Coupling of a multi-GPU accelerated elasto-visco-plastic fast Fourier transform constitutive model with the implicit finite element method

Adnan Eghtesad[a], Kai Germaschewski[b], and Marko Knezevic[a,*]

[a] Department of Mechanical Engineering, University of New Hampshire, Durham, NH 03824, USA
[b] Department of Physics, University of New Hampshire, Durham, NH 03824, USA.

Abstract

This paper presents an implementation of the elasto-visco-plastic fast Fourier transform (EVPFFT) crystal plasticity model in the implicit finite element (FE) method of Abaqus standard through a user material (UMAT) subroutine to provide a constitutive relationship between stress and strain at FE integration points. To facilitate the implicit coupling ensuring fast convergence rates, an analytical Jacobian is provided. The constitutive response at every integration point is obtained by the full-field homogenization over an explicit microstructural cell. The implementation is a parallel computing approach involving multi-core central processing units (CPUs) and graphics processing units (GPUs) for computationally efficient simulations of large plastic deformation of metallic components with arbitrary geometry and loading boundary conditions. To this end, the EVPFFT solver takes advantages of GPU acceleration utilizing Nvidia's high performance computing software development kit (SDK) compiler and compute unified device architecture (CUDA) FFT libraries, while the FE solver leverages the message passing interface (MPI) for parallelism across CPUs. The high-performance hybrid CPU-GPU multi-level framework is referred to as FE-GPU-EVPCUFFT. Simulations of simple compression of Cu and large strain cyclic reversals of dual phase (DP) 590 have been used to benchmark the accuracy of the implementation in predicting the mechanical response and texture evolution. Subsequently, two applications are presented to illustrate the potential and utility of the multi-level simulation strategy: 4-point bending of textured Zr bars, in which the model captures the shape variations as a consequence of texture with respect to the bending plane and another bending of DP1180, in which the model reveals details of spatial micromechanical fields.

Keywords: Crystal plasticity; Finite element method; Microstructures; Parallel computing; FE-GPU-EVPCUFFT

* Corresponding author at: Department of Mechanical Engineering, University of New Hampshire, 33 Academic Way, Kingsbury Hall, W119, Durham, NH 03824, USA. Tel.: +1 603 862 5179; fax: +1 603 862 1865.
E-mail address: marko.knezevic@unh.edu (M. Knezevic).

1. Introduction

Metallic materials experience large plastic shape changes and develop non-linear strain fields in service and during metal forming. These materials are usually polycrystalline and as such exhibit anisotropy owing to the preferred distribution of crystallographic orientations (i.e., texture) as a consequence of prior thermo-mechanical processing history [1-5]. Macroscopic yield surface-based continuum models are extensively used to model metal plasticity [6-8]. However, these models lack the ability to consider the anisotropy at a single-crystal level. Advanced multi-level constitutive relations such as those based on crystal plasticity theory, considering microstructural evolution and the directionality of deformation mechanisms operating at the single crystal level, are being developed [9-14].

Crystal plasticity models have been advanced significantly in the course of last few decades in formulations of Taylor-type upper bound [15-20] and mean-field self-consistent [21-28] models. While these models are computationally efficient and have proven effective in predicting the homogenized flow stress and texture evolution of polycrystalline metals, they lack the ability to explicitly model microstructures and constituent grains. Therefore, these models are unable to provide qualification and quantification of spatial micromechanical fields resulting from grain-to-grain interactions. Nevertheless, these models have been coupled with finite elements to model geometrical shape changes while relaxing the homogenization assumptions with spatial gradients [11, 12, 29-34]. To facilitate spatial i.e. full-field simulations accounting for grain-to-grain interactions, crystal plasticity finite element (CPFE) [35-42] and elasto-visco-plastic fast Fourier transform (EVPFFT) models [43-48] have been developed. The full-field models are more accurate because the constituent grains deform differently according to their crystal orientation and interact with surrounding grains crystallographically and morphologically instead of relying on simpler Taylor iso-strain approximation or self-consistent homogenizations. The major advantage of EVPFFT over CPFE is in its computational efficiency, especially because of its suitability for high-performance parallel implementations using multi-core central processing units (CPUs) and graphics processing units (GPUs) [49-55]. The present paper is concerned with the coupling of EVPFFT with the implicit finite element method (FEM) inside a user material (UMAT) subroutine to provide a microstructure-sensitive constitutive response at the meso-scale for each integration point within a boundary value problem solved with the FEM at the macro-scale. The resulting constitutive behavior is that of a *full-field* EVPFFT material, implemented in

the implicit *full-field* FE code Abaqus. As a result of the two spatial levels, the implementation is a multi-level *full-field*^{2} computational plasticity framework.

Düsseldorf Advanced Material Simulation Kit (DAMASK) has recently been advanced to incorporate a version of a spectral crystal plasticity full-field model based on FFTs for implicit FE solvers [56, 57]. The version utilized the finite strain theory and a Piola-Kirchhoff stress-based constitutive relationship. The framework chains the transformation from the Piola-Kirchhoff stress to obtain a tangent Jacobian matrix appropriate to the Jaumann rate of Cauchy stress required by the Abaqus FE solver. The implementation provided approximately a linear convergence. As presented, the implementation is limited to CPU-only computations not taking advantages of advanced parallel computing and hardware. The present development is aimed at improving numerical aspects by adapting EVPFFT and deriving an analytical Jacobian to achieve better convergence rates. Moreover, the objective is a high-performance parallel computing implementation involving a hybrid of CPUs and GPUs.

Specifically, the first high performance computing (HPC) implementation of the GPU accelerated EVPFFT crystal plasticity model [50] into the implicit FE framework [58] is developed and presented. The obvious motivation for the development is the superior ability of the EVPFFT formulation compared with any other crystal plasticity formulation to capture the strong explicit microstructure-induced gradients and anisotropic hardening in a computationally efficient manner. A key novel aspect of the coupled implementation is the tangent stiffness matrix (Jacobian) for the nonlinear FE solver obtained analytically. The tangent stiffness matrix is obtained primarily as a function of the local tangent moduli already available as part of the non-linear EVPFFT numerical scheme and the elastic stiffness tensor. The convergence behavior is examined and presented in function of time increment and resolution.

The EVPFFT UMAT subroutine is designed to run on single or multi-GPU hardware, while the FE model runs on multiple processors leveraging MPI. The framework can take advantages of both Intel and Nvidia HPC software development kit (SDK) compiler to meet the requirements of performance portability and platform compatibility leveraging OpenACC as well as the compute unified device architecture (CUDA) FFT libraries facilitated by OpenACC-CUDA interoperability [50]. The implementation is referred to as full-field^2 FE-GPU-EVPCUFFT. The performance improvements of the hybrid CPU-GPU implementation (FE-GPU-EVPCUFFT) running on single

and multiple GPUs is compared with the CPU-only model (FE-CPU-EVPFFTW) which utilizes CPUs and the FFTW library.

Accuracy of the implementation is demonstrated using several benchmarks, a simple compression of polycrystalline Cu and load reversals of an advance high strength (AHSS) dual phase (DP) 590 steel, comparing the simulation results of the FE-GPU-EVPCUFFT model, the standalone (SA) EVPFFT solver, and experimental data. Additionally, compression of a Cu single crystal is simulated to predicts extreme anisotropy and shape changes. Excellent agreement is achieved because the EVPFFT accounts for the elastic anisotropy, dislocation density-based thermally activated hardening, development of back-stress, and non-Schmid effects. Subsequently, two applications are presented to illustrate the potential and versatility of the multi-level simulation strategy: 4-point bending of textured Zr bars, in which the model captures the shape variations as a consequence of texture with respect to the bending plane and bending of a cantilever beam of DP1180, in which the model reveals details of spatial micromechanical fields.

2. Coupling of EVPFFT and FEM

Figure 1 illustrates a graphical abstract of the multi-level FE-EVPFFT model implementation as a UMAT in Abaqus. In the implementation, every integration point within the finite element mesh (grid) contains the information of the overall constitutive response of the microstructural cell also called a representative volume element (RVE) obtained by the homogenization over the FFT voxels. The RVE is an explicit grain structure consisting of voxels, which incorporate a crystal orientation and associated slip systems.

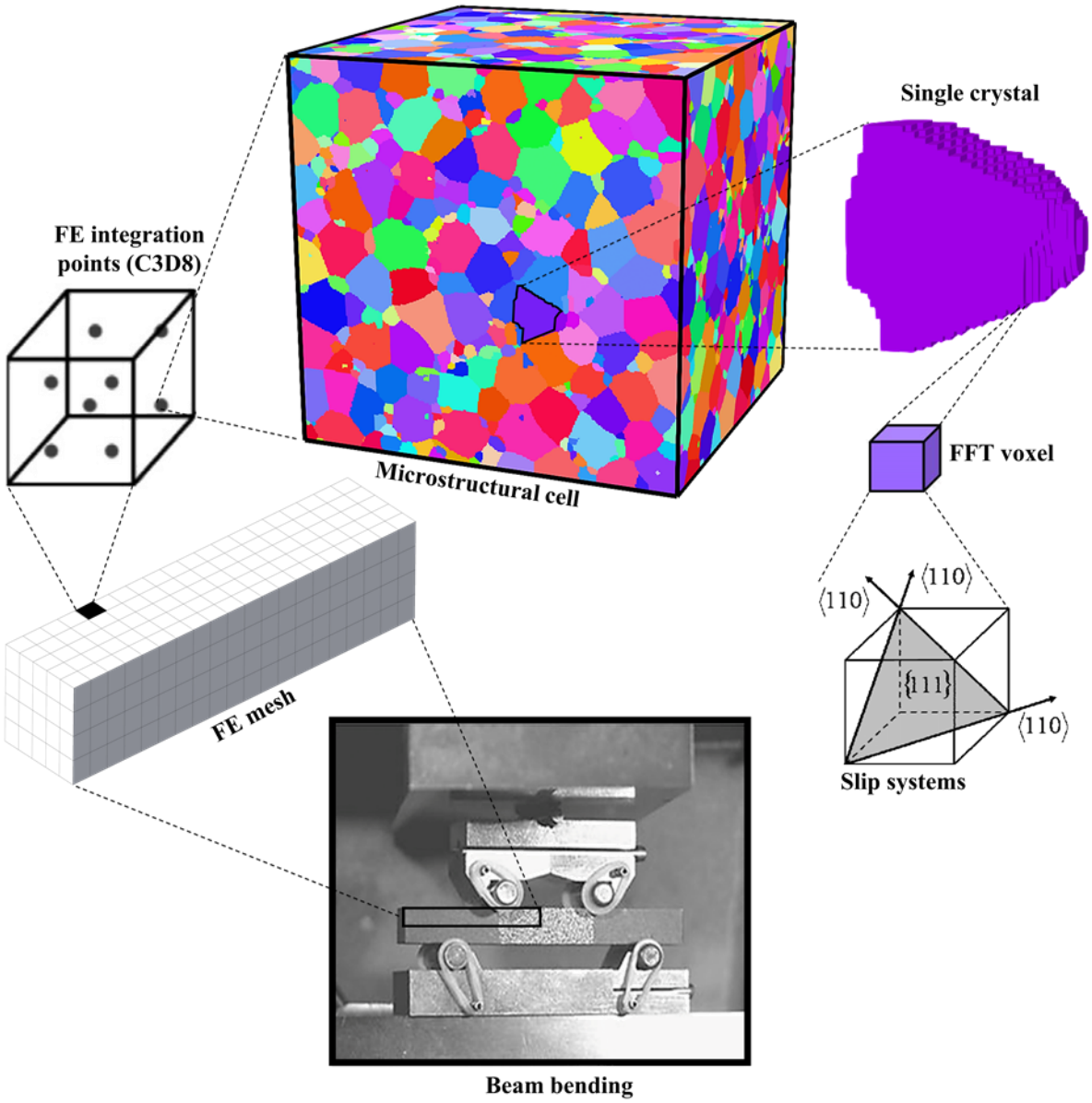


Fig. 1. A graphical illustration of the full-field[2] multi-level FE-EVPFFT modeling strategy linking sub-grain physics of deformation at a voxel-level to a grain-level to a microstructural cell-level to an FE element-level and, finally, to a component-level.

The main equations pertaining to the standard SA EVPFFT solver are given in appendix A for reference and completeness. This section presents equations pertaining to the implementation of the EVPFFT model into the implicit finite elements. In the notation that follows, tensors are denoted by non-italic bold letters, while scalars and tensor components are italic and not bold. Symbols $\cdot$ and $\otimes$ are used to denote contracted (dot product) and uncontracted (tensor product) operators.

The Abaqus FE solver satisfies the stress equilibrium and strain compatibility through the FE weak formulation. To this end, the solver breaks down the applied displacement/load into increments and the equilibrium state of the solution is obtained using an iterative Newton's procedure at the end of each increment. The corresponding principal of virtual work linearized FE equilibrium equation for each element is

$$\left(\int_V \bar{\mathbf{B}}^{\mathrm{T}}\mathbf{J}\bar{\mathbf{B}}dV\right)\Delta\mathbf{U} = {}^{\tau}\mathbf{f} - \int_V \bar{\mathbf{B}}\boldsymbol{\Sigma}dV, \tag{1}$$

where, $\bar{\mathbf{B}}, \mathbf{J}, \Delta\mathbf{U}, \boldsymbol{\Sigma}$ and $\mathbf{f}$ represent the strain-displacement matrix, Jacobian, displacement increment, macroscopic Cauchy stress and applied forces, respectively. At every UMAT call from Abaqus solver, the strain increment $\Delta\mathbf{E} = \mathbf{D}\Delta t$ (calculated using the stretching tensor $\mathbf{D}$ in Abaqus), rotation increment $\Delta\mathbf{R}$, the solution dependent state variables, and time increment Δt are provided to the UMAT subroutine. The UMAT subroutine provides the updated stress, the Jacobian matrix, and updated state variables at every interrogation by the FE solver. It is important to note that while the calculated Jacobian does not affect the accuracy of the solution, it determines the convergence rate of the Newton's iterative solver involved in the implicit FE solver.

Appendix B shows the UMAT subroutine interface provided by Abaqus to incorporate user defined constitutive laws in the Fortran programing language. The variables relevant for the present coupling are STATEV(NSTATV), DDSDDE (NTENS, NTENS), STRESS(NTENS), DSTRAN (NTENS) and DROT (3,3) standing for state-variables, Jacobian, stress, strain increment, and rotation increment, respectively. Strain increment, rotation increment, stress, and state variable at the beginning of the time increment are input for a UMAT call, the updated stress, updated state variables, and Jacobian are returned from the UMAT to the FE solver.

2.1 Spin tensors

The EVPFFT solver is designed to operate under applied stretching tensor, $\mathbf{D}$, calculated by dividing the strain increment $\Delta\mathbf{E}$ by the time increment Δt. The macroscopically imposed spin, $\mathbf{W}$, is calculated using the rotation increment tensor provided by Abaqus

$$\mathbf{W} = \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix}, \tag{2-1}$$

where w_i denote the components of angular velocity as a function of rotation unit vectors, n_i, rotation angle increment, $\Delta\alpha$, and the rotation increment tensor, $\Delta\mathbf{R}$, as follows

$$w_i = \dot{\alpha} n_i = \frac{\Delta\alpha}{\Delta t} n_i \; ; \; n_i = -\in_{ijk} \Delta R_{jk} \; ; \Delta\alpha = \cos^{-1}\left(\frac{1}{2}\left[trace(\Delta\mathbf{R}) - 1\right]\right). \tag{2-2}$$

Having the imposed spin, the lattice spin used to update the crystal orientations can be calculated

$$\mathbf{W}^*(\mathbf{x}) = \mathbf{W} - \mathbf{W}^p(\mathbf{x}), \tag{2-3}$$

where the plastic spin $\mathbf{W}^p(\mathbf{x})$ is

$$\mathbf{W}^p(\mathbf{x}) = \sum_{s=1}^{N} \mathbf{\Lambda}^s(\mathbf{x}) \dot{\gamma}^s(\mathbf{x}); \; \mathbf{\Lambda}^s(\mathbf{x}) = \frac{1}{2}\left(\mathbf{b}^s \otimes \mathbf{n}^s - \mathbf{n}^s \otimes \mathbf{b}^s\right), \tag{3}$$

where $\mathbf{\Lambda}^s$ is the antisymmetric Schmid tensor and $\dot{\gamma}^s(\mathbf{x})$ is the shear rate on the slip system *s* of the total number of available slip systems, *N*.

2.2 Derivation of Jacobian

The evaluation of the appropriate Jacobian plays a significant role in determining the order of convergence. The Jacobian is used by the FE solver to iteratively guess the nodal displacement field in order to satisfy the equilibrium requirement in the current time increment from *t* to $t + \Delta t$. A more accurate Jacobian facilitates a closer guess to the actual solution. The Jacobian matrix can be defined as a continuum operator $\frac{\partial \mathbf{\Sigma}}{\partial \mathbf{E}}$ or as a consistent tangent operator $\frac{\partial \Delta\mathbf{\Sigma}}{\partial \Delta\mathbf{E}}$. The present plasticity problem is formulated incrementally through the use of the Jaumann rate as the objective rate with the consistent tangent operator. Therefore, the constitutive equation at the crystal level in EVPFFT (appendix A) is appropriately adjusted to a generic form of the Jaumann stress rate, $\overset{\nabla}{\boldsymbol{\sigma}}(\mathbf{x}) = \mathbf{C}(\mathbf{x})\left(\dot{\boldsymbol{\varepsilon}}(\mathbf{x}) - \dot{\boldsymbol{\varepsilon}}^p(\mathbf{x})\right)$ [31, 59-62]

$$\dot{\boldsymbol{\sigma}}(\mathbf{x}) = \mathbf{C}(\mathbf{x})\dot{\boldsymbol{\varepsilon}}^e(\mathbf{x}) + \mathbf{W}^*(\mathbf{x})\boldsymbol{\sigma}(\mathbf{x}) - \boldsymbol{\sigma}(\mathbf{x})\mathbf{W}^*(\mathbf{x}) = \mathbf{C}(\mathbf{x})\left(\dot{\boldsymbol{\varepsilon}}(\mathbf{x}) - \dot{\boldsymbol{\varepsilon}}^p(\mathbf{x})\right) + \mathbf{W}^*(\mathbf{x})\boldsymbol{\sigma}(\mathbf{x}) - \boldsymbol{\sigma}(\mathbf{x})\mathbf{W}^*(\mathbf{x}) \tag{4-1}$$

where $\mathbf{W}^*$ is the lattice spin, $\dot{\boldsymbol{\sigma}}$ is the rate of Cauchy stress, and $\mathbf{C}$ is the elastic stiffness tensor. Integrating Eq. (4-1) in the fixed coordinate system from time *t* to $t + \Delta t$ leads to

$$\Delta\boldsymbol{\sigma}(\mathbf{x}) = \mathbf{C}(\mathbf{x})\left(\Delta\varepsilon(\mathbf{x}) - \Delta\varepsilon^p(\mathbf{x},\boldsymbol{\sigma})\right) + \Delta\mathbf{W}^*(\mathbf{x})\boldsymbol{\sigma}(\mathbf{x}) - \boldsymbol{\sigma}(\mathbf{x})\Delta\mathbf{W}^*(\mathbf{x}) \tag{4-2}$$

The above incremental form meets the requirement for the FE integration scheme in Abaqus to

update stress using $\boldsymbol{\Sigma}^{t+\Delta t} = \boldsymbol{\Sigma}^{t} + \langle \Delta\boldsymbol{\sigma}(\mathbf{x}) \rangle$. Note that $\Delta\mathbf{W}^{*}$ is $\mathbf{W}^{*}\Delta t$. We would like to point out that the field and state variables (such as texture) are all expressed in the global frame at the current configuration. Therefore, our implementation is consistent with the default treatment of rotations in Abaqus.

