npj | computational materials

Article

Published in partnership with the Shanghai Institute of Ceramics of the Chinese Academy of Sciences

https://doi.org/10.1038/s41524-025-01528-2

Efficient GPU-computing simulation platform JAX-CPFEM for differentiable crystal plasticity finite element method

Fanglei Hu[1], Stephen Niezgoda[2], Tianju Xue[1,3] ✉ & Jian Cao[1] ✉

We present the formulation and applications of JAX-CPFEM, an open-source, GPU-accelerated, and differentiable 3-D crystal plasticity finite element method (CPFEM) software package. Leveraging the modern computing architecture JAX, JAX-CPFEM features high performance through array programming and GPU acceleration, achieving a 39× speedup in a polycrystal case with ~52,000 degrees of freedom compared to MOOSE with MPI (8 cores). Furthermore, JAX-CPFEM utilizes the automatic differentiation technique, enabling users to handle complex, non-linear constitutive materials laws without manually deriving the case-specific Jacobian matrix. Beyond solving forward problems, JAX-CPFEM demonstrates its potential in an inverse design pipeline, where initial crystallographic orientations of polycrystal copper are optimized to achieve targeted mechanical properties under deformations. The end-to-end differentiability of JAX-CPFEM allows automatic sensitivity calculations and high-dimensional inverse design using gradient-based optimization. The concept of differentiable JAX-CPFEM provides an affordable, flexible, and multi-purpose tool, advancing efficient and accessible computational tools for inverse design in smart manufacturing.

Introducion

In recent years, smart manufacturing has gained significant attention due to its increasing promise in precise and intelligent control of manufacturing processes, enhancing the possibility of forming various materials with diverse combinations of strength and formability[1–3]. The Integrated Computational Materials Engineering (ICME) approach, which has been pivotal in linking the microstructures of materials and manufacturing processes to the resulting properties, has contributed significantly to the forward design process[4–10]. Given the rich knowledge in linking material microstructure deformation mechanisms to resulting mechanical properties, Crystal Plasticity (CP) has been developed as a key tool incorporated into ICME[11–20], which predicts the mechanical response of polycrystals up to an industrially relevant component scale[21]. CP leverages extensive knowledge of single-crystal deformation and dislocations from experimental and theoretical studies, forming the basis for modeling the co-deformation of multiple constituents in a polycrystalline aggregate[22,23]. Based on Peirce's work in 1982[24], the Finite Element Method (FEM) has been widely used to solve CP problems. FEM provides the flexibility in applying complex boundary conditions to arbitrarily shaped geometries. Beyond capturing the homogenized behavior of the sample, the FEM solver is also useful for working on a regular grid of material points, that is, component-scale simulations. These features have made CPFEM crucial in establishing microstructure-processing-property relationships in various study areas, such as texture evolution[25–28], multiphase mechanics[29–31], crack propagation[32–34], and so on.

For smart manufacturing, inverse approaches to enable the effective and efficient design of materials microstructure and/or processing parameters are more desirable[35–37]. Inverse design uses target properties as input to derive the initial blank geometry, initial microstructure, and subsequent manufacturing process parameters[6]. Currently, optimization strategies for inverse design fall into two categories: gradient-free and gradient-based approaches. Gradient-free methods, like genetic algorithm[38,39] and Bayesian optimization[40–42], are used for rational searches in the design parameter space. In contrast, gradient-based optimization, which uses search directions defined by the gradient of the function at the current point, is often advantageous for high-dimensional design problems. For example, for processing design in additive manufacturing, gradient-based optimization is more promising because it needs to handle a high-dimensional design space, including the overall thermal history and heat treatment time for each material point[43,44]. Despite its potential, the application of gradient-based optimization algorithms based on CPFEM remains largely unexplored for

[1]Department of Mechanical Engineering, Northwestern University, Evanston, IL, USA. [2]Department of Materials Science and Engineering, The Ohio State University, Columbus, OH, USA. [3]Present address: Department of Civil and Environmental Engineering, The Hong Kong University of Science and Technology, Hong Kong, China. ✉e-mail: cetxue@ust.hk; jcao@northwestern.edu

three main reasons. First, integrating case-specific constitutive laws of CPFEM for slip/twinning hardening into the FEM framework can be a lengthy process and has hindered its wider adoption in design and manufacturing. Second, gradient-based algorithms for inverse design require computing the sensitivity, i.e., the gradient of the objective function with respect to target design parameters, which is usually not available by CPFEM codes. Finally, inverse design usually requires solving forward problems in iterations, but CPFEM simulations are computationally extremely expensive. To address these challenges, we have developed `JAX-CPFEM` based on our recent open-source finite element method library, `JAX-FEM`[45]. `JAX-CPFEM` is an efficient, flexible, and collaborative platform for general-purpose differential CPFEM simulation packages. The following sections will review the current CPFEM software and provide the background and motivation for our developed software.

