A Note on Crystal Plasticity Finite Element Method (CPFEM)

Tianju Xue

December 10, 2022

1 Motivation

Crystal plasticity (CP) is a well-developed and important area. As a beginner in this field, I wrote this note to document my understanding of crystal plasticity with a focus on implementation in JAX-FEM.

What you should and should NOT expect from this note?

1. This note is more of a tutorial, not a research document. The phenomenological constitutive models discussed in this note are classic in literature.
2. We focus heavily on implementation, not on physical intuition of the CP models themselves. Particularly, we focus on how to implement CPFEM with JAX.
3. This note assumes that you know the Finite Element Method (FEM), so the FEM part is omitted.

2 Governing equation

The balance of momentum in reference configuration (ignoring inertial term and body force) gives

$$\begin{aligned} \mathrm{Div}\, \boldsymbol{P} &= \boldsymbol{0} \quad \text{in } \Omega, \\ \boldsymbol{u} &= \boldsymbol{u}_D \quad \text{on } \Gamma_D, \\ \boldsymbol{P} \cdot \boldsymbol{n} &= \boldsymbol{t} \quad \text{on } \Gamma_N, \end{aligned} \tag{1}$$

where $\boldsymbol{P}$ is the first Piola-Kirchhoff stress tensor.

3 Crystal plasticity

We follow heavily the MOOSE tutorial on this part, but providing more details and focusing on the use of automatic differentiation feature by JAX.

3.1 Constitutive model

The deformation gradient $\boldsymbol{F}$ is assumed to be multiplicatively decomposed in its elastic and plastic parts [1]:

$$\boldsymbol{F} = \boldsymbol{F}^e \boldsymbol{F}^p. \tag{2}$$

The total plastic velocity gradient can be expressed in terms of the plasticity deformation gradient as

$$\boldsymbol{L}^p = \dot{\boldsymbol{F}}^p (\boldsymbol{F}^p)^{-1}. \tag{3}$$

The elastic Lagrangian strain $\boldsymbol{E}^e$ is defined as

$$\boldsymbol{E}^e = \frac{1}{2}\big(\boldsymbol{F}^{e\top}\boldsymbol{F}^e - \boldsymbol{I}\big). \tag{4}$$

The second Piola-Kirchhoff stress $\boldsymbol{S}$ is given by

$$\boldsymbol{S} = \mathbb{C} : \boldsymbol{E}^e. \tag{5}$$

The Cauchy stress is given by

$$\boldsymbol{\sigma} = \frac{1}{\det(\boldsymbol{F}^e)}\, \boldsymbol{F}^e\, \boldsymbol{S}\, \boldsymbol{F}^{e\top}. \tag{6}$$

The first Piola-Kirchhoff stress $\boldsymbol{P}$ is given by

$$\boldsymbol{P} = \det(\boldsymbol{F})\, \boldsymbol{\sigma}\, \boldsymbol{F}^{-\top}. \tag{7}$$

The plastic velocity gradient $\boldsymbol{L}^p$ is computed as the sum of contributions from all slip systems

$$\boldsymbol{L}^p = \sum_{\alpha} \dot{\gamma}^{\alpha} \boldsymbol{s}_0^{\alpha} \otimes \boldsymbol{m}_0^{\alpha}, \tag{8}$$

where $\dot{\gamma}^{\alpha}$ is the slip rate for slip system α, $\boldsymbol{s}_0^{\alpha}$ and $\boldsymbol{m}_0^{\alpha}$ are unit vectors describing the slip direction and the normal to the slip plane of the slip system in reference configuration. The resolved shear stress is defined as

$$\tau^{\alpha} = \boldsymbol{S} : \boldsymbol{s}_0^{\alpha} \otimes \boldsymbol{m}_0^{\alpha}. \tag{9}$$

The slip rate $\dot{\gamma}^{\alpha}$ is expressed as a power law relationship:

$$\dot{\gamma}^{\alpha} = \dot{\gamma}_0 \left| \frac{\tau^{\alpha}}{g^{\alpha}} \right|^{1/m} \operatorname{sign}(\tau^{\alpha}), \tag{10}$$

where $\dot{\gamma}_0$ is a reference slip rate, g^{α} is the slip resistance (or critical resolved shear stress), and m is the strain rate sensitivity exponent. The rate of slip resistance is given by

$$\dot{g}^{\alpha} = \sum_{\beta} h^{\alpha\beta} |\dot{\gamma}^{\beta}|. \tag{11}$$