The superscript $\square^{t+\Delta t}$ implies the quantity at the current time increment and is dropped from all tensors for brevity. The derivative of the increment in Cauchy stress with respect to strain increment is

$$\frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} = \mathbf{C}(\mathbf{x}) \underbrace{\frac{\partial \Delta\varepsilon(\mathbf{x})}{\partial \Delta\varepsilon(\mathbf{x})}}_{\mathbf{I}} - \mathbf{C}(\mathbf{x}) \frac{\partial \left(\Delta\boldsymbol{\varepsilon}^{p}(\mathbf{x},\boldsymbol{\sigma}) \right)}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} + \frac{\partial \left(\Delta\mathbf{W}^{*}(\mathbf{x}) \left[\boldsymbol{\sigma}^{t}(\mathbf{x}) + \Delta\boldsymbol{\sigma}(\mathbf{x}) \right] \right)}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} - \frac{\partial \left(\left[\boldsymbol{\sigma}^{t}(\mathbf{x}) + \Delta\boldsymbol{\sigma}(\mathbf{x}) \right] \Delta\mathbf{W}^{*}(\mathbf{x}) \right)}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})}$$

$$\frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} = \mathbf{C}(\mathbf{x}) - \mathbf{C}(\mathbf{x}) \left[\frac{\partial \left(\Delta\boldsymbol{\varepsilon}^{p}(\mathbf{x},\boldsymbol{\sigma}) \right)}{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})} \frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \right] +$$

$$\frac{\partial \left(\Delta\mathbf{W}^{*}(\mathbf{x})\boldsymbol{\sigma}^{t}(\mathbf{x}) + \Delta\mathbf{W}^{*}(\mathbf{x})\Delta\boldsymbol{\sigma}(\mathbf{x}) \right)}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} - \frac{\partial \left(\boldsymbol{\sigma}^{t}(\mathbf{x})\Delta\mathbf{W}^{*}(\mathbf{x}) + \Delta\boldsymbol{\sigma}(\mathbf{x})\Delta\mathbf{W}^{*}(\mathbf{x}) \right)}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})}$$

$$\frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} = \mathbf{C}(\mathbf{x}) - \mathbf{C}(\mathbf{x}) \left[\frac{\partial \left(\Delta\boldsymbol{\varepsilon}^{p}(\mathbf{x},\boldsymbol{\sigma}) \right)}{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})} \frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \right] \tag{5-1}$$

$$+ \left(\frac{\partial \Delta\mathbf{W}^{*}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \boldsymbol{\sigma}^{t}(\mathbf{x}) + \Delta\mathbf{W}^{*}(\mathbf{x}) \underbrace{\frac{\partial \boldsymbol{\sigma}^{t}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})}}_{0} + \frac{\partial \Delta\mathbf{W}^{*}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \Delta\boldsymbol{\sigma}(\mathbf{x}) + \Delta\mathbf{W}^{*}(\mathbf{x}) \frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \right)$$

$$- \left(\underbrace{\frac{\partial \boldsymbol{\sigma}^{t}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})}}_{0} \Delta\mathbf{W}^{*}(\mathbf{x}) + \boldsymbol{\sigma}^{t}(\mathbf{x}) \frac{\partial \Delta\mathbf{W}^{*}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} + \frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \Delta\mathbf{W}^{*}(\mathbf{x}) + \Delta\boldsymbol{\sigma}(\mathbf{x}) \frac{\partial \Delta\mathbf{W}^{*}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \right).$$

Note that $\frac{\partial \boldsymbol{\sigma}^{t}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} = 0$ since the stress in crystals at the beginning of the time increment is constant

$$\frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} = \mathbf{C}(\mathbf{x}) - \mathbf{C}(\mathbf{x}) \left[\frac{\partial \left(\Delta\boldsymbol{\varepsilon}^{p}(\mathbf{x},\boldsymbol{\sigma}) \right)}{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})} \frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \right]$$

$$+ \left(\left[\frac{\partial \Delta\mathbf{W}^{*}(\mathbf{x})}{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})} \frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \right] \boldsymbol{\sigma}^{t}(\mathbf{x}) + \left[\frac{\partial \Delta\mathbf{W}^{*}(\mathbf{x})}{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})} \frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \right] \Delta\boldsymbol{\sigma}(\mathbf{x}) + \Delta\mathbf{W}^{*}(\mathbf{x}) \frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \right)$$

$$- \left(\boldsymbol{\sigma}^{t}(\mathbf{x}) \left[\frac{\partial \Delta\mathbf{W}^{*}(\mathbf{x})}{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})} \frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \right] + \frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \Delta\mathbf{W}^{*}(\mathbf{x}) + \Delta\boldsymbol{\sigma}(\mathbf{x}) \left[\frac{\partial \Delta\mathbf{W}^{*}(\mathbf{x})}{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})} \frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \right] \right) \tag{5-2}$$

$$\frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} = \mathbf{C}(\mathbf{x}) - \mathbf{C}(\mathbf{x}) \left[\frac{\partial \left(\Delta\boldsymbol{\varepsilon}^{p}(\mathbf{x},\boldsymbol{\sigma}) \right)}{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})} \frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \right]$$

$$+ \left(\left[\frac{\partial \Delta\mathbf{W}^{*}(\mathbf{x})}{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})} \frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \right] \boldsymbol{\sigma}(\mathbf{x}) + \Delta\mathbf{W}^{*}(\mathbf{x}) \frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \right) - \left(\boldsymbol{\sigma}(\mathbf{x}) \left[\frac{\partial \Delta\mathbf{W}^{*}(\mathbf{x})}{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})} \frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \right] + \frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \Delta\mathbf{W}^{*}(\mathbf{x}) \right).$$

Since strain and stress tensors are symmetric, the terms above including $\Delta\mathbf{W}^*(\mathbf{x})$ are antisymmetric owing to $\Delta\mathbf{W}^*(\mathbf{x})$ being antisymmetric $\left[\left(\Delta\mathbf{W}^*(\mathbf{x})\right)^T = -\Delta\mathbf{W}^*(\mathbf{x})\right]$. This allows us to write the above equations while keeping the operation of $\frac{\partial\Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial\Delta\boldsymbol{\varepsilon}(\mathbf{x})}$ consistently from the right, as follows

$$\begin{aligned}&\frac{\partial\Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial\Delta\boldsymbol{\varepsilon}(\mathbf{x})} = \mathbf{C}(\mathbf{x}) - \mathbf{C}(\mathbf{x})\left[\frac{\partial\left(\Delta\boldsymbol{\varepsilon}^p(\mathbf{x},\boldsymbol{\sigma})\right)}{\partial\Delta\boldsymbol{\sigma}(\mathbf{x})}\frac{\partial\Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial\Delta\boldsymbol{\varepsilon}(\mathbf{x})}\right] \\ &-\left\{\left(\boldsymbol{\sigma}(\mathbf{x})\left[\frac{\partial\Delta\mathbf{W}^*(\mathbf{x})}{\partial\Delta\boldsymbol{\sigma}(\mathbf{x})}\frac{\partial\Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial\Delta\boldsymbol{\varepsilon}(\mathbf{x})}\right]\right)+\left(\boldsymbol{\sigma}(\mathbf{x})\left[\frac{\partial\Delta\mathbf{W}^*(\mathbf{x})}{\partial\Delta\boldsymbol{\sigma}(\mathbf{x})}\frac{\partial\Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial\Delta\boldsymbol{\varepsilon}(\mathbf{x})}\right]\right)^T\right\}+ \\ &\left(\Delta\mathbf{W}^*(\mathbf{x})\frac{\partial\Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial\Delta\boldsymbol{\varepsilon}(\mathbf{x})}\right)+\left(\Delta\mathbf{W}^*(\mathbf{x})\frac{\partial\Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial\Delta\boldsymbol{\varepsilon}(\mathbf{x})}\right)^T.\end{aligned} \tag{5-3}$$

Since for any tensor $\square$, $\left(\square+\square^T\right) = 2sym\left(\square\right)$ we can rewrite the above equation in a more condensed form as

$$\begin{aligned}&\frac{\partial\Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial\Delta\boldsymbol{\varepsilon}(\mathbf{x})} = \mathbf{C}(\mathbf{x}) - \mathbf{C}(\mathbf{x})\left[\frac{\partial\left(\Delta\boldsymbol{\varepsilon}^p(\mathbf{x},\boldsymbol{\sigma})\right)}{\partial\Delta\boldsymbol{\sigma}(\mathbf{x})}\frac{\partial\Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial\Delta\boldsymbol{\varepsilon}(\mathbf{x})}\right]+ \\ &2sym\left(-\boldsymbol{\sigma}(\mathbf{x})\left[\frac{\partial\Delta\mathbf{W}^*(\mathbf{x})}{\partial\Delta\boldsymbol{\sigma}(\mathbf{x})}\frac{\partial\Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial\Delta\boldsymbol{\varepsilon}(\mathbf{x})}\right]+\Delta\mathbf{W}^*(\mathbf{x})\frac{\partial\Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial\Delta\boldsymbol{\varepsilon}(\mathbf{x})}\right).\end{aligned} \tag{5-4}$$

We also provide the indicial form of the above equation

$$\begin{aligned}&\frac{\partial\Delta\sigma_{ij}(\mathbf{x})}{\partial\Delta\varepsilon_{kl}(\mathbf{x})} = C_{ijkl}(\mathbf{x}) - C_{ijmn}(\mathbf{x})\frac{\partial\left(\Delta\varepsilon^p_{mn}(\mathbf{x},\boldsymbol{\sigma})\right)}{\partial\Delta\sigma_{pq}(\mathbf{x})}\frac{\partial\Delta\sigma_{pq}(\mathbf{x})}{\partial\Delta\varepsilon_{kl}(\mathbf{x})}+ \\ &2sym\left(-\sigma_{ip}(\mathbf{x})\frac{\partial\Delta W^*_{pj}(\mathbf{x})}{\partial\Delta\sigma_{mn}(\mathbf{x})}\frac{\partial\Delta\sigma_{mn}(\mathbf{x})}{\partial\Delta\varepsilon_{kl}(\mathbf{x})}+\Delta W^*_{ip}(\mathbf{x})\frac{\partial\Delta\sigma_{pj}(\mathbf{x})}{\partial\Delta\varepsilon_{kl}(\mathbf{x})}\right).\end{aligned} \tag{5-5}$$

After incorporating the local tangent moduli, $\frac{\partial\Delta\boldsymbol{\varepsilon}^p}{\partial\Delta\boldsymbol{\sigma}}$, already available as part of the non-linear EVPFFT numerical scheme (see appendix A) and similarly deriving $\frac{\partial\Delta\mathbf{W}^*}{\partial\Delta\boldsymbol{\sigma}} = \frac{-\partial\Delta\mathbf{W}^p}{\partial\Delta\boldsymbol{\sigma}}$, we obtain

$$\frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} = \mathbf{C}(\mathbf{x}) - n\Delta\gamma_0 \mathbf{C}(\mathbf{x}) \sum_{s=1}^{N} \frac{\mathbf{P}^s(\mathbf{x}) \otimes \mathbf{P}^s(\mathbf{x})}{{\tau_c}^s(\boldsymbol{\sigma}(\mathbf{x}))} \left(\frac{\mathbf{P}^s(\mathbf{x}) \cdot \boldsymbol{\sigma}(\mathbf{x}) - {\tau_{bs}}^s}{{\tau_c}^s(\boldsymbol{\sigma}(\mathbf{x}))} \right)^{n-1} \frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})}$$
$$+2\Delta\gamma_0 sym \left(\begin{array}{l} \boldsymbol{\sigma}(\mathbf{x}) \left[\left\{ n \sum_{s=1}^{N} \frac{\boldsymbol{\Lambda}^s(\mathbf{x}) \otimes \mathbf{P}^s(\mathbf{x})}{{\tau_c}^s(\boldsymbol{\sigma}(\mathbf{x}))} \left(\frac{\mathbf{P}^s(\mathbf{x}) \cdot \boldsymbol{\sigma}(\mathbf{x}) - {\tau_{bs}}^s}{{\tau_c}^s(\boldsymbol{\sigma}(\mathbf{x}))} \right)^{n-1} \right\} \frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \right] - \\ \left\{ \sum_{s=1}^{N} \boldsymbol{\Lambda}^s(\mathbf{x}) \left(\frac{\mathbf{P}^s(\mathbf{x}) \cdot \boldsymbol{\sigma}(\mathbf{x}) - {\tau_{bs}}^s}{{\tau_c}^s(\boldsymbol{\sigma}(\mathbf{x}))} \right)^{n} \right\} \frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \end{array} \right). \quad (5\text{-}6)$$

Using Voigt notation (e.g., $\sigma_{ij} \to \sigma_k$, $\varepsilon_{ij} \to \varepsilon_k$, $C_{ijkl} \to C_{kl}$; $k,l = 1,6$), the above equation generates a system of 36 linear equations and 36 unknowns which is solved for the local Jacobian $\frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})}$ using Gauss-Jordan elimination (GJE). This is accomplished by grouping the terms that include the term $\frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})}$ on one side ($\mathbf{A}\frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} = \mathbf{B}$) and the matrix inversion ($\frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} = \mathbf{A}^{-1}\mathbf{B}$). The overall consistent tangent stiffness is then obtained by applying the homogenization (i.e., the volume averaging) over voxels of a given RVE, $\mathbf{J} = \frac{\partial \Delta\boldsymbol{\Sigma}}{\partial \Delta\mathbf{E}} = \left\langle \frac{\partial \Delta\boldsymbol{\sigma}(\mathbf{x})}{\partial \Delta\boldsymbol{\varepsilon}(\mathbf{x})} \right\rangle$. The derived macroscopic average Jacobian is an approximation because the volume average of local partial derivatives does not include explicit couplings over the microstructure. Given that the macro-scale stress is the volume average of local stresses, the Jacobian does not include a chain rule contribution for the change in local deformation increment with macro-scale deformation increment. The consequence of such approximation in the Jacobian is decreased rate/speed of convergence, while solution accuracy is not affected. Nevertheless, the enhanced Jacobian without the approximation would contain extra calculations, which would extend the overall computational time.

2.3 Solution recovery and identification of state variables

State variables, STATEV(NSTATV), need to be available to recover the fields at the beginning of each time increment at each integration point. The number of state variables plays a role in computational efficiency and memory usage. Simulations of large models where the high-resolution FE model embeds high-resolution RVEs are memory demanding. State variables that need to be saved per voxel in the FE-EVPFFT implementation include total strain, $\boldsymbol{\varepsilon}(\mathbf{x})$, Cauchy

stress, $\boldsymbol{\sigma}(\mathbf{x})$, plastic strain, $\boldsymbol{\varepsilon}^{p,t}(\mathbf{x})$, crystallographic orientation, $\mathbf{g}(\varphi_1, \Phi, \varphi_2)$, debris dislocation density, ρ_{deb}, forward and reversible dislocation densities, $\rho_{forw}^{s}, \rho_{rev}^{s+}, \rho_{rev}^{s-}$, and back-stress, $\tau_{bs}{}^{s}(\mathbf{x})$. Note that the last four are per slip system, *N*. Considering that these variables are per voxel, the total storage depends on the RVE size as

$$\text{NSTATV} = N_1 \times N_2 \times N_3 \times (22 + 4 \times N), \tag{6}$$

where N_1, N_2, N_3 represent the number of voxels in X, Y, Z. State variables are read as input at the beginning of a time increment, updated in UMAT, and passed back to Abaqus for storage at the end of the time increment for the next increment.

3. Nvidia HPC compiler and hybrid CPU-GPU FE-GPU-EVPCUFFT

Abaqus' custom UMAT subroutines are usually compiled and linked by Intel Fortran compiler, which does not support GPU programming. To the authors' knowledge, none of the crystal plasticity UMAT developments in the literature so far are able to utilize GPUs. The Nvidia HPC SDK compiler suite, which is based on the former Portland Group, Inc. (PGI) compilers, supports both CUDA [63, 64] and OpenACC [65, 66] to facilitate GPU programming in Fortran and C/C++.

To ensure performance portability, CUDA-OpenACC interoperability [50] is used in the SA EVPFFT termed here as SA-GPU-EVPCUFFT. As a result, the SA-GPU-EVPCUFFT solver is capable of running high-resolution crystal plasticity simulations with dramatic reduction in computational time. For instance, utilizing a single Nvidia Tesla V100 GPU, up to 43x speed up is obtained relative to the original EVPFFT utilizing FOURN for the FFT calculations [48, 50]. Moreover, the multi-GPU performance of the solver lies within just about perfect scalability on distributed nodes of supercomputers [50].

To enable running our GPU accelerated UMAT linked with the FEM, the Abaqus environment file "abaqus_v6.env" is modified to include the Nvidia compiler and linker flags, enabling Abaqus FE solver to link with the Nvidia HPC compiler instead of Intel. Note that switching back to Intel compiler is still possible by changing the compiler flags in the environment file and therefore we are able to run our simulations leveraging both Intel and Nvidia compilers. Appendix C presents a schematic of the environment file where the Nvidia complier and its corresponding flags replace the Intel compiler to facilitate GPU utilization. The overall schematic

and the detailed flow-chart of the hybrid FE-GPU-EVPCUFFT are shown in Fig. 2. Abaqus FE solver utilizes CPUs and MPI for domain decomposition at the mesh-level, while the underlying voxelized microstructural RVE embedded in each integration point of an element runs on GPU. Fig. 3 illustrates the GPU utilization and the details pertaining to performance profiling of the GPU accelerated UMAT called as an internal process by the ABAQUS standard solver. The profile data is obtained by Nvidia visual profiler (nvvp) v2020. Note that the GPU utilized to report this figure is a Nvidia GeForce RTX 2060. Examining the performance benchmark results indicate proper utilization of GPU hardware i.e. the GPU is kept busy during the simulation. Furthermore, the amount of time spent to transfer the data from GPU to CPU and vice versa [indicated by narrow brown bars – CPU-GPU data copy] is small [orders of milli seconds] w.r.t the compute time where calculations are performed on GPU [wide green bars]. The performance benchmarks provided later in the text are reported for more advanced Nvidia Tesla GPUs.

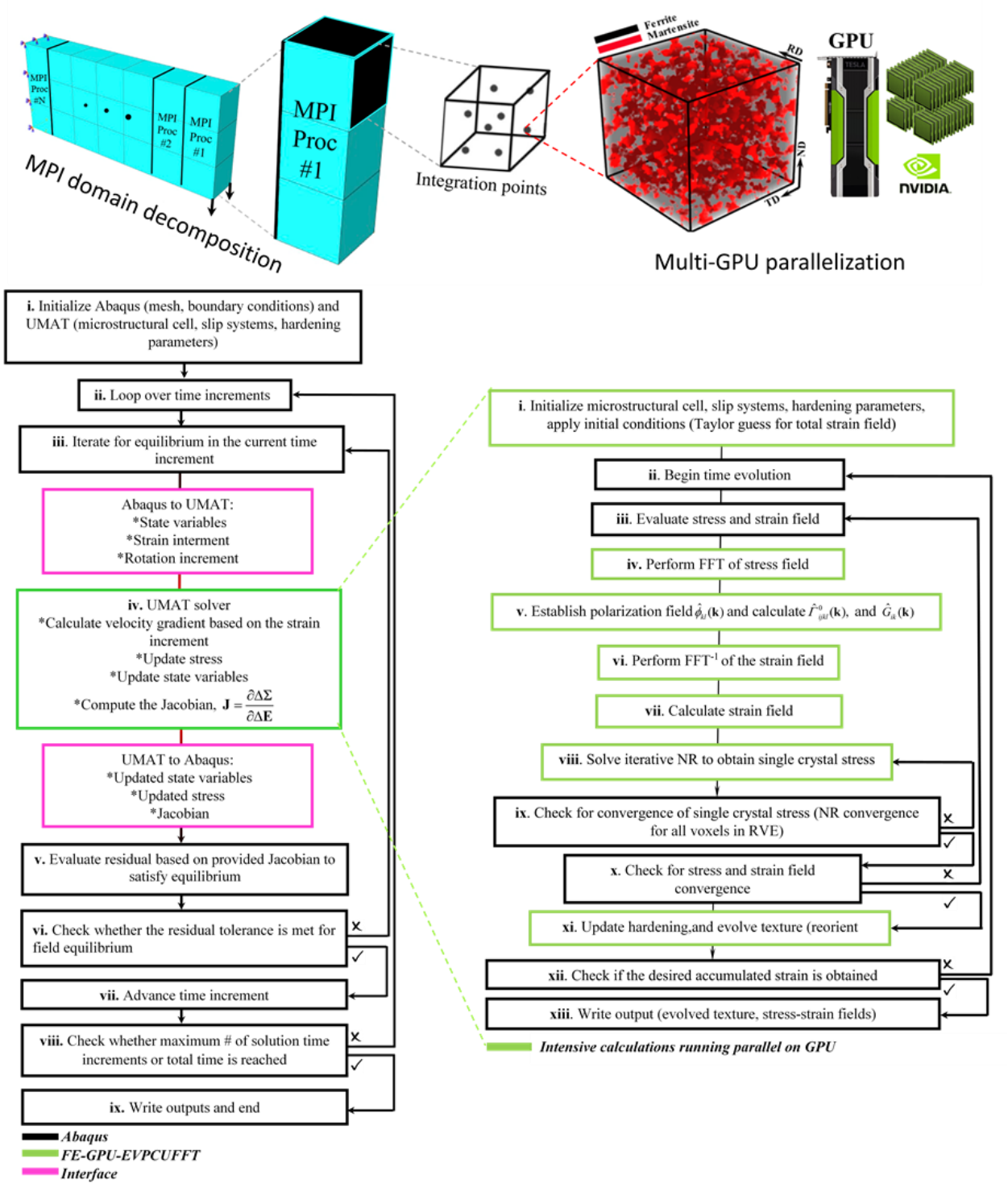
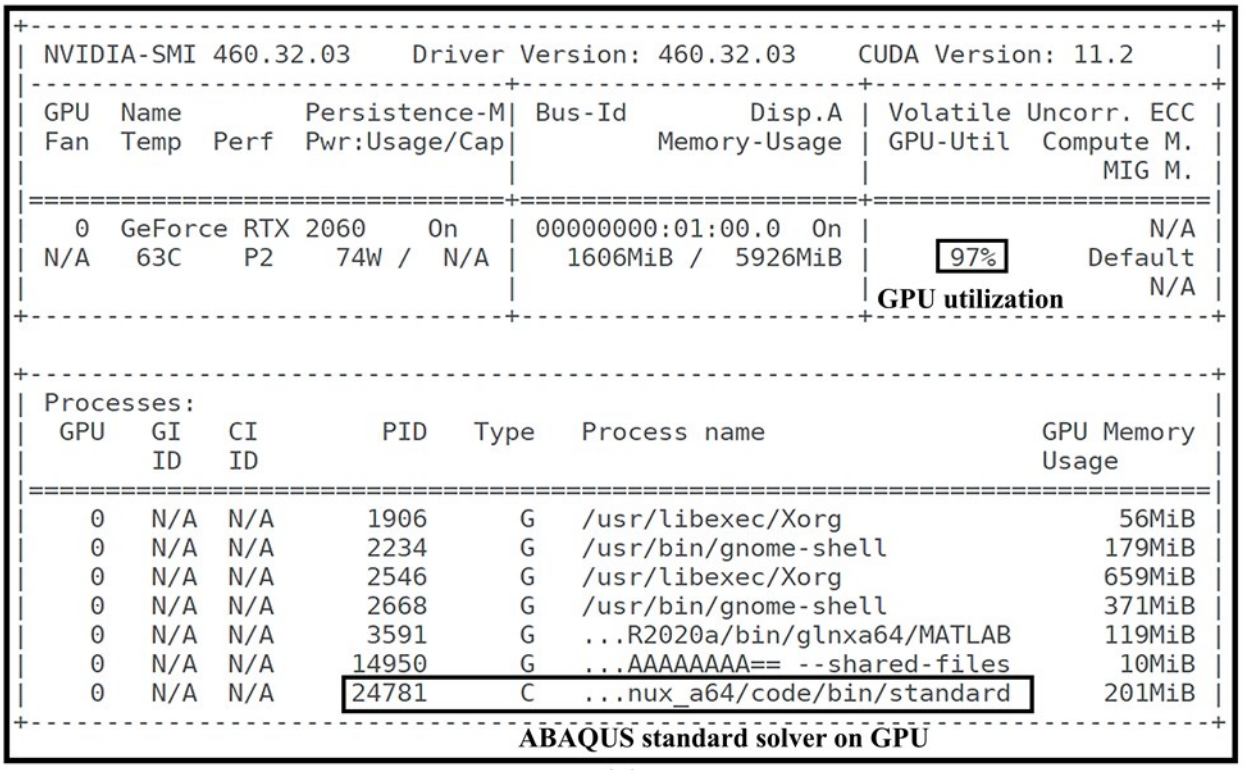
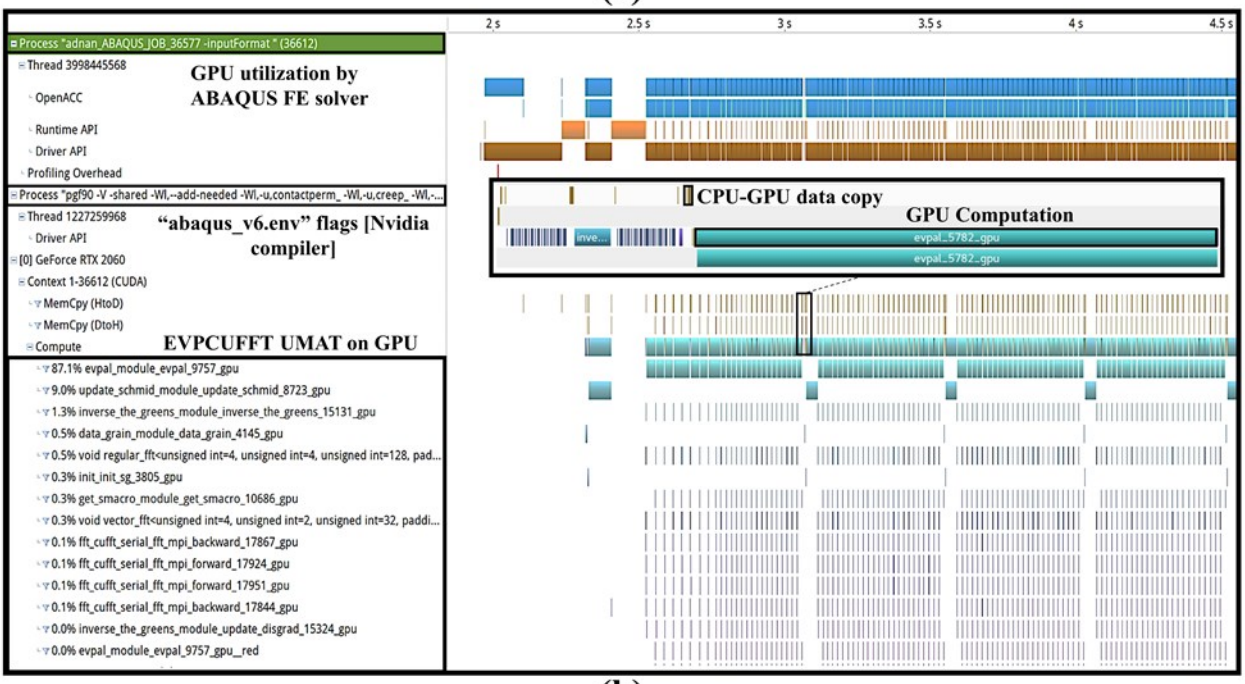


Fig. 2. Illustration of Abaqus utilizing CPUs and MPI for domain decomposition at the mesh level, while a microstructural cell embedded at an integration point runs on GPU. Flowchart of the developed hybrid implementation.