First, traditional CPFEM approaches often employ a class of constitutive laws for deformation mechanisms as user subroutines, such as `UMAT/VUMAT` in `Abaqus`[24], incorporated into FEM codes. Implicit schemes in CP typically use a predictor-corrector method to update stresses and solution-dependent state variables at the end of the increment, forming a clockwise loop of calculations during stress determination[23], as detailed in Section "Methods". Subsequently, stress predictions at each integration point of an element are iteratively updated using a Newton-Raphson scheme until convergence. However, deriving the algorithmic tangent modulus, necessary for calculating the Jacobian matrix in Newton's iterations, is complex and numerically intensive. The inherent non-linearity in the flow rule model that characterizes the crystal deformation system would require considering CP slip/twinning derivatives and Jacobian contributions from each constitutive model[46], leading to two main issues. First, it is often simplified by neglecting the derivative of increment of normal vectors to slip planes and slip directions with respect to the strain increments, whose error is on the order of the elastic strain increments[47]. Second, when incorporating various shear mechanisms and constitutive relationships, deriving case-by-case Jacobian leads to time-consuming analytical considerations[15,21]. `JAX-CPFEM` addresses these challenges by utilizing automatic differentiation (AD) in `JAX-FEM`, which can compute precise derivatives (up to machine precision) of a given function, automatically yielding the Jacobian matrix without manual derivative calculations[48]. Although the use of AD in the forward problem for finite deformation plasticity has been explored in the last few years[49–52], to the authors' knowledge, few researchers applied AD in CPFEM. Again, AD simplifies the use of complex mechanical models by automating tangent matrix computations necessary for nonlinear solutions. Also, with a Python frontend, `JAX-CPFEM` offers a user-friendly experience for users and developers, enabling the efficient handling of large-scale complex problems.

Second, accurate and efficient sensitivity computation (gradient of the objective function to design parameters), essential for gradient-based optimization algorithms, is a critical aspect of solving inverse problems[43,53]. For example, to understand the effect of each material parameter, a quantitative sensitivity analysis is needed, that is, the derivative of the objective function of stress/strain status of materials to target materials parameters[54,55]. However, for CPFEM, involving complex constitutive and internal variable details of the process history and environmental factors into the structure–property relations leads to strong nonlinearity of the system. Because of this complicated nonlinearity, the derivation of the sensitivity would require considerable effort. Although `JAX-FEM` provides automatic sensitivity analysis functions, the current version has incorporated only simple nonlinear models, e.g., hyperelasticity[56]. For CPFEM, the relationship between strain and stress is implicitly related, so customized differentiation rules will be introduced in `JAX-CPFEM`. Additionally, the implementation of AD for `JAX-CPFEM` is superior to non-AD approaches, like the finite-difference-based numerical derivatives, in terms of accuracy while maintaining efficiency especially for high-dimensional material parameter spaces.

Finally, CPFEM simulations are computationally intensive due to the iterative Newton method over residual functions and Jacobian calculations based on complicated constitutive laws. Coupling CPFEM with inverse design for processing/microstructure design requires calling forward problems iteratively, hence, further increasing computational demands[57–59]. Some existing CPFEM software, such as `MOOSE`[60] and `PRISMS-Plasticity`[61], have developed parallel codes that scale well with increasing processing power (CPU cores). However, large-scale numerical calculations are inevitable and still take days (wall time) for practical applications using 32 or 64 processors, while CPU supercomputers with more than 100 processors are not available for most researchers. There is a need for a faster simulation tool to reduce computational time, and the rapid development of Graphical Processing Units (GPUs) offers a potential solution. Specifically, general-purpose GPUs, which specialize in compute-intensive, highly parallel computation, are more capable than CPUs in data processing[62]. `JAX-CPFEM`, leveraging the `XLA` (Accelerated Linear Algebra) backend of `JAX`[48], demonstrates highly competitive performance, especially when GPUs are utilized. Thus, `JAX-CPFEM` could significantly enhance computational efficiency and promote broader applications in various fields.

In summary, built on the solid foundation of conventional CPFEM, the activation of deformation mechanisms in `JAX-CPFEM` is related to the physics of the material behavior through the constitutive law, e.g., empirical visco-plastic models[63], phenomenological modes[64,65], and physics-based models[66]. In this study, all CPFEM models mentioned later focus on phenomenological constitutive equations and consider dislocation slip as the only deformation mechanism[67]. We want to emphasize the following three features that differential `JAX-CPFEM` differ from other CPFEM software:

1. Automatic Constitutive Laws: Free to realize different deformation mechanics represented by constitutive materials laws by evaluating the case-by-case Jacobian matrix using automatic differentiation.
2. Automatic Sensitivity: Differential simulation for sensitivity analysis with respect to design parameters used in single crystal or polycrystal, which can be seamlessly integrated with inverse design.
3. GPU-acceleration: Efficient solution to forward CPFEM (involving complicated nonlinear relations) with GPU acceleration based on array programming style and matrix formulation.

The remainder of the paper is organized as follows. Section "Results" tests three representative forward CPFEM cases, including single-crystal tantalum (body-centered cubic structure), single-crystal copper (face-centered cubic structure), and polycrystal 304 steel (face-centered cubic structure) under various boundary conditions. The computational performance of `JAX-CPFEM` is further compared with that of a popular CPFEM software, the open-source software `MOOSE`[60]. Additionally, this section conducts the sensitivity analysis realized by differential programming and compares the computational cost of the AD technique and finite-difference-based numerical method. Based on the verified sensitivities, we further proposed a pipeline for inverse design and illustrate its power by taking the example of an inverse design of the initial microstructure of a polycrystal metal featuring the targeted mechanical property after applied deformations. Section "Methods" reviews the formulations of CPFEM, including the governing equation, constitutive laws, and numerical implementation, and introduces several key features of `JAX-CPFEM` that are distinguished from the classic implementation of CPFEM, including automatic constitutive law, array programming style, matrix formulation, and automatic sensitivity. The scheme of notation and other technical information is compiled in Supplementary Note 1.

