tag{4.16}$$

The fiber direction is updated using

$$\hat{\mathbf{f}}_{n+1} = \frac{J^{-1/3}}{\bar{\lambda}}\,\boldsymbol{F} \cdot \hat{\mathbf{f}}_n\,. \tag{4.17}$$

The Cauchy stress is given by

$$\boldsymbol{\sigma} = p\boldsymbol{I} + \boldsymbol{\sigma}_d + \boldsymbol{\sigma}_f \tag{4.18}$$

where

$$\begin{aligned} p &= \kappa\frac{\ln(J)}{J} \\ \boldsymbol{\sigma}_d &= \frac{2}{J}\left[(C_1 + C_2\overline{I}_1)\overline{\boldsymbol{B}} - C_2\overline{\boldsymbol{B}} \cdot \overline{\boldsymbol{B}} - \tfrac{1}{3}(C_1\overline{I}_1 + 2C_2\overline{I}_2)\boldsymbol{I}\right] \\ \boldsymbol{\sigma}_f &= \frac{\bar{\lambda}}{J}\frac{\partial W_d}{\partial \bar{\lambda}}\left(\hat{\mathbf{f}}_{n+1} \otimes \hat{\mathbf{f}}_{n+1} - \tfrac{1}{3}\boldsymbol{I}\right) \end{aligned} \tag{4.19}$$

The model also contains a failure feature that sets $\boldsymbol{\sigma}_d = 0$ when the maximum shear strain, defined as the difference between the maximum and mimum eigenvalues of $\boldsymbol{C}$, exceeds a critical shear strain value. Also, a fiber stretch failure criterion can be used that compares $\sqrt{I_4}$ with a critical stretch value and sets $\boldsymbol{\sigma}_f = 0$ is this value is exceeded.

4.3 Elastic modulus models

Applicable to: Hypoelastic Tabular material models

For selected material models that use isotropic hypoelasticity models and require bulk and shear moduli, specialized elastic moduli models can be used. Some of these models are discussed in this section.

4.3.1 Support vector regression model

The support vector regression (SVR) approach [25, 26] can be used to fit bulk modulus models to data without the need for closed form expressions. The advantage of this approach is that the resulting model requires few function evaluations and can, in principle, be computed as fast as a closed-form model.

For the purpose of fitting a bulk modulus model we assume that the input (training) data are of the form $\{(\boldsymbol{\varepsilon}_1, p_1), (\boldsymbol{\varepsilon}_2, p_2), \ldots, (\boldsymbol{\varepsilon}_m, p_m)\} \subset \mathbb{R}^2 \times \mathbb{R}$. Here $\boldsymbol{\varepsilon}_i = (\varepsilon_i, \varepsilon_i^p)$ where ε is the total volumetric strain and ε^p is the plastic volumetric strain and p_i is the mean stress (assumed positive in compression). The aim of SVR is to find a function $p = f(\boldsymbol{\varepsilon})$ that fits the data such that the function is as flat as possible (in $d+1$-dimensional space), and deviates from p_i by at most ϵ (a small quantity).

In nonlinear support vector regression we fit functions of the form

$$p = f(\boldsymbol{\varepsilon}) = \mathbf{w} \cdot \boldsymbol{\phi}(\boldsymbol{\varepsilon}) + b \tag{4.20}$$

where $\mathbf{w}$ is a vector of parameters, $\boldsymbol{\phi}(\boldsymbol{\varepsilon})$ are vector-valued basis functions, $(\cdot)$ is an inner product, and b is a scalar offset. The fitting process can be posed as the following primal convex optimization problem [27]:

$$\begin{aligned} &\underset{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^\star}{\text{minimize}} \quad \tfrac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{i=1}^{m}(\xi_i + \xi_i^\star) \\ &\text{subject to} \quad \begin{cases} -(\xi_i + \epsilon) \le p_i - \mathbf{w}\cdot\boldsymbol{\phi}(\boldsymbol{\varepsilon}_i) - b \le \xi_i^\star + \epsilon \\ \xi_i, \xi_i^\star \ge 0\,, \quad i = 1\ldots m \end{cases} \end{aligned} \tag{4.21}$$

where C is a constraint multiplier, m is the number of data points, and $\xi_i, \xi_i^\star$ are constraints.

In practice, it is easier to solve the dual problem for which the expansion for $f(\boldsymbol{\varepsilon})$ becomes

$$p = f(\boldsymbol{\varepsilon}) = \sum_{i=1}^{m}(\lambda_i^\star - \lambda_i)K(\boldsymbol{\varepsilon}_i, \boldsymbol{\varepsilon}) + b\,, \quad K(\boldsymbol{\varepsilon}_i, \boldsymbol{\varepsilon}) = \boldsymbol{\phi}(\boldsymbol{\varepsilon}_i)\cdot\boldsymbol{\phi}(\boldsymbol{\varepsilon}) \tag{4.22}$$

where $\boldsymbol{\varepsilon}_i$ are the sample vectors, λ_i and $\lambda_i^\star$ are dual coefficients, and $K(\boldsymbol{\varepsilon}_i, \boldsymbol{\varepsilon})$ is a kernel function. The dual convex optimization problem has the form

$$\begin{aligned} &\underset{\boldsymbol{\lambda},\boldsymbol{\lambda}^\star}{\text{minimize}} \quad \tfrac{1}{2}\sum_{i,j=1}^{m}(\lambda_i - \lambda_i^\star)K(\boldsymbol{\varepsilon}_i, \boldsymbol{\varepsilon}_j)(\lambda_j - \lambda_j^\star) + \epsilon\sum_{i=1}^{m}(\lambda_i + \lambda_i^\star) + \sum_{i=1}^{m} p_i(\lambda_i - \lambda_i^\star) \\ &\text{subject to} \quad \begin{cases} \sum_{i=1}^{m}(\lambda_i - \lambda_i^\star) = 0 \\ \lambda_i, \lambda_i^\star \in [0, C]\,, \quad i = 1\ldots m\,. \end{cases} \end{aligned} \tag{4.23}$$

The free parameters for the fitting process are the quantities ϵ and C. SVR accuracy also depends strongly on the choice of kernel function. In this paper, we use the Gaussian radial basis function:

$$K(\boldsymbol{\varepsilon}_i, \boldsymbol{\varepsilon}_j) = \exp\left[-\frac{(\boldsymbol{\varepsilon}_i - \boldsymbol{\varepsilon}_j)\cdot(\boldsymbol{\varepsilon}_i - \boldsymbol{\varepsilon}_j)}{\sigma^2\, d}\right] = \exp\left[-\gamma\left\|\boldsymbol{\varepsilon}_i - \boldsymbol{\varepsilon}_j\right\|^2\right]\,, \quad \gamma := \frac{1}{\sigma^2\, d} \tag{4.24}$$

where d is the dimension of $\boldsymbol{\varepsilon}$ and σ^2 is the width of the support of the kernel (assumed to be equal to the norm of the covariance matrix of the training data in this paper).

The minimization problem solves for the difference in the dual coefficients $(\boldsymbol{\lambda} - \boldsymbol{\lambda}^\star)$ and the intercept (b), and outputs a reduced set $(m_{\text{SV}} < m)$ of values of $\boldsymbol{\varepsilon}_i$ called "support vectors". Given these quantities, the function (4.22) can be evaluated quite efficiently, particularly if the number of support vectors is small. SVR fits to data can be computed using software such as the LIBSVM library [28]. A variation of the above approach, called ν-SVR [25] can also be used if sufficient computational resources are available.

The bulk modulus can be computed from (4.22) using

$$\kappa(\boldsymbol{\varepsilon}) = \frac{\partial p}{\partial \varepsilon^e} = \frac{\partial p}{\partial \varepsilon} = \sum_{i=1}^{m_{\mathrm{SV}}} (\lambda_i^\star - \lambda_i) \frac{\partial K_i}{\partial \varepsilon} \,, \quad K_i := K(\boldsymbol{\varepsilon}_i, \boldsymbol{\varepsilon}) \tag{4.25}$$

From (4.24),

$$\frac{\partial K_i}{\partial \varepsilon} = 2\gamma(\varepsilon_i - \varepsilon) \exp\left[-\gamma \left\|\boldsymbol{\varepsilon}_i - \boldsymbol{\varepsilon}\right\|^2\right] \tag{4.26}$$

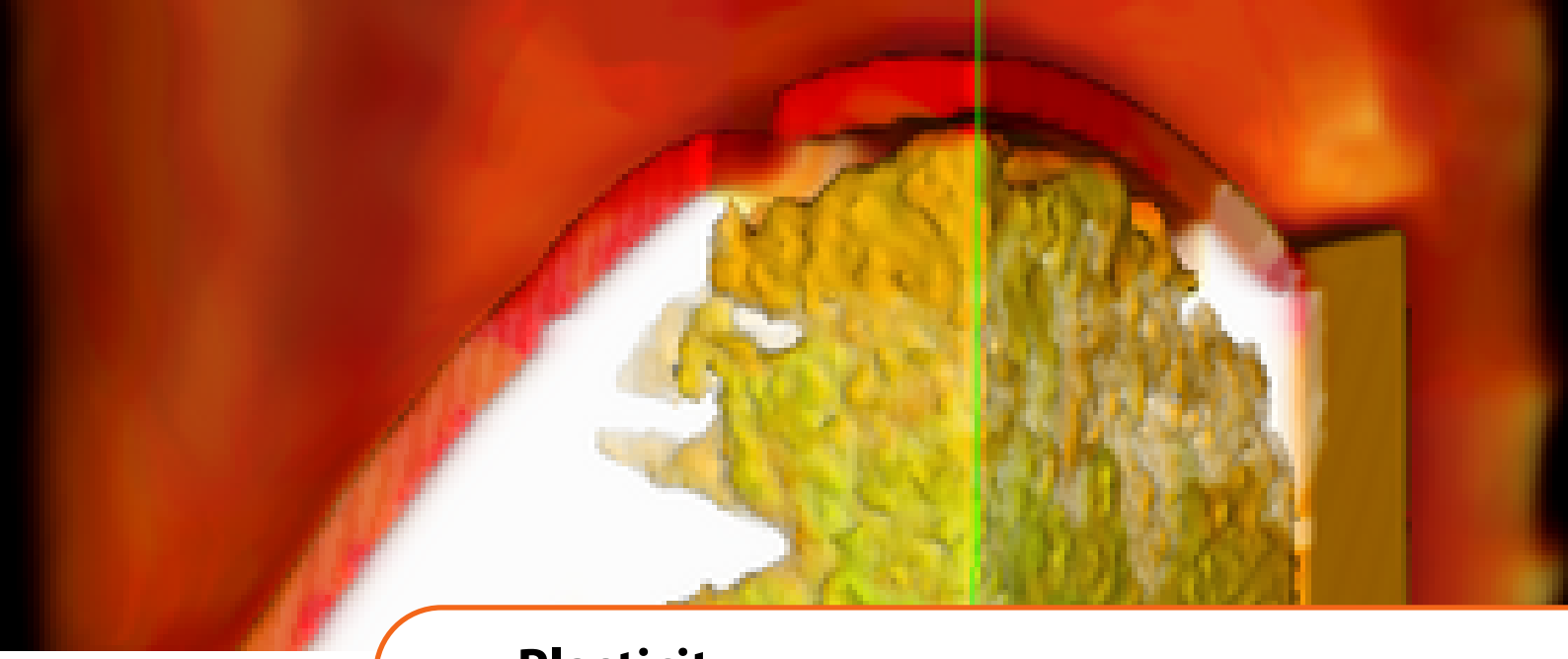
Therefore, the bulk modulus is given by,

$$\kappa(\boldsymbol{\varepsilon}) = \sum_{i=1}^{m_{\mathrm{SV}}} 2\gamma(\lambda_i^\star - \lambda_i)(\varepsilon_i - \varepsilon) \exp\left[-\gamma \left\|\boldsymbol{\varepsilon}_i - \boldsymbol{\varepsilon}\right\|^2\right] \tag{4.27}$$

If we need to account for elastic-plastic coupling, we may also need the derivative

$$\frac{\partial \kappa}{\partial \varepsilon^p} = \sum_{i=1}^{m_{\mathrm{SV}}} 4\gamma^2(\lambda_i^\star - \lambda_i)(\varepsilon_i - \varepsilon)(\varepsilon_i^p - \varepsilon^p) \exp\left[-\gamma \left\|\boldsymbol{\varepsilon}_i - \boldsymbol{\varepsilon}\right\|^2\right] \tag{4.28}$$

The bulk modulus model is also associated with a shear modulus model that computes the value of μ using a Poisson's ratio (ν) based on the value of κ. The VAANGO implementation can be accessed in the tabular plasticity models, using the tag <elastic_moduli_model type="support_vector"> .



5 — Plasticity

Most plasticity models in VAANGO are implemented as stand-alone models with their own elasticity law, yield condition, flow rule, and internal variable evolution rules. However, because of the large number of possible combinations of these, a few metal plasticity models are available that allow the user to swap out one set of rules for another.

> The list below is not comprehensive. Please see the following chapters for details of the models actually available in VAANGO.

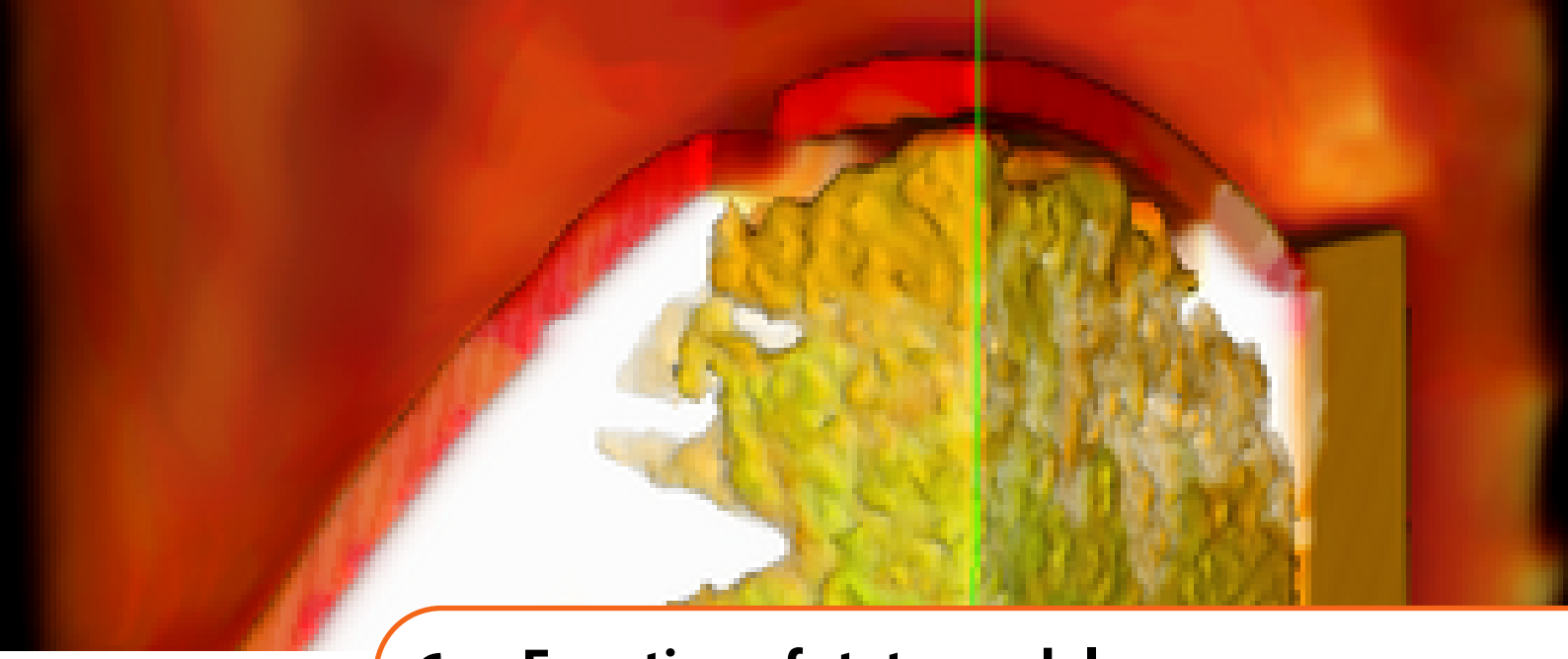
The plasticity implementations typically contain the following:

1. An elasticity model:
   - Isotropic linear elastic model.
   - Anisotropic linear elastic models.
   - Isotropic nonlinear elastic models.
   - Anisotropic nonlinear elastic models.
   - Equation of state to determine the pressure (or volumetric response), for example,
     - Mie-Gruneisen equation of state.
   - Deviatoric stress model to determine the shear response.
     - Nadal-LePoac shear modulus model
     - Steinberg-Guinan shear modulus model
2. A yield condition:
   - von Mises yield condition.
   - Drucker-Prager yield condition.
   - Mohr-Coulomb yield condition.
   - Gurson-Needleman-Tvergaard yield condition.
3. A flow rule:
   - Associated flow.
   - Non-Associated flow using either a flow potential or a material parameter.
4. Isotropic hardening:
   - Perfect plasticity (no hardening).
   - Johnson-Cook plasticity.
   - Mechanical Threshold Stress (MTS) plasticity.
5. Kinematic hardening:
   - Isotropic backstress.

- Deviatoric backstress.

6. Isotropic hardening internal variable evolution rules :
   - Mechanical threshold stress evolution.
   - Gurson's porosity evolution law.
7. Kinematic hardening internal variable evolution rules :
   - Ziegler-Prager evolution rule.
   - Armstrong-Frederick evolution rule
8. Damage evolution rules :
   - Johnson-Cook damage model.
   - Brittle damage model.
9. Melting temperature models
10. Specific heat models
11. Material stability-based localization models :
    - Acoustic tensor.
    - Drucker stability.

The models used by the main plasticity codes in the current implementation of VAANGO are described in the following chapters, followed by descriptions of the plasticity algorithms themselves.



6 — Equation of state models

In the isotropic metal plasticity models implemented in VAANGO , the volumetric part of the Cauchy stress can be calculated using an equation of state. The equations of state that are implemented in VAANGO are described below.

6.1 Hypoelastic equation of state

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

$$\dot{\boldsymbol{\sigma}} = \lambda \, \mathrm{tr}(\boldsymbol{d}^e) \, \boldsymbol{I} + 2 \, \mu \, \boldsymbol{d}^e \tag{6.1}$$

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

If $\mathrm{dev}(\boldsymbol{d}^e)$ is the deviatoric part of $\boldsymbol{d}^e$ then we can write

$$\dot{\boldsymbol{\sigma}} = \left(\lambda + \frac{2}{3}\,\mu\right) \mathrm{tr}(\boldsymbol{d}^e) \, \boldsymbol{I} + 2 \, \mu \, \mathrm{dev}(\boldsymbol{d}^e) = \kappa \, \mathrm{tr}(\boldsymbol{d}^e) \, \boldsymbol{I} + 2 \, \mu \, \mathrm{dev}(\boldsymbol{d}^e) \, . \tag{6.2}$$

If we split $\boldsymbol{\sigma}$ into a volumetric and a deviatoric part, i.e., $\boldsymbol{\sigma} = p \, \boldsymbol{I} + \mathbf{s}$, take the time derivative to get $\dot{\boldsymbol{\sigma}} = \dot{p} \, \boldsymbol{I} + \dot{\mathbf{s}}$, and compare the result with (6.2), we see that

$$\dot{p} = \kappa \, \mathrm{tr}(\boldsymbol{d}^e) \, . \tag{6.3}$$

In addition we assume that $\boldsymbol{d} = \boldsymbol{d}^e + \boldsymbol{d}^p$. If we also assume that the plastic volume change is negligible ($\mathrm{tr}(\boldsymbol{d}^p) \approx 0$), which is reasonable for a void-free metal matrix, we have

$$\dot{p} = \kappa \, \mathrm{tr}(\boldsymbol{d}) \, . \tag{6.4}$$

This is the equation that is used to calculate the pressure p in the default hypoelastic equation of state. For a forward Euler integration step,

$$\boxed{p_{n+1} = p_n + \kappa \, \mathrm{tr}(\boldsymbol{d}_{n+1}) \, \Delta t \, .} \tag{6.5}$$

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

$$\dot{p} = \frac{\partial p}{\partial J} \, \dot{J} = \frac{\partial p}{\partial J} \, J \, \mathrm{tr}(\boldsymbol{d}) \, . \tag{6.6}$$

Therefore,

$$\boxed{\frac{\partial p}{\partial J} = \frac{\kappa}{J}.} \tag{6.7}$$

This model is invoked in VAANGO using

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

6.2 Default hyperelastic equation of state

In this model the pressure is computed using the relation

$$p = \tfrac{1}{2}\,\kappa\left(J^e - \frac{1}{J^e}\right) \tag{6.8}$$

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

We can also compute

$$\frac{dp}{dJ} = \tfrac{1}{2}\,\kappa\left(1 + \frac{1}{(J^e)^2}\right). \tag{6.9}$$

The metal plasticity implementations in VAANGO assume that the volume change of the matrix during plastic deformation can be neglected, i.e., $J^e = J$.

This model is invoked using

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

6.3 Mie-Gruneisen equation of state

The pressure (p) is calculated using a Mie-Grüneisen equation of state of the form ([14, 29])

$$p = -\frac{\rho_o\, C_o^2\,(1-J^e)[1-\Gamma_o(1-J^e)/2]}{[1-S_\alpha(1-J^e)]^2} - \Gamma_o\, E\,; \quad J^e := \det \boldsymbol{F}^e \tag{6.10}$$

where C_o is the bulk speed of sound, ρ_o is the initial mass density, Γ_o 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,

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

The internal energy is computed using

$$E = \frac{1}{V_o}\int C_v\, dT \approx \frac{C_v(T - T_o)}{V_o} \tag{6.11}$$

where $V_o = 1/\rho_o$ is the reference specific volume at temperature $T = T_o$, and C_v is the specific heat at constant volume.

Also,

$$\boxed{\frac{\partial p}{\partial J^e} = \frac{\rho_o\, C_o^2\, [1 + (S_\alpha - \Gamma_o)\,(1 - J^e)]}{[1 - S_\alpha\,(1 - J^e)]^3} - \Gamma_o\, \frac{\partial E}{\partial J^e}.} \tag{6.12}$$

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>
  <rho_0> 7200 </rho_0>
</equation_of_state>
```

An alternative formulation is also available that can be used for models where a linear Hugoniot is not accurate enough. A cubic model can be used in that formulation.

$$p_{n+1} = -\frac{\rho_o\, C_o^2\, (1 - J^e_{n+1})[1 - \Gamma_o(1 - J^e_{n+1})/2]}{[1 - S_\alpha(1 - J^e_{n+1}) - S_2(1 - J^e_{n+1})^2 - S_3(1 - J^e_{n+1})^3]^2} - \Gamma_o\, e_{n+1}\,; \quad J^e := \det \boldsymbol{F}^e \tag{6.13}$$

This model is invoked using the label mie_gruneisen_energy .

6.4 Equations of state used in the ARENA model

In many models, a tangent bulk modulus is computed using the equation of state and the pressure is updated using an integration step. While this approach less accurate than directly evaluating the equation of state, it is useful when a composite material is being simulated that does not have well-characterized equations of state at all states.

The equations of state used by the ARENA model for soils are described below. The bars above quantities indicate negation.

6.4.1 Solid matrix material

The pressure in the solid matrix is expressed as

$$\bar{p}_s = K_s \bar{\varepsilon}^s_v\,; \quad \bar{\varepsilon}^s_v := \ln\left(\frac{V_{so}}{V_s}\right) \tag{6.14}$$

where $\bar{p}_s = -p_s$ is the solid matrix pressure, K_s is the solid bulk modulus, $\bar{\varepsilon}^s_v$ 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:

$$K_s(\bar{p}_s) = K_{so} + n_s\,(\bar{p}_s - \bar{p}_{so}) \tag{6.15}$$

where K_{so} and n_s are material properties, and $\bar{p}_{so}$ is a reference pressure.

6.4.2 Pore water

The equation of state of the pore water is

$$\bar{p}_w = K_w \bar{\varepsilon}^w_v + \bar{p}_o\,; \quad \bar{\varepsilon}^w_v := \ln\left(\frac{V_{wo}}{V_w}\right) \tag{6.16}$$

where $\bar{p}_w = -p_w$ 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:

$$K_w(\bar{p}_w) = K_{wo} + n_w\,(\bar{p}_w - \bar{p}_{wo}) \tag{6.17}$$

where K_{wo} and n_w are material properties, and $\bar{p}_{wo}$ is a reference pressure.

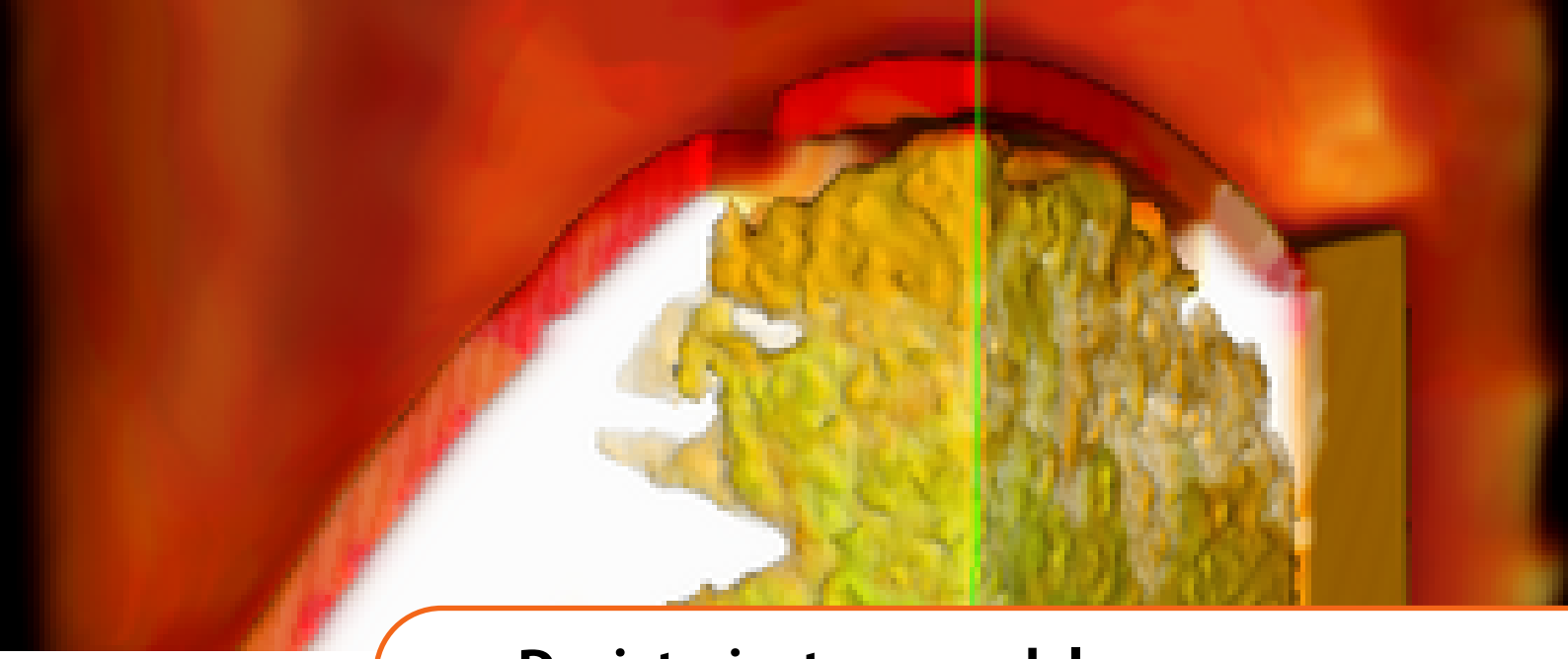
6.4.3 Pore air

The isentropic ideal gas equation of state for the pore air is

$$\bar{p}_a = \bar{p}_r\left[\exp(\gamma\,\overline{\varepsilon_v^a}) - 1\right]\;;\;\; \overline{\varepsilon_v^a} := \ln\left(\frac{V_{ao}}{V_a}\right) \tag{6.18}$$

where the quantities with subscript a represent quantities for the air model analogous to those for the water model in (20.10), $\bar{p}_r$ 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:

$$K_a = \frac{d\bar{p}_a}{d\overline{\varepsilon_v^a}} = \gamma\,\bar{p}_r\,\exp(\gamma\,\overline{\varepsilon_v^a}) = \gamma\,(\bar{p}_a + \bar{p}_r)\,. \tag{6.19}$$



7 — Deviatoric stress models

Isotropic plasticity models in VAANGO typically assume hypoelasticity, for which the stress rate is given by

$$\dot{\boldsymbol{\sigma}} = \dot{p}\,\boldsymbol{I} + \dot{\mathbf{s}} = \kappa\,\mathrm{tr}(\boldsymbol{d}^e)\,\boldsymbol{I} + 2\,\mu\,\mathrm{dev}(\boldsymbol{d}^e) \tag{7.1}$$

where $\boldsymbol{\sigma} = p\boldsymbol{I} + \mathbf{s}$ is the Cauchy stress, $p = \mathrm{tr}(\boldsymbol{\sigma})$, $\mathbf{s}$ is the deviatoric stress, $\boldsymbol{d}^e$ is the elastic part of the rate of deformation, and κ, μ are the bulk and shear moduli.

The pressure is computed using an equation of state as described in the chapter 6. The deviatoric stress is computed using the relation

$$\dot{\mathbf{s}} = 2\,\mu\,\mathrm{dev}(\boldsymbol{d}^e)\,. \tag{7.2}$$

If a forward Euler stress update is used, we have

$$\mathbf{s}_{n+1} = \mathbf{s}_n + 2\mu\mathrm{dev}(\boldsymbol{d}^e_{n+1})\Delta t\,. \tag{7.3}$$

For linear elastic materials, the shear modulus can vary with temperature and pressure. Several shear modulus models area available in VAANGO for computing the value for a given state.

For linear viscoelastic materials to be used with plasticity, a Maxwell model is available in VAANGO where the deviatoric stress is computed as a sum of Maxwell elements:

$$\mathbf{s}_{n+1} = \mathbf{s}_n + 2\sum_j \mu_j \mathrm{dev}(\boldsymbol{d}^e_{n+1})\Delta t\,. \tag{7.4}$$

7.1 Shear modulus models

Shear modulus models that are available in VAANGO are described below.

7.1.1 Constant shear modulus

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

```
<shear_modulus_model type="constant_shear">
  <shear_modulus> 1.0e8 </shear_modulus>
</shear_modulus_model>
```

7.1.2 Mechanical Threshold Stress shear modulus

The simplest model is of the form suggested by [30] ([31])

$$\mu(T) = \mu_o - \frac{D}{exp(T_o/T) - 1} \tag{7.5}$$

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>
```

7.1.3 SCG shear modulus

The Steinberg-Cochran-Guinan (SCG) shear modulus model ([29, 32]) is pressure dependent and has the form

$$\mu(p, T) = \mu_o + \frac{\partial \mu}{\partial p}\frac{p}{\eta^{1/3}} + \frac{\partial \mu}{\partial T}(T - 300); \quad \eta = \rho/\rho_o \tag{7.6}$$

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>
```

7.1.4 Nadal-LePoac (NP) shear modulus

A modified version of the SCG model has been developed by [33] 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

$$\mu(p, T) = \frac{1}{\mathcal{J}(\hat{T})}\left[\left(\mu_o + \frac{\partial \mu}{\partial p}\frac{p}{\eta^{1/3}}\right)(1 - \hat{T}) + \frac{\rho}{Cm}\, k_b\, T\right]; \quad C := \frac{(6\pi^2)^{2/3}}{3} f^2 \tag{7.7}$$

where

$$\mathcal{J}(\hat{T}) := 1 + \exp\left[-\frac{1 + 1/\zeta}{1 + \zeta/(1 - \hat{T})}\right] \quad \text{for} \quad \hat{T} := \frac{T}{T_m} \in [0, 1 + \zeta], \tag{7.8}$$

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>
```

7.1.5 Preston-Tonks-Wallace (PTW) shear modulus

The PTW shear model [34] is a simplified version of the SCG shear model. This model suggests computing the shear modulus using

$$\mu(p, T) = \mu_o \left(1 + \beta \frac{\bar{p}}{\eta^{1/3}}\right)\left(1 - \alpha_p \frac{T}{T_m}\right) \tag{7.9}$$

where μ_o is the shear modulus at room temperature and pressure, $\bar{p} = -p$, α_p is a material parameter, T_m is the melting temperature, and

$$\eta = \frac{\rho}{\rho_o}, \quad \beta = \frac{\frac{d\mu}{dp}}{\mu_o}. \tag{7.10}$$

7.1.6 Borja's shear modulus model

Borja's deviatoric stress model [35] assumes that the deviatoric part of the elastic strain energy density has the form

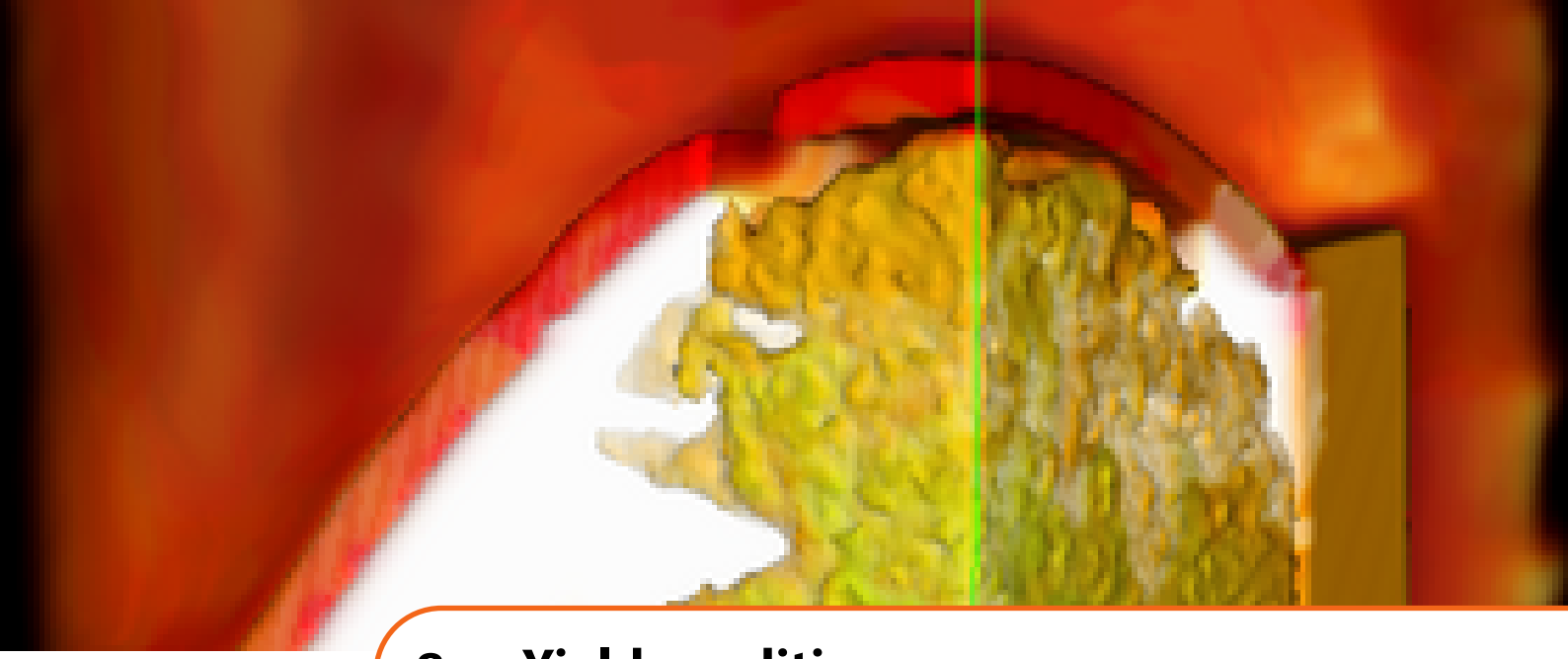
$$W_{\text{dev}}(\varepsilon_v^e, \varepsilon_s^e) = \tfrac{3}{2}\,\mu\,(\varepsilon_s^e)^2 \tag{7.11}$$

where $\varepsilon_v^e = \text{tr}(\boldsymbol{\epsilon}_v^e)$ is the volumetric part of the elastic strain, $\varepsilon_s^e = \sqrt{2/3\text{dev}(\boldsymbol{\epsilon}_v^e) : \text{dev}(\boldsymbol{\epsilon}_v^e)}$ is the deviatoric part of the elastic strain, and μ is the shear modulus.

The shear modulus in the Borja model is computed as

$$\mu(p) = \mu_o - \alpha p_o \exp\left(-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}}\right) \tag{7.12}$$

where μ_o is a reference shear modulus, ε_{vo}^e is the volumetric strain corresponding to a mean normal compressive stress p_o, and $\tilde{\kappa}$ is the elastic compressibility index.



8 — Yield condition

The yield condition models in VAANGO are of two types: yield conditions that are tightly tied to material models such as CamClay, Arenisca3, Arena, Mohr-Coulomb etc. and those that can be switched in the input file. This chapter discusses those yield conditions that can be easily substituted while simulating isotropic metal plasticity. The other yield conditions are described in the chapters that deal with specific models.

8.1 von Mises yield

The von Mises yield function implemented in VAANGO has the form

$$f = \sigma_{\text{eff}}^{\xi} - \sigma_y(\varepsilon_p^{\text{eq}}, \dot{\varepsilon}_p^{\text{eq}}, \phi, T, \dot{\varepsilon}^{\text{eq}}, \dots) \tag{8.1}$$

where σ_y is the flow stress, $\varepsilon_p^{\text{eq}}$ is the equivalent plastic strain, $\dot{\varepsilon}_p^{\text{eq}}$ is the equivalent plastic strain rate, ϕ is the porosity, and T is the temperature. The equivalent stress is defined as

$$\sigma_{\text{eff}}^{\xi} = \sqrt{3J_2^{\xi}} = \sqrt{\tfrac{3}{2}\boldsymbol{\xi} : \boldsymbol{\xi}}\,, \quad \boldsymbol{\xi} = \boldsymbol{s} - \text{dev}(\boldsymbol{\beta})\,, \quad \boldsymbol{s} = \boldsymbol{\sigma} - \tfrac{1}{3}\text{tr}(\boldsymbol{\sigma})\boldsymbol{I} \tag{8.2}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress and $\boldsymbol{\beta}$ is the kinematic hardening backstress.

The normal to the yield surface is

$$\boldsymbol{N} = \frac{\partial f}{\partial \boldsymbol{\sigma}} = \frac{\partial f}{\partial \boldsymbol{\xi}} : \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\sigma}} = \frac{\partial f}{\partial \boldsymbol{\xi}} : \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{s}} : \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{\sigma}} \tag{8.3}$$

Noting that

$$\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{\sigma}} = \text{symm}(\mathbb{I}) - \tfrac{1}{3}\boldsymbol{I} \otimes \boldsymbol{I} \quad \text{and} \quad \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{s}} = \text{symm}(\mathbb{I}) \tag{8.4}$$

where $\mathbb{I}$ is the fourth-order identity tensor and $\boldsymbol{I}$ is the second-order identity tensor, we have

$$\begin{aligned} \boldsymbol{N} &= \frac{\partial f}{\partial \boldsymbol{\xi}} : \text{symm}(\mathbb{I}) : (\text{symm}(\mathbb{I}) - \tfrac{1}{3}\boldsymbol{I} \otimes \boldsymbol{I}) = \frac{\partial f}{\partial \boldsymbol{\xi}} : (\text{symm}(\mathbb{I}) - \tfrac{1}{3}\boldsymbol{I} \otimes \boldsymbol{I}) \\ &= \frac{\partial f}{\partial \boldsymbol{\xi}} - \tfrac{1}{3}\text{tr}\left(\frac{\partial f}{\partial \boldsymbol{\xi}}\right)\boldsymbol{I}\,. \end{aligned} \tag{8.5}$$

Next we compute the derivative of f:

$$\frac{\partial f}{\partial \boldsymbol{\xi}} = \frac{\partial f}{\partial \sigma_{\text{eff}}^{\xi}} \frac{\partial \sigma_{\text{eff}}^{\xi}}{\partial \boldsymbol{\xi}} = \sqrt{\tfrac{3}{2}} \frac{\boldsymbol{\xi}}{\sqrt{\boldsymbol{\xi} : \boldsymbol{\xi}}} \quad \Longrightarrow \quad \text{tr}\left(\frac{\partial f}{\partial \boldsymbol{\xi}}\right) = 0 \,. \tag{8.6}$$

Therefore,

$$\boldsymbol{N} = \sqrt{\tfrac{3}{2}} \frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|} \tag{8.7}$$

The unit normal to the yield surface is

$$\hat{\boldsymbol{N}} = \frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|} \,. \tag{8.8}$$

The von Mises yield condition is the default for metal plasticity and can be invoked using the tag

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

8.2 The Gurson-Tvergaard-Needleman (GTN) yield condition

The Gurson-Tvergaard-Needleman (GTN) yield condition [36, 37] depends on porosity. The GTN yield function can be written as

$$f = \left(\frac{\sigma_{\text{eff}}^{\xi}}{\sigma_y}\right)^2 + 2q_1\phi_\star \cosh\left(q_2 \frac{\text{tr}(\boldsymbol{\sigma}^{\xi})}{2\sigma_y}\right) - (1 + q_3\phi_\star^2) \tag{8.9}$$

where $\boldsymbol{\sigma}^{\xi} = \boldsymbol{\sigma} - \boldsymbol{\beta}$, $\boldsymbol{\sigma}$ is the Cauchy stress, $\boldsymbol{\beta}$ is the backstress, $\sigma_{\text{eff}}^{\xi}$ is the equivalent stress defined in (8.2), σ_y is the flow stress of the void-free material, q_1, q_2, q_3 are material constants, and $\phi_\star$ is the porosity function defined as

$$\phi_\star = \begin{cases} \phi & \text{for } \phi \le \phi_c, \\ \phi_c + k(\phi - \phi_c) & \text{for } \phi > \phi_c \end{cases} \tag{8.10}$$

where k is a constant and ϕ is the porosity (void volume fraction).

The normal to the yield surface is

$$\boldsymbol{N} = \frac{\partial f}{\partial \boldsymbol{\sigma}} = \frac{\partial f}{\partial \boldsymbol{\xi}} : \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\sigma}} + \frac{\partial f}{\partial I_1^{\xi}} \frac{\partial I_1^{\xi}}{\partial \boldsymbol{\sigma}} = \frac{\partial f}{\partial \boldsymbol{\xi}} : \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{s}} : \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{\sigma}} + \frac{\partial f}{\partial I_1^{\xi}} \frac{\partial I_1^{\xi}}{\partial I_1} \frac{\partial I_1}{\partial \boldsymbol{\sigma}} \tag{8.11}$$

where $I_1 = \text{tr}(\boldsymbol{\sigma})$ and $I_1^{\xi} = \text{tr}(\boldsymbol{\sigma}^{\xi}) = \text{tr}(\boldsymbol{\sigma} - \boldsymbol{\beta})$. Using (8.5),

$$\boldsymbol{N} = \frac{\partial f}{\partial \boldsymbol{\xi}} - \tfrac{1}{3}\text{tr}\left(\frac{\partial f}{\partial \boldsymbol{\xi}}\right)\boldsymbol{I} + \frac{\partial f}{\partial I_1^{\xi}} \frac{\partial I_1^{\xi}}{\partial I_1} \frac{\partial I_1}{\partial \boldsymbol{\sigma}} \,. \tag{8.12}$$

Noting that

$$\frac{\partial I_1^{\xi}}{\partial I_1} = 1 \quad \text{and} \quad \frac{\partial I_1}{\partial \boldsymbol{\sigma}} = \boldsymbol{I} \tag{8.13}$$

we have

$$\boldsymbol{N} = \frac{\partial f}{\partial \boldsymbol{\xi}} - \tfrac{1}{3}\text{tr}\left(\frac{\partial f}{\partial \boldsymbol{\xi}}\right)\boldsymbol{I} + \frac{\partial f}{\partial I_1^{\xi}}\boldsymbol{I} \,. \tag{8.14}$$

Computation of the derivatives of f gives

$$\frac{\partial f}{\partial \boldsymbol{\xi}} = \frac{\partial f}{\partial \sigma_{\text{eff}}^{\xi}} \frac{\partial \sigma_{\text{eff}}^{\xi}}{\partial \boldsymbol{\xi}} = \sqrt{\tfrac{3}{2}} \left(\frac{2\sigma_{\text{eff}}^{\xi}}{\sigma_y^2} \right) \frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|} = \frac{3\boldsymbol{\xi}}{\sigma_y^2} \quad \Longrightarrow \quad \text{tr}\left(\frac{\partial f}{\partial \xi} \right) = 0 \,. \tag{8.15}$$

and

$$\frac{\partial f}{\partial I_1^{\xi}} = \frac{q_1 q_2 \phi_\star}{\sigma_y} \sinh\left(q_2 \frac{\text{tr}(\boldsymbol{\sigma}^{\xi})}{2\sigma_y} \right) . \tag{8.16}$$

Therefore,

$$\boldsymbol{N} = \frac{3\boldsymbol{\xi}}{\sigma_y^2} + \frac{q_1 q_2 \phi_\star}{\sigma_y} \sinh\left(q_2 \frac{\text{tr}(\boldsymbol{\sigma}^{\xi})}{2\sigma_y} \right) \boldsymbol{I} \,. \tag{8.17}$$

The unit normal to the GTN yield surface is given by

$$\hat{\boldsymbol{N}} = \frac{\boldsymbol{N}}{\|\boldsymbol{N}\|} \,. \tag{8.18}$$

The GTN 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>
```

8.3 The Rousselier yield condition

The Rousselier yield condition [38] is another porosity-based yield condition that has been used for ductile tearing simulations.

The yield function is

$$f = \frac{\sigma_{\text{eff}}^{\xi}}{1-\phi} + D\sigma_1 \phi \exp\left(\frac{\text{tr}(\boldsymbol{\sigma}^{\xi})}{3(1-\phi)\sigma_1} \right) - \sigma_y \tag{8.19}$$

where D, σ_1 are material constants, and the remaining quantities have been defined in the previous section.

The normal to the yield surface is

$$\boldsymbol{N} = \frac{\partial f}{\partial \boldsymbol{\xi}} - \tfrac{1}{3}\text{tr}\left(\frac{\partial f}{\partial \boldsymbol{\xi}} \right) \boldsymbol{I} + \frac{\partial f}{\partial I_1^{\xi}} \boldsymbol{I} \tag{8.20}$$

where

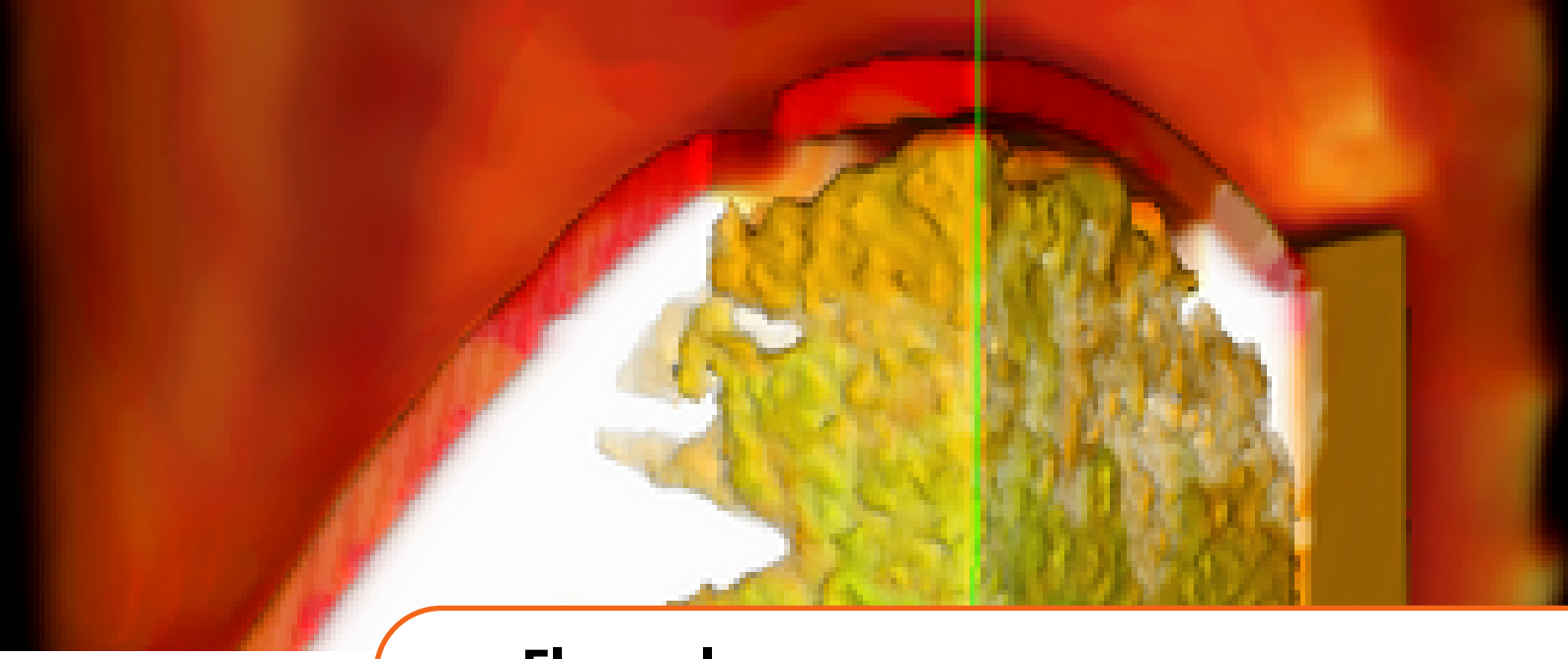
$$\frac{\partial f}{\partial \boldsymbol{\xi}} = \frac{\partial f}{\partial \sigma_{\text{eff}}^{\xi}} \frac{\partial \sigma_{\text{eff}}^{\xi}}{\partial \boldsymbol{\xi}} = \sqrt{\tfrac{3}{2}} \frac{1}{1-\phi} \frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|} \quad \Longrightarrow \quad \text{tr}\left(\frac{\partial f}{\partial \xi} \right) = 0 \tag{8.21}$$

and

$$\frac{\partial f}{\partial I_1^{\xi}} = \frac{D\phi}{3(1-\phi)} \exp\left(\frac{\text{tr}(\boldsymbol{\sigma}^{\xi})}{3(1-\phi)\sigma_1} \right) . \tag{8.22}$$

Therefore,

$$\boldsymbol{N} = \frac{1}{1-\phi} \left[\sqrt{\tfrac{3}{2}} \frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|} + \frac{D\phi}{3} \exp\left(\frac{\text{tr}(\boldsymbol{\sigma}^{\xi})}{3(1-\phi)\sigma_1} \right) \right] . \tag{8.23}$$



9 — Flow rule

Plastic flow rules in VAANGO have the form

$$\boldsymbol{d}^p = \dot{\boldsymbol{\varepsilon}}^p = \dot{\lambda}\boldsymbol{M} \tag{9.1}$$

where $\boldsymbol{d}^p = \dot{\boldsymbol{\varepsilon}}^p$ is the plastic strain rate tensor, λ is the consistency parameter, and $\boldsymbol{M}$ is a unit tensor in the direction of the plastic strain rate.

9.1 Associated plasticity

For associated plasticity, VAANGO uses the classical approach in plasticity theory and assumes that $\boldsymbol{M} = \hat{\boldsymbol{N}}$ is the unit normal to the yield surface:

$$\dot{\boldsymbol{\varepsilon}}^p = \dot{\lambda}\hat{\boldsymbol{N}}\,, \quad \hat{\boldsymbol{N}} = \frac{\dfrac{\partial f}{\partial \boldsymbol{\sigma}}}{\left\| \dfrac{\partial f}{\partial \boldsymbol{\sigma}} \right\|} \tag{9.2}$$

where f is the yield function.

9.2 Non-associated plasticity

VAANGO use two approaches for non-associated plasticity ($\boldsymbol{M} \neq \hat{\boldsymbol{N}}$). The first approach, implemented in the Mohr-Coulomb model, is to use a separate plastic potential (g) to compute $\boldsymbol{M}$:

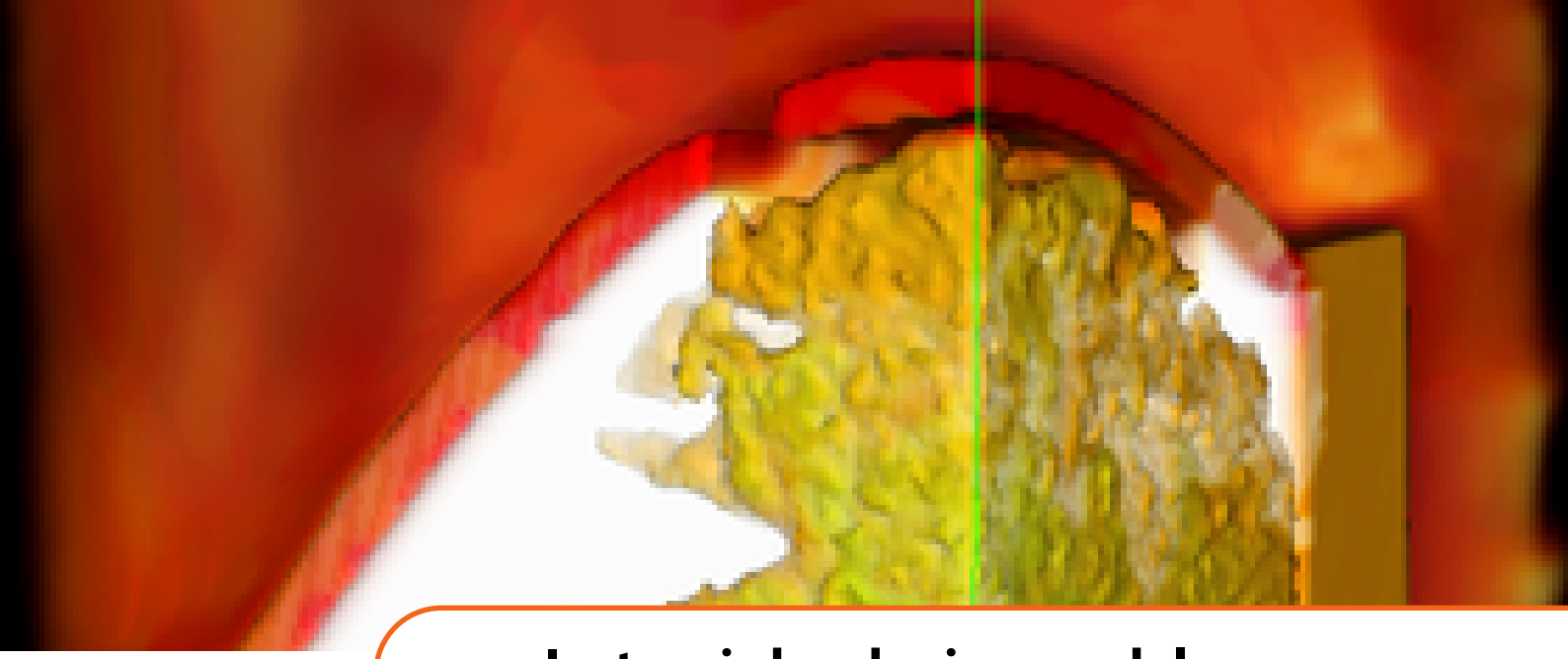
$$\dot{\boldsymbol{\varepsilon}}^p = \dot{\lambda}\hat{\boldsymbol{M}}\,, \quad \hat{\boldsymbol{M}} = \frac{\dfrac{\partial g}{\partial \boldsymbol{\sigma}}}{\left\| \dfrac{\partial g}{\partial \boldsymbol{\sigma}} \right\|} \tag{9.3}$$

The plastic potential is assumed to have the same form as the yield function, but different parameters to match experimental data on dilatation.

An alternative approach, used in Arenisca3 and Arena, is to compute the direction of the plastic strain rate tensor using

$$\dot{\boldsymbol{\varepsilon}}^p = \dot{\lambda}\hat{\boldsymbol{M}}\,, \quad \hat{\boldsymbol{M}} = \frac{\operatorname{dev}(\boldsymbol{N}) + \beta \operatorname{tr}(\boldsymbol{N})}{\left\| \operatorname{dev}(\boldsymbol{N}) + \beta \operatorname{tr}(\boldsymbol{N}) \right\|} \tag{9.4}$$

where β is an adjustable parameter.



10 — Isotropic hardening models

Several flow stress models have been implemented in VAANGO . These are described in this chapter.

10.1 Linear hardening model

The linear hardening model in VAANGO has the form

$$\sigma_y(\varepsilon_p^{eq}) = \sigma_o + K\varepsilon_p^{eq} \tag{10.1}$$

where σ_o is the initial yield stress, K is a hardening modulus, and ε_p^{eq} is the equivalent plastic strain.

The linear hardening model can be invoked using