(a)



(b)

Fig. 3. (a) Evidence of UMAT GPU utilization within Abaqus standard solver provided by Nvidia-smi. (b) Performance and profiling details pertaining to the GPU accelerated UMAT called as an internal process by Abaqus solver. The information is provided by Nvidia visual profiler v2020. The GPU utilized to report this figure is the Nvidia GeForce RTX 2060.

4. Model validation

In order to evaluate the accuracy of the FE-GPU-EVPCUFFT multiscale model, several benchmarks are performed, and results are compared with the SA-GPU-EVPCUFFT solver and experimental observations. We compare the modeling results under the monotonic, cyclic, and shear boundary conditions, which the SA model can handle.

4.1. Simple compression and shear of polycrystalline copper

First, we compare simple compression and shear of a polycrystalline oxygen free high conductivity copper (OFHC) between SA-GPU-EVPCUFFT and multiscale FE-GPU-EVPCUFFT. Copper has face-centered cubic (FCC) structure deforming on $\{110\}\langle\bar{1}11\rangle$ family of slip systems. Calibration of true stress-true strain is performed in compression using SA-GPU-EVPCUFFT up to a true strain level of 0.47 under an applied quasi static strain rate of 0.001 s^{-1} at room temperature. The model features a dislocation density (DD)-based hardening law. A polycrystalline RVE of 8^3 voxels with 100 crystal orientations (i.e., ~5 voxels per grain) and randomly distributed orientations (i.e., uniform texture) was prepared synthetically in the DREAM.3D software package [67]. Single crystal elastic stiffness constants (i.e., C_{11}, C_{12}, C_{44}) and the established DD hardening parameters are listed in Table 1. Experimental data used for calibration of data is reported in [68].

The FE-GPU-EVPCUFFT model was run using a single linear FE element of type C3D8 with 8 integration points, each embedding the same 8^3 RVE microstructure of 100 grains. The analysis of simple shear in 23 (YZ) direction was performed in order to verify the correct treatment of rotations in the finite-deformation kinematic framework. It should be noted that a multiscale FE-GPU-EVPCUFFT simulations, even for resolution RVEs (i.e., 8^3), requires a large number of state variables since the information is stored per voxel and per slip system. Figure 4 compares the true stress - true strain curves and deformed textures for the SA-GPU-EVPCUFFT and multiscale FE-GPU-EVPCUFFT. Textures are also represented as individual orientations to facilitate a complete visual comparisons. The SA model utilized a constant applied strain rate of 0.001 s^{-1}. In contrast, some numerical fluctuations in state variables from one integration point to another within the element were unavoidable. Even though the fluctuations were extremely small, they

were sufficient to cause some appreciable differences between curves calculated using the SA and FE models.

Table 1. Single crystal elastic constants [69] and dislocation density hardening parameters established for OFHC Cu.

Parameter	Eq. #	Value
$C_{11}[MPa]$	A2	168,400
$C_{12}[MPa]$		121,400
$C_{44}[MPa]$		74,500
$\tau_0[MPa]$	A24b	7.0
$k_1[m^{-1}]$	A31	1.5e08
$D[MPa]$	A31	55.0
g	A31	0.13
$\rho^s_{forw}\|_{t=0}[m^{-2}]$	A30	1.0e10
H	A25	0.0
q	A33	0.1
$\mu[MPa]$	A25	46,500
n	A1a	20

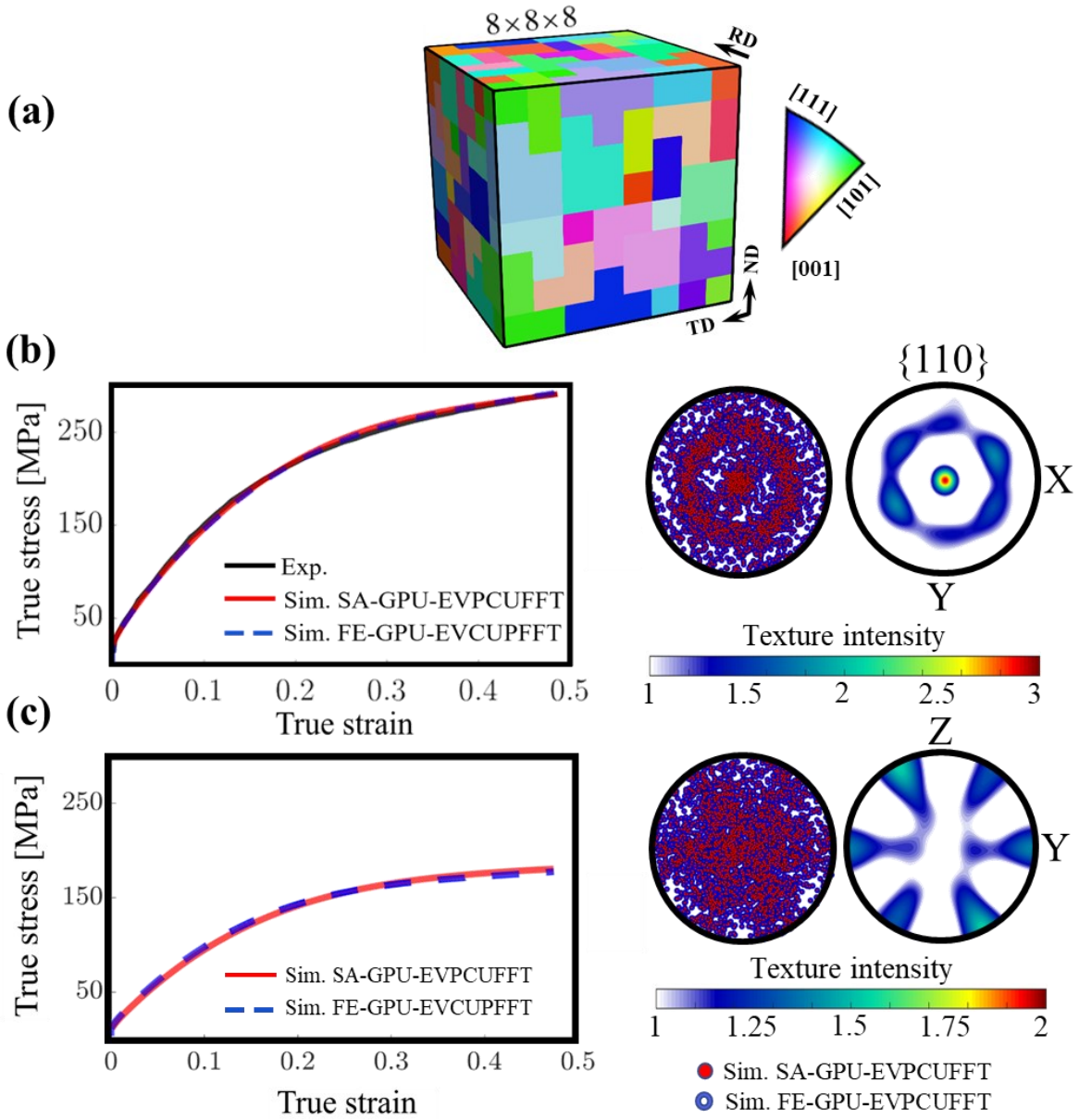


Fig. 4. (a) An 8^3 microstructural cell used for the compression and shear simulations of polycrystalline Cu using SA-GPU-EVPCUFFT and FE-GPU-EVPCUFFT wherein the cell was embedded at each integration points of a single FE element of type C3D8. (b) True stress – true strain curves and pole figures showing comparisons between simulated strength and texture evolution in simple compression using SA-GPU-EVPCUFFT and FE-GPU-EVPCUFFT at 0.47 strain. The measured stress-strain curve is also provided. (c) True stress – true strain curves and pole figures showing comparisons between simulated strength and texture evolution in simple shear using SA-GPU-EVPCUFFT and FE-GPU-EVPCUFFT at 0.47 strain. Tensor components $\blacksquare_{33}$ for simple compression and $\blacksquare_{23}$ for simple shear are plotted. The provided texture intensity pole figures are simulated using the FE-GPU-EVPCUFFT model. Those simulated using SA-GPU-EVPCUFFT are not appreciably different and are not shown.

4.2. Cyclic loading of DP590 steel

The simple compression verification in the last section has verified the accuracy of the multiscale model for monotonic loading and thus the correct implementation and state variable manipulation. In addition to monotonic deformations, the SA-GPU-EVPCUFFT solver and its advanced multiscale FE-GPU-EVPCUFFT version are further compared in predicting the cyclic deformation and underlying non-linear unloading and the Bauschinger effect. To this end, we simulate load reversals for DP 590 steel and compare the results with the experimental data from [48]. The FE-GPU-EVPCUFFT model utilized an 8^3 RVE with 100 ferrite grains within a single C3D8 element. The material 7.7% martensitic phase distributed uniformly within the ferritic phase. Considering such a distribution, # (the number) of martensitic voxels spreading within ferritic phase is 40 out of total 512. The microstructure together with the initial texture of ferrite as the predominating phase are provided in Fig. 5a [48]. As for Cu, the microstructure is synthetically generated and initialized with texture in DREAM.3D [67]. The texture shown in the figure is enforced in the ferrite, while martensite has a uniformly distributed texture. Since the volume fraction of martensite is small compared to that of ferrite, the resulting overall texture is very close to the ferrite texture.

The cyclic loading using FE-GPU-EVPCUFFT was set as three loading steps (i.e., tension-compression-tension). Figure 5b compares the SA and FE multiscale model simulation results with experiments at tensile pre-strains of 1%, 2%, 4%, 6%, 8%, and 10% under a quasi-static strain rate of $\pm 0.001 s^{-1}$ at room temperature. Material constants used in the DD hardening law and the back-stress law are taken from [48].

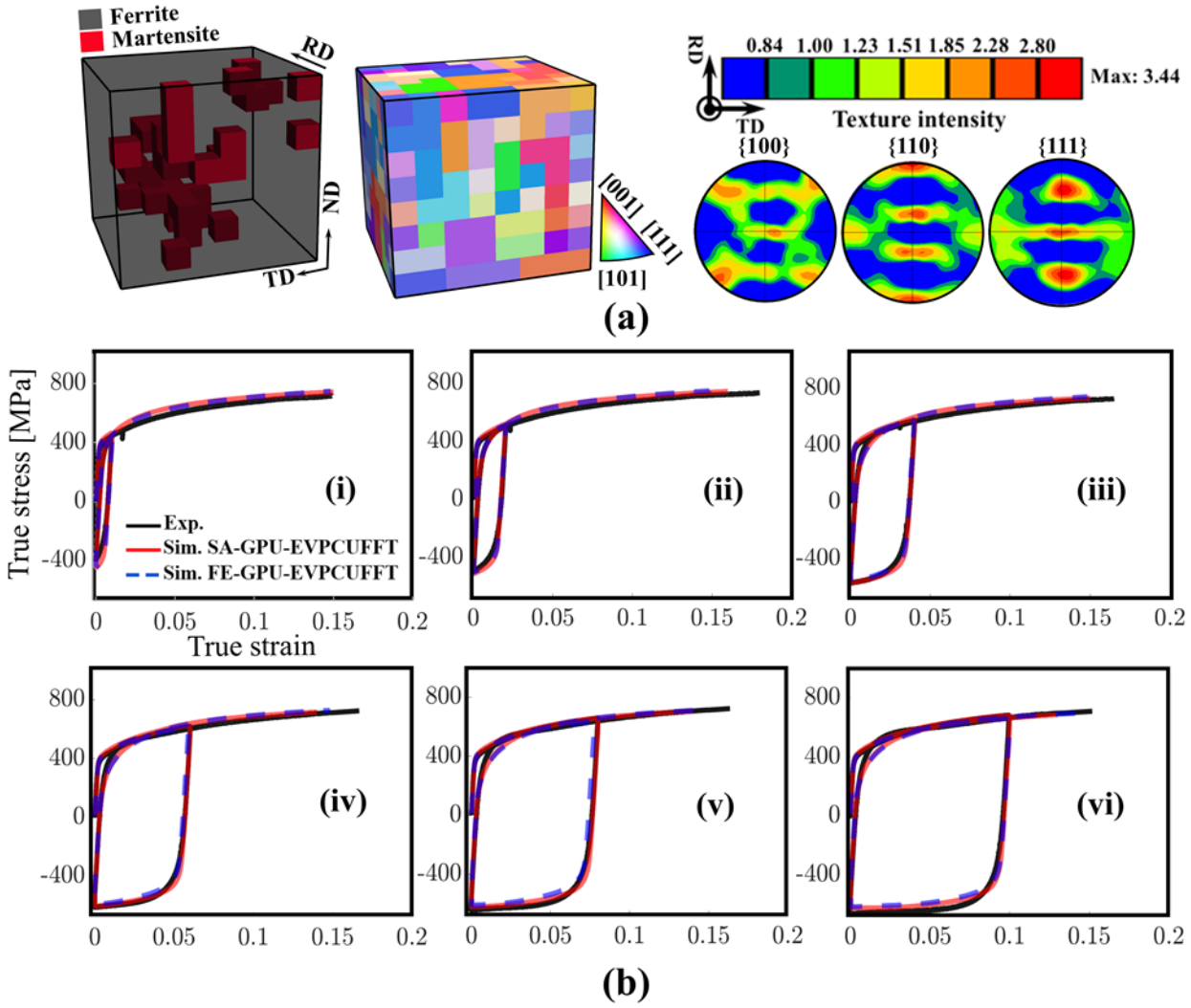


Fig. 5. Load reversal simulation cases for DP590 steel under a quasi-static strain rate of $\pm 0.001\,s^{-1}$ at room temperature to validate the FE-GPU-EVPCUFFT implementation by comparing the simulation results with the SA predictions and experimental measurements from [48]. The simulations utilized an 8^3 microstructural cell consisting of 100 different crystal orientations in ferrite and 40 in martensite embedded at integration points of a single C3D8 element. (a) DP590 starting microstructure with 7.7% martensitic fraction [48], (b) cyclic tension-compression response for (i) 1%, (ii) 2%, (iii) 4%, (iv) 6%, (v) 8%, and, (vi) 10% tensile pre-strain levels.

4.3. Uniaxial compression of a cylinder embedding a single crystal of Cu

Validations are further advanced to simulate simple compression of a multi-element FE cylinder embedding 8^3 RVE microstructures with an underlying single copper crystal of Goss texture (i.e., [001][110] in miller-indices notation or [90° 45° 90°] in Bunge-Euler notation). In order to improve the computational efficiency of the simulation, symmetries are applied to the FE

model. The imposed symmetries are justified by the crystal orientation, which is orthorhombic. One eighth of the cylinder is modeled and discretized with 240 C3D8 brick elements and 48 C3D6 wedge elements. This elemental configuration is chosen to ensure consistency in validation against the data reported in [31, 68]. The FE model is then compressed along the Z axis to a strain of 0.5 under quasi-static strain rate of $0.001\,s^{-1}$ at room temperature and the deformed models are compared with the experimental observations. Parameters listed in Table 1 are used for this simulation.

Figure 6 illustrates the model configuration and compares the deformed shape of the cylinder simulated using the FE-GPU-EVPCUFFT model against the experimental data [68]. The cross-sectional geometry of the simulated model is obtained by extracting the nodal coordinates of the FE mesh at the end of the simulation. Results show that the model predicts the ovality of the deformed cylinder's cross section owing to the anisotropy of deformation along X and Y directions (i.e., slip systems accommodating strain in the Y direction have zero Schmid factors and therefore no strain is obtained in the Y direction). This case study shows the superiority of crystal plasticity in predicting anisotropic deformations considering crystallographic slips compared to the isotropic plasticity (e.g., J_2), for which prediction of an oval shape is not possible. The deformed cross-section would remain circular due to isotropic deformation in all directions.

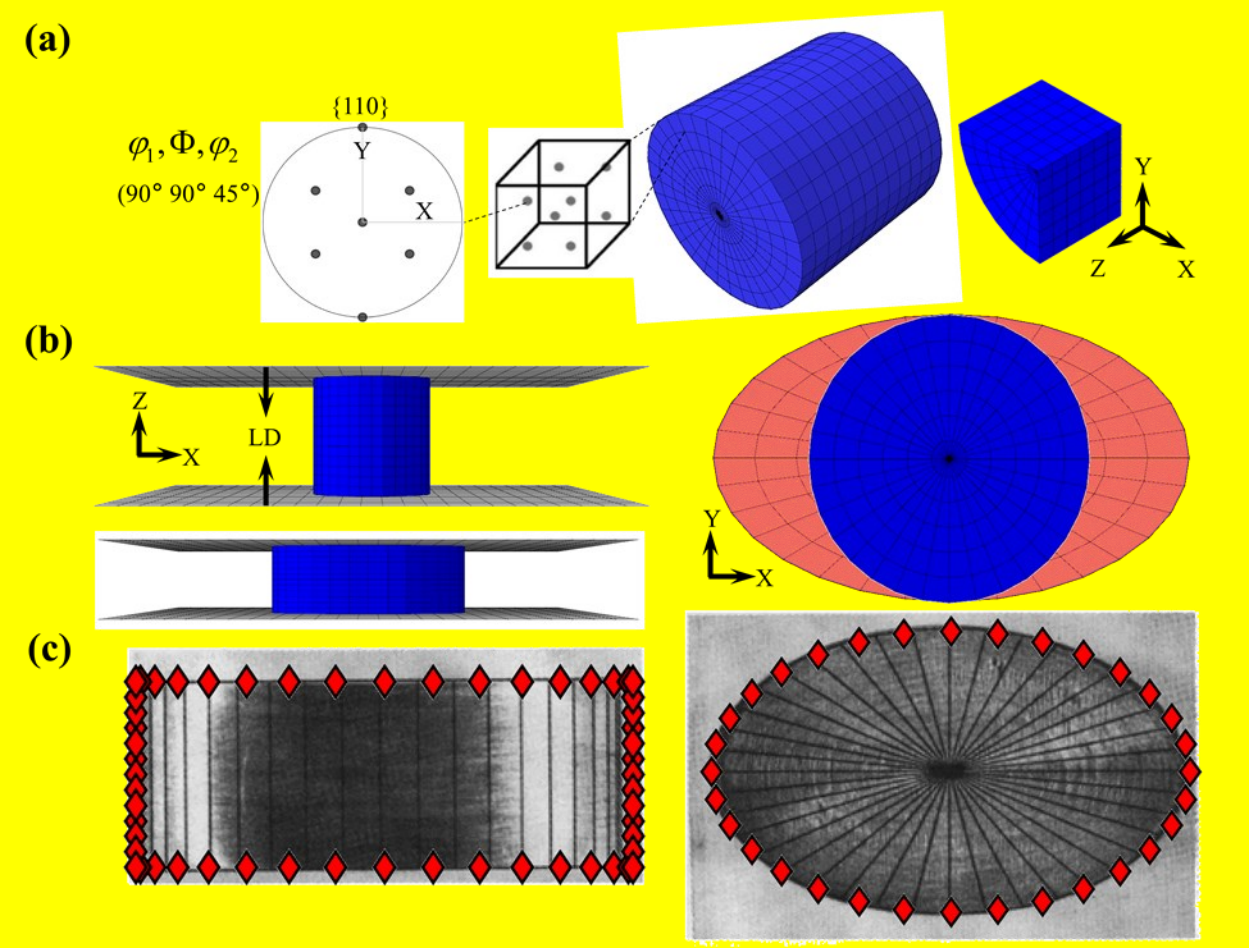


Fig. 6. FE-GPU-EVPCUFFT simulation of compression of a Cu single crystal discretized with 240 C3D8 brick elements and 48 C3D6 wedge elements under quasi static strain rate of $0.001\,s^{-1}$ at room temperature: (a) {110} pole figure showing the crystal orientation embedded at every voxel of an 8^3 microstructural cell, which is the embedded at every integration point of the 1/8 FE mesh used in the simulation, (b) simulation setup before and after compression and a view along Z before (blue) and after (red) compression, (c) external coordinates of nodes in the deformed configuration at 0.5 strain superimposed on the experimentally deformed specimen of Cu single crystal. Absence of flow of the cylinder in the Y direction demonstrates accuracy of the model.