Results

For benchmarking the performance of `JAX-CPFEM`, we first tested three numerical examples using Kalidindi's phenomenological constitutive law[65], which considers various crystal structures and boundary conditions. Each case was solved with various levels of mesh resolution using `JAX-CPFEM` with both CPU-only mode and GPU mode, and `MOOSE` with MPI for parallel programming. To showcase the inherent efficiency of `JAX-CPFEM`, leveraging advanced features of `JAX`, we compare computational

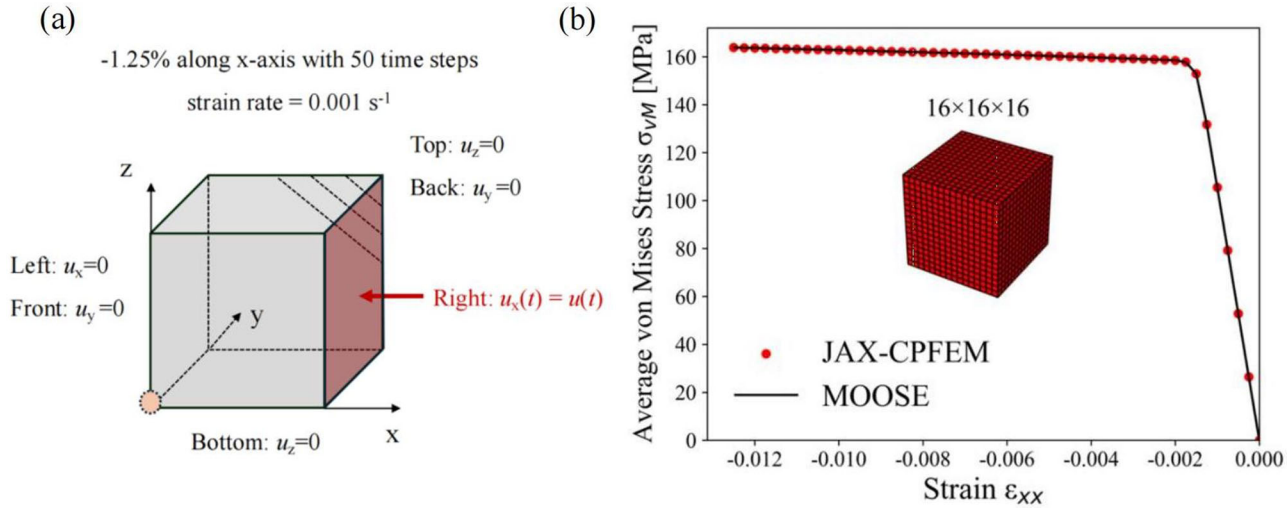

Fig. 1 | Comparison of the CPFEM results between `JAX-CPFEM` and `MOOSE` for Case study 1: single crystal tantalum (BCC). In (**a**), we show the boundary condition: the compressive experiment compresses the right surface with a prescribed displacement condition and applies constraints on the other five faces. **b** A comparison of simulation results of the same problem between `JAX-CPFEM` (points) and `MOOSE` (line).

performance of `JAX-CPFEM` on CPU (1 core) with `MOOSE` using MPI (8 cores). Then in section "AD-based Sensitivity Analysis", to demonstrate the differentiable capabilities of `JAX-CPFEM`, we introduced and verified a sensitivity analysis concerning the grain orientation of each grain in the CPFEM simulation, realized through automatic differentiation (AD). Finally, in section "Inverse Design of Grain Orientations via AD-based Sensitivities", based on the verified AD sensitivities, we conducted the inverse design of grain orientations of each grain in a polycrystal featuring targeted mechanical properties using gradient-based optimization.

Case Study 1: single crystal tantalum (BCC)

We simulated a classic single-crystal tantalum with BCC crystal structure under compressive loading, as shown in Fig. 1a. The domain dimensions are 0.1 mm × 0.1 mm × 0.1 mm, with a strain rate of 0.001 s^{-1} and quasi-static compressive strain from 0 to −1.25% applied in 50 steps along the x-axis on the right surface. All the other five faces are subject to corresponding constraints. The materials parameters used for Kalidindi's self and latent hardening law were sourced from recent literature about calibration[40], including latent hardening coefficient, saturated slip system strength, hardening constants, and so on. These parameters are summarized in Supplementary Note 2. We first mapped the microstructure of Case 1 to the mesh (16^3), as mentioned in Fig. 1b, and computed the volume-averaged von Mises stress versus applied strain from both `JAX-CPFEM` (GPU mode) and `MOOSE` running with eight processes of MPI parallel programming. Both `JAX-CPFEM` and `MOOSE` employed the line search method[68] to determine a suitable step size for Newton iteration, which is used for the CP constitutive relationship. In Fig. 1b, the points and the solid line represent the results obtained from `JAX-CPFEM` and `MOOSE`, respectively. The results matched exactly with each other, validating the accuracy of our software in a single crystal.