```
<flow_model type="linear">
  <sigma_0> 700.0e6 </sigma_0>
  <K>1.5e6</K>
</flow_model>
```

10.2 Johnson-Cook model

The Johnson-Cook (JC) model ([39]) has the following relation for the flow stress (σ_y)

$$\sigma_y(\varepsilon_p^{eq}, \dot{\varepsilon}^{eq}, T) = \left[A + B(\varepsilon_p^{eq})^n\right]\left[1 + C\ln(\dot{\varepsilon}^\star)\right]\left[1 - (T^*)^m\right] \tag{10.2}$$

where ε_p^{eq} is the equivalent plastic strain, A, B, C, n, m are material constants, and

$$\dot{\varepsilon}^\star = \frac{\dot{\varepsilon}^{eq}}{\dot{\varepsilon}^o}; \quad T^* = \frac{(T - T_o)}{(T_m - T_o)}. \tag{10.3}$$

In the above equations, $\dot{\varepsilon}^{eq}$ is the equivalent strain rate, $\dot{\varepsilon}^o$ is a reference strain rate, T_o is a reference temperature, and T_m is the melt temperature. For conditions where $T^\star < 0$, we assume that $m = 1$.

The inputs for this model have the form

```
<flow_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>
</flow_model>
```

10.3 Steinberg-Guinan model

The Steinberg-Cochran-Guinan-Lund (SCG) model is a semi-empirical model that was developed by [32] for high strain rate situations and extended to low strain rates and bcc materials by [40]. The flow stress in this model is given by

$$\sigma_y(\varepsilon_p^{\text{eq}}, \dot{\varepsilon}_p^{\text{eq}}, T) = \left[\sigma_a f(\varepsilon_p^{\text{eq}}) + \sigma_t(\dot{\varepsilon}_p^{\text{eq}}, T)\right] \frac{\mu(p,T)}{\mu_o} \tag{10.4}$$

where σ_a is the athermal component of the flow stress, $f(\varepsilon_p^{\text{eq}})$ 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_o is the shear modulus at standard temperature and pressure. The strain hardening function has the form

$$f(\varepsilon_p^{\text{eq}}) = \left[1 + \beta(\varepsilon_p^{\text{eq}} + \varepsilon_{pi})\right]^n; \quad \sigma_a f(\varepsilon_p^{\text{eq}}) \le \sigma_{\max} \tag{10.5}$$

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 [41])

$$\dot{\varepsilon}_p^{\text{eq}} = \left[\frac{1}{C_1}\exp\left[\frac{2U_k}{k_b\,T}\left(1 - \frac{\sigma_t}{\sigma_p}\right)^2\right] + \frac{C_2}{\sigma_t}\right]^{-1}; \quad \sigma_t \le \sigma_p \tag{10.6}$$

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

$$C_1 := \frac{\rho_d L_d a b^2 \nu}{2w^2}; \quad C_2 := \frac{D}{\rho_d b^2} \tag{10.7}$$

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

```
<flow_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>
</flow_model>
```

10.4 Zerilli-Armstrong model

The Zerilli-Armstrong (ZA) model ([42–44]) is based on simplified dislocation mechanics. The general form of the equation for the flow stress is

$$\sigma_y(\varepsilon_p^{\text{eq}}, \dot{\varepsilon}_p^{\text{eq}}, T) = \sigma_a + B\exp(-\beta(\dot{\varepsilon}_p^{\text{eq}})T) + B_0\sqrt{\varepsilon_p^{\text{eq}}}\exp(-\alpha(\dot{\varepsilon}_p^{\text{eq}})T) \tag{10.8}$$

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

$$\sigma_a := \sigma_g + \frac{k_h}{\sqrt{l}} + K(\varepsilon_p^{\text{eq}})^n, \tag{10.9}$$

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

$$\alpha = \alpha_0 - \alpha_1\ln(\dot{\varepsilon}_p^{\text{eq}}); \quad \beta = \beta_0 - \beta_1\ln(\dot{\varepsilon}_p^{\text{eq}}); \tag{10.10}$$

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 [45] for better performance at high temperatures. However, we have not used the modified equations in our computations.

The input for this model is of the form

```
<flow_model type="zerilli_armstrong">
  <sigma_g>      46.5e6    </sigma_g>
  <k_H>          5.0e6     </k_H>
  <sqrt_l_inv>   3.7       </sqrt_l_inv>
  <B>            0.0       </B>
  <beta_0>       0.0       </beta_0>
  <beta_1>       0.0       </beta_1>
  <B_0>          890.0e6   </B_0>
  <alpha_0>      0.0028    </alpha_0>
  <alpha_1>      0.000115  </alpha_1>
  <K>            0.0       </K>
  <n>            0.0       </n>
</flow_model>
```

10.5 Polymer Zerilli-Armstrong model

The Zerilli-Armstrong model for polymers has the form:

$$\sigma_y(\varepsilon_p^{\text{eq}}, \dot{\varepsilon}_p^{\text{eq}}, T) = \sigma_g + B\exp(-\beta T^*) + B_0\sqrt{\omega\varepsilon_p^{\text{eq}}}\exp(-\alpha T^*) \tag{10.11}$$

where σ_g is the athermal component of the flow stress and

$$\omega = \omega_a + \omega_b\ln(\dot{\varepsilon}_p^{\text{eq}}) + \omega_p\sqrt{\bar{p}} \tag{10.12}$$

where ω_a, ω_b, ω_p are material parameters and $\bar{p} = -p$ is the pressure (positive in compression). The functional forms of the exponents α and β are

$$\alpha = \alpha_0 - \alpha_1\ln(\dot{\varepsilon}_p^{\text{eq}}); \quad \beta = \beta_0 - \beta_1\ln(\dot{\varepsilon}_p^{\text{eq}}); \tag{10.13}$$

where $\alpha_0, \alpha_1, \beta_0, \beta_1$ are material parameters. The factors B and B_0 are defined as

$$B = B_{\text{pa}}\left(1 + B_{\text{pb}}\sqrt{\bar{p}}\right)^{B_{\text{pn}}}, \quad B_0 = B_{\text{opa}}\left(1 + B_{\text{opb}}\sqrt{\bar{p}}\right)^{B_{\text{opn}}} \tag{10.14}$$

where B_{pa}, B_{opa}, B_{pb}, B_{opb}, B_{pn}, and B_{opn} are material parameters. Also,

$$T^{\star} = \frac{T}{T_o} \tag{10.15}$$

where T_o is a reference temperature.

The input tags for the polymer ZA model are:

```
<flow_model type="zerilli_armstrong_polymer">
  <sigma_g>      46.5e6   </sigma_g>
  <B_pa>         0.0      </B_pa>
  <B_pb>         0.0      </B_pb>
  <B_pn>         1.0      </B_pn>
  <beta_0>       0.0      </beta_0>
  <beta_1>       0.0      </beta_1>
  <T_0>          300.0    </T_0>
  <B_0pa>        890.0e6  </B_0pa>
  <B_0pb>        0.0      </B_0pb>
  <B_0pn>        1.0      </B_0pn>
  <omega_a>      0.0      </omega_a>
  <omega_b>      0.0      </omega_b>
  <omega_p>      0.0      </omega_p>
</flow_model>
```

10.6 Mechanical thresold stress model

The Mechanical Threshold Stress (MTS) model ([46–48]) gives the following form for the flow stress

$$\sigma_y(\varepsilon_p^{eq}, \dot{\varepsilon}_p^{eq}, T) = \sigma_a + (S_i\sigma_i + S_e\sigma_e)\frac{\mu(p,T)}{\mu_o} \tag{10.16}$$

where σ_a is the athermal component of mechanical threshold stress, μ_o is the shear modulus at o 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

$$S_i = \left[1 - \left(\frac{k_b\, T}{g_{oi} b^3 \mu(p,T)} \ln\frac{\dot{\varepsilon}_{poi}}{\dot{\varepsilon}_p^{eq}}\right)^{1/q_i}\right]^{1/p_i} \tag{10.17}$$

$$S_e = \left[1 - \left(\frac{k_b\, T}{g_{oe} b^3 \mu(p,T)} \ln\frac{\dot{\varepsilon}_{poe}}{\dot{\varepsilon}_p^{eq}}\right)^{1/q_e}\right]^{1/p_e} \tag{10.18}$$

where k_b is the Boltzmann constant, b is the magnitude of the Burgers' vector, (g_{oi}, g_{oe}) 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

$$\frac{d\sigma_e}{d\varepsilon_p^{eq}} = \theta(\sigma_e) \tag{10.19}$$

where

$$\theta(\sigma_e) = \theta_0[1 - F(\sigma_e)] + \theta_{IV}F(\sigma_e) \tag{10.20}$$

$$\theta_0 = a_0 + a_1 \ln \dot{\varepsilon}_p^{\text{eq}} + a_2\sqrt{\dot{\varepsilon}_p^{\text{eq}}} - a_3 T \tag{10.21}$$

$$F(\sigma_e) = \frac{\tanh\left(\alpha\dfrac{\sigma_e}{\sigma_{es}}\right)}{\tanh(\alpha)} \tag{10.22}$$

$$\ln\left(\frac{\sigma_{es}}{\sigma_{oes}}\right) = \left(\frac{kT}{g_{oes}b^3\mu(p,T)}\right)\ln\left(\frac{\dot{\varepsilon}_p^{\text{eq}}}{\dot{\varepsilon}_{poes}}\right) \tag{10.23}$$

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_{oes} is the saturation threshold stress for deformation at 0 K, g_{oes} 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

```
<flow_model type="mechanical_threshold_stress">
  <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>
</flow_model>
```

10.7 Preston-Tonks-Wallace model

The Preston-Tonks-Wallace (PTW) model ([34]) 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

$$\sigma_y(\varepsilon_p^{\text{eq}}, \dot{\varepsilon}_p^{\text{eq}}, T) = \begin{cases} 2\left[\tau_s + \alpha \ln\left[1 - \varphi \exp\left(-\beta - \dfrac{\theta\varepsilon_p^{\text{eq}}}{\alpha\varphi}\right)\right]\right]\mu(p,T) & \text{thermal regime} \\ 2\tau_s\mu(p,T) & \text{shock regime} \end{cases} \tag{10.24}$$

with

$$\alpha := \frac{s_0 - \tau_y}{d}; \quad \beta := \frac{\tau_s - \tau_y}{\alpha}; \quad \varphi := \exp(\beta) - 1 \tag{10.25}$$

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

$$\tau_s = \max\left\{ s_0 - (s_0 - s_\infty)\operatorname{erf}\left[\kappa \hat{T} \ln\left(\frac{\gamma\dot{\xi}}{\dot{\varepsilon}_p^{\text{eq}}}\right)\right], s_0\left(\frac{\dot{\varepsilon}_p^{\text{eq}}}{\gamma\dot{\xi}}\right)^{s_1}\right\} \tag{10.26}$$

$$\tau_y = \max\left\{ y_0 - (y_0 - y_\infty)\operatorname{erf}\left[\kappa \hat{T} \ln\left(\frac{\gamma\dot{\xi}}{\dot{\varepsilon}_p^{\text{eq}}}\right)\right], \min\left\{ y_1\left(\frac{\dot{\varepsilon}_p^{\text{eq}}}{\gamma\dot{\xi}}\right)^{y_2}, s_0\left(\frac{\dot{\varepsilon}_p^{\text{eq}}}{\gamma\dot{\xi}}\right)^{s_1}\right\}\right\} \tag{10.27}$$

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

$$\dot{\xi} = \frac{1}{2}\left(\frac{4\pi\rho}{3M}\right)^{1/3}\left(\frac{\mu(p,T)}{\rho}\right)^{1/2} \tag{10.28}$$

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

The inputs for this model are of the form

```
<flow_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>
</flow_model>
```

10.8 SUVIC-I model

SUVIC-I is a viscoplastic model for ice that has been used for modeling the interaction of tires with ice [49, 50]. The model is an extension of the SUVIC model (Strain-rate history dependent Unified Viscoplastic model with Internal Variables for Crystalline materials). The model is applicable for strain rates in the range of 10^{-8} to 10^{-2} s^{-1}. Further details on the range of applicability of the model can be found in [49].

This model is driven by a specialized viscoplastic constitutive model for which the yield criterion is given by

$$\left\langle \frac{X_{ae} - R}{K} \right\rangle, \qquad \langle x \rangle = \begin{cases} x & \text{for} \quad x > 0 \\ 0 & \text{for} \quad x \le 0 \end{cases} \tag{10.29}$$

where the effective reduced stress is defined as

$$X_{ae} := \sqrt{\tfrac{3}{2}(\boldsymbol{s} - \boldsymbol{\beta}) : (\boldsymbol{s} - \boldsymbol{\beta})} \tag{10.30}$$

and R is the yield stress, K is the drag stress, $\boldsymbol{s} = \operatorname{dev}(\boldsymbol{\sigma})$ is the deviatoric stress, $\boldsymbol{\beta}$ is the deviatoric backstress.

The inelastic strain rate is given by

$$\dot{\boldsymbol{\varepsilon}}^i = \left[A\left\langle \frac{X_{ae} - R}{K} \right\rangle^N \exp\left(-\frac{Q}{RT}\right)\right]\boldsymbol{n}, \quad \boldsymbol{n} = \tfrac{3}{2}\left(\frac{\boldsymbol{s} - \boldsymbol{\beta}}{X_{ae}}\right) \tag{10.31}$$

where A is a kinetic-law material constants, and Q, R, T are the activation energy, Universal gas constant and the absolute temperature, respectively.

The equivalent inelastic strain rate for the model is defined as

$$\dot{\varepsilon}_i^{\text{eq}} = \sqrt{\tfrac{2}{3}\dot{\boldsymbol{\varepsilon}}^i : \dot{\boldsymbol{\varepsilon}}^i} = A\left\langle \frac{X_{ae} - R}{K} \right\rangle^N \exp\left(-\frac{Q}{RT}\right). \tag{10.32}$$

The evolution of the deviatoric backstress is given by

$$\dot{\boldsymbol{\beta}} = \tfrac{2}{3}A_1\dot{\boldsymbol{\varepsilon}}^i - A_1(\beta_{\text{sat}})^{-1}\dot{\varepsilon}_i^{\text{eq}}\boldsymbol{\beta} - A_2(\beta_{\text{sat}}^{\text{eff}})^{q-1}C^{-q}\boldsymbol{\beta} \tag{10.33}$$

where A_1 is a constant that is fitted to the kinematic hardening and dynanamic recovery curves, β_{sat} is a saturation value of the backstress, and A_2, q, C are constants fitted to static recovery curves. The isotropic hardening yield stress evolves as

$$\dot{R} = \frac{A_3}{d_g}\dot{\varepsilon}_i^{\text{eq}}\left(1 - \frac{R}{R_{\text{sat}}}\right) - \frac{A_4}{d_g}\frac{(R - R_{\text{sat}})^p}{C} \tag{10.34}$$

where A_3 is a constant, d_g is the grain size, R_{sat} is the saturation value of R, and A_4, p, C are constants associated with static recovery. The drag stress evolves as

$$\dot{K} = \frac{A_5}{d_g}\dot{\varepsilon}_i^{\text{eq}}\left(1 - \frac{K}{K_{\text{sat}}}\right) - \frac{A_6}{d_g}\frac{(K - K_{\text{sat}})^q}{C} \tag{10.35}$$

and has the same form as the yield stress evolution rule. The static recovery terms are ignored in the VAANGO implementation. The saturation value for the effective stress is given by

$$\sigma_{\text{sat}}^{\text{eff}} = \sigma_{\text{o}}\left(\frac{\dot{\varepsilon}_i^{\text{eq}}}{\dot{\varepsilon}_{\text{o}}}\right)^{1/n} \tag{10.36}$$

where σ_{o} and $\dot{\varepsilon}_{\text{o}}$ are reference values and n is a constant. Similar relations are assumed for the reference values of the other quantities:

$$\beta_{\text{sat}}^{\text{eff}} = \beta_{\text{o}}\left(\frac{\dot{\varepsilon}_i^{\text{eq}}}{\dot{\varepsilon}_{\text{o}}}\right)^{1/n} \quad ; \quad R_{\text{sat}} = R_{\text{o}}\left(\frac{\dot{\varepsilon}_i^{\text{eq}}}{\dot{\varepsilon}_{\text{o}}}\right)^{1/n} \tag{10.37}$$

The drag stress saturation value is given by

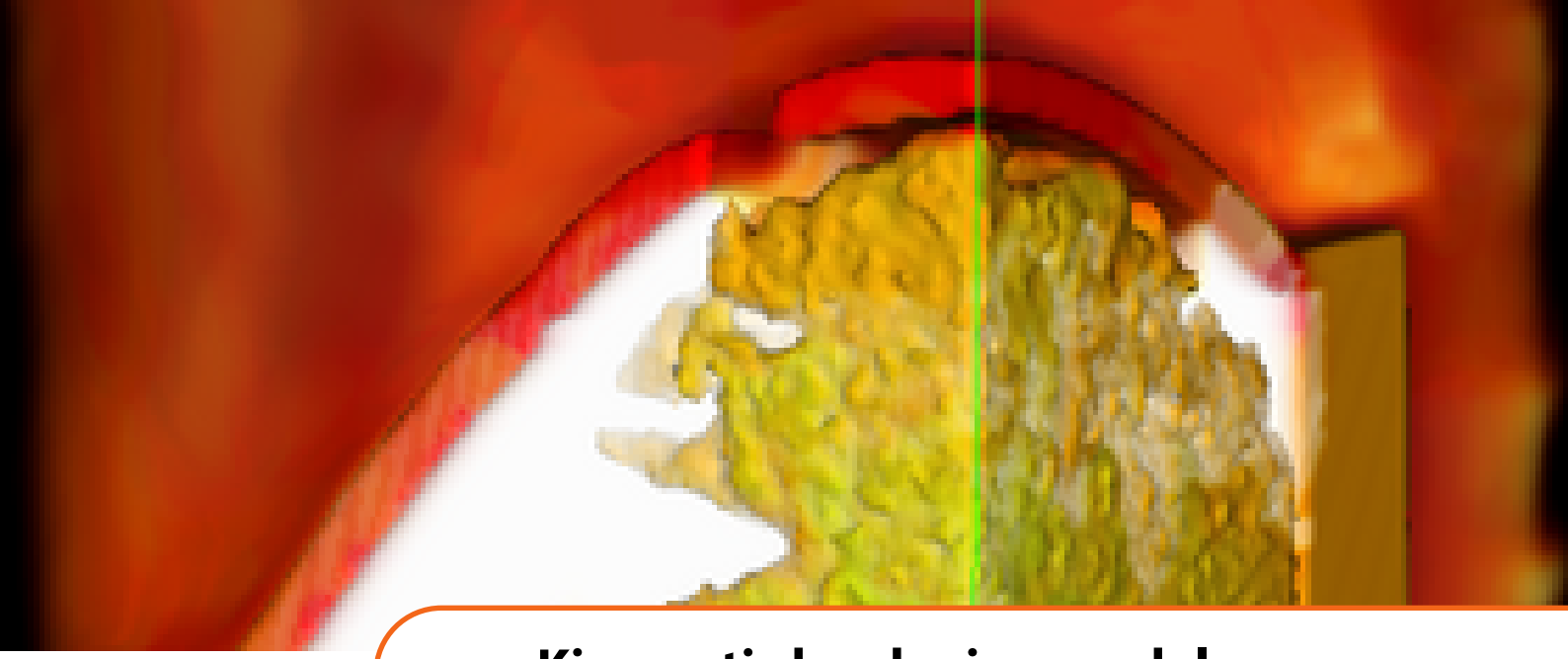
$$K_{\text{sat}} = \left[\frac{\dot{\varepsilon}_i^{\text{eq}}}{A\exp\left(\frac{-Q}{RT}\right)}\right](X_{\text{sat}} - R_{\text{sat}}) \tag{10.38}$$

where X_{sat} is the saturation value of X_{ae}.

A typical input deck for the SUVIC-I model is shown below.

```
<constitutive_model type="visco_plastic">
  <shear_modulus>3.52e9</shear_modulus>
  <bulk_modulus>8.9e9</bulk_modulus>
  <remove_particles> false </remove_particles>
  <zero_stress_upon_failure> false </zero_stress_upon_failure>
  <stability_check type="none">  </stability_check>
  <equation_of_state type="default_hypo">
    <bulk_modulus>8.9e9</bulk_modulus>
  </equation_of_state>
  <viscoplastic_flow_model type="suvic_i">
    <coeff_backstress_evol>75e6</coeff_backstress_evol>
    <exponent_backstress_evol> 1</exponent_backstress_evol>
    <normalizing_backstress> 1e6</normalizing_backstress>
    <coeff_saturation_backstress> 0.1e6</coeff_saturation_backstress>
    <exponent_backstress> 4</exponent_backstress>
```

```
      <ref_strainrate>7.794e-08</ref_strainrate>
      <normalizing_inelastic_strainrate>5.0e9 </normalizing_inelastic_strainrate>
      <activation_energy> 67500.0</activation_energy>
      <universal_gas_constant>8.3144 </universal_gas_constant>
      <temperature>269.15</temperature>
      <exponent_inelastic_strainrate>4.0 </exponent_inelastic_strainrate>
      <coeff_yieldstress_saturation>0.8e6 </coeff_yieldstress_saturation>
      <exponent_yieldstress>4.0 </exponent_yieldstress>
      <coeff_yieldstress_evol> 1600.0e6</coeff_yieldstress_evol>
      <exponent_dragstress> 4.0</exponent_dragstress>
      <coeff_dragstress_evol>95e6 </coeff_dragstress_evol>
      <coeff_stress_saturation>1.0e6</coeff_stress_saturation>
      <intial_drag>0.05e6</intial_drag>
      <initial_yield> 0.0</initial_yield>
      <integration_parameter_theta> 0.5</integration_parameter_theta>
    </viscoplastic_flow_model>
  </constitutive_model>
```



11 — Kinematic hardening models

Kinematic hardening in VAANGO is modeled with a backstress ($\boldsymbol{\beta}$) that is subtracted from the stress while evaluating the yield condition. In Arena, the pore water pressure ($\bar{p}_w = -p_w$) acts as a backstress, i.e.,

$$\boldsymbol{\beta} = \bar{p}_w \boldsymbol{I} \,. \tag{11.1}$$

For metals, the backstress can either be ignored or modeled using the approaches described in this chapter.

11.1 Ziegler-Prager model

If the evolution of the backstress is given by the Ziegler-Prager kinematic hardening rule, we have

$$\dot{\boldsymbol{\beta}} = \frac{2}{3}\, \beta H\, \dot{\boldsymbol{\varepsilon}}^p \tag{11.2}$$

where $\boldsymbol{\beta}$ is the backstress, βH is a constant hardening modulus, and $\dot{\boldsymbol{\varepsilon}}^p$ is the plastic strain rate.

The Prager model is invoked using

```
<kinematic_hardening_model type="prager_hardening">
  <beta> 1.0 </beta>
  <hardening_modulus>1.5e6</hardening_modulus>
</kinematic_hardening_model>
```

11.2 Armstrong-Frederick model

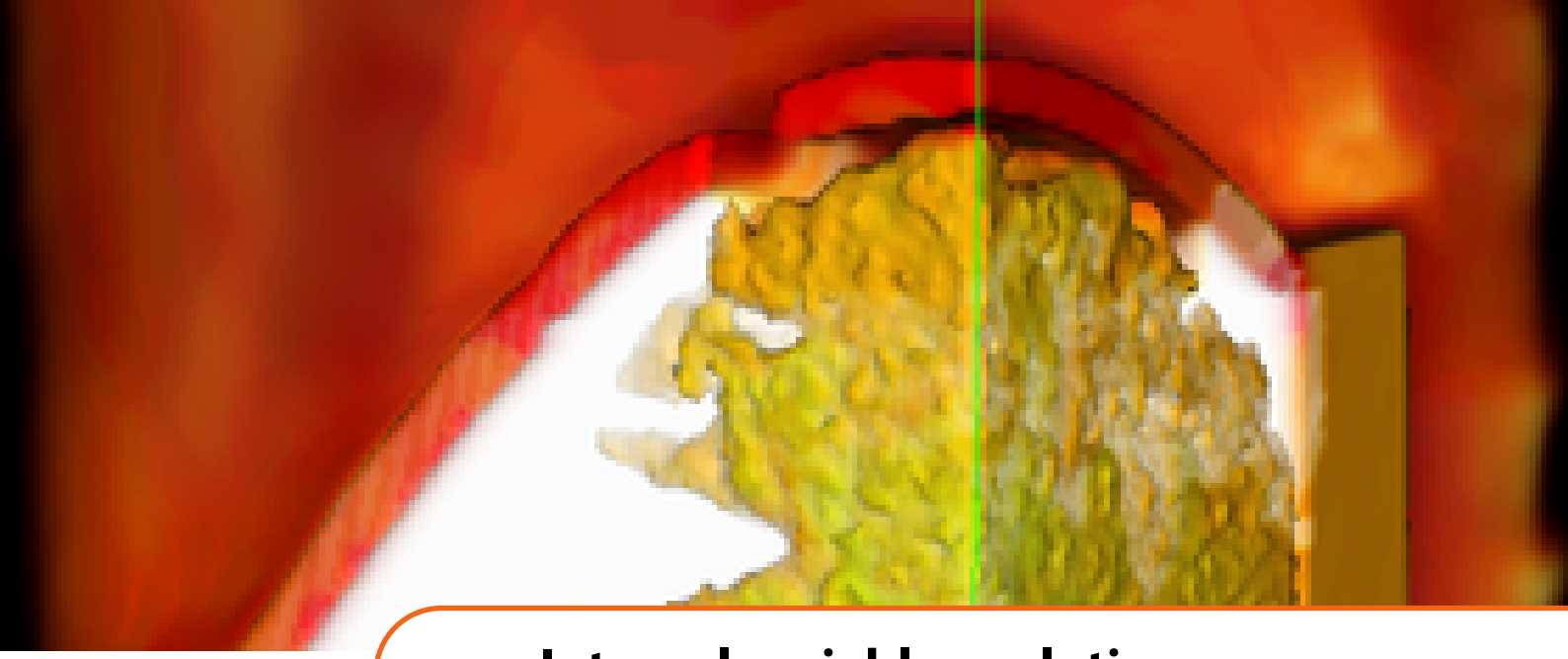
The Armstrong-Frederick model evolves the backstress using

$$\dot{\boldsymbol{\beta}} = \frac{2}{3}\, \beta H_1\, \dot{\boldsymbol{\varepsilon}}^p - \beta H_2\, \boldsymbol{\beta}\, \left\| \dot{\boldsymbol{\varepsilon}}^p \right\| \tag{11.3}$$

where β, H_1 and H_2 are material parameters.

The Armstrong-Frederick model is invoked using

```
<kinematic_hardening_model type="armstrong_frederick_hardening">
  <beta> 1.0 </beta>
  <hardening_modulus_1>1.5e6</hardening_modulus_1>
  <hardening_modulus_2>1.5e4</hardening_modulus_2>
</kinematic_hardening_model>
```



12 — Internal variable evolution

Internal variables are used to model isotropic hardening/softening behavior in VAANGO. The evolution of these internal variables is assumed to be given by first-order differential equations of the form

$$\dot{\boldsymbol{\eta}} = \dot{\lambda}\boldsymbol{h}_{\eta} \tag{12.1}$$

where $\boldsymbol{\eta}$ is the internal variable, λ is the consistency parameter, and $\boldsymbol{h}_{\eta}$ is a hardening/softening modulus.

> The kinematic hardening backstress is also an internal variable. Equations for the evolution of backstress are given in Chapter 11. Other internal variables that are specific to CamCay, Arena, Tabular plasticity, etc. are discussed in separate chapters associated with these models.

12.1 Equivalent plastic strain

Recall from the flow rule (9.1) that

$$\dot{\boldsymbol{\varepsilon}}^{p} = \dot{\lambda}\boldsymbol{M} \tag{12.2}$$

where $\boldsymbol{M}$ is a unit tensor ($\boldsymbol{M} : \boldsymbol{M} = 1$). Therefore, using the definition of the equivalent plastic strain rate from (2.16),

$$\dot{\boldsymbol{\varepsilon}}^{p} : \dot{\boldsymbol{\varepsilon}}^{p} = \left(\dot{\lambda}\right)^{2} \quad \Longrightarrow \quad \dot{\lambda} = \sqrt{\dot{\boldsymbol{\varepsilon}}^{p} : \dot{\boldsymbol{\varepsilon}}^{p}} = \dot{\varepsilon}_{p}^{\mathrm{eq}} . \tag{12.3}$$

Therefore, from the definition of the equivalent plastic strain in (2.17), we see that the evolution rule for the equivalent plastic strain can be expressed in the the form (12.1) as

$$\dot{\varepsilon}_{p}^{\mathrm{eq}} = \dot{\lambda} h_{\varepsilon_p} , \quad h_{\varepsilon_p} = 1 . \tag{12.4}$$

12.2 Porosity

The evolution of porosity is assumed to be given by the sum of the rate of void growth and the rate of void nucleation [51]. In VAANGO these rates are computed as [52]:

$$\dot{\phi} = \dot{\phi}_{\mathrm{nucl}} + \dot{\phi}_{\mathrm{grow}} \tag{12.5}$$

$$\dot{\phi}_{\mathrm{grow}} = (1 - \phi)\mathrm{tr}(\dot{\boldsymbol{\varepsilon}}^{p}) \tag{12.6}$$

$$\dot{\phi}_{\mathrm{nucl}} = \frac{f_n}{(s_n\sqrt{2\pi})} \exp\left[-\tfrac{1}{2}\frac{(\varepsilon_p^{\mathrm{eq}} - \varepsilon_n)^2}{s_n^2}\right] \dot{\varepsilon}_{p}^{\mathrm{eq}} \tag{12.7}$$

where $\dot{\boldsymbol{\varepsilon}}_p$ is the plastic strain rate, 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.

From the flow rule (9.1),

$$\text{tr}(\dot{\boldsymbol{\varepsilon}}^p) = \dot{\lambda}\text{tr}(\boldsymbol{M}) \tag{12.8}$$

Therefore, using (12.3),

$$\dot{\phi} = \dot{\lambda} h_\phi \,, \quad h_\phi = (1-\phi)\text{tr}(\boldsymbol{M}) + \frac{f_n}{(s_n\sqrt{2\pi})} \exp\left(-\tfrac{1}{2}\frac{(\varepsilon_p^{\text{eq}} - \varepsilon_n)^2}{s_n^2}\right). \tag{12.9}$$

VAANGO allows for the possibly of porosity to be different in each MPM particle. The inputs tags for defining the porosity and its distribution are:

```
<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>
```

12.3 Backstress

The backstress evolution rule can also be expressed in terms of the consistency parameter in the form

$$\dot{\boldsymbol{\beta}} = \dot{\lambda}\boldsymbol{h}_\beta \tag{12.10}$$

For the Ziegler-Prager model in (11.2),

$$\boldsymbol{h}_\beta = \tfrac{2}{3}\beta H \boldsymbol{M} \tag{12.11}$$

where $\boldsymbol{M}$ is the unit tensor in the direction of th eplastic flow rate.

For the Armstrong-Frederick model in (11.3),

$$\boldsymbol{h}_\beta = \tfrac{2}{3}\beta H_1 \boldsymbol{M} - \beta H_2 \boldsymbol{\beta}\,. \tag{12.12}$$

12.4 Damage

The evolution of damage models in VAANGO is detailed in Chapter 15. These models have the general form

$$\dot{D} = g(\boldsymbol{\sigma}, T)\,\dot{\varepsilon}_p^{\text{eq}} = \dot{\lambda} h_D \,, \quad h_D = g(\boldsymbol{\sigma}, T) \tag{12.13}$$

where D is the damage parameter and $g(\boldsymbol{\sigma}, T)$ is a damage function.

12.5 Temperature

The rise in temperature due to plastic dissipation can also be treated as an internal variable that causes softening. This may be considered to be equivalent to treating the plastic work as an internal variable.

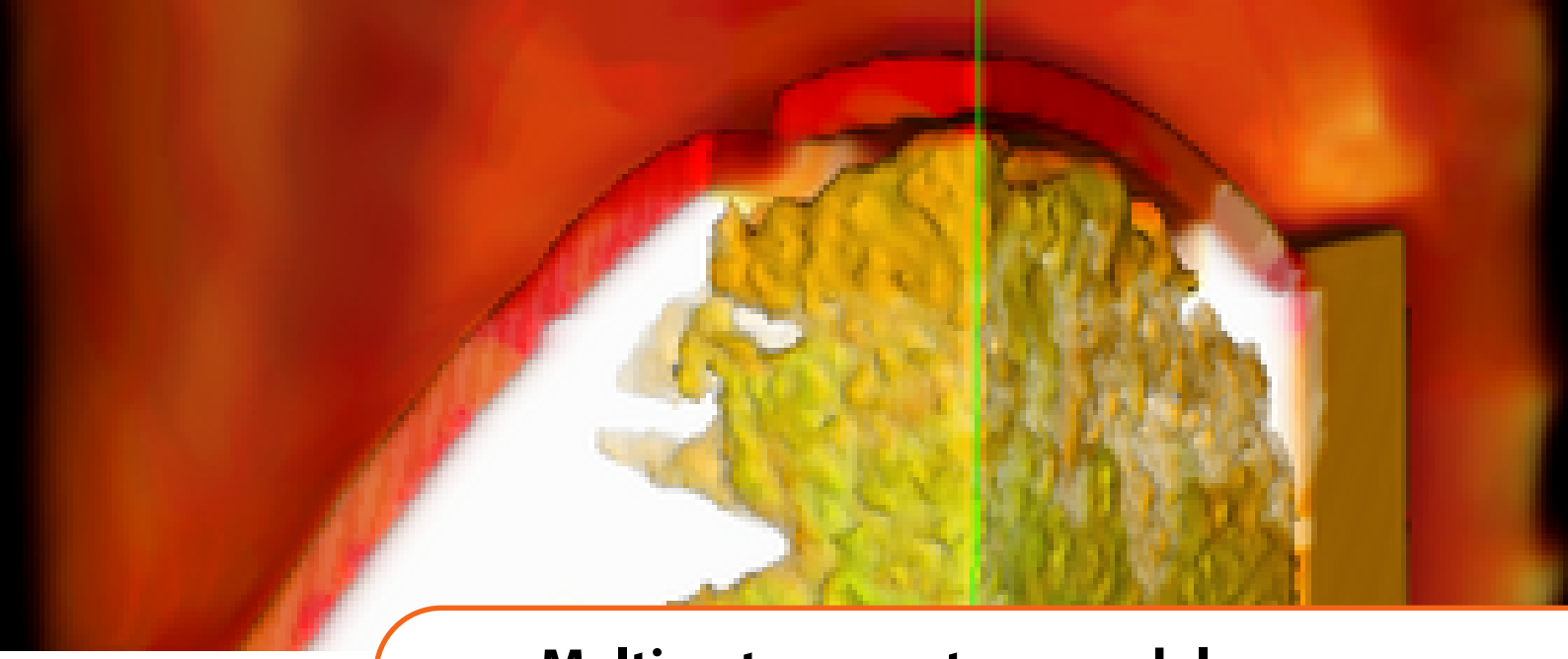
The evolution of temperature (T) due to plastic work is given by the equation

$$\dot{T} = \frac{\chi}{\rho C_p}\boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}}^p \tag{12.14}$$

where χ is the Taylor-Quinney coefficient, ρ is the density, and C_p is the specific heat.
Expressed in terms of the consistency parameter,

$$\dot{T} = \dot{\lambda} h_T \,, \quad h_T = \frac{\chi}{\rho C_p} \boldsymbol{\sigma} : \boldsymbol{M} \tag{12.15}$$

where $\boldsymbol{M}$ is the plastic flow rate direction defined in the flow rule.



13 — Melting temperature models

The melting temperature is used by several models in VAANGO to compute the shear modulus. Failure and transitioning into fluid-like behavior is also controlled by the melting temperature. The melt temperature models implemented in VAANGO are described below.

13.1 Constant melting temperature

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

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

13.2 Steinberg-Cochran-Guinan melting temperature

A pressure dependent relation to determine the melting temperature (T_m) in the Steinberg-Cochran-Guinan (SCG) melt model ([32]).

This model is based on a modified Lindemann law and has the form

$$T_m(\rho) = T_{mo} \exp\left[2a\left(1-\frac{1}{\eta}\right)\right]\eta^{2(\Gamma_o - a - 1/3)}; \quad \eta = \frac{\rho}{\rho_o} \tag{13.1}$$

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

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>
```

13.3 Burakovsky-Preston-Silbar melting temperature

The Burakovsky-Preston-Silbar (BPS) model is based on dislocation-mediated phase transitions [53]. The BPS model has the form

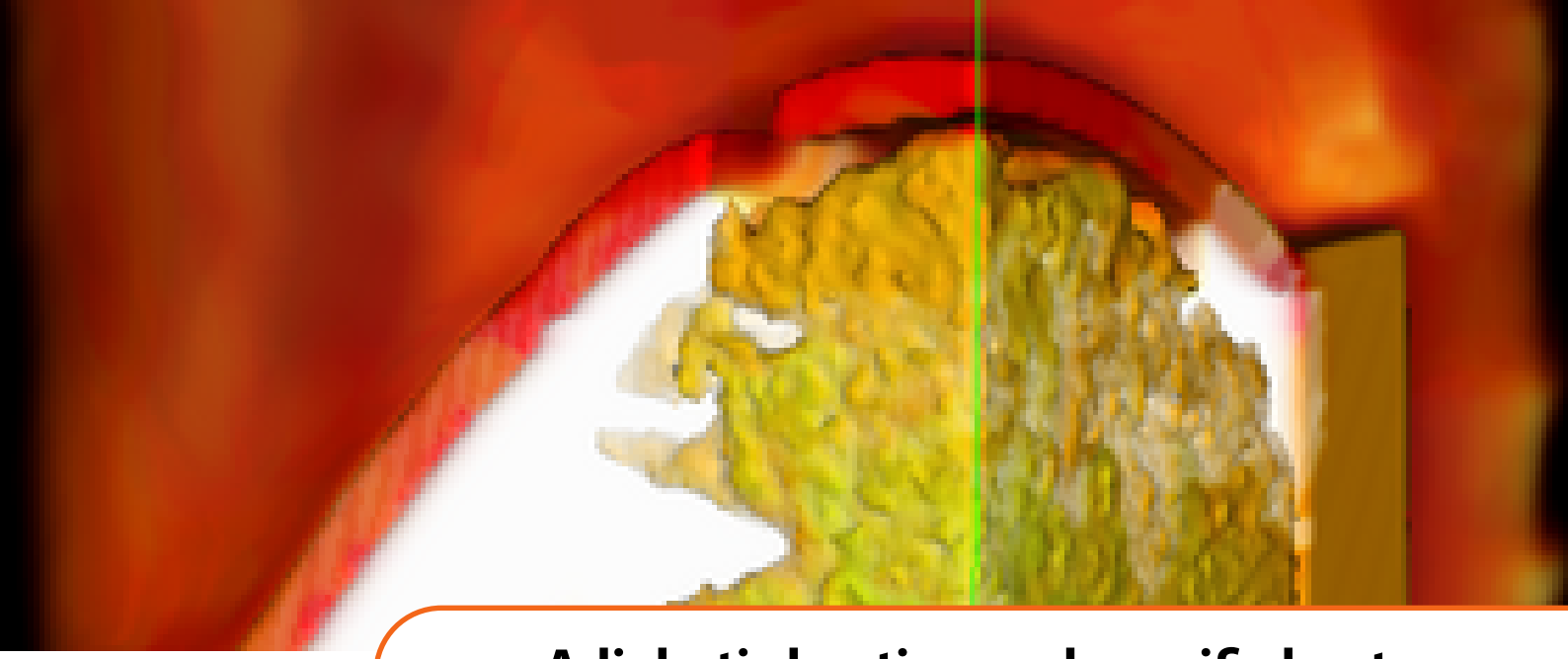
$$T_m(p) = T_m(0)\left[\frac{1}{\eta} + \frac{1}{\eta^{4/3}}\frac{\mu_0'}{\mu_0}p\right]; \quad \eta = \left(1 + \frac{K_0'}{K_0}p\right)^{1/K_0'} \tag{13.2}$$

$$T_m(0) = \frac{\kappa\lambda\mu_0 v_{WS}}{8\pi\ln(z-1)\,k_b}\ln\left(\frac{\alpha^2}{4\,b^2\rho_c(T_m)}\right) \tag{13.3}$$

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>
```



14 — 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 ($\Delta\varepsilon_p^{\text{eq}}$) is given by the equation

$$\Delta T = \frac{\chi\sigma_y}{\rho C_p}\Delta\varepsilon_p^{\text{eq}} \tag{14.1}$$

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 [54] for more details on the variation of χ with strain and strain rate).

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

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

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.

14.1 Constant specific heat model

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

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

14.2 Specific heat model for copper

The specific heat model for copper is of the form

$$Cp = \begin{cases} A_0\, T^3 - B_0\, T^2 + C_0\, T - D_0 & \text{if } T < T_0 \\ A_1\, T + B_1 & \text{if } T \geq T_0\,. \end{cases} \tag{14.2}$$

The model is invoked using

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

14.3 Specific heat model for steel

A relation for the dependence of C_p upon temperature is used for the steel ([55]).

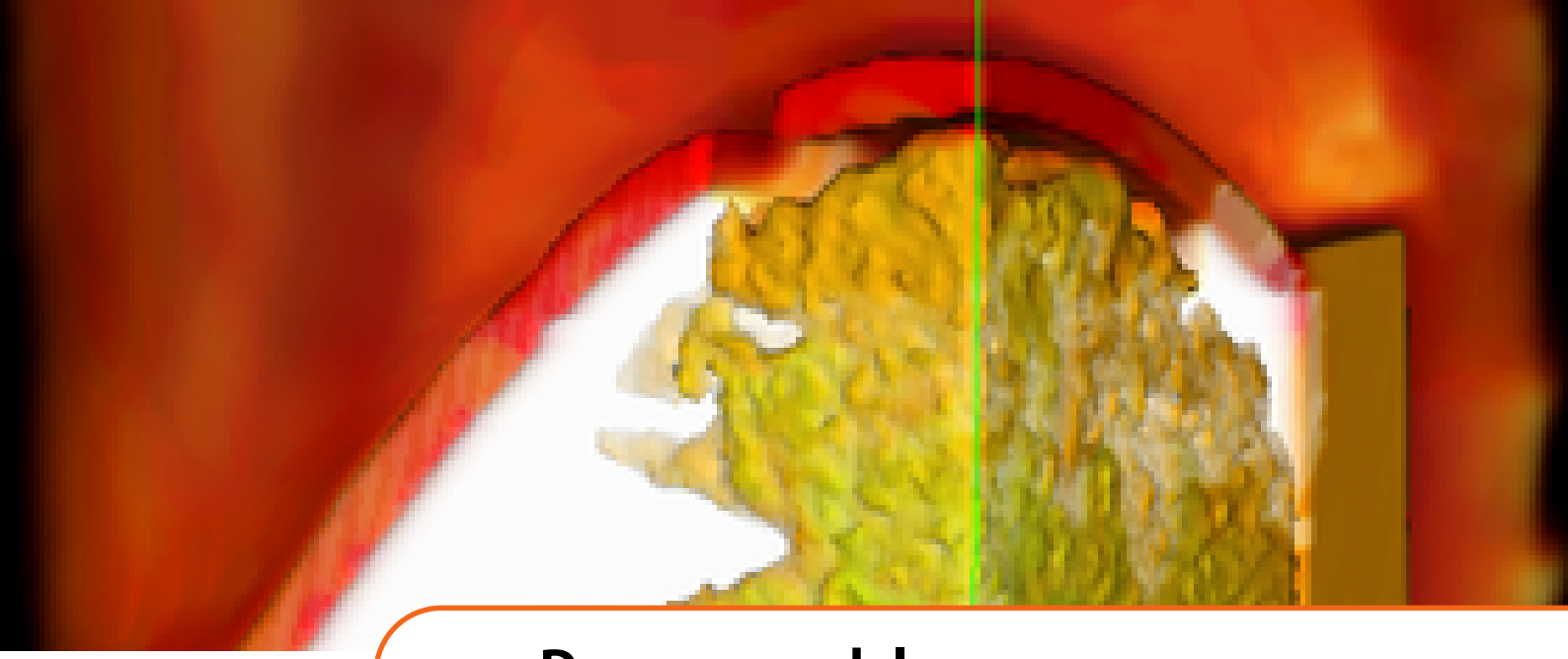
$$C_p = \begin{cases} A_1 + B_1\, t + C_1\, |t|^{-\alpha} & \text{if } T < T_c \\ A_2 + B_2\, t + C_2\, t^{-\alpha'} & \text{if } T > T_c \end{cases} \tag{14.3}$$

$$t = \frac{T}{T_c} - 1 \tag{14.4}$$

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>
```



15 — Damage models

The damage models implemented in VAANGO are described in this chapter. The most common model evolves a scalar damage parameter that can either be used to flag failure when a critical value is reached or to modify the stress as in continuum damage mechanics.

15.1 Hancock-MacKenzie model

The Hancock-MacKenzie model [56] evolves a scalar damage parameter (D) using the rule:

$$\dot{D} = \frac{1}{1.65}\dot{\varepsilon}_p^{\text{eq}} \exp\left(\frac{\text{tr}(\boldsymbol{\sigma})}{2\sigma_{\text{eff}}}\right) \tag{15.1}$$

where $D = 0$ for virgin material, $\dot{\varepsilon}_p$ is the equivalent plastic strain rate, $\boldsymbol{\sigma}$ is the Cauchy stress, and $\sigma_{\text{eq}} = \sqrt{3J_2}$ is the von Mises equivalent stress.

Expressed as an evolution equation in terms of the plastic consistency parameter, the above can be written as

$$\dot{D} = \dot{\lambda} h_D \,, \quad h_D = \frac{1}{1.65} \exp\left(\frac{\text{tr}(\boldsymbol{\sigma})}{2\sigma_{\text{eff}}}\right) . \tag{15.2}$$

The input tags for the damage model are :

```
<damage_model type="hancock_mackenzie">
  <D0>0.05</D1>
  <Dc>3.44</D2>
</damage_model>
```

15.2 Johnson-Cook model

The Johnson-Cook damage model [57] depends on temperature, plastic strain, and strain rate. The damage evolution rule for the damage parameter (D) can be written as

$$\dot{D} = \frac{\dot{\varepsilon}_p^{\text{eq}}}{\varepsilon_p^f} \,, \quad \varepsilon_p^f = \left[D_1 + D_2 \exp\left(D_3 \sigma^\star\right)\right]\left[1 + D_4 \ln(\dot{\varepsilon}_p^\star)\right]\left[1 + D_5 T^\star\right] . \tag{15.3}$$

The damage parameter D 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. In the above equation,

$$\sigma^\star = \frac{\mathrm{tr}(\boldsymbol{\sigma})}{3\sigma_{\mathrm{eff}}} \tag{15.4}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress and σ_{eff} is the von Mises equivalent stress. The scaled plastic strain rate and temperature are defined as

$$\dot{\varepsilon}_p^\star = \frac{\dot{\varepsilon}_p^{\mathrm{eq}}}{\dot{\varepsilon}_{p0}} \,, \quad T^\star = \frac{T - T_0}{T_m - T_0} \tag{15.5}$$

where $\dot{\varepsilon}_{p0}$ is a reference strain rate and T_0 is a reference temperature, T_m is the melting temperature, and $\dot{\varepsilon}_p^{\mathrm{eq}}$ is the equivalent plastic strain rate.

