PRISMS-Plasticity
Continuum Elastoplasticity Formulation

In the following formulation, we consider continuum elastoplasticity with quasistatic finite strain deformation of an isotropic material, with isochoric plasticity and isotropic strain hardening.

1 Kinematics

We model continuum elastoplasticity with a multiplicative decomposition of the deformation gradient into elastic and plastic parts.

$$\boldsymbol{F} = \boldsymbol{F}^e \boldsymbol{F}^p \tag{1}$$

We also specify the following (expected) relations:

$$\boldsymbol{b}^e = \boldsymbol{F}^e \boldsymbol{F}^{e^T} \tag{2}$$

$$\boldsymbol{C}^p = \boldsymbol{F}^{p^T} \boldsymbol{F}^p \tag{3}$$

$$\implies \boldsymbol{b}^e = \boldsymbol{F} \boldsymbol{C}^{p^{-1}} \boldsymbol{F}^T \tag{4}$$

The eigenvalues of $\boldsymbol{b}^e$ are the squares of the principal elastic stretches (λ_e^A), and the associated spectral decomposition of $\boldsymbol{b}^e$ is

$$\boldsymbol{b}^e = \sum_{A=1}^{3} \lambda_e^{A^2} \boldsymbol{n}^A \otimes \boldsymbol{n}^A \tag{5}$$

2 Constitutive laws

For nonlinear hyperelastic models, we have

$$\boldsymbol{\tau} : \boldsymbol{d} = \dot{w} \tag{6}$$

where $\boldsymbol{\tau}$ is the Kirchhoff stress, $\boldsymbol{d}$ is the rate of deformation tensor, and w is the strain energy density function. For isotropic hyperelasticity, the strain energy density can be written in terms of $\boldsymbol{b}^e$ or in terms of the invariants or eigenvalues of $\boldsymbol{b}^e$. It is also common to represent energy stored in the plastic deformation using a scalar α to represent equivalent plastic strain, so that $w = \hat{w}(\boldsymbol{b}^e, \alpha) = \bar{w}(\lambda_e^{a^2}, \alpha)$. Recognizing that the dissipation of energy must be non-negative, this equation can then be written as

$$\boldsymbol{\tau} : \boldsymbol{d}^p - \frac{\partial w}{\partial \alpha} \dot{\alpha} \geq 0 \tag{7}$$

The eigenvalues of $\boldsymbol{\tau}$ are the principal stresses (β^A), and the eigenvectors of $\boldsymbol{\tau}$ are equal to those of $\boldsymbol{b}^e$, resulting in a spectral decomposition of

$$\boldsymbol{\tau} = \sum_{A=1}^{3} \beta^A \boldsymbol{n}^A \otimes \boldsymbol{n}^A \tag{8}$$

The following relation holds in this case:

$$\beta^A = \frac{\partial \bar{w}}{\partial \lambda_e^A} \lambda_e^A \tag{9}$$

2.1 Yield condition

We specify a yield function $f(\boldsymbol{\tau})$ with the yield condition $f \leq 0$. Isochoric plasticity implies that f is independent of $\mathrm{tr}(\boldsymbol{\tau})$, so we have

$$f(\boldsymbol{\tau}) = \bar{f}(\mathrm{dev}(\boldsymbol{\tau})) \tag{10}$$

$$\text{where } \mathrm{dev}(\boldsymbol{\tau}) = \boldsymbol{\tau} - \tfrac{1}{3}\,\mathrm{tr}(\boldsymbol{\tau})\mathbb{1} \tag{11}$$

We define $q = -\frac{\partial w}{\partial \alpha}$ to be the conjugate equivalent stress associated with α and use a particular model $f(\boldsymbol{\tau}) = \tilde{f}(\boldsymbol{\tau}, q)$, where $q = \bar{q}(\alpha)$ is used to model the isotropic hardening. As an example, the isotropic von Mises yield function is

$$\tilde{f}(\boldsymbol{\tau}, q) = |\,\mathrm{dev}(\boldsymbol{\tau})| - \sqrt{\frac{2}{3}}(\tau_y - q) \tag{12}$$