5. Applications

After the successful validation of the FE-GPU-EVPCUFFT implementation, we apply the model for two metal forming case studies leveraging multi-element FE models. In section 5.1, we perform two simulations of four point bending of clock-rolled Zr with hexagonal close-packed (HCP) crystal structure, where the deformed beam's shape and cross sections are examined and

compared with the experimental photographs. In section 5.2, we present another bending case study to reveal micromechanical fields over a higher resolution two-phase microstructural cell. Table 2 lists the hardware specifications used to perform the simulations.

Table 2. Hardware specs of workstations leveraged for simulations and benchmarks

Workstation #	1 (1 node)	2 (1 node)	3 (5 nodes)
OS	CentOS Linux release 7.6	CentOS Linux release 7.0	CentOS Linux release 7.6
Compiler	Nvidia HPC SDK 2020 (v20.9)	Nvidia HPC SDK 2020 (v20.9)	Nvidia HPC SDK 2020 (v20.9)
ABAQUS release	2020	2020	2020
CPU	Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz	Intel(R) Xeon(R) CPU E5-2695 v4 @ 2.10GHz	Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz
System memory (GB)	376	512	772
# of CPU cores	72	72	32
# of threads per core	2	2	2
# of sockets	2	2	2
# of cores per socket	18	18	16
GPU	NVIDIA Tesla V100 (32 GB)	2 NVIDIA Tesla K80 GPUs (4 Gk210 GPUs)	NVIDIA Tesla V100 (32 GB)
CUDA toolkit version	11.0	9.1	10.1
System memory (GB)	376	512	772

5.1. Four-point bending of Zr beams

In this section, we perform two simulations of four-point bending for a Zr bar by applying bending in two orthogonal directions. Plastic anisotropy relative to the loading directionally is simulated using FE-GPU-EVPCUFFT and validated against experimental observations.

The Zr beams cut from a clock-rolled Zr plate are deformed as shown in Fig. 7a. Since Zr has an HCP crystal structure, the slip systems accommodating plastic deformation are more complex than FCC or body centered cubic (BCC) material. We consider three modes of prismatic

<**a**> slip $\{\bar{1}100\}\langle\bar{1}\,\bar{1}20\rangle$, basal <**a**> slip $\{0001\}\langle\bar{1}\,\bar{1}20\rangle$, and, 1st order pyramidal <**c**+**a**> slip $\{10\bar{1}1\}\langle\bar{1}\,\bar{1}23\rangle$. The FE-GPU-EVPCUFFT solver in its present form does not handle twinning. However, under quasi-static strain rates of 0.001 s^{-1} and room temperature, twining does not play a significant role in the deformation of the material. Implementation of twining in the solver is planned in the future research. Fig. 7b and c shows the initial microstructure and crystallographic texture (pole figures), which show a strong basal component and orthotropic symmetry.

The specimen is then deformed in two orthogonal directions of in-plane-compression (IPC) where the crystal <**c**> axis is perpendicular to the bending plane and through-thickness-compression (TTC) where the <**c**> axis is parallel to the bending plane. The calibration of true stress – true strain curves are then facilitated by a set of dislocation density hardening parameters for all three slip systems. It is notable that deformation of the modes is tied to each other owing to their internal interactions, therefore, modification of DD hardening parameters is done concurrently. In addition, tension-compression asymmetry observed in IPC and IPT (in-plane-tension) is predicted by considering the non-Schmid effects for the prismatic and pyramidal slip modes [70, 71]. It is known for HCP metals that these two modes contribute the most for non-Schmid stress projections on the glide plane, and, in the glide direction where two orthogonal shear stress components and the three normal stress components are included in the activation criterion [70]. Figure 7d presents the calibrated Zr curves using the SA-GPU-EVPCUFFT solver under compression in two orthogonal IPC and TTC directions, and tension in the IPT direction. The accuracy of fitting/calibration justifies the adequacy of considering only slip deformations as responsible for the prediction of material response under quasi-static loading and room temperature. It is important to mention that a single set of parameters are established to fit all IPC, TTC, and IPT curves concurrently. Material constants and DD hardening parameters for Zr are presented in Table 3. The non-Schmid parameters are provided by Table 4.

Once the material is calibrated using the SA solver, the four-point beam bending simulations of Zr are performed using the FE-GPU-EVPCUFFT model. The deformation setup is illustrated in Fig. 7a. A bending test is carried out utilizing two moving internal pins and two fixed external ones. The internal pins are placed 6.35 *mm* apart while the distance between the fixed pins is 12.7 *mm*. The deformation is applied to the sample by moving the internal pins in the TTC direction (as shown in Fig.7a) by 6 *mm*. The Zr bar has a length of 50.8 *mm* and square cross

section of 6.35 *mm* length. The FE model of the bar is then created for only a quarter of the beam owing to the orthotropic sample symmetry. This reduces the total number of FE elements and thus improves in computational cost. The model is discretized with 300 ($20\times5\times3$) quadratic C3D20R elements with reduced integration points embedding 8^3 microstructural RVEs consisting of 100 different crystal orientations (grains).

The deformed shapes and cross sections of the beams between experimentally measured and FE-GPU-EVPCUFFT simulated are compared in Figs. 7e and f. The cross-sectional shapes of the beams pertaining to the simulations are obtained by extracting the nodal coordinates of the deformed FE model and superimposed on top of experimental photographs [72]. Examining Fig. 7f reveals that anisotropy of the deformed shape of the beam's cross section is well captured in loading in two different directions of IPC and TTC. In case of loading in the TTC (Z) direction, the cross section becomes more distorted towards a wedge-shape area, while in case of loading in a perpendicular direction to TTC (X or Z), the shape remains almost intact, maintaining a square cross section. In other words, the Zr bars show much higher strength in TTC direction compared to IPC direction as evident by stress-strain curves. This is owing to the strong anisotropy of the material introduced by sharp basal component of the crystallographic texture and consequently activation of harder-to-activate slip systems in the <**c**> direction.

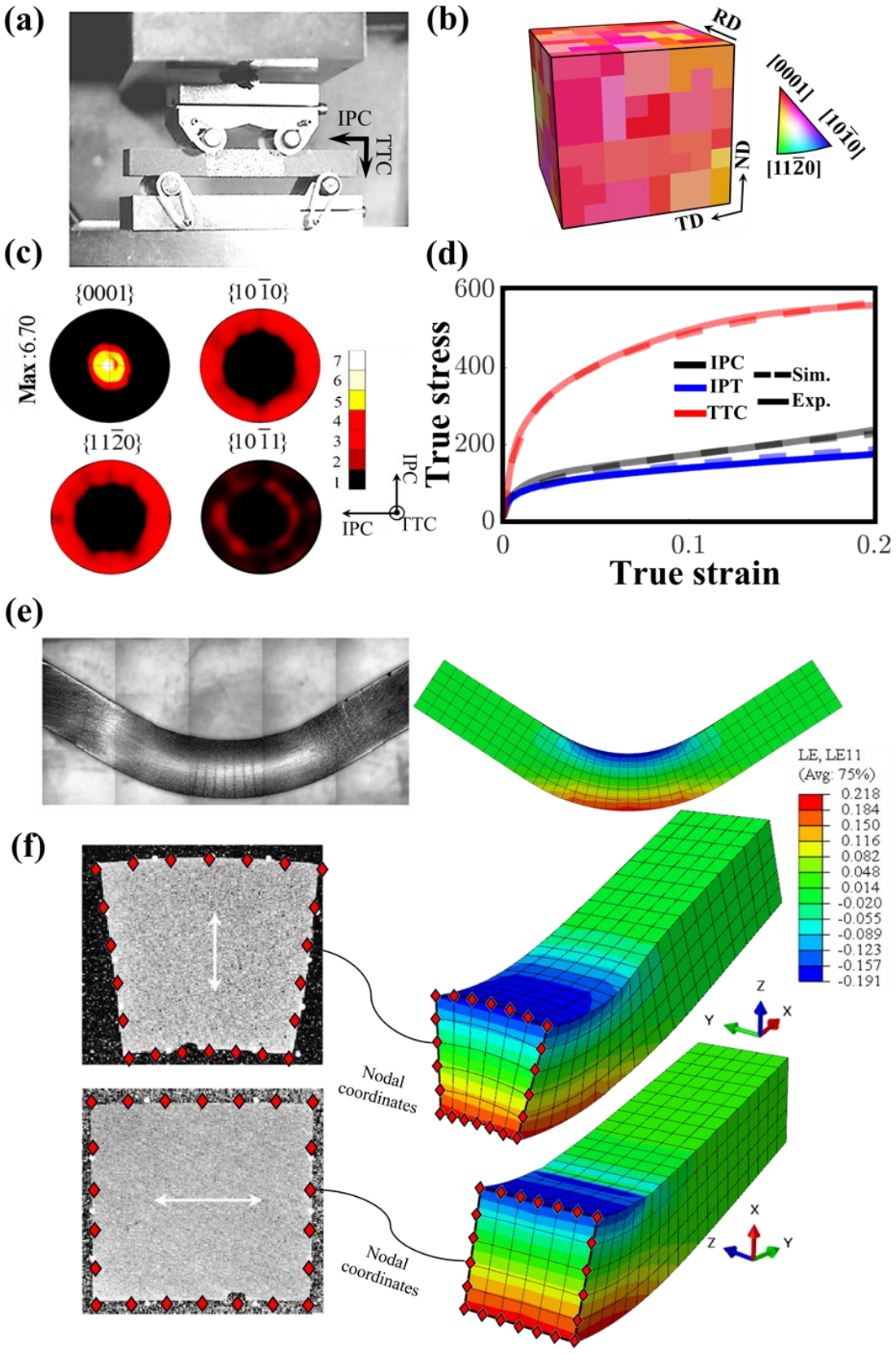

(a)
IPC
TTC
(b)
RD
ND
TD
[0001]
[10$\bar{1}$0]
[11$\bar{2}$0]
(c)
{0001}
{10$\bar{1}$0}
{11$\bar{2}$0}
{10$\bar{1}$1}
Max :6.70
7
6
5
4
3
2
1
IPC
TTC
(d)
True stress
True strain
600
400
200
0
0
0.1
0.2
IPC
IPT
TTC
Sim.
Exp.
(e)
LE, LE11
(Avg: 75%)
0.218
0.184
0.150
0.116
0.082
0.048
0.014
-0.020
-0.055
-0.089
-0.123
-0.157
-0.191
(f)
Nodal coordinates
Nodal coordinates

Fig. 7. Simulation of four-point beam bending of clock-rolled (CR) Zr under quasi-static strain rate of 0.001 s^{-1} at room temperature using FE-GPU-EVPCUFFT. Due to the orthotropic sample symmetry, a quarter of the beam is discretized with 300 ($20\times5\times3$) quadratic C3D20R elements with reduced integration. (a) Experimental setup featuring internal pins (displacing downwards for 6 mm) and two fixed external pins. (b) Microstructural cell (8^3) embedded at each integration point. (c) Pole figures showing the initial texture of the Zr plate. (d) Comparison of measured and simulated true stress – true strain curves used to calibrate the SA-GPU-EVPCUFFT model (IPT – in-plane tension, IPC – in-plane compression, and TTC – through-thickness compression). (e) Experimentally deformed and predicted beam shoving the strain fields. (f) Comparison between measured and predicted cross-sections for the Zr beams (IP=X=Y, and TT=Z).

Table 3. Single crystal elastic constants [69]
$C_{11}=143,500MPa, C_{12}=72,500MPa, C_{13}=65,400MPa, C_{33}=164,900MPa, C_{44}=32,100MPa$,
μ=33.5GPa and calibrated dislocation density hardening law parameters for clock-rolled Zr.

Parameter	Prismatic	Pyramidal	Basal
Slip mode	$\{\bar{1}100\}\langle\bar{1}\,\bar{1}20\rangle$	$\{10\bar{1}1\}\langle\bar{1}\,\bar{1}23\rangle$	$\{0001\}\langle\bar{1}\,\bar{1}20\rangle$
$\tau^{\alpha}{}_{0}[MPa]$	15.5	194.5	58.5
$k_1^{\alpha}[m^{-1}]$	5.0e7	5.0e8	4.0e8
$D^{\alpha}[MPa]$	15.0	500.0	60.0
g^{α}	0.04	0.01	0.015
$\rho_0^{\alpha}[m^{-2}]$	1.0e11	1.0e11	1.0e11
H^{α}	0.0	0.0	0.0
q^{α}	14.4	14.4	14.4

Table 4. Non-Schmid constants calibrated to promote the tension-compression asymmetry exhibited by the clock-rolled Zr plate.

Prismatic $\{\bar{1}100\}\langle\bar{1}\,\bar{1}20\rangle$				Pyramidal $\{10\bar{1}1\}\langle\bar{1}\,\bar{1}23\rangle$			
c_1	c_2	c_3	c_4	c_1	c_2	c_3	c_4
0.0	0.0	0.12	-0.012	0.0	0.0	0.05	-0.05

5.2. Bending of a cantilever beam of DP1180 steel to reveal heterogeneous micromechanical fields

This section demonstrates the advantage of the full-field[2] implementation over mean-field models (e.g. self-consistent FE models) [59]. We show that the FE-GPU-EVPCUFFT framework facilitates the qualification and quantification of micromechanical fields within the underlying microstructures embedded at each integration point of the FE elements.

To demonstrate this potential, a cantilever beam bending simulation of dual-phase DP1180 steel microstructure of higher resolution ($128\times4\times128$) is embedded. Instead of a cubed RVE, we choose a slice of RVE for computational efficiency. It is important to mention that while higher resolutions (e.g. 256^3) are simulated with the SA solver in earlier works [48, 50, 73], a resolution of $128\times4\times128$ for the multiscale FE-GPU-EVPCUFFT model is considered as a large data set. This is owing to the large number of required state variables and high memory usage correspond to such resolution which slows down the FE simulations considerably. Memory requirement for running FE-GPU-EVPCUFFT with underlying RVE resolution of $128\times4\times128$ was 364 GBs. Simulation was run using 5 MPI processes controlling 5 Nvidia Tesla V100 GPUs. Workstation #3 was utilized for this simulation.

The DP 1180 steel contains 45% martensite distributed within the ferritic phase. Figure 8a shows the initial texture, which is evidently orthotropic justifying the use of mirror symmetries in the model setup. Calibration of the material and corresponding DD hardening parameters were presented in the earlier work [48]. In addition to those calibrations, we add the predictions of tension-compression asymmetry promoted by the non-Schmid law. Figure 8b illustrates the true stress – true strain predictions by SA-GPU-EVPCUFFT for quasi-static strain rate of 0.001 s^{-1} and room temperature. Table 5 lists the non-Schmid constants utilized for the concurrent modeling of tension and compression curves for DP 1180 steel to give rise to the asymmetry.

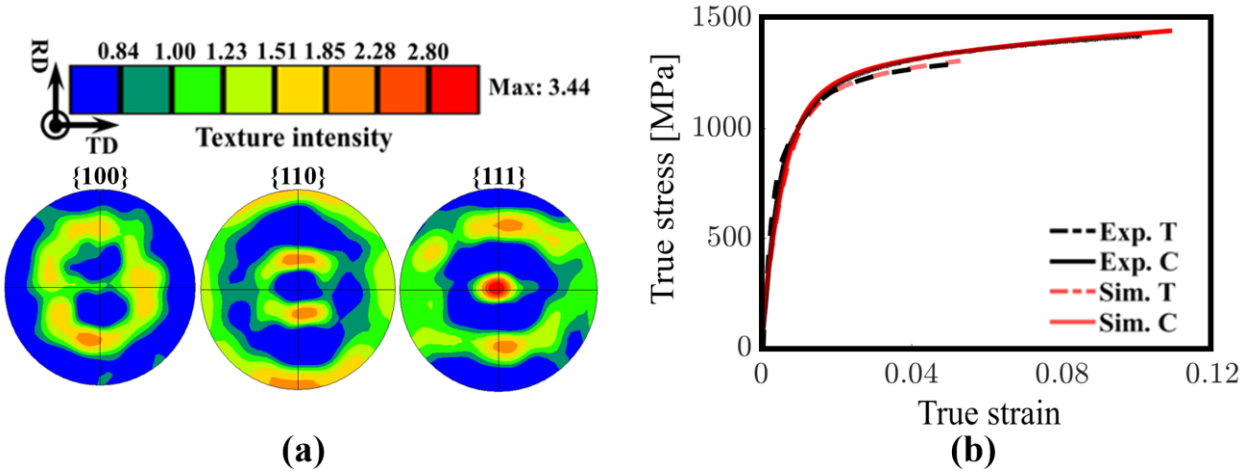


Fig. 8. Calibration of non-Schmid constants to predict the tension-compression asymmetry for DP1180 using the SA-GPU-EVPCUFFT model: (a) pole figures showing the initial texture [48] and (b) comparison of measured (Exp.) and simulated (Sim.) true stress – true strain curves in tension (T) and compression (C) under the quasi-static strain rate of 0.001 s^{-1} at room temperature.

Table 5. Non-Schmid constants calibrated to predict the tension-compression asymmetry exhibited by DP1180 steel.

c_1	c_2	c_3	c_4
0.0	0.0	0.04	-0.04

The FE model corresponding to the DP 1180 beam is discretized over 10 ($2\times1\times5$) C3D20 elements embedding $128\times4\times128$ microstructural RVE with 253 grains. Of these, 114 grains with 29,494 voxels are martensite. The XZ plane symmetry is exploited to double the computational efficiency.

Figure 9a shows the FE model consisting of C3D20 elements, the corresponding boundary conditions, and underlying DP1180 martensitic-ferritic microstructure embedded in each integration point of the FE model. The beam's geometry has a dimension of $10\,mm\times5\,mm\times40\,mm$, fixed at one end and deformed $1\,mm$ downwards at the other end (cantilever beam). Figure. 9b shows the deformed FE model displaying the axial strain and VM stress contours. The wall clock time for this simulation was about 7.2 days.

To examine the embedded microstructural evolution, two elements, one with the high stress and another with low stress are selected and the correspond underlying micromechanical stress

fields are extracted and plotted in Fig. 9c. The multiscale FE-GPU-EVPCUFFT predicts the contrast between the martensitic and ferritic phases, where the higher levels of stress are predicted in the harder martensitic regions compared to the softer ferrite grains. The results demonstrate the capability of the full-field^2 implementation to capture the heterogeneity of micromechanical fields corresponding to various FE elements experiencing dissimilar states of deformation.

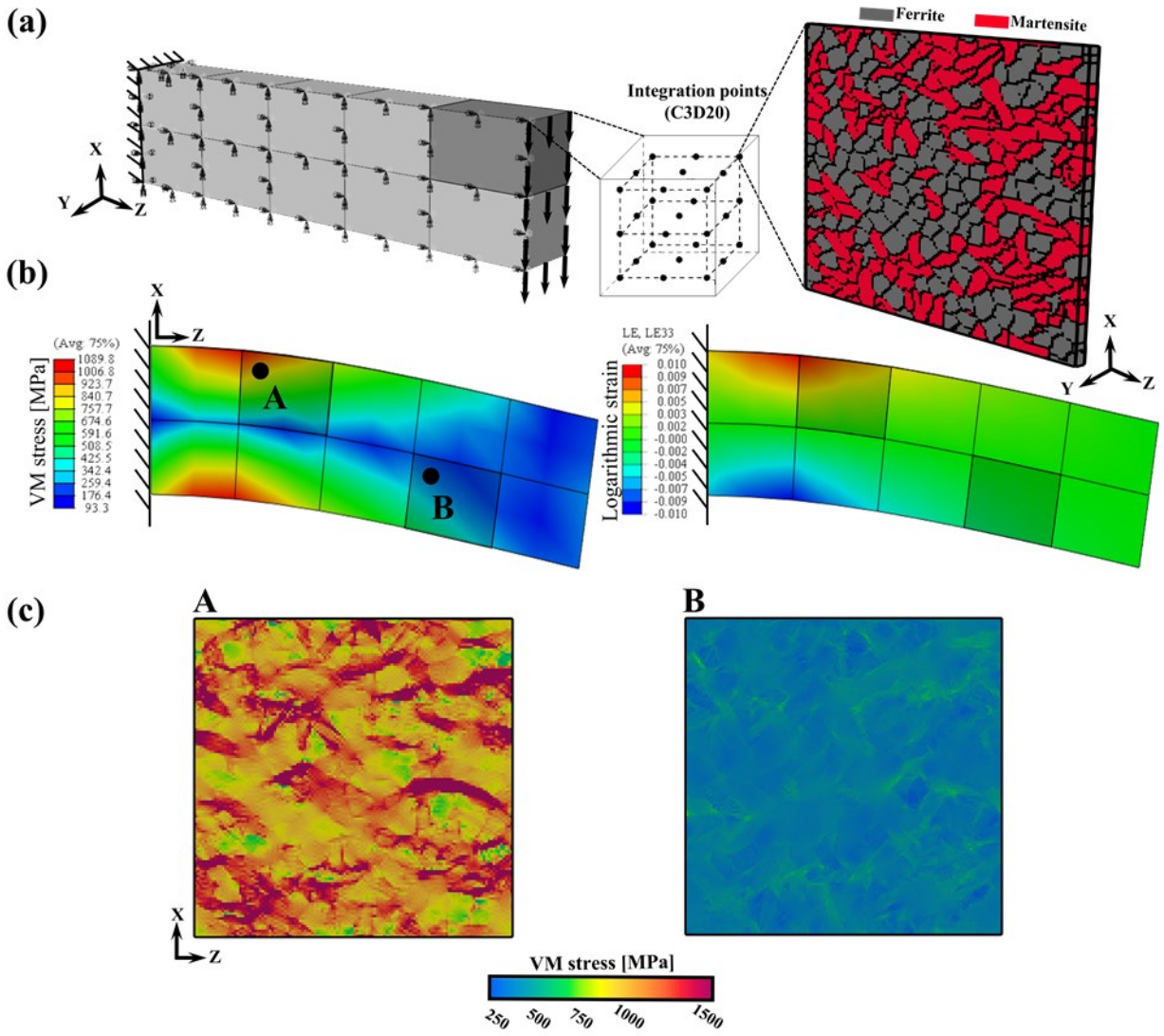


Fig. 9. Simulations of DP1180 steel cantilever beam bending using the FE-GPU-EVPCUFFT model. The beam has dimension of $10\,mm \times 5\,mm \times 40\,mm$ and is displaced downwards in –X at the free end for $1\,mm$. (a) FE model consisting of C3D20 quadratic element depicting the applied XZ symmetry and microstructural cell of $128 \times 4 \times 128$ resolution, (b) the deformed FE model with some exaggeration (the deformation scale factor of 5) showing the von Mises (VM) stress and

axial strain contours, and (c) predicted fields of VM stress over the cell after bending in the regions of high and low deformation.