When expressed in terms of the consistency parameter, the Johnson-Cook damage model has the form,

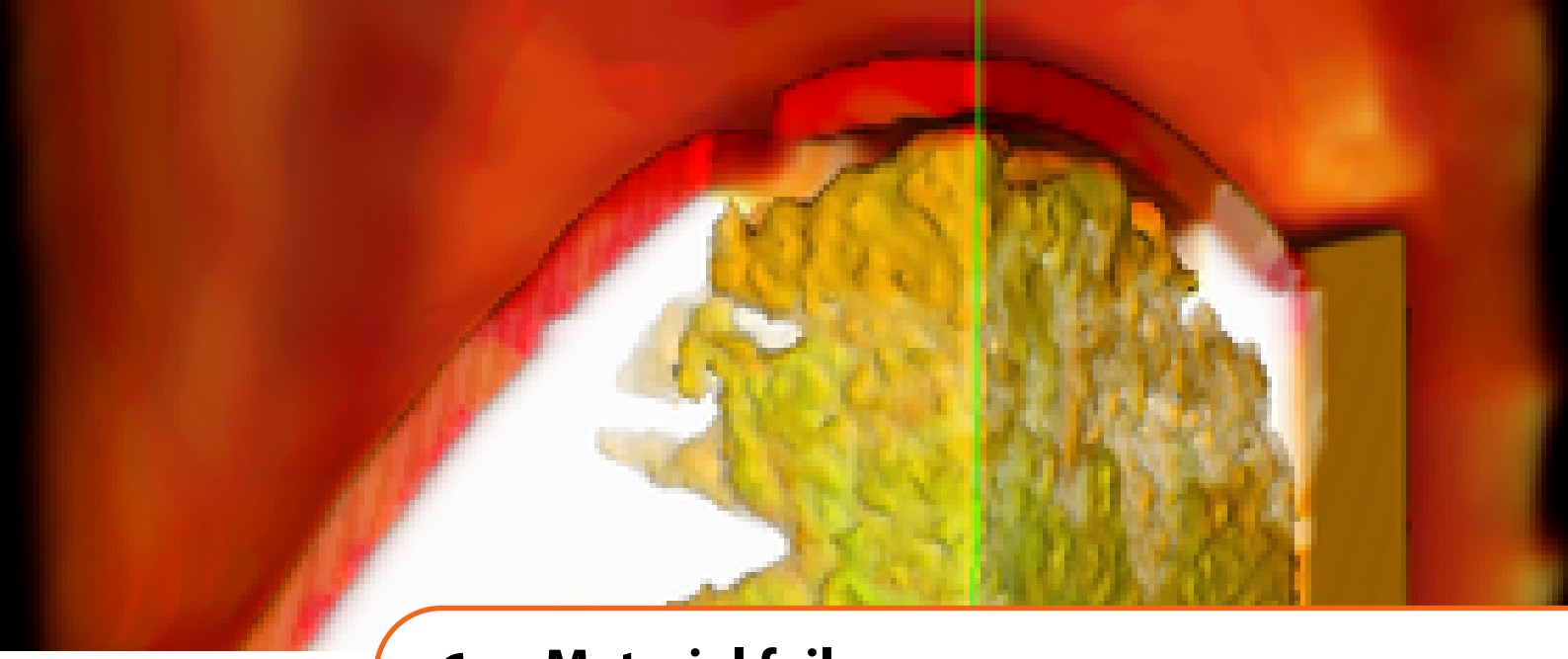
$$\dot{D} = \dot{\lambda} h_D \,, \quad h_D = \left[\left[D_1 + D_2 \exp\left(D_3 \sigma^\star\right)\right]\left[1 + D_4 \ln(\dot{\varepsilon}_p^\star)\right]\left[1 + D_5 T^\star\right]\right]^{-1} . \tag{15.6}$$

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>
```



16 — Material failure

16.1 Introduction

The primary technique used in Vaango to simulate failure is damage evolution. A particle is tagged as "failed" when its temperature is greater than the melting point of the material at the applied pressure. Failure is also flagged when the porosity of a particle is greater critical limit (typically 0.9) and the strain exceeds the fracture strain of the material.

An alternative approach that can be used in the metal plasticity models implemented in Vaango is to test material stability conditions to determine and propagate failure. Upon failure detection, 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 [58–60].

16.2 Erosion algorithm

In metal plasticity simulations, 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 [61] is satisfied. This condition can be written as

$$\left(\frac{\phi}{\phi_c}\right)^2 + \left(\frac{\varepsilon_p^{\text{eq}}}{\varepsilon_p^f}\right)^2 = 1 \tag{16.1}$$

  where ϕ is the current porosity, ϕ_c is the maximum allowable porosity, $\varepsilon_p^{\text{eq}}$ is the current equivalent plastic strain, and ε_p^f is the equivalent plastic strain at fracture.
- An alternative to ad-hoc damage criteria is to use the concept of material stability bifurcation to determine whether a particle has failed or not.

Since the material unloads 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 [62–64]. Viscoplastic stress update models or nonlocal/gradient plasticity models [65, 66] can be used to eliminate some of these effects. Such models that have been implemented in Vaango are discussed later.

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>
```

16.3 Material stability conditions

16.3.1 Drucker's condition

The simplest criterion that can be used is the Drucker stability postulate [67] 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

$$\dot{\boldsymbol{\sigma}} : \boldsymbol{d}^p \leq 0 \tag{16.2}$$

16.3.2 Acoustic tensor criterion

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 [68–70].

We assume that the strain is localized in a thin band with normal $\mathbf{n}$. The band is assumed be homogeneous but has slightly different material properties than the surrounding material.

To develop the bifurcation relations [69], assume that $\mathbf{v}^b$ is the velocity of a material point in the band (Ω_b) and $\mathbf{v}^o$ is the velocity of the material outside the band (Ω_o). The deformation of the material outside the band is assumed to be uniform. The deformation within the band is also assumed to be homogeneous.

We assume that stresses and rates of deformation have been rotated to the undeformed configuration using the polar decomposition of the deformation gradient.

Consider the case where the local coordinates of points in the band are expressed in terms of an orthonormal basis $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ where $\mathbf{e}_2 = \mathbf{n}$. Then a point $\mathbf{x}$ inside (or outside) the band can be expressed as $\mathbf{x} = x_i \mathbf{e}_i$.

Continuity and the homogeneity of deformation in the two regions requires that only the velocity in the $\mathbf{n}$ direction can be different in the two regions. This implies

$$\mathbf{v}^b(\mathbf{x}) - \mathbf{v}^o(\mathbf{x}) = \mathbf{f}(\mathbf{n} \cdot \mathbf{x}) = \mathbf{f}(x_2) \tag{16.3}$$

where $\mathbf{f}$ is an unknown function. The velocity gradient can be computed from the above relation as

$$\nabla \mathbf{v}^b = \nabla \mathbf{v}^o + \frac{d\mathbf{f}}{dx_2} \otimes \mathbf{n} = \nabla \mathbf{v}^o + \mathbf{q} \otimes \mathbf{n} \tag{16.4}$$

where

$$\mathbf{q} := \frac{d\mathbf{f}}{dx_2} = \begin{cases} \mathbf{o} & \text{for} \quad \mathbf{x} \in \Omega_o \\ \mathbf{q} & \text{for} \quad \mathbf{x} \in \Omega_b \end{cases} \tag{16.5}$$

The rate of deformation in the band is

$$\boldsymbol{d}^b = \tfrac{1}{2}\left[\nabla \mathbf{v}^b + (\nabla \mathbf{v}^b)^T\right] = \tfrac{1}{2}\left[\nabla \mathbf{v}^o + (\nabla \mathbf{v}^o)^T\right] + \tfrac{1}{2}(\mathbf{q}\otimes\mathbf{n} + \mathbf{n}\otimes\mathbf{q}) = \boldsymbol{d}^o + \tfrac{1}{2}(\mathbf{q}\otimes\mathbf{n} + \mathbf{n}\otimes\mathbf{q})\,. \tag{16.6}$$

The stress rate is related to the rate of deformation by

$$\dot{\boldsymbol{\sigma}}^o = \mathbb{C}^o : \boldsymbol{d}^o\,, \quad \dot{\boldsymbol{\sigma}}^b = \mathbb{C}^b : \boldsymbol{d}^b = \mathbb{C}^b : \boldsymbol{d}^o + \tfrac{1}{2}\mathbb{C}^b : (\mathbf{q}\otimes\mathbf{n} + \mathbf{n}\otimes\mathbf{q})\,. \tag{16.7}$$

The minor symmetry of $\mathbb{C}$ implies that

$$\dot{\boldsymbol{\sigma}}^b = \mathbb{C}^b : \boldsymbol{d}^o + \mathbb{C}^b : (\mathbf{q}\otimes\mathbf{n})\,. \tag{16.8}$$

Homogeneity of the deformation also implies that

$$\mathbf{n}\cdot\boldsymbol{\sigma}^b = \mathbf{n}\cdot\boldsymbol{\sigma}^o\,. \tag{16.9}$$

Taking the material time derivative of the above gives

$$\dot{\mathbf{n}}\cdot\boldsymbol{\sigma}^b + \mathbf{n}\cdot\dot{\boldsymbol{\sigma}}^b = \dot{\mathbf{n}}\cdot\boldsymbol{\sigma}^o + \mathbf{n}\cdot\dot{\boldsymbol{\sigma}}^o\,. \tag{16.10}$$

We need an expression for $\dot{\mathbf{n}}$. To find that, note that if $\mathbf{n}_o$ and $\mathbf{n}$ are the unit normals to the band in the reference and current configurations, using Nanson's formula, we have

$$\mathbf{n}da = J(\boldsymbol{F}^{-T}\cdot\mathbf{n}_o)dA\,, \quad J = \det\boldsymbol{F}\,. \tag{16.11}$$

Taking the time derivative of (16.11),

$$\dot{\mathbf{n}}da + \mathbf{n}\dot{da} = \frac{dJ}{dt}(\boldsymbol{F}^{-T}\cdot\mathbf{n}_o)dA + J\left(\frac{d\boldsymbol{F}^{-1}}{dt}\right)^T\cdot\mathbf{n}_o dA \tag{16.12}$$

For the derivative of J, we have

$$\frac{dJ}{dt} = \frac{\partial J}{\partial \boldsymbol{F}} : \dot{\boldsymbol{F}} = J\boldsymbol{F}^{-T} : \dot{\boldsymbol{F}} = J\mathrm{tr}(\dot{\boldsymbol{F}}\cdot\boldsymbol{F}^{-1}) = J\mathrm{tr}(\nabla\mathbf{v})\,. \tag{16.13}$$

We can also show that

$$\frac{d\boldsymbol{F}^{-1}}{dt} = -\boldsymbol{F}^{-1}\cdot\dot{\boldsymbol{F}}\cdot\boldsymbol{F}^{-1} = -\boldsymbol{F}^{-1}\cdot\nabla\mathbf{v}\,. \tag{16.14}$$

Therefore, using (16.11),

$$\begin{aligned}\dot{\mathbf{n}}da + \mathbf{n}\dot{da} &= J\mathrm{tr}(\nabla\mathbf{v})(\boldsymbol{F}^{-T}\cdot\mathbf{n}_o)dA - J(\nabla\mathbf{v})^T\cdot\boldsymbol{F}^{-T}\cdot\mathbf{n}_o dA \\ &= \mathbf{n}\cdot\left[\mathrm{tr}(\nabla\mathbf{v})\boldsymbol{I} - \nabla\mathbf{v}\right]da\,.\end{aligned} \tag{16.15}$$

To find $\dot{da}$, we compute a dot product of both sides of (16.11) to get

$$da^2 = J^2(\boldsymbol{F}^{-T}\cdot\mathbf{n}_o)\cdot(\boldsymbol{F}^{-T}\cdot\mathbf{n}_o)dA^2 = J^2\mathbf{n}_o\cdot(\boldsymbol{F}^{-1}\cdot\boldsymbol{F}^{-T})\cdot\mathbf{n}_o dA^2\,. \tag{16.16}$$

The material time derivative of the above expression is

$$2da\dot{da} = 2J\frac{dJ}{dt}\mathbf{n}_o\cdot(\boldsymbol{F}^{-1}\cdot\boldsymbol{F}^{-T})\cdot\mathbf{n}_o dA^2 + J^2\mathbf{n}_o\cdot\frac{d}{dt}(\boldsymbol{F}^{-1}\cdot\boldsymbol{F}^{-T})\cdot\mathbf{n}_o dA^2 \tag{16.17}$$

For the derivative of the $\boldsymbol{F}^{-1}$ product, we have

$$\begin{aligned}\frac{d}{dt}(\boldsymbol{F}^{-1}\cdot\boldsymbol{F}^{-T}) &= \frac{d\boldsymbol{F}^{-1}}{dt}\cdot\boldsymbol{F}^{-T}+\boldsymbol{F}^{-1}\cdot\left(\frac{d\boldsymbol{F}^{-1}}{dt}\right)^{T} = -\boldsymbol{F}^{-1}\cdot\dot{\boldsymbol{F}}\cdot\boldsymbol{F}^{-1}\cdot\boldsymbol{F}^{-T}-\boldsymbol{F}^{-1}\cdot\boldsymbol{F}^{-T}\cdot\dot{\boldsymbol{F}}^{T}\cdot\boldsymbol{F}^{-T}\\ &= -\boldsymbol{F}^{-1}\cdot\nabla\mathbf{v}\cdot\boldsymbol{F}^{-T}-\boldsymbol{F}^{-1}\cdot(\nabla\mathbf{v})^{T}\cdot\boldsymbol{F}^{-T} = -2\boldsymbol{F}^{-1}\cdot\boldsymbol{d}\cdot\boldsymbol{F}^{-T}.\end{aligned} \tag{16.18}$$

Substitution into (16.17) gives

$$\begin{aligned}da\dot{da} &= J^{2}\mathrm{tr}(\nabla\mathbf{v})\mathbf{n}_{o}\cdot(\boldsymbol{F}^{-1}\cdot\boldsymbol{F}^{-T})\cdot\mathbf{n}_{o}dA^{2}-J^{2}\mathbf{n}_{o}\cdot(\boldsymbol{F}^{-1}\cdot\boldsymbol{d}\cdot\boldsymbol{F}^{-T})\cdot\mathbf{n}_{o}dA^{2}\\ &= \mathrm{tr}(\nabla\mathbf{v})(J\boldsymbol{F}^{-T}\cdot\mathbf{n}_{o}dA)\cdot(J\boldsymbol{F}^{-T}\cdot\mathbf{n}_{o}dA)-(J\boldsymbol{F}^{-T}\cdot\mathbf{n}_{o}dA)\cdot\boldsymbol{d}\cdot(J\boldsymbol{F}^{-T}\cdot\mathbf{n}_{o}dA)\\ &= \mathrm{tr}(\nabla\mathbf{v})\mathbf{n}\cdot\mathbf{n}da^{2}-\mathbf{n}\cdot\boldsymbol{d}\cdot\mathbf{n}da^{2} = \mathrm{tr}(\nabla\mathbf{v})da^{2}-\mathbf{n}\cdot\boldsymbol{d}\cdot\mathbf{n}da^{2}.\end{aligned} \tag{16.19}$$

Therefore,

$$\dot{da} = \left[\mathrm{tr}(\nabla\mathbf{v})-\mathbf{n}\cdot\boldsymbol{d}\cdot\mathbf{n}\right]da = \left[\mathrm{tr}(\nabla\mathbf{v})-\mathbf{n}\cdot\nabla\mathbf{v}\cdot\mathbf{n}\right]da. \tag{16.20}$$

Plugging (16.20) into (16.15), we have

$$\dot{\mathbf{n}}da+\mathbf{n}\left[\mathrm{tr}(\nabla\mathbf{v})-\mathbf{n}\cdot\boldsymbol{d}\cdot\mathbf{n}\right]da = \mathbf{n}\cdot\left[\mathrm{tr}(\nabla\mathbf{v})\boldsymbol{I}-\nabla\mathbf{v}\right]da. \tag{16.21}$$

That gives us the expression for $\dot{\mathbf{n}}$ that we seek,

$$\dot{\mathbf{n}} = \mathbf{n}\cdot\left[\boldsymbol{d}\cdot(\mathbf{n}\otimes\mathbf{n})-\nabla\mathbf{v}\right] = \mathbf{n}\cdot\nabla\mathbf{v}\cdot(\mathbf{n}\otimes\mathbf{n}-\boldsymbol{I}). \tag{16.22}$$

Using (16.22) in (16.10), we have

$$\mathbf{n}\cdot\left[(\dot{\boldsymbol{\sigma}}^{b}-\dot{\boldsymbol{\sigma}}^{\circ})+\nabla\mathbf{v}^{b}\cdot(\mathbf{n}\otimes\mathbf{n}-\boldsymbol{I})\cdot(\boldsymbol{\sigma}^{b}-\boldsymbol{\sigma}^{\circ})\right] = 0. \tag{16.23}$$

Substituting equations (16.7), (16.8), and (16.4),

$$\mathbf{n}\cdot\left[(\mathbb{C}^{b}:\boldsymbol{d}^{\circ}+\mathbb{C}^{b}:(\mathbf{q}\otimes\mathbf{n})-\mathbb{C}^{\circ}:\boldsymbol{d}^{\circ})+(\nabla\mathbf{v}^{\circ}+\mathbf{q}\otimes\mathbf{n})\cdot(\mathbf{n}\otimes\mathbf{n}-\boldsymbol{I})\cdot(\boldsymbol{\sigma}^{b}-\boldsymbol{\sigma}^{\circ})\right] = 0. \tag{16.24}$$

Using the symmetry of stress and the projection $\mathbf{n}\otimes\mathbf{n}-\boldsymbol{I}$, we can reorganize the above expression into

$$\mathbf{n}\cdot\left[\left(\mathbb{C}^{b}+(\boldsymbol{\sigma}^{b}-\boldsymbol{\sigma}^{\circ})\cdot(\mathbf{n}\otimes\mathbf{n}-\boldsymbol{I})\cdot\mathbb{I}\right):(\mathbf{q}\otimes\mathbf{n})+(\mathbb{C}^{b}-\mathbb{C}^{\circ}):\boldsymbol{d}^{\circ}+(\boldsymbol{\sigma}^{b}-\boldsymbol{\sigma}^{\circ})\cdot(\mathbf{n}\otimes\mathbf{n}-\boldsymbol{I})\cdot\nabla\mathbf{v}^{\circ}\right] = 0. \tag{16.25}$$

Further rearrangement leads to

$$\left[\mathbf{n}\cdot\left(\mathbb{C}^{b}+(\boldsymbol{\sigma}^{b}-\boldsymbol{\sigma}^{\circ})\cdot(\mathbf{n}\otimes\mathbf{n}-\boldsymbol{I})\cdot\mathbb{I}\right)\cdot\mathbf{n}\right]\cdot\mathbf{q} = -\mathbf{n}\cdot\left[(\mathbb{C}^{b}-\mathbb{C}^{\circ}):\boldsymbol{d}^{\circ}+(\boldsymbol{\sigma}^{b}-\boldsymbol{\sigma}^{\circ})\cdot(\mathbf{n}\otimes\mathbf{n}-\boldsymbol{I})\cdot\nabla\mathbf{v}^{\circ}\right] \tag{16.26}$$

This equation has a solution ($\mathbf{q}$) only if

$$\det\left[\mathbf{n}\cdot\left(\mathbb{C}^{b}+(\boldsymbol{\sigma}^{b}-\boldsymbol{\sigma}^{\circ})\cdot(\mathbf{n}\otimes\mathbf{n}-\boldsymbol{I})\cdot\mathbb{I}\right)\cdot\mathbf{n}\right]\neq 0. \tag{16.27}$$

The canonical bifurcation condition is obtained if $\boldsymbol{\sigma}^{b}=\boldsymbol{\sigma}^{\circ}$:

$$\det(\boldsymbol{A}) := \det(\mathbf{n}\cdot\mathbb{C}\cdot\mathbf{n}) = 0 \tag{16.28}$$

where $\boldsymbol{A}$ is the acoustic tensor.

Evaluation of the acoustic tensor requires a search for a normal vector around the material point and is therefore computationally expensive.

16.3.3 Becker's simplification

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 [71].

Let the principal stresses be $\sigma_1 > \sigma_2 > \sigma_3$, and the corresponding principal directions (eigenvectors) are $\boldsymbol{E}_1, \boldsymbol{E}_2, \boldsymbol{E}_3$. We can express the unit normal to the band in this basis as $\mathbf{n} = n_i \boldsymbol{E}_i$.

The components of the tangent modulus in this coordinate system are given by

$$C'_{ijk\ell} = Q_{im} Q_{jn} Q_{kp} Q_{\ell q} C_{mnpq} \,. \tag{16.29}$$

where the 3×3 matrix used for this coordinate transformation, $\boldsymbol{Q}$, is given by

$$\boldsymbol{Q}^T := \begin{bmatrix} \boldsymbol{E}_1 & \boldsymbol{E}_2 & \boldsymbol{E}_3 \end{bmatrix} \tag{16.30}$$

Then the acoustic tensor in (16.28) has the components

$$A_{jk} = C'_{ijk\ell} n_i n_\ell \tag{16.31}$$

If $\det(A_{jk}) = 0$, then $q_j = df_j/dx_2$ can be arbitrary and there is a possibility of strain localization. Also, this condition indicates when a material transitions from stable behavior where $\det(A_{jk} > 0$. 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.

Becker's simplification is to consider only selected components of the acoustic tensor by assuming that the stress state in the band can be approximated as a planar tension problem. Then the acoustic tensor takes the form

$$\boldsymbol{A} = \begin{bmatrix} C'_{1111} n_1^2 + C'_{3113} n_3^2 & 0 & (C'_{1133} + C'_{3131}) n_1 n_3 \\ 0 & C'_{1221} n_1^2 + C'_{3223} n_3^2 & 0 \\ (C'_{3311} + C'_{1313}) n_1 n_3 & 0 & C'_{1331} n_1^2 + C'_{3333} n_3^2 \end{bmatrix} \tag{16.32}$$

Without loss of generality, we can divide this matrix by n_1^2 to get

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} C'_{1111} + C'_{3113} \frac{n_3^2}{n_1^2} & 0 & (C'_{1133} + C'_{3131}) \frac{n_3}{n_1} \\ 0 & C'_{1221} + C'_{3223} \frac{n_3^2}{n_1^2} & 0 \\ (C'_{3311} + C'_{1313}) \frac{n_3}{n_1} & 0 & C'_{1331} + C'_{3333} \frac{n_3^2}{n_1^2} \end{bmatrix} \tag{16.33}$$

Let $\alpha := n_3/n_1$. Then we can write

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} C'_{1111} + C'_{3113} \alpha^2 & 0 & (C'_{1133} + C'_{3131}) \alpha \\ 0 & C'_{1221} + C'_{3223} \alpha^2 & 0 \\ (C'_{3311} + C'_{1313}) \alpha & 0 & C'_{1331} + C'_{3333} \alpha^2 \end{bmatrix} \tag{16.34}$$

and we have

$$\det(\tilde{\boldsymbol{A}}) = (C'_{1111} + C'_{3113} \alpha^2) \alpha^2 (C'_{1331} + C'_{3333} \alpha^2) - (C'_{1133} + C'_{3131}) \alpha (C'_{1221} + C'_{3223} \alpha^2) (C'_{3311} + C'_{1313}) \alpha \tag{16.35}$$

Setting the determinant to zero allows us to get the following quadratic equation in $\beta := \alpha^2$:

$$(C'_{1111} + C'_{3113} \beta)(C'_{1331} + C'_{3333} \beta) - (C'_{1133} + C'_{3131})(C'_{1221} + C'_{3223} \beta)(C'_{3311} + C'_{1313}) = 0 \,. \tag{16.36}$$

We can express the above in Voigt notation (convention 11, 22, 33, 23, 31, 12) as

$$(\hat{C}_{11} + \hat{C}_{55} \beta)(\hat{C}_{55} + \hat{C}_{33} \beta) - (\hat{C}_{13} + \hat{C}_{55})(\hat{C}_{66} + \hat{C}_{44} \beta)(\hat{C}_{31} + \hat{C}_{55}) = 0 \,. \tag{16.37}$$

or

$$a_1\beta^2 + a_2\beta + a_3 = 0 \tag{16.38}$$

where,

$$\begin{aligned} a_1 &= \hat{C}_{33}\hat{C}_{55} \\ a_2 &= \hat{C}_{11}\hat{C}_{33} - \hat{C}_{13}\hat{C}_{31}\hat{C}_{44} - \hat{C}_{13}\hat{C}_{44}\hat{C}_{55} - \hat{C}_{31}\hat{C}_{44}\hat{C}_{55} - \hat{C}_{44}\hat{C}_{55}^2 + \hat{C}_{55}^2 \\ a_3 &= \hat{C}_{11}\hat{C}_{55} - \hat{C}_{13}\hat{C}_{31}\hat{C}_{66} - \hat{C}_{13}\hat{C}_{55}\hat{C}_{66} - \hat{C}_{31}\hat{C}_{55}\hat{C}_{66}\,. \end{aligned} \tag{16.39}$$

The four roots are

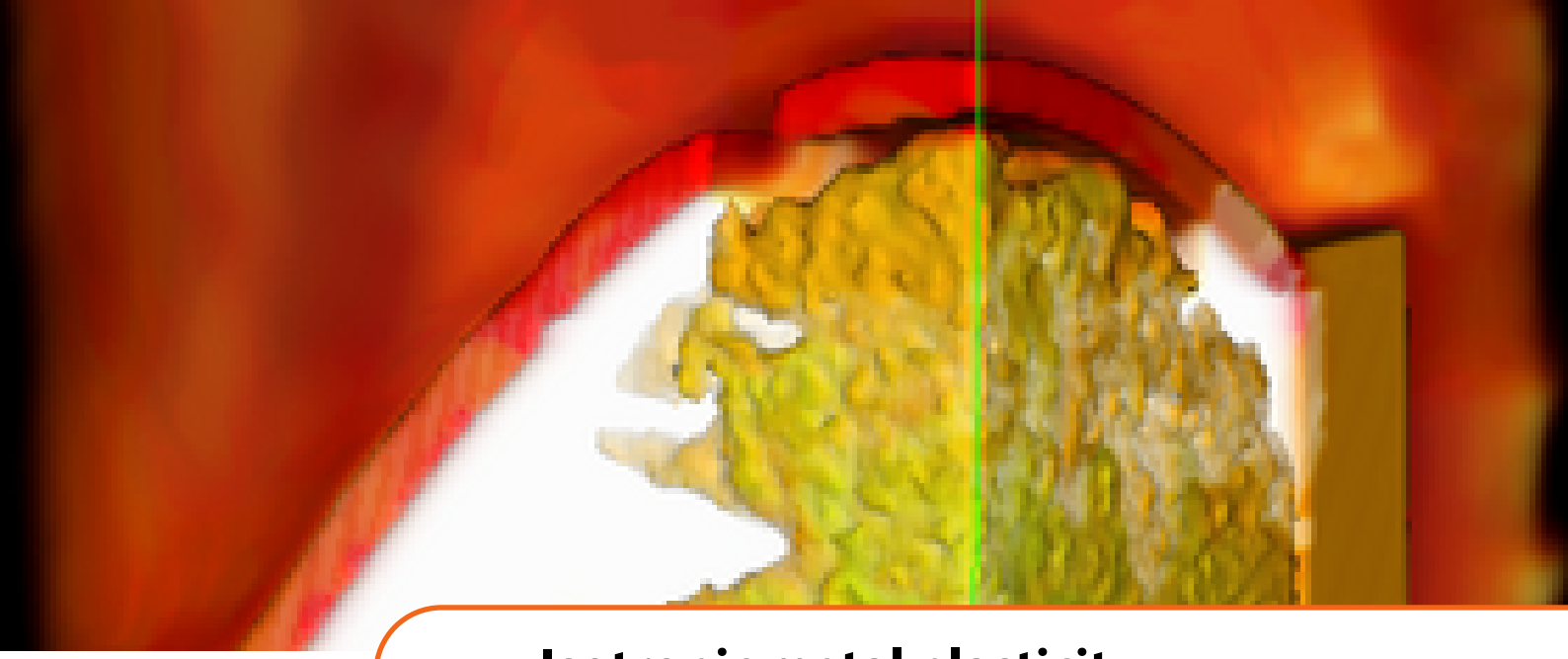
$$\frac{n_3}{n_1} = \pm\sqrt{\frac{-a_2 \pm \sqrt{a_2^2 - 4a_1a_3}}{2a_1}}\,. \tag{16.40}$$

If there are no real roots, a band cannot form and there is no bifurcation. If there are four real roots then bifurcation is possible. Two real roots indicate an intermediate condition that may not be realized in practice but is considered stable in Vaango .

More explicitly, for unstable deformation,

$$a_2^2 - 4a_1a_3 \geq 0 \quad \text{and} \quad \frac{-a_2 \pm \sqrt{a_2^2 - 4a_1a_3}}{2a_1} \geq 0\,. \tag{16.41}$$

If these conditions are satisfied, the MPM particle is assumed to have failed.



17 — Isotropic metal plasticity

The deformation gradient ($\boldsymbol{F}$) can be decomposed into a rotation tensor ($\boldsymbol{R}$) and a stretch tensor ($\boldsymbol{U}$) with the polar decomposition $\boldsymbol{F} = \boldsymbol{R} \cdot \boldsymbol{U}$. In the isotropic metal plasticity model implemented in VAANGO, $\boldsymbol{R}$ is used to rotate the stress ($\boldsymbol{\sigma}$) and the rate of deformation ($\boldsymbol{d}$) into the unrotated configuration before the updated stress is computed:

$$\hat{\boldsymbol{\sigma}} = \boldsymbol{R}^T \cdot \boldsymbol{\sigma} \cdot \boldsymbol{R}\,; \quad \dot{\boldsymbol{\varepsilon}} = \boldsymbol{R}^T \cdot \boldsymbol{d} \cdot \boldsymbol{R} \tag{17.1}$$

where $\dot{\boldsymbol{\varepsilon}}$ is a "natural" strain rate. After the stress has been updated, it is rotated back using

$$\boldsymbol{\sigma} = \boldsymbol{R} \cdot \hat{\boldsymbol{\sigma}} \cdot \boldsymbol{R}^T\,. \tag{17.2}$$

In the following discussion, all equations should be treated as referring to the hatted quantities even though we drop the hats for convenience.

17.1 The model

The metal plasticity model assumes that we know the total strain rate ($\dot{\boldsymbol{\varepsilon}}^t$) and that this strain rate can be decomposed into a mechanical component ($\dot{\boldsymbol{\varepsilon}}$) and a thermal expansion component ($\dot{\boldsymbol{\varepsilon}}^\alpha$):

$$\dot{\boldsymbol{\varepsilon}}^t = \dot{\boldsymbol{\varepsilon}} + \dot{\boldsymbol{\varepsilon}}^\alpha\,. \tag{17.3}$$

The thermal expansion component is assumed to be of the rate form

$$\dot{\boldsymbol{\varepsilon}}^\alpha = \frac{\partial \boldsymbol{\varepsilon}^\alpha}{\partial T}\dot{T} = \alpha \dot{T} \tag{17.4}$$

where T is the temperature and α is a coefficient of thermal expansion. Then the mechanical strain rate can be expressed as

$$\dot{\boldsymbol{\varepsilon}} = \dot{\boldsymbol{\varepsilon}}^t - \alpha \dot{T}\,. \tag{17.5}$$

The primary function of the metal plasticity model is to compute the stress when a mechanical strain rate $\dot{\boldsymbol{\varepsilon}}$ is given, which we assume can be additively decomposed into elastic ($\dot{\boldsymbol{\varepsilon}}^e$) and plastic ($\dot{\boldsymbol{\varepsilon}}^p$) parts:

$$\dot{\boldsymbol{\varepsilon}} = \dot{\boldsymbol{\varepsilon}}^e + \dot{\boldsymbol{\varepsilon}}^p\,. \tag{17.6}$$

The Cauchy stress ($\boldsymbol{\sigma}$) is decomposed into volumetric and deviatoric parts:

$$\boldsymbol{\sigma} = p\,\boldsymbol{I} + \boldsymbol{s} \quad \text{where} \quad p = \tfrac{1}{3}\,\mathrm{tr}(\boldsymbol{\sigma}) \quad \text{and} \quad \boldsymbol{s} = \mathrm{dev}(\boldsymbol{\sigma}) = \boldsymbol{\sigma} - \tfrac{1}{3}\mathrm{tr}(\boldsymbol{\sigma})\,. \tag{17.7}$$

In the above $p = \sigma_m$ is the mean stress and $\boldsymbol{s}$ is the deviatoric stress. An alternative decomposition that can be used for the isotropic metal plasticity models implemented in Vaango is

$$\boldsymbol{\sigma} = \sigma_p \hat{\boldsymbol{I}} + \sigma_s \hat{\boldsymbol{s}}\,, \quad \hat{\boldsymbol{I}} = \frac{1}{\sqrt{3}}\boldsymbol{I}\,, \quad \hat{\boldsymbol{s}} = \frac{\boldsymbol{s}}{\|\boldsymbol{s}\|}\,, \quad \sigma_p = \sqrt{3}p\,, \quad \sigma_s = \|\boldsymbol{s}\|\,. \tag{17.8}$$

This decomposition is useful because $\hat{\boldsymbol{I}}$ and $\hat{\boldsymbol{s}}$ for a basis that can be used to express several other quantities in the metal plasticity models implemented in Vaango . The time derivative of stress can then be expressed as

$$\dot{\boldsymbol{\sigma}} = \dot{p}\,\boldsymbol{I} + \dot{\boldsymbol{s}} = \dot{\sigma}_p\,\hat{\boldsymbol{I}} + \dot{\sigma}_s\,\hat{\boldsymbol{s}}\,. \tag{17.9}$$

The isotropy of the material allows us to compute the mean stress using an equation of state if desired. The deviatoric stress is computed using a rate-form stress-strain relation. For convenience, we assume that rate-form relations are used for both the mean stress and the deviatoric stress.

17.1.1 Purely elastic loading/unloading

The elastic constitutive relation is assumed to be of the form

$$\dot{\boldsymbol{\sigma}}^e = \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\varepsilon}^e} : \dot{\boldsymbol{\varepsilon}}^e = \mathbb{C}^e : \dot{\boldsymbol{\varepsilon}}^e\,, \quad \mathbb{C}^e = \left(\kappa - \tfrac{2}{3}\mu\right)\boldsymbol{I}\otimes\boldsymbol{I} + 2\mu\,\mathrm{symm}(\mathbb{I}) \tag{17.10}$$

or,

$$\dot{\boldsymbol{\sigma}}^e = \left(\kappa - \tfrac{2}{3}\mu\right)\mathrm{tr}(\dot{\boldsymbol{\varepsilon}}^e)\boldsymbol{I} + 2\mu\,\dot{\boldsymbol{\varepsilon}}^e\,. \tag{17.11}$$

In the above, $\mu(\rho, p, T, \phi, D)$ is the shear modulus, $\kappa(\rho, p, T, \phi, D)$ is the tangent bulk modulus, $\boldsymbol{I}$ is the second-order identity tensor and $\mathbb{I}$ is the fourth-order identity tensor. Also, ρ is the mass density, p is the pressure, T is the current temperature, ϕ is the current porosity and D is a scalar damage parameter.

The inverse relationship is

$$\dot{\boldsymbol{\varepsilon}}^e = \mathbb{S}^e : \dot{\boldsymbol{\sigma}}^e\,, \quad \mathbb{S}^e = \frac{1}{3}\left(\frac{1}{3\kappa} - \frac{1}{2\mu}\right)\boldsymbol{I}\otimes\boldsymbol{I} + \frac{1}{2\mu}\mathrm{symm}(\mathbb{I}) \tag{17.12}$$

Using the decomposition (17.8), we can write

$$\dot{\boldsymbol{\varepsilon}}^e = \mathbb{S}^e : \left(\dot{\sigma}_p^e\,\hat{\boldsymbol{I}} + \dot{\sigma}_s^e\,\hat{\boldsymbol{s}}^e\right) = \frac{\dot{\sigma}_p^e}{3\kappa}\hat{\boldsymbol{I}} + \frac{\dot{\sigma}_s^e}{2\mu}\hat{\boldsymbol{s}}^e\,. \tag{17.13}$$

17.1.2 Yield condition

The isotropic metal plasticity yield conditions implemented in Vaango have the form

$$f(\boldsymbol{\sigma}_\beta, \varepsilon_p^{\mathrm{eq}}, \dot{\varepsilon}_p^{\mathrm{eq}}, \phi, D, T, \dot{\varepsilon}^{\mathrm{eq}}, \dots) = 0\,, \quad \boldsymbol{\sigma}_\beta := \boldsymbol{\sigma} - \boldsymbol{\beta}\,. \tag{17.14}$$

The quantity $\boldsymbol{\sigma}_\beta$ is further decomposed into isotropic and deviatoric parts:

$$\boldsymbol{\sigma}_\beta = p_\beta \boldsymbol{I} + \boldsymbol{\xi} \tag{17.15}$$

where $\boldsymbol{\xi} = \mathrm{dev}(\boldsymbol{\sigma}_\beta)$ and $p_\beta = \mathrm{tr}(\boldsymbol{\sigma}_\beta)/3$. Most of the metal yield conditions in Vaango use this notation. Derivatives of f with respect to the stress ($\boldsymbol{\sigma}$) can therefore be expressed as

$$\frac{\partial f}{\partial \boldsymbol{\sigma}} = \frac{\partial f}{\partial \boldsymbol{\sigma}_\beta} : \frac{\partial \boldsymbol{\sigma}_\beta}{\partial \boldsymbol{\sigma}} = \frac{\partial f}{\partial \boldsymbol{\sigma}_\beta} = \frac{\partial f}{\partial p_\beta}\frac{\partial p_\beta}{\partial \boldsymbol{\sigma}_\beta} + \frac{\partial f}{\partial \boldsymbol{\xi}} : \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\sigma}_\beta} = \tfrac{1}{3}\frac{\partial f}{\partial p_\beta}\boldsymbol{I} + \frac{\partial f}{\partial \boldsymbol{\xi}} - \tfrac{1}{3}\mathrm{tr}\left(\frac{\partial f}{\partial \boldsymbol{\xi}}\right)\boldsymbol{I}\,. \tag{17.16}$$

The isotropic metal yield conditions in VAANGO are expressed in terms of $\sigma^{\xi}_{\text{eff}} = \sqrt{3J^{\xi}_2} = \sqrt{\frac{3}{2}\boldsymbol{\xi} : \boldsymbol{\xi}}$. Therefore,

$$\frac{\partial f}{\partial \boldsymbol{\sigma}} = \tfrac{1}{3}\frac{\partial f}{\partial p_\beta}\boldsymbol{I} + \frac{\partial f}{\partial \sigma^{\xi}_{\text{eff}}}\frac{\partial \sigma^{\xi}_{\text{eff}}}{\partial \boldsymbol{\xi}} = \tfrac{1}{3}\frac{\partial f}{\partial p_\beta}\boldsymbol{I} + \sqrt{\tfrac{3}{2}}\frac{\partial f}{\partial \sigma^{\xi}_{\text{eff}}}\frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|} \,. \tag{17.17}$$

If we express both $\boldsymbol{I}$ and $\boldsymbol{\xi}$ in terms of the basis $\hat{\boldsymbol{I}}$ and $\hat{\boldsymbol{s}}$, we have

$$\boldsymbol{I} = \sqrt{3}\hat{\boldsymbol{I}} \,, \quad \boldsymbol{\xi} = \xi_s \hat{\boldsymbol{s}} \,, \quad \xi_s = \boldsymbol{\xi} : \hat{\boldsymbol{s}} \quad \text{and} \quad \|\boldsymbol{\xi}\| = \xi_s \quad \Longrightarrow \quad \frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|} = \hat{\boldsymbol{s}} \,. \tag{17.18}$$

Therefore,

$$\boldsymbol{N} = \frac{\partial f}{\partial \boldsymbol{\sigma}} = \frac{1}{\sqrt{3}}\frac{\partial f}{\partial p_\beta}\hat{\boldsymbol{I}} + \sqrt{\tfrac{3}{2}}\frac{\partial f}{\partial \sigma^{\xi}_{\text{eff}}}\hat{\boldsymbol{s}} \,, \quad \|\boldsymbol{N}\| = \sqrt{\tfrac{1}{3}\left(\frac{\partial f}{\partial p_\beta}\right)^2 + \tfrac{3}{2}\left(\frac{\partial f}{\partial \sigma^{\xi}_{\text{eff}}}\right)^2} \,, \quad \hat{\boldsymbol{N}} = \frac{\boldsymbol{N}}{\|\boldsymbol{N}\|} \,. \tag{17.19}$$

The Kuhn-Tucker loading-unloading conditions are

$$\dot{\lambda} \geq 0 \,; \quad f \leq 0 \,; \quad \dot{\lambda}\, f = 0 \tag{17.20}$$

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

17.1.3 Flow rule

We assume that the plastic rate of deformation is given by the flow rule

$$\dot{\boldsymbol{\varepsilon}}^p = \dot{\lambda}\, \hat{\boldsymbol{M}} \,. \tag{17.21}$$

For the isotropic metal plasticity models in VAANGO , we assume associated plasticity:

$$\dot{\boldsymbol{\varepsilon}}^p = \dot{\lambda}\, \hat{\boldsymbol{N}} = \frac{\dot{\lambda}}{\|\boldsymbol{N}\|}\left[\tfrac{1}{\sqrt{3}}\frac{\partial f}{\partial p_\beta}\hat{\boldsymbol{I}} + \sqrt{\tfrac{3}{2}}\frac{\partial f}{\partial \sigma^{\xi}_{\text{eff}}}\hat{\boldsymbol{s}}\right] \,. \tag{17.22}$$

17.1.4 Isotropic and kinematic hardening/softening rules

The equivalent plastic strain ($\varepsilon^{\text{eq}}_p$) evolves according to the relation

$$\dot{\varepsilon}^{\text{eq}}_p = \dot{\lambda}\, h^{\varepsilon_p} \,. \tag{17.23}$$

The back stress ($\boldsymbol{\beta}$) evolves according to the relation

$$\dot{\boldsymbol{\beta}} = \dot{\lambda}\, \boldsymbol{h}^{\beta} \,. \tag{17.24}$$

The porosity (ϕ) is assumed to evolve according to the relation

$$\dot{\phi} = \dot{\lambda}\, h^{\phi} \,. \tag{17.25}$$

The damage parameter (D) evolves as

$$\dot{D} = \dot{\lambda}\, h^{D} \,. \tag{17.26}$$

The temperature (T_p) due to plastic dissipation evolves as

$$\dot{T}_p = \dot{\lambda}\, h^{T} \,. \tag{17.27}$$

17.1.5 Elastic-plastic loading/unloading

During purely elastic loading and unloading

$$\dot\lambda = 0\,,\quad \dot{\boldsymbol\varepsilon}^p = \mathbf{0}\,,\quad \dot{\boldsymbol\varepsilon} = \dot{\boldsymbol\varepsilon}^e\,. \tag{17.28}$$

In that situation, the stress is updated using (17.10).

However, during elastic-plastic deformation, $\dot\lambda > 0$, and we have

$$\begin{aligned}\dot{\boldsymbol\sigma} &= \frac{\partial\boldsymbol\sigma}{\partial\boldsymbol\varepsilon^e} : \dot{\boldsymbol\varepsilon}^e + \frac{\partial\boldsymbol\sigma}{\partial\boldsymbol\beta} : \dot{\boldsymbol\beta} + \frac{\partial\boldsymbol\sigma}{\partial\varepsilon_p^{\text{eq}}}\dot\varepsilon_p^{\text{eq}} + \frac{\partial\boldsymbol\sigma}{\partial\phi}\dot\phi + \frac{\partial\boldsymbol\sigma}{\partial D}\dot D + \frac{\partial\boldsymbol\sigma}{\partial T_p}\dot T_p \\ &= \mathbb{C}^e : \dot{\boldsymbol\varepsilon}^e + \dot\lambda\left[\frac{\partial\boldsymbol\sigma}{\partial\boldsymbol\beta} : \boldsymbol h^\beta + \frac{\partial\boldsymbol\sigma}{\partial\varepsilon_p^{\text{eq}}}h^{\varepsilon_p} + \frac{\partial\boldsymbol\sigma}{\partial\phi}h^\phi + \frac{\partial\boldsymbol\sigma}{\partial D}h^D + \frac{\partial\boldsymbol\sigma}{\partial T_p}h^T\right] \\ &= \mathbb{C}^e : \dot{\boldsymbol\varepsilon} - \dot\lambda\left[\mathbb{C}^e : \hat{\boldsymbol M} - \frac{\partial\boldsymbol\sigma}{\partial\boldsymbol\beta} : \boldsymbol h^\beta - \frac{\partial\boldsymbol\sigma}{\partial\varepsilon_p^{\text{eq}}}h^{\varepsilon_p} - \frac{\partial\boldsymbol\sigma}{\partial\phi}h^\phi - \frac{\partial\boldsymbol\sigma}{\partial D}h^D - \frac{\partial\boldsymbol\sigma}{\partial T_p}h^T\right]\end{aligned} \tag{17.29}$$

Define

$$\boldsymbol P := \mathbb{C}^e : \hat{\boldsymbol M} - \frac{\partial\boldsymbol\sigma}{\partial\boldsymbol\beta} : \boldsymbol h^\beta - \frac{\partial\boldsymbol\sigma}{\partial\varepsilon_p^{\text{eq}}}h^{\varepsilon_p} - \frac{\partial\boldsymbol\sigma}{\partial\phi}h^\phi - \frac{\partial\boldsymbol\sigma}{\partial D}h^D - \frac{\partial\boldsymbol\sigma}{\partial T_p}h^T\,. \tag{17.30}$$

Then,

$$\dot{\boldsymbol\sigma} = \mathbb{C}^e : \dot{\boldsymbol\varepsilon} - \dot\lambda\boldsymbol P = \dot{\boldsymbol\sigma}^{\text{trial}} - \dot\lambda\boldsymbol P \quad\text{where}\quad \dot{\boldsymbol\sigma}^{\text{trial}} := \mathbb{C}^e : \dot{\boldsymbol\varepsilon}\,. \tag{17.31}$$

In Vaango, we assume that the coupling terms $\partial\boldsymbol\sigma/\partial\boldsymbol\beta$ are zero and $T = T_p$ for elastic-plastic coupling. From (17.10) we have

$$\dot{\boldsymbol\sigma}^e = \left(\kappa - \tfrac{2}{3}\mu\right)\text{tr}(\dot{\boldsymbol\varepsilon}^e)\boldsymbol I + 2\mu\,\dot{\boldsymbol\varepsilon}^e \tag{17.32}$$

We can use this relation to estimate the coupling terms for the internal variables $\eta \in \{\varepsilon_p^{\text{eq}}, \phi, D, T\}$:

$$\frac{\partial\boldsymbol\sigma}{\partial\eta} = \left(\frac{\partial\kappa}{\partial\eta} - \tfrac{2}{3}\frac{\partial\mu}{\partial\eta}\right)\text{tr}(\boldsymbol\varepsilon^e)\boldsymbol I + 2\frac{\partial\mu}{\partial\eta}\,\boldsymbol\varepsilon^e \tag{17.33}$$

Similarly, from (17.13), choosing the basis to be $\hat{\boldsymbol I}$ and $\hat{\boldsymbol s}^{\text{trial}} = \text{dev}(\boldsymbol\sigma^{\text{trial}})/\left\|\text{dev}(\boldsymbol\sigma^{\text{trial}})\right\|$,

$$\boldsymbol\varepsilon^e = \frac{\sigma_p^e}{3\kappa}\hat{\boldsymbol I} + \frac{\sigma_s^e\sigma_{ss}}{2\mu}\hat{\boldsymbol s}^{\text{trial}} \quad\text{and}\quad \text{tr}(\boldsymbol\varepsilon^e) = \frac{\sigma_p^e}{\sqrt{3}\kappa} \tag{17.34}$$

where $\sigma_{ss} = \hat{\boldsymbol s}^e : \hat{\boldsymbol s}^{\text{trial}}$. Substitution into (17.33) leads to

$$\frac{\partial\boldsymbol\sigma}{\partial\eta} = \frac{1}{\kappa}\frac{\partial\kappa}{\partial\eta}\sigma_p^e\hat{\boldsymbol I} + \frac{1}{\mu}\frac{\partial\mu}{\partial\eta}\sigma_s^e\sigma_{ss}\hat{\boldsymbol s}^{\text{trial}}\,. \tag{17.35}$$

Also, for associated plasticity and using (17.19),

$$\mathbb{C}^e : \hat{\boldsymbol M} = \frac{1}{\|\boldsymbol N\|}\left[\sqrt{3}\kappa\frac{\partial f}{\partial p_\beta}\hat{\boldsymbol I} + 2\sqrt{\tfrac{3}{2}}\mu\frac{\partial f}{\partial\sigma_{\text{eff}}^\xi}\hat{\boldsymbol s}\right] = \frac{1}{\|\boldsymbol N\|}\left[\sqrt{3}\kappa\frac{\partial f}{\partial p_\beta}\hat{\boldsymbol I} + \sqrt{6}\mu\frac{\partial f}{\partial\sigma_{\text{eff}}^\xi}\sigma_{ss}\hat{\boldsymbol s}^{\text{trial}}\right] \tag{17.36}$$

Therefore,

$$\boldsymbol P = \left[\frac{\sqrt{3}\kappa}{\|\boldsymbol N\|}\frac{\partial f}{\partial p_\beta} - \frac{1}{\kappa}\sum_\eta\frac{\partial\kappa}{\partial\eta}\sigma_p^e\right]\hat{\boldsymbol I} + \left[\frac{\sqrt{6}\mu}{\|\boldsymbol N\|}\frac{\partial f}{\partial\sigma_{\text{eff}}^\xi} - \frac{1}{\mu}\sum_\eta\frac{\partial\mu}{\partial\eta}\sigma_s^e\right]\sigma_{ss}\hat{\boldsymbol s}^{\text{trial}} \tag{17.37}$$

For an elastic-plastic load step, we can compute the plastic strain rate using (17.13):

$$\dot{\boldsymbol\varepsilon}^p = \dot{\boldsymbol\varepsilon} - \frac{\dot\sigma_p^e}{3\kappa}\hat{\boldsymbol I} - \frac{\dot\sigma_s^e}{2\mu}\hat{\boldsymbol s}^{\text{trial}}\,. \tag{17.38}$$

17.1.6 Consistency condition

The consistency condition requires that, when $\dot{\lambda} > 0$,

$$\dot{f}(\boldsymbol{\sigma}, \boldsymbol{\beta}, \varepsilon_p^{\text{eq}}, \dot{\varepsilon}_p^{\text{eq}}, \phi, D, T, \dot{\varepsilon}^{\text{eq}}, \dots) = 0 \,. \tag{17.39}$$

For rate-independent plasticity, from the chain rule,

$$\dot{f} = \frac{\partial f}{\partial \boldsymbol{\sigma}} : \dot{\boldsymbol{\sigma}} + \frac{\partial f}{\partial \boldsymbol{\beta}} : \dot{\boldsymbol{\beta}} + \frac{\partial f}{\partial \varepsilon_p^{\text{eq}}} \dot{\varepsilon}_p^{\text{eq}} + \frac{\partial f}{\partial \phi} \dot{\phi} + \frac{\partial f}{\partial D} \dot{D} + \frac{\partial f}{\partial T_p} \dot{T}_p = 0 \,. \tag{17.40}$$

Using the hardening/softening rules,

$$\frac{\partial f}{\partial \boldsymbol{\sigma}} : \dot{\boldsymbol{\sigma}} + \dot{\lambda} \left[\frac{\partial f}{\partial \boldsymbol{\beta}} : \boldsymbol{h}^{\beta} + \frac{\partial f}{\partial \varepsilon_p^{\text{eq}}} h^{\varepsilon_p} + \frac{\partial f}{\partial \phi} h^{\phi} + \frac{\partial f}{\partial D} h^{D} + \frac{\partial f}{\partial T_p} h^{T} \right] = 0 \tag{17.41}$$

or

$$\frac{\partial f}{\partial \boldsymbol{\sigma}} : \dot{\boldsymbol{\sigma}} + \dot{\lambda} H = 0 \,. \tag{17.42}$$

Define,

$$\boldsymbol{N} := \frac{\partial f}{\partial \boldsymbol{\sigma}} \,, \quad \hat{\boldsymbol{N}} := \frac{\boldsymbol{N}}{\|\boldsymbol{N}\|} \,, \quad \hat{H} := \frac{H}{\|\boldsymbol{N}\|} \,. \tag{17.43}$$

Then,

$$\hat{\boldsymbol{N}} : \dot{\boldsymbol{\sigma}} + \dot{\lambda} \hat{H} = 0 \,. \tag{17.44}$$

Combining the stress-rate equation (17.31) with the consistency equation (17.44), we have

$$\hat{\boldsymbol{N}} : \dot{\boldsymbol{\sigma}}^{\text{trial}} = \hat{\boldsymbol{N}} : \mathbb{C}^e : \dot{\boldsymbol{\varepsilon}} = \dot{\lambda} (\hat{\boldsymbol{N}} : \boldsymbol{P} - \hat{H}) \,. \tag{17.45}$$

Therefore,

$$\dot{\lambda} = \frac{\hat{\boldsymbol{N}} : \dot{\boldsymbol{\sigma}}^{\text{trial}}}{\hat{\boldsymbol{N}} : \boldsymbol{P} - \hat{H}} = \frac{\hat{\boldsymbol{N}} : \mathbb{C}^e : \dot{\boldsymbol{\varepsilon}}}{\hat{\boldsymbol{N}} : \boldsymbol{P} - \hat{H}} \tag{17.46}$$

Substituting this expression to (17.31), we have

$$\dot{\boldsymbol{\sigma}} = \dot{\boldsymbol{\sigma}}^{\text{trial}} - \frac{\hat{\boldsymbol{N}} : \dot{\boldsymbol{\sigma}}^{\text{trial}}}{\hat{\boldsymbol{N}} : \boldsymbol{P} - \hat{H}} \boldsymbol{P} = \dot{\boldsymbol{\sigma}}^{\text{trial}} - \frac{\boldsymbol{P} \otimes \hat{\boldsymbol{N}}}{\hat{\boldsymbol{N}} : \boldsymbol{P} - \hat{H}} : \dot{\boldsymbol{\sigma}}^{\text{trial}} \tag{17.47}$$

or,

$$\dot{\boldsymbol{\sigma}} = \mathbb{C}^e : \dot{\boldsymbol{\varepsilon}} - \frac{(\boldsymbol{P} \otimes \hat{\boldsymbol{N}}) : \mathbb{C}^e}{\hat{\boldsymbol{N}} : \boldsymbol{P} - \hat{H}} : \dot{\boldsymbol{\varepsilon}} = \mathbb{C}^{ep} : \dot{\boldsymbol{\varepsilon}} \,. \tag{17.48}$$

The quantity $\mathbb{C}^{ep}$ is the continuum elastic-plastic tangent modulus.