For performance benchmarking, this case was solved at various levels of mesh resolution (2^3, 5^3, 10^3, 16^3, 20^3, and 25^3 mesh) using different software. Wall time measurements relative to the number of degrees of freedom (DOF) are shown in Fig. 2. `JAX-CPFEM` running on GPU demonstrated a significant advantage for moderate to larger-size problems. For instance, for a problem with 52,728 DOF (25^3 mesh), `JAX-CPFEM` on CPU and GPU took 1908 s and 629 s, respectively, while `MOOSE` on CPU with MPI (8 cores) took 2812 s. `JAX-CPFEM` achieved 1.5× (CPU mode) and 4.5× (GPU mode) acceleration compared to `MOOSE` with MPI. The computing platforms used for those numerical experiments are summarized in Supplementary Note 3.

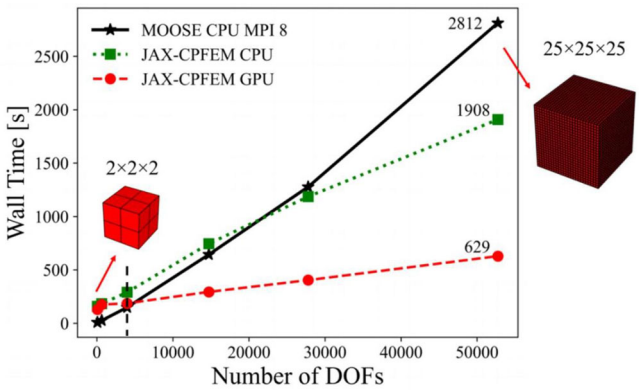

Fig. 2 | Performance test for Case study 1 with different levels of mesh resolution (2^3, 5^3, 10^3, 16^3, 20^3, and 25^3): single crystal tantalum (BCC) under compressive loading boundary condition. Here, "`MOOSE` CPU MPI 8" means `MOOSE` runs with eight processes of MPI parallel programming.

Notably, even without MPI acceleration, `JAX-CPFEM` on CPU (1 core) outperformed `MOOSE` with MPI (8 cores) when the problem scales up to 17,783 DOF (20^3 mesh). The superior performance is due to the `XLA` compiler infrastructure backend of `JAX`, which generates optimized code, and the domain-specific tracing just-in-time (`JIT`) compiler that further allows for high-performance acceleration[69]. Also, `MOOSE`'s MPI acceleration is less effective for larger DOF problems due to message passing delay when transient variables exceed the CPU memory limit. Specifically, transient variables, especially from the global stiffness matrix for solving momentum balance, are stored on local storage, leading to delays. Conversely, `JAX-CPFEM` on GPU exhibits better performance, particularly in more nonlinear cases, as highlighted in the subsequent polycrystal simulation case.

In addition to the computational reasons for the seen speed improvement, another possible reason comes from the numerical aspects. `MOOSE` uses the Jacobian-Free Newton–Krylov (JFNK) method to solve the nonlinear system of equations of FEM, as detailed in ref. 46. JFNK approximates the effect of the Jacobian through finite differences of the residual vector, which avoids the need to compute and store the Jacobian matrix explicitly and, hence, can handle complex nonlinear problems[70]. For the CPFEM case study of tantalum mentioned in ref. 46, it was found that

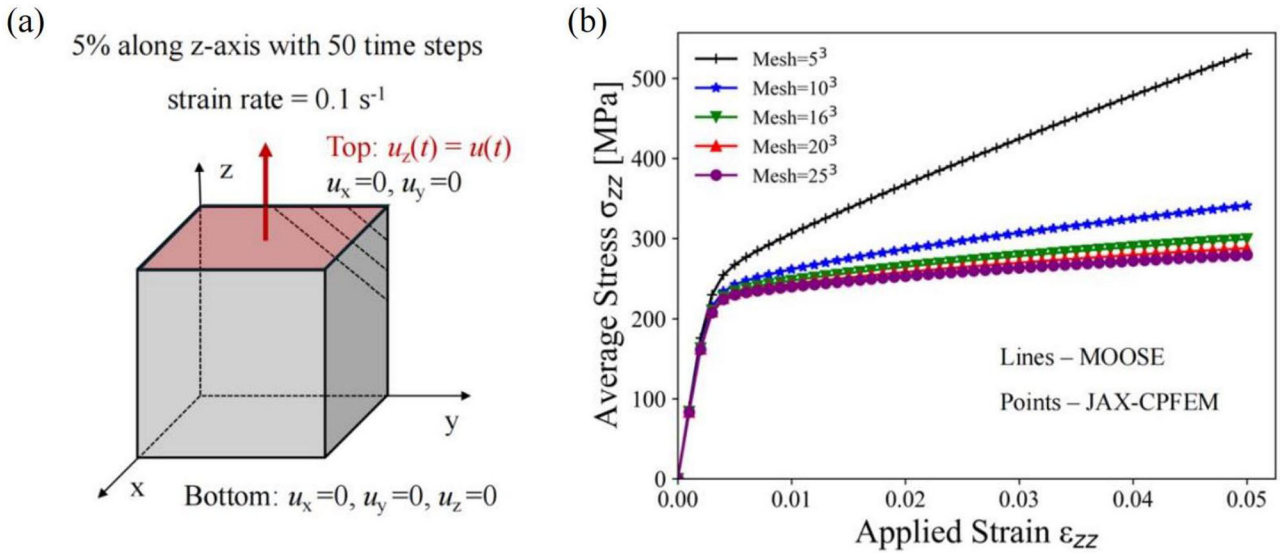

Fig. 3 | Comparison of the CPFEM results between `JAX-CPFEM` and `MOOSE` for Case study 2: single crystal copper (FCC). In (**a**), we show the boundary condition: the tensile experiment fixes the bottom side and pulls the top surface with a prescribed displacement condition. **b** A comparison of simulation results with different levels of mesh resolution (5^3, 10^3, 16^3, 20^3, and 25^3) between `MOOSE` (lines) and `JAX-CPFEM` (points).