6. Numerical tests for order of convergence

In order to examine the convergence of the FE-GPU-EVPCUFFT solver, we follow the method provided in [31], where a generic system of nonlinear equations is solved using the standard Newton method. Governing equations of the study are written in terms of the macroscopic/global quantities (the capital letters). A residual defined as

$$\Delta\mathbf{X}(\Delta\boldsymbol{\varepsilon}) = \Delta\boldsymbol{\Sigma}(\Delta\boldsymbol{\varepsilon}, \Delta t) - \Delta\boldsymbol{\Sigma}^{app}, \tag{7-1}$$

where $\Delta\boldsymbol{\Sigma}^{app}$ denotes the applied macroscopic stress, which known in advance. The constant applied stress is $\Delta\boldsymbol{\Sigma}^{app} = \langle\Delta\boldsymbol{\sigma}^{app}\rangle = \{5.42, 7.10, -12.51, -1.01, 0.38, 4.01\}$. The corresponding Jacobian for the Newton method is that of the FE-GPU-EVPCUFFT UMAT

$$\mathbf{J} = \frac{\partial\Delta\mathbf{X}}{\partial\Delta\mathbf{E}} = \frac{\partial\Delta\boldsymbol{\Sigma}}{\partial\Delta\mathbf{E}}. \tag{7-2}$$

The FE-GPU-EVPCUFFT UMAT calculates stress $\Delta\boldsymbol{\sigma}(\Delta\boldsymbol{\varepsilon}, \Delta t)$ at every voxel given an increment in strain and time (in seconds). The local stress field is subsequently averaged into $\Delta\boldsymbol{\Sigma} = \langle\Delta\boldsymbol{\sigma}\rangle$ for the comparison with the applied macroscopic stress $\Delta\boldsymbol{\Sigma}^{app}$ to obtain a next guess. The material parameters and texture were those of polycrystalline Cu (Fig. 4). The residual is iteratively minimized. The solution error e_i at the i^{th} iteration is then defined based on norm of the sought solution and the current guess, as follows

$$e_i = \left\|\Delta\mathbf{E}^* - \Delta\mathbf{E}_i\right\|. \tag{7-3}$$

The sought solution of the problem, which corresponds to the applied stress, is known a priori $\Delta\mathbf{E}^* = \{0.0008235, 0.001477, -0.002301, -0.0008762, -0.0001393, 0.001745\}$. The initial guess was the elastic strain corresponding to the applied stress. By definition, the order of convergence, a, is

$$\frac{e_{i+1}}{e_i^a} < b;\ b > 0;\ \begin{cases} a = 1: \text{linear} \\ 1 < a < 2: \text{super-linear} \\ a \geq 2: \text{quadratic} \end{cases}. \tag{7-4}$$

where *b* indicates the rate of convergence. Utilizing this methodology, we study the effects of both time increment and underlying microstructural resolution on the order of convergence for the FE-GPU-EVPCUFFT multiscale model. Fig. 10a shows the e_{i+1} versus e_i log plots as a function of time increment: Δt =0.02, Δt =0.05, and, Δt =0.1 second. Table 6a lists numbers for subsequent Newton iterations arriving at the converged solution. Note that a larger time increment results in higher # of iteration attempts to reach the same tolerance. Results indicate a quadratic order of convergence upon the elastic guess to reach the error in the range of $10^{-3} < e_{i+1} < 1$, followed by a super-linear order of convergence for a tolerance range of $10^{-6} < e_{i+1} < 10^{-3}$. Varying the time increment from Δt =0.02 to Δt =0.05 and Δt =0.1, results in a similar trend with slightly lower convergence rates and an increase in number of attempts to obtain the same tolerance.

Fig. 10b shows the effect of RVE resolution on the order of convergence for a time increment of Δt =0.1. Table 6b lists the orders of convergence as a function of RVE size with correspond resolutions of $8\times4\times8$, $16\times4\times16$, $32\times4\times32$, and $64\times4\times64$. The frequency indicates the subsequent NR iterations. Results indicate while the convergence rate for lower RVE resolution of $8\times4\times8$ is slightly faster with lower # of solution attempts, negligible difference is observed for the orders of convergence belonging to the microstructural resolutions of $16\times4\times16$ and higher.

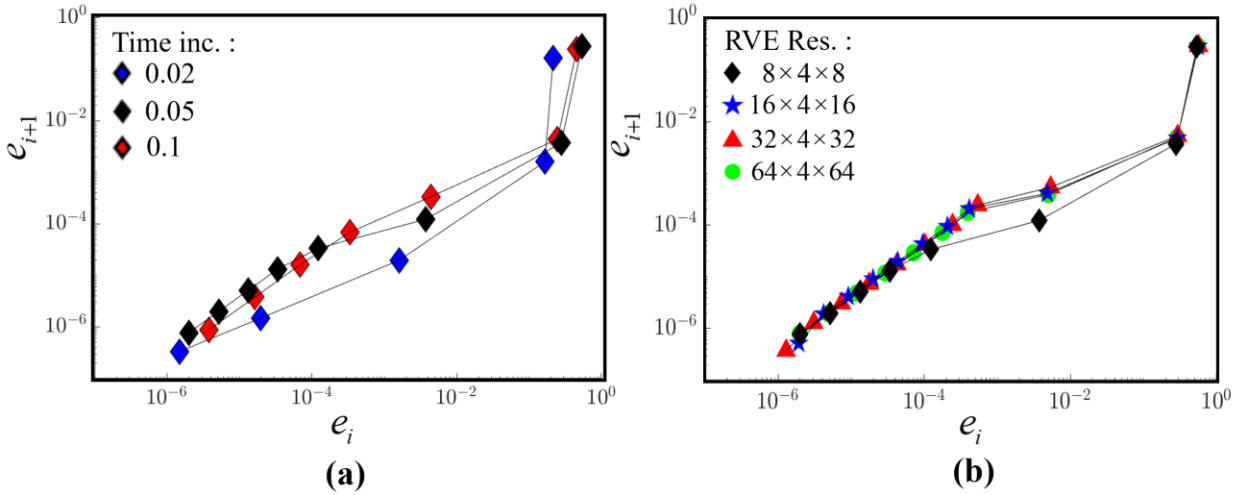


Fig. 10. Convergence benchmarks in compression of a cube FE model with eight C3D8 elements using the FE-GPU-EVPCUFFT implementation showing $\log(e_{i+1})$-$\log(e_i)$ for: (a) microstructural RVE resolution of $8\times4\times8$ and time increments of Δt =0.02, Δt =0.05, and, Δt =0.1 and (b)

microstructural RVE resolutions of $8\times4\times8$, $16\times4\times16$, $32\times4\times32$, and $64\times4\times64$ for time increment of Δt =0.1.

Table 6a. Variation in the order of convergence as function of time increment for the compression of the cube FE model with eight C3D8 elements embedding a microstructural cell of $8\times4\times8$ voxel resolution.

Time increment	Orders of convergence for subsequent Newton iterations
Δt =0.02	3.5671, 1.6835, 1.2385, 1.1112
Δt =0.05	3.8722, 1.4731, 1.1984, 1.1512, 1.1321, 1.1166
Δt =0.1	4.3686, 1.6102, 1.1439, 1.0907, 1.0842, 1.0776, 1.0720

Table 6b. Order of convergence as a function of RVE resolution for the compression of the cube FE model embedding eight C3D8 elements using a time increment of Δt =0.1.

RVE resolution	Orders of convergence for subsequent Newton iterations
$8\times4\times8$	4.3686, 1.6102, 1.1439, 1.0907, 1.0842, 1.0776, 1.0720
$16\times4\times16$	4.3050, 1.4615, 1.0881, 1.0924, 1.0841, 1.0777, 1.0720, 1.0672, 1.0630
$32\times4\times32$	4.2505, 1.4388, 1.1057, 1.1054, 1.0947, 1.0868, 1.0797, 1.0739, 1.0688
$64\times4\times64$	4.2684, 1.4793, 1.1016, 1.1046, 1.0941, 1.0861, 1.0793, 1.0735, 1.0684

An additional convergence benchmark was performed involving a simple compression of a multi-element cube FE model discretized using eight C3D8 elements and underlying RVE resolutions of $8\times4\times8$, $16\times4\times16$, $32\times4\times32$, and $64\times4\times64$ to a strain of 20% also for Cu. We performed this test even though Fig. 10 and Tables 5a and 5b indicate that microstructural resolution and time increment play a secondary effect in the convergence behavior. We list the # of iterations based on the Abaqus ".sta" output for the first 20 increments. Table 6c lists the # of iterations as a function of underlying microstructural RVE resolutions and time increments of Δt =0.02 and Δt =0.1. It is realized that choosing a smaller time increment of Δt =0.02 results in only one iteration per increment. Increasing the time increment to Δt =0.1 increases the # of iterations to 2-3 per increment at the start of the simulation, converging to one iteration at some point. As is evident, the # of iterations is really small compared to the first convergence test likely

because simple compression is easier to converge than the generic test and because Abaqus relies on a modified NR scheme to accelerate the convergence.

Table 6c. # of iterations per increment taken from the ABAQUS ".sta" output as a function of embedded RVE resolutions and time increment for the compression of the cube FE model discretized with eight C3D8 elements.

	# of iterations							
Increment #	Δt =0.1				Δt =0.02			
	8×4×8	16×4×16	32×4×32	64×4×64	8×4×8	16×4×16	32×4×32	64×4×64
1	3	3	3	3	3	3	3	3
2	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1
5	2	2	2	2	1	1	1	1
6	3	3	3	3	1	1	1	1
7	3	3	3	3	1	1	1	1
8	2	2	2	2	1	1	1	1
9	1	1	1	1	1	1	1	1
10	2	2	1	1	1	1	1	1
11	2	2	2	2	1	1	1	1
12	2	2	2	1	1	1	1	1
13	1	1	1	2	1	1	1	1
14	2	2	2	1	1	1	1	1
15	1	1	1	1	1	1	1	1
16	1	1	1	1	1	1	1	1
17	1	1	1	1	1	1	1	1
18	1	1	1	1	1	1	1	1
19	1	1	1	1	1	1	1	1
20	1	1	1	1	1	1	1	1

7. Performance benchmarks

This section is dedicated to parallel performance benchmarks of the CPU-only and hybrid CPU-GPU implementations of the multiscale full-field^2 model. First, the MPI performance offered by ABAQUS standard solver is assessed where the strong scalability as a function of # of FE elements and # of MPI processes are explored. Second, the performance improvements of hybrid FE-GPU-EVPCUFFT for single and multiple GPU(s) over the FE-CPU-EVPFFTW implementation are discussed in detail. Workstations #1 and #2 listed in Table 2 are utilized for these benchmarks.

7.1 MPI scalability

The FE solver in ABAQUS standard takes the advantage of MPI domain decomposition [49, 74-80], where slices of the FE model containing one or more elements run on their own MPI processors, concurrently. MPI parallel simulations are enabled when the models are discretized with more than one element and is enabled by passing the flag "cpus=*N*" where *N* indicates the total number of CPUs utilized for parallel simulations. Thread-based parallelization is also an option for parallel simulations on shared memory (i.e., OpenMP) by adding the flag "mp_mode=THREADS", however, the MPI implementation has been shown to be much more efficient [49, 81, 82]. Threaded parallelization usually suffers from false sharing of caches and the page size granularity that occurs in physical memory mapping. In contrast, such memory access problems are automatically avoided when using MPI [49].

In order to benchmark the performance of MPI parallelization offered by implicit FE solver in ABAQUS standard, two performance benchmarks are performed. The benchmarks are performed on workstation 1. Only "real" cores are used with no hyperthreading. Also, the locality was ensured meaning that the domains are bound to always the same CPU. First, we study the effect of the increase in number of FE elements on the wall clock time for simulations of simple compression of a cube to a strain of 2% in 10 increments with #1, #2, #4, #8, #16, #32, and, #64 C3D8 elements utilizing a single CPU (i.e. serial mode). The CPU-only implementation (aka FE-CPU-EVPFFTW) utilizes the FFTW3 library [49, 83]. Fig. 11a presents the variation of wall clock time as a function of # of FE elements. Results indicate an almost linear trend where an increase in wall clock time is proportional to the # of elements utilized in the simulation.

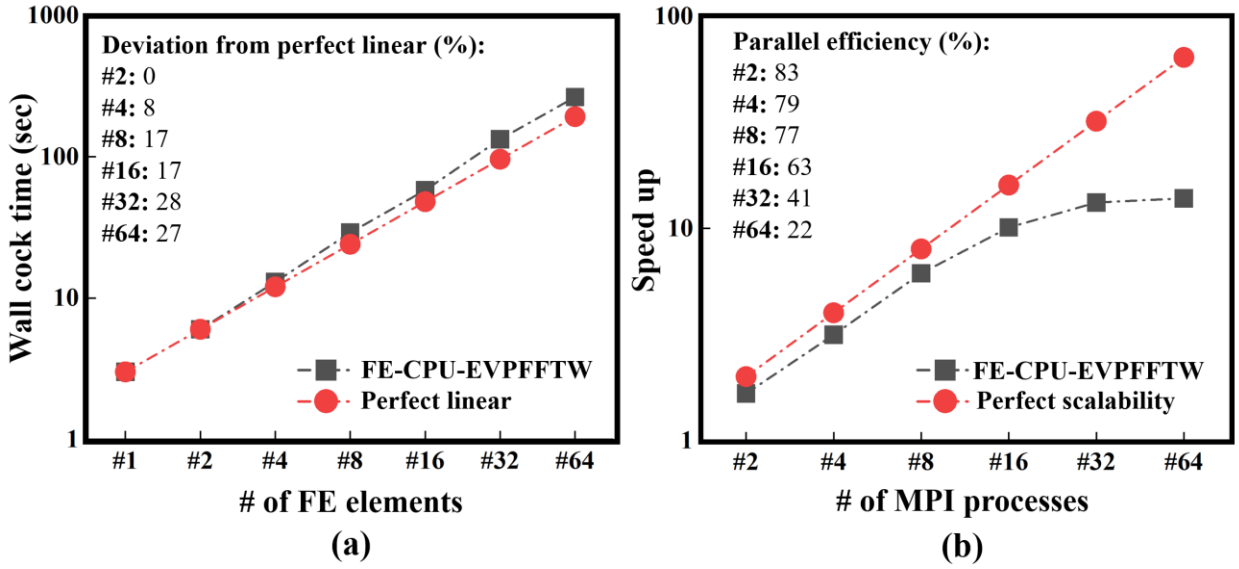


Fig. 11. Performance benchmarks of CPU-only FE-CPU-EVPFFTW implementation as a function of # of FE elements and # of MPI processes. (a) Scalability of wall cock time as a function of # of FE elements for a uniaxial compression of a cube with #1, #2, #4, #8, #16, #32, and, #64 C3D8 elements. (b) Strong scalability leveraging #1, #2, #4, #8, #16, #32, and, #64 MPI processes.

A second benchmark is conducted by simple compressions of the same FE cube with 64 C3D8 elements leveraging 1, 2, 4, 8, 16, 32, and 64 CPU cores to obtain the strong scalability of MPI runs on a single node. Fig. 11b shows the results of this benchmark. A saturation is observed in the strong scalability, where, deviations from perfect strong scalability become more significant utilizing 32 and 64 MPI processes. The current benchmark indicates best performance gains leveraging #16 or less CPUs where individual CPUs are assigned with at least #4 elements. Each MPI process should be assigned with a minimum computational workload to obtain an efficient parallel scalability. The performance degradation occurred with increasing number of CPU cores is owing to hyperthreading deficiency and shared memory bandwidth among all the threads and may be improved by running on distributed nodes with non-uniform memory access (NUMA) .The current benchmark indicates a computationally efficient parallel simulation leveraging #16 or less CPUs with a correspond parallel efficiency of 50% and more, where, each CPU is assigned with at least #4 elements. This is because each MPI process should be assigned with considerable computational workload to obtain an efficient parallel scalability. Load imbalance and performance degradation that occurs depending on which elements of the model

(i.e., element IDs) are assigned to a specific CPU core are potentially another source of imperfect strong scalability [84].

7.2 Hybrid FE-GPU-EVPCUFFT performance

This section elaborates on the performance improvements of the hybrid CPU-GPU FE-GPU-EVPCUFFT implementation over the CPU-only FE-CPU-EVPFFTW model. The hybrid solver utilizes single or multiple GPU(s) leveraging OpenACC and the CUFFT library [50, 66, 85, 86].

7.2.1 Single-GPU

Single GPU performance is evaluated as a function of microstructure resolutions by simulation of simple compression of a single FE element of type C3D8 to a strain of 2% in 10 increments embedding microstructural RVEs with $16\times4\times16$, $32\times4\times32$, $64\times4\times64$ and, $128\times4\times128$ discretizing voxels. The benchmark is performed on workstation 1 featuring a Nvidia Tesla V100 GPU (32 GB) controlled by a single core of Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz.

Figure 12 depicts the outcomes of the single GPU benchmark. It is realized that the performance gain of FE-GPU-EVPCUFFT over the FE-CPU-EVPFFTW becomes more significant with increase in resolution of embedded microstructures. This is expected since GPUs are much more efficient in computation of large data sets. The inferior result for the low resolution 16x4x16 RVE is due to a low workload to take advantage of the GPU hardware. Results indicate improvement in performance gain with an increase in underlying microstructural RVE resolutions. The hybrid FE-GPU-EVPCUFFT speeds up the simulations ~2.5x for an embedded RVE resolution of $128\times4\times128$.

According to Amdahl's law [49, 50, 87], the net obtained speed up is contingent on the ratio of UMAT computations to the FE solver itself which is not trivial to measure since the Abaqus solver reports the total time spent in FE and UMAT altogether. Amdahl's law limits the overall GPU performance. The speed up reported for UMAT solver does not isolate the speed up obtained in the UMAT itself but the overall performance of Abaqus FE solver with the UMAT. Even in case of one element model, execution time of the FE solver is not negligible compared to

that of UMAT utilization. For instance, if FE solver is taking 30% of the total wall clock time (i.e. the fraction to accelerate is f=0.7) and if the UMAT solver runs 26x (p=26) faster leveraging GPUs (the SA solver has a speed up of 26x for the resolution of $128\times4\times128$), the net speed up according to Amdahl's law [49, 87] is $S = \frac{1}{(1-f)+\frac{f}{p}} = 3.04$, which is close to the obtained speed up herein. Figure 13 reflects the Amdahl's law for the UMAT implementation presented herein, where the variation in the net speed up vs the speedup in UMAT is shown.

The memory motion associated with transferring data between the CPU and GPU is the key issue for GPU scalability of the UMAT. While the SA solver is independent meaning that all data except I/O is created on GPU to avoid memory/data transfer between CPU and GPU hardware, the state variables are inevitably transferred back and forth from GPU to CPU and vice versa in the UMAT. As Abaqus FE solver is opaque not allowing for any GPU programming, all state variables that are already on GPU's memory need to be copied to CPU for Abaqus at the end of each iteration for every integration point. Profiling the UMAT solver to compare the memory copy time w.r.t. the compute time shows that the memory copy time is small compared to compute time but the number of invocations of memory copy is large, which is the limiting factor in the net performance.

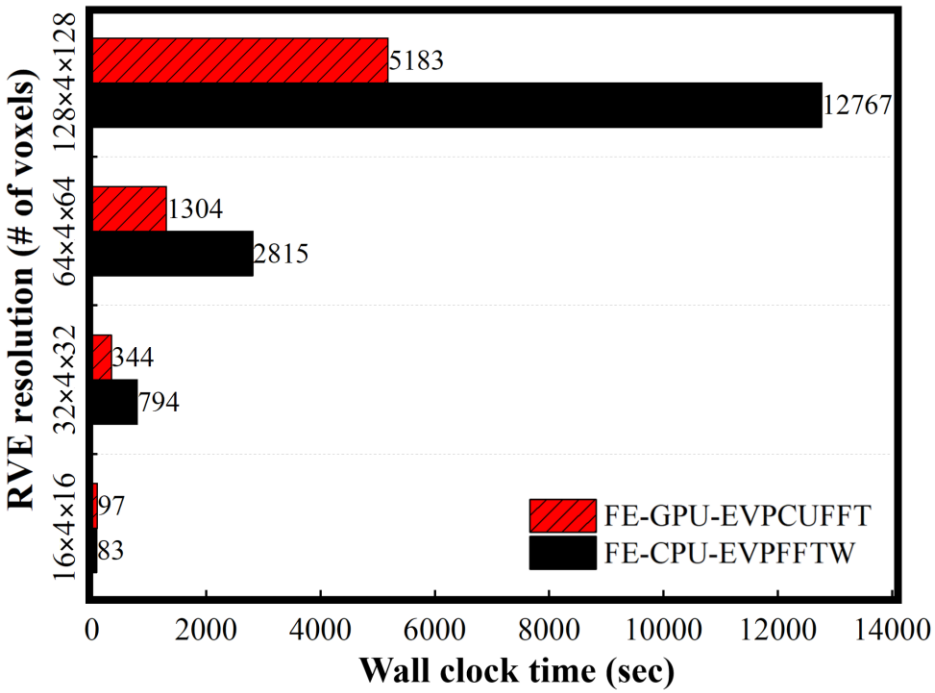


Fig. 12. (a) Performance benchmark comparison for the hybrid CPU-GPU implementation (FE-GPU-EVPCUFFT) versus the CPU-only UMAT (FE-CPU-EVPFFTW) for the compression of a single C3D8 FE model. The FE-CPU-EVPFFTW model utilizes a single core of Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz and FFTW3 libraries, while the FE-GPU-EVPCUFFT hybrid solver takes the advantage of a single Nvidia Tesla V100 GPU (32 GB) leveraging OpenACC and CUFFT libraries.