17.2 Stress update

The first step in the stress update procedure is to compute a trial stress state from

$$\dot{\boldsymbol{\sigma}}^{\text{trial}} = \mathbb{C}^e : \dot{\boldsymbol{\varepsilon}} \,. \tag{17.49}$$

We assume that

$$\boldsymbol{\sigma}^{\text{trial}} = \boldsymbol{\sigma}_n + \Delta t (\mathbb{C}_n^e : \dot{\boldsymbol{\varepsilon}}_{n+1}) \tag{17.50}$$

where $\boldsymbol{\sigma}_n$ is the stress at the end of time t_n, $\mathbb{C}^e_n$ is the elastic modulus at that time, $\dot{\boldsymbol{\varepsilon}}_{n+1}$ is the strain rate computed from the symmetric part of the unrotated velocity gradient, and $\Delta t = t_{n+1} - t_n$ is the timestep size.

The trial state contains the vector

$$\boldsymbol{\eta}^{\text{trial}} = \left[\boldsymbol{\sigma}^{\text{trial}}, \boldsymbol{\beta}_n, (\varepsilon_p^{\text{eq}})_n, (\dot{\varepsilon}_p^{\text{eq}})_n, \phi_n, D_n, T_n, \dot{\varepsilon}_n^{\text{eq}}, \kappa_n, \mu_n, \dots\right]. \tag{17.51}$$

where the subscript (n) indicates the state at the end of time t_n.

The trial state is used to compute the yield function

$$f_y = f[\boldsymbol{\sigma}^{\text{trial}}, \boldsymbol{\beta}_n, (\varepsilon_p^{\text{eq}})_n, (\dot{\varepsilon}_p^{\text{eq}})_n, \phi_n, D_n, T_n, \dot{\varepsilon}_n^{\text{eq}}, \kappa_n, \mu_n, \dots] \tag{17.52}$$

If $f_y \le 0$, the trial state is in the elastic regime and we update the stress using

$$\begin{aligned}
&\boldsymbol{\sigma}_{n+1} = \boldsymbol{\sigma}^{\text{trial}}\,, \quad \boldsymbol{\beta}_{n+1} = \boldsymbol{\beta}_n\,, \quad (\varepsilon_p^{\text{eq}})_{n+1} = (\varepsilon_p^{\text{eq}})_n\,, \quad (\dot{\varepsilon}_p^{\text{eq}})_{n+1} = (\dot{\varepsilon}_p^{\text{eq}})_n \\
&\phi_{n+1} = \phi_n\,, \quad D_{n+1} = D_n\,, \quad T_{n+1} = T_n \\
&\kappa_{n+1} = \kappa(p_{n+1}, T_n)\,, \quad \mu_{n+1} = \mu(p_{n+1}, T_n)\,.
\end{aligned} \tag{17.53}$$

If $f_y > 0$, the trial state is outside the yield surface in the elastic-plastic regime. We can used a backward Euler algorithm to compute the updated stress state:

$$\frac{\boldsymbol{\sigma}_{n+1} - \boldsymbol{\sigma}_n}{\Delta t} = \frac{\boldsymbol{\sigma}^{\text{trial}} - \boldsymbol{\sigma}_n}{\Delta t} - \frac{\lambda_{n+1} - \lambda_n}{\Delta t}\boldsymbol{P}_{n+1} \quad \text{or} \quad \boldsymbol{\sigma}_{n+1} = \boldsymbol{\sigma}^{\text{trial}} - \Delta\lambda_{n+1}\boldsymbol{P}_{n+1}\,. \tag{17.54}$$

The plastic strain and the internal variables can similarly be updated using

$$\begin{aligned}
\boldsymbol{\varepsilon}^p_{n+1} &= \boldsymbol{\varepsilon}^p_n &&+\Delta\lambda_{n+1}\,\hat{\boldsymbol{M}}_{n+1} \\
(\varepsilon_p^{\text{eq}})_{n+1} &= (\varepsilon_p^{\text{eq}})_n &&+\Delta\lambda_{n+1}\,h^{\varepsilon_p}_{n+1} \\
\boldsymbol{\beta}_{n+1} &= \boldsymbol{\beta}_n &&+\Delta\lambda_{n+1}\,\boldsymbol{h}^{\beta}_{n+1} \\
\phi_{n+1} &= \phi_n &&+\Delta\lambda_{n+1}\,h^{\phi}_{n+1} \\
D_{n+1} &= D_n &&+\Delta\lambda_{n+1}\,h^{D}_{n+1} \\
(T_p)_{n+1} &= (T_p)_n &&+\Delta\lambda_{n+1}\,h^{T}_{n+1}\,.
\end{aligned} \tag{17.55}$$

In addition, the stress state has to lie on the yield surface:

$$f(\boldsymbol{\sigma}^{\text{trial}} - \Delta\lambda_{n+1}\boldsymbol{P}_{n+1}) = 0\,. \tag{17.56}$$

Finally, the consistency condition needs to be satisfied:

$$\hat{\boldsymbol{N}}_{n+1} : (\boldsymbol{\sigma}_{n+1} - \boldsymbol{\sigma}_n) + \Delta\lambda_{n+1}\hat{H}_{n+1} = 0 \tag{17.57}$$

or,

$$\hat{\boldsymbol{N}}_{n+1} : (\boldsymbol{\sigma}^{\text{trial}} - \boldsymbol{\sigma}_n) = \Delta\lambda_{n+1}(\hat{\boldsymbol{N}}_{n+1} : \boldsymbol{P}_{n+1} - \hat{H}_{n+1})\,. \tag{17.58}$$

17.2.1 Iterative solution

Vaango assumes associated plasticity for isotropic metals, i.e., $\hat{\boldsymbol{M}} = \hat{\boldsymbol{N}}$. Therefore, the following coupled equations, not all of which are independent, have to be solved for $\Gamma := \Delta\lambda_{n+1}$ and the updated state

$\left[\boldsymbol{\sigma}_{n+1}, \kappa_{n+1}, \mu_{n+1}, \boldsymbol{\varepsilon}^p_{n+1}, (\varepsilon^{\text{eq}}_p)_{n+1}, \boldsymbol{\beta}_{n+1}, \phi_{n+1}, D_{n+1}, (T_p)_{n+1}\right]$:

$$
\begin{aligned}
\boldsymbol{\sigma}_{n+1} &= \boldsymbol{\sigma}^{\text{trial}} - \Gamma \boldsymbol{P}_{n+1} \\
\boldsymbol{\varepsilon}^p_{n+1} &= \boldsymbol{\varepsilon}^p_n + \Gamma \hat{\boldsymbol{N}}_{n+1} \\
\boldsymbol{\beta}_{n+1} &= \boldsymbol{\beta}_n + \Gamma \boldsymbol{h}^{\beta}_{n+1} \\
(\varepsilon^{\text{eq}}_p)_{n+1} &= (\varepsilon^{\text{eq}}_p)_n + \Gamma h^{\varepsilon_p}_{n+1} \\
\phi_{n+1} &= \phi_n + \Gamma h^{\phi}_{n+1} \\
D_{n+1} &= D_n + \Gamma h^{D}_{n+1} \\
(T_p)_{n+1} &= (T_p)_n + \Gamma h^{T}_{n+1} \\
\hat{\boldsymbol{N}}_{n+1} : (\boldsymbol{\sigma}^{\text{trial}} - \boldsymbol{\sigma}_n) - \Gamma(\hat{\boldsymbol{N}}_{n+1} : \boldsymbol{P}_{n+1} - \hat{H}_{n+1}) &= 0 \\
f_{n+1} = f(\boldsymbol{\sigma}^{\text{trial}} - \Gamma \boldsymbol{P}_{n+1}) &= 0
\end{aligned}
\tag{17.59}
$$

where

$$
\begin{aligned}
\boldsymbol{P}_{n+1} &= \mathbb{C}^e_{n+1} : \hat{\boldsymbol{N}}_{n+1} - \left.\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\beta}}\right|_{n+1} : \boldsymbol{h}^{\beta}_{n+1} - \left.\frac{\partial \boldsymbol{\sigma}}{\partial \varepsilon^{\text{eq}}_p}\right|_{n+1} h^{\varepsilon_p}_{n+1} - \left.\frac{\partial \boldsymbol{\sigma}}{\partial \phi}\right|_{n+1} h^{\phi}_{n+1} - \left.\frac{\partial \boldsymbol{\sigma}}{\partial D}\right|_{n+1} h^{D}_{n+1} - \left.\frac{\partial \boldsymbol{\sigma}}{\partial T_p}\right|_{n+1} h^{T}_{n+1} \\
\hat{\boldsymbol{N}}_{n+1} &= \frac{\boldsymbol{N}_{n+1}}{\|\boldsymbol{N}_{n+1}\|}\,, \quad \boldsymbol{N}_{n+1} = \left.\frac{\partial f}{\partial \boldsymbol{\sigma}}\right|_{n+1} \\
\hat{H}_{n+1} &= \frac{H_{n+1}}{\|\boldsymbol{N}_{n+1}\|}\,, \quad H_{n+1} = \left.\frac{\partial f}{\partial \boldsymbol{\beta}}\right|_{n+1} : \boldsymbol{h}^{\beta}_{n+1} + \left.\frac{\partial f}{\partial \varepsilon^{\text{eq}}_p}\right|_{n+1} h^{\varepsilon_p}_{n+1} + \left.\frac{\partial f}{\partial \phi}\right|_{n+1} h^{\phi}_{n+1} + \left.\frac{\partial f}{\partial D}\right|_{n+1} h^{D}_{n+1} + \left.\frac{\partial f}{\partial T_p}\right|_{n+1} h^{T}_{n+1}\,.
\end{aligned}
\tag{17.60}
$$

Let us express the stresses and strains as vectors:

$$
\mathbf{S} := \left[\sigma_{11}, \sigma_{22}, \sigma_{33}, \sqrt{2}\sigma_{23}, \sqrt{2}\sigma_{31}, \sqrt{2}\sigma_{12}\right] \qquad \mathbf{E}^p := \left[\varepsilon^p_{11}, \varepsilon^p_{22}, \varepsilon^p_{33}, \sqrt{2}\varepsilon^p_{23}, \sqrt{2}\varepsilon^p_{31}, \sqrt{2}\varepsilon^p_{12}\right]\,. \tag{17.61}
$$

We can also write

$$
\mathbf{N} := \left[\frac{\partial f}{\partial \sigma_{11}}, \frac{\partial f}{\partial \sigma_{22}}, \frac{\partial f}{\partial \sigma_{33}}, \sqrt{2}\frac{\partial f}{\partial \sigma_{23}}, \sqrt{2}\frac{\partial f}{\partial \sigma_{31}}, \sqrt{2}\frac{\partial f}{\partial \sigma_{12}}\right]\,, \quad \hat{\mathbf{N}} = \frac{\mathbf{N}}{\|\mathbf{N}\|}\,. \tag{17.62}
$$

Because of the isotropy of the elasticity tensor, we have

$$
\begin{aligned}
\mathbb{C}^e : \hat{\mathbf{N}} = \mathbf{C} = \Big[&2\mu N_1 + (\kappa - \tfrac{2}{3}\mu)(N_1 + N_2 + N_3), 2\mu N_2 + (\kappa - \tfrac{2}{3}\mu)(N_1 + N_2 + N_3), \\
&2\mu N_3 + (\kappa - \tfrac{2}{3}\mu)(N_1 + N_2 + N_3), 2\sqrt{2}\mu N_4, 2\sqrt{2}\mu N_5, 2\sqrt{2}\mu N_6\Big]
\end{aligned}
\tag{17.63}
$$

where N_i are the components of the vector $\hat{\mathbf{N}}$. If we ignore the elastic-plastic coupling term $\partial\boldsymbol{\sigma}/\partial\boldsymbol{\beta}$, and define

$$
\begin{aligned}
\mathbf{Z}^{\varepsilon_p} &:= \left[\frac{\partial \sigma_{11}}{\partial \varepsilon^{\text{eq}}_p} h^{\varepsilon_p}, \frac{\partial \sigma_{22}}{\partial \varepsilon^{\text{eq}}_p} h^{\varepsilon_p}, \frac{\partial \sigma_{33}}{\partial \varepsilon^{\text{eq}}_p} h^{\varepsilon_p}, \sqrt{2}\frac{\partial \sigma_{23}}{\partial \varepsilon^{\text{eq}}_p} h^{\varepsilon_p}, \sqrt{2}\frac{\partial \sigma_{31}}{\partial \varepsilon^{\text{eq}}_p} h^{\varepsilon_p}, \sqrt{2}\frac{\partial \sigma_{12}}{\partial \varepsilon^{\text{eq}}_p} h^{\varepsilon_p}\right] \\
\mathbf{Z}^{\phi} &:= \left[\frac{\partial \sigma_{11}}{\partial \phi} h^{\phi}, \frac{\partial \sigma_{22}}{\partial \phi} h^{\phi}, \frac{\partial \sigma_{33}}{\partial \phi} h^{\phi}, \sqrt{2}\frac{\partial \sigma_{23}}{\partial \phi} h^{\phi}, \sqrt{2}\frac{\partial \sigma_{31}}{\partial \phi} h^{\phi}, \sqrt{2}\frac{\partial \sigma_{12}}{\partial \phi} h^{\phi}\right] \\
\mathbf{Z}^{D} &:= \left[\frac{\partial \sigma_{11}}{\partial D} h^{D}, \frac{\partial \sigma_{22}}{\partial D} h^{D}, \frac{\partial \sigma_{33}}{\partial D} h^{D}, \sqrt{2}\frac{\partial \sigma_{23}}{\partial D} h^{D}, \sqrt{2}\frac{\partial \sigma_{31}}{\partial D} h^{D}, \sqrt{2}\frac{\partial \sigma_{12}}{\partial D} h^{D}\right] \\
\mathbf{Z}^{T_p} &:= \left[\frac{\partial \sigma_{11}}{\partial T_p} h^{T}, \frac{\partial \sigma_{22}}{\partial T_p} h^{T}, \frac{\partial \sigma_{33}}{\partial T_p} h^{T}, \sqrt{2}\frac{\partial \sigma_{23}}{\partial T_p} h^{T}, \sqrt{2}\frac{\partial \sigma_{31}}{\partial T_p} h^{T}, \sqrt{2}\frac{\partial \sigma_{12}}{\partial T_p} h^{T}\right]
\end{aligned}
\tag{17.64}
$$

we have

$$\mathbf{P} = \mathbf{C} - \mathbf{Z}^{\varepsilon_p} - \mathbf{Z}^{\phi} - \mathbf{Z}^{D} - \mathbf{Z}^{T_p} \,. \tag{17.65}$$

Noting that $\boldsymbol{\beta}$ is required to be symmetric for the conservation of angular momentum, we can express the internal variables as a vector such that

$$\begin{aligned}
\mathbf{Q} &:= \left[\beta_{11}, \beta_{22}, \beta_{33}, \sqrt{2}\beta_{23}, \sqrt{2}\beta_{31}, \sqrt{2}\beta_{12}, \varepsilon_p^{\text{eq}}, \phi, D, T_p\right] \\
\mathbf{H} &:= \left[h_{11}^{\beta}, h_{22}^{\beta}, h_{33}^{\beta}, \sqrt{2}h_{23}^{\beta}, \sqrt{2}h_{31}^{\beta}, \sqrt{2}h_{12}^{\beta}, h^{\varepsilon_p}, h^{\phi}, h^{D}, h^{T}\right] \\
\frac{\partial f}{\partial \mathbf{Q}} &:= \left[\frac{\partial f}{\partial \beta_{11}}, \frac{\partial f}{\partial \beta_{22}}, \frac{\partial f}{\partial \beta_{33}}, \sqrt{2}\frac{\partial f}{\partial \beta_{23}}, \sqrt{2}\frac{\partial f}{\partial \beta_{31}}, \sqrt{2}\frac{\partial f}{\partial \beta_{12}}, \frac{\partial f}{\partial \varepsilon_p^{\text{eq}}}, \frac{\partial f}{\partial \phi}, \frac{\partial f}{\partial D}, \frac{\partial f}{\partial T_p}\right]
\end{aligned} \tag{17.66}$$

Therefore,

$$H = \frac{\partial f}{\partial \mathbf{Q}} \cdot \mathbf{H} \,, \quad \hat{H} = \frac{H}{\|\mathbf{N}\|} \,. \tag{17.67}$$

Then (17.59), can be written in vector form as

$$\begin{aligned}
&\mathbf{S}_{n+1} = \mathbf{S}^{\text{trial}} - \Gamma \mathbf{P}_{n+1} \\
&\mathbf{E}^p_{n+1} = \mathbf{E}^p_n + \Gamma \hat{\mathbf{N}}_{n+1} \\
&\mathbf{Q}_{n+1} = \mathbf{Q}_n + \Gamma \mathbf{H}_{n+1} \\
&\hat{\mathbf{N}}_{n+1} \cdot (\mathbf{S}^{\text{trial}} - \mathbf{S}_n) - \Gamma(\hat{\mathbf{N}}_{n+1} \cdot \mathbf{P}_{n+1} - \hat{H}_{n+1}) = 0 \quad \Longrightarrow \quad \Gamma = \frac{\hat{\mathbf{N}}_{n+1} \cdot (\mathbf{S}^{\text{trial}} - \mathbf{S}_n)}{\left(\hat{\mathbf{N}}_{n+1} \cdot \mathbf{P}_{n+1} - \hat{H}_{n+1}\right)} \\
&f_{n+1} = f(\mathbf{S}^{\text{trial}} - \Gamma \mathbf{P}_{n+1}) = 0
\end{aligned} \tag{17.68}$$

Since $\mathbf{P} = \mathbf{P}(\mathbf{S}, \mathbf{E}^p, \mathbf{Q})$, $\mathbf{N} = \mathbf{N}(\mathbf{S}, \mathbf{E}^p, \mathbf{Q})$, $\mathbf{H} = \mathbf{H}(\mathbf{S}, \mathbf{E}^p, \mathbf{Q})$, and $H = H(\mathbf{S}, \mathbf{E}^p, \mathbf{Q})$, we can write the equations above as residuals:

$$\begin{aligned}
\mathbf{r}_S(\Gamma, \mathbf{S}, \mathbf{E}^p, \mathbf{Q}) &:= -\mathbf{S}_{n+1} + \mathbf{S}^{\text{trial}} - \Gamma \mathbf{P}_{n+1}(\mathbf{S}, \mathbf{E}^p, \mathbf{Q}) = \mathbf{0} \\
\mathbf{r}_E(\Gamma, \mathbf{S}, \mathbf{E}^p, \mathbf{Q}) &:= -\mathbf{E}^p_{n+1} + \mathbf{E}^p_n + \Gamma \hat{\mathbf{N}}_{n+1}(\mathbf{S}, \mathbf{E}^p, \mathbf{Q}) = \mathbf{0} \\
\mathbf{r}_q(\Gamma, \mathbf{S}, \mathbf{E}^p, \mathbf{Q}) &:= -\mathbf{Q}_{n+1} + \mathbf{Q}_n + \Gamma \mathbf{H}_{n+1}(\mathbf{S}, \mathbf{E}^p, \mathbf{Q}) = \mathbf{0} \\
r_f(\Gamma, \mathbf{S}, \mathbf{E}^p, \mathbf{Q}) &:= f(\mathbf{S}_{n+1}, \mathbf{E}^p_{n+1}, \mathbf{Q}_{n+1}) = 0
\end{aligned} \tag{17.69}$$

First-order Taylor series expansions of these functions at $(\Gamma, \mathbf{S}_n, \mathbf{E}^p_n, \mathbf{Q}_n)$, give

$$\begin{aligned}
\mathbf{r}_S(\Gamma, \mathbf{S}, \mathbf{E}^p, \mathbf{Q}) &\approx \mathbf{r}_S(\Gamma, \mathbf{S}_n, \mathbf{E}^p_n, \mathbf{Q}_n) + \\
&\left.\frac{\partial \mathbf{r}_S}{\partial \Gamma}\right|_n (\Gamma - \Gamma_n) + \left.\frac{\partial \mathbf{r}_S}{\partial \mathbf{S}}\right|_n \cdot (\mathbf{S} - \mathbf{S}_n) + \left.\frac{\partial \mathbf{r}_S}{\partial \mathbf{E}^p}\right|_n \cdot (\mathbf{E}^p - \mathbf{E}^p_n) + \left.\frac{\partial \mathbf{r}_S}{\partial \mathbf{Q}}\right|_n \cdot (\mathbf{Q} - \mathbf{Q}_n) \\
\mathbf{r}_E(\Gamma, \mathbf{S}, \mathbf{E}^p, \mathbf{Q}) &\approx \mathbf{r}_E(\Gamma, \mathbf{S}_n, \mathbf{E}^p_n, \mathbf{Q}_n) + \\
&\left.\frac{\partial \mathbf{r}_Q}{\partial \Gamma}\right|_n (\Gamma - \Gamma_n) + \left.\frac{\partial \mathbf{r}_Q}{\partial \mathbf{S}}\right|_n \cdot (\mathbf{S} - \mathbf{S}_n) + \left.\frac{\partial \mathbf{r}_Q}{\partial \mathbf{E}^p}\right|_n \cdot (\mathbf{E}^p - \mathbf{E}^p_n) + \left.\frac{\partial \mathbf{r}_Q}{\partial \mathbf{Q}}\right|_n \cdot (\mathbf{Q} - \mathbf{Q}_n) \\
\mathbf{r}_Q(\Gamma, \mathbf{S}, \mathbf{E}^p, \mathbf{Q}) &\approx \mathbf{r}_Q(\Gamma, \mathbf{S}_n, \mathbf{E}^p_n, \mathbf{Q}_n) + \\
&\left.\frac{\partial \mathbf{r}_E}{\partial \Gamma}\right|_n (\Gamma - \Gamma_n) + \left.\frac{\partial \mathbf{r}_E}{\partial \mathbf{S}}\right|_n \cdot (\mathbf{S} - \mathbf{S}_n) + \left.\frac{\partial \mathbf{r}_E}{\partial \mathbf{E}^p}\right|_n \cdot (\mathbf{E}^p - \mathbf{E}^p_n) + \left.\frac{\partial \mathbf{r}_E}{\partial \mathbf{Q}}\right|_n \cdot (\mathbf{Q} - \mathbf{Q}_n) \\
r_f(\Gamma, \mathbf{S}, \mathbf{E}^p, \mathbf{Q}) &\approx r_f(\Gamma, \mathbf{S}_n, \mathbf{E}^p_n, \mathbf{Q}_n) + \\
&\left.\frac{\partial r_f}{\partial \Gamma}\right|_n (\Gamma - \Gamma_n) + \left.\frac{\partial r_f}{\partial \mathbf{S}}\right|_n \cdot (\mathbf{S} - \mathbf{S}_n) + \left.\frac{\partial r_f}{\partial \mathbf{E}^p}\right|_n \cdot (\mathbf{E}^p - \mathbf{E}^p_n) + \left.\frac{\partial r_f}{\partial \mathbf{Q}}\right|_n \cdot (\mathbf{Q} - \mathbf{Q}_n)
\end{aligned} \tag{17.70}$$

Since the residuals are required to be zero at the end of the timestep, we get the following rule for the k-th iteration,

$$\begin{aligned}
&\left.\frac{\partial \mathbf{r}_S}{\partial \Gamma}\right|_k \Delta\Gamma + \left.\frac{\partial \mathbf{r}_S}{\partial \mathbf{S}}\right|_k \cdot \Delta\mathbf{S} + \left.\frac{\partial \mathbf{r}_S}{\partial \mathbf{E}^p}\right|_k \cdot \Delta\mathbf{E}^p + \left.\frac{\partial \mathbf{r}_S}{\partial \mathbf{Q}}\right|_k \cdot \Delta\mathbf{Q} = -\mathbf{r}_S(\Gamma_k, \mathbf{S}_k, \mathbf{E}^p_k, \mathbf{Q}_k)\\
&\left.\frac{\partial \mathbf{r}_E}{\partial \Gamma}\right|_k \Delta\Gamma + \left.\frac{\partial \mathbf{r}_E}{\partial \mathbf{S}}\right|_k \cdot \Delta\mathbf{S} + \left.\frac{\partial \mathbf{r}_E}{\partial \mathbf{E}^p}\right|_k \cdot \Delta\mathbf{E}^p + \left.\frac{\partial \mathbf{r}_E}{\partial \mathbf{Q}}\right|_k \cdot \Delta\mathbf{Q} = -\mathbf{r}_E(\Gamma_k, \mathbf{S}_k, \mathbf{E}^p_k, \mathbf{Q}_k)\\
&\left.\frac{\partial \mathbf{r}_Q}{\partial \Gamma}\right|_k \Delta\Gamma + \left.\frac{\partial \mathbf{r}_Q}{\partial \mathbf{S}}\right|_k \cdot \Delta\mathbf{S} + \left.\frac{\partial \mathbf{r}_Q}{\partial \mathbf{E}^p}\right|_k \cdot \Delta\mathbf{E}^p + \left.\frac{\partial \mathbf{r}_Q}{\partial \mathbf{Q}}\right|_k \cdot \Delta\mathbf{Q} = -\mathbf{r}_Q(\Gamma_k, \mathbf{S}_k, \mathbf{E}^p_k, \mathbf{Q}_k)\\
&\left.\frac{\partial r_f}{\partial \Gamma}\right|_k \Delta\Gamma + \left.\frac{\partial r_f}{\partial \mathbf{S}}\right|_k \cdot \Delta\mathbf{S} + \left.\frac{\partial r_f}{\partial \mathbf{E}^p}\right|_k \cdot \Delta\mathbf{E}^p + \left.\frac{\partial r_f}{\partial \mathbf{Q}}\right|_k \cdot \Delta\mathbf{Q} = -r_f(\Gamma_k, \mathbf{S}_k, \mathbf{E}^p_k, \mathbf{Q}_k)
\end{aligned} \tag{17.71}$$

where

$$\Delta\Gamma = \Gamma_{k+1} - \Gamma_k\,,\quad \Delta\mathbf{S} = \mathbf{S}_{k+1} - \mathbf{S}_k\,,\quad \Delta\mathbf{E}^p = \mathbf{E}^p_{k+1} - \mathbf{E}^p_k\,,\quad \Delta\mathbf{Q} = \mathbf{Q}_{k+1} - \mathbf{Q}_k\,. \tag{17.72}$$

The derivatives of the residuals are (dropping subscripts $n+1$ for convenience),

$$\begin{aligned}
&\frac{\partial \mathbf{r}_S}{\partial \Gamma} = -\mathbf{P}\,,\quad \frac{\partial \mathbf{r}_E}{\partial \Gamma} = \hat{\mathbf{N}}\,,\quad \frac{\partial \mathbf{r}_Q}{\partial \Gamma} = \mathbf{H}\,,\quad \frac{\partial r_f}{\partial \Gamma} = 0\\
&\frac{\partial \mathbf{r}_S}{\partial \mathbf{S}} = -\left[\mathbb{I}\right] - \Gamma\frac{\partial \mathbf{P}}{\partial \mathbf{S}}\,,\quad \frac{\partial \mathbf{r}_E}{\partial \mathbf{S}} = \Gamma\frac{\partial \hat{\mathbf{N}}}{\partial \mathbf{S}}\,,\quad \frac{\partial \mathbf{r}_Q}{\partial \mathbf{S}} = \Gamma\frac{\partial \mathbf{H}}{\partial \mathbf{S}}\,,\quad \frac{\partial r_f}{\partial \mathbf{S}} = \frac{\partial f}{\partial \mathbf{S}}\\
&\frac{\partial \mathbf{r}_S}{\partial \mathbf{E}^p} = -\Gamma\frac{\partial \mathbf{P}}{\partial \mathbf{E}^p}\,,\quad \frac{\partial \mathbf{r}_E}{\partial \mathbf{E}^p} = -\left[\mathbb{I}\right] + \Gamma\frac{\partial \hat{\mathbf{N}}}{\partial \mathbf{E}^p}\,,\quad \frac{\partial \mathbf{r}_Q}{\partial \mathbf{E}^p} = \Gamma\frac{\partial \mathbf{H}}{\partial \mathbf{E}^p}\,,\quad \frac{\partial r_f}{\partial \mathbf{E}^p} = \frac{\partial f}{\partial \mathbf{E}^p}\\
&\frac{\partial \mathbf{r}_S}{\partial \mathbf{Q}} = -\Gamma\frac{\partial \mathbf{P}}{\partial \mathbf{Q}}\,,\quad \frac{\partial \mathbf{r}_E}{\partial \mathbf{Q}} = \Gamma\frac{\partial \hat{\mathbf{N}}}{\partial \mathbf{Q}}\,,\quad \frac{\partial \mathbf{r}_Q}{\partial \mathbf{Q}} = -\left[\mathbb{I}\right] + \Gamma\frac{\partial \mathbf{H}}{\partial \mathbf{Q}}\,,\quad \frac{\partial r_f}{\partial \mathbf{Q}} = \frac{\partial f}{\partial \mathbf{Q}}
\end{aligned} \tag{17.73}$$

Therefore, using $\mathbf{N} = \partial f/\partial \mathbf{S}$,

$$\begin{aligned}
&-\mathbf{P}_k\Delta\Gamma - \left(\left[\mathbb{I}\right] + \Gamma_k \left.\frac{\partial \mathbf{P}}{\partial \mathbf{S}}\right|_k\right)\cdot\Delta\mathbf{S} - \Gamma_k \left.\frac{\partial \mathbf{P}}{\partial \mathbf{E}^p}\right|_k \cdot \Delta\mathbf{E}^p - \Gamma_k \left.\frac{\partial \mathbf{P}}{\partial \mathbf{Q}}\right|_k \cdot \Delta\mathbf{Q} = -\mathbf{r}_S(\Gamma_k, \mathbf{S}_k, \mathbf{E}^p_k, \mathbf{Q}_k)\\
&\hat{\mathbf{N}}_k\Delta\Gamma + \Gamma_k \left.\frac{\partial \hat{\mathbf{N}}}{\partial \mathbf{S}}\right|_k \cdot \Delta\mathbf{S} - \left(\left[\mathbb{I}\right] - \Gamma_k \left.\frac{\partial \hat{\mathbf{N}}}{\partial \mathbf{E}^p}\right|_k\right)\cdot\Delta\mathbf{E}^p + \Gamma_k \left.\frac{\partial \hat{\mathbf{N}}}{\partial \mathbf{Q}}\right|_k \cdot \Delta\mathbf{Q} = -\mathbf{r}_E(\Gamma_k, \mathbf{S}_k, \mathbf{E}^p_k, \mathbf{Q}_k)\\
&\mathbf{H}_k\Delta\Gamma + \Gamma_k \left.\frac{\partial \mathbf{H}}{\partial \mathbf{S}}\right|_k \cdot \Delta\mathbf{S} + \Gamma_k \left.\frac{\partial \mathbf{H}}{\partial \mathbf{E}^p}\right|_k \cdot \Delta\mathbf{E}^p - \left(\left[\mathbb{I}\right] - \Gamma_k \left.\frac{\partial \mathbf{H}}{\partial \mathbf{Q}}\right|_k\right)\cdot\Delta\mathbf{Q} = -\mathbf{r}_Q(\Gamma_k, \mathbf{S}_k, \mathbf{E}^p_k, \mathbf{Q}_k)\\
&\mathbf{N}_k \cdot \Delta\mathbf{S} + \left.\frac{\partial f}{\partial \mathbf{E}^p}\right|_k \cdot \Delta\mathbf{E}^p + \left.\frac{\partial f}{\partial \mathbf{Q}}\right|_k \cdot \Delta\mathbf{Q} = -r_f(\Gamma_k, \mathbf{S}_k, \mathbf{E}^p_k, \mathbf{Q}_k)
\end{aligned} \tag{17.74}$$

Because the derivatives of $\hat{\mathbf{N}}$, $\mathbf{P}$, $\mathbf{H}$ with respect to $\mathbf{S}$, $\mathbf{E}^p$, $\mathbf{Q}$ may be difficult to calculate, it is more convenient to use a semi-implicit scheme where the quantities $\hat{\mathbf{N}}$, $\mathbf{P}$, $\mathbf{H}$ are evaluated at t_n. In that case we have

$$\begin{aligned}
&-\mathbf{P}_k\Delta\Gamma - \Delta\mathbf{S} = -\mathbf{r}_S(\Gamma_k, \mathbf{S}_k, \mathbf{E}^p_k, \mathbf{Q}_k)\\
&\hat{\mathbf{N}}_k\Delta\Gamma - \Delta\mathbf{E}^p = -\mathbf{r}_E(\Gamma_k, \mathbf{S}_k, \mathbf{E}^p_k, \mathbf{Q}_k)\\
&\mathbf{H}_k\Delta\Gamma - \Delta\mathbf{Q} = -\mathbf{r}_Q(\Gamma_k, \mathbf{S}_k, \mathbf{E}^p_k, \mathbf{Q}_k)\\
&\mathbf{N}_k \cdot \Delta\mathbf{S} + \left.\frac{\partial f}{\partial \mathbf{E}^p}\right|_k \cdot \Delta\mathbf{E}^p + \left.\frac{\partial f}{\partial \mathbf{Q}}\right|_k \cdot \Delta\mathbf{Q} = -r_f(\Gamma_k, \mathbf{S}_k, \mathbf{E}^p_k, \mathbf{Q}_k)
\end{aligned} \tag{17.75}$$

We now force $\mathbf{r}_S$, $\mathbf{r}_E$, and $\mathbf{r}_Q$ to be zero at all times, leading to the expressions

$$\begin{aligned}
&\Delta\mathbf{S} = -\mathbf{P}_k\Delta\Gamma\\
&\Delta\mathbf{E}^p = \hat{\mathbf{N}}_k\Delta\Gamma\\
&\Delta\mathbf{Q} = \mathbf{H}_k\Delta\Gamma\\
&r_f(\Gamma_k, \mathbf{S}_k, \mathbf{E}^p_k, \mathbf{Q}_k) + \mathbf{N}_k \cdot \Delta\mathbf{S} + \left.\frac{\partial f}{\partial \mathbf{E}^p}\right|_k \cdot \Delta\mathbf{E}^p + \left.\frac{\partial f}{\partial \mathbf{Q}}\right|_k \cdot \Delta\mathbf{Q} = 0
\end{aligned} \tag{17.76}$$

Plugging the expressions for $\Delta\mathbf{S}$, $\Delta\mathbf{E}^p$, $\Delta\mathbf{Q}$ from the first three equations into the fourth gives us

$$r_f(\Gamma_k, \mathbf{S}_k, \mathbf{E}^p_k, \mathbf{Q}_k) - \mathbf{N}_k \cdot \mathbf{P}_k \Delta\Gamma + \left.\frac{\partial f}{\partial \mathbf{E}^p}\right|_k \cdot \hat{\mathbf{N}}_k \Delta\Gamma + \left.\frac{\partial f}{\partial \mathbf{Q}}\right|_k \cdot \mathbf{H}_k \Delta\Gamma = 0 \tag{17.77}$$

or

$$\Delta\Gamma = \frac{f(\mathbf{S}_k, \mathbf{E}^p_k, \mathbf{Q}_k)}{\mathbf{N}_k \cdot \mathbf{P}_k - \left.\frac{\partial f}{\partial \mathbf{E}^p}\right|_k \cdot \hat{\mathbf{N}}_k - \left.\frac{\partial f}{\partial \mathbf{Q}}\right|_k \cdot \mathbf{H}_k} \tag{17.78}$$

In the metal plasticity models implemented in Vaango , there is no direct dependence of f on $\mathbf{E}^p$. Therefore, using (17.67),

$$\Gamma_{k+1} = \Gamma_k + \frac{f(\mathbf{S}_k, \mathbf{Q}_k)}{\mathbf{N}_k \cdot \mathbf{P}_k - H_k} \,. \tag{17.79}$$

All quantities on the right hand side of the above equation are known, and we can compute Γ_{k+1}. The other variables can now be updated using

$$\mathbf{S}_{k+1} = \mathbf{S}_k - \mathbf{P}_k \Delta\Gamma \,, \quad \mathbf{E}^p_{k+1} = \mathbf{E}^p_k + \hat{\mathbf{N}}_k \Delta\Gamma \,, \quad \mathbf{Q}_{k+1} = \mathbf{Q}_k + \mathbf{H}_k \Delta\Gamma \tag{17.80}$$

The iterative process can be stopped when r_f is close to 0 and Γ_{k+1} is close to the value required to satisfy consistency given in (17.68).

17.2.2 Stress update in reduced stress space

The isotropic metal yield functions in Vaango depend only on two invariants of stress. Therefore, the return mapping can be carried out in the reduced stress space spanned by $\hat{\boldsymbol{I}}$ and $\hat{\boldsymbol{s}}^{\text{trial}}$ where

$$\hat{\boldsymbol{I}} = \frac{\boldsymbol{I}}{\|\boldsymbol{I}\|} = \frac{\boldsymbol{I}}{\sqrt{3}} \quad \text{and} \quad \hat{\boldsymbol{s}}^{\text{trial}} = \frac{\boldsymbol{s}^{\text{trial}}}{\|\boldsymbol{s}^{\text{trial}}\|} = \frac{\text{dev}(\boldsymbol{\sigma}^{\text{trial}})}{\|\text{dev}(\boldsymbol{\sigma}^{\text{trial}})\|} \,. \tag{17.81}$$

The stress can be expressed in terms of this basis as,

$$\boldsymbol{\sigma}_{n+1} = (\sigma_p)_{n+1}\hat{\boldsymbol{I}} + (\sigma_s)_{n+1}\hat{\boldsymbol{s}}^{\text{trial}} \quad \text{where} \quad (\sigma_p)_{n+1} = \boldsymbol{\sigma}_{n+1} : \hat{\boldsymbol{I}} = \frac{\text{tr}(\boldsymbol{\sigma}_{n+1})}{\sqrt{3}} \,, \quad (\sigma_s)_{n+1} = \boldsymbol{\sigma}_{n+1} : \hat{\boldsymbol{s}}^{\text{trial}} \,. \tag{17.82}$$

As before,

$$\boldsymbol{\sigma}^{\text{trial}} = \boldsymbol{\sigma}_n + \Delta t(\mathbb{C}^e_n : \dot{\boldsymbol{\varepsilon}}_{n+1}) = \boldsymbol{\sigma}_n + \Delta t\left[\left(\kappa_n - \tfrac{2}{3}\mu_n\right)\text{tr}(\dot{\boldsymbol{\varepsilon}}_{n+1})\boldsymbol{I} + 2\mu_n\dot{\boldsymbol{\varepsilon}}_{n+1}\right] \tag{17.83}$$

The trial stress is then decomposed into

$$\boldsymbol{\sigma}^{\text{trial}} = \sigma_p^{\text{trial}}\hat{\boldsymbol{I}} + \sigma_s^{\text{trial}}\hat{\boldsymbol{s}}^{\text{trial}} \quad \text{where} \quad \sigma_p^{\text{trial}} = \frac{\text{tr}(\boldsymbol{\sigma}^{\text{trial}})}{\sqrt{3}} \,, \quad \sigma_s^{\text{trial}} = \boldsymbol{\sigma}^{\text{trial}} : \hat{\boldsymbol{s}}^{\text{trial}} = \left\|\boldsymbol{s}^{\text{trial}}\right\| \,. \tag{17.84}$$

The yield function is computed using

$$f_y = f[\boldsymbol{\sigma}_\beta^{\text{trial}}, (\varepsilon_p^{\text{eq}})_n, (\dot{\varepsilon}_p^{\text{eq}})_n, \phi_n, D_n, T_n, \dot{\varepsilon}_n^{\text{eq}}, \kappa_n, \mu_n, \dots] \,, \quad \boldsymbol{\sigma}_\beta^{\text{trial}} = \boldsymbol{\sigma}^{\text{trial}} - \boldsymbol{\beta}_n \,. \tag{17.85}$$

If $f_y \le 0$, the state is updated using

$$\begin{aligned} &\boldsymbol{\sigma}_{n+1} = \boldsymbol{\sigma}^{\text{trial}} \,, \quad \boldsymbol{\beta}_{n+1} = \boldsymbol{\beta}_n \,, \quad (\varepsilon_p^{\text{eq}})_{n+1} = (\varepsilon_p^{\text{eq}})_n \,, \quad (\dot{\varepsilon}_p^{\text{eq}})_{n+1} = (\dot{\varepsilon}_p^{\text{eq}})_n \\ &\phi_{n+1} = \phi_n \,, \quad D_{n+1} = D_n \,, \quad T_{n+1} = T_n \\ &\kappa_{n+1} = \kappa(p_{n+1}, T_n) \,, \quad \mu_{n+1} = \mu(p_{n+1}, T_n) \,. \end{aligned} \tag{17.86}$$

If $f_y > 0$, integration of the stress rate by backward Euler leads to

$$\boldsymbol{\sigma}_{n+1} = \boldsymbol{\sigma}^{\text{trial}} - \Delta\lambda_{n+1}\boldsymbol{P}_{n+1}\,. \tag{17.87}$$

Expressed in terms of the trial basis using (17.37), and noting that $\boldsymbol{\sigma}_{n+1} = \boldsymbol{\sigma}^e_{n+1}$ and $(\sigma_{ss})_{n+1} = \hat{\boldsymbol{s}}_{n+1} : \hat{\boldsymbol{s}}^{\text{trial}}$,

$$\begin{aligned}\boldsymbol{P}_{n+1} = &\left[\frac{\sqrt{3}\kappa_{n+1}}{\|\boldsymbol{N}_{n+1}\|}\frac{\partial f_{n+1}}{\partial p_\beta} - \frac{1}{\kappa_{n+1}}\sum_\eta \frac{\partial \kappa_{n+1}}{\partial \eta}(\sigma_p)_{n+1}\right]\hat{\boldsymbol{I}} \\ &+ \left[\frac{\sqrt{6}\mu_{n+1}}{\|\boldsymbol{N}_{n+1}\|}\frac{\partial f_{n+1}}{\partial \sigma^\xi_{\text{eff}}} - \frac{1}{\mu_{n+1}}\sum_\eta \frac{\partial \mu_{n+1}}{\partial \eta}(\sigma_s)_{n+1}\right](\sigma_{ss})_{n+1}\hat{\boldsymbol{s}}^{\text{trial}}\end{aligned} \tag{17.88}$$

Therefore,

$$\begin{aligned}(\sigma_p)_{n+1} &= \sigma_p^{\text{trial}} - \Delta\lambda_{n+1}\left[\frac{\sqrt{3}\kappa_{n+1}}{\|\boldsymbol{N}_{n+1}\|}\frac{\partial f_{n+1}}{\partial p_\beta} - \frac{1}{\kappa_{n+1}}\sum_\eta \frac{\partial \kappa_{n+1}}{\partial \eta}(\sigma_p)_{n+1}\right] \\ (\sigma_s)_{n+1} &= \sigma_s^{\text{trial}} - \Delta\lambda_{n+1}\left[\frac{\sqrt{6}\mu_{n+1}}{\|\boldsymbol{N}_{n+1}\|}\frac{\partial f_{n+1}}{\partial \sigma^\xi_{\text{eff}}} - \frac{1}{\mu_{n+1}}\sum_\eta \frac{\partial \mu_{n+1}}{\partial \eta}(\sigma_s)_{n+1}\right](\sigma_{ss})_{n+1}\end{aligned} \tag{17.89}$$

The plastic strain can be updated using (17.38):

$$\boldsymbol{\varepsilon}^p_{n+1} = \boldsymbol{\varepsilon}^p_n + \Delta t\dot{\boldsymbol{\varepsilon}}_{n+1} - \left(\frac{(\sigma_p)_{n+1} - (\sigma_p)_n}{3\kappa_{n+1}}\right)\hat{\boldsymbol{I}} - \left(\frac{(\sigma_s)_{n+1} - (\sigma_s)_n}{2\mu_{n+1}}\right)\hat{\boldsymbol{s}}^{\text{trial}}\,. \tag{17.90}$$

The internal variables can be updated using

$$\boldsymbol{\eta}_{n+1} = \boldsymbol{\eta}_n + \Delta\lambda_{n+1}\,\boldsymbol{h}^\eta_{n+1} \tag{17.91}$$

Also, as before, the stress state has to lie on the yield surface:

$$f(\boldsymbol{\sigma}^{\text{trial}} - \Delta\lambda_{n+1}\boldsymbol{P}_{n+1}) = 0 \tag{17.92}$$

and the consistency condition needs to be satisfied:

$$\hat{\boldsymbol{N}}_{n+1} : (\boldsymbol{\sigma}^{\text{trial}} - \boldsymbol{\sigma}_n) = \Delta\lambda_{n+1}(\hat{\boldsymbol{N}}_{n+1} : \boldsymbol{P}_{n+1} - \hat{H}_{n+1}) \tag{17.93}$$

where

$$\hat{\boldsymbol{N}}_{n+1} = \frac{\boldsymbol{N}_{n+1}}{\|\boldsymbol{N}_{n+1}\|}\,,\quad \boldsymbol{N}_{n+1} = \tfrac{1}{\sqrt{3}}\frac{\partial f_{n+1}}{\partial p_\beta}\hat{\boldsymbol{I}} + \sqrt{\tfrac{3}{2}}\frac{\partial f_{n+1}}{\partial \sigma^\xi_{\text{eff}}}\hat{\boldsymbol{s}}^{\text{trial}}\,. \tag{17.94}$$

We can now attempt to express the iterative semi-implicit stress update algorithm given in (17.79) and (17.80) in terms of the trial basis. Recall that

$$\Gamma_{k+1} = \Gamma_k + \frac{f(\mathbf{S}_k, \mathbf{Q}_k)}{\mathbf{N}_k \cdot \mathbf{P}_k - H_k}\,,\quad \mathbf{E}^p_{k+1} = \mathbf{E}^p_k + \hat{\mathbf{N}}_k\Delta\Gamma\,,\quad \mathbf{S}_{k+1} = \mathbf{S}_k - \mathbf{P}_k\Delta\Gamma\,,\quad \mathbf{Q}_{k+1} = \mathbf{Q}_k + \mathbf{H}_k\Delta\Gamma \tag{17.95}$$

Reverting back to tensor notation,

$$\Gamma_{k+1} = \Gamma_k + \frac{f(\boldsymbol{\sigma}_k, \boldsymbol{\eta}_k)}{\|\boldsymbol{N}_k\|\,(\hat{\boldsymbol{N}}_k : \boldsymbol{P}_k - \hat{H}_k)}\,,\quad \boldsymbol{\varepsilon}^p_{k+1} = \boldsymbol{\varepsilon}^p_k + \hat{\boldsymbol{N}}_k\Delta\Gamma\,,\quad \boldsymbol{\sigma}_{k+1} = \boldsymbol{\sigma}_k - \boldsymbol{P}_k\Delta\Gamma\,,\quad \boldsymbol{\eta}_{k+1} = \boldsymbol{\eta}_k + \boldsymbol{H}_k\Delta\Gamma \tag{17.96}$$

Using (17.94) and (17.88), we have

$$\begin{aligned}\|\boldsymbol{N}_k\|\,(\hat{\boldsymbol{N}}_k : \boldsymbol{P}_k) = &\tfrac{1}{\sqrt{3}}\frac{\partial f_k}{\partial p_\beta}\left[\sqrt{3}\kappa_k\frac{\partial f_k}{\partial p_\beta} - \frac{\|\boldsymbol{N}_k\|}{\kappa_k}\sum_\eta \frac{\partial \kappa_k}{\partial \eta}(\sigma_p)_k\right] + \\ &\sqrt{\tfrac{3}{2}}\frac{\partial f_k}{\partial \sigma^\xi_{\text{eff}}}\left[\sqrt{6}\mu_k\frac{\partial f_k}{\partial \sigma^\xi_{\text{eff}}} - \frac{\|\boldsymbol{N}_k\|}{\mu_k}\sum_\eta \frac{\partial \mu_k}{\partial \eta}(\sigma_s)_k\right](\sigma_{ss})_k\end{aligned} \tag{17.97}$$

Also,

$$\|\boldsymbol{N}_k\|\,\hat{H}_k = \frac{\partial f_k}{\partial \boldsymbol{\beta}} : (\boldsymbol{h}^{\beta})_k + \frac{\partial f_k}{\partial \varepsilon_p^{\text{eq}}}(h^{\varepsilon_p})_k + \frac{\partial f_k}{\partial \phi}(h^{\phi})_k + \frac{\partial f_k}{\partial D}(h^{D})_k + \frac{\partial f_k}{\partial T_p}(h^{T})_k\,. \tag{17.98}$$

At the end of the iterative process, the plastic strain tensor may also be updated using (17.90) and a non-hardening return used to force the computed stress state on the final yield surface.

17.2.3 Algorithm 1

Two implementations of the model described in this chapter have been implemented in VAANGO. The algorithm described in this section is used when the elastic_plastic_hp algorithm is invoked in an input file. This plastic return algorithm is more robust but does not include kinematic hardening or softening due to damage and temperature changes. Softening and damage are treated in an uncoupled manner after the stress has been updated. Also, elastic-plastic coupling is ignored. For this algorithm, the timestep is divided into substeps and the following algorithm is applied at each substep. At the end of the last substep, the stress state is projected back to the updated yield surface without any changes to the internal variables.