JFNK is approximately seven times faster compared to Newton's method with a numerical Jacobian. Despite these advantages of JFNK used in `MOOSE`, the method inherently has truncation errors because the Jacobian-vector product is approximated using a first-order difference, which is directly related to the choice of the perturbation parameter. These truncation errors make `MOOSE` more sensitive to time integration errors, especially at the point of plastic yielding, where the largest error occurs due to the highly nonlinear solution[46]. Thus, beyond the line search algorithm[71] (the same as `JAX-CPFEM`), `MOOSE` also employs a sub-increments algorithm[72], which allows a smaller time step size to be used at the transition region of plastic yielding to reach convergence. This algorithm enhances solver robustness but increases computational time because it needs more time steps to finish simulations. In contrast, `JAX-CPFEM` employs AD to get precise Jacobian matrixes (up to machine precision), and thus, it needs fewer time steps at the transition region and further enhances efficiency.

Case Study 2: single crystal copper (FCC)

To further validate `JAX-CPFEM` for different crystal structures and boundary conditions, we simulated a single-crystal copper with an FCC crystal structure under tensile loading (Fig. 3a). The domain dimensions were 1 mm × 1 mm × 1 mm with a strain rate of 0.1 s^{-1} and quasi-static incremental loadings from 0 to 0.005 mm applied in 50 steps to the top surface along the z-axis. In this scenario, both the top and bottom surfaces were under-constrained, which had more strict applied constraints. Materials parameters were sourced from the `MOOSE` benchmark and literature[65] (see Supplementary Note 2).

The plot of the z-z component of volume-averaged stress versus applied strain is shown in Fig. 3b, comparing results from `JAX-CPFEM` (GPU mode) and `MOOSE` (CPU MPI), solving the Case 2 with different levels of mesh. Similarly, in Fig. 3b, the points and lines represent the results obtained from `JAX-CPFEM` and `MOOSE`, respectively. The mesh-dependent results match perfectly, validating our software's accuracy in simulating a single crystal under more constraints. This case demonstrated mesh sensitivity compared to the previous case. Although quantitative mesh convergence tests are vital for CPFEM scenarios like this one, they are often overlooked. As Lim et al.[73]. noted, mesh convergence in the single crystal is influenced by initial crystal orientation, boundary conditions, intragranular heterogeneity, and the hardening model. However, due to computational cost and the need for explicit discretization of individual grains using many finite elements, mesh convergence studies in CPFEM models are more challenging compared to conventional FEM, which is impossible to obtain reasonable estimates of mesh sensitivity in single crystals. Fortunately, GPU-accelerated `JAX-CPFEM` offers a potential solution for these challenges, since it has the ability to generate enough output results for study, taking a lot of different factors into account while finishing in an acceptable amount of time.

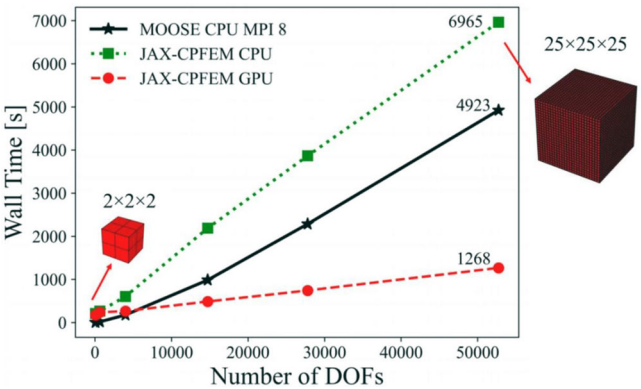

Fig. 4 | Performance test for Case study 2 with different levels of mesh resolution (2^3, 5^3, 10^3, 16^3, 20^3, and 25^3): single crystal copper (FCC) under a tensile loading boundary condition.

Wall time measurements for different mesh resolutions were also conducted using different software. Figure 4 shows the wall time measurements with respect to the number of DOF, of those last five columns of data points correspond to the problems mentioned in Fig. 3b. `JAX-CPFEM` on GPU maintained a significant advantage for larger problem sizes. For a problem with 52,728 DOF (25^3 mesh), it took 6965 s and 1268 s for `JAX-CPFEM` on CPU and GPU, respectively, compared to 4923 s for `MOOSE` on CPU with MPI (8 cores). `JAX-CPFEM` on GPU achieved 1.4× acceleration compared to `MOOSE` with MPI. Although `JAX-CPFEM` on CPU/GPU did not scale as well as in Case 1 (4.5×) due to the increased applied constraints, large CPFEM problems can still be readily solved using `JAX-CPFEM` on GPU. To further demonstrate the computational advantage of `JAX-CPFEM`, we consider a case study of bi-crystal with two different microstructures (BCC + FCC), as detailed in Supplementary Note 6.