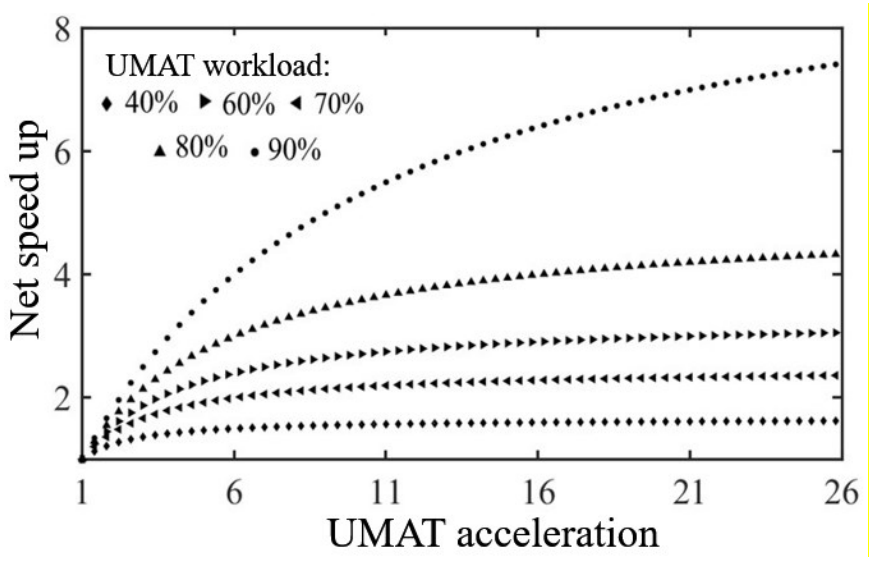


Fig. 13. Net speed up as a function of UMAT acceleration on GPU and UMAT workload.

7.2.2 Multi-GPUs

The purpose of this section is not to compare a multi-GPU performance w.r.t to a single-GPU performance - such benchmarks have been presented in a previously published work [88] - but to elaborate on the potential performance gains where adequate # of GPUs are not available for utilization. That is, to confirm that one is still able to take advantages of a hybrid MPI-GPU parallel implementation maintaining the MPI utilization offered by FE domain decomposition while benefiting from the GPU acceleration either on a single GPU or multiple GPUs with imperfect strong scalability.

Multi-GPU utilization is more involved compared to single-GPU implementation and requires the parallel information at FE domain decomposition level which is obtained from ABAQUS standard built-in functions. To determine the element IDs assigned to a MPI process, ABAQUS offers the functions “GETNUMCPUS” and “GETRANK”, representing the size (i.e., total number of MPI processes) and rank (i.e., current MPI process #), respectively. These

functions are similar to "MPI_Comm_size" and "MPI_Comm_rank" commonly used by MPI standard libraries such as OpenMPI. Note that "GETNUMCPUS" output is determined by the flag "cpus=*N*" accompanying the ABAQUS job submission, where, *N* is the # of MPI processes.

Simple compressions of a cube FE model with 4 elements of type C3D8 to a strain of 2% in 10 increments embedding microstructural RVEs with $64\times4\times64$ resolutions are simulated on workstation 2 utilizing 1, 2, and 4 Nvidia Tesla K80 GPUs where the total number of MPI processes is kept at 4 for all benchmarks. Table 7 presents the four configuration setups. A 1:1 ratio of CPUs has been used alongside the GPUs. We have ensured that each MPI process is bound to one CPU using always the same GPU, otherwise, significant overheads would arise from switching across multiple devices adversely impacting the multi-GPU performances.

The benchmark narrows down the focus on UMAT-only acceleration facilitated by GPUs by maintaining the same number of MPI processes utilized for FE solver. Such benchmark is helpful when access to GPUs is limited on a workstation but it is still preferred to take the advantage of GPUs to accelerate UMAT. Fig. 14 shows an almost perfect strong scalability of the multi-GPU benchmark for FE-GPU-EVPCUFFT (i.e., wall clock times is halved by doubling the # of GPUs). It is also important to mention that when number of GPUs pertaining to each element is less than one (i.e., several elements access a GPU), the performance may be adversely affected by overutilization resulted from concurrent access to the GPU hardware. Therefore, it is advised to limit the number of GPUs per MPI process. The ideal performance is obtained when # of FE elements and # of GPUs are the same where each FE element is assigned to a unique GPU hardware. The GPU parallelization would become profoundly effective if there are more GPUs available than MPI processes as more than one GPU would be solving one RVE.

Table 7. Configuration setups for the multi-GPU performance benchmark utilizing FE-GPU-EVPCUFFT on workstation 2.

configuration	# of FE elements	# of MPI processes for FE domain decomposition	# of GPUs accelerating the GPU-EVPCUFFT UMAT	# of elements per GPU
1	4	4	1	4
2	4	4	2	2
3	4	4	4	1

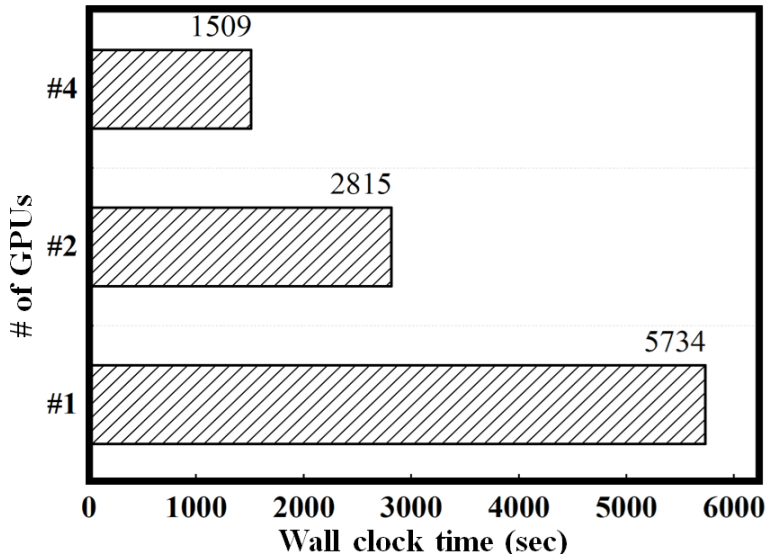


Fig. 14. Multi-GPU performance benchmark of the hybrid FE-GPU-EVPCUFFT model simulating the compression of the cube FE model with #4 elements of type C3D8 embedding microstructural cell of $64\times4\times64$ resolution. The benchmark is performed on workstation #2 utilizing #1, #2, and #4 Nvidia Tesla K80 GPUs with #4 MPI processes for all three cases. Results indicate almost perfect strong scalability of the multi-GPU utilization.

8. Summary and conclusions

The paper presented the first GPU-enabled parallel implementation of the EVPFFT full-field crystal plasticity solver in the implicit FEM, leveraging a hybrid CPU-GPU hardware. The FE solver utilizes MPI for domain decomposition while the underlying microstructural RVE runs on multiple GPUs concurrently. The implementation is referred to as full-field^2 FE-GPU-EVPCUFFT. GPU-supported multiscale simulations are facilitated by Nvidia HPC SDK compiler leveraging OpenACC and CUDA FFT libraries.

An analytical Jacobian is derived and implemented to facilitate the implicit coupling and ensure a fast order of convergence. The convergence rates of the multiscale implementation are verified as a function of time increment and embedded microstructural resolutions and regarded as satisfactory.

The FE-GPU-EVPCUFFT model is validated against the SA solver and experimental measurements for monotonic simple compression and texture evolution of FCC Cu under large deformation. Validation is extended for cyclic response of DP 590 steel with for a wide range of

strain amplitudes. Benchmarks corroborate the accuracy of the multiscale implementation. In these simulations, the multiscale model takes the advantage of a reversible dislocation density hardening law that enables dislocation annihilation and utilizes an intra-granular kinematic hardening back-stress law to capture the nonlinear unloading and Bauschinger effect upon load reversals. Predictions of tension-compression asymmetry are enabled leveraging the non-Schmid effects.

After validations, the full-field^2 model is applied to study several metal forming simulations. First, a uniaxial compression of a multi-element cylinder embedding a single copper crystal is simulated and the deformed shape and ovality of the cross section is compared to the experimental photographs of the deformed cylinder. Second, two simulations of four-point bending of clock-rolled Zr bars are performed in two different loading directions of parallel and perpendicular to the bending plane where the deformed shapes and cross sections are compared with the measured data. Finally, the unique capability of the full-field^2 implementation is demonstrated leveraging a beam bending of a martensitic-ferritic DP 1180 steel embedding high-resolution underlying microstructures per integration point. The results provide insights into variations of micromechanical fields within individual elements experiencing dissimilar states of deformation.

The paper concludes showing several performance benchmarks comparing the computational efficiency of the CPU-only FE-CPU-EVPFFTW model with the more advanced hybrid CPU-GPU FE-GPU-EVPCUFFT solver taking the advantage of multiple GPUs. The hybrid full-field^2 FE-GPU-EVPCUFFT spectral crystal plasticity package presented herein can improve computational efficiency for simulating microstructure-property-processing linkage of metallic materials.

Acknowledgments

This research was sponsored by the U.S. National Science Foundation and accomplished under the CMMI-1650641 and OIA-1757371 grants. A. E. also acknowledges support from the Dissertation Year Fellowship (DYF) and Summer TA Fellowship (STAF) programs at the University of New Hampshire.

Appendix A

Description of EVPFFT

A standard EVPFFT is based on a power-law relation between a plastic strain rate, $\dot{\boldsymbol{\varepsilon}}^{p}(\mathbf{x})$, and Cauchy stress, $\boldsymbol{\sigma}(\mathbf{x})$, through a superposition of shearing rates on slip systems, N [89, 90]

$$\dot{\boldsymbol{\varepsilon}}^{p}(\mathbf{x}) = \sum_{s=1}^{N} \mathbf{P}^{s}(\mathbf{x})\,\dot{\gamma}^{s}(\mathbf{x}) = \dot{\gamma}_{0} \sum_{s=1}^{N} \mathbf{P}^{s}(\mathbf{x}) \left(\frac{\mathbf{P}^{s}(\mathbf{x}) \cdot \boldsymbol{\sigma}(\mathbf{x}) - \tau_{bs}{}^{s}(\mathbf{x})}{\tau_{c}{}^{s}(\mathbf{x})} \right)^{n}, \tag{A1a}$$

$$\mathbf{P}_{sc}^{s} = \frac{1}{2}\left(\mathbf{b}^{s} \otimes \mathbf{n}^{s} + \mathbf{n}^{s} \otimes \mathbf{b}^{s} \right), \tag{A1b}$$

$$\mathbf{P}_{ns}{}^{s} = c_{1}(\mathbf{t}^{s} \otimes \mathbf{b}^{s}) + c_{2}(\mathbf{t}^{s} \otimes \mathbf{n}^{s}) + c_{3}(\mathbf{n}^{s} \otimes \mathbf{n}^{s}) + c_{4}(\mathbf{t}^{s} \otimes \mathbf{t}^{s}) - (c_{3} + c_{4})(\mathbf{b}^{s} \otimes \mathbf{b}^{s}), \tag{A1c}$$

$$\mathbf{P}^{s} = \mathbf{P}_{sc}^{s} + \mathbf{P}_{ns}^{s}. \tag{A1d}$$

In the above equations, $\dot{\gamma}^{s}$ and $\tau_{c}{}^{s}$ are the shearing rate and the slip resistance, respectively. Furthermore, the parameter $\dot{\gamma}_{0}$ is a reference shearing rate (taken as 0.001 /s) and n is the power-law visco-plastic exponent chosen to be 20 to ensure proper selection of the active slip systems. The term $\tau_{bs}{}^{s}(\mathbf{x})$ is the slip-system level kinematic back-stress influencing the driving force to slip. The Burgers vector $\mathbf{b}^{s}$ and the slip system normal $\mathbf{n}^{s}$ with $\mathbf{t}^{s} = \mathbf{b}^{s} \times \mathbf{n}^{s}$ define the geometry of a slip system, s. Selection of slip systems is based on the crystal structure of the simulated materials varying from FCC which deforms by $\{110\}\langle\bar{1}11\rangle$ family/mode of slip systems to BCC structure deforming by $\{\bar{1}11\}\langle 110\rangle$ and $\{\bar{1}11\}\langle 211\rangle$ slip modes to HCP structure deforming by prismatic slip $\{\bar{1}100\}\langle\bar{1}\,\bar{1}20\rangle$, basal slip $\{0001\}\langle\bar{1}\,\bar{1}20\rangle$ and pyramidal slip $\{10\bar{1}1\}\langle\bar{1}\,\bar{1}23\rangle$ modes. It should be noted that positive s+ and negative s- directions per slip systems are considered separately. The onset of acitvation for these slip systems varies depends on the local crystal orientation relative to a loading direction and a value of the resistance to slip. However, in addition to loading projected on the glide plane and in the glide direction, the model can consider two orthogonal shear stress components and three normal stress components to influence the activation criterion. These aspects are referred to as the non-Schmid effects [73, 91-94] and help in predicting the tension-

compression asymmetry. The non-Schmid coefficients c_1, c_2, c_3, c_4 weight the non-Schmid contributions.

The elasto-plastic constitutive response is described using the Hooke's law

$$\boldsymbol{\sigma}^{t+\Delta t}(\mathbf{x}) = \mathbf{C}(\mathbf{x})\boldsymbol{\varepsilon}^{e,t+\Delta t}(\mathbf{x}) = \mathbf{C}(\mathbf{x})\left(\boldsymbol{\varepsilon}^{t+\Delta t}(\mathbf{x}) - \boldsymbol{\varepsilon}^{p,t}(\mathbf{x}) - \dot{\boldsymbol{\varepsilon}}^{p,t+\Delta t}(\mathbf{x}, \boldsymbol{\sigma}^{t+\Delta t})\Delta t\right), \tag{A2}$$

where $\boldsymbol{\sigma}(\mathbf{x})$ is the Cauchy stress, $\mathbf{C}(\mathbf{x})$ is the elastic stiffness tensor in the global (sample) frame obtained by applying crystal to sample transformations utilizing the crystal elastic constants, and $\boldsymbol{\varepsilon}(\mathbf{x})$, $\boldsymbol{\varepsilon}^e(\mathbf{x})$, and $\boldsymbol{\varepsilon}^p(\mathbf{x})$ denote the total, elastic, and plastic strains, respectively. Based on Eq. (A2), the total strain is

$$\boldsymbol{\varepsilon}^{t+\Delta t}(\mathbf{x}) = \mathbf{C}^{-1}(\mathbf{x})\boldsymbol{\sigma}^{t+\Delta t}(\mathbf{x}) + \boldsymbol{\varepsilon}^{p,t}(\mathbf{x}) + \dot{\boldsymbol{\varepsilon}}^{p,t+\Delta t}(\mathbf{x}, \boldsymbol{\sigma}^{t+\Delta t})\Delta t. \tag{A3}$$

After adding and subtracting the stiffness of a reference linear medium, $\mathbf{C}^0$, multiplied by the displacement gradient $u_{k,l}(\mathbf{x})$ from the Cauchy stress, we obtain [44]

$$\sigma_{ij}(\mathbf{x}) = \sigma_{ij}(\mathbf{x}) + C^0{}_{ijkl}u_{k,l}(\mathbf{x}) - C^0{}_{ijkl}u_{k,l}(\mathbf{x}). \tag{A4}$$

Furthermore, we can write

$$\sigma_{ij}(\mathbf{x}) = C^0{}_{ijkl}u_{k,l}(\mathbf{x}) + \phi_{ij}(\mathbf{x}), \tag{A5}$$

with

$$\phi_{ij}(\mathbf{x}) = \sigma_{ij}(\mathbf{x}) - C^0{}_{ijkl}u_{k,l}(\mathbf{x}), \tag{A6}$$

where the term $\phi_{ij}(\mathbf{x})$ represents the polarization field. After incorporating the equilibrium equation, $\sigma_{ij,j}(\mathbf{x}) = 0$, into Eq. (A5) we obtain

$$C^0{}_{ijkl}u_{k,lj}(\mathbf{x}) + \phi_{ij,j}(\mathbf{x}) = 0. \tag{A7}$$

Relying on Green's approach to solve the partial differential equations [95], Green's function $G_{km}(\mathbf{x})$ is associated to the displacement field $u_k(\mathbf{x})$ as

$$C^0{}_{ijkl}G_{km,lj}(\mathbf{x}-\mathbf{x}') + \delta_{im}\delta(\mathbf{x}-\mathbf{x}') = 0. \tag{A8}$$

The convolution theorem [96] is then used to obtain the local displacement gradient fluctuation tensor

$$\tilde{u}_{k,l}(\mathbf{x}) = \int_{R^3} G_{ki,jl}(\mathbf{x}-\mathbf{x}')\phi_{ij}(\mathbf{x}')\,d\mathbf{x}'. \tag{A9}$$

The strain field around the average strain value, E_{ij}, is

$$\varepsilon_{ij}(\mathbf{x}) = E_{ij} + FT^{-1}\left(sym\left(\hat{\Gamma}^0_{ijkl}(\mathbf{k})\right)\hat{\phi}_{kl}(\mathbf{k})\right), \tag{A10}$$

where the symbols FT^{-1} and "^" indicate inverse Fourier transforms and direct Fourier transform, respectively. $\mathbf{k}$ is a point (frequency) in the Fourier space. The tensor $\hat{\Gamma}^0_{ijkl}(\mathbf{k})$ is

$$\hat{\Gamma}^0_{ijkl}(\mathbf{k}) = -k_j k_l \hat{G}_{ik}(\mathbf{k});\ \hat{G}_{ik}(\mathbf{k}) = [C_{kjil}k_l k_j]^{-1}. \tag{A11}$$

An iterative procedure is used to obtain a solution for Eq. (A7). If we consider $e_{ij}^{(i)}$ and $\lambda_{ij}^{(i)}$ to be an initial guess for the strain and stress fields, respectively, we get from Eq. (A6)

$$\phi_{ij}^{(i)}(\mathbf{x}) = \lambda_{ij}^{(i)}(\mathbf{x}) - C^0_{ijkl}e_{kl}^{(i)}(\mathbf{x}). \tag{A12}$$

Eq. (A10) is then used for the next guess for the strain field

$$e_{ij}^{(i+1)}(\mathbf{x}) = E_{ij} + FT^{-1}\left(sym\left(\hat{\Gamma}^0_{ijkl}(\mathbf{k})\right)\hat{\phi}^{(i)}_{kl}(\mathbf{k})\right). \tag{A13}$$

To use the stress directly rather than the polarization, Eq. (A13) is revised as [97]

$$e_{ij}^{(i+1)}(\mathbf{x}) = E_{ij} + FT^{-1}(\hat{\bar{e}}_{ij}^{(i)} + sym(\hat{\Gamma}^0_{ijkl}(\mathbf{k}))\hat{\lambda}_{kl}^{(i)}(\mathbf{k})). \tag{A14}$$

An augmented Lagrangian scheme is used [98] to minimize the residual as a function of the stress, $\boldsymbol{\sigma}^{(i+1)}$, and strain, $\boldsymbol{\varepsilon}^{(i+1)}$

$$R_k(\boldsymbol{\sigma}^{(i+1)}) = \sigma_k^{(i+1)} + C_{kl}{}^0\varepsilon_l^{(i+1)}(\boldsymbol{\sigma}^{(i+1)}) - \lambda_k^{(i)} - C_{kl}{}^0 e_l^{(i+1)}, \tag{A15}$$

In Eq. (A15), the following notation is used for the symmetric tensors σ_{ij} and C_{ijkl}

$$\begin{aligned} &\sigma_{ij} \to \sigma_k\,, \quad k = 1,6 \\ &C_{ijkl} \to C_{kl}\,,\ k,l = 1,6. \end{aligned} \tag{A16}$$

Eq. (A15) can be solved using the Newton Raphson (NR) method as

$$\sigma_k^{(i+1,j+1)} = \sigma_k^{(i+1,j)} - \left(\frac{\partial R_k}{\partial \sigma_l}\Big|_{\boldsymbol{\sigma}^{(i+1,j)}}\right)^{-1} R_l(\boldsymbol{\sigma}^{(i+1,j)}), \tag{A17}$$

where $\sigma_k^{(i+1,j+1)}$ is the (j+1) trial for stress field $\sigma_k^{(i+1)}$. Note that "*j*" counts the NR stress iterations, while "*i*" counts the field equilibrium iterations. Using Eq. (A3), the Jacobian is

$$\frac{\partial R_k}{\partial \sigma_l}\Big|_{\boldsymbol{\sigma}^{(i+1,j)}} = \delta_{kl} + C_{kq}{}^{0} C_{ql}{}^{-1} + \Delta t\, C_{kq}{}^{0} \frac{\partial \dot{\varepsilon}_q{}^{p}}{\partial \sigma_l}\Big|_{\boldsymbol{\sigma}^{(i+1,j)}}. \tag{A18}$$

The term $\frac{\partial \dot{\varepsilon}_q{}^{p}}{\partial \sigma_l}\Big|_{\boldsymbol{\sigma}^{(i+1,j)}}$ is

$$\frac{\partial \dot{\varepsilon}_q{}^{p}}{\partial \sigma_l}\Big|_{\boldsymbol{\sigma}^{(i+1,j)}} \cong n\dot{\gamma}_0 \sum\nolimits_{s=1}^{N} \frac{P^s{}_q P^s{}_l}{\tau_c{}^{s}\left(\boldsymbol{\sigma}^{(i+1,j)}(\mathbf{x})\right)} \left(\frac{\mathbf{P}^s(\mathbf{x}) \cdot \boldsymbol{\sigma}(\mathbf{x}) - \tau_{bs}{}^{s}}{\tau_c{}^{s}\left(\boldsymbol{\sigma}^{(i+1,j)}(\mathbf{x})\right)} \right)^{n-1}. \tag{A19}$$

Incorporating Eq. (A19) into Eq. (A18) gives

$$\frac{\partial R_k}{\partial \sigma_l}\Big|_{\boldsymbol{\sigma}^{(i+1,j)}} \cong \delta_{kl} + C_{kq}{}^{0} C_{ql}{}^{-1} + (\Delta t\, n\, \dot{\gamma}_0) C_{kq}{}^{0} \sum\nolimits_{s=1}^{N} \frac{P^s{}_q P^s{}_l}{\tau_c{}^{s}\left(\boldsymbol{\sigma}^{(i+1,j)}(\mathbf{x})\right)} \left(\frac{\mathbf{P}^s(\mathbf{x}) \cdot \boldsymbol{\sigma}(\mathbf{x}) - \tau_{bs}{}^{s}}{\tau_c{}^{s}\left(\boldsymbol{\sigma}^{(i+1,j)}(\mathbf{x})\right)} \right)^{n-1}. \tag{A20}$$

Minimizing the residual R_k by the NR's iterations, the solution for stress is obtained at each FFT voxel as the next trial for Eq. (A12) and Eq. (A14). The procedure continues until the convergence to TOL_{NR} is reached for each crystal stress

$$\frac{\left(\sigma_k{}^{(i+1,j+1)} - \sigma_k{}^{(i+1,j)}\right)\left(\sigma_k{}^{(i+1,j+1)} - \sigma_k{}^{(i+1,j)}\right)}{\sqrt{\lambda_{ij}{}^{(i)} \lambda_{ij}{}^{(i)}}} \prec TOL_{NR} = 10^{-6}. \tag{A21}$$

Given the power-law exponent *n* and the tolerance, the total number of iterations for stress is in the range between 3 and 6.