1. Inputs:

$$\begin{aligned}
\text{Timestep size:}\quad & \Delta t \\
\text{New strain rate:}\quad & \dot{\boldsymbol{\varepsilon}}_{n+1},\, \dot{\varepsilon}^{\text{eq}}_{n+1} \\
\text{Old stress:}\quad & \boldsymbol{\sigma}_n \\
\text{Old moduli:}\quad & \kappa_n,\, \mu_n \\
\text{Old plastic strain:}\quad & \boldsymbol{\varepsilon}^p_n \\
\text{Old equivalent plastic rate:}\quad & (\dot{\varepsilon}^{\text{eq}}_p)_n \\
\text{Old internal variables:}\quad & (\varepsilon^{\text{eq}}_p)_n,\, \phi_n,\, T_n \\
\text{Trial stress:}\quad & \boldsymbol{\sigma}^{\text{trial}} = \boldsymbol{\sigma}_n + \Delta t\left[\left(\kappa_n - \tfrac{2}{3}\mu_n\right)\text{tr}(\dot{\boldsymbol{\varepsilon}}_{n+1})\boldsymbol{I} + 2\mu_n\dot{\boldsymbol{\varepsilon}}_{n+1}\right]
\end{aligned} \tag{17.99}$$

2. Decompose trial stress

$$\begin{aligned}
&p^{\text{trial}} = \tfrac{1}{3}\text{tr}(\boldsymbol{\sigma}^{\text{trial}})\,,\quad \boldsymbol{s}^{\text{trial}} = \boldsymbol{\sigma}^{\text{trial}} - \tfrac{1}{3}\text{tr}(\boldsymbol{\sigma}^{\text{trial}})\boldsymbol{I}\,,\quad \hat{\boldsymbol{s}}^{\text{trial}} = \frac{\boldsymbol{s}^{\text{trial}}}{\|\boldsymbol{s}^{\text{trial}}\|} \\
&\boldsymbol{\sigma}^{\text{trial}} = \sigma_p^{\text{trial}}\hat{\boldsymbol{I}} + \sigma_s^{\text{trial}}\hat{\boldsymbol{s}}^{\text{trial}}\,,\quad \sigma_p^{\text{trial}} = \sqrt{3}p^{\text{trial}}\,,\quad \sigma_s^{\text{trial}} = \|\boldsymbol{s}^{\text{trial}}\|
\end{aligned} \tag{17.100}$$

3. Decompose start-of-timestep stress

$$\begin{aligned}
&\boldsymbol{\sigma}_n = p_n\boldsymbol{I} + \boldsymbol{s}_n = (\sigma_p)_n\hat{\boldsymbol{I}} + (\sigma_s)_n\hat{\boldsymbol{s}}^{\text{trial}}\,,\quad (\sigma_{\text{eff}})_n = \sqrt{\tfrac{3}{2}\boldsymbol{s}_n : \boldsymbol{s}_n} \\
&(\sigma_p)_n = \boldsymbol{\sigma}_n : \hat{\boldsymbol{I}}\,,\quad (\sigma_s)_n = \boldsymbol{\sigma}_n : \hat{\boldsymbol{s}}^{\text{trial}} = \boldsymbol{s}_n : \hat{\boldsymbol{s}}^{\text{trial}}
\end{aligned} \tag{17.101}$$

4. Compute f_n and derivatives

$$f_n = f\left(\boldsymbol{s}_n, p_n, (\varepsilon^{\text{eq}}_p)_n, (\dot{\varepsilon}^{\text{eq}}_p)_n, \phi_n, T_n, \kappa_n, \mu_n, \dot{\varepsilon}^{\text{eq}}_{n+1}, \dots\right)\,,\quad \frac{\partial f_n}{\partial p}\,,\quad \frac{\partial f_n}{\partial \sigma_{\text{eff}}} \tag{17.102}$$

5. Compute components of $\boldsymbol{N}_n$ and $\|\boldsymbol{N}_n\|$

$$\begin{aligned}
&\boldsymbol{N}_n = (N_p)_n\hat{\boldsymbol{I}} + (N_s)_n\hat{\boldsymbol{s}}^{\text{trial}}\,,\quad \|\boldsymbol{N}_n\| = \sqrt{(N_p)_n^2 + (N_s)_n^2} \\
&(N_p)_n = \tfrac{1}{\sqrt{3}}\frac{\partial f_n}{\partial p}\,,\quad (N_s)_n = \sqrt{\tfrac{3}{2}}\frac{\partial f_n}{\partial \sigma_{\text{eff}}}
\end{aligned} \tag{17.103}$$

6. Compute components of $\boldsymbol{P}_n$

$$\begin{aligned} \boldsymbol{P}_n &= (P_p)_n \hat{\boldsymbol{I}} + (P_s)_n \hat{\boldsymbol{s}}^{\text{trial}} \\ (P_p)_n &= \frac{\sqrt{3}\kappa_n}{\|\boldsymbol{N}_n\|} \frac{\partial f_n}{\partial p} \;, \quad (P_s)_n = \frac{\sqrt{6}\mu_n}{\|\boldsymbol{N}_n\|} \frac{\partial f_n}{\partial \sigma_{\text{eff}}} \end{aligned} \tag{17.104}$$

7. Initialize:

$$\begin{aligned} k &= 0 \;, \quad \Gamma = 0 \\ \boldsymbol{\sigma}_k &= \boldsymbol{\sigma}^{\text{trial}} \end{aligned} \tag{17.105}$$

8. Decompose current stress

$$\begin{aligned} \boldsymbol{\sigma}_k &= p_k \boldsymbol{I} + \boldsymbol{s}_k = (\sigma_p)_k \hat{\boldsymbol{I}} + (\sigma_s)_k \hat{\boldsymbol{s}}^{\text{trial}} \;, \quad (\sigma_{\text{eff}})_k = \sqrt{\tfrac{3}{2} \boldsymbol{s}_k : \boldsymbol{s}_k} \\ (\sigma_p)_k &= \boldsymbol{\sigma}_k : \hat{\boldsymbol{I}} \;, \quad (\sigma_s)_k = \boldsymbol{\sigma}_k : \hat{\boldsymbol{s}}^{\text{trial}} = \boldsymbol{s}_k : \hat{\boldsymbol{s}}^{\text{trial}} \;, \quad (\sigma_{ss})_k = \frac{\boldsymbol{s}_k}{\|\boldsymbol{s}_k\|} : \hat{\boldsymbol{s}}^{\text{trial}} = 1 \end{aligned} \tag{17.106}$$

9. Compute f_k and derivatives

$$f_k = f\left(\boldsymbol{s}_k, p_k, (\varepsilon_p^{\text{eq}})_k, (\dot{\varepsilon}_p^{\text{eq}})_k, \phi_k, T_n, \kappa_n, \mu_n, \dot{\varepsilon}_{n+1}^{\text{eq}}, \dots\right) \;, \quad \frac{\partial f_k}{\partial p} \;, \quad \frac{\partial f_k}{\partial \sigma_{\text{eff}}} \tag{17.107}$$

10. Compute components of $\boldsymbol{N}_k$ and $\|\boldsymbol{N}_k\|$

$$\begin{aligned} \boldsymbol{N}_k &= (N_p)_k \hat{\boldsymbol{I}} + (N_s)_k \hat{\boldsymbol{s}}^{\text{trial}} \;, \quad \|\boldsymbol{N}_k\| = \sqrt{(N_p)_k^2 + (N_s)_k^2} \\ (N_p)_k &= \tfrac{1}{\sqrt{3}} \frac{\partial f_k}{\partial p} \;, \quad (N_s)_k = \sqrt{\tfrac{3}{2}} \frac{\partial f_k}{\partial \sigma_{\text{eff}}} \;. \end{aligned} \tag{17.108}$$

11. Compute $\boldsymbol{N}_k : \boldsymbol{P}_n$

$$\boldsymbol{N}_k : \boldsymbol{P}_n = (N_p)_k (P_p)_n + (N_s)_k (P_s)_n \tag{17.109}$$

12. Compute updated $\Delta\Gamma$

$$\Gamma_{k+1} = \Gamma_k + \frac{f_k}{\boldsymbol{N}_k : \boldsymbol{P}_n} \;, \quad \Delta\Gamma = \Gamma_{k+1} - \Gamma_k \tag{17.110}$$

13. Compute updated stress components:

$$(\sigma_p)_{k+1} = (\sigma_p)_k - (P_p)_n \Delta\Gamma \;, \quad (\sigma_s)_{k+1} = (\sigma_s)_k - (P_s)_n \Delta\Gamma \tag{17.111}$$

14. Compute f_{k+1}

$$f_{k+1} = f\left(\boldsymbol{s}_{k+1}, p_{k+1}, (\varepsilon_p^{\text{eq}})_k, (\dot{\varepsilon}_p^{\text{eq}})_{k+1}, \phi_{k+1}, D_{k+1}, T_{k+1}, \kappa_{k+1}, \mu_{k+1}, \dot{\varepsilon}_{n+1}^{\text{eq}}, \dots\right) \tag{17.112}$$

15. If $|f_{k+1}| < f_{\text{tolerance}}$ and $|\Gamma_{k+1} - \Gamma_k| < \Gamma_{\text{tolerance}}$ go to step 18.
16. Set $k \leftarrow k + 1$ and go to step 8.
17. Update the stress:

$$\boldsymbol{\sigma}_{n+1} = (\sigma_p)_{k+1} \hat{\boldsymbol{I}} + (\sigma_s)_{k+1} \hat{\boldsymbol{s}}^{\text{trial}} \tag{17.113}$$

18. Compute internal variable hardening/softening moduli

$$(h^{\varepsilon_p})_{k+1} \;, \quad (h^{\phi})_{k+1} \tag{17.114}$$

19. Compute updated internal variables

$$(\varepsilon_p^{\text{eq}})_{n+1} = (\varepsilon_p^{\text{eq}})_n + (h^{\varepsilon_p})_{k+1}\Delta\Gamma\ , \quad \phi_{n+1} = \phi_n + (h^{\phi})_{k+1}\Delta\Gamma\ , \tag{17.115}$$

20. Compute updated elastic strain

$$\boldsymbol{\varepsilon}^e_{n+1} = \boldsymbol{\varepsilon}^e_n + \left(\frac{(\sigma_p)_{k+1} - (\sigma_p)_n}{3\kappa_n}\right)\hat{\boldsymbol{I}} - \left(\frac{(\sigma_s)_{k+1} - (\sigma_s)_n}{2\mu_n}\right)\hat{\boldsymbol{s}}^{\text{trial}}\ . \tag{17.116}$$

21. Compute the updated plastic strain

$$\boldsymbol{\varepsilon}^p_{n+1} = \boldsymbol{\varepsilon}^e_n + \boldsymbol{\varepsilon}^p_n + \Delta t\dot{\boldsymbol{\varepsilon}}_{n+1} - \boldsymbol{\varepsilon}^e_{n+1}\ , \quad (\dot{\varepsilon}_p^{\text{eq}})_{n+1} = \left\|\boldsymbol{\varepsilon}^p_{n+1}\right\| \tag{17.117}$$

22. Compute updated elastic moduli

$$\kappa_{n+1} = \kappa\left((\sigma_p)_{k+1}, \boldsymbol{\varepsilon}^e_{n+1}, (T_p)_n\right)\ , \quad \mu_{n+1} = \mu\left((\sigma_p)_{k+1}, \boldsymbol{\varepsilon}^e_{n+1}, (T_p)_n\right) \tag{17.118}$$

17.2.4 Algorithm 2

The following stress update algorithm is used for each (plastic) time step for models that require kinematic hardening.

1. Inputs:

$$\begin{aligned}
\text{Timestep size:} &\quad \Delta t \\
\text{New strain rate:} &\quad \dot{\boldsymbol{\varepsilon}}_{n+1}, \dot{\varepsilon}^{\text{eq}}_{n+1} \\
\text{Old stress:} &\quad \boldsymbol{\sigma}_n \\
\text{Old moduli:} &\quad \kappa_n, \mu_n \\
\text{Old plastic strain:} &\quad \boldsymbol{\varepsilon}^p_n \\
\text{Old equivalent plastic rate:} &\quad (\dot{\varepsilon}_p^{\text{eq}})_n \\
\text{Old internal variables:} &\quad \boldsymbol{\beta}_n, (\varepsilon_p^{\text{eq}})_n, \phi_n, D_n, (T_p)_n \\
\text{Trial stress:} &\quad \boldsymbol{\sigma}^{\text{trial}} = \boldsymbol{\sigma}_n + \Delta t\left[\left(\kappa_n - \tfrac{2}{3}\mu_n\right)\text{tr}(\dot{\boldsymbol{\varepsilon}}_{n+1})\boldsymbol{I} + 2\mu_n\dot{\boldsymbol{\varepsilon}}_{n+1}\right]
\end{aligned} \tag{17.119}$$

2. Decompose trial stress:

$$\begin{aligned}
&p^{\text{trial}} = \tfrac{1}{3}\text{tr}(\boldsymbol{\sigma}^{\text{trial}})\ , \quad \boldsymbol{s}^{\text{trial}} = \boldsymbol{\sigma}^{\text{trial}} - \tfrac{1}{3}\text{tr}(\boldsymbol{\sigma}^{\text{trial}})\boldsymbol{I}\ , \quad \hat{\boldsymbol{s}}^{\text{trial}} = \frac{\boldsymbol{s}^{\text{trial}}}{\left\|\boldsymbol{s}^{\text{trial}}\right\|} \\
&\boldsymbol{\sigma}^{\text{trial}} = \sigma_p^{\text{trial}}\hat{\boldsymbol{I}} + \sigma_s^{\text{trial}}\hat{\boldsymbol{s}}^{\text{trial}}\ , \quad \sigma_p^{\text{trial}} = \sqrt{3}p^{\text{trial}}\ , \quad \sigma_s^{\text{trial}} = \left\|\boldsymbol{s}^{\text{trial}}\right\|
\end{aligned} \tag{17.120}$$

3. Initialize:

$$\begin{aligned}
&k = 0\ , \quad \Gamma = 0 \\
&\boldsymbol{\sigma}_k = \boldsymbol{\sigma}^{\text{trial}}, \quad \kappa_k = \kappa_n, \quad \mu_k = \mu_n, \quad \boldsymbol{\varepsilon}^p_k = \boldsymbol{\varepsilon}^p_n \\
&(\varepsilon_p^{\text{eq}})_k = (\varepsilon_p^{\text{eq}})_n, \quad \boldsymbol{\beta}_k = \boldsymbol{\beta}_n\ . \quad \phi_k = \phi_n, \quad D_k = D_n, \quad (T_p)_k = (T_p)_n \\
&(\dot{\varepsilon}_p^{\text{eq}})_k = (\dot{\varepsilon}_p^{\text{eq}})_n, \quad \dot{\varepsilon}^{\text{eq}} = \dot{\varepsilon}^{\text{eq}}_{n+1}
\end{aligned} \tag{17.121}$$

4. Compute shifted stress using backstress:

$$(\boldsymbol{\sigma}_\beta)_k = \boldsymbol{\sigma}_k - \boldsymbol{\beta}_k\ , \quad (p_\beta)_k = \tfrac{1}{3}\text{tr}(\boldsymbol{\sigma}_\beta)_k\ , \quad \boldsymbol{\xi}_k = (\boldsymbol{\sigma}_\beta)_k - (p_\beta)_k\boldsymbol{I}\ , \quad (\sigma^{\xi}_{\text{eff}})_k = \sqrt{\tfrac{3}{2}\boldsymbol{\xi}_k : \boldsymbol{\xi}_k} \tag{17.122}$$

5. Compute f_k and derivatives

$$f_k = f\left(\boldsymbol{\xi}_k, (p_\beta)_k, (\varepsilon_p^{\text{eq}})_k, (\dot{\varepsilon}_p^{\text{eq}})_k, \phi_k, D_k, T_k, \kappa_k, \mu_k, \dot{\varepsilon}^{\text{eq}}_{n+1}, \dots\right)\ , \quad \frac{\partial f_k}{\partial p_\beta}\ , \quad \frac{\partial f_k}{\partial \sigma^{\xi}_{\text{eff}}} \tag{17.123}$$

6. Compute components of $\boldsymbol{N}_k$ and $\|\boldsymbol{N}_k\|$

$$\begin{aligned} \boldsymbol{N}_k &= (N_p)_k \hat{\boldsymbol{I}} + (N_s)_k \hat{\boldsymbol{s}}^{\text{trial}} \,, \quad \|\boldsymbol{N}_k\| = \sqrt{(N_p)_k^2 + (N_s)_k^2} \\ (N_p)_k &= \tfrac{1}{\sqrt{3}} \frac{\partial f_k}{\partial p_\beta} \,, \quad (N_s)_k = \sqrt{\tfrac{3}{2}} \frac{\partial f_k}{\partial \sigma_{\text{eff}}^{\xi}} \,. \end{aligned} \tag{17.124}$$

7. Compute derivatives of bulk and shear modulus with respect to internal variables

$$\frac{\partial \kappa_k}{\partial \varepsilon_p^{\text{eq}}}, \frac{\partial \kappa_k}{\partial \phi}, \frac{\partial \kappa_k}{\partial D_k}, \frac{\partial \kappa_k}{\partial T_k} \,, \quad \frac{\partial \mu_k}{\partial \varepsilon_p^{\text{eq}}}, \frac{\partial \mu_k}{\partial \phi}, \frac{\partial \mu_k}{\partial D_k}, \frac{\partial \mu_k}{\partial T_k} \tag{17.125}$$

8. Decompose current stress

$$\begin{aligned} \boldsymbol{\sigma}_k &= p_k \boldsymbol{I} + \boldsymbol{s}_k = (\sigma_p)_k \hat{\boldsymbol{I}} + (\sigma_s)_k \hat{\boldsymbol{s}}^{\text{trial}} \\ (\sigma_p)_k &= \boldsymbol{\sigma}_k : \hat{\boldsymbol{I}} \,, \quad (\sigma_s)_k = \boldsymbol{\sigma}_k : \hat{\boldsymbol{s}}^{\text{trial}} = \boldsymbol{s}_k : \hat{\boldsymbol{s}}^{\text{trial}} \,, \quad (\sigma_{ss})_k = \frac{\boldsymbol{s}_k}{\|\boldsymbol{s}_k\|} : \hat{\boldsymbol{s}}^{\text{trial}} = 1 \end{aligned} \tag{17.126}$$

9. Compute components of $\boldsymbol{P}_k$

$$\begin{aligned} \boldsymbol{P}_k &= (P_p)_k \hat{\boldsymbol{I}} + (P_s)_k \hat{\boldsymbol{s}}^{\text{trial}} \\ (P_p)_k &= \left[\frac{\sqrt{3}\kappa_k}{\|\boldsymbol{N}_k\|} \frac{\partial f_k}{\partial p_\beta} - \frac{1}{\kappa_k} \sum_\eta \frac{\partial \kappa_k}{\partial \eta} (\sigma_p)_k \right] \\ (P_s)_k &= \left[\frac{\sqrt{6}\mu_k}{\|\boldsymbol{N}_k\|} \frac{\partial f_k}{\partial \sigma_{\text{eff}}^{\xi}} - \frac{1}{\mu_k} \sum_\eta \frac{\partial \mu_k}{\partial \eta} (\sigma_s)_k \right] (\sigma_{ss})_k \end{aligned} \tag{17.127}$$

10. Compute $\|\boldsymbol{N}_k\| \, (\hat{\boldsymbol{N}}_k : \boldsymbol{P}_k)$

$$\|\boldsymbol{N}_k\| \, (\hat{\boldsymbol{N}}_k : \boldsymbol{P}_k) = (N_p)_k (P_p)_k + (N_s)_k (P_s)_k \tag{17.128}$$

11. Compute derivatives of f_k with respect to internal variables

$$\frac{\partial f_k}{\partial \boldsymbol{\beta}} \,, \quad \frac{\partial f_k}{\partial \varepsilon_p^{\text{eq}}} \,, \quad \frac{\partial f_k}{\partial \phi} \,, \quad \frac{\partial f_k}{\partial D} \,, \quad \frac{\partial f_k}{\partial T_p} \tag{17.129}$$

12. Compute internal variable hardening/softening moduli

$$(\boldsymbol{h}^\beta)_k \,, \quad (h^{\varepsilon_p})_k \,, \quad (h^\phi)_k \,, \quad (h^D)_k \,, \quad (h^T)_k \tag{17.130}$$

13. Compute $\|\boldsymbol{N}_k\| \, \hat{H}_k$

$$\|\boldsymbol{N}_k\| \, \hat{H}_k = \frac{\partial f_k}{\partial \boldsymbol{\beta}} : (\boldsymbol{h}^\beta)_k + \frac{\partial f_k}{\partial \varepsilon_p^{\text{eq}}} (h^{\varepsilon_p})_k + \frac{\partial f_k}{\partial \phi} (h^\phi)_k + \frac{\partial f_k}{\partial D} (h^D)_k + \frac{\partial f_k}{\partial T_p} (h^T)_k \,. \tag{17.131}$$

14. Compute updated $\Delta\Gamma$

$$\Gamma_{k+1} = \Gamma_k + \frac{f_k}{\|\boldsymbol{N}_k\| \, (\hat{\boldsymbol{N}}_k : \boldsymbol{P}_k - \hat{H}_k)} \,, \quad \Delta\Gamma = \Gamma_{k+1} - \Gamma_k \tag{17.132}$$

15. Compute updated stress components:

$$(\sigma_p)_{k+1} = (\sigma_p)_k - (P_p)_k \Delta\Gamma \,, \quad (\sigma_s)_{k+1} = (\sigma_s)_k - (P_s)_k \Delta\Gamma \tag{17.133}$$

16. Compute updated internal variables:

$$\begin{aligned} \boldsymbol{\beta}_{k+1} &= \boldsymbol{\beta}_k + \boldsymbol{h}^\beta \Delta\Gamma \,, \quad (\varepsilon_p^{\text{eq}})_{k+1} = (\varepsilon_p^{\text{eq}})_k + h^{\varepsilon_p} \Delta\Gamma \\ \phi_{k+1} &= \phi_k + h^\phi \Delta\Gamma \,, \quad D_{k+1} = D_k + h^D \Delta\Gamma \,, \quad (T_p)_{k+1} = (T_p)_k + h^T \Delta\Gamma \end{aligned} \tag{17.134}$$

17. Compute updated elastic strain:

$$\boldsymbol{\varepsilon}^e_{k+1} = \boldsymbol{\varepsilon}^e_n + \left(\frac{(\sigma_p)_{k+1} - (\sigma_p)_n}{3\kappa_k}\right)\hat{\boldsymbol{I}} - \left(\frac{(\sigma_s)_{k+1} - (\sigma_s)_n}{2\mu_k}\right)\hat{\boldsymbol{s}}^{\text{trial}} . \tag{17.135}$$

18. Compute the updated plastic strain:

$$\boldsymbol{\varepsilon}^p_{k+1} = \boldsymbol{\varepsilon}^e_n + \boldsymbol{\varepsilon}^p_n + \Delta t\dot{\boldsymbol{\varepsilon}}_{n+1} - \boldsymbol{\varepsilon}^e_{k+1} , \quad (\dot{\varepsilon}^{\text{eq}}_p)_{k+1} = \left\|\boldsymbol{\varepsilon}^p_{k+1}\right\| \tag{17.136}$$

19. Compute updated elastic moduli

$$\kappa_{k+1} = \kappa\left((\sigma_p)_{k+1}, \boldsymbol{\varepsilon}^e_{k+1}, (T_p)_{k+1}\right) , \quad \mu_{k+1} = \mu\left((\sigma_p)_{k+1}, \boldsymbol{\varepsilon}^e_{k+1}, (T_p)_{k+1}\right) \tag{17.137}$$

20. Compute f_{k+1}

$$f_{k+1} = f\left(\boldsymbol{\xi}_{k+1}, (p_\beta)_{k+1}, (\varepsilon^{\text{eq}}_p)_{k+1}, (\dot{\varepsilon}^{\text{eq}}_p)_{k+1}, \phi_{k+1}, D_{k+1}, T_{k+1}, \kappa_{k+1}, \mu_{k+1}, \dot{\varepsilon}^{\text{eq}}_{n+1}, \dots\right) \tag{17.138}$$

21. If $|f_{k+1}| < f_{\text{tolerance}}$ and $|\Gamma_{k+1} - \Gamma_k| < \Gamma_{\text{tolerance}}$ go to step 23.
22. Set $k \leftarrow k+1$ and go to step 4.
23. Update the state:

$$\begin{aligned}
&\boldsymbol{\sigma}_{n+1} = (\sigma_p)_{k+1}\hat{\boldsymbol{I}} + (\sigma_s)_{k+1}\hat{\boldsymbol{s}}^{\text{trial}} , \quad \boldsymbol{\beta}_{n+1} = \boldsymbol{\beta}_{k+1} , \quad (\varepsilon^{\text{eq}}_p)_{n+1} = (\varepsilon^{\text{eq}}_p)_{k+1} \\
&\phi_{n+1} = \phi_{k+1} , \quad D_{n+1} = D_{k+1} , \quad (T_p)_{n+1} = (T_p)_{k+1} , \quad (\dot{\varepsilon}^{\text{eq}}_p)_{n+1} = (\dot{\varepsilon}^{\text{eq}}_p)_{k+1} \\
&\kappa_{n+1} = \kappa_k , \quad \mu_{n+1} = \mu_k \\
&\boldsymbol{\varepsilon}^e_{n+1} = \boldsymbol{\varepsilon}^p_n + \left(\frac{(\sigma_p)_{k+1} - (\sigma_p)_n}{3\kappa_{n+1}}\right)\hat{\boldsymbol{I}} - \left(\frac{(\sigma_s)_{k+1} - (\sigma_s)_n}{2\mu_{n+1}}\right)\hat{\boldsymbol{s}}^{\text{trial}} \\
&\boldsymbol{\varepsilon}^p_{n+1} = \boldsymbol{\varepsilon}^e_n + \boldsymbol{\varepsilon}^p_n + \Delta t\dot{\boldsymbol{\varepsilon}}_{n+1} - \boldsymbol{\varepsilon}^e_{n+1}
\end{aligned} \tag{17.139}$$

17.3 Example 1: von Mises plasticity

Consider the case of J_2 plasticity with the yield condition

$$f := \sqrt{\tfrac{3}{2}}\,\|\boldsymbol{s} - \text{dev}(\boldsymbol{\beta})\| - \sigma_y(\varepsilon^{\text{eq}}_p, \dot{\varepsilon}^{\text{eq}}, \phi, T, \dots) = \sqrt{\tfrac{3}{2}}\,\|\boldsymbol{\xi}\| - \sigma_y(\varepsilon^{\text{eq}}_p, \dot{\varepsilon}^{\text{eq}}, \phi, T, \dots) \le 0 \tag{17.140}$$

where $\|\boldsymbol{\xi}\| = \sqrt{\boldsymbol{\xi} : \boldsymbol{\xi}}$. The derivatives of the yield function with respect to the internal variables are

$$\frac{\partial f}{\partial \boldsymbol{\beta}} = -\sqrt{\tfrac{3}{2}}\frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|} , \quad \frac{\partial f}{\partial \varepsilon^{\text{eq}}_p} = -\frac{\partial \sigma_y}{\partial \varepsilon^{\text{eq}}_p} , \quad \frac{\partial f}{\partial \phi} = -\frac{\partial \sigma_y}{\partial \phi} , \quad \frac{\partial f}{\partial D} = -\frac{\partial \sigma_y}{\partial D} , \quad \frac{\partial f}{\partial T_p} = -\frac{\partial \sigma_y}{\partial T_p} \tag{17.141}$$

Assume the associated flow rule

$$\boldsymbol{d}^p = \dot{\lambda}\,\hat{\boldsymbol{N}} = \dot{\lambda}\frac{\boldsymbol{N}}{\|\boldsymbol{N}\|} \quad \text{where} \quad \boldsymbol{N} = \frac{\partial f}{\partial \boldsymbol{\sigma}} = \frac{\partial f}{\partial \boldsymbol{\xi}} = \sqrt{\tfrac{3}{2}}\,\frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|} , \quad \|\boldsymbol{N}\| = \sqrt{\tfrac{3}{2}} \tag{17.142}$$

Then

$$\boldsymbol{d}^p = \dot{\lambda}\,\frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|} ; \quad \text{dev}(\boldsymbol{d}^p) = \boldsymbol{d}^p , \quad \text{tr}(\boldsymbol{d}^p) = 0, \quad \left\|\boldsymbol{d}^p\right\| = \dot{\varepsilon}^{\text{eq}}_p = \dot{\lambda} . \tag{17.143}$$

The evolution of the equivalent plastic strain is given by

$$\dot{\varepsilon}^{\text{eq}}_p = \dot{\lambda}\,h^{\varepsilon_p} \quad \Longrightarrow \quad h^{\varepsilon_p} = 1 . \tag{17.144}$$

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

$$\dot{\phi} = \dot{\lambda}\,h^{\phi} = 0 \quad \Longrightarrow \quad h^{\phi} = 0 . \tag{17.145}$$

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

$$\dot{\boldsymbol{\beta}} = \dot{\lambda}\boldsymbol{h}^{\beta} = H'\boldsymbol{d}^{p} \quad \Longrightarrow \quad \boldsymbol{h}^{\beta} = H'\frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|} \tag{17.146}$$

where H' is a hardening modulus. For the Armstrong-Frederick kinematic hardening model,

$$\dot{\boldsymbol{\beta}} = \dot{\lambda}\ \boldsymbol{h}^{\beta} = H_1\ \boldsymbol{d}^{p} - H_2\ \boldsymbol{\beta}\ \left\|\boldsymbol{d}^{p}\right\| = \dot{\lambda}\left[H_1\ \frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|} - H_2\ \boldsymbol{\beta}\right] \quad \Longrightarrow \quad \boldsymbol{h}^{\beta} = H_1\ \frac{\boldsymbol{\xi}}{\|\boldsymbol{\xi}\|} - H_2\ \boldsymbol{\beta} \tag{17.147}$$

17.4 Example 2: Gurson-type model

Consider a Gurson-type yield condition with kinematic hardening. In this case the yield condition can be written as

$$f := \tfrac{3}{2}\frac{(\sigma^{\xi}_{\text{eff}})^2}{\sigma_y^2} + 2\ q_1\ \phi^{\star}\ \cosh\left(\tfrac{3}{2}\frac{q_2\ p_\beta}{\sigma_y}\right) - \left[1 + q_3\ (\phi^{*})^2\right] \tag{17.148}$$

where σ_y is the yield stress of the matrix material (zero-porosity),

$$\begin{aligned}
&\sigma^{\xi}_{\text{eff}} = \boldsymbol{\xi} : \boldsymbol{\xi}\,, \quad \boldsymbol{\xi} = \text{dev}(\boldsymbol{\sigma} - \boldsymbol{\beta})\,, \quad p_\beta = \tfrac{1}{3}\text{tr}(\boldsymbol{\sigma} - \boldsymbol{\beta}) \\
&\phi^{\star} = \begin{cases} \phi & \text{for } \phi \le \phi_c \\ \phi_c - \frac{\phi^{\star}_u - \phi_c}{\phi_f - \phi_c}\ (\phi - \phi_c) & \text{for } \phi > \phi_c \end{cases}
\end{aligned} \tag{17.149}$$

and ϕ is the porosity. Final fracture occurs for $\phi = \phi_f$ or when $\phi^{\star}_u = 1/q_1$. In this case, the derivatives of f are

$$\frac{\partial f}{\partial \sigma^{\xi}_{\text{eff}}} = \frac{3\sigma^{\xi}_{\text{eff}}}{\sigma_y^2}\,, \quad \frac{\partial f}{\partial p_\beta} = \frac{3q_1q_2\phi^{\star}}{\sigma_y}\sinh\left(\tfrac{3}{2}\frac{q_2\ p_\beta}{\sigma_y}\right)\,, \quad \frac{\partial f}{\partial \sigma_y} = -\left[\frac{3(\sigma^{\xi}_{\text{eff}})^2}{\sigma_y^3} + \frac{3q_1q_2p_\beta\phi^{\star}}{\sigma_y^2}\sinh\left(\tfrac{3}{2}\frac{q_2\ p_\beta}{\sigma_y}\right)\right] \tag{17.150}$$

and

$$\frac{\partial f}{\partial \boldsymbol{\beta}} = -\left[2\frac{\partial f}{\partial \sigma^{\xi}_{\text{eff}}}\boldsymbol{\xi} + \frac{\partial f}{\partial p_\beta}\boldsymbol{I}\right]\,, \quad \frac{\partial f}{\partial \varepsilon^{\text{eq}}_p} = \frac{\partial f}{\partial \sigma_y}\frac{\partial \sigma_y}{\partial \varepsilon^{\text{eq}}_p}\,, \quad \frac{\partial f}{\partial D} = \frac{\partial f}{\partial \sigma_y}\frac{\partial \sigma_y}{\partial D}\,, \quad \frac{\partial f}{\partial T_p} = \frac{\partial f}{\partial \sigma_y}\frac{\partial \sigma_y}{\partial T_p} \tag{17.151}$$

For the derivative with respect to ϕ,

$$\frac{\partial f}{\partial \phi} = \frac{\partial f}{\partial \phi^{\star}}\frac{\partial \phi^{\star}}{\partial \phi} + \frac{\partial f}{\partial \sigma_y}\frac{\partial \sigma_y}{\partial \phi} = \frac{\partial f}{\partial \phi^{\star}}\frac{\partial \phi^{\star}}{\partial \phi} \tag{17.152}$$

where

$$\frac{\partial f}{\partial \phi^{\star}} = 2q_1\cosh\left(\tfrac{3}{2}\frac{q_2\ p_\beta}{\sigma_y}\right) - 2q_3\phi^{\star} \quad \text{and} \quad \frac{\partial \phi^{\star}}{\partial \phi} = \begin{cases} 1 & \text{for } \phi \le \phi_c \\ -\frac{\phi^{\star}_u - \phi_c}{\phi_f - \phi_c} & \text{for } \phi > \phi_c \end{cases} \tag{17.153}$$

Using an associated flow rule, we have

$$\boldsymbol{d}^{p} = \dot{\lambda}\ \hat{\boldsymbol{N}} = \dot{\lambda}\frac{\boldsymbol{N}}{\|\boldsymbol{N}\|}\,, \quad \boldsymbol{N} = \frac{\partial f}{\partial \boldsymbol{\sigma}} = 2\frac{\partial f}{\partial \sigma^{\xi}_{\text{eff}}}\boldsymbol{\xi} + \frac{\partial f}{\partial p_\beta}\boldsymbol{I}\,, \quad \|\boldsymbol{N}\| = \sqrt{4\left(\frac{\partial f}{\partial \sigma^{\xi}_{\text{eff}}}\|\boldsymbol{\xi}\|\right)^2 + 3\left(\frac{\partial f}{\partial p_\beta}\right)^2} \tag{17.154}$$

For the evolution equation for the plastic strain we use

$$(\boldsymbol{\sigma}-\boldsymbol{\beta}):\boldsymbol{d}^p=(1-\phi)\sigma_y\dot{\varepsilon}_p^{\text{eq}} \tag{17.155}$$

where $\dot{\varepsilon}_p^{\text{eq}}$ is the equivalent plastic strain rate in the matrix material. Hence,

$$\dot{\varepsilon}_p^{\text{eq}}=\dot{\lambda}\,h^{\varepsilon_p}=\dot{\lambda}\,\frac{(p_\beta\boldsymbol{I}+\boldsymbol{\xi}):\hat{\boldsymbol{N}}}{(1-\phi)\,\sigma_y}\quad\Longrightarrow\quad h^{\varepsilon_p}=\frac{3p_\beta\dfrac{\partial f}{\partial p_\beta}+2\dfrac{\partial f}{\partial\sigma_{\text{eff}}^{\xi}}\boldsymbol{\xi}:\boldsymbol{\xi}}{(1-\phi)\,\sigma_y\,\|\boldsymbol{N}\|} \tag{17.156}$$

The evolution equation for the porosity is assumed to be given by

$$\dot{\phi}=(1-\phi)\,\text{tr}(\boldsymbol{d}^p)+A\,\dot{\varepsilon}_p^{\text{eq}} \tag{17.157}$$

where

$$A=\frac{f_n}{\varepsilon_s\sqrt{2\pi}}\exp\left[-\tfrac{1}{2}\frac{(\varepsilon_p^{\text{eq}}-\varepsilon_m)^2}{\varepsilon_s^2}\right] \tag{17.158}$$

and f_n is the volume fraction of void nucleating particles, ε_m is the mean of the normal distribution of nucleation strains, and ε_s is the standard deviation of the distribution. Therefore,

$$\dot{\phi}=\dot{\lambda}\,h^{\phi}=\dot{\lambda}\left[(1-\phi)\,\text{tr}(\hat{\boldsymbol{N}})+A\,\frac{(p_\beta\boldsymbol{I}+\boldsymbol{\xi}):\hat{\boldsymbol{N}}}{(1-\phi)\,\sigma_y}\right]\Longrightarrow\ h^{\phi}=\frac{1}{\|\boldsymbol{N}\|}\left[3(1-\phi)\frac{\partial f}{\partial p_\beta}+A\frac{3p_\beta\dfrac{\partial f}{\partial p_\beta}+2\dfrac{\partial f}{\partial\sigma_{\text{eff}}^{\xi}}\boldsymbol{\xi}:\boldsymbol{\xi}}{(1-\phi)\,\sigma_y}\right] \tag{17.159}$$

If the evolution of the backstress is given by the Prager kinematic hardening rule

$$\dot{\boldsymbol{\beta}}=\dot{\lambda}\,\boldsymbol{h}^{\beta}=H'\,\boldsymbol{d}^p\quad\Longrightarrow\ \boldsymbol{h}^{\beta}=\frac{H'}{\|\boldsymbol{N}\|}\left[2\frac{\partial f}{\partial\sigma_{\text{eff}}^{\xi}}\boldsymbol{\xi}+\frac{\partial f}{\partial p_\beta}\boldsymbol{I}\right] \tag{17.160}$$

For the Armstrong-Frederick model,

$$\dot{\boldsymbol{\beta}}=\dot{\lambda}\,\boldsymbol{h}^{\beta}=H_1\,\boldsymbol{d}^p-H_2\,\boldsymbol{\beta}\,\|\boldsymbol{d}^p\|\quad\Longrightarrow\ \boldsymbol{h}^{\beta}=\frac{H_1}{\|\boldsymbol{N}\|}\left(2\frac{\partial f}{\partial\sigma_{\text{eff}}^{\xi}}\boldsymbol{\xi}+\frac{\partial f}{\partial p_\beta}\boldsymbol{I}\right)-H_2\boldsymbol{\beta}\,. \tag{17.161}$$

17.5 Example 3: Nonlinear elasticity and isotropic hardening

Let the flow stress be given by the Johnson-Cook model:

$$\sigma_y(\varepsilon_p^{\text{eq}},\dot{\varepsilon}^{\text{eq}},T)=\left[A+B(\varepsilon_p^{\text{eq}})^n\right]\left[1+C\ln(\dot{\varepsilon}^{\star})\right]\left[1-(T^{\star})^m\right] \tag{17.162}$$

The volumetric part of the stress in the intact metal is given by a Mie-Grüneisen equation of state:

$$p(J^e,T)=-\left[\frac{\rho_{\text{o}}\,C_{\text{o}}^2\,(1-J^e)[1-\Gamma_{\text{o}}(1-J^e)/2]}{[1-S_\alpha(1-J^e)]^2}+\Gamma_{\text{o}}\,E\right],\quad J^e=\det\boldsymbol{F}^e\,,\quad E\approx\frac{\rho C_v(T-T_{\text{o}})}{V} \tag{17.163}$$

The tangent bulk modulus of the intact metal is defined as

$$\kappa_m(p,J^e,T)=V^e\frac{\partial p}{\partial V^e}=V^e\frac{\partial p}{\partial J^e}\frac{\partial J^e}{\partial V^e}=\frac{V^e}{V_{\text{o}}}\frac{\partial p}{\partial J^e}=J^e\frac{\partial p}{\partial J^e}\approx\frac{\rho_{\text{o}}J^eC_{\text{o}}^2\,[1+(S_\alpha-\Gamma_{\text{o}})(1-J^e)]}{[1-S_\alpha(1-J^e)]^3}. \tag{17.164}$$

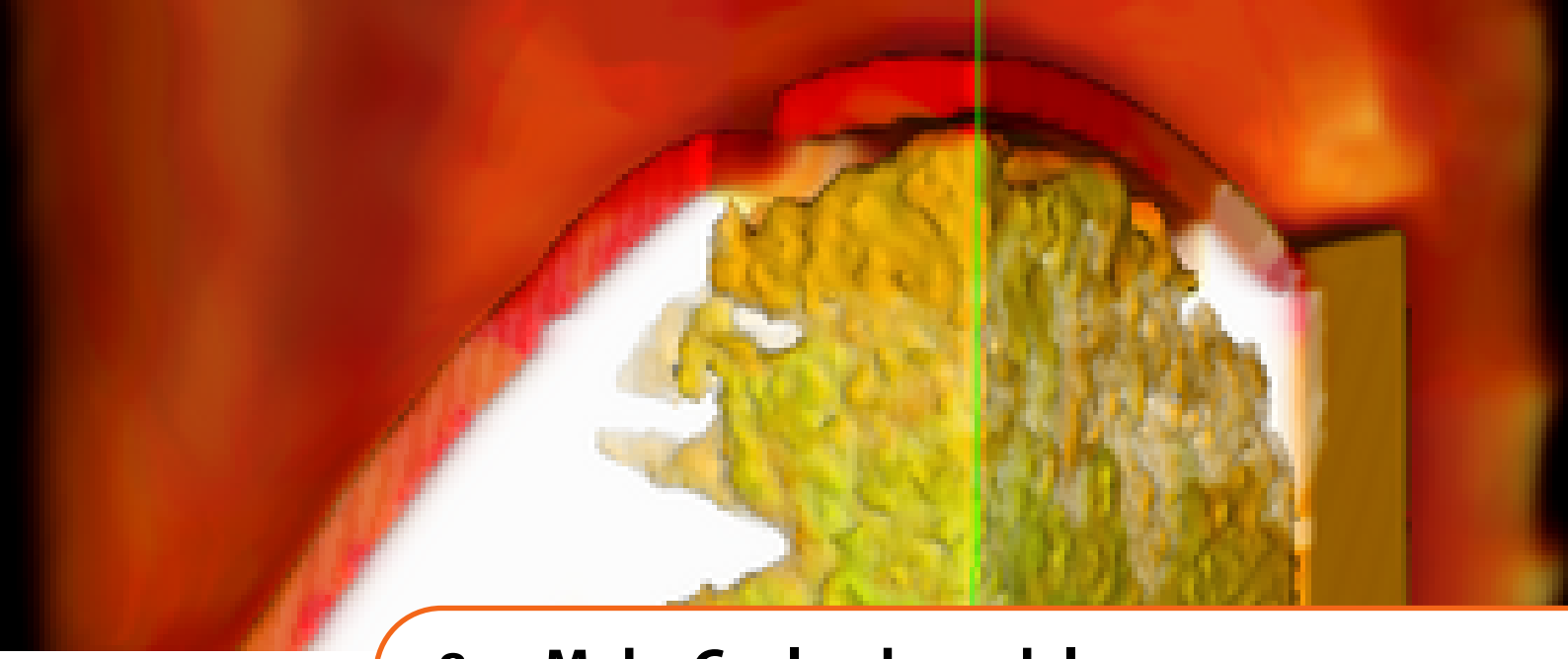
Since the rate of deformation is unrotated in the VAANGO metal plasticity implementation, we can identify $\mathrm{tr}(\boldsymbol{\varepsilon}^e)$ with $\ln(J^e)$ [72] and use that quantity in the calculation. The deviatoric part of the stress in the intact metal is given by the Steinberg-Cochran-Guinan (SCG) shear modulus model:

$$\mu_m(p, J^e, T) = \mu_0 + p\frac{\partial \mu_m}{\partial p}(J^e)^{1/3} + \frac{\partial \mu_m}{\partial T}(T - T_0) \tag{17.165}$$

When we include porosity-dependence for the bulk and shear moduli, we have

$$\begin{aligned} \kappa(p, J^e, \phi, T) &= \frac{(1-\phi)\kappa_m}{1-(1-K)\phi}\,, \quad K = \frac{3\kappa_m + 4\mu_m}{4\mu_m} \\ \mu(p, J^e, \phi, T) &= \frac{(1-\phi)\mu_m}{1-(1-G)\phi}\,, \quad G = \frac{5(3\kappa_m + 4\mu_m)}{9\kappa_m + 8\mu_m} \end{aligned} \tag{17.166}$$

The derivatives of σ_y, κ, and μ required for the algorithm can be calculated from these expressions.



18 — Mohr-Coulomb model

18.1 Introduction

The classical Mohr-Coulomb model is a workhorse of rock and soil plasticity modeling. This model is typically hard to implement for implicit codes because of the difficulties encountered in computing tangent stiffness matrices near the corners (as viewed from the hydrostatic axis). However, that problem is not encountered in explicit codes and corners can be handled relatively easily.

The VAANGO implementation of Mohr-Coulomb plasticity uses a linear elastic model and perfect plasticity. There are also features that allow the shear modulus, cohesion etc. to vary with deformation and for the effect of water content to be modelled without a fully coupled saturation/porosity model. A nonlocal correction features is also included.

Stresses and the rate-of-deformation are unrotated using the beginning of the timestep deformation gradient polar decomposition before any constitutive relations are evaluated. The updated stress is rotated back using the deformation gradient decomposition at the end of the time step.

The convention used for this model is that stresses are positive in compression and that the principal stresses are in the order $\sigma_1 > \sigma_2 > \sigma_3$. The model assumes that the plastic potential (alternatively referred to as the dilation model) and yield function have the same form but the angles may differ. The angle of the plastic potential function is denoted ψ.

The implementation is largely in Voigt notation with stresses arranged in the sequence (σ_{11}, σ_{22}, σ_{33}, σ_{12}, σ_{13}, σ_{23}).

18.2 Elasticity model

Isotropic linear hypoelastic behavior is assumed, i.e, the stress-rate $\dot{\boldsymbol{\sigma}}$ is linearly related to the rate-of-deformation $\boldsymbol{d}$.

$$\dot{\boldsymbol{\sigma}} = \left(K - \tfrac{2}{3}G\right)\operatorname{tr}(\boldsymbol{d})\boldsymbol{I} + 2G\boldsymbol{d} \tag{18.1}$$

where K is the bulk modulus and G is the shear modulus. These are related to the Young's modulus (E) and the Poisson's ratio (ν) by

$$K = \frac{E}{3(1-2\nu)} \quad \text{and} \quad G = \frac{E}{2(1+\nu)}\,. \tag{18.2}$$

The elastic tangent modulus at any time t is given by

$$\mathbb{C} = \begin{bmatrix} K+\frac{4}{3}G & K-\frac{2}{3}G & K-\frac{2}{3}G & 0 & 0 & 0 \\ K-\frac{2}{3}G & K+\frac{4}{3}G & K-\frac{2}{3}G & 0 & 0 & 0 \\ K-\frac{2}{3}G & K-\frac{2}{3}G & K+\frac{4}{3}G & 0 & 0 & 0 \\ 0 & 0 & 0 & 2G & 0 & 0 \\ 0 & 0 & 0 & 0 & 2G & 0 \\ 0 & 0 & 0 & 0 & 0 & 2G \end{bmatrix} \tag{18.3}$$

A variable modulus model that depends on a varying cohesion can also be activated if desired. The model has the form

$$G(t) = \frac{Mc(t)}{2(1+\nu_y)}\,, \quad K(t) = \frac{Mc(t)}{3(1-2\nu_y)} \tag{18.4}$$

where M and ν_y are model parameters, and $c(t)$ is a time-varying cohesive strength. The model is initialized such that $G(0) = G$ and $K(0) = K$ when $c(0) = c$, the initial cohesive strength, i.e., $E = Mc$ and $\nu_y = \nu$.

18.3 Yield functions

The model includes two variations on the shape of the yield surface ($f(\boldsymbol{\sigma}) = 0$):

- the classical model
- the Sheng et al. variation of the yield surface [73].

The second does not have sharp edges but is not strictly convex and should not be applied when stress states close to the vertex of Mohr-Coulomb cone are expected. Plastic states are achieved when $f(\boldsymbol{\sigma}) \geq 0$ and elastic states when $f(\boldsymbol{\sigma}) < 0$.

18.3.1 Classical Mohr-Coulomb yield surface

The classical Mohr-Coulomb yield surface expressed in terms of the principal stresses is

$$\begin{aligned} \pm\frac{\sigma_1-\sigma_2}{2} &= \left[\frac{\sigma_1+\sigma_2}{2}\right]\sin(\phi) + c\cos(\phi) \\ \pm\frac{\sigma_2-\sigma_3}{2} &= \left[\frac{\sigma_2+\sigma_3}{2}\right]\sin(\phi) + c\cos(\phi) \\ \pm\frac{\sigma_1-\sigma_3}{2} &= \left[\frac{\sigma_1+\sigma_3}{2}\right]\sin(\phi) + c\cos(\phi). \end{aligned} \tag{18.5}$$

where c is the cohesive strength and ϕ is the angle of internal friction.

The eigenvalues of the stress tensor can be computed in closed form. The resulting expressions are

$$\sigma_1 = p + \frac{2}{3}q\,\cos\theta \quad \text{and} \quad \sigma_3 = p + \frac{2}{3}q\,\cos\left(\theta + \frac{2\pi}{3}\right) \tag{18.6}$$

where

$$p = \frac{1}{3}I_1\,, \quad q = \sqrt{3J_2}\,, \quad \cos 3\theta = \left(\frac{r}{q}\right)^3 = \frac{3\sqrt{3}}{2}\frac{J_3}{J_2^{3/2}}\,, \quad r^3 = \frac{27}{2}J_3 \tag{18.7}$$

and

$$I_1 = \operatorname{tr}(\boldsymbol{\sigma}), \quad J_2 = \frac{1}{2}\mathbf{s}:\mathbf{s}, \quad J_3 = \det(\mathbf{s}), \quad \mathbf{s} = \boldsymbol{\sigma} - \frac{I_1}{3}\boldsymbol{I}\,. \tag{18.8}$$

Summary 18.3.1 ***Classical Mohr-Coulomb yield function in terms of invariants***

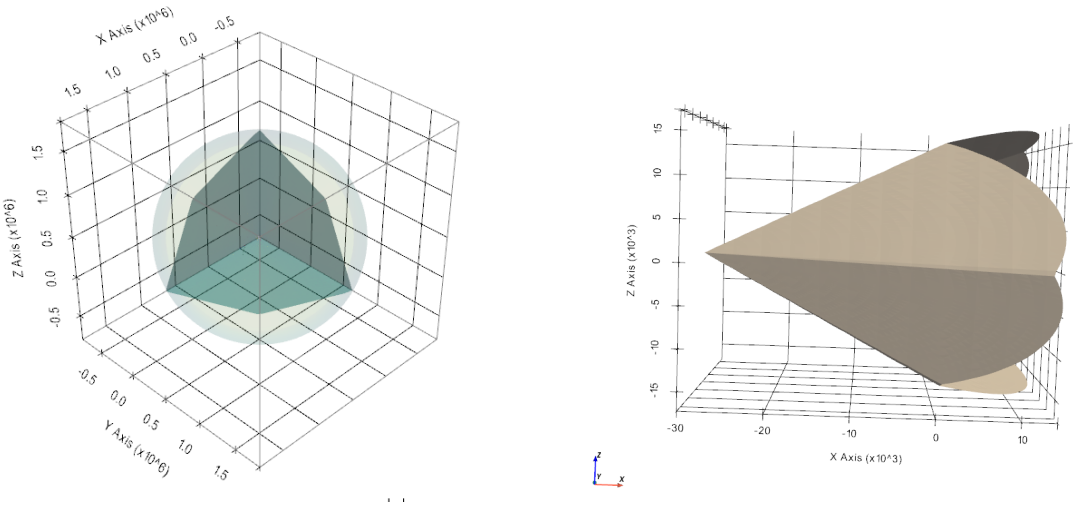
In terms of these invariants, the Mohr-Coulomb yield function in (18.5) can be expressed as

$$f(\boldsymbol{\sigma}) = R(\theta)\, q - p\, \sin\phi - c\cos\phi \tag{18.9}$$

where

$$R(\theta) = \frac{1}{\sqrt{3}} \sin\left(\theta + \frac{\pi}{3}\right) - \frac{1}{3}\sin\phi\, \cos\left(\theta + \frac{\pi}{3}\right). \tag{18.10}$$

Plots of the Mohr-Coulomb surface in the octahedral and Rendulic planes are shown in Figure 18.1.



(a) Octahedral profile. (b) Rendulic profile.

Figure 18.1: Profiles of the classical Mohr-Coulomb yield surface.

18.3.2 Sheng et al. yield surface

The yield function of the Mohr-Coulomb yield surface can be expressed in p–q space as

$$f(\boldsymbol{\sigma}) = q - Mp - \tilde{c}\,. \tag{18.11}$$

Comparison with (18.9),

$$f(\boldsymbol{\sigma}) = R(\theta)\, q - p\, \sin\phi - c\cos\phi \tag{18.12}$$

indicates that

$$M(\theta) = \frac{\sin\phi}{R(\theta)} \quad \text{and} \quad \tilde{c}(\theta) = \frac{c\cos\phi}{R(\theta)} \tag{18.13}$$

for the classical Mohr-Coulomb model.

Summary **18.3.2** ***Sheng et al. Mohr-Coulomb yield function***

Sheng et al. [73] suggest a modified model designed for CAMClay type models, which when applied to the Mohr-Coulomb yield function takes the form

$$f(\boldsymbol{\sigma}) = q - \tilde{M}p - \tilde{c}\,. \tag{18.14}$$

where

$$\tilde{M}(\theta) = M(\theta = \pi/3)\left(\frac{2\alpha^4}{1+\alpha^4+(1-\alpha^4)\cos 3\theta}\right)^{1/4}, \quad \text{and} \quad \alpha = \frac{3-\sin\phi}{3+\sin\phi}\,. \tag{18.15}$$

Note that from (18.10),

$$R(\pi/3) = \frac{3+\sin\phi}{6} \quad \Longrightarrow \quad M(\pi/3) = \frac{6\sin\phi}{3+\sin\phi}\,. \tag{18.16}$$

Plots of the modified yield surface surface in the octahedral and front view are shown in Figure 18.2. This yield surface is not convex and should be avoided in computations.

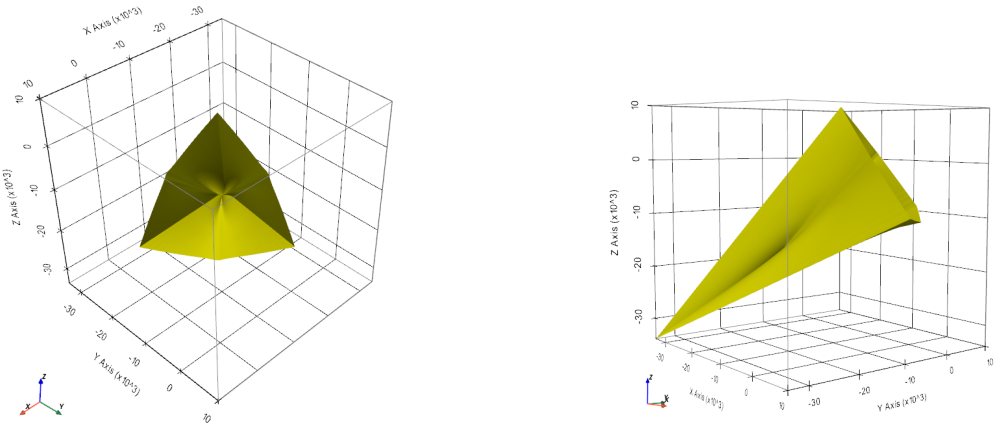


Figure 18.2: Profiles of the modified Mohr-Coulomb yield surface.