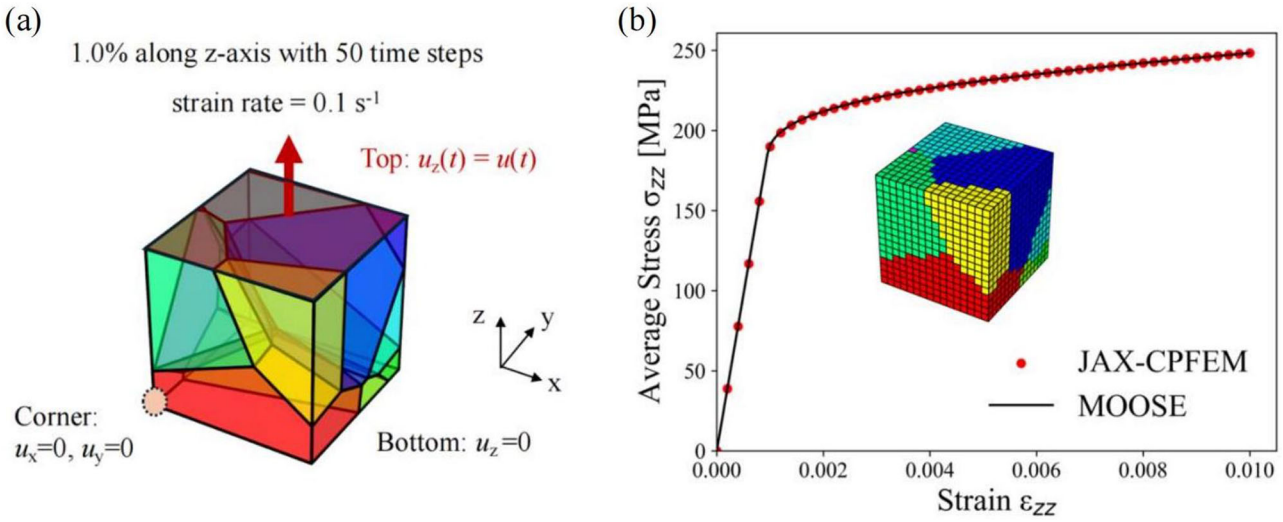

Fig. 5 | Comparison of the CPFEM results between `JAX-CPFEM` and `MOOSE` for Case study 3: polycrystal 304 steel (FCC). In (**a**), we show the boundary condition: the tensile experiment fixes the bottom and a corner and pulls the top surface with a prescribed displacement condition. **b** A comparison of simulation results with the same level of mesh resolution (16^3) between `JAX-CPFEM` (points) and `MOOSE` (line).

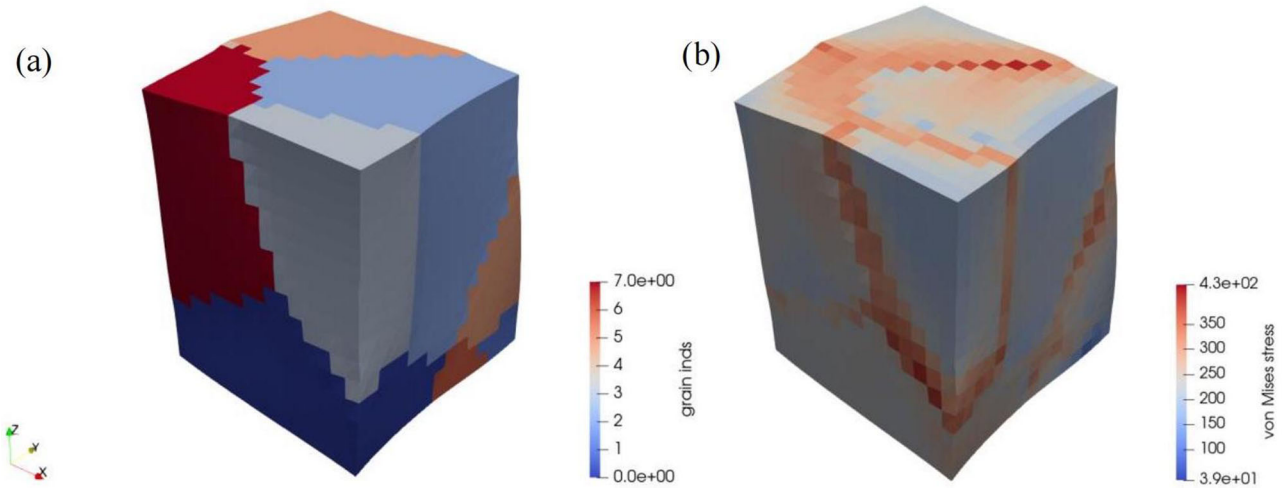

Fig. 6 | Visualization of local microstructure information of polycrystal 304 steel mapped to deformed sample (scale factor: 15) utilizing `ParaView`. a Grain distribution. **b** von Mises equivalent stress. The microstructure of the sample is generated using Neper and simulated using conforming FEM discretization scheme with hexahedral elements.