Kalidindi et al. [2] self and latent hardening laws gives

$$h^{\alpha\beta} = q^{\alpha\beta}\, h_0 \left| 1 - \frac{g^{\beta}}{g_{\text{sat}}} \right|^{a} \operatorname{sign}\left(1 - \frac{g^{\beta}}{g_{\text{sat}}} \right). \tag{12}$$

Peirce et al. [3] self-hardening law gives

$$h^{\alpha\beta} = q^{\alpha\beta}\, h_0 \operatorname{sech}^2\left(\frac{h_0 \gamma}{g_{\text{sat}} - g_{\text{ini}}} \right), \quad \gamma = \sum_{\eta} \int_0^t |\dot{\gamma}^{\eta}|\, \mathrm{d}t. \tag{13}$$

Here,

$$q^{\alpha\beta} = \begin{cases} 1 & \text{if} \quad \alpha = \beta \\ r & \text{if} \quad \text{otherwise.} \end{cases} \tag{14}$$

Respectively, g_{ini} is the initial slip resistance and g_{sat} is the saturation slip resistance.

3.2 Time discretization

We have introduced the constitutive model in its continuous form. In order to implement the model in an FEM solver, we need to discretize the constitutive equations in time. At current time step $n+1$, we are given the total deformation gradient $\boldsymbol{F}_{n+1}$ (at any quadrature point) and some internal variables from previous time step n , including g_n^α, γ_n^α and $(\boldsymbol{F}_n^p)^{-1}$ (we will see why these internal variables are needed). The goal is to compute the first Piola-Kirchhoff tensor $\boldsymbol{P}_{n+1}$ and $\frac{\partial \boldsymbol{P}_{n+1}}{\partial \boldsymbol{F}_{n+1}}$ so that Eq. (1) can be solved with FEM.

The tricky part about mapping $\boldsymbol{F}_{n+1}$ to $\boldsymbol{P}_{n+1}$ is that you cannot write $\boldsymbol{P}_{n+1}$ explicitly as a function of $\boldsymbol{F}_{n+1}$ (like a hyperelastic model). Instead, $\boldsymbol{P}_{n+1}$ and $\boldsymbol{F}_{n+1}$ are implicitly related, i.e., given $\boldsymbol{F}_{n+1}$, you must solve a set of nonlinear equations to get $\boldsymbol{P}_{n+1}$.

It turns out that $\boldsymbol{S}$ based formulation is easier to solve instead of directly solving for $\boldsymbol{P}$, which is also practiced by MOOSE. The same idea is found in the textbook by Roters et al. [4] (see Page 110). I believe other choices are also possible, e.g., γ based formulation is found here. Still, let us stick to $\boldsymbol{S}$ based formulation. Given $\boldsymbol{F}_{n+1}$, we first solve the following nonlinear equations to get $\boldsymbol{S}_{n+1}$

$$\boldsymbol{R}(\boldsymbol{F}_{n+1}, \boldsymbol{S}_{n+1}) = \boldsymbol{0}, \tag{15}$$

where $\boldsymbol{R} : \mathbb{R}^{\text{dim}\times\text{dim}} \times \mathbb{R}^{\text{dim}\times\text{dim}} \to \mathbb{R}^{\text{dim}\times\text{dim}}$ is the residual function. Once we know $\boldsymbol{S}_{n+1}$, computing $\boldsymbol{P}_{n+1}$ is straightforward. We will focus on explaining this $\boldsymbol{R}$ function in this subsection, while leaving the computation of $\frac{\partial \boldsymbol{P}_{n+1}}{\partial \boldsymbol{F}_{n+1}}$ to the next subsection.

The residual function in Eq. (15) is explicitly expressed as the following

$$\boldsymbol{S}_{n+1} - \mathbb{C} : \frac{1}{2}\left(\boldsymbol{F}_{n+1}^{e\top}\boldsymbol{F}_{n+1}^{e} - \boldsymbol{I}\right) = \boldsymbol{0}, \tag{16}$$

where $\boldsymbol{F}_{n+1}^e$ can be explicitly computed from $\boldsymbol{S}_{n+1}$. Fig. 1 explains the general procedure of computing $\boldsymbol{F}_{n+1}^e$ from $\boldsymbol{S}_{n+1}$.

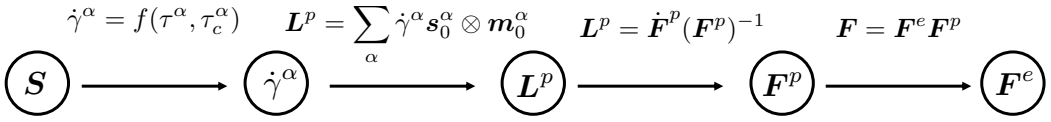

Figure 1: Mapping from $\boldsymbol{S}$ to $\boldsymbol{F}^e$ (or $\boldsymbol{S}_{n+1}$ to $\boldsymbol{F}_{n+1}^e$ if time is discretized).