Note that the angle θ_s in [73] is defined as

$$\sin 3\theta_s = -\frac{27}{2}\frac{J_3}{q^3} = -\cos 3\theta\,, \quad \theta_s = \theta - \frac{\pi}{6}\,. \tag{18.17}$$

For that definition, triaxial compression occurs at $\theta_s = \pi/6$ whereas with our definition it occurs at $\theta = \pi/3$.

A variant of the model in (18.14) is implemented in VAANGO :

$$f = \frac{q}{M} - \frac{2c}{M} - p\,, \quad M = \frac{M_o\alpha}{\left[\frac{1}{2}\left[1+\alpha^4-(1-\alpha^4)\sin(3\theta_s)\right]\right]^{1/4}}\,, \quad M_o = \frac{6\sin\phi}{3-\sin\phi}\,. \tag{18.18}$$

This model is visualized in Figure 18.3 and is not convex.

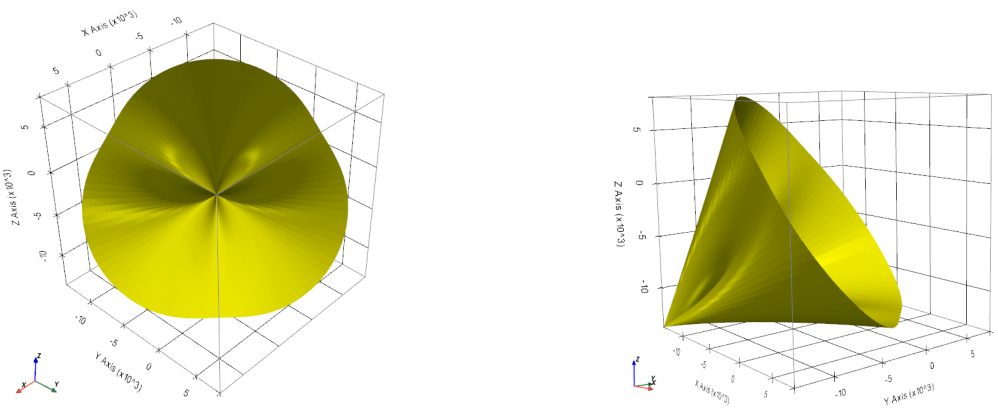


Figure 18.3: Profiles of the modified Mohr-Coulomb yield surface as implemented in VAANGO .

18.4 Variable cohesive strength

A time varying cohesive strength model can be activated if necessary. Under certain circumstances, the cohesive strength is assumed to depend on the effective strain ε_{eff} (see definition in (2.9)) and an effective strain-rate $\dot{\varepsilon}_{\text{eff}}$. The strain is computed from the unrotated rate-of-deformation $\boldsymbol{d}$ using

$$\boldsymbol{\varepsilon}(t_{n+1}) = \boldsymbol{\varepsilon}(t_n) + \boldsymbol{d}\Delta t \quad \text{where} \quad \Delta t = t_{n+1} - t_n \,. \tag{18.19}$$

The effective strain is then computed using

$$\varepsilon_{\text{eff}} = \sqrt{\frac{2}{3}\left[\frac{1}{3}\left[(\varepsilon_{11}-\varepsilon_{22})^2 + (\varepsilon_{22}-\varepsilon_{33})^2 + (\varepsilon_{33}-\varepsilon_{11})^2\right] + 2\left(\varepsilon_{12}^2 + \varepsilon_{23}^2 + \varepsilon_{13}^2\right)\right]}\,. \tag{18.20}$$

An effective strain-rate, $\dot{\varepsilon}_{\text{eff}}$, is also computed from

$$\dot{\varepsilon}_{\text{eff}} = \sqrt{\frac{2}{3}\left[\frac{1}{3}\left[(d_{11}-d_{22})^2 + (d_{22}-d_{33})^2 + (d_{33}-d_{11})^2\right] + 2\left(d_{12}^2 + d_{23}^2 + d_{13}^2\right)\right]}\,. \tag{18.21}$$

For situations during which a rate-dependent undrained shear transition is important, the cohesion model takes the form

$$c(t) = \begin{cases} S_t a_1 W^{-b_1} & \text{for } \dot{\varepsilon}_{\text{eff}} \le \dot{\varepsilon}_{\text{ref}} \\ S_t a_1 W^{-b_1} \left(\dfrac{\dot{\varepsilon}_{\text{eff}}}{\dot{\varepsilon}_{\text{ref}}}\right)^{\beta} & \text{for } \dot{\varepsilon}_{\text{eff}} > \dot{\varepsilon}_{\text{ref}} \end{cases} \tag{18.22}$$

where S_t is a softening parameter, a_1, b_1 are water influence parameters, $\dot{\varepsilon}_{\text{ref}}$ is a reference strain-rate, and β is a strain-rate parameter.

If the cohesion varies linearly with depth, the model that can be used to compute depth-dependent values has the form

$$c(t) = c + A(\mathbf{x}_p \cdot \mathbf{n}_d - y_{\text{ref}}) \tag{18.23}$$

where A is a slope parameter and y_{ref} is a reference depth value. The particle position is $\mathbf{x}_p$ and $\mathbf{n}_d$ is the depth direction (aligned with the axes of the computational domain).

If softening is activated, the cohesive strength is modified if the condition

$$\varepsilon_{\text{eff}} > \frac{c}{G} \tag{18.24}$$

where c is the cohesive strength and G is the shear modulus. The softening model has the form

$$c(t) = c\left[\frac{1}{S_t} + \left(1 - \frac{1}{S_t}\right) 2.71^{\varepsilon_{\text{eff}}/\varepsilon_{95}}\right] \tag{18.25}$$

where S_t is the softening parameter used earlier and ε_{95} is a reference effective strain.

Finally, if a water retention model is used to modify the cohesive strength, a suction value ψ_m is computed and the cohesion is computed as

$$c(t) = c + \psi_m \tan\phi_b \tag{18.26}$$

where ϕ_b is a water retention parameter. The suction is computed using the van Genuchten model:

$$\theta(\psi) = \theta_r + \frac{\theta_s - \theta_r}{\left[1 + (\alpha|\psi_m|)^n\right]^{1-1/n}} \tag{18.27}$$

where $\theta(\psi)$ is the water retention curve, $|\psi_m|$ is the suction pressure, θ_s is the saturated water content, θ_r is the residual water content, $\alpha > 0$ is a parameter related to the inverse of the air entry suction, and $n > 1$ is a measure of the pore-size distribution.

18.5 Flow rule

A non-associated flow rule is assumed such that the plastic strain-rate $\boldsymbol{d}_p$ is given by

$$\boldsymbol{d}_p = \dot{\lambda}\frac{\partial g}{\partial \boldsymbol{\sigma}} \tag{18.28}$$

where

$$g(\boldsymbol{\sigma}) = R(\theta)\, q - p\, \sin\psi - c\cos\psi \tag{18.29}$$

and ψ is the dilation angle. Typically ψ is taken to be equal to ϕ, the friction angle.

The normal to the plastic potential surface is given by

$$\boldsymbol{n} = \frac{\partial g}{\partial \boldsymbol{\sigma}} = \frac{dR}{d\theta}\frac{\partial \theta}{\partial \boldsymbol{\sigma}}\, q + R(\theta)\,\frac{\partial q}{\partial \boldsymbol{\sigma}} - \frac{\partial p}{\partial \boldsymbol{\sigma}}\,\sin\psi\,. \tag{18.30}$$

where

$$\begin{aligned}
\frac{dR}{d\theta} &= \tfrac{1}{\sqrt{3}}\,\cos\left(\theta + \frac{\pi}{3}\right) + \tfrac{1}{3}\sin\psi\,\sin\left(\theta + \frac{\pi}{3}\right)\\
\frac{\partial \theta}{\partial \boldsymbol{\sigma}} &= -\frac{1}{\sin 3\theta}\left[\frac{9}{2q^3}\frac{\partial J_3}{\partial \boldsymbol{\sigma}} - \frac{r^3}{q^4}\frac{\partial q}{\partial \boldsymbol{\sigma}}\right] = -\frac{1}{\sin 3\theta}\left[\frac{9}{2q^3}\left(\mathbf{s}\cdot\mathbf{s} - \tfrac{2}{3}J_2\boldsymbol{I}\right) - \frac{r^3}{q^4}\frac{\partial q}{\partial \boldsymbol{\sigma}}\right]\\
\frac{\partial q}{\partial \boldsymbol{\sigma}} &= \frac{\sqrt{3}}{2\sqrt{J_2}}\frac{\partial J_2}{\partial \boldsymbol{\sigma}} = \frac{\sqrt{3}}{2\sqrt{J_2}}\mathbf{s}\\
\frac{\partial p}{\partial \boldsymbol{\sigma}} &= \tfrac{1}{3}\frac{\partial I_1}{\partial \boldsymbol{\sigma}} = \tfrac{1}{3}\boldsymbol{I}\,.
\end{aligned} \tag{18.31}$$

18.6 Nonlocal shear correction

The Mohr-Coulomb model contains a nonlocal correction feature that uses neighbor information to regularize solutions. Neighboring particles that contribute to the nonlocal effect are identified using a nonlocal length (ℓ_n). Let there be N_q such particles in the neighborhood of particle p.

The nonlocal effective strain ($\varepsilon^n_{\text{eff}}$) and effective strain rate ($\dot{\varepsilon}^n_{\text{eff}}$) are computed as

$$\varepsilon^n_{\text{eff}} = (1-n)\varepsilon_{\text{eff}} + n\left(\frac{\sum_{q=1}^{N_q} V_w^q \varepsilon_{\text{eff}}}{\sum_{q=1}^{N_q} V_w^q}\right) \quad \text{and} \quad \dot{\varepsilon}^n_{\text{eff}} = (1-n)\dot{\varepsilon}_{\text{eff}} + n\left(\frac{\sum_{q=1}^{N_q} V_w^q \dot{\varepsilon}_{\text{eff}}}{\sum_{q=1}^{N_q} V_w^q}\right) \tag{18.32}$$

where n is a nonlocal parameter, ε_{eff}, $\dot{\varepsilon}_{\text{eff}}$ are the local effective strain and strain-rate, respectively, and V_w^q is a weighted volume of neighboring particle q whose local volume is V_q. The expression for V_w^q is

$$V_w^q = w_{pq} V_q \quad \text{where} \quad w_{pq} = \frac{\ell}{\ell_n} \exp\left(-\frac{\ell^2}{\ell_n^2}\right), \quad \ell = \left\|\mathbf{x}_q - \mathbf{x}_p\right\| . \tag{18.33}$$

If a regularization flag is activated, and $\varepsilon^n_{\text{eff}} > c/G$, where c is the cohesion and G is the shear modulus, a time-scale is included in the computation:

$$\varepsilon^n_{\text{eff}} \leftarrow \varepsilon^n_{\text{eff}} \frac{t_{\text{FE}}}{t_{\text{shear}}}, \quad \dot{\varepsilon}^n_{\text{eff}} \leftarrow \dot{\varepsilon}^n_{\text{eff}} \frac{t_{\text{FE}}}{t_{\text{shear}}} . \tag{18.34}$$

where t_{FE} and t_{shear} are regularization time scales. The nonlocal effective strain is typically not used directly in the model except for modifying the cohesion and the elastic moduli.

However, after the stress has been updated, a nonlocal correction can be applied using a similar approach:

$$\boldsymbol{\sigma}^n = (1-n)\boldsymbol{\sigma} + n\left(\frac{\sum_{q=1}^{N_q} V_w^q \boldsymbol{\sigma}}{\sum_{q=1}^{N_q} V_w^q}\right) \tag{18.35}$$

where $\boldsymbol{\sigma}_n$ is the nonlocal stress.

18.7 Explicit stress integration

An additive decomposition of the unrotated rate-of-deformation into elastic and plastic parts is assumed:

$$\boldsymbol{d} = \boldsymbol{d}_e + \boldsymbol{d}_p \tag{18.36}$$

Elastic and plastic strains are defined using

$$\boldsymbol{\varepsilon} = \int_0^t \boldsymbol{d}(\tau) d\tau = \boldsymbol{\varepsilon}_e + \boldsymbol{\varepsilon}_p \quad \Longrightarrow \quad \boldsymbol{\varepsilon}_e = \int_0^t \boldsymbol{d}_e(\tau) d\tau, \quad \boldsymbol{\varepsilon}_p = \int_0^t \boldsymbol{d}_p(\tau) d\tau . \tag{18.37}$$

The strain increment during a timestep, $\Delta t = t_{n+1} - t_n$, is computed as

$$\Delta\boldsymbol{\varepsilon} = \boldsymbol{d}\Delta t . \tag{18.38}$$

The stress update begins with a check whether the stress state at the beginning of the time step is on the yield surface. This is necessary because the modification of the cohesion and the elastic moduli may have affected the shape of the yield surface. Special checks are used to determine if the strain increment leads to unloading. Details are omitted for brevity.

An elastic trial stress state is computed

$$\boldsymbol{\sigma}^{\text{trial}} = \boldsymbol{\sigma}_n + \mathbb{C}_n : \Delta\boldsymbol{\varepsilon} \tag{18.39}$$

where $\mathbb{C}$ tangent elastic modulus tensor.

If the trial stress is in the plastic region, the intersection of the stress increment vector $\Delta\boldsymbol{\sigma} = \boldsymbol{\sigma}^{\text{trial}} - \boldsymbol{\sigma}_n$ with the yield surface is computed using bisection. The strain increment corresponding to this reduced

stress increment is the increment in elastic strain and the stress on the yield surface is the stress at the end of the elastic strain increment.

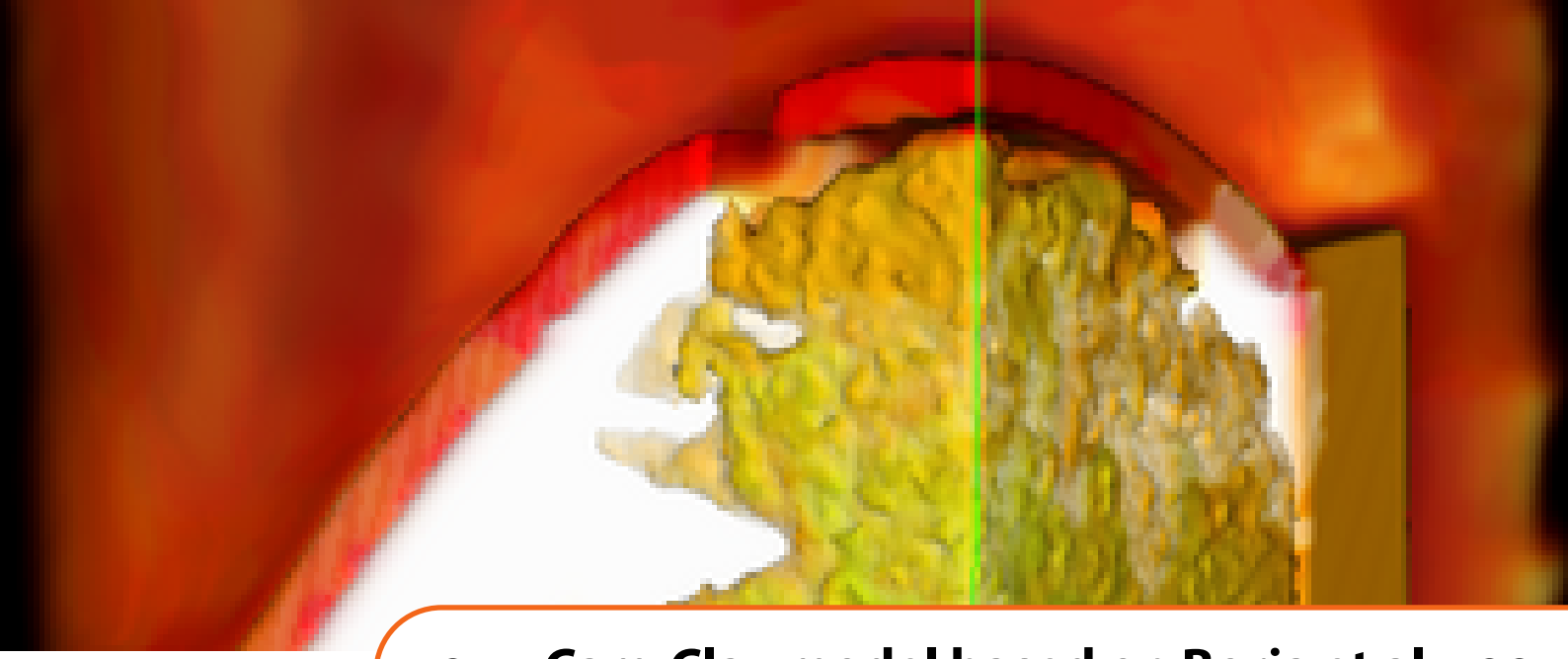
The remainder of the strain increment is purely plastic. The trial stress now has to be projected back to the yield surface along the projection vector $\boldsymbol{P}$ [74], given by

$$\boldsymbol{P} = -\mathbb{C} : \frac{\partial g}{\partial \boldsymbol{\sigma}} \,. \tag{18.40}$$

The intersection of the projection vector with the yield surface is found via bisection. The updated stress is the intersection point on the yield surface.

Several Runge-Kutta schemes are implemented in the VAANGO code to break-up large The intersection of the projection vector with the yield surface is found via bisection. strain increments into smaller steps. However, the two-step modified Euler scheme is accurate enough for our purposes.

This procedure is repeated for each timestep.



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

19.1 Introduction

The Cam-clay plasticity model and its modified ellipsoidal version [75–78] is widely considered to be an accurate model for the prediction of the compressive and shear behavior of clays. The Borja model [35, 79–81] 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.

19.2 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.

19.2.1 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

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

19.2.2 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

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

19.3 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:

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

19.3.1 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

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

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

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

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

$$\frac{\partial f}{\partial \boldsymbol{\sigma}} = \frac{2p - p_c}{3}\,\boldsymbol{1} + \sqrt{\tfrac{3}{2}}\,\frac{2q}{M^2}\,\boldsymbol{n}\,. \tag{19.1}$$

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

$$\begin{aligned}
(\varepsilon_v^e)_{n+1} &= (\varepsilon_v^e)_{\text{trial}} - \Delta\gamma\left[\frac{\partial f}{\partial p}\right]_{n+1} \\
(\varepsilon_s^e)_{n+1} &= (\varepsilon_s^e)_{\text{trial}} - \Delta\gamma\left[\frac{\partial f}{\partial q}\right]_{n+1} \\
f_{n+1} &= 0\,.
\end{aligned} \tag{19.2}$$

19.3.2 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,

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

where

$$\frac{\partial p}{\partial \varepsilon_v^e} = -\frac{p_0\,\beta}{\tilde{\kappa}}\exp\left[-\frac{\varepsilon_v^e - \varepsilon_{v0}^e}{\tilde{\kappa}}\right] = \frac{p}{\tilde{\kappa}} \quad , \quad \frac{\partial p_c}{\partial \varepsilon_v^e} = \frac{(p_c)_n}{\tilde{\kappa} - \tilde{\lambda}}\exp\left[\frac{\varepsilon_v^e - (\varepsilon_v^e)_{\text{trial}}}{\tilde{\kappa} - \tilde{\lambda}}\right] \quad \text{and}$$

$$\frac{\partial p}{\partial \varepsilon_s^e} = \frac{3\,p_0\,\alpha\,\varepsilon_s^e}{\tilde{\kappa}}\exp\left[-\frac{\varepsilon_v^e - \varepsilon_{v0}^e}{\tilde{\kappa}}\right].$$

For the second row,

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

where

$$\frac{\partial q}{\partial \varepsilon_v^e} = -\frac{3p_0\,\alpha\,\varepsilon_s^e}{\tilde{\kappa}}\exp\left[-\frac{\varepsilon_v^e - \varepsilon_{v0}^e}{\tilde{\kappa}}\right] = \frac{\partial p}{\partial \varepsilon_s^e} \quad \text{and} \quad \frac{\partial q}{\partial \varepsilon_s^e} = 3\mu_0 + 3p_0\,\alpha\exp\left[-\frac{\varepsilon_v^e - \varepsilon_{v0}^e}{\tilde{\kappa}}\right] = 3\mu\,.$$

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.

19.3.3 Tangent calculation: elastic

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

$$\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\epsilon}} = \boldsymbol{1} \otimes \frac{\partial p}{\partial \boldsymbol{\epsilon}} + \sqrt{\tfrac{2}{3}}\, \boldsymbol{n} \otimes \frac{\partial q}{\partial \boldsymbol{\epsilon}} + \sqrt{\tfrac{2}{3}}\, q\, \frac{\partial \boldsymbol{n}}{\partial \boldsymbol{\epsilon}} \,. \tag{19.3}$$

For the first term above,

$$\frac{\partial p}{\partial \boldsymbol{\epsilon}} = p_{\text{o}}\, \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{v\text{o}}^e}{\tilde{\kappa}}\right] \frac{\partial \beta}{\partial \boldsymbol{\epsilon}} - p_{\text{o}}\, \frac{\beta}{\tilde{\kappa}}\, \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{v\text{o}}^e}{\tilde{\kappa}}\right] \frac{\partial \varepsilon_v^e}{\partial \boldsymbol{\epsilon}} = p_{\text{o}}\, \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{v\text{o}}^e}{\tilde{\kappa}}\right] \left(\frac{\partial \beta}{\partial \boldsymbol{\epsilon}} - \frac{\beta}{\tilde{\kappa}}\, \frac{\partial \varepsilon_v^e}{\partial \boldsymbol{\epsilon}}\right) .$$

Now,

$$\frac{\partial \beta}{\partial \boldsymbol{\epsilon}} = \frac{3\alpha}{\tilde{\kappa}}\, \varepsilon_s^e\, \frac{\partial \varepsilon_s^e}{\partial \boldsymbol{\epsilon}} \,.$$

Therefore,

$$\frac{\partial p}{\partial \boldsymbol{\epsilon}} = \frac{p_{\text{o}}}{\tilde{\kappa}}\, \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{v\text{o}}^e}{\tilde{\kappa}}\right] \left(3\alpha\, \varepsilon_s^e\, \frac{\partial \varepsilon_s^e}{\partial \boldsymbol{\epsilon}} - \beta\, \frac{\partial \varepsilon_v^e}{\partial \boldsymbol{\epsilon}}\right) .$$

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)

$$\frac{\partial \boldsymbol{\epsilon}^e}{\partial \boldsymbol{\epsilon}} = \frac{\partial \boldsymbol{\epsilon}}{\partial \boldsymbol{\epsilon}} = \mathbb{I}^{(s)} \quad \text{and} \quad \frac{\partial \varepsilon_v^e}{\partial \boldsymbol{\epsilon}} = \frac{\partial \varepsilon_v}{\partial \boldsymbol{\epsilon}} = \boldsymbol{1} \,.$$

That gives us

$$\frac{\partial \varepsilon_s^e}{\partial \boldsymbol{\epsilon}} = \sqrt{\tfrac{2}{3}}\, \frac{1}{\|\boldsymbol{e}^e\|} \left(\mathbb{I}^{(s)} - \tfrac{1}{3}\boldsymbol{1} \otimes \boldsymbol{1}\right) : \boldsymbol{e}^e = \sqrt{\tfrac{2}{3}}\, \frac{1}{\|\boldsymbol{e}^e\|} \left[\boldsymbol{e}^e - \tfrac{1}{3}\text{tr}(\boldsymbol{e}^e)\boldsymbol{1}\right] .$$

But $\text{tr}(\boldsymbol{e}^e) = 0$ 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

$$\frac{\partial p}{\partial \boldsymbol{\epsilon}} = \frac{p_{\text{o}}}{\tilde{\kappa}}\, \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{v\text{o}}^e}{\tilde{\kappa}}\right] \left(\sqrt{6}\, \alpha\, \varepsilon_s^e\, \boldsymbol{n} - \beta\, \boldsymbol{1}\right) . \tag{19.4}$$

The derivative of q with respect to $\boldsymbol{\epsilon}$ can be calculated in a similar way, i.e.,

$$\frac{\partial q}{\partial \boldsymbol{\epsilon}} = 3\mu\, \frac{\partial \varepsilon_s^e}{\partial \boldsymbol{\epsilon}} + 3\varepsilon_s^e\, \frac{\partial \mu}{\partial \boldsymbol{\epsilon}} = 3\mu\, \frac{\partial \varepsilon_s^e}{\partial \boldsymbol{\epsilon}} - 3\frac{p_{\text{o}}}{\tilde{\kappa}}\, \alpha\, \varepsilon_s^e\, \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{v\text{o}}^e}{\tilde{\kappa}}\right] \frac{\partial \varepsilon_v^e}{\partial \boldsymbol{\epsilon}} \,.$$

Using the expressions in the boxes above,

$$\frac{\partial q}{\partial \boldsymbol{\epsilon}} = \sqrt{6}\, \mu\, \boldsymbol{n} - 3\frac{p_{\text{o}}}{\tilde{\kappa}}\, \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{v\text{o}}^e}{\tilde{\kappa}}\right] \alpha\, \varepsilon_s^e\, \boldsymbol{1} \,. \tag{19.5}$$

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

$$\frac{\partial \boldsymbol{n}}{\partial \boldsymbol{\epsilon}} = \sqrt{\tfrac{2}{3}} \frac{1}{\varepsilon_s^e} \left[\mathbb{I}^{(s)} - \tfrac{1}{3} \boldsymbol{1} \otimes \boldsymbol{1} - \sqrt{\tfrac{2}{3}} \frac{1}{\varepsilon_s^e} \frac{\boldsymbol{e}^e \otimes \boldsymbol{e}^e}{\|\boldsymbol{e}^e\|} \right]$$

or

$$\frac{\partial \boldsymbol{n}}{\partial \boldsymbol{\epsilon}} = \sqrt{\tfrac{2}{3}} \frac{1}{\varepsilon_s^e} \left[\mathbb{I}^{(s)} - \tfrac{1}{3} \boldsymbol{1} \otimes \boldsymbol{1} - \boldsymbol{n} \otimes \boldsymbol{n} \right] . \tag{19.6}$$

Plugging the expressions for these derivatives in the original equation, we get

$$\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\epsilon}} = \frac{p_o}{\tilde{\kappa}} \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}} \right] \left(\sqrt{6}\, \alpha\, \varepsilon_s^e\, \boldsymbol{1} \otimes \boldsymbol{n} - \beta\, \boldsymbol{1} \otimes \boldsymbol{1} \right) + 2\mu\, \boldsymbol{n} \otimes \boldsymbol{n} - \sqrt{6} \frac{p_o}{\tilde{\kappa}} \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}} \right] \alpha\, \varepsilon_s^e\, \boldsymbol{n} \otimes \boldsymbol{1} + \tfrac{2}{3} \frac{q}{\varepsilon_s^e} \left[\mathbb{I}^{(s)} - \tfrac{1}{3} \boldsymbol{1} \otimes \boldsymbol{1} - \boldsymbol{n} \otimes \boldsymbol{n} \right] .$$

Reorganizing,

$$\boxed{\frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\epsilon}} = \frac{\sqrt{6}\, p_o\, \alpha\, \varepsilon_s^e}{\tilde{\kappa}} \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}} \right] (\boldsymbol{1} \otimes \mathbf{n} + \mathbf{n} \otimes \boldsymbol{1}) - \left(\frac{p_o \beta}{\tilde{\kappa}} \exp\left[-\frac{\varepsilon_v^e - \varepsilon_{vo}^e}{\tilde{\kappa}} \right] + \tfrac{2}{9} \frac{q}{\varepsilon_s^e} \right) \boldsymbol{1} \otimes \boldsymbol{1} + 2 \left(\mu - \tfrac{1}{3} \frac{q}{\varepsilon_s^e} \right) \boldsymbol{n} \otimes \boldsymbol{n} + \tfrac{2}{3} \frac{q}{\varepsilon_s^e} \mathbb{I}^{(s)} .} \tag{19.7}$$

19.3.4 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

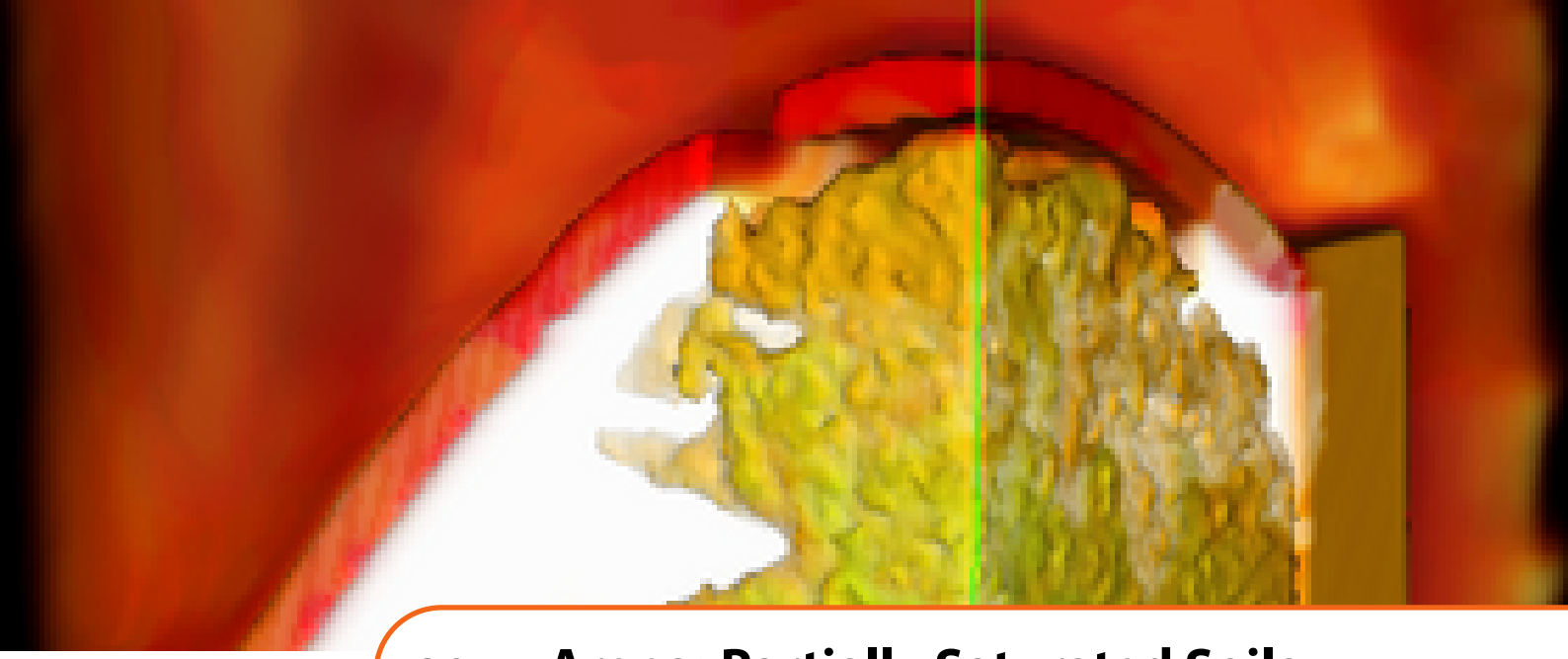
$$\boldsymbol{\epsilon}_{n+1}^e = \boldsymbol{\epsilon}_{\text{trial}}^e - \Delta\gamma \left[\frac{\partial f}{\partial \boldsymbol{\sigma}} \right]_{n+1} .$$

Dropping the subscript $n+1$ for convenience, we have

$$\frac{\partial \boldsymbol{\epsilon}^e}{\partial \boldsymbol{\epsilon}} = \frac{\partial \boldsymbol{\epsilon}_{\text{trial}}^e}{\partial \boldsymbol{\epsilon}} - \frac{\partial f}{\partial \boldsymbol{\sigma}} \otimes \frac{\partial \Delta\gamma}{\partial \boldsymbol{\epsilon}} - \Delta\gamma \frac{\partial}{\partial \boldsymbol{\epsilon}} \left[\frac{\partial f}{\partial \boldsymbol{\sigma}} \right] = \mathbb{I}^{(s)} - \left[\frac{2p - p_c}{3} \boldsymbol{1} + \sqrt{\tfrac{3}{2}} \frac{2q}{M^2} \boldsymbol{n} \right] \otimes \frac{\partial \Delta\gamma}{\partial \boldsymbol{\epsilon}} - \Delta\gamma \frac{\partial}{\partial \boldsymbol{\epsilon}} \left[\frac{2p - p_c}{3} \boldsymbol{1} + \sqrt{\tfrac{3}{2}} \frac{2q}{M^2} \boldsymbol{n} \right] .$$

19.4 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.



20 — 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,

$$\bar{x} := -x\,, \quad \dot{x} := \frac{\partial x}{\partial t}\,. \tag{20.1}$$

20.1 Elasticity

The elasticity model used by Arena has the form

$$\dot{\boldsymbol{\sigma}}^{\text{eff}} = \dot{\boldsymbol{\sigma}} - \dot{\boldsymbol{\alpha}} = \mathbb{C}^{\text{e}}(\boldsymbol{\sigma}, \boldsymbol{\varepsilon}^{\text{p}}, \phi, S_w) : \dot{\boldsymbol{\varepsilon}}^{\text{e}} - \dot{\lambda}\boldsymbol{Z} \tag{20.2}$$

where $\boldsymbol{\sigma}^{\text{eff}}$ is the effective stress, $\boldsymbol{\sigma}$ is the unrotated Cauchy stress, $\boldsymbol{\alpha}$ is the backstress, $\mathbb{C}^{\text{e}}$ 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

$$\mathbb{C}^{\text{e}} = \left(K - \tfrac{2}{3}G\right)\boldsymbol{I}\otimes\boldsymbol{I} + 2G\,\mathbb{I} \tag{20.3}$$

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 $\mathbb{I}$ is the symmetric part of the rank-4 identity tensor.

If the effective stress is decomposed into volumetric and deviatoric parts:

$$\boldsymbol{\sigma}^{\text{eff}} = -\bar{p}\,\boldsymbol{I} + \mathbf{s}\,, \quad p := \tfrac{1}{3}\text{tr}(\boldsymbol{\sigma}^{\text{eff}})\,, \quad \mathbf{s} := \boldsymbol{\sigma}^{\text{eff}} - \tfrac{1}{3}\text{tr}(\boldsymbol{\sigma}^{\text{eff}})\boldsymbol{I} \tag{20.4}$$

and the elastic strain is also decomposed into volumetric and deviatoric parts

$$\boldsymbol{\varepsilon}^{\text{e}} = -\tfrac{1}{3}\bar{\varepsilon}^{\text{e}}_v\,\boldsymbol{I} + \boldsymbol{\gamma}^{\text{e}}\,, \quad \varepsilon^{\text{e}}_v := \text{tr}(\boldsymbol{\varepsilon}^{\text{e}})\,, \quad \boldsymbol{\gamma}^{\text{e}} := \boldsymbol{\varepsilon}^{\text{e}} - \tfrac{1}{3}\text{tr}(\boldsymbol{\varepsilon}^{\text{e}})\boldsymbol{I} \tag{20.5}$$

the elasticity model (without the coupling term), simplifies to

$$\dot{\bar{p}} = K(\boldsymbol{\sigma}, \alpha, \boldsymbol{\varepsilon}^{\text{p}}, \phi, S_w)\dot{\bar{\varepsilon}}^{\text{e}}_v\,, \quad \dot{\mathbf{s}} = 2G(\boldsymbol{\sigma}, \alpha, \boldsymbol{\varepsilon}^{\text{p}}, \phi, S_w)\,\dot{\boldsymbol{\gamma}}^{\text{e}}\,. \tag{20.6}$$

The partially saturated Arena model assumes the moduli depend only on

$$I_1 := \text{tr}(\boldsymbol{\sigma})\,, \quad \zeta = \text{tr}(\boldsymbol{\alpha})\,, \quad \varepsilon^{\text{p}}_v := \text{tr}(\boldsymbol{\varepsilon}^{\text{p}})\,, \quad \phi\,, \quad S_w\,. \tag{20.7}$$

20.1.1 Bulk modulus model: Solid matrix material

The pressure in the solid matrix is expressed as

$$\bar{p}_s = K_s \overline{\varepsilon_v^s} \; ; \quad \overline{\varepsilon_v^s} := \ln\left(\frac{V_{so}}{V_s}\right) \tag{20.8}$$

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:

$$K_s(\bar{p}_s) = K_{so} + n_s\,(\bar{p}_s - \bar{p}_{so}) \tag{20.9}$$

where K_{so} and n_s are material properties, and $\bar{p}_{so}$ is a reference pressure.

20.1.2 Bulk modulus model: Pore water

The equation of state of the pore water is

$$\bar{p}_w = K_w \overline{\varepsilon_v^w} + \bar{p}_o \; ; \quad \overline{\varepsilon_v^w} := \ln\left(\frac{V_{wo}}{V_w}\right) \tag{20.10}$$

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:

$$K_w(\bar{p}_w) = K_{wo} + n_w\,(\bar{p}_w - \bar{p}_{wo}) \tag{20.11}$$

where K_{wo} and n_w are material properties, and $\bar{p}_{wo}$ is a reference pressure.

20.1.3 Bulk modulus model: Pore air

The isentropic ideal gas equation of state for the pore air is

$$\bar{p}_a = \bar{p}_r\left[\exp(\gamma\,\overline{\varepsilon_v^a}) - 1\right] \; ; \quad \overline{\varepsilon_v^a} := \ln\left(\frac{V_{ao}}{V_a}\right) \tag{20.12}$$

where the quantities with subscript a represent quantities for the air model analogous to those for the water model in (20.10), $\bar{p}_r$ 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:

$$K_a = \frac{d\bar{p}_a}{d\overline{\varepsilon_v^a}} = \gamma\,\bar{p}_r\,\exp(\gamma\,\overline{\varepsilon_v^a}) = \gamma\,(\bar{p}_a + \bar{p}_r)\,. \tag{20.13}$$

20.1.4 Bulk modulus model: Drained soil

The pressure model for drained soils has the form

$$\frac{\bar{p}^{\text{eff}}}{K_s(\bar{p}^{\text{eff}})} = b_0\,\overline{\varepsilon_v^e} + \frac{b_1(\overline{\varepsilon_v^e})^{b_4}}{b_2(\overline{\varepsilon_v^e})^{b_4} + b_3} \tag{20.14}$$

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

$$K_d(\bar{p}^{\text{eff}}) := \frac{d\bar{p}^{\text{eff}}}{d\overline{\varepsilon_v^e}}\,. \tag{20.15}$$

Then, using (20.14),

$$K_d(\bar{p}^{\text{eff}}) = \frac{[K_s(\bar{p}^{\text{eff}})]^2}{[K_s(\bar{p}^{\text{eff}}) - n_s\bar{p}^{\text{eff}}]}\left[b_0 + \frac{b_1 b_3 b_4 (\bar{\varepsilon}_v^e)^{b_4-1}}{\left[b_2(\bar{\varepsilon}_v^e)^{b_4} + b_3\right]^2}\right]. \tag{20.16}$$

To express (20.16) 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 (20.14) by assuming $b_0 \to 0$. This approximation is valid at moderate to large strains. Then, from (20.14) with $b_0 = 0$, we have

$$\bar{\varepsilon}_v^e \approx \left[\frac{b_3\bar{p}^{\text{eff}}}{b_1 K_s(\bar{p}^{\text{eff}}) - b_2\bar{p}^{\text{eff}}}\right]^{1/b_4} \tag{20.17}$$

and (20.16) can be expressed in terms of $\bar{p}$ as

$$K_d(\bar{p}^{\text{eff}}) = \frac{[K_s(\bar{p}^{\text{eff}})]^2}{[K_s(\bar{p}^{\text{eff}}) - n_s\bar{p}^{\text{eff}}]}\left[b_0 + \frac{b_1 b_3 b_4 \left(\frac{b_3\bar{p}^{\text{eff}}}{b_1 K_s(\bar{p}^{\text{eff}}) - b_2\bar{p}^{\text{eff}}}\right)^{1-1/b_4}}{\left[b_2\left(\frac{b_3\bar{p}^{\text{eff}}}{b_1 K_s(\bar{p}^{\text{eff}}) - b_2\bar{p}^{\text{eff}}}\right) + b_3\right]^2}\right]. \tag{20.18}$$

20.1.5 Bulk modulus model: Partially saturated soil

The pressure in the partially saturated soil ($\bar{p}$) is given by

$$\bar{p} = \int K(\bar{I}_1, \bar{\zeta}, \bar{\varepsilon}_v^p, \phi, S_w)\, \mathrm{d}\bar{\varepsilon}_v^e. \tag{20.19}$$

Note that

$$\bar{p}^{\text{eff}} = \tfrac{1}{3}(\bar{I}_1 - \bar{\zeta}). \tag{20.20}$$

The tangent bulk modulus of the partially saturated soil is found using a variation on the Grassman model for fully saturated rocks:

$$K(\bar{p}^{\text{eff}}, \bar{\varepsilon}_v^p, \phi, S_w) = K_d(\bar{p}^{\text{eff}}) + \frac{\left(1 - \frac{K_d(\bar{p}^{\text{eff}})}{K_s(\bar{p}^{\text{eff}})}\right)^2}{\frac{1}{K_s(\bar{p}^{\text{eff}})}\left(1 - \frac{K_d(\bar{p}^{\text{eff}})}{K_s(\bar{p}^{\text{eff}})}\right) + \phi\left(\frac{1}{K_f(\bar{\zeta})} - \frac{1}{K_s(\bar{p}^{\text{eff}})}\right)} \tag{20.21}$$

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):

$$\frac{1}{K_f(\bar{\zeta})} = \frac{S_w}{K_w(\bar{\zeta})} + \frac{1 - S_w}{K_a(\bar{\zeta})}. \tag{20.22}$$

20.1.6 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

$$\nu = \nu_1 + \nu_2 \exp\left[-\frac{K_d(\bar{p}^{\text{eff}}, \bar{\varepsilon}_v^p, \phi, S_w)}{K_s(\bar{p}^{\text{eff}})}\right] \tag{20.23}$$

where ν_1 and ν_2 are material parameters. The shear modulus is computed using the Poisson's ratio and the drained bulk modulus:

$$G(\bar{p}^{\text{eff}}, \bar{\varepsilon}_v^p, \phi, S_w) = \frac{3K_d(\bar{p}^{\text{eff}}, \bar{\varepsilon}_v^p, \phi, S_w)(1 - 2\nu)}{2(1 + \nu)}. \tag{20.24}$$

20.2 Rate-independent plasticity

20.2.1 Yield function

The Arena yield function is

$$f = \sqrt{J_2} - F_f(\bar{I}_1, \zeta)\, F_c(\bar{I}_1, \bar{\zeta}, \overline{X}, \bar{\kappa}) = \sqrt{J_2} - F_f(\bar{p}^{\text{eff}})\, F_c(\bar{p}^{\text{eff}}, \overline{X}, \bar{\kappa}) \tag{20.25}$$

where

$$F_f(\bar{p}^{\text{eff}}) = a_1 - a_3 \exp[-3a_2\bar{p}^{\text{eff}})] + 3a_4\bar{p}^{\text{eff}} \tag{20.26}$$

and

$$F_c(\bar{p}^{\text{eff}}, \overline{X}, \bar{\kappa}) = \begin{cases} 1 & \text{for} \quad 3\bar{p}^{\text{eff}} \le \bar{\kappa} \\ \sqrt{1 - \left(\dfrac{3\bar{p}^{\text{eff}} - \bar{\kappa}}{\overline{X} - \bar{\kappa}}\right)^2} & \text{for} \quad 3\bar{p}^{\text{eff}} > \bar{\kappa}\,. \end{cases} \tag{20.27}$$

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

$$J_2 = \tfrac{1}{2}\mathbf{s} : \mathbf{s}\,. \tag{20.28}$$

Non-associativity is modeled using a parameter β that modifies $\sqrt{J_2}$ (see 20.8).

20.2.2 Hydrostatic compressive strength: Drained soil

The drained crush curve model is used to compute $\overline{X}$ and has the form

$$\overline{\varepsilon_v^p} - p_3 = \ln\left[1 - \frac{1 - \exp(-p_3)}{1 + \left(\frac{\overline{X}_d - p_0}{p_1}\right)^{p_2}}\right]. \tag{20.29}$$

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

$$\overline{X}_d(\overline{\varepsilon_v^p}, \phi_0) - p_0 = p_1\left[\frac{1 - \exp(-p_3)}{1 - \exp(-p_3 + \overline{\varepsilon_v^p})} - 1\right]^{1/p_2}, \quad p_3 := -\ln(1 - \phi_0)\,. \tag{20.30}$$

20.2.3 Hydrostatic compressive strength: Partially saturated soil

The elastic part of the volumetric strain at yield is defined in the model as

$$\varepsilon_v^{e,\text{yield}}(\overline{\varepsilon_v^p}) = \frac{\overline{X}_d(\overline{\varepsilon_v^p}, \phi_0)}{3\,K_d\left(\frac{1}{2}\frac{\overline{X}_d(\overline{\varepsilon_v^p}, \phi_0)}{3}\right)} \tag{20.31}$$

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

$$\overline{X}(\overline{\varepsilon_v^p}) = 3K(\bar{p}^{\text{eff}}, \overline{\varepsilon_v^p}, \phi, S_w)\, \bar{\varepsilon}_v^{e,\text{yield}}(\overline{\varepsilon_v^p}) \tag{20.32}$$

where K is the bulk modulus of the partially saturated material.

20.2.4 Backstress: Pore pressure

The pore pressure as an isotropic backstress (ζ) that translates the Cauchy stress to the effective stress:

$$\boldsymbol{\sigma}^{\text{eff}} = \boldsymbol{\sigma} - \zeta \boldsymbol{I}\,, \quad \zeta := -[(1-S_w)\bar{p}_a + S_w \bar{p}_w]\,. \tag{20.33}$$

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

$$\exp(-\overline{\varepsilon_v^p}) = \phi_o\,(1-S_o)\exp\left[-\frac{1}{\gamma}\ln\left(\frac{\bar{\zeta}}{\bar{p}_r}+1\right)\right] + \phi_o\,S_o\,\exp\left(-\frac{\bar{\zeta}-\bar{p}_o}{K_w}\right) + (1-\phi_o)\exp\left(-\frac{\bar{\zeta}}{K_s}\right)\,. \tag{20.34}$$

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:

$$\zeta = \int \frac{d\zeta}{d\varepsilon_v^p}\mathrm{d}\varepsilon_v^p\,. \tag{20.35}$$

where

$$\frac{d\zeta}{d\varepsilon_v^p} = \frac{\exp(-\overline{\varepsilon_v^p})}{\mathcal{B}}\,, \tag{20.36}$$

and

$$\mathcal{B} := \left[\frac{\phi_o\,(1-S_o)}{\gamma(\bar{p}_r+\bar{\zeta})}\right]\exp\left[-\frac{1}{\gamma}\ln\left(\frac{\bar{\zeta}}{\bar{p}_r}+1\right)\right] + \frac{\phi_o\,S_o}{K_w}\exp\left(\frac{\bar{p}_o-\bar{\zeta}}{K_w}\right) + \frac{1-\phi_o}{K_s}\exp\left(-\frac{\bar{\zeta}}{K_s}\right)\,. \tag{20.37}$$

20.3 Rate-dependent plasticity

20.4 Porosity and saturation

The total volumetric strain is given by

$$\exp(\varepsilon_v) = (1-S_o)\phi_o\exp(\varepsilon_v^a) + S_o\phi_o\exp(\varepsilon_v^w) + (1-\phi_o)\exp(\varepsilon_v^s) \tag{20.38}$$

where ϕ_o, S_o are the initial porosity and saturation, and

$$\varepsilon_v^w(\varepsilon_v) = -\frac{\bar{p}(\varepsilon_v)-\bar{p}_o}{K_w}\,, \quad \varepsilon_v^a(\varepsilon_v) = -\frac{1}{\gamma}\ln\left[1+\frac{\bar{p}(\varepsilon_v)}{\bar{p}_r}\right]\,, \quad \varepsilon_v^s(\varepsilon_v) = -\frac{\bar{p}(\varepsilon_v)}{K_s}\,. \tag{20.39}$$

We can combine (20.38) and (20.39) to solve for $\bar{p}(\varepsilon_v)$ and then compute the volumetric strain in the air in terms of the total volumetric strain.

20.4.1 Saturation

The saturation function $S_w(\varepsilon_v)$, is given by

$$S_w(\varepsilon_v) = \frac{\mathcal{C}(\varepsilon_v)}{1+\mathcal{C}(\varepsilon_v)}\,, \quad \mathcal{C}(\varepsilon_v) := \left(\frac{S_o}{1-S_o}\right)\exp(\varepsilon_v^w)\exp(-\varepsilon_v^a)\,. \tag{20.40}$$

20.4.2 Porosity

The porosity evolution equation (in the elastically unloaded state) for partially saturated sand has the form

$$\phi(\varepsilon_v) = \phi_o\left(\frac{1-S_o}{1-S_w(\varepsilon_v)}\right)\left[\frac{\exp(\varepsilon_v^a)}{\exp(\varepsilon_v)}\right]\,. \tag{20.41}$$

20.5 Summary of partially saturated soil model

Summary 20.5.1 ***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

$$K_s(\bar{p}) = K_{s0} + n_s(\bar{p} - \bar{p}_{s0}), \quad K_w(\bar{p}) = K_{w0} + n_w(\bar{p} - \bar{p}_{w0}), \quad K_a(\bar{p}) = \gamma(\bar{p} + \bar{p}_r)$$

Summary 20.5.2 ***Shear modulus model***

The shear modulus is either a constant (G_0) 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 20.5.3 ***Yield function***

The Arena yield function is

$$f = \sqrt{J_2} - F_f(\bar{I}_1, \zeta)\, F_c(\bar{I}_1, \bar{\zeta}, \bar{X}, \bar{\kappa}) = \sqrt{J_2} - F_f(\bar{p}^{\text{eff}})\, F_c(\bar{p}^{\text{eff}}, \bar{X}, \bar{\kappa}) \tag{20.42}$$

where

$$F_f(\bar{p}^{\text{eff}}) = a_1 - a_3 \exp[-3a_2 \bar{p}^{\text{eff}})] + 3a_4 \bar{p}^{\text{eff}} \tag{20.43}$$

and

$$F_c(\bar{p}^{\text{eff}}, \bar{X}, \bar{\kappa}) = \begin{cases} 1 & \text{for} \quad 3\bar{p}^{\text{eff}} \le \bar{\kappa} \\ \sqrt{1 - \left(\dfrac{3\bar{p}^{\text{eff}} - \bar{\kappa}}{\bar{X} - \bar{\kappa}} \right)^2} & \text{for} \quad 3\bar{p}^{\text{eff}} > \bar{\kappa}\,. \end{cases} \tag{20.44}$$

Non-associativity is modeled using a parameter β that modifies $\sqrt{J_2}$.