The stress and strain field tolerances (TOL_{stress_field} , TOL_{strain_field}) after solving Eq. (A15) are

$$\frac{\left\langle \left(\sigma_k{}^{(i+1)} - \lambda_k{}^{(i)}\right)\left(\sigma_k{}^{(i+1)} - \lambda_k{}^{(i)}\right) \right\rangle}{\sqrt{\frac{3}{2} \Sigma'_{ij}{}^{(i)} \Sigma'_{ij}{}^{(i)}}} \prec TOL_{stress\_field} = 10^{-6}) \tag{A22a}$$

$$\frac{\left\langle \left(\varepsilon_k{}^{(i+1)} - e_k{}^{(i)}\right)\left(\varepsilon_k{}^{(i+1)} - e_k{}^{(i)}\right) \right\rangle}{\sqrt{\frac{2}{3} E'_{ij}{}^{(i)} E'_{ij}{}^{(i)}}} \prec TOL_{strain\_field} = 10^{-6} \tag{A22b}$$

where $\langle\rangle$ represents the volume average. Tensors Σ'_{ij} and E'_{ij} are the deviatoric stress and the plastic strain of the homogenized polycrystal by averaging over voxels

$$\Sigma_{ij}^{'} = \frac{\sum_{X,Y,Z}\left(\sigma_{ij}^{'}(\mathbf{x})\right)}{N_1 \times N_2 \times N_3};\ N_1, N_2, N_3 = \#\,of\ voxels\ in\ X, Y, Z,$$
$$E_{ij}^{'} = \frac{\sum_{X,Y,Z}\left(\varepsilon_{ij}^{p}(\mathbf{x})\right)}{N_1 \times N_2 \times N_3}. \tag{A23}$$

The total number of iterations for obtaining the field solution varies between 10 and 25 for the selected tolerances in Eq. (A22).

Slip-system level reversible dislocation density hardening

A dislocation density-based hardening law is implemented in the EVPFFT model in earlier works [48, 73]. In the description, s and s' denote the slip systems interacting with each other. The rate of dislocation density evolution is a thermally activated process, dependent on the temperature and strain rate. The law considers bi-directional motion of dislocations on a given plane allowing dislocations to annihilate when moved in the opposite direction. The evolution of slip resistance (the critical resolves shear stress, CRSS) per slip system is driven by the following three contributing terms

$$\tau_c^s = \tau_0^s + \tau_{forest}^s + \tau_{deb}. \tag{A24a}$$

The first term, τ_0^s, contributes to the initial value (i.e., initial yield point) which does not evolve and itself consists of three terms

$$\tau_0^s = \tau_0 + \tau_{0,HP}^s + \tau_{0,forest}, \tag{A24b}$$

where τ_0 is the frictional term embedding the effects of solid solution and precipitate hardening depending on the material (i.e. Peierls stress) [73], $\tau_{0,HP}^s$ represents the Hall-Petch contribution of grain size and shape (barrier effects), and $\tau_{0,forest}$ denotes the forest dislocation density content distributed through the initial material. The Hall-Petch term is defined as [99]

$$\tau^s_{0,HP} = \frac{H\ \mu\ \sqrt{b}}{\sqrt{d^s_{mfp}}},\ d^s_{mfp} = \frac{2}{\sqrt{\left(\frac{\hat{b}^s_x}{a}\right)^2 + \left(\frac{\hat{b}^s_y}{b}\right)^2 + \left(\frac{\hat{b}^s_z}{c}\right)^2}}, \tag{A25}$$

where b, μ, H are the Burgers vector, shear modulus, and Hall-Petch calibration coefficient, respectively. The quantities a,b,c are the ellipsoidal dimensions representing a grain and $\hat{b}^s_x,\hat{b}^s_y,\hat{b}^s_z$ denote a unit vector in the Burgers direction in the grain to estimate the dislocation mean-free-path, d^s_{mfp}. While the term is not always calibrated due to lack of mechanical data with variable grain size, the model provides an opportunity to account for it.

Contribution of statistically stored dislocation density, τ^s_{forest} , to the total slip resistance, is defined using

$$\tau^s_{forest} = b\ \chi\ \mu\ \sqrt{\rho^s_{tot} + L\sum_{s\neq s'}\rho^{s'}_{tot}}, \tag{A26}$$

where χ denotes the system interaction parameter usually taken to be 0.9 [100-102], while L is the latent hardening coefficient, usually set to 1.05 [48, 103]. The total dislocation density ρ^s_{tot} is described based on two contributions of forward (i.e., non-reversible) and reversible dislocation populations [104], to possibly capture a deformation path dependence in the hardening law and dislocation dissolution upon load reversal. Implementation of the concept necessitates the consideration of two slip system directions with opposite Burger directions (i.e. s^+, s^-) on a given plane attached to the reversible dislocation populations. Consequently, the total dislocation density is

$$\rho^s_{tot} = \rho^s_{forw} + \rho^{s+}_{rev} + \rho^{s-}_{rev}, \tag{A27}$$

with ρ^s_{forw} denotes the forward dislocation density and ρ^{s+}_{rev}, ρ^{s-}_{rev} imply reversible populations correspond to the slip directions s^+ and s^-, respectively. The forward dislocation density population is evolved as a function of rate of storage and the rate of recovery as [48, 73, 101, 103]

$$\frac{\partial\rho^s_{forw}}{\partial\gamma^s} = (1-p)k_1\sqrt{\rho^s_{forw}+\rho^{s+}_{rev}} - k_2(\dot{\varepsilon},T)\,\rho^s_{forw}\Big]\ for\ d\gamma^{s+} > 0\ , \tag{A28a}$$

$$\frac{\partial \rho^{s}_{forw}}{\partial \gamma^{s}} = (1-p)k_1\sqrt{\rho^{s}_{forw}+\rho^{s-}_{rev}} - k_2(\dot{\varepsilon},T)\,\rho^{s}_{forw}\Big] for\ d\gamma^{s-}>0\ , \tag{A28b}$$

where k_1 is a calibration constant driving the rate of generation, while k_2 drives the rate of recovery [99]. The constant p is the shear reversibility in the range between 0 and 1 to separate the fraction of forward and reversible populations. For low to moderate strain levels, $p=1$ [104] meaning that dislocations can glide in the opposite direction at the loading reversal. The reversible populations depend on the direction of shearing according to

$$\left.\begin{aligned}\frac{\partial \rho^{s+}_{rev}}{\partial \gamma^{s}} &= pk_1\sqrt{\rho^{s}_{forw}+\rho^{s+}_{rev}} - k_2(\dot{\varepsilon},T)\rho^{s+}_{rev},\\ \frac{\partial \rho^{s-}_{rev}}{\partial \gamma^{s}} &= -k_1\sqrt{\rho^{s}_{forw}+\rho^{s+}_{rev}}\left(\frac{\rho^{s-}_{rev}}{\rho^{s}_{tot}}\right)^{m},\end{aligned}\right\} for\, d\gamma^{s+}>0 \tag{A29}$$

$$\left.\begin{aligned}\frac{\partial \rho^{s-}_{rev}}{\partial \gamma^{s}} &= pk_1\sqrt{\rho^{s}_{forw}+\rho^{s-}_{rev}} - k_2(\dot{\varepsilon},T)\rho^{s-}_{rev},\\ \frac{\partial \rho^{s+}_{rev}}{\partial \gamma^{s}} &= -k_1\sqrt{\rho^{s}_{forw}+\rho^{s-}_{rev}}\left(\frac{\rho^{s+}_{rev}}{\rho^{s}_{tot}}\right)^{m},\end{aligned}\right\} for\, d\gamma^{s-}>0 \tag{A30}$$

$$\rho^{s+}_{rev}\big|_{t=0}=0\ ;\ \rho^{s-}_{rev}\big|_{t=0}=0\ ;\ \rho^{s}_{forw}\big|_{t=0}=\rho^{s}_{initial,}$$

where, the exponent m controls the rate of dislocation recombination and is usually taken as 0.5 [105]. The parameter $k_2(\dot{\varepsilon},T)$ is defined as a function of temperature and strain rate as follows

$$k_2 = \frac{k_1\chi b}{g}\left[1-\frac{k_B T}{D\ (b\ )^3}\ln\left(\frac{\dot{\varepsilon}}{\dot{\varepsilon}_0}\right)\right], \tag{A31}$$

where g is an effective activation enthalpy established by calibration, k_B is the Boltzmann constant, $\dot{\varepsilon}_0$ is a reference rate of strain set to 10^7 s^{-1}, and D is the drag stress also established by calibration.

The last term of slip resistance, τ_{deb}, is a consequence of the debris population [106] and is defined as

$$\tau_{deb} = 0.086\,\mu\ b\sqrt{\rho_{deb}}\ \log\left(\frac{1}{b\sqrt{\rho_{deb}}}\right), \tag{A32}$$

where, ρ_{deb} is the debris dislocation density, which evolves using

$$d\rho_{deb} = \sum_s q\ b\sqrt{\rho_{deb}}k_2(\dot{\varepsilon},T)\rho_{tot}^s\left|d\gamma^s\right|. \tag{A33}$$

In Eq. (A33), the parameter q is a debris-related fitting constant separating the portion of removed dislocations [$\frac{\partial\rho^s{}_{rem,tot}}{\partial\gamma^s} = k_2(\dot{\varepsilon},T)\rho_{tot}^s$] from the debris.

Back-stress law

A phenomenological approach to intra-granular kinematic back-stress effects has been implemented in EVPFFT [48, 73, 107]. The inter-granular effects are explicitly accounted for. The approach is a computationally efficient simplification of the back-stress estimation possible using strain gradient plasticity formulations [108]. Consistent with the self-internal back-stress formulations [109], the back-stress either assists or hinders the resolved shear stresses on the slip systems [110, 111]. Specifically, while $\tau_{bs}^{s^+}$ acts in the direction opposing the driving stress on s^+, i.e. $\mathbf{P}^{s^+}\cdot\boldsymbol{\sigma} - \tau_{bs}^{s^+} = \tau_c^s$, meaning that $\tau_{bs}^{s^+}$ lowers the driving stress, $\tau_{bs}^{s^-}$ benefits the driving stress on the slip system s^-: $\mathbf{P}^{s^-}\cdot\boldsymbol{\sigma} - \tau_{bs}^{s^-} = \tau_c^s$. The back-stress evolves once the grains start deforming plastically ($\gamma^{s+/-} > 0$) and can saturate [48]

$$\left.\begin{aligned}
&\overset{*}{\gamma}{}^{s+/-} = \frac{-1}{\nu}\ln\left(1-\frac{\tau_{bs,sys}^{s+/-}}{\tau_{bs}^{sat}}\right)\\
&\tau_{bs,sys}^{s+/-} = \tau_{bs}^{sat}\left(1-\exp\left(-\nu\left(\overset{*}{\gamma}{}^{s+/-} + d\gamma^{s+/-}\right)\right)\right),\\
&\tau_{bs,sys}^{s-/+} = -A_{bs}\tau_{bs,sys}^{s+/-},
\end{aligned}\right\}\ for\ \tau_{bs,sys}^{s+/-} > 0, \tag{A34a}$$

$$\left.\begin{aligned}
&\overset{**}{\gamma}{}^{s+/-} = -\gamma_b\ln\left(\frac{\tau_{bs}^{sat}-\tau_{bs,sys}^{s+/-}}{(A_{bs}+1)\tau_{bs}^{sat}}\right)\\
&\tau_{bs,sys}^{s+/-} = -(A_{bs}+1)\tau_{bs,sys}^{s+/-}\exp\left(-\frac{\overset{**}{\gamma}{}^{s+/-} + d\gamma^{s+/-}}{\gamma_b}\right) + \tau_{bs}^{sat},\\
&\tau_{bs,sys}^{s-/+} = -\frac{1}{A_{bs}}\tau_{bs,sys}^{s+/-},
\end{aligned}\right\}\ for\ \tau_{bs,sys}^{s+/-} < 0, \tag{A34b}$$

where $\tau_{bs,sys}^{s+/-}$ is the back-stress for the two opposite directions, s^{+} and s^{-}. A_{bs} and ν are the back-stress calibration parameters. The calibration parameter A_{bs} controls the asymmetry of non-linear unloading [112]. Finally, the calibration parameter τ_{bs}^{sat} is the saturation limit for evolution of the back-stress. The proposed back-stress law enables predictions of non-linear unloading and Bauschinger effect (BE) upon load reversal under cyclic loadings and during strain-path-changes [48].

Appendix B

UMAT variables are shown in Fig. B1. The array size NTENS is set to 6 e.g. ($\Sigma_{xx}, \Sigma_{yy}, \Sigma_{zz}, \Sigma_{xy}, \Sigma_{xz}, \Sigma_{yz}$) using Abaqus' notation, which does not conform to Voigt. While the order in Abaqus notation is (11, 22, 33, 12, 13,23), the order in EVPFFT corresponds to the Voigt order (11, 22, 33, 23, 13, 23). Therefore, proper substitution in the components 12 and 23 was necessary. The number of direct stress components are defined by NDI, which is 3 ($\Sigma_{xx}, \Sigma_{yy}, \Sigma_{zz}$). The FE element number and integration point ID are defined by NOEL and NPT, respectively. The variable TIME is a vector of two components where TIME (1) and TIME (2) imply the values of step time and total time at the beginning of the current increment, respectively. The increment number and time increment value are KINC and DTIME, respectively.

```
SUBROUTINE UMAT(STRESS,STATEV,DDSDDE,SSE,SPD,SCD,RPL,DDSDDT,DRPLDE,DRPLDT,
STRAN, DSTRAN,TIME,DTIME,TEMP,DTEMP, PREDEF,DPRED,CMNAME, NDI,NSHR,NTENS,
NSTATV,PROPS,NPROPS,COORDS,DROT,PNEWDT,CELENT,DFGRD0,DFGRD1,NOEL,NPT,
LAYER,KSPT,KSTEP,KINC)
```

Fig. B1. UMAT subroutine interface for incorporating constitutive laws in Abaqus.

Appendix C

Fig. C1. Shown the environment file adjusted for UMAT simulations on GPUs using Nvidia HPC SDK compiler.

```
#abaqus_v6.env : Modified to suit GPU utilization by Nvidia HPC SDK compiler
fortCmd = "pgf90" # <-- Fortran compiler

compile_fortran = [fortCmd, '-V','-Mpreprocess','-Minfo=accel,mp', '-Mextend'
,'-fast', '-tp=px' , '-c','-fPIC','-Mr8', '-DABQ_LNX86_64', '-DABQ_FORTRAN', '-
I%I', '%P', '-DGPU','-DUSEACCPARALLEL', '-DCollapse', '-DGPU_NR', '-
DGPU_Update_Schmid', '-DGPU_DGrain', '-DGPU_Inverse_the_Greens', '-
DGPU_update_orient', '-DGPU_update_disgrad', '-DGPU_step_vm_calc', '-
DGPU_Copy_disgrad_to_velgrad', '-DGPU_Get_Smacro' ,'-DGPU_fft_dim_init_xk_gb'
,'-DGPU_init_sg', '-DGPU_init_ept' ,'-DGPU_fft3d_c2r', '-DGPU_fft3d_r2c', '-
DGPU_step_update_disgrad_etc' ,'-UGPU_DD_Harden' ,'-Mcuda', '-
ta=tesla:cc70,cuda11.1' ,'-ta=tesla:cc70,cuda11.1,pinned', '-
ta=tesla:cc70,cuda11.1,fastmath','-DUSE_CUFFT' ,'-DUSE_CUFFT_SERIAL', '-
DFFT_choice=fft_cufft', '-Mcudalib=cufft' ,'-DFFT_MPI_MODULE=fft_cufft_serial',
'-I/opt/nvidia/hpc_sdk/Linux_x86_64/20.11/cuda/11.1/include/', '-
L/opt/nvidia/hpc_sdk/Linux_x86_64/20.11/cuda/11.1/lib64/', '-Mr8' ,'-Mextend']

link_sl = [fortCmd, '-shared', '%E','-o', '%U', '%F', '%A', '%L','%B', '-
DGPU','-DUSEACCPARALLEL','-DCollapse', '-DGPU_NR', '-DGPU_Update_Schmid', '-
DGPU_DGrain', '-DGPU_Inverse_the_Greens', '-DGPU_update_orient', '-
DGPU_update_disgrad', '-DGPU_step_vm_calc', '-DGPU_Copy_disgrad_to_velgrad', '-
DGPU_Get_Smacro' ,'-DGPU_fft_dim_init_xk_gb' ,'-DGPU_init_sg', '-DGPU_init_ept'
,'-DGPU_fft3d_c2r', '-DGPU_fft3d_r2c', '-DGPU_step_update_disgrad_etc' ,'-
UGPU_DD_Harden' ,'-Mcuda', '-ta=tesla:cc70,cuda11.1' ,'-
ta=tesla:cc70,cuda11.1,pinned', '-ta=tesla:cc70,cuda11.1,fastmath','-
DUSE_CUFFT' ,'-DUSE_CUFFT_SERIAL', '-DFFT_choice=fft_cufft', '-Mcudalib=cufft'
,'-DFFT_MPI_MODULE=fft_cufft_serial', '-
I/opt/nvidia/hpc_sdk/Linux_x86_64/20.11/cuda/11.1/include/', '-
L/opt/nvidia/hpc_sdk/Linux_x86_64/20.11/cuda/11.1/lib64/', '-Mr8' ,'-Mextend',
'/home/adnan/FE_GPU_EVPCUFFT/fftc_cufft.o']
```

Fig. C1. Abaqus environment file "Abaqus_v6.env" for UMAT simulations on GPUs using Nvidia HPC SDK compiler.

Data availability

The raw/processed data required to reproduce these findings cannot be shared at this time due to technical or time limitations.

References

[1] W.F. Hosford, R.M. Caddell. Metal forming: mechanics and metallurgy, Cambridge University Press, New York, USA, 2011.

[2] U.F. Kocks, C.N. Tomé, H.-R. Wenk. Texture and Anisotropy, Cambridge University Press, Cambridge, UK, 1998.

[3] M. Jahedi, M.H. Paydar, S. Zheng, I.J. Beyerlein, M. Knezevic. Texture evolution and enhanced grain refinement under high-pressure-double-torsion, Mater. Sci. Eng. A 611 (2014) 29-36.

[4] M. Knezevic, C.M. Poulin, X. Zheng, S. Zheng, I.J. Beyerlein. Strengthening of alloy AA6022-T4 by continuous bending under tension, Mater. Sci. Eng. A 758 (2019) 47-55.
[5] R.E. Marki, K.A. Brindley, R.J. McCabe, M. Knezevic. Crystal mechanics-based thermo-elastic constitutive modeling of orthorhombic uranium using generalized spherical harmonics and first-order bounding theories, Journal of Nuclear Materials 560 (2022) 153472.
[6] F. Barlat, H. Aretz, J.W. Yoon, M. Karabin, J. Brem, R. Dick. Linear transfomation-based anisotropic yield functions, Int. J. Plast. 21 (2005) 1009-1039.
[7] Z. Feng, S.-Y. Yoon, J.-H. Choi, T.J. Barrett, M. Zecevic, F. Barlat, M. Knezevic. A comparative study between elasto-plastic self-consistent crystal plasticity and anisotropic yield function with distortional hardening formulations for sheet metal forming, Mechanics of Materials 148 (2020) 103422.
[8] M. Knezevic, R.A. Lebensohn, O. Cazacu, B. Revil-Baudard, G. Proust, S.C. Vogel, M.E. Nixon. Modeling bending of α-titanium with embedded polycrystal plasticity in implicit finite elements, Mater. Sci. Eng. A 564 (2013) 116-126.
[9] N.R. Barton, J.V. Bernier, R.A. Lebensohn, D.E. Boyce. The use of discrete harmonics in direct multi-scale embedding of polycrystal plasticity, Computer Methods in Applied Mechanics and Engineering 283 (2015) 224-242.
[10] N.R. Barton, J. Knap, A. Arsenlis, R. Becker, R.D. Hornung, D.R. Jefferson. Embedded polycrystal plasticity and adaptive sampling, Int. J. Plast. 24 (2008) 242-266.
[11] J. Segurado, R.A. Lebensohn, J. Llorca, C.N. Tomé. Multiscale modeling of plasticity based on embedding the viscoplastic self-consistent formulation in implicit finite elements, Int. J. Plast. 28 (2012) 124-140.
[12] M. Zecevic, M. Knezevic. A new visco-plastic self-consistent formulation implicit in dislocation-based hardening within implicit finite elements: Application to high strain rate and impact deformation of tantalum, Computer Methods in Applied Mechanics and Engineering 341 (2018) 888-916.
[13] T.J. Barrett, M. Knezevic. Deep drawing simulations using the finite element method embedding a multi-level crystal plasticity constitutive law: Experimental verification and sensitivity analysis, Computer Methods in Applied Mechanics and Engineering 354 (2019) 245-270.
[14] M. Knezevic, H.F. Al-Harbi, S.R. Kalidindi. Crystal plasticity simulations using discrete Fourier transforms, Acta. Mater. 57 (2009) 1777-1784.
[15] M. Knezevic, S.R. Kalidindi, D. Fullwood. Computationally efficient database and spectral interpolation for fully plastic Taylor-type crystal plasticity calculations of face-centered cubic polycrystals, International Journal of Plasticity 24 (2008) 1264-1276.
[16] G.I. Taylor. Plastic strain in metals, J. Inst. Metals 62 (1938) 307-324.
[17] P. Van Houtte, S. Li, O. Engler. Taylor-Type Homogenization Methods for Texture and Anisotropy, Continuum Scale Simulation of Engineering Materials: Fundamentals–Microstructures–Process Applications (2004) 459-472.
[18] H.F. Al-Harbi, M. Knezevic, S.R. Kalidindi. Spectral approaches for the fast computation of yield surfaces and first-order plastic property closures for polycrystalline materials with cubic-triclinic textures, Computers, Materials, & Continua 15 (2010) 153-172.
[19] G.I. Taylor, H. Quinney. The plastic distortion of metals, Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character 230 (1931) 323-362.