where τ_y is the yield stress. If we define the following

$$\boldsymbol{\beta} = [\beta^1, \beta^2, \beta^3]^T \tag{13}$$

$$\mathbf{1} = [1, 1, 1]^T \tag{14}$$

$$\mathrm{Dev}(\boldsymbol{\beta}) = \boldsymbol{\beta} - \tfrac{1}{3}(\boldsymbol{\beta} \cdot \mathbf{1})\mathbf{1} \tag{15}$$

then we can write the yield function $f = \bar{f}(\boldsymbol{\beta}, q)$ as

$$\bar{f}(\boldsymbol{\beta}, q) = |\,\mathrm{Dev}(\boldsymbol{\beta})| - \sqrt{\frac{2}{3}}(\tau_y - q) \tag{16}$$

We could model linear isotropic hardening by defining $q = -K\alpha$.

2.2 Flow rules

Consider the associative flow rules

$$\boldsymbol{d}^p = \gamma \frac{\partial f}{\partial \boldsymbol{\tau}} \tag{17}$$

$$\dot{\alpha} = \gamma \frac{\partial f}{\partial q} \tag{18}$$

where $\gamma \geq 0$ is the plastic multiplier.

Then we have the following relation:

$$\implies \gamma \frac{\partial f}{\partial \boldsymbol{\tau}} : \boldsymbol{\tau} + \gamma \frac{\partial f}{\partial q} q \geq 0 \tag{19}$$

$$\iff \gamma \frac{\partial f}{\partial \boldsymbol{\beta}} \cdot \boldsymbol{\beta} + \gamma \frac{\partial f}{\partial q} q \geq 0 \tag{20}$$

The loading/unloading (Kuhn-Tucker) and consistency conditions give the following relations:

$$\gamma f = 0 \tag{21}$$

$$\gamma \dot{f} = 0 \tag{22}$$

3 Algorithmic integration of the flow rules

We consider the case of displacment loading. The total applied displacement is discretized into pseudo-time steps. We solve for $\boldsymbol{F}$ iteratively within each pseudo-time step, so we have an assumed value for $\boldsymbol{F}_{n+1}$ when updating from t_n to t_{n+1}. We define a trial state by assuming that all deformation between t_n and t_{n+1} is elastic, namely

$$\alpha_{n+1}^{tr} = \alpha_n \tag{23}$$

$$\boldsymbol{b}_{n+1}^{e^{tr}} = \boldsymbol{F}_{n+1}\boldsymbol{C}_n^{p^{-1}}\boldsymbol{F}_{n+1}^T \tag{24}$$

The yield function is evaluated using the trial state. If $f_{n+1}^{tr} < 0$, then the trial state holds and no plastic flow has ocurred. Otherwise, the body has undergone plastic deformation. The flow rules are algorithmically integrated and stress and strain values are updated using the return mapping algorithm, as follows:

$$\alpha_{n+1} = \alpha_{n+1}^{tr} + \gamma_{n+1}\Delta t\Big(\frac{\partial f}{\partial q}\Big)_{n+1} \tag{25}$$

$$\lambda_{e_{n+1}}^{A^2} = \exp\Big(- 2\gamma_{n+1}\Delta t\Big(\frac{\partial f}{\partial \beta^A}\Big)_{n+1}\Big)\lambda_{e_{n+1}}^{A^{tr^2}} \tag{26}$$

$$\boldsymbol{n}_{n+1}^A = \boldsymbol{n}_{n+1}^{A^{tr}} \tag{27}$$

Recall that the spectral decompositions can be used to relate $\boldsymbol{b}^e$ and $\boldsymbol{\tau}$ to the principal stretches and stresses, respectively.

$$\boldsymbol{b}^e = \sum_{A=1}^{3} \lambda_e^{A^2}\boldsymbol{n}^A \otimes \boldsymbol{n}^A \tag{28}$$

$$\boldsymbol{\tau} = \sum_{A=1}^{3} \beta^A \boldsymbol{n}^A \otimes \boldsymbol{n}^A \tag{29}$$