Summary 20.5.4 ***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 20.5.5 ***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 20.5.6 ***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:

$$\phi(\varepsilon_v) = \phi_0\left(\frac{1-S_0}{1-S_w(\varepsilon_v)}\right)\left[\frac{\exp(\varepsilon_v^a)}{\exp(\varepsilon_v)}\right]. \tag{20.45}$$

Note that

$$\exp(\varepsilon_v) = (1-S_0)\phi_0\exp(\varepsilon_v^a) + S_0\phi_0\exp(\varepsilon_v^w) + (1-\phi_0)\exp(\varepsilon_v^s)$$

20.6 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

$$\boldsymbol{\sigma}^{\mathrm{trial}} = \boldsymbol{\sigma}^{\mathrm{n}} + \mathbb{C}^{\mathrm{e}} : (\mathbf{d}\,\Delta t) \tag{20.46}$$

where $\mathbb{C}^{\mathrm{e}}$ 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 $\mathbb{C}^e$ 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
```

20.7 The consistency bisection algorithm

20.7.1 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

$$\Delta\boldsymbol{\sigma}^{\text{o}} = \boldsymbol{\sigma}^{\text{new,o}} - \boldsymbol{\sigma}^{\text{old}} = \mathbb{C} : \Delta\boldsymbol{\varepsilon}^{\text{e,o}} \tag{20.47}$$

where $\mathbb{C}$ 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 $\mathbb{C}$, we can compute the elastic strain increment using

$$\Delta\boldsymbol{\varepsilon}^{\text{e,o}} = \mathbb{C}^{-1} : \Delta\boldsymbol{\sigma}^{\text{o}} \,. \tag{20.48}$$

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

$$\Delta\boldsymbol{\varepsilon}^{\mathrm{p,o}} = \Delta\boldsymbol{\varepsilon} - \Delta\boldsymbol{\varepsilon}^{\mathrm{e,o}} \,. \tag{20.49}$$

20.7.2 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

$$\Delta\boldsymbol{\varepsilon}^{\mathrm{p}} > \Delta\boldsymbol{\varepsilon}^{\mathrm{p,o}} \tag{20.50}$$

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

$$\Delta\varepsilon_v^{\mathrm{p}} = \mathrm{tr}(\Delta\boldsymbol{\varepsilon}^{\mathrm{p}}) \,. \tag{20.51}$$

Because

$$\Delta\varepsilon_v^{\mathrm{p}} > \Delta\varepsilon_v^{\mathrm{p,o}} \tag{20.52}$$

we can define a parameter, $\eta \in (0,1)$, such that

$$\eta := \frac{\Delta\varepsilon_v^{\mathrm{p}}}{\Delta\varepsilon_v^{\mathrm{p,o}}} \,. \tag{20.53}$$

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

20.7.3 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 $\text{abs}(\|\delta\boldsymbol{\varepsilon}^{\text{p,new}}\|_2 - \eta^{\text{mid}}\,\|\delta\boldsymbol{\varepsilon}^{\text{p,o}}\|_2) < \text{TOLERANCE}$

$\boldsymbol{\varepsilon}^{\text{p,new}} = \boldsymbol{\varepsilon}^{\text{p,old}} + \delta\boldsymbol{\varepsilon}^{\text{p,new}}$ ▷*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

20.8 The nonhardening return algorithm

Let the plastic flow direction be $\boldsymbol{M}$. Then

$$\dot{\boldsymbol{\varepsilon}}^p = \dot{\lambda}\boldsymbol{M}\,. \tag{20.54}$$

The nonhardening return algorithm uses a transformed space where the computation is carried out in special Lode coordinates (z', r') where

$$z' = z - \frac{\zeta}{\sqrt{3}}, \quad z := \frac{I_1}{\sqrt{3}} \quad \text{and} \quad r' = \sqrt{\frac{3K}{2G}}\, r, \quad r := \sqrt{2J_2}. \tag{20.55}$$

If the flow rule is non-associative, the yield surface parameter $\beta \neq 1$. In that case,

$$r' \leftarrow \beta r'. \tag{20.56}$$

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*
  $\delta\boldsymbol{\varepsilon}^{\text{p,new}} = \delta\boldsymbol{\varepsilon} - \mathbb{C}^{-1} : (\boldsymbol{\sigma}^{\text{new}} - \boldsymbol{\sigma}^{\text{old}})$ ▷*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.

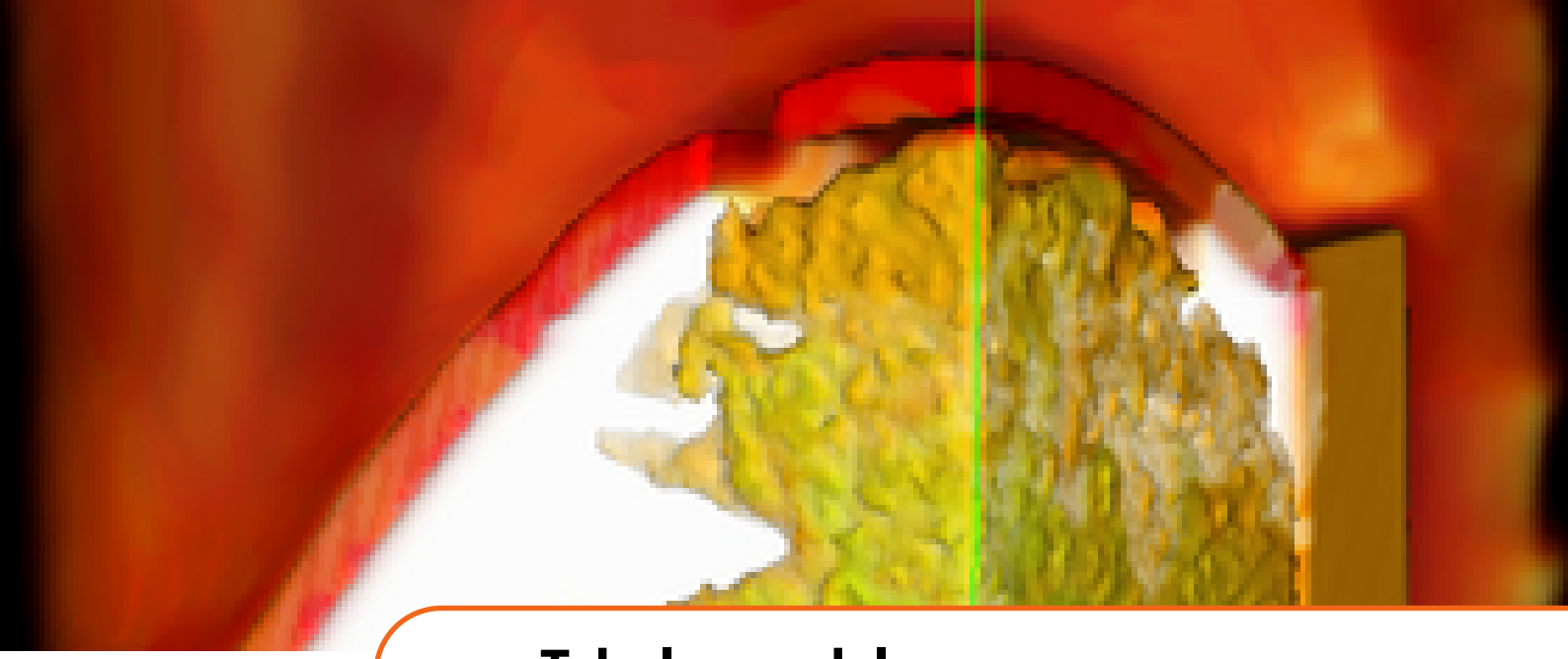
procedure APPLYBISECTIONALGORITHM(z^{o}, $(r')^{\text{o}}$, z^{trial}, $(r')^{\text{trial}}$, X^{old}, ζ^{old}, K^{old}, G^{old}, β)
  $\eta^{\text{in}} \leftarrow 0$, $\eta^{\text{out}} \leftarrow 1$
  while $\eta^{\text{out}} - \eta^{\text{in}} \geq \text{TOL}$ **do**
    $\eta^{\text{mid}} = \frac{1}{2}(\eta^{\text{in}} + \eta^{\text{out}})$
    $\begin{bmatrix} z^{\text{mid}} \\ (r')^{\text{mid}} \end{bmatrix} \leftarrow \eta^{\text{mid}} \begin{bmatrix} z^{\text{trial}} - z^{\text{o}} \\ (r')^{\text{trial}} - (r')^{\text{o}} \end{bmatrix} + \begin{bmatrix} z^{\text{o}} \\ (r')^{\text{o}} \end{bmatrix}$
    isElastic $\leftarrow$ 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 $\leftarrow$ 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 $\leftarrow$ EVALYIELDCONDITION(I_1^{new}, $\sqrt{J_2^{\text{new}}}$, X^{old}, ζ^{old}, K^{old}, G^{old}, β)
  return isElastic
end procedure



21 — Tabular models

At present we allow only three independent variables in VAANGO .

MPM tabular material data is often of the form shown in Figure 21.1. 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.

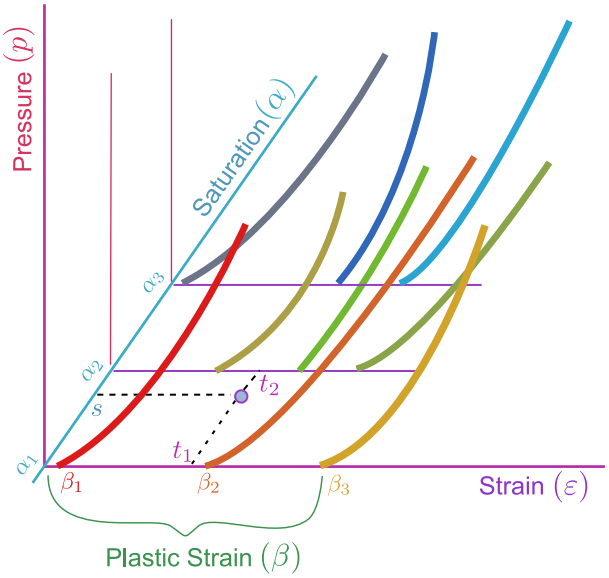


Figure 21.1: Schematic of tabular material data for MPM constitutive models. The circle in blue is the input data point for which we would like to find the pressure.

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.

21.1 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 21.2.

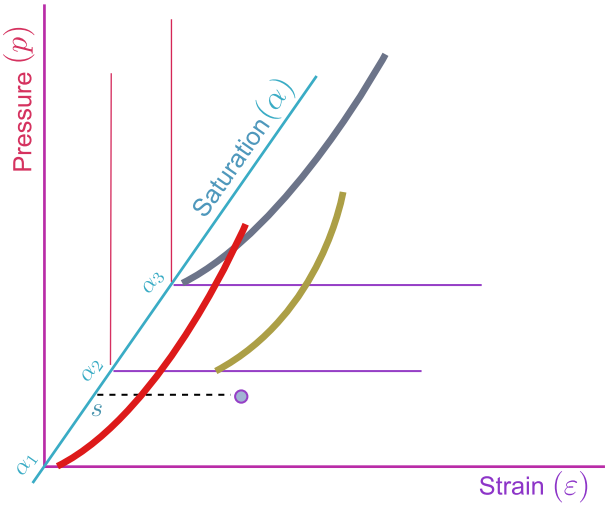


Figure 21.2: Schematic of a three variable table of material data. The circle in blue is the input point for which we would like to find the pressure.

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

$$s = \frac{\alpha_0 - \alpha_k}{\alpha_{k+1} - \alpha_k}, \quad k = 1, 2, \ldots, N-1 \tag{21.1}$$

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 21.3. The two associated parameters t_1 and t_2 are calculated using

$$\begin{aligned} t_1 &= \frac{\varepsilon_0 - \varepsilon_{j,k}}{\varepsilon_{j,k+1} - \varepsilon_{j,k}}, \quad k = 1, 2, \ldots, M_j - 1 \\ t_2 &= \frac{\varepsilon_0 - \varepsilon_{j+1,k}}{\varepsilon_{j+1,k+1} - \varepsilon_{j+1,k}}, \quad k = 1, 2, \ldots, M_{j+1} - 1 \end{aligned} \tag{21.2}$$

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

$$\begin{aligned} p_1 &= (1 - t_1) p_{j,k} + t_1 p_{j,k+1} \\ p_2 &= (1 - t_2) p_{j+1,k} + t_2 p_{j+1,k+1} \end{aligned} \tag{21.3}$$

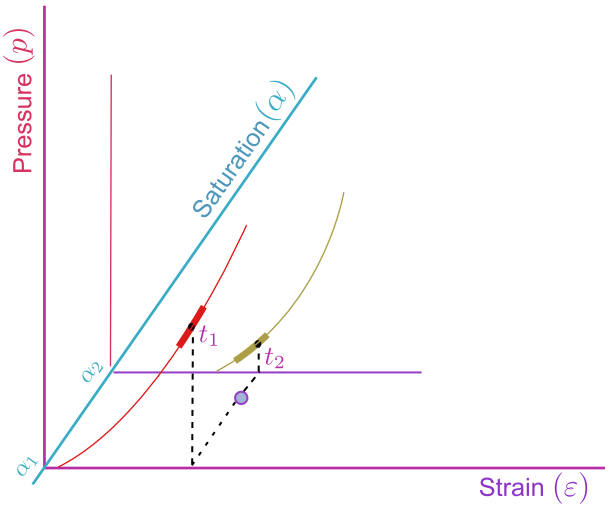


Figure 21.3: Second stage of interpolation of a three variable table of material data. The circle in blue is the input point for which we would like to find the pressure.

The final step of the process is to compute the interpolated pressure p_o using

$$p_o = (1 - s)p_1 + sp_2 \,. \tag{21.4}$$

A schematic of this operation is shown in Figure 21.4.

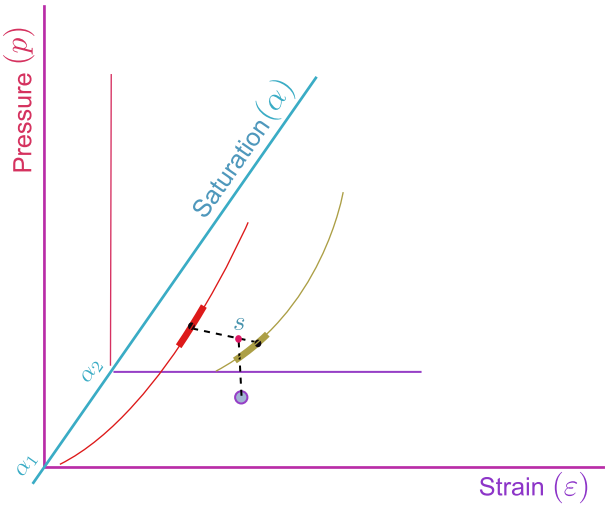


Figure 21.4: Final stage of interpolation of a three variable table of material data. The circle in blue is the input point and the red circle is the interpolated value.

21.2 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

$$K = \rho \left[\frac{p(\eta + \epsilon) - p(\eta - \epsilon)}{2\epsilon} \right] \tag{21.5}$$

The tolerance ϵ is hardcoded to 10^{-6} in VAANGO but may not be adequate for some problems.

21.3 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, $K(p)$:

$$G = \frac{3K(1-2\nu)}{2(1+\nu)} \tag{21.6}$$

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 21.5 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:

$$K(\bar{\varepsilon}_{v0}^p, \bar{\varepsilon}_{v0}^e) = \frac{p(\bar{\varepsilon}_{v0}^p, \bar{\varepsilon}_{v0}^e + \epsilon) - p(\bar{\varepsilon}_{v0}^p, \bar{\varepsilon}_{v0}^e - \epsilon)}{2\epsilon} \tag{21.7}$$

The tolerance ϵ is hardcoded to 10^{-6} in VAANGO and may not be adequate for some problems.

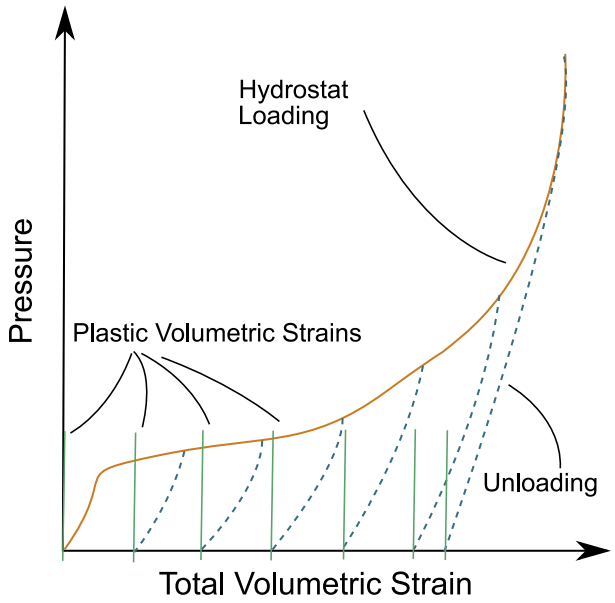
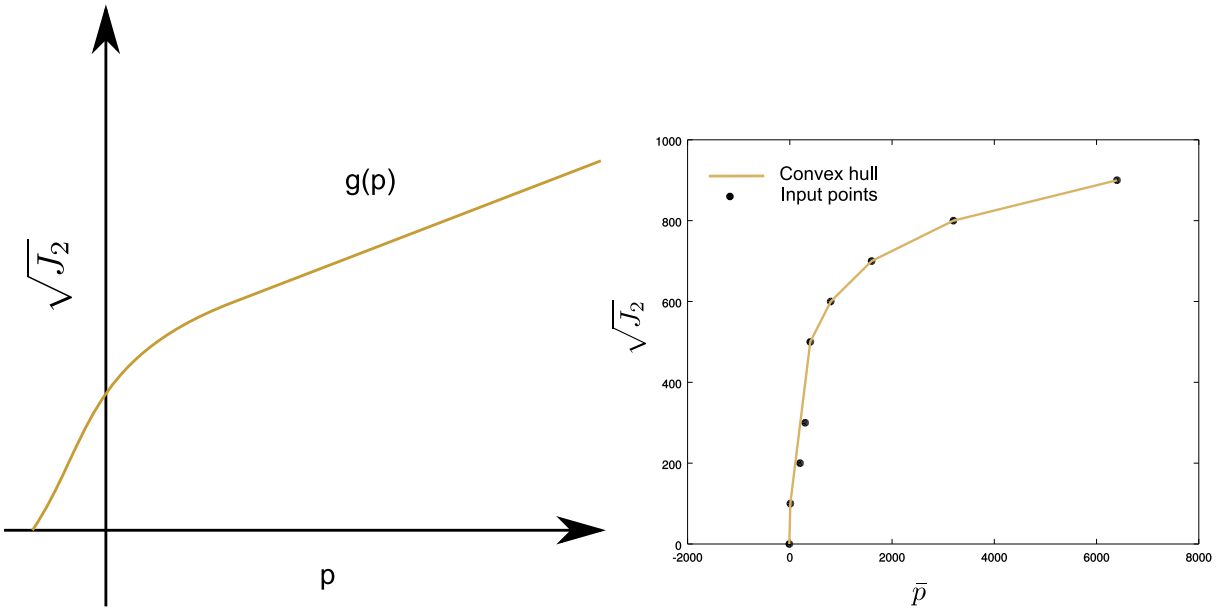


Figure 21.5: Unloading curves uses to determine the tangent bulk modulus for the tabular plasticity model at various plastic strain value.

The tabular yield condition has the form

$$f = \sqrt{J_2} - g(\bar{p}) = 0 \tag{21.8}$$

The function $g(\bar{p})$ is provided in tabular form and is depicted in Figure 21.6(a). To ensure convexity of the tabular data, a convex hull of the data points is computed first as shown in Figure 21.6(b).



(a) Yield function

(b) Computed hull of the input data

Figure 21.6: 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

$$s_x = \mathbf{a} \cdot (\boldsymbol{M}_j \cdot \mathbf{p}_x) \,, \quad s_y = \mathbf{a} \cdot (\boldsymbol{M}_j \cdot \mathbf{p}_y) \tag{21.9}$$

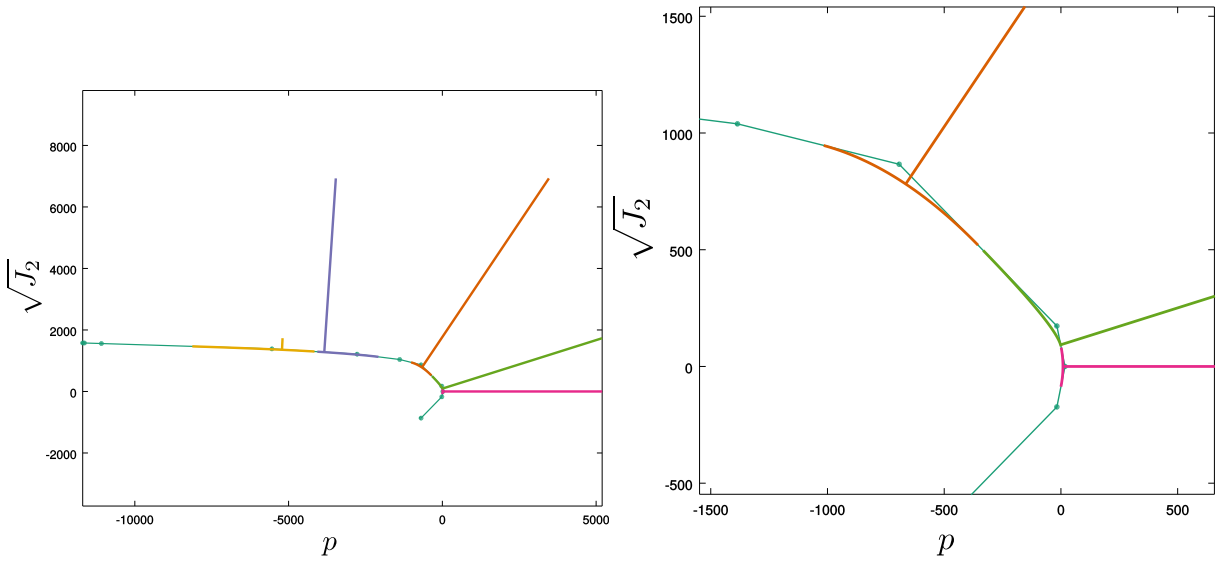
where $\mathbf{a} = (1, t, t^2)$, $t \in [0, 1]$ parameterizes each segment of the tabular data, $\mathbf{p}_x = (x_k, x_{k+1}, x_{k+2})$, $\mathbf{p}_y = (y_k, y_{k+1}, y_{k+2})$, and (x_k, y_k) are the input $0, \ldots, N-1$ tabular data points. The associated matrices that are used are:

$$\boldsymbol{M}_{j=0} = 0.5 \begin{bmatrix} 2 & 0 & 0 \\ -4 & 4 & 0 \\ 2 & -3 & 1 \end{bmatrix} \,, \quad \boldsymbol{M}_j = 0.5 \begin{bmatrix} 1 & 1 & 0 \\ -2 & 2 & 0 \\ 1 & -3 & 2 \end{bmatrix} \,, \quad \boldsymbol{M}_{j=N-1} = 0.5 \begin{bmatrix} 1 & 1 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{bmatrix} \tag{21.10}$$

Closest point projections of stress states outside the yield surface to fitted B-splines along the yield surface are shown in Figure 21.7.

21.4 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 21.7: Closest point projections to yield function used by the tabular plasticity.

21.4.1 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

$$\begin{aligned} &\text{minimize} && f(\mathbf{x}) \\ &\text{subject to} && g_i(\mathbf{x}) \le 0, \quad i = 1, \ldots, m \\ &&& h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, p \end{aligned} \tag{21.11}$$

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

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f(\mathbf{x}) + \boldsymbol{\lambda} \cdot \mathbf{g}(\mathbf{x}) + \boldsymbol{\nu} \cdot \mathbf{h}(\mathbf{x}) \tag{21.12}$$

where

$$\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{bmatrix}, \quad \mathbf{g} = \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_m \end{bmatrix}, \quad \boldsymbol{\nu} = \begin{bmatrix} \nu_1 \\ \nu_2 \\ \vdots \\ \nu_p \end{bmatrix}, \quad \mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_p \end{bmatrix}. \tag{21.13}$$

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

$$\mathcal{F}(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\mathbf{x}} \left[f(\mathbf{x}) + \boldsymbol{\lambda} \cdot \mathbf{g}(\mathbf{x}) + \boldsymbol{\nu} \cdot \mathbf{h}(\mathbf{x}) \right] \tag{21.14}$$

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

$$\begin{aligned} &\text{maximize} \quad \mathcal{F}(\boldsymbol{\lambda}, \boldsymbol{\nu}) \\ &\text{subject to} \quad \boldsymbol{\lambda} \succeq \mathbf{o} \end{aligned} \tag{21.15}$$

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.,

$$f(\mathbf{x}^\star) = \mathcal{F}(\boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) \tag{21.16}$$

the Lagrangian at that optimal point is

$$\mathcal{L}(\mathbf{x}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) = f(\mathbf{x}^\star) + \boldsymbol{\lambda}^\star \cdot \mathbf{g}(\mathbf{x}^\star) + \boldsymbol{\nu}^\star \cdot \mathbf{h}(\mathbf{x}^\star) \tag{21.17}$$

Also, since $\boldsymbol{\lambda}^\star \succeq \mathrm{o}$ and $\mathbf{h} = \mathrm{o}$,

$$f(\mathbf{x}^\star) = \mathcal{F}(\boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) = \inf_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) \le \mathcal{L}(\mathbf{x}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) \le f(\mathbf{x}^\star) \tag{21.18}$$

The only way for the above to be true is when

$$\boldsymbol{\lambda}^\star \cdot \mathbf{g}(\mathbf{x}^\star) = \mathrm{o} \quad \leftrightarrow \quad \lambda_i^\star g_i(\mathbf{x}^\star) = \mathrm{o}\,. \tag{21.19}$$

Also, since $\mathbf{x}^\star$ minimizes the Lagrangian, its gradient is zero at that point:

$$\frac{\partial}{\partial \mathbf{x}} \mathcal{L}(\mathbf{x}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\nu}^\star) = \mathbf{o} = \frac{\partial f(\mathbf{x}^\star)}{\partial \mathbf{x}} + \boldsymbol{\lambda}^\star \cdot \frac{\partial \mathbf{g}(\mathbf{x}^\star)}{\partial \mathbf{x}} + \boldsymbol{\nu}^\star \cdot \frac{\partial \mathbf{h}(\mathbf{x}^\star)}{\partial \mathbf{x}} \tag{21.20}$$

These results, along with the original constraints of the primal and dual problems, are collected together into the *Karush-Kuhn-Tucker optimality conditions*:

$$\begin{aligned} &g_i(\mathbf{x}^\star) \le \mathrm{o} && h_j(\mathbf{x}^\star) = \mathrm{o} \\ &\lambda_i^\star \ge \mathrm{o} && \lambda_i^\star g_i(\mathbf{x}^\star) = \mathrm{o} \\ &\frac{\partial f(\mathbf{x}^\star)}{\partial \mathbf{x}} + \boldsymbol{\lambda}^\star \cdot \frac{\partial \mathbf{g}(\mathbf{x}^\star)}{\partial \mathbf{x}} + \boldsymbol{\nu}^\star \cdot \frac{\partial \mathbf{h}(\mathbf{x}^\star)}{\partial \mathbf{x}} = \mathbf{o} \end{aligned} \tag{21.21}$$

21.4.2 Similarity with plasticity

The plastic loading-unloading conditions are similar to the Karush-Kush-Tucker optimality conditions in that we have

$$g(\boldsymbol{\sigma}) \le 0\,, \quad \dot{\lambda} \ge 0\,, \quad \dot{\lambda} g(\boldsymbol{\sigma}) = 0 \tag{21.22}$$

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:

$$-\dot{\boldsymbol{\varepsilon}}^p + \dot{\lambda}\frac{\partial g}{\partial \boldsymbol{\sigma}} = 0 \quad \text{where} \quad -\dot{\boldsymbol{\varepsilon}}^p =: \frac{\partial f}{\partial \boldsymbol{\sigma}} \tag{21.23}$$

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.,

$$f(\boldsymbol{\sigma}) = -\boldsymbol{\sigma} : \dot{\boldsymbol{\varepsilon}}^p\,. \tag{21.24}$$

If we use a first-order update approach, the discretized equations for perfect plasticity are

$$\begin{aligned}
&\boldsymbol{\sigma}_{n+1} = \mathbf{C} : (\boldsymbol{\varepsilon}_{n+1} - \boldsymbol{\varepsilon}^p_{n+1}) = \boldsymbol{\sigma}^{\text{trial}}_{n+1} - \mathbf{C} : (\boldsymbol{\varepsilon}^p_{n+1} - \boldsymbol{\varepsilon}^p_n)\\
&\boldsymbol{\varepsilon}^p_{n+1} = \boldsymbol{\varepsilon}^p_n + \Delta\lambda \left.\frac{\partial g}{\partial \boldsymbol{\sigma}}\right|_{\boldsymbol{\sigma}_n} \quad \text{or} \quad \boldsymbol{\varepsilon}^p_{n+1} = \boldsymbol{\varepsilon}^p_n + \Delta\lambda \left.\frac{\partial g}{\partial \boldsymbol{\sigma}}\right|_{\boldsymbol{\sigma}_{n+1}}\\
&\quad g(\boldsymbol{\sigma}_{n+1}) \le 0\,, \quad \Delta\lambda \ge 0\,, \quad \Delta\lambda g(\boldsymbol{\sigma}_{n+1}) = 0
\end{aligned} \tag{21.25}$$

> 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.

21.4.3 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:

$$\begin{aligned}
&\text{minimize} \quad f(\boldsymbol{\sigma}) = \|\boldsymbol{\sigma}^{\text{trial}} - \boldsymbol{\sigma}\|^2\\
&\text{subject to} \quad g(\boldsymbol{\sigma}) \le 0
\end{aligned} \tag{21.26}$$

where

$$\|\boldsymbol{\sigma}\| = \sqrt{\boldsymbol{\sigma} : \boldsymbol{\sigma}} \tag{21.27}$$

The Lagrangian for this problem is

$$\mathcal{L}(\boldsymbol{\sigma}, \lambda) = f(\boldsymbol{\sigma}) + \Delta\lambda g(\boldsymbol{\sigma}) = \|\boldsymbol{\sigma}^{\text{trial}} - \boldsymbol{\sigma}\|^2 + \Delta\lambda g(\boldsymbol{\sigma}) \tag{21.28}$$

The Karush-Kuhn-Tucker conditions for this problem at the optimum value $\boldsymbol{\sigma}_{n+1}$ are

$$\begin{aligned}
&g(\boldsymbol{\sigma}_{n+1}) \le 0\,, \quad \Delta\lambda \ge 0\,, \quad \Delta\lambda g(\boldsymbol{\sigma}_{n+1}) = 0\\
&\frac{\partial f(\boldsymbol{\sigma}_{n+1})}{\partial \boldsymbol{\sigma}} + \Delta\lambda \frac{\partial g(\boldsymbol{\sigma}_{n+1})}{\partial \boldsymbol{\sigma}} = -2(\boldsymbol{\sigma}^{\text{trial}} - \boldsymbol{\sigma}_{n+1}) + \Delta\lambda \frac{\partial g(\boldsymbol{\sigma}_{n+1})}{\partial \boldsymbol{\sigma}} = \mathbf{0}
\end{aligned} \tag{21.29}$$

From the last condition we see that the closest distance using this criterion leads to a stress value of

$$\boldsymbol{\sigma}_{n+1} = \boldsymbol{\sigma}^{\text{trial}} - \tfrac{1}{2}\Delta\lambda \frac{\partial g(\boldsymbol{\sigma}_{n+1})}{\partial \boldsymbol{\sigma}} \tag{21.30}$$

But we have seen previously that the first-order stress update with backward Euler leads to

$$\boldsymbol{\sigma}_{n+1} = \boldsymbol{\sigma}^{\text{trial}} - \Delta\lambda \mathbf{C} : \frac{\partial g(\boldsymbol{\sigma}_{n+1})}{\partial \boldsymbol{\sigma}} \tag{21.31}$$

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

$$\|\boldsymbol{\sigma}\|_{\mathbf{C}^{-1}} = \sqrt{\boldsymbol{\sigma} : \mathbf{C}^{-1} : \boldsymbol{\sigma}} \tag{21.32}$$

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

$$\|\boldsymbol{\sigma}\|_{\mathsf{C}^{-1}} = \sqrt{\boldsymbol{\sigma} : \mathsf{C}^{-1} : \boldsymbol{\sigma}} \tag{21.33}$$

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.

21.4.4 Eigendecompositions in linear elasticity

The stiffness tensor for an isotropic elastic material is

$$\mathsf{C} = \lambda\, \mathbf{I} \otimes \mathbf{I} + 2\mu\, \mathsf{I}^s \tag{21.34}$$

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

$$\mathsf{C}^{-1} = \mathsf{S} = -\frac{\lambda}{2\mu(3\lambda + 2\mu)}\, \mathbf{I} \otimes \mathbf{I} + \frac{1}{2\mu}\, \mathsf{I}^s \tag{21.35}$$

Eigendecompositions of the stiffness and compliance tensors are defined via

$$\mathsf{C} : \mathbf{V} = \lambda \mathbf{V}\,, \quad \mathsf{S} : \mathbf{V} = \frac{1}{\lambda}\mathbf{V} \tag{21.36}$$

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.,

$$\mathbf{V}_i : \mathbf{V}_i = 1 \quad \text{and} \quad \mathbf{V}_i : \mathbf{V}_j = 0\,. \tag{21.37}$$

The stiffness and compliance tensors may then be represented as:

$$\mathsf{C} = \sum_{i=1}^{m} \lambda_i \mathbf{V}_i \otimes \mathbf{V}_i\,, \quad \mathsf{S} = \sum_{i=1}^{m} \frac{1}{\lambda_i} \mathbf{V}_i \otimes \mathbf{V}_i \tag{21.38}$$

where m is the number of non-zero and distinct eigenvalues. Note also that, in this eigenbasis, the symmetric rank-4 identity tensor is

$$\mathsf{I}^s = \sum_{i=1}^{m} \mathbf{V}_i \otimes \mathbf{V}_i\,. \tag{21.39}$$

Eigenprojectors are defined as rank-4 tensors that have the property (for $i \neq j$)

$$\mathsf{P}_i : \mathbf{V}_i = \mathbf{V}_i \,, \quad \mathsf{P}_i : \mathbf{V}_j = \mathbf{o} \tag{21.40}$$

If we apply the eigenprojector to the rank-4 identity tensor, we get

$$\mathsf{P}_k = \mathsf{P}_k : \mathsf{I}^s = \sum_{i=1}^{m} \mathsf{P}_k : (\mathbf{V}_i \otimes \mathbf{V}_i) = \sum_{i=1}^{m} (\mathsf{P}_k : \mathbf{V}_i) \otimes \mathbf{V}_i = (\mathsf{P}_k : \mathbf{V}_k) \otimes \mathbf{V}_k = \mathbf{V}_k \otimes \mathbf{V}_k \tag{21.41}$$

Therefore we may also write the eigendecomposition in terms of the eigenprojectors

$$\mathsf{C} = \sum_{i=1}^{m} \lambda_i \mathsf{P}_i \,, \quad \mathsf{S} = \sum_{i=1}^{m} \frac{1}{\lambda_i} \mathsf{P}_i \,, \quad \mathsf{I}^s = \sum_{i=1}^{m} \mathsf{P}_i \,. \tag{21.42}$$

For isotropic materials, a small amount of algebra shows that there are two unique eigenvectors which lead to the decomposition

$$\mathsf{C} = \lambda_1 \mathsf{P}_1 + \lambda_2 \mathsf{P}_2 \quad \text{where} \quad \mathsf{S} = \frac{1}{\lambda_1} \mathsf{P}_1 + \frac{1}{\lambda_2} \mathsf{P}_2 \tag{21.43}$$

and

$$\mathsf{P}_1 = \tfrac{1}{3} \mathbf{I} \otimes \mathbf{I} \,, \quad \mathsf{P}_2 = \mathsf{I}^s - \mathsf{P}_1 \,. \tag{21.44}$$

We can now express the stiffness and compliance tensors in terms of these eigenprojections:

$$\begin{aligned} \mathsf{C} &= \left(\kappa - \tfrac{2}{3} \mu \right) \mathbf{I} \otimes \mathbf{I} + 2\mu \, \mathsf{I}^s \\ &= 3\kappa \left(\tfrac{1}{3} \mathbf{I} \otimes \mathbf{I} \right) + 2\mu \left(\mathsf{I}^s - \tfrac{1}{3} \mathbf{I} \otimes \mathbf{I} \right) \end{aligned} \tag{21.45}$$

where κ is the bulk modulus and μ is the shear modulus. Also,

$$\begin{aligned} \mathsf{S} &= \frac{1}{3} \left(\frac{1}{3\kappa} - \frac{1}{2\mu} \right) \mathbf{I} \otimes \mathbf{I} + \frac{1}{2\mu} \mathsf{I}^s \\ &= \frac{1}{3\kappa} \left(\tfrac{1}{3} \mathbf{I} \otimes \mathbf{I} \right) + \frac{1}{2\mu} \left(\mathsf{I}^s - \tfrac{1}{3} \mathbf{I} \otimes \mathbf{I} \right) \end{aligned} \tag{21.46}$$

Therefore, we can write

$$\mathsf{C} = 3\kappa \, \mathsf{P}^{\text{iso}} + 2\mu \, \mathsf{P}^{\text{symdev}} \quad \text{and} \quad \mathsf{S} = \frac{1}{3\kappa} \mathsf{P}^{\text{iso}} + \frac{1}{2\mu} \mathsf{P}^{\text{symdev}} \tag{21.47}$$

where $\mathsf{P}^{\text{iso}} = \mathsf{P}_1$ and $\mathsf{P}^{\text{symdev}} = \mathsf{P}_2$.

It is also worth noting that if

$$\mathsf{C}^{1/2} : \mathsf{C}^{1/2} := \mathsf{C} \quad \text{and} \quad \mathsf{S}^{1/2} : \mathsf{S}^{1/2} := \mathsf{S} \tag{21.48}$$

then, using the property that $\mathsf{P}_1 : \mathsf{P}_2 = 0$,

$$\mathsf{C}^{1/2} = \sqrt{\lambda_1} \mathsf{P}_1 + \sqrt{\lambda_2} \mathsf{P}_2 \quad \text{where} \quad \mathsf{S}^{1/2} = \frac{1}{\sqrt{\lambda_1}} \mathsf{P}_1 + \frac{1}{\sqrt{\lambda_2}} \mathsf{P}_2 \tag{21.49}$$

In that case, we have

$$\mathsf{C}^{1/2} = \sqrt{3\kappa} \, \mathsf{P}^{\text{iso}} + \sqrt{2\mu} \, \mathsf{P}^{\text{symdev}} \quad \text{and} \quad \mathsf{S}^{1/2} = \frac{1}{\sqrt{3\kappa}} \mathsf{P}^{\text{iso}} + \frac{1}{\sqrt{2\mu}} \mathsf{P}^{\text{symdev}} \tag{21.50}$$

21.4.5 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

$$\|\boldsymbol{\sigma}\|_{\mathsf{S}} = \sqrt{\boldsymbol{\sigma} : \mathsf{S} : \boldsymbol{\sigma}} \tag{21.51}$$

can be interpreted as a standard Euclidean distance measure in a transformed stress space by observing that

$$\begin{aligned} \|\boldsymbol{\sigma}\|_{\mathsf{S}} &= \sqrt{(\boldsymbol{\sigma} : \mathsf{S}^{1/2}) : (\mathsf{S}^{1/2} : \boldsymbol{\sigma})} \quad \text{where} \quad \mathsf{S}^{1/2} : \mathsf{S}^{1/2} := \mathsf{S} \\ &= \sqrt{(\mathsf{S}^{1/2} : \boldsymbol{\sigma}) : (\mathsf{S}^{1/2} : \boldsymbol{\sigma})} \quad \text{using the major symmetry of} \quad \mathsf{S} \\ &= \sqrt{\boldsymbol{\sigma}^\star : \boldsymbol{\sigma}^\star} = \|\boldsymbol{\sigma}^\star\| \end{aligned} \tag{21.52}$$

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:

$$\boldsymbol{s} = \operatorname{dev}(\boldsymbol{\sigma}) = \boldsymbol{\sigma} - \tfrac{1}{3}\operatorname{tr}(\boldsymbol{\sigma})\mathbf{I} \quad \text{and} \quad \boldsymbol{t} = \operatorname{dev}(\boldsymbol{s}\cdot\boldsymbol{s}) = \boldsymbol{s}\cdot\boldsymbol{s} - \tfrac{1}{3}\operatorname{tr}(\boldsymbol{s}\cdot\boldsymbol{s})\mathbf{I} \tag{21.53}$$

The quantity $\boldsymbol{t}$ is also called the *Hill tensor*.

The Lode invariants of a stress tensor are

$$z = \tfrac{1}{\sqrt{3}}\operatorname{tr}(\boldsymbol{\sigma})\,, \quad r = \|\boldsymbol{s}\|\,, \quad \sin 3\theta = 3\sqrt{6}\det\left(\frac{\boldsymbol{s}}{\|\boldsymbol{s}\|}\right) \tag{21.54}$$

These invariants are associated with an orthonormal set of unit tensors

$$\mathbf{E}_z = \tfrac{1}{\sqrt{3}}\mathbf{I}\,, \quad \mathbf{E}_r = \frac{\boldsymbol{s}}{\|\boldsymbol{s}\|}\,, \quad \mathbf{E}_\theta = \frac{\frac{\boldsymbol{t}}{\|\boldsymbol{t}\|} - \sin 3\theta\,\frac{\boldsymbol{s}}{\|\boldsymbol{s}\|}}{\cos 3\theta} \tag{21.55}$$

The stress can be expressed in terms of the Lode basis as

$$\boldsymbol{\sigma} = z\,\mathbf{E}_z + r\,\mathbf{E}_r\,. \tag{21.56}$$

The transformed stress tensor

We can now compute the transformed stress tensor:

$$\boldsymbol{\sigma}^\star = \mathsf{S}^{1/2} : \boldsymbol{\sigma} = \left[\frac{1}{\sqrt{3\kappa}}\mathsf{P}^{\text{iso}} + \frac{1}{\sqrt{2\mu}}\mathsf{P}^{\text{symdev}}\right] : (z\,\mathbf{E}_z + r\,\mathbf{E}_r)\,. \tag{21.57}$$

We can show that

$$\mathsf{P}^{\text{iso}} : \mathbf{E}_z = \mathbf{E}_z\,, \quad \mathsf{P}^{\text{iso}} : \mathbf{E}_r = \mathbf{o}\,, \quad \mathsf{P}^{\text{symdev}} : \mathbf{E}_z = \mathbf{o}\,, \quad \mathsf{P}^{\text{symdev}} : \mathbf{E}_r = \mathbf{E}_r \tag{21.58}$$

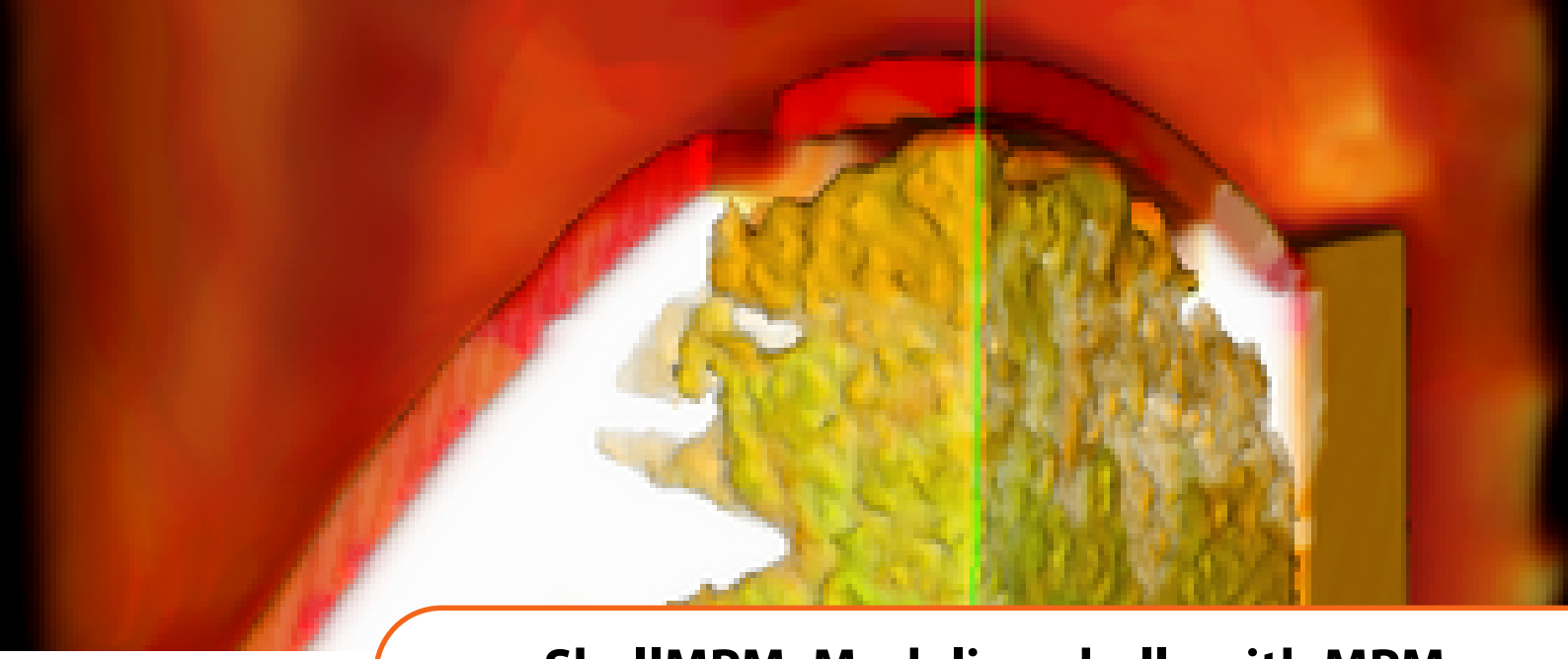
Therefore,

$$\boldsymbol{\sigma}^\star = \frac{z}{\sqrt{3\kappa}}\mathbf{E}_z + \frac{r}{\sqrt{2\mu}}\mathbf{E}_r \tag{21.59}$$

We can also show that the transformed stress vector remains geometrically unchanged (in the sense that angles are unchanged) if we express it as

$$\boldsymbol{\sigma}^{\star} = z\,\mathbf{E}_z + \sqrt{\frac{3\kappa}{2\mu}}\,r\,\mathbf{E}_r =: z\,\mathbf{E}_z + r'\,\mathbf{E}_r \tag{21.60}$$

So we have a straightforward way of computing stresses in the transformed space and use this idea in the geometrical closest point return algorithm.



22 — ShellMPM: Modeling shells with MPM

22.1 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 [82, 83]

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. [83]. A discussion of the formulation follows.

The velocity field in the shell is given by

$$\mathbf{w}(\alpha,\beta) = \mathbf{u}(\alpha,\beta) + z\,\boldsymbol{\omega}(\alpha,\beta)\times\mathbf{n}(\alpha,\beta) + \dot{z}\,\mathbf{n}(\alpha,\beta) \tag{22.1}$$

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 (22.1) can therefore be written as

$$\mathbf{w}(\alpha,\beta) = \mathbf{u}(\alpha,\beta) + z\,\mathbf{r}(\alpha,\beta) \tag{22.2}$$

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

$$\nabla \mathbf{w} = \left[\nabla^{(s)} \mathbf{u} + z \, \nabla^{(s)} \mathbf{r} \right] + \mathbf{r} \otimes \mathbf{n} \tag{22.3}$$

where $\mathbf{r} \otimes \mathbf{n}$ represents the dyadic product, and $\nabla^{(s)}$ is the in-surface gradient operator, defined as,

$$\nabla^{(s)} = [\nabla(\ )] \bullet \mathbf{I}^{(s)} \, . \tag{22.4}$$

The $\bullet$ represents a tensor inner product and $\mathbf{I}^{(s)}$ is the in-surface identity tensor (or the projection operator), defined as,

$$\mathbf{I}^{(s)} = \mathbf{I} - \mathbf{n} \otimes \mathbf{n}. \tag{22.5}$$

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,

$$\nabla \bullet \boldsymbol{\sigma} = \rho \, \mathbf{a} \tag{22.6}$$

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 (22.6) with respect to the surface of the shell is given by

$$\nabla^{(s)} \bullet \langle \boldsymbol{\sigma} \rangle = \rho \, \mathbf{a} \, . \tag{22.7}$$

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

$$\langle \boldsymbol{\sigma} \rangle := \frac{1}{h} \int_{-h^-}^{h^+} \boldsymbol{\sigma}(z) \, dz \tag{22.8}$$

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. [83] 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 [84], given by

$$\nabla^{(s)} \bullet \mathbf{M} - \mathbf{n} \bullet \langle \boldsymbol{\sigma} \rangle \bullet \mathbf{I}^{(s)} = \frac{1}{12} \rho \, h^2 \, \dot{\mathbf{r}} \tag{22.9}$$

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

$$\mathbf{M} := \mathbf{I}^{(s)} \bullet \left[\frac{1}{h} \int_{-h^-}^{h^+} \boldsymbol{\sigma}(z) \, z \, dz \right] \bullet \mathbf{I}^{(s)} \, . \tag{22.10}$$

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.

22.2 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 $S^{(1)}_{p,v}$.

22.2.1 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 [85]. 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,

$$\sum_p \chi_p(\mathbf{x}) = 1 \quad \forall \quad \mathbf{x} \in \Omega \tag{22.11}$$

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

$$f(\mathbf{x}) = \sum_p f_p\ \chi_p(\mathbf{x}) \tag{22.12}$$

where f_p is the value at a material point. Similarly, a continuous representation of the grid data is given by

$$g(\mathbf{x}) = \sum_v g_v\ S_v(\mathbf{x}) \tag{22.13}$$

where

$$\sum_v S_v(\mathbf{x}) = 1 \quad \forall \quad \mathbf{x} \in \Omega\ . \tag{22.14}$$

To interpolate particle data to the grid, the interpolation (or weighting functions) $S^{(1)}_{p,v}$ are used, which are defined as

$$S^{(1)}_{p,v} = \frac{1}{V_p} \int_{\Omega_p \cap \Omega} \chi_p(\mathbf{x})\ S_v(\mathbf{x})\ d\mathbf{x} \tag{22.15}$$

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

$$\sum_v S^{(1)}_{p,v} = 1 \quad \forall \quad \mathbf{x_p} \in \Omega_p\ . \tag{22.16}$$

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

$$m_v = \sum_p m_p\ S^{(1)}_{p,v}\ . \tag{22.17}$$

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 [85].

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

$$\mathbf{p}_v = \sum_p \mathbf{p}_p \, S^{(1)}_{p,v} \, . \tag{22.18}$$

At the grid, the rotation rate is recovered using

$$\mathbf{r}_v = \mathbf{p}_v / m_v \tag{22.19}$$

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.

22.2.2 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, 16]. subsection Compute the stress tensor. The stress tensor computation follows the procedure for hyperelastic materials cited in reference [23]. 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 (22.3). 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

$$\nabla S^{(1)}_{p,v} = \frac{1}{V_p} \int_{\Omega_p \cap \Omega} \chi_p(\mathbf{x}) \, \nabla S_v(\mathbf{x}) \, d\mathbf{x} \tag{22.20}$$

so that

$$\nabla \mathbf{u}_p = \sum_v \mathbf{u}_v \, \nabla S^{(1)}_{p,v} \, . \tag{22.21}$$

The gradient of the rotation rate ($\nabla \mathbf{r}$) is also interpolated to the particles using the same procedure, i.e.,

$$\nabla \mathbf{r}_p = \sum_v \mathbf{r}_v \, \nabla S^{(1)}_{p,v} \, . \tag{22.22}$$

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

$$\nabla^{(s)} \mathbf{u}_p = \nabla \mathbf{u}_p \bullet \left(\mathbf{I} - \mathbf{n}^n_p \otimes \mathbf{n}^n_p \right) \tag{22.23}$$

$$\nabla^{(s)} \mathbf{r}_p = \nabla \mathbf{r}_p \bullet \left(\mathbf{I} - \mathbf{n}^n_p \otimes \mathbf{n}^n_p \right) \tag{22.24}$$

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 (22.3). For a shell with three layers (top, center and bottom), the velocity gradients are given by

$$\nabla \mathbf{w}^{\text{top}}_p = \left[\nabla^{(s)} \mathbf{u}_p + h^+ \, \nabla^{(s)} \mathbf{r}_p \right] + \mathbf{r}^n_p \otimes \mathbf{n}^n_p \tag{22.25}$$

$$\nabla \mathbf{w}^{\text{cen}}_p = \nabla^{(s)} \mathbf{u}_p + \mathbf{r}^n_p \otimes \mathbf{n}^n_p \tag{22.26}$$

$$\nabla \mathbf{w}^{\text{bot}}_p = \left[\nabla^{(s)} \mathbf{u}_p - h^- \, \nabla^{(s)} \mathbf{r}_p \right] + \mathbf{r}^n_p \otimes \mathbf{n}^n_p \tag{22.27}$$

The increment of deformation gradient ($\Delta\mathbf{F}$) in each layer is computed using

$$\Delta\mathbf{F}_p = \Delta t\ \nabla\mathbf{w}_p + \mathbf{I} \tag{22.28}$$

The total deformation gradient ($\mathbf{F}$) in each layer is updated using

$$\tilde{\mathbf{F}}_p^{n+1} = \Delta\mathbf{F}_p \bullet \mathbf{F}_p^n \tag{22.29}$$

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

$$W = \frac{1}{2}K\left[\frac{1}{2}(J^2 - 1) - \ln J\right] + \frac{1}{2}G\left[\mathrm{tr}(\bar{\mathbf{b}}) - 3\right] \tag{22.30}$$

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

$$\bar{\mathbf{b}} := J^{-\frac{2}{3}}\mathbf{F} \bullet \mathbf{F}^T \tag{22.31}$$

The Cauchy stress then has the form

$$\boldsymbol{\sigma} = \frac{1}{2}K\left(J - \frac{1}{J}\right)\mathbf{I} + \frac{G}{J}\left[\bar{\mathbf{b}} - \frac{1}{3}\mathrm{tr}(\bar{\mathbf{b}})\right]. \tag{22.32}$$

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)

$$\mathbf{R} = \cos\theta\,(\mathbf{I} - \mathbf{a}\otimes\mathbf{a}) + \mathbf{a}\otimes\mathbf{a} - \sin\theta\,\mathbf{A} \tag{22.33}$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}. \tag{22.34}$$

The rotated deformation gradient in each layer is given by

$$\mathbf{F}_p^{\mathrm{rot}} = \mathbf{R} \bullet \tilde{\mathbf{F}}_p^{n+1} \bullet \mathbf{R}^T. \tag{22.35}$$

The updated stress ($\boldsymbol{\sigma}_p^{\mathrm{rot}}$) is calculated in this rotated coordinate system using equation (22.32). Thus,

$$\boldsymbol{\sigma}_p^{\mathrm{rot}} = \frac{1}{2}K\left(J_p^{\mathrm{rot}} - \frac{1}{J_p^{\mathrm{rot}}}\right)\mathbf{I} + \frac{G}{J_p^{\mathrm{rot}}}\left[\bar{\mathbf{b}}_p^{\mathrm{rot}} - \frac{1}{3}\mathrm{tr}(\bar{\mathbf{b}}_p^{\mathrm{rot}})\right]. \tag{22.36}$$

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