Case Study 3: polycrystal 304 steel (FCC)

CPFEM is a crucial tool in ICME for investigating the structure-property relationship of polycrystals, bridging microscale microstructure and macroscale performance. Based on recent work on 304 steel (FCC)[40], we validated `JAX-CPFEM` software for a general polycrystal simulation. The average equiaxed grain size was set as 8.0 μm, with randomly generated crystallographic texture and crystal orientation due to the lack of experimental data. Using `Neper`[74], an open-source software package for polycrystal generation and meshing, an RVE of 0.016 mm × 0.016 mm × 0.016 mm was constructed, and crystallographic texture was generated using a seed from a random number generator for computing the initial seed position. Crystal orientations for each grain were represented by quaternion rotation and were generated randomly through `SciPy`[75], an open-source package used for statistics, optimization, and so on. In this section, simulations were conducted on a polycrystal case with ensured crystallographic texture and crystal orientations with various mesh resolutions, as shown in Fig. 5a. A quasi-static tensile strain from 0 to 1% with 50 steps was applied along the z-axis at a strain rate of 0.1 s^{-1}, while the bottom surface and a corner were constrained. The plot of the z-z component of volume-averaged stress versus applied strain is shown in Fig. 5b, comparing results from `JAX-CPFEM` (GPU mode) and `MOOSE` (CPU MPI), based on the same levels of mesh resolution (16^3 mesh). Both showed similar results, validating our software for polycrystal simulations.

Besides homogenized behavior, such as volume-averaged stress, `JAX-CPFEM` can capture detailed local microstructural information. Utilizing `ParaView`[76], an open-source post-processing software, `JAX-CPFEM` can generate various visual outputs. For instance, grain distribution and von Moses equivalent stress field can be mapped onto the deformed sample (Fig. 6). A comparison of visualization of the deformed sample between `JAX-CPFEM` and `MOOSE` is summarized in Supplementary Note 5, which further validates the accuracy of our software. Such detailed visual outputs can also provide the examination of thresholding, clipping, and slicing of microstructural field for analysis. These visualizations provide additional insights into the local fields within the microstructure, offering a deeper understanding of material behavior and enhancing CPFEM studies.

Performance tests with different levels of mesh resolution were conducted, considering 5^3, 10^3, 16^3, 20^3, and 25^3 mesh, with wall time measurement comparisons shown in Fig. 7. Compared to the previous single crystal case study, `JAX-CPFEM` on GPU exhibited a more predominant advantage for polycrystal problems from small to large sizes. For a problem with 52,728 DOF (25^3 mesh), `JAX-CPFEM` on CPU and GPU took 6712 s (~1.9 h) and 2203 s (~0.6 h), respectively, compared to 85,892 s (~1 d) for `MOOSE` on CPU with MPI (8 cores). `JAX-CPFEM` achieved 12.8× (CPU mode) and 39.0× (GPU mode) acceleration compared to `MOOSE` with MPI, drastically reducing computation time from days to less than an hour. Polycrystalline simulations in `MOOSE` with MPI took longer than single-crystal simulations, primarily due to the complexity of polycrystalline materials, including material anisotropy, grain boundary effects, and micromechanical interactions. These factors require more iterative steps for convergence. Each iteration involves solving larger sets of nonlinear equations, increasing computation time. Also, they need to store and manage larger amounts of data related to each grain, such as grain orientations, stress-strain states, etc., increasing memory demands. As mentioned previously, the transient variables exceeding the CPU memory limit are stored in local storage and lead to delays. These factors decrease performance of `MOOSE` MPI acceleration. `JAX-CPFEM` on GPU offers a superior alternative; in summary, this degree of acceleration shows great potential to enhance both forward and inverse design in smart manufacturing.

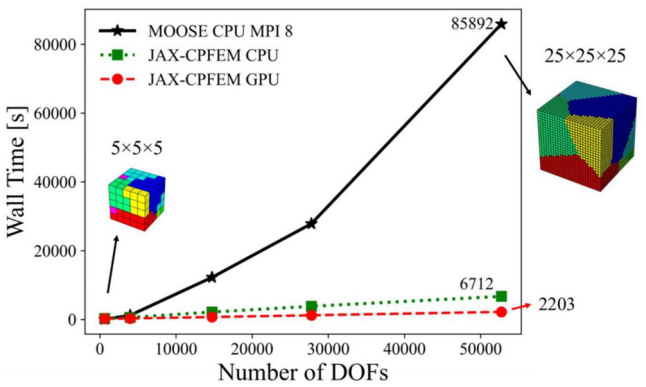

Fig. 7 | Performance test for Case study 3 with different levels of mesh resolution (5^3, 10^3, 16^3, 20^3, and 25^3): polycrystal 304 steel (FCC) under a tensile loading boundary condition.

AD-based sensitivity analysis

Inverse design problems are critical in various engineering applications, such as processing design, microstructure design, and property design. Mathematically, these problems can be formulated as PDE-constrained optimization (PDE-CO) problems[77]. The key to solving these challenges lies in computing the "sensitivity" (gradient of the objective function to design parameters) accurately and efficiently, which is essential for gradient-based optimization algorithms[43]. To demonstrate the differentiable capabilities of `JAX-CPFEM`, we first introduced and verified a sensitivity analysis concerning the grain orientation of each grain in the CPFEM simulation, realized through automatic differentiation (AD). Through these AD sensitivities, we could conduct the inverse design of grain orientations of each grain in a polycrystal featuring targeted mechanical properties using gradient-based optimization, which will be shown in the next section.