Fig. 1 still shows the continuous form in time. Let us discretize the equations in time and show how to map from $\boldsymbol{S}_{n+1}$ to $\boldsymbol{F}_{n+1}^e$.

From $\boldsymbol{S}$ to $\dot{\gamma}^\alpha$. We will use the Kalidindi law as an example (others like the Peirce law is similar). Rewrite Eq. (9) as

$$\tau_{n+1}^\alpha = \boldsymbol{S}_{n+1} : \boldsymbol{s}_0^\alpha \otimes \boldsymbol{m}_0^\alpha. \tag{17}$$

Discretize Eq. (10) as

$$\gamma_{n+1}^\alpha - \gamma_n^\alpha = \dot{\gamma}_0 \Delta t \left|\frac{\tau_{n+1}^\alpha}{g_n^\alpha}\right|^{1/m} \text{sign}(\tau_{n+1}^\alpha). \tag{18}$$

Discretize Eq. (11) as

$$g_{n+1}^\alpha - g_n^\alpha = \sum_\beta q^{\alpha\beta} h_0 \left|1 - \frac{g_n^\beta}{g_{\text{sat}}}\right|^a \text{sign}\left(1 - \frac{g_n^\beta}{g_{\text{sat}}}\right) |\gamma_{n+1}^\beta - \gamma_n^\beta|. \tag{19}$$

Here, g_n^α and γ_n^α are assumed to be known from previous step. Now you see the reason why we need to store these internal variables.

From $\dot{\gamma}^\alpha$ to $\boldsymbol{L}^p$. With $\gamma^\alpha_{n+1} - \gamma^\alpha_n$ available, we discretize Eq. (8) as

$$\boldsymbol{L}^p \Delta t = \sum_\alpha (\gamma^\alpha_{n+1} - \gamma^\alpha_n) \boldsymbol{s}^\alpha_0 \otimes \boldsymbol{m}^\alpha_0. \tag{20}$$

From $\boldsymbol{L}^p$ to $\boldsymbol{F}^p$. Discretizing Eq. (3) as

$$(\boldsymbol{F}^p_{n+1})^{-1} = (\boldsymbol{F}^p_n)^{-1}(\boldsymbol{I} - \boldsymbol{L}^p \Delta t). \tag{21}$$

where $\boldsymbol{I}$ is the second order identity tensor and Δt is the time step size. In this case, $(\boldsymbol{F}^p_n)^{-1}$ is known from previous step, and this should be one of the internal variables to store.

From $\boldsymbol{F}^p$ to $\boldsymbol{F}^e$. This step is easy, we have

$$\boldsymbol{F}^e_{n+1} = \boldsymbol{F}_{n+1}(\boldsymbol{F}^p_{n+1})^{-1}. \tag{22}$$

Up to this point, we are able to evaluate the residual function $\boldsymbol{R}$ in Eq. (15) given $\boldsymbol{S}_{n+1}$. Solving the equation requires Newton's method:

$$\boldsymbol{S}^{(k+1)}_{n+1} = \boldsymbol{S}^{(k)}_{n+1} + \theta \Delta \boldsymbol{S}, \quad \left(\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{S}^{(k)}_{n+1}} \right) \Delta \boldsymbol{S} = -\boldsymbol{R}(\boldsymbol{F}_{n+1}, \boldsymbol{S}^{(k)}_{n+1}), \tag{23}$$

where the superscript k denotes the iteration step in Newton's method. The step size θ is usually set to be 1, but in some cases Newton's method will not converge. My experience is that using the line search method to determine a suitable value for θ helps the solver to converge. You may check the JAX-FEM repository to learn the details about how we determine this θ. You may also check MOOSE source code to see their approach.

I want to point out that a fundamental advantage of our JAX-FEM implementation for crystal plasticity is that the Jacobian matrix $\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{S}^{(k)}_{n+1}}$ required in Eq. (23) is evaluated by automatic differentiation. Therefore we will not derive this Jacobian matrix in this note, because the program automatically calculates it for us.