$$h_{n+1}^{+} = h_0^{+}\int_0^1 \overset{\circ}{F}_{zz}(+z)\,dz \tag{22.37}$$

$$h_{n+1}^{-} = h_0^{-}\int_0^1 \overset{\circ}{F}_{zz}(-z)\,dz \tag{22.38}$$

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

$$\mathbf{F}_p^{n+1} = \mathbf{R}^T \bullet \overset{\circ}{\mathbf{F}} \bullet \mathbf{R} \tag{22.39}$$

$$\boldsymbol{\sigma}_p^{n+1} = \mathbf{R}^T \bullet \overset{\circ}{\boldsymbol{\sigma}} \bullet \mathbf{R} \,. \tag{22.40}$$

The deformed volume of the shell is approximated using the Jacobian of the deformation gradient at the center of mass of the shell

$$V_p^{n+1} = V_p^o \, J_p^{n+1} \,. \tag{22.41}$$

22.2.3 Compute the internal force and moment.

The internal force for general materials is computed at the grid using the relation

$$\mathbf{f}_v^{\text{int}} = \sum_p \left[\boldsymbol{\sigma}_p^{n+1} \bullet \nabla S_{p,v}^{(1)} \right] V_p^{n+1} \tag{22.42}$$

For shell materials, this relation takes the form

$$\mathbf{f}_v^{\text{int}} = \sum_p \left[\left\langle \boldsymbol{\sigma}_p^{n+1} \right\rangle \bullet \nabla S_{p,v}^{(1)} \right] V_p^{n+1} \tag{22.43}$$

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 (22.9). To obtain the discretized form of equation (22.9), the equation is integrated over the volume of the shell leading to [83]

$$-\sum_p \left[\left(\mathbf{M}_p \bullet \nabla S_{p,v}^{(1)} \bullet \mathbf{I}^{(s)} \right) + \left(\mathbf{n}_p \bullet \left\langle \boldsymbol{\sigma}_p \right\rangle \bullet \mathbf{I}^{(s)} \right) S_{p,v}^{(o)} \right] V_p = \left(\frac{1}{12} \sum_p S_{p,v}^{(o)} \, m_p \, h_p^2 \right) \dot{\mathbf{r}}_v \,. \tag{22.44}$$

The average stress over the thickness of the shell is calculated using equation (22.8) and the average moment is calculated using equation (22.10). The trapezoidal rule is used in both cases. Thus,

$$\left\langle \boldsymbol{\sigma}_p^{n+1} \right\rangle = \frac{1}{h_{n+1}} \int_{-h_{n+1}^-}^{h_{n+1}^+} \boldsymbol{\sigma}_p^{n+1}(z) \, dz \tag{22.45}$$

$$\mathbf{M}_p^{n+1} = \mathbf{I}^{(s)} \bullet \left[\frac{1}{h_{n+1}} \int_{-h_{n+1}^-}^{h_{n+1}^+} \boldsymbol{\sigma}_p^{n+1}(z) \, z \, dz \right] \bullet \mathbf{I}^{(s)} \tag{22.46}$$

where

$$\mathbf{I}^{(s)} = \mathbf{I} - \mathbf{n}_p^n \otimes \mathbf{n}_p^n \tag{22.47}$$

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

$$\mathbf{m}_v^{\text{int}} = \sum_p \left[\left(\mathbf{M}_p^{n+1} \bullet \nabla S_{p,v}^{(1)} \bullet \mathbf{I}^{(s)} \right) + \left(\mathbf{n}_p^n \bullet \left\langle \boldsymbol{\sigma}_p^{n+1} \right\rangle \bullet \mathbf{I}^{(s)} \right) S_{p,v}^{(o)} \right] V_p^{n+1} \tag{22.48}$$

In practice, only the first term of equation (22.48) is interpolated to the grid and back to the particles. The equation of motion for rotational inertia is solved on the particles.

22.2.4 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

$$\dot{\mathbf{u}}_v = \frac{1}{m_v}\left(\mathbf{f}_v^{\text{ext}} - \mathbf{f}_v^{\text{int}}\right) \tag{22.49}$$

where $\mathbf{f}^{\text{ext}}$ are external forces.

The angular momentum equations are solved on the particles after interpolating the term

$$\tilde{\mathbf{m}}_v = \sum_p \left(\mathbf{M}_p^{n+1} \bullet \nabla S_{p,v}^{(1)} \bullet \mathbf{I}^{(s)}\right) \tag{22.50}$$

back to the material points to get $\tilde{\mathbf{m}}_p$. The rotational acceleration is calculated using

$$\dot{\mathbf{r}}_p = \left(\frac{12\ V_p}{m_p\ h_p^2}\right)\left[\mathbf{m}_p^{\text{ext}} - \tilde{\mathbf{m}}_p - \mathbf{n}_p \bullet \langle \boldsymbol{\sigma}_p \rangle \bullet \mathbf{I}^{(s)}\right] \tag{22.51}$$

22.2.5 Integrate the acceleration.

The linear acceleration in integrated using a forward Euler rule on the grid, giving the updated velocity on the grid as

$$\mathbf{u}_v^{n+1} = \mathbf{u}_v^n + \Delta t\ \dot{\mathbf{u}}_v \tag{22.52}$$

For the rotational acceleration, the same procedure is followed at each material point to obtain an intermediate increment

$$\Delta\tilde{\mathbf{r}}_p = \Delta t\ \dot{\mathbf{r}}_p \tag{22.53}$$

The factor $m_p\ h_p^2$ in the denominator of the right hand side of equation (22.51) 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 [86]

$$\left[\mathbf{I} + \beta\left(\mathbf{I} - \mathbf{n}_p^n \otimes \mathbf{n}_p^n\right)\right]\Delta\overset{\circ}{\mathbf{r}}_p = \Delta\tilde{\mathbf{r}}_p \tag{22.54}$$

where $\Delta\overset{\circ}{\mathbf{r}}_p$ is the corrected value of $\Delta\tilde{\mathbf{r}}_p$ and

$$\beta = \frac{6\ E}{V_p\ m_p}\left(\frac{\Delta t}{h}\right)^2 \tag{22.55}$$

which uses the Young's modulus E of the shell material. The intermediate rotation rate is updated using the corrected increment. Thus,

$$\overset{\star}{\boldsymbol{r}}_p^{n+1} = \mathbf{r}_p^n + \Delta\overset{\circ}{\mathbf{r}}_p\ . \tag{22.56}$$

22.2.6 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 (22.33) with rotation angle $\theta = |r|\Delta t$ and axis of rotation

$$\mathbf{a} = \frac{\mathbf{n}_p^n \times \overset{\star}{\boldsymbol{r}}_p^{n+1}}{|\mathbf{n}_p^n \times \overset{\star}{\boldsymbol{r}}_p^{n+1}|}\ . \tag{22.57}$$

The updated director is

$$\mathbf{n}_p^{n+1} = \Delta\mathbf{R} \bullet \mathbf{n}_p^n\ . \tag{22.58}$$

In addition, the rate of rotation has to be rotated so that the direction is perpendicular to the director using,

$$\mathbf{r}_p^{n+1} = \Delta\mathbf{R} \bullet \overset{\star}{\boldsymbol{r}}_p^{n+1}\ . \tag{22.59}$$

22.2.7 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

$$\mathbf{u}_p^{n+1} = \sum_v \mathbf{u}_v^{n+1} S_{p,v}^{(1)} \tag{22.60}$$

22.3 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.

22.3.1 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 22.1.

Table 22.1: 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 22.1. Substantial deformation of the shell

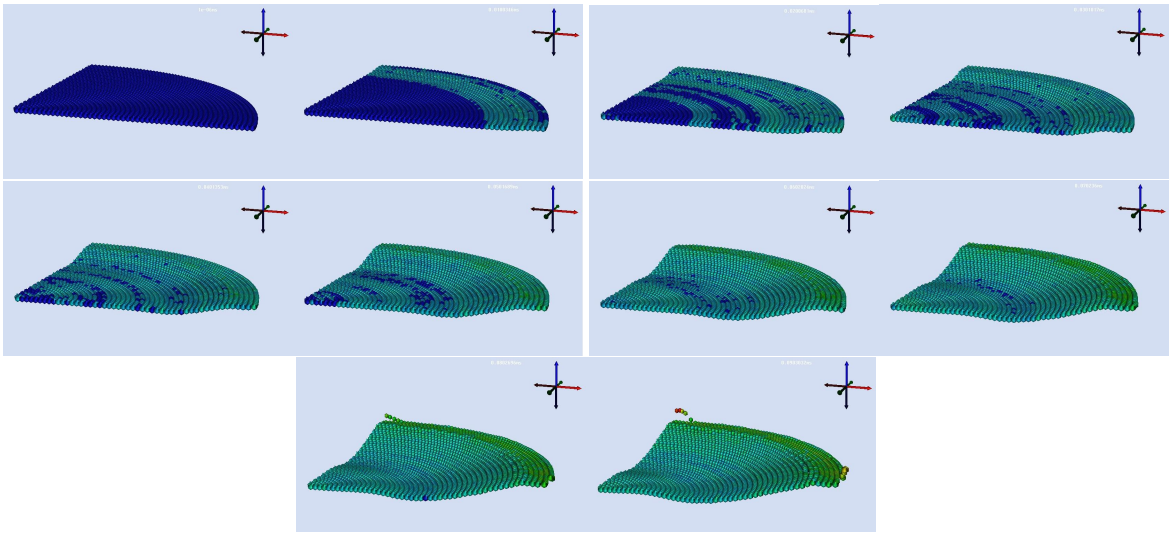


Figure 22.1: Deformation of punched circular plane shell.

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 (22.51). 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 [87]. 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 [82, 88]. Such locking effects can be reduced using an addition hour glass control step [82] in the simulation.

22.3.2 Pinched Cylindrical Shell

The pinched cylindrical shell is one of the benchmark problems proposed by MacNeal and Harder [89]. The cylindrical shell that has been simulated in this work has dimensions similar to those used by Li et al. [90]. 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 22.1). 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 22.2. The deformation

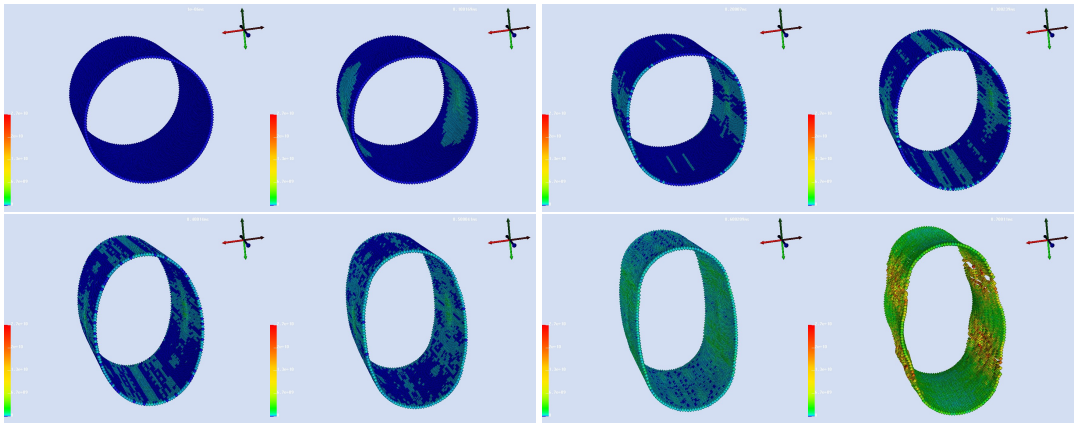


Figure 22.2: Deformation of pinched cylindrical shell.

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.

22.3.3 Inflating Spherical Shell

The inflating spherical shell problem is similar to that used to model lipid bilayers by Ayton et al. [91]. 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 22.3. 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 [92].

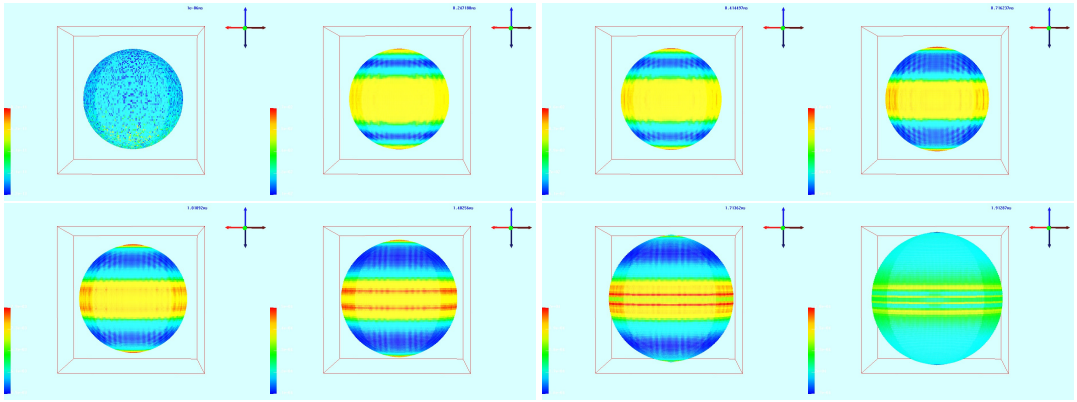


Figure 22.3: Deformation of inflating spherical shell.

22.4 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. [83]. 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.

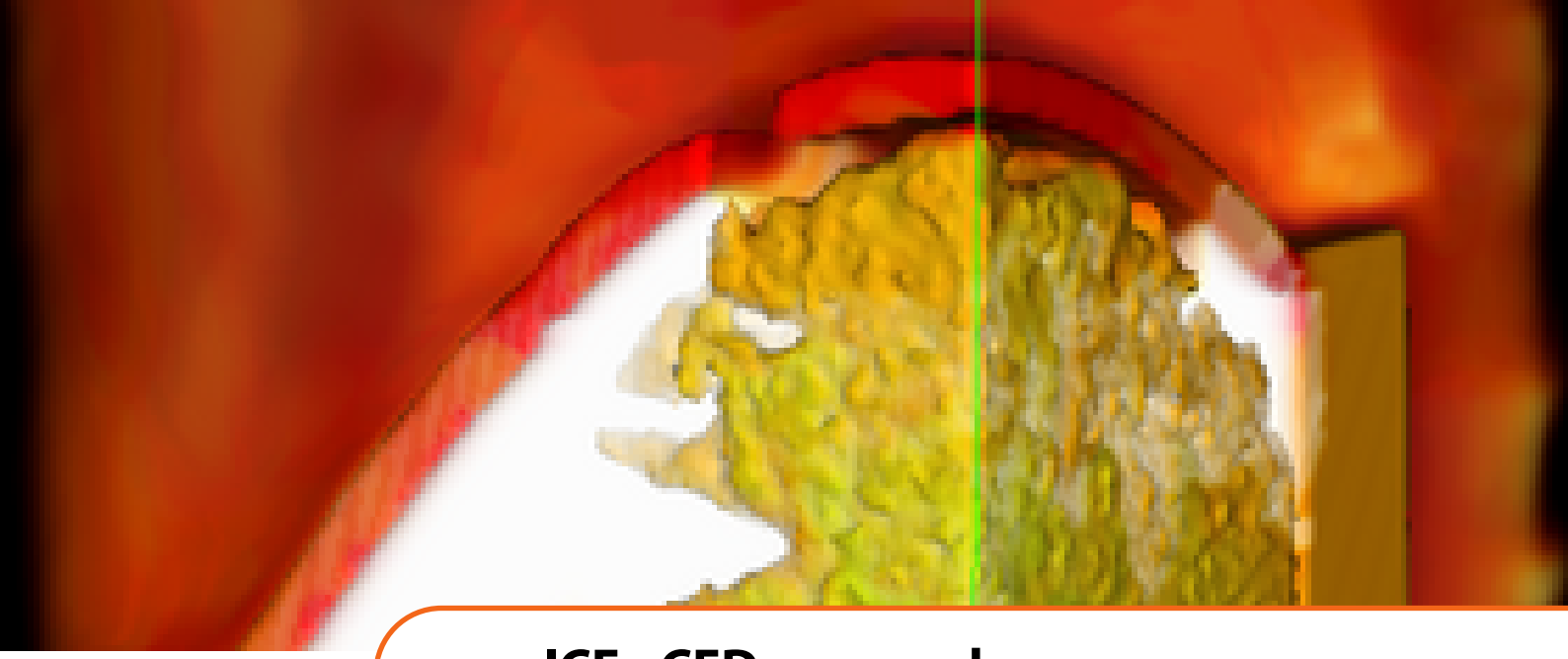
22.5 Alternative approaches

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. [93] 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 .



23 — ICE - CFD approach

23.1 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.

23.1.1 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:

$$\frac{1}{V}\frac{D_r M_r}{Dt} = \sum_{i=1}^{N} \Gamma_{rs} \tag{23.1}$$

$$\frac{1}{V}\frac{D_r}{Dt}(M_r \mathbf{u}_r) = \theta_r \nabla \cdot \boldsymbol{\sigma} + \nabla \cdot \theta_r(\boldsymbol{\sigma}_r - \boldsymbol{\sigma}) + \rho_r \mathbf{g} + \sum_{i=1}^{N} \mathbf{f}_{rs} + \sum_{i=1}^{N} \mathbf{u}_{rs}^{+} \Gamma_{rs} \tag{23.2}$$

$$\frac{1}{V}\frac{D_r}{Dt}(M_r e_r) = -\rho_r p \frac{D_r v_r}{Dt} + \theta_r \boldsymbol{\tau}_r : \nabla \mathbf{u}_r - \nabla \cdot \mathbf{j}_r + \sum_{i=1}^{N} q_{rs} + \sum_{i=1}^{N} h_{rs}^{+} \Gamma_{rs} \tag{23.3}$$

Equations (23.1-23.3) 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. (23.2) 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. (23.3), $\sum_{s=1}^{N} q_{rs}$ 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. (23.1-23.3) 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. (23.2) and (23.3), 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:

$$e_r = e_r(v_r, T_r) \tag{23.4}$$

$$v_r = v_r(p, T_r) \tag{23.5}$$

and

$$\theta_r = \rho_r v_r \tag{23.6}$$

$$0 = 1 - \sum_{i=1}^{N} \rho_s v_s \tag{23.7}$$

Equations (23.4) and (23.5) are, respectively, the caloric and thermal equations of state. Equation (23.6) defines the volume fraction, θ, as the volume of r-material per total material volume, and with that definition, Equation (23.7), 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 [94].

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:

$$\boldsymbol{\sigma}_r \equiv \boldsymbol{\sigma}_r(\nabla \mathbf{u}_r, \boldsymbol{E}_r, T_r, \ldots) \tag{23.8}$$

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 (23.1-23.8) form a set of eight equations for the eight-element state vector, $[M_r, \mathbf{u}_r, e_r, T_r, v_r, \theta_r, \boldsymbol{\sigma}_r, p]$, 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:

$$V_t = \sum_{r=1}^{N} M_r v_r \tag{23.9}$$

so the volume fraction is $\theta_r = M_r v_r / V_t$ (which sums to one by definition). An evolution equation for the r-material specific volume, derived from the time variation of Eqs. (23.4-23.7), has been developed in [11]. It is stated here as:

$$\frac{1}{V}\frac{D_r}{Dt}(M_r v_r) = f_r^{\theta} \nabla \cdot \mathbf{u} + \left[v_r \Gamma_r - f_r^{\theta} \sum_{i=1}^{N} v_s \Gamma_s \right] + \left[\theta_r \beta_r \frac{D_r T_r}{Dt} - f_r^{\theta} \sum_{i=1}^{N} \theta_s \beta_s \frac{D_s T_s}{Dt} \right]. \tag{23.10}$$

where

$$f_r^{\theta} = \frac{\theta_r \kappa_r}{\sum_{s=1}^{N} \theta_s \kappa_s} \tag{23.11}$$

and κ_r is the r-material bulk compressibility.

The evaluation of the multi-material equation of state (Eq. (23.7)) 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 23.2 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. 25.

23.2 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 [95] developed by Kashiwa and others at Los Alamos National Laboratory [96]. "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. 23.1-23.3); 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 [96].

In the mixed frame approach used here, a modification to the multi-material equation of state is needed. Equation (23.7) 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. (23.7) has no physical solutions (because the gas pressure is only positive). This can be seen schematically in Fig. 23.1, 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.

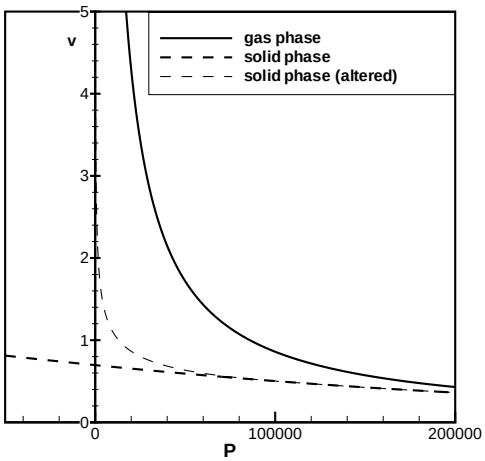
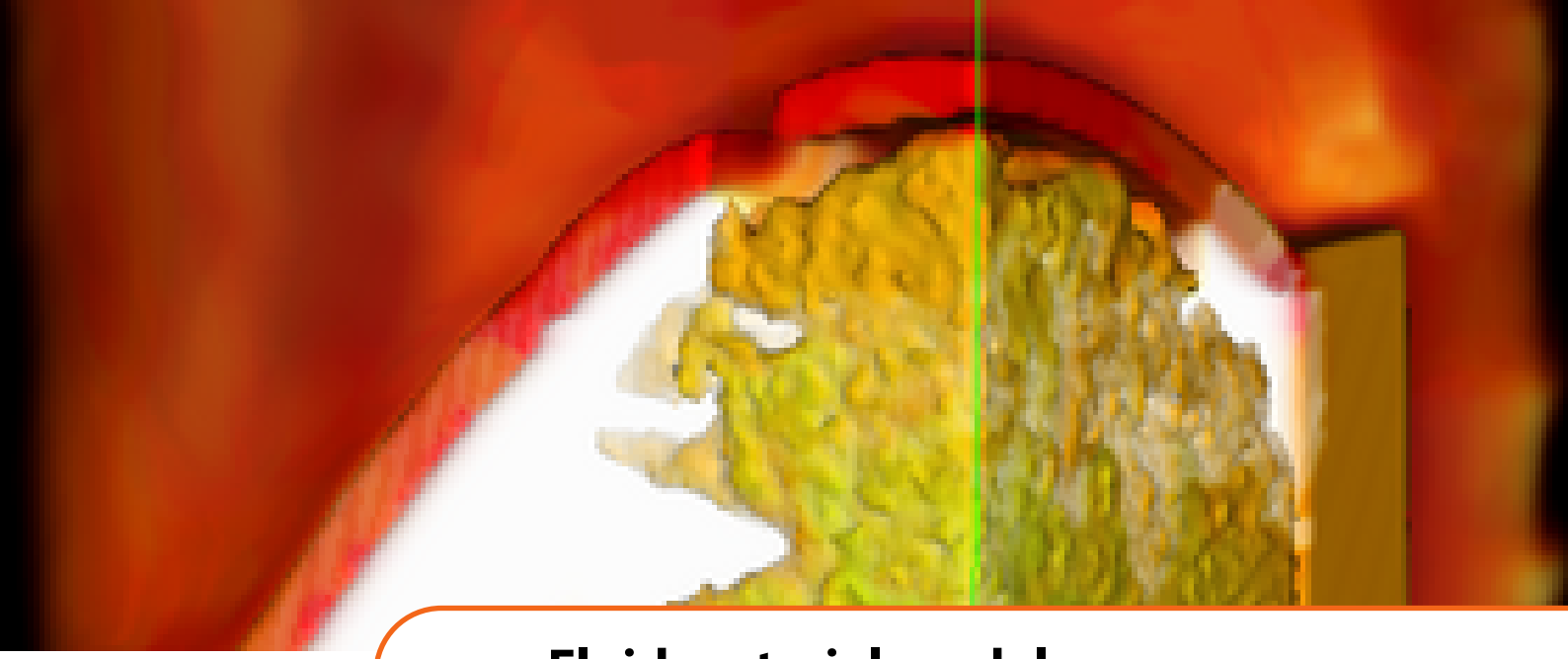


Figure 23.1: Specific volume *vs* pressure for a gas phase material and a solid phase material. Light dashed line reflects an altered solid phase equation of state to keep all materials in positive equilibration pressure space.



24 — 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 .

24.1 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.

24.1.1 The JWL++ Detonation Model

The detonation model used in two of the calculations discussed in Section 25.3 is a reactive flow model known as JWL++[97]. 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 25.1, 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 [21] used for the solid reactant material is given by:

$$p = \frac{1}{n\kappa}\left(\frac{1}{v^n} - 1\right) \tag{24.1}$$

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:

$$P = A\exp(-R_1 v) + B\exp(-R_2 v) + \frac{C}{\rho_0 \kappa v^{n-1}} \tag{24.2}$$

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:

$$\frac{dF}{dt} = G(p+q)^b(1-F) \tag{24.3}$$

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:

$$F = \frac{\rho_{\text{product}}}{\rho_{\text{reactant}} + \rho_{\text{product}}} \tag{24.4}$$

is the burn fraction in a cell. This can be differentiated and solved for a mass burn rate in terms of dF:

$$\Gamma = \frac{dF}{dt}\left(\rho_{\text{reactant}} + \rho_{\text{product}}\right) \tag{24.5}$$

24.1.2 Deflagration Model

The rate of thermal burning, or deflagration, of a monopropellant solid explosive is typically assumed to behave as:

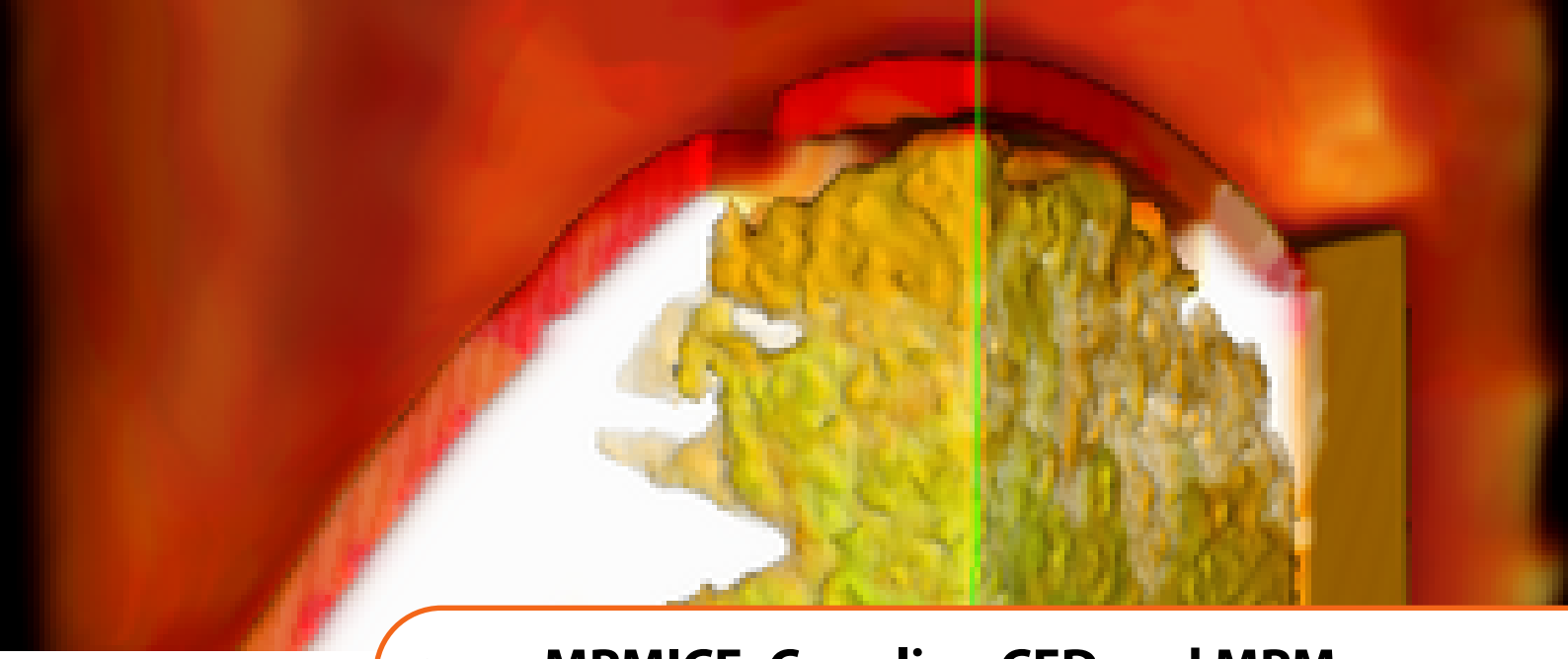
$$D = Ap^n \tag{24.6}$$

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 [98]. 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:

$$\Gamma = aD\rho_{\text{reactant}} \tag{24.7}$$

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 [99], which includes representation of the material's viscoelastic response, and considers effects of micro-crack growth within the granular composite material.



25 — 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 [100, 101], 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 23 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 25.3 to better appreciate the direction that the subsequent development is headed.

25.1 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. 25.1.3.

25.1.1 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 [102] developed by Kashiwa and others at Los Alamos National Laboratory [103]. "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. 23.1-23.3); 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 [103].

In the mixed frame approach used here, a modification to the multi-material equation of state is needed. Equation 23.7 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. 23.7 has no physical solutions (because the gas pressure is only positive). This can be seen schematically in Fig. 23.1, 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.

25.1.2 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 [104], large deformations [105] and fracture [106], to name a few. After the description of MPM, its incorporation within the multi-material solution is described in Sec. 25.1.3.

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.

25.1.3 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 23.2-23.3. 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. 1.69, 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. 23.7 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. 23.10. 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. 23.7 can be carried out at each cell using a Newton-Raphson technique[107], 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 [103] and [11] but here, only the result is given:

$$u_r^* = \frac{\rho_{r_L} u_{r_L} + \rho_{r_R} u_{r_R}}{\rho_{r_L} + \rho_{r_R}} - \left(\frac{2 v_{r_L} v_{r_R} \Delta t}{v_{r_L} + v_{r_R}}\right)\left(\frac{p_{\text{eq}_R} - p_{\text{eq}_L}}{\Delta x}\right) + g\Delta t \tag{25.1}$$

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. 24.1.
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:

$$\Delta p = \Delta t \frac{\sum_{r=1}^{N} v_r \Gamma_r - \sum_{r=1}^{N} \nabla \cdot (\theta_r^* u_r^*)}{\sum_{r=1}^{N} \theta_r \kappa_r} \tag{25.2}$$

$$p = p_{\text{eq}} + \Delta p \tag{25.3}$$

where κ_r is the r-material bulk compressibility. The first term in the numerator of Eq. 25.2 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 [103].

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:

$$p^* = \left(\frac{p_L}{\rho_L} + \frac{p_R}{\rho_R}\right) \Big/ \left(\frac{1}{\rho_L} + \frac{1}{\rho_R}\right) \tag{25.4}$$

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. 1.73) 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:

$$\Delta(m)_r = \Delta t\ V \sum_{s=1,s\neq r}^{N} \Gamma_s \tag{25.5}$$

$$\Delta(mu)_r = -\Delta t\ V \left[\theta_r \nabla p^* + \nabla\cdot\theta_r(\boldsymbol{\sigma}_r - \boldsymbol{\sigma}) + \sum_{s=1,s\neq r}^{N} u_s \Gamma_s \right] \tag{25.6}$$

$$\Delta(me)_r = -\Delta t\ V \left[f^{\theta r} p \sum_{s=1}^{N} \nabla\cdot(\theta_r^* u_r^*) + \sum_{s=1,s\neq r}^{N} e_s \Gamma_s \right] \tag{25.7}$$

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.

$$(m)_r^L = (m)_r^t + \Delta(m)_r \tag{25.8}$$

$$(mu)_r^{L-} = (mu)_r^t + \Delta(mu)_r \tag{25.9}$$

$$(me)_r^{L-} = (me)_r^t + \Delta(me)_r \tag{25.10}$$

10. **Momentum and heat exchange:** The exchange of momentum and heat between materials is computed according to:

$$(mu)_r^L = (mu)_r^{L-} + \Delta t\ m_r \sum_{s=1}^{N} \theta_r\theta_s K_{rs}(u_s^L - u_r^L) \tag{25.11}$$

$$(me)_r^L = (me)_r^{L-} + \Delta t\ m_r\ c_{v_r} \sum_{s=1}^{N} \theta_r\theta_s H_{rs}\left(T_s^L - T_r^L\right) \tag{25.12}$$

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:

$$\Delta(mv)_r = \Delta t\ V \left[v_r\Gamma_r + f^{\theta r}\nabla\cdot\sum_{s=1}^{N}\theta_s^* u_s^* + \theta_r\beta_r\dot{T}_r - f^{\theta r}\sum_{s=1}^{N}\theta_s\beta_s\dot{T}_s \right] \tag{25.13}$$

$$(mv)_r^L = (mv)_r^n + \Delta(mv)_r \tag{25.14}$$

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 [108], 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. 1.72.
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. 1.76.

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 [103].

25.2 Models

The governing equations given in Section 23.1.1 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 24. Materials used by the MPM component are discussed elsewhere in the Vaango Theory Manual.

25.3 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 [109], 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.

25.3.1 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 [97]. 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

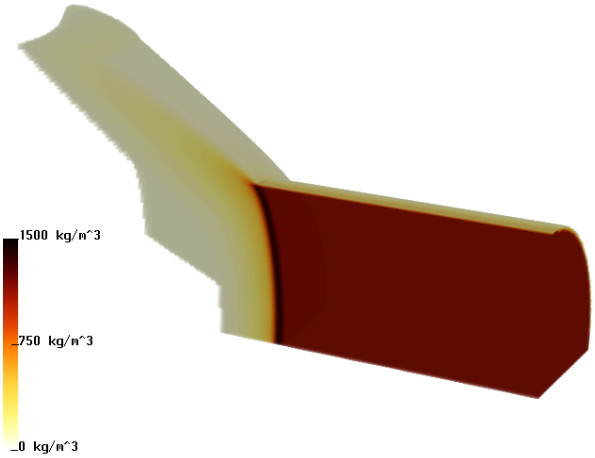


Figure 25.1: Unconfined 12 mm "rate-stick". The mass density of the reactant material is volume rendered, and shows evidence of the curvature of the reaction front, and the compression of the reactant just ahead of the reaction. Behind the detonation, most of the reactant material is consumed.

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 25.1, 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 25.2 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.

25.3.2 Cylinder Test Simulation

The cylinder test is an experiment which is frequently used to calibrate equations of state for detonation products of reaction [110]. 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

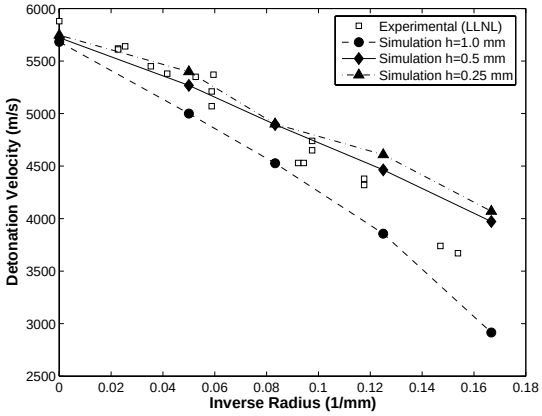


Figure 25.2: Detonation velocity vs. inverse radius. Experimental and numerical data are presented, and indicate good agreement of the model with experiment, as well as convergence of detonation velocity with grid resolution.

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 25.3 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 25.4. 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.

25.3.3 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

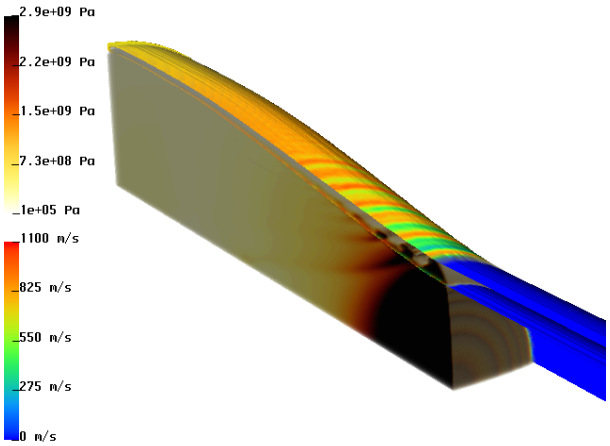


Figure 25.3: Copper cylinder test simulation. The walls of the copper tube are depicted as an isosurface of density of the copper material and are colored by velocity magnitude. Pressure is represented by a volume rendering, and indicates the progress of the detonation, as well as the interaction of the pressurized products of reaction with the confining walls.

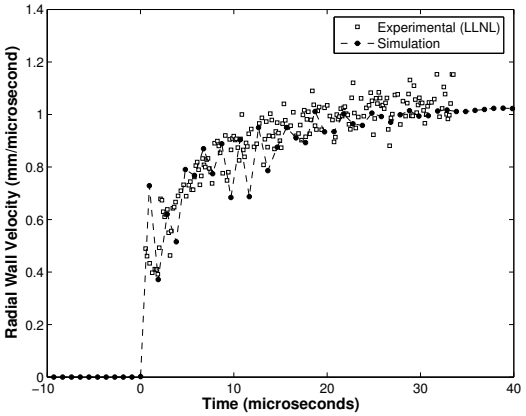


Figure 25.4: Copper cylinder test simulation. Experimental and computational velocities of the cylinder vs. time. Data was collected at a point 25 cm from the point of initiation of the detonation.

Table 25.1: 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.

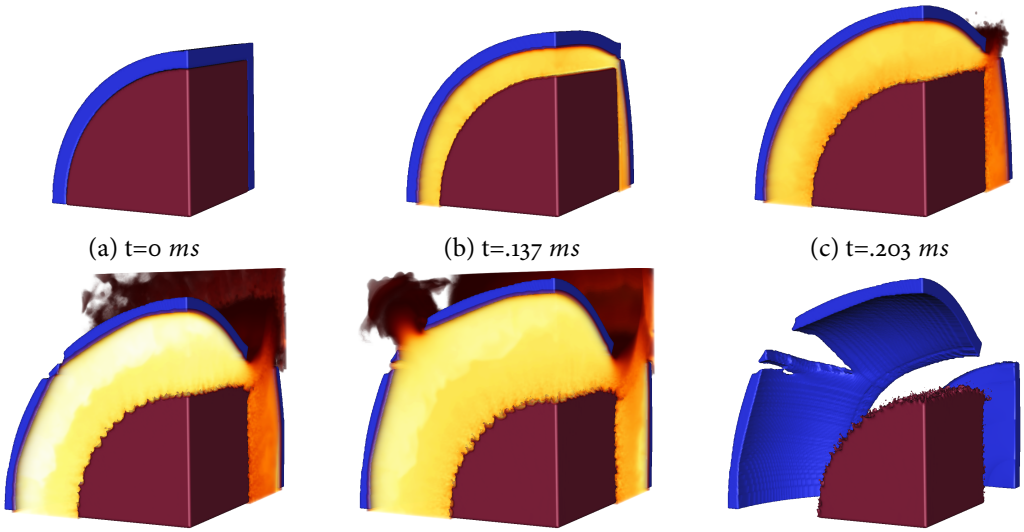
Mechanical properties for PBX 9501 were obtained from the literature [99], while the material constants used in the modeling of 4340 steel are shown in Table 25.1. A temperature-dependent specific heat model [111] was used to compute the internal energy and the rate of temperature increase in the material. 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. 25.5. In each panel, the container and explosive are depicted by isosurfaces, blue and red, respectively. In Fig. 25.5b-25.5e, a volume rendering of the mass density of the product material of the reaction is also included. Fig. 25.5a 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. 25.5c), and ultimately also through a rupture in middle (Fig. 25.5e). The formation of these openings is governed by material localization.

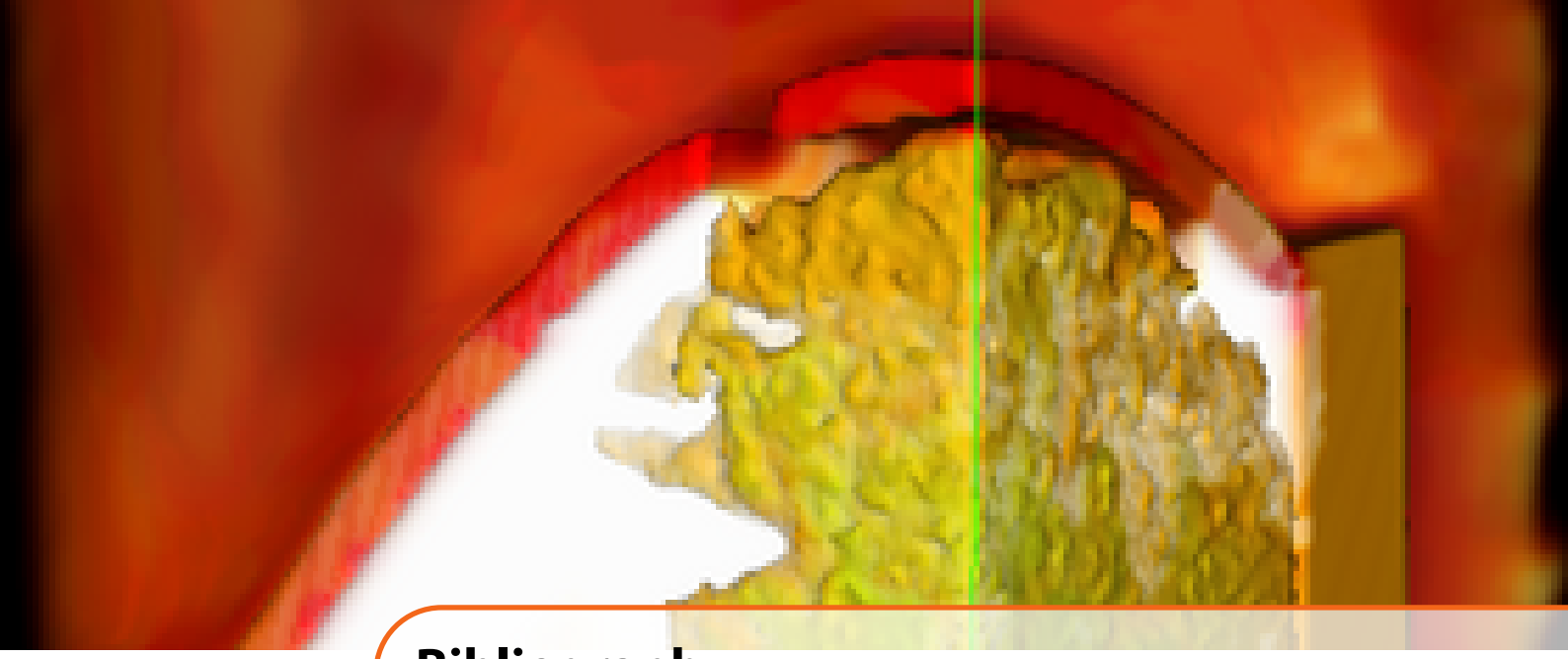
Since no surface tracking is required in this method, there is no requirement to track the creation of the new surfaces that occur due to material failure. Gas is free to escape through the openings simply because there is no longer anything in those computational cells to prevent it once the gap is 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 25.5: 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 9, 18, 175, 176, 191, 193).

[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 9, 13, 16, 18, 191, 193).

[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 9, 193).

[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 9).

[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 9).

[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 10, 176).

[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 10, 14, 18, 21).

[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 10, 14, 25).

[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 10).

[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 10).

[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 10, 183, 185, 194).

[12] 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 pages 13, 28–30).

[13] J. VonNeumann and R. D. Richtmyer. "A method for the numerical calculation of hydrodynamic shocks". In: *J. Appl. Phys.* 21 (1950), page 232 (cited on page 18).

[14] M. L. Wilkins. *Computer Simulation of Dynamic Phenomena*. Berlin: Springer-Verlag, 1999 (cited on pages 18, 68).

[15] 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 19).

[16] 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 20, 28, 29, 176).

[17] 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 25).

[18] Raymond W Ogden. *Non-linear elastic deformations*. Courier Corporation, 1997 (cited on page 36).

[19] Morteza M Mehrabadi and Stephen C Cowin. "Eigentensors of linear anisotropic elastic materials". In: *The Quarterly Journal of Mechanics and Applied Mathematics* 43.1 (1990), pages 15–41 (cited on page 50).

[20] L. Cueto-Felgueroso et al. "On the Galerkin formulation of the smoothed particle hydrodynamics method". In: *Int. J. Numer. Meth. Engng* 60 (2004), pages 1475–1512 (cited on page 56).

[21] F. D. Murnaghan. "The compressibility of media under extreme pressures". In: *Proc. Natl. Acad. Sci.* 30 (1944), pages 244–247 (cited on pages 56, 189).

[22] P. C. Souers S. Anderson J. Mercer E. McGuire and P. Vitello. "JWL++: A Simple Reactive Flow Code Package for Detonation". In: *Propellants, Explosives, Pyrotechnics* 25 (2000), pages 54–58 (cited on page 56).

[23] JC Simo and TJR Hughes. *Computational Inelasticity*. New York: Springer-Verlag, 1998 (cited on pages 60, 176).

[24] JA Weiss. "A constitutive model and finite element representation for transversely isotropic soft tissues". In: *Ph.D. Dissertation* Department of Bioengineering, University of Utah (1994) (cited on page 61).

[25] Bernhard Schölkopf et al. "New support vector algorithms". In: *Neural computation* 12.5 (2000), pages 1207–1245 (cited on page 62).

[26] Alex J Smola and Bernhard Schölkopf. "A tutorial on support vector regression". In: *Statistics and computing* 14.3 (2004), pages 199–222 (cited on page 62).

[27] Vladimir Vapnik. "The support vector method of function estimation". In: *Nonlinear Modeling*. Springer, 1998, pages 55–85 (cited on page 62).

[28] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: A library for support vector machines". In: *ACM transactions on intelligent systems and technology (TIST)* 2.3 (2011), page 27 (cited on page 62).

[29] 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 68, 72).

[30] Y. P. Varshni. "Temperature dependence of the elastic constants". In: *Physical Rev. B* 2.10 (1970), pages 3952–3958 (cited on page 72).

[31] 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 72).

[32] 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 72, 82, 95).

[33] 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 72).

[34] 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 pages 73, 85).

[35] 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 pages 73, 135).

[36] 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 76).

[37] 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 76).

[38] Günter Bernauer and Wolfgang Brocks. "Micro-mechanical modelling of ductile damage and tearing–results of a European numerical round robin". In: *Fatigue and Fracture of Engineering Materials and Structures* 25.4 (2002), pages 363–384 (cited on page 77).

[39] 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 81).

[40] 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 82).

[41] 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 82).

[42] 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 83).

[43] 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 83).

[44] F. J. Zerilli. "Dislocation mechanics-based constitutive equations". In: *Metall. Mater. Trans. A* 35A (2004), pages 2547–2555 (cited on page 83).

[45] 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 83).

[46] 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 84).

[47] 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 84).

[48] U. F. Kocks. "Realistic constitutive relations for metal plasticity". In: *Materials Science and Engrg.* A317 (2001), pages 181–187 (cited on page 84).

[49] Jeevana Dasari. "A unified viscoplastic model for the inelastic behavior of ice". PhD thesis. Fairbanks, Alaska: University of Alaska, 2006 (cited on page 86).

[50] Jonah H Lee. "Plane strain indentation of snow at the microscale". In: *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. Volume 43291. 2008, pages 719–727 (cited on page 86).

[51] 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 91).

[52] 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 91).

[53] 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 96).

[54] 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 97).

[55] 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 98).

[56] J. W. Hancock and A. C. MacKenzie. "On the mechanisms of ductile failure in high-strength steels subjected to multi-axial stress-states". In: *J. Mech. Phys. Solids* 24 (1976), pages 147–167 (cited on page 99).

[57] 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 99).

[58] B. Banerjee. "Material Point Method simulations of fragmenting cylinders". In: *Proc. 17th ASCE Engineering Mechanics Conference (EM2004)*. Newark, Delaware, 2004 (cited on page 101).

[59] 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 101).

[60] 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 101).

[61] 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 101).

[62] 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 102).

[63] 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 102).

[64] 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 102).

[65] 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 102).

[66] 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 102).

[67] D. C. Drucker. "A definition of stable inelastic material". In: *J. Appl. Mech.* 26 (1959), pages 101–106 (cited on page 102).

[68] 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 102).

[69] Hiroyasu Yamamoto. "Conditions for shear localization in the ductile fracture of void-containing materials". In: *International Journal of Fracture* 14.4 (1978), pages 347–365 (cited on page 102).

[70] 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 102).

[71] 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 105).

[72] Patrizio Neff, Bernhard Eidel, and Robert J Martin. "Geometry of logarithmic strain measures in solid mechanics". In: *Archive for Rational Mechanics and Analysis* 222.2 (2016), pages 507–572 (cited on page 125).

[73] D Sheng, SW Sloan, and HS Yu. "Aspects of finite element implementation of critical state models". In: *Computational mechanics* 26.2 (2000), pages 185–196 (cited on pages 128, 130).

[74] Rebecca M Brannon. "Elements of phenomenological plasticity: geometrical insight, computational algorithms, and topics in shock physics". In: *Shockwave science and technology reference library*. Springer, 2007, pages 225–274 (cited on page 134).

[75] 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 135).

[76] K. H. Roscoe and J. B. Burland. *On the generalized stress-strain behavior of wet clay*. 1968 (cited on page 135).

[77] A. N. Schofield and P. Wroth. *Critical state soil mechanics*. McGraw-Hill, 1968 (cited on page 135).

[78] 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 135).

[79] 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 135).

[80] 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 135).

[81] 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 135).

[82] 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 173, 181).

[83] 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 173, 174, 178, 182).

[84] H. L. Schreyer. *Lecture Notes in Plate Theory*. Albuquerque: University of New Mexico, 1997 (cited on page 174).

[85] 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 175, 176).

[86] B. A. Kashiwa and M. W Lewis. *CFDLIB version 02. 1*. Los Alamos National Laboratories. Los Alamos, New Mexico, 2002 (cited on page 179).

[87] J. E. Guilkey and J. A. Weiss. "A general membrane formulation for use with the material point method". In preparation. 2002 (cited on page 181).

[88] A. Libai and J. G. Simmonds. *The Nonlinear Theory of Elastic Shells: 2nd Edition*. Cambridge, UK: Cambridge University Press, 1998 (cited on page 181).

[89] 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 181).

[90] 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 181).

[91] 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 181).

[92] 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 181).

[93] 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 182).

[94] 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 184).

[95] 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 185).

[96] 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 185, 186).

[97] 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 189, 196).

[98] 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 190).

[99] 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 190, 200).

[100] 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 191).

[101] 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 191).

[102] 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 192).

[103] 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 192, 194, 196).

[104] 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 193).

[105] 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 193).

[106] 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 193).

[107] F.H. Harlow and A.A. Amsden. "Flow of Interpenetrating Material Phases". In: *J. Comp. Phys.* 18 (1975), pages 440–464 (cited on page 194).

[108] 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 195).

[109] 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 196).

[110] 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 197).

[111] 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 200).