[20] M. Knezevic, S.R. Kalidindi. Fast computation of first-order elastic-plastic closures for polycrystalline cubic-orthorhombic microstructures, Comput. Mater. Sci. 39 (2007) 643-648.
[21] R. Lebensohn, C. Tomé, P.P. Castaneda. Self-consistent modelling of the mechanical behaviour of viscoplastic polycrystals incorporating intragranular field fluctuations, Philosophical Magazine 87 (2007) 4287-4322.
[22] M. Zecevic, M. Knezevic. A dislocation density based elasto-plastic self-consistent model for the prediction of cyclic deformation: Application to AA6022-T4, International Journal of Plasticity 72 (2015) 200-217.
[23] M. Zecevic, M. Knezevic, I.J. Beyerlein, C.N. Tomé. An elasto-plastic self-consistent model with hardening based on dislocation density, twinning and de-twinning: application to strain path changes in HCP metals, Materials Science and Engineering: A 638 (2015) 262-274.
[24] R.A. Lebensohn, C. Tomé. A self-consistent anisotropic approach for the simulation of plastic deformation and texture development of polycrystals: application to zirconium alloys, Acta metallurgica et materialia 41 (1993) 2611-2624.
[25] M. Zecevic, K.C. Bennett, D.J. Luscher, R.A. Lebensohn. New self-consistent homogenization for thermo-elastic polycrystals with imperfect interfaces, Mechanics of Materials 155 (2021) 103651.
[26] M. Knezevic, M. Zecevic, I.J. Beyerlein, J.F. Bingert, R.J. McCabe. Strain rate and temperature effects on the selection of primary and secondary slip and twinning systems in HCP Zr, Acta. Mater. 88 (2015) 55-73.
[27] M. Knezevic, J.S. Carpenter, M.L. Lovato, R.J. McCabe. Deformation behavior of the cobalt-based superalloy Haynes 25: Experimental characterization and crystal plasticity modeling, Acta. Mater. 63 (2014) 162-168.
[28] Z. Feng, M. Zecevic, M. Knezevic. Stress-assisted ($\gamma \rightarrow \alpha'$) and strain-induced ($\gamma \rightarrow \varepsilon \rightarrow \alpha'$) phase transformation kinetics laws implemented in a crystal plasticity model for predicting strain path sensitive deformation of austenitic steels, Int. J. Plast. 136 (2021) 102807.
[29] M. Knezevic, R.J. McCabe, R.A. Lebensohn, C.N. Tomé, C. Liu, M.L. Lovato, B. Mihaila. Integration of self-consistent polycrystal plasticity with dislocation density based hardening laws within an implicit finite element framework: application to low-symmetry metals, Journal of the Mechanics and Physics of Solids 61 (2013) 2034-2046.
[30] M. Zecevic, M. Knezevic. An implicit formulation of the elasto-plastic self-consistent polycrystal plasticity model and its implementation in implicit finite elements, Mechanics of Materials 136 (2019) 103065.
[31] M. Zecevic, I.J. Beyerlein, M. Knezevic. Coupling elasto-plastic self-consistent crystal plasticity and implicit finite elements: applications to compression, cyclic tension-compression, and bending to large strains, International Journal of Plasticity 93 (2017) 187-211.
[32] W.G. Feather, D.J. Savage, M. Knezevic. A crystal plasticity finite element model embedding strain-rate sensitivities inherent to deformation mechanisms: Application to alloy AZ31, Int. J. Plast. 143 (2021) 103031.
[33] W.G. Feather, S. Ghorbanpour, D.J. Savage, M. Ardeljan, M. Jahedi, B.A. McWilliams, N. Gupta, C. Xiang, S.C. Vogel, M. Knezevic. Mechanical response, twinning, and texture evolution of WE43 magnesium-rare earth alloy as a function of strain rate: Experiments and multi-level crystal plasticity modeling, Int. J. Plast. 120 (2019) 180-204.
[34] M. Zecevic, M. Knezevic. Modeling of Sheet Metal Forming Based on Implicit Embedding of the Elasto-Plastic Self-Consistent Formulation in Shell Elements: Application to Cup Drawing of AA6022-T4, JOM 69 (2017) 922-929.

[35] L. Anand. Single-crystal elasto-viscoplasticity: application to texture evolution in polycrystalline metals at large strains, Computer Methods in Applied Mechanics and Engineering 193 (2004) 5359-5383.
[36] M. Ardeljan, I.J. Beyerlein, M. Knezevic. A dislocation density based crystal plasticity finite element model: application to a two-phase polycrystalline HCP/BCC composites, J. Mech. Phys. Solids 66 (2014) 16-31.
[37] M. Ardeljan, I.J. Beyerlein, M. Knezevic. Effect of dislocation density-twin interactions on twin growth in AZ31 as revealed by explicit crystal plasticity finite element modeling, International Journal of Plasticity 99 (2017) 81-101.
[38] S.R. Kalidindi, C.A. Bronkhorst, L. Anand. Crystallographic texture evolution in bulk deformation processing of FCC metals, J. Mech. Phys. Solids 40 (1992) 537-569.
[39] M. Knezevic, A. Levinson, R. Harris, R.K. Mishra, R.D. Doherty, S.R. Kalidindi. Deformation twinning in AZ31: Influence on strain hardening and texture evolution, Acta. Mater. 58 (2010) 6230-6242.
[40] M. Knezevic, M.R. Daymond, I.J. Beyerlein. Modeling discrete twin lamellae in a microstructural framework, Scr. Mater. 121 (2016) 84-88.
[41] M. Ardeljan, D.J. Savage, A. Kumar, I.J. Beyerlein, M. Knezevic. The plasticity of highly oriented nano-layered Zr/Nb composites, Acta. Mater. 115 (2016) 189-203.
[42] M. Ardeljan, M. Knezevic, T. Nizolek, I.J. Beyerlein, N.A. Mara, T.M. Pollock. A study of microstructure-driven strain localizations in two-phase polycrystalline HCP/BCC composites using a multi-scale model, Int. J. Plast. 74 (2015) 35-57.
[43] R.A. Lebensohn. N-site modeling of a 3D viscoplastic polycrystal using fast Fourier transform, Acta materialia 49 (2001) 2723-2737.
[44] R.A. Lebensohn, A.K. Kanjarla, P. Eisenlohr. An elasto-viscoplastic formulation based on fast Fourier transforms for the prediction of micromechanical fields in polycrystalline materials, International Journal of Plasticity 32 (2012) 59-69.
[45] S.-B. Lee, R. Lebensohn, A.D. Rollett. Modeling the viscoplastic micromechanical response of two-phase materials using Fast Fourier Transforms, International Journal of Plasticity 27 (2011) 707-727.
[46] S. Berbenni, V. Taupin, R.A. Lebensohn. A fast Fourier transform-based mesoscale field dislocation mechanics study of grain size effects and reversible plasticity in polycrystals, J. Mech. Phys. Solids 135 (2020) 103808.
[47] A. Eghtesad, M. Zecevic, R.A. Lebensohn, R.J. McCabe, M. Knezevic. Spectral database constitutive representation within a spectral micromechanical solver for computationally efficient polycrystal plasticity modelling, Computational Mechanics 61 (2018) 89-104.
[48] A. Eghtesad, M. Knezevic. High-performance full-field crystal plasticity with dislocation-based hardening and slip system back-stress laws: Application to modeling deformation of dual-phase steels, J. Mech. Phys. Solids 134 (2020) 103750.
[49] A. Eghtesad, T.J. Barrett, K. Germaschewski, R.A. Lebensohn, R.J. McCabe, M. Knezevic. OpenMP and MPI implementations of an elasto-viscoplastic fast Fourier transform-based micromechanical solver for fast crystal plasticity modeling, Advances in Engineering Software 126 (2018) 46-60.
[50] A. Eghtesad, K. Germaschewski, R.A. Lebensohn, M. Knezevic. A multi-GPU implementation of a full-field crystal plasticity solver for efficient modeling of high-resolution microstructures, Computer Physics Communications (2020) 107231.

[51] A. Eghtesad, K. Germaschewski, I.J. Beyerlein, A. Hunter, M. Knezevic. Graphics processing unit accelerated phase field dislocation dynamics: Application to bi-metallic interfaces, Advances in Engineering Software 115 (2018) 248-267.
[52] A. Eghtesad, M. Knezevic. A new approach to fluid–structure interaction within graphics hardware accelerated smooth particle hydrodynamics considering heterogeneous particle size distribution, Computational Particle Mechanics 5 (2018) 387-409.
[53] B. Mihaila, M. Knezevic, A. Cardenas. Three orders of magnitude improved efficiency with high-performance spectral crystal plasticity on GPU platforms, International Journal for Numerical Methods in Engineering 97 (2014) 785-798.
[54] D.J. Savage, M. Knezevic. Computer implementations of iterative and non-iterative crystal plasticity solvers on high performance graphics hardware, Computational Mechanics 56 (2015) 677–690.
[55] M. Knezevic, D.J. Savage. A high-performance computational framework for fast crystal plasticity simulations, Comput. Mater. Sci. 83 (2014) 101-106.
[56] F. Han, F. Roters, D. Raabe. Microstructure-based multiscale modeling of large strain plastic deformation by coupling a full-field crystal plasticity-spectral solver with an implicit finite element solver, International Journal of Plasticity 125 (2020) 97-117.
[57] F. Roters, M. Diehl, P. Shanthraj, P. Eisenlohr, C. Reuber, S.L. Wong, T. Maiti, A. Ebrahimi, T. Hochrainer, H.-O. Fabritius. DAMASK–The Düsseldorf Advanced Material Simulation Kit for modeling multi-physics crystal plasticity, thermal, and damage phenomena from the single crystal up to the component scale, Computational Materials Science 158 (2019) 420-478.
[58] K.-J. Bathe. Finite element procedures, Englewood Cliffs, N.J.: Prentice Hall, 1996.
[59] M. Zecevic, I.J. Beyerlein, R.J. McCabe, B.A. McWilliams, M. Knezevic. Transitioning rate sensitivities across multiple length scales: Microstructure-property relationships in the Taylor cylinder impact test on zirconium, Int. J. Plast. 84 (2016) 138-159.
[60] M. Messner, A. Beaudoin, R. Dodds. Consistent crystal plasticity kinematics and linearization for the implicit finite element method, Engineering Computations (2015).
[61] F. Harewood, P. McHugh. Comparison of the implicit and explicit finite element methods using crystal plasticity, Computational Materials Science 39 (2007) 481-494.
[62] J. Nagtegaal, F. Veldpaus. On the implementation of finite strain plasticity equations in a numerical model. Numerical methods in industrial forming processes:[papers presented at the international conference, held at Swansea, UK, 12-16 July 1982]/Ed. JFT Pittman: Wiley-Interscience, 1984. p.351-371.
[63] D. Kirk. NVIDIA CUDA software and GPU parallel computing architecture. ISMM, vol. 7, 2007. p.103-104.
[64] I. Buck. Gpu computing with nvidia cuda. ACM SIGGRAPH 2007 courses. 2007. pp. 6-es.
[65] 2011-2014 OpenACC.org, http://www.openacc-standard.org/.
[66] R. Farber. Parallel programming with OpenACC, Newnes, 2016.
[67] M.A. Groeber, M.A. Jackson. DREAM. 3D: a digital representation environment for the analysis of microstructure in 3D, Integrating materials and manufacturing innovation 3 (2014) 56-72.
[68] S.R. Kalidindi, L. Anand. Large deformation simple compression of a copper single crystal, Metall. Trans. A 24 (1993) 989-992.

[69] M.A. Meyers, K.K. Chawla. Mechanical Behavior of Materials., Prentice Hall, Upper Saddle River, New Jersey, 1998.
[70] M. Poschmann, M. Asta, D. Chrzan. Effect of non-Schmid stresses on< a>-type screw dislocation core structure and mobility in titanium, Computational Materials Science 161 (2019) 261-264.
[71] W. Roberts, J. Gong, A.J. Wilkinson, E. Tarleton. Tension–compression asymmetry of 〈c+a〉 slip in Ti–6Al, Scr. Mater. 178 (2020) 119-123.
[72] G. Kaschner, J. Bingert, C. Liu, M. Lovato, P. Maudlin, M. Stout, C. Tomé. Mechanical response of zirconium—II. Experimental and finite element analysis of bent beams, Acta Materialia 49 (2001) 3097-3108.
[73] A. Eghtesad, M. Knezevic. A full-field crystal plasticity model including the effects of precipitates: Application to monotonic, load reversal, and low-cycle fatigue behavior of Inconel 718, Mater. Sci. Eng. A 803 (2021) 140478.
[74] W. Gropp, W.D. Gropp, E. Lusk, A. Skjellum, A.D.F.E.E. Lusk. Using MPI: portable parallel programming with the message-passing interface, MIT press, 1999.
[75] N.T. Karonis, B. Toonen, I. Foster. MPICH-G2: A grid-enabled implementation of the message passing interface, Journal of Parallel and Distributed Computing 63 (2003) 551-563.
[76] W. Gropp, E. Lusk, N. Doss, A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard, Parallel computing 22 (1996) 789-828.
[77] D.W. Walker, J.J. Dongarra. MPI: a standard message passing interface, Supercomputer 12 (1996) 56-68.
[78] L. Clarke, I. Glendinning, R. Hempel. The MPI message passing interface standard. Programming environments for massively parallel distributed systems. Springer, 1994. pp. 213-218.
[79] A. Geist, W. Gropp, S. Huss-Lederman, A. Lumsdaine, E. Lusk, W. Saphir, T. Skjellum, M. Snir. MPI-2: Extending the message-passing interface. European Conference on Parallel Processing: Springer, 1996. p.128-135.
[80] W.G. Feather, H. Lim, M. Knezevic. A numerical study into element type and mesh resolution for crystal plasticity finite element modeling of explicit grain structures, Comput. Mech. (2020).
[81] H. Jin, D. Jespersen, P. Mehrotra, R. Biswas, L. Huang, B. Chapman. High performance computing using MPI and OpenMP on multi-core parallel systems, Parallel Computing 37 (2011) 562-575.
[82] G. Krawezik. Performance comparison of MPI and three OpenMP programming styles on shared memory multiprocessors. Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures, 2003. p.118-127.
[83] M. Frigo, S.G. Johnson. The design and implementation of FFTW3, Proceedings of the IEEE 93 (2005) 216-231.
[84] M. Zecevic, R.A. Lebensohn, M. Rogers, J. Moore, V. Chiravalle, E. Lieberman, D. Dunning, G. Shipman, M. Knezevic, N. Morgan. Viscoplastic self-consistent formulation as generalized material model for solid mechanics applications, Applications in Engineering Science 6 (2021) 100040.
[85] C.T. NVIDIA. CUDA CUFFT Library. version PG-05327-032_V02, 2010.
[86] S. Wienke, P. Springer, C. Terboven, D. an Mey. OpenACC—first experiences with real-world applications. European Conference on Parallel Processing: Springer, 2012. p.859-870.
[87] M.D. Hill, M.R. Marty. Amdahl's law in the multicore era, Computer 41 (2008) 33-38.

[88] A. Eghtesad, K. Germaschewski, R.A. Lebensohn, M. Knezevic. A multi-GPU implementation of a full-field crystal plasticity solver for efficient modeling of high-resolution microstructures, Computer Physics Communications 254 (2020) 107231.
[89] J.W. Hutchinson. Bounds and self-consistent estimates for creep of polycrystalline materials, Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences 348 (1976) 101-126.
[90] R.J. Asaro, A. Needleman. Texture development and strain hardening in rate dependent polycrystals, Acta Metall. Mater. 33 (1985) 923-953.
[91] D.J. Savage, I.J. Beyerlein, M. Knezevic. Coupled texture and non-Schmid effects on yield surfaces of body-centered cubic polycrystals predicted by a crystal plasticity finite element approach, International Journal of Solids and Structures 109 (2017) 22-32.
[92] M. Knezevic, I.J. Beyerlein, M.L. Lovato, C.N. Tomé, A.W. Richards, R.J. McCabe. A strain-rate and temperature dependent constitutive model for BCC metals incorporating non-Schmid effects: Application to tantalum–tungsten alloys, Int. J. Plast. 62 (2014) 93-104.
[93] S. Ghorbanpour, M. Zecevic, A. Kumar, M. Jahedi, J. Bicknell, L. Jorgensen, I.J. Beyerlein, M. Knezevic. A crystal plasticity model incorporating the effects of precipitates in superalloys: Application to tensile, compressive, and cyclic deformation of Inconel 718, Int. J. Plast. 99 (2017) 162-185.
[94] S. Ghorbanpour, M.E. Alam, N.C. Ferreri, A. Kumar, B.A. McWilliams, S.C. Vogel, J. Bicknell, I.J. Beyerlein, M. Knezevic. Experimental characterization and crystal plasticity modeling of anisotropy, tension-compression asymmetry, and texture evolution of additively manufactured Inconel 718 at room and elevated temperatures, Int. J. Plast. 125 (2020) 63-79.
[95] R. Bellman, G. Adomian. Green's Functions for Partial Differential Equations. Partial Differential Equations. Springer, 1985. pp. 243-247.
[96] A.I. Zayed. A convolution and product theorem for the fractional Fourier transform, IEEE Signal processing letters 5 (1998) 101-103.
[97] J. Michel, H. Moulinec, P. Suquet. A computational scheme for linear and non-linear composites with arbitrary phase contrast, International Journal for Numerical Methods in Engineering 52 (2001) 139-160.
[98] R.A. Lebensohn, A.K. Kanjarla, P. Eisenlohr. An elasto-viscoplastic formulation based on fast Fourier transforms for the prediction of micromechanical fields in polycrystalline materials, Int. J. Plast. 32-33 (2012) 59-69.
[99] I.J. Beyerlein, C.N. Tomé. A dislocation-based constitutive law for pure Zr including temperature effects, Int. J. Plast. 24 (2008) 867-895.
[100] F.F. Lavrentev. The type of dislocation interaction as the factor determining work hardening, Materials Science and Engineering 46 (1980) 191-208.
[101] H. Mecking, U.F. Kocks. Kinetics of flow and strain-hardening., Acta Metall. Mater. 29 (1981) 1865-1875.
[102] M. Knezevic, L. Capolungo, C.N. Tomé, R.A. Lebensohn, D.J. Alexander, B. Mihaila, R.J. McCabe. Anisotropic stress-strain response and microstructure evolution of textured α-uranium, Acta. Mater. 60 (2012) 702-715.
[103] M. Zecevic, Y.P. Korkolis, T. Kuwabara, M. Knezevic. Dual-phase steel sheets under cyclic tension–compression to large strains: experiments and crystal plasticity modeling, J. Mech. Phys. Solids 96 (2016) 65-87.

[104] K. Kitayama, C. Tomé, E. Rauch, J. Gracio, F. Barlat. A crystallographic dislocation model for describing hardening of polycrystals during strain path changes. Application to low carbon steels, Int. J. Plast. 46 (2013) 54-69.
[105] W. Wen, M. Borodachenkova, C. Tomé, G. Vincze, E. Rauch, F. Barlat, J.J.I.J.o.P. Grácio. Mechanical behavior of Mg subjected to strain path changes: experiments and modeling, 73 (2015) 171-183.
[106] R. Madec, B. Devincre, L. Kubin, T. Hoc, D.J.S. Rodney. The role of collinear interaction in dislocation-induced hardening, 301 (2003) 1879-1882.
[107] T.J. Barrett, R.J. McCabe, D.W. Brown, B. Clausen, S.C. Vogel, M. Knezevic. Predicting deformation behavior of α-uranium during tension, compression, load reversal, rolling, and sheet forming using elasto-plastic, multi-level crystal plasticity coupled with finite elements, J. Mech. Phys. Solids 138 (2020) 103924.
[108] R.A. Lebensohn, A. Needleman. Numerical implementation of non-local polycrystal plasticity using fast Fourier transforms, J. Mech. Phys. Solids 97 (2016) 333-351.
[109] C.J. Bayley, W.A.M. Brekelmans, M.G.D. Geers. A comparison of dislocation induced back stress formulations in strain gradient crystal plasticity, International Journal of Solids and Structures 43 (2006) 7268-7286.
[110] L.P. Evers, W.A.M. Brekelmans, M.G.D. Geers. Non-local crystal plasticity model with intrinsic SSD and GND effects, Journal of the Mechanics and Physics of Solids 52 (2004) 2379-2401.
[111] J.H. Kim, D. Kim, F. Barlat, M.-G. Lee. Crystal plasticity approach for predicting the Bauschinger effect in dual-phase steels, Materials Science and Engineering: A 539 (2012) 259-270.
[112] T. Sritharan, R.S. Chandel. Phenomena in interrupted tensile tests of heat treated aluminium alloy 6061, Acta. Mater. 45 (1997) 3155-3161.