Depending on the yield function f, hardening function q, and strain energy density function w, it may be necessary to iteratively solve for $\gamma_{n+1}\Delta_t$ (which is treated as a single variable) and $\lambda_{e_{n+1}}^A$, $A = 1, 2, 3$, using Equation 26 and the fact that the yield function should be equal to zero:

$$f_{n+1} = 0 \tag{30}$$

4 Residual and jacobian

The element residual and jacobian have the same form as finite strain elasticity, namely

$$r_e = \int_{\Omega_e} \left(P^h_{iJ} \frac{\partial w^h_i}{\partial X_J} - w^h_i f_i \right) dV - \sum_i \int_{\partial \Omega_{e_{T_i}}} w^h_i T_i \, dS \tag{31}$$

$$j_e = \int_{\Omega_e} \left(\frac{\partial w^h_i}{\partial x_j} c^{ep}_{ijkl} \frac{\partial \Delta u^h_k}{\partial x_l} + \frac{\partial w^h_i}{\partial x_j} \tau^h_{jk} \frac{\partial \Delta u^h_i}{\partial x_k} \right) dV \tag{32}$$

However, note that instead of the spatial elastic tangent, we are using the spatial algorithmic elastoplastic tangent $\boldsymbol{c}^{ep}$. Assuming the yield function f can be written as $f(\boldsymbol{\beta}, q) = g(\boldsymbol{\beta}) + h(q)$ for some functions g and h, we can write $\boldsymbol{c}^{ep}$ as the following [1]:

$$\boldsymbol{c}^{ep} = \sum_{A=1}^{3} \sum_{B=1}^{3} a^{ep}_{AB} \boldsymbol{m}^{A^{tr}} \otimes \boldsymbol{m}^{B^{tr}} + 2 \sum_{A=1}^{3} \beta^A \boldsymbol{c}^{A^{tr}} \tag{33}$$

where

$$\boldsymbol{a}^{ep} = \boldsymbol{h} - \frac{\left(1 - \gamma \Delta t \frac{\partial q}{\partial \alpha} \frac{\partial^2 f}{\partial q^2}\right) \left(\boldsymbol{h} \frac{\partial f}{\partial \boldsymbol{\beta}}\right) \otimes \left(\boldsymbol{h} \frac{\partial f}{\partial \boldsymbol{\beta}}\right)}{\left(1 - \gamma \Delta t \frac{\partial q}{\partial \alpha} \frac{\partial^2 f}{\partial q^2}\right) \frac{\partial f}{\partial \boldsymbol{\beta}} \cdot \boldsymbol{h} \frac{\partial f}{\partial \boldsymbol{\beta}} + \frac{\partial q}{\partial \alpha} \left(\frac{\partial f}{\partial q}\right)^2} \tag{34}$$

$$\boldsymbol{h} = \left(\boldsymbol{a}^{e^{-1}} + \gamma \Delta t \frac{\partial^2 f}{\partial \boldsymbol{\beta}^2} \right)^{-1} \tag{35}$$

$$a^e_{AB} = \frac{\partial^2 w}{\partial \lambda^A_e \partial \lambda^B_e} \lambda^A_e \lambda^B_e + \frac{\partial w}{\partial \lambda^A_e} \lambda^A_e \delta_{AB} \tag{36}$$

$$\boldsymbol{m}^{A^{tr}} = \boldsymbol{n}^{A^{tr}} \otimes \boldsymbol{n}^{A^{tr}} \tag{37}$$

$$\begin{aligned} \boldsymbol{c}^{A^{tr}} = \frac{1}{d_A} \Bigg[& \mathbb{I}_{b^{tr}_e} - \boldsymbol{b}^{tr}_e \otimes \boldsymbol{b}^{tr}_e - \frac{\det(\boldsymbol{b}^{tr}_e)}{\lambda^{A^2}_e} \left(\mathbb{I} - (\mathbb{1} - \boldsymbol{m}^{A^{tr}}) \otimes (\mathbb{1} - \boldsymbol{m}^{A^{tr}}) \right) \\ & + \lambda^{A^2}_e \left(\boldsymbol{b}^{tr}_e \otimes \boldsymbol{m}^{A^{tr}} + \boldsymbol{m}^{A^{tr}} \otimes \boldsymbol{b}^{tr}_e + (\mathrm{tr}(\boldsymbol{b}^{tr}_e) - 4\lambda^{A^2}_e) \boldsymbol{m}^{A^{tr}} \otimes \boldsymbol{m}^{A^{tr}} \right) \Bigg] \end{aligned} \tag{38}$$