3.3 Implicit differentiation

In order to compute $\frac{\partial \boldsymbol{P}_{n+1}}{\partial \boldsymbol{F}_{n+1}}$ as we mentioned earlier, we need to first evaluate $\frac{\partial \boldsymbol{S}_{n+1}}{\partial \boldsymbol{F}_{n+1}}$. The problem falls into the framework of "implicit differntiation". I highly recommend this NeurIPS (2022) paper by Blondel et al. [5] that has a nice introduction to implicit differentiation and the implementation in JAX. Basically, we take the total derivative of Eq. (15) with respect to $\boldsymbol{F}_{n+1}$ and obtain

$$\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{S}_{n+1}} \frac{\partial \boldsymbol{S}_{n+1}}{\partial \boldsymbol{F}_{n+1}} + \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{F}_{n+1}} = \boldsymbol{0}. \tag{24}$$

Therefore,

$$\frac{\partial \boldsymbol{S}_{n+1}}{\partial \boldsymbol{F}_{n+1}} = -\left(\frac{\partial \boldsymbol{R}}{\partial \boldsymbol{S}_{n+1}} \right)^{-1} \frac{\partial \boldsymbol{R}}{\partial \boldsymbol{F}_{n+1}}. \tag{25}$$

The computation of $\frac{\partial \boldsymbol{P}_{n+1}}{\partial \boldsymbol{F}_{n+1}}$ is trivial once we have access to $\frac{\partial \boldsymbol{S}_{n+1}}{\partial \boldsymbol{F}_{n+1}}$, and we omit the details. Just one thing to notice: We always use automatic differentiation whenever there is a derivative to evaluate.

4 Benchmark

We have implemented CPFEM in JAX-FEM framework. Here, we consider a simple benchmark and show that JAX-FEM produces consistent results with MOOSE. A quasi-static loading condition is imposed on a copper material has an FCC crystal structure. One 8-node hexahedron element is used to show the stress-strain response in Fig. 2. The material parameters are taken from [6].

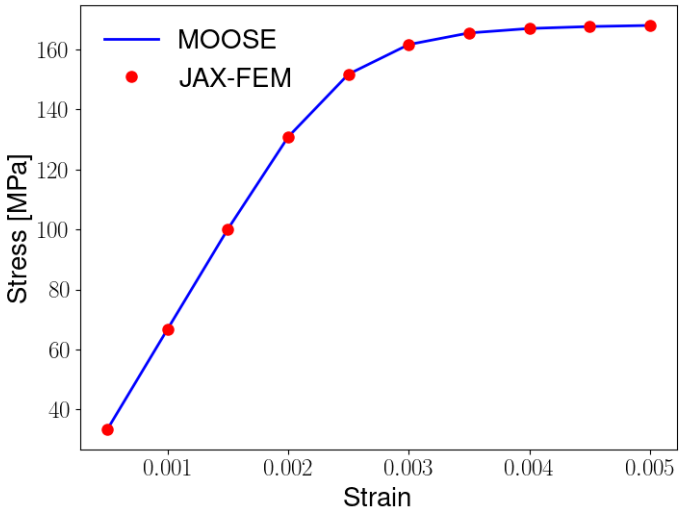

Figure 2: Comparison between JAX-FEM and MOOSE.

5 Remarks

While I am studying the microstructure evolution in metal additive manufacturing, I feel a strong need to have a more in-depth investigation of CP. That is the reason why I implemented CPFEM (almost from scratch) and wrote this note. There are contents that are not covered in this note yet, e.g., polycrystal structure. Also, I did not mention the performance. Usually, CPFEM is quite expensive and we do care about performance. Our code of CPFEM runs on GPU, and we are continuously working on enhancing the performance. Please let me know what you would like to see more on this note, and also let me know if you find any mistake or typo in the note. Thanks.

References

[1] R. J. Asaro, "Crystal plasticity," 1983.

[2] S. R. Kalidindi, C. A. Bronkhorst, and L. Anand, "Crystallographic texture evolution in bulk deformation processing of fcc metals," *Journal of the Mechanics and Physics of Solids*, vol. 40, no. 3, pp. 537–569, 1992.

[3] D. Peirce, R. J. Asaro, and A. Needleman, "Material rate dependence and localized deformation in crystalline solids," *Acta metallurgica*, vol. 31, no. 12, pp. 1951–1976, 1983.

[4] F. Roters, P. Eisenlohr, T. R. Bieler, and D. Raabe, *Crystal plasticity finite element methods: in materials science and engineering*. John Wiley & Sons, 2011.

[5] M. Blondel, Q. Berthet, M. Cuturi, R. Frostig, S. Hoyer, F. Llinares-López, F. Pedregosa, and J.-P. Vert, "Efficient and modular implicit differentiation," *arXiv preprint arXiv:2105.15183*, 2021.

[6] K. Chockalingam, M. Tonks, J. Hales, D. Gaston, P. Millett, and L. Zhang, "Crystal plasticity with jacobian-free newton–krylov," *Computational Mechanics*, vol. 51, no. 5, pp. 617–627, 2013.