$$\mathbb{I}^{b^{tr}_e}_{ijkl} = \tfrac{1}{2} (b^{tr}_{e_{ik}} b^{tr}_{e_{jl}} + b^{tr}_{e_{il}} b^{tr}_{e_{jk}}) \tag{39}$$

$$d_A = \frac{\lambda^{A^{tr^2}}_e - \lambda^{B^{tr^2}}_e}{\lambda^{A^{tr^2}}_e - \lambda^{C^{tr^2}}_e}, \text{ with } A, B, C \text{ even permutations of } \{1, 2, 3\} \tag{40}$$

The PRISMS continuum plasticity code also uses an enhanced strain field to prevent element locking, as described in [2].

5 Modifying the PRISMS continuum plasticity deal.II model

5.1 Material parameters and constitutive laws

Most material parameters and constitutive laws can modified in the "materialProperties.json" (or "parameters.h") file, including the Lamé parameters λ and μ, the yield stress τ_y, the coefficient for linear isotropic strain hardening K, the strain energy density function w, and the yield function f. The two functions are specified by name in the .json file. Although there is a place to specify the path to the pfunction header file, it is not currently being read by the code. All pfunction files are assumed to be placed in the "continuumPlasticity/models" folder. There are three strain energy density functions included in the models library: St. Venant-Kirchhoff (stvenkir), compressible neo-hookian (neohook), and quadratic-logarithmic (quadlog). There is only one yield function included in the models library: von Mises (von_mises). Although the hardening function q is not specified in the .json file, there are two hardening functions included in the models library. The default function is a linear hardening function (with coefficient K (linear_hardening), and the second is a nonlinear hardening function based on a crystal plasticity model for copper (Cu_hardening). The hardening function is also specified by name, but it is set in the code (line 194 of continuumPlasticity.h). Note that currently all hardening laws accept K as an input for consistency, but only the linear hardening law uses that value.

It is possible to create your own pfunctions using Integration Tools Writer, similiar to what is shown in the shell script "continuumPlasticity/models/functions.sh." The following input variables/parameters are required:

- Strain energy density function:
  1. first Lamé parameter λ
  2. second Lamé parameter μ
  3. first principle stretch λ_1
  4. second principle stretch λ_2
  5. third principle stretch λ_3
- Yield function:
  1. first principle stress β_1
  2. second principle stress β_2
  3. third principle stress β_3
  4. yield stress τ_y
  5. conjugate stress-like quantity (equivalent plastic stress from hardening function) q
- Hardening function:
  1. Equivalent plastic strain α
  2. strain hardening parameter (coefficient to linear hardening) K

5.2 Boundary conditions

Dirichlet boundary conditions are set in "main.cc". The class "BCFunction" specifies the value of the nonzero Dirichlet conditions and on what nodal degrees-of-freedom they apply. The Dirichlet conditions are applied to specific surfaces in the function "applyDirichletBCs".

5.3 Mesh refinement, basis function order, quadrature order, number of pseudo-time steps (increments), etc.

These values are specified at the top of "main.cc" through macros (or in "parameters.h"). The domain and mesh refinement are defined in the function "mesh" in "main.cc". Quadratic convergence can best be achieved using a direct solver, like mumps or superlu_dist. These can be called from the command line, using a command such as: `mpirun -np 16 ./main -pc_type lu -pc_factor_mat_solver_package mumps`

References

Unless otherwise noted, the information in this document was taken from the ME 605 class taught by Krishna Garikipati, University of Michigan. This material is also treated in the following text:

Simo, J.C. and Hughes, T.J.R., *Computational Inelasicity*, Springer, New York, 2000.

[1] Simo, J.C., "Algorithms for statis and dynamic multiplicative plasticity that preserve the classical return mapping schemes of the infinitesimal theory," *Computer Methods in Applied Mechanics and Engineering*, 99 (1992), 61-112.

[2] Simo, J.C. and Armero, F., "Improved versions of assumed enhanced strain tri-linear elements for 3D finite deformation problems," *Computer Methods in Applied Mechanics and Engineering*, 110 (1993), 359-386.