In this section, we consider a general copper (FCC) subjected to tensile loadings. The domain dimensions are 0.1 mm × 0.1 mm × 0.1 mm, discretized with 2 × 2 × 2 mesh. Figure 8a shows the boundary conditions and crystallographic geometry. Here, we assume the eight mesh/cells represent different grains of the polycrystal metal, and each with its own crystal orientation represented by three Euler angles (α, β, γ). These angles define a three-dimensional rotation based on sequential rotations around the Z, Y, and X axes relative to its starting position. For the sensitivity analysis of a polycrystal targeting mechanical properties with respect to the grain orientation of each grain, the material response $\hat{O}(\theta)$ is defined as the local mechanical status under deformation. Specifically, the average stress σ_{zz} is extracted from the corner mesh/cell labeled "1", located at the bottom-left corner in the x-y plane of the domain, adjacent to the origin (0, 0, 0), as shown in Fig. 8a. The stress status is computed under a 2% applied deformation. The input parameters are summarized and flatten into a vector

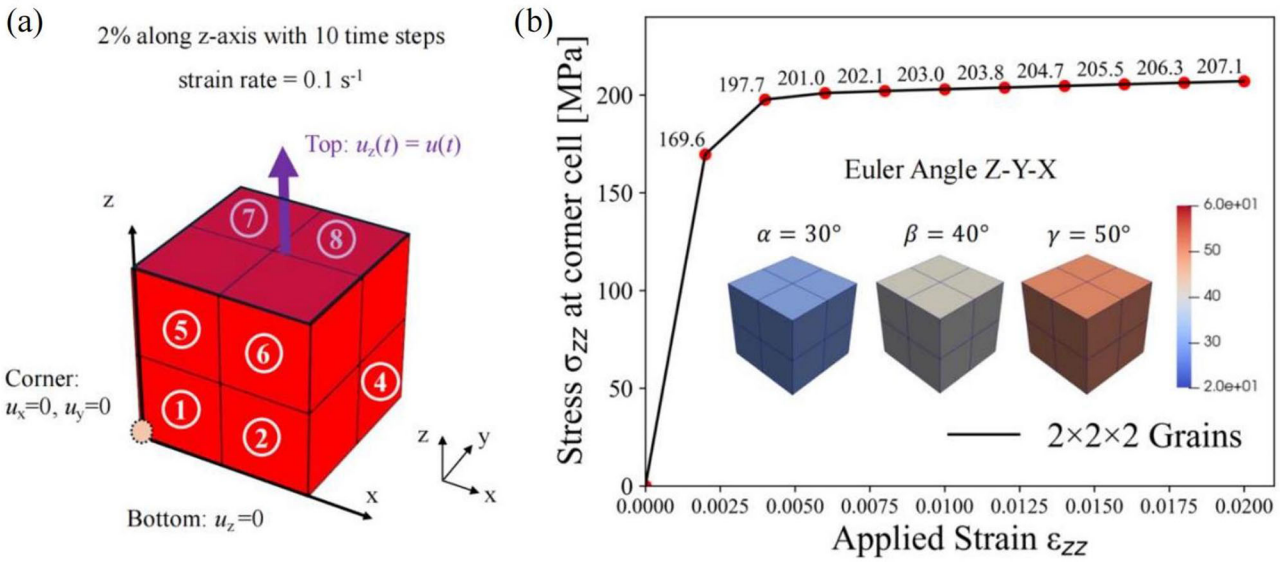

Fig. 8 | Case study for sensitivity analysis with respect to crystal orientation of each cell. In (**a**), we show the boundary condition: a polycrystal copper (FCC) with 2 × 2 × 2 mesh/grains under a tensile loading along the z-axis. The tensile experiment fixes the bottom surface and a corner and pulls the top surface with a prescribed displacement condition. **b** The simulation result of the z-z component of stress σ_{zz} extracted from the corner mesh/cell labeled "1" versus applied strain, obtained from `JAX-CPFEM`. Three Euler angles $[\alpha_i, \beta_i, \gamma_i]$ applied to each mesh/grain are [30°, 40°, 50°]. The color contour bar in the figure represents the value of Euler angle.

$\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_{24}]$, comprising 24 variables, including the three Euler angles $[\alpha_i, \beta_i, \gamma_i]$ applied on all eight mesh/grains. The material response and its corresponding sensitivity are expressed as follows:

$$\hat{O}(\boldsymbol{\theta}) = \sigma_{zz}|_{\epsilon=2\%}(\alpha_1, \beta_1, \gamma_1, \ldots, \alpha_8, \beta_8, \gamma_8) = \sigma_{zz}|_{\epsilon=2\%}(\theta_1, \theta_2, \ldots, \theta_{24}), \quad \text{and } \boldsymbol{J} = \frac{\partial \hat{O}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}. \tag{1}$$

This sensitivity analysis can help us understand the mechanical effect of slight angular deflection of all eight mesh/grains after applying rotation on initial microstructure, additionally, the analysis facilitates automatic inverse design through gradient-based optimization based on experimental mechanical tests. The total derivative of $\hat{O}(\boldsymbol{\theta})$ with respect to parameter variables $\boldsymbol{\theta}$ can be calculated through chain rules, see Supplementary Note 4 for details. However, CPFEM involves complicated nonlinear constitutive relations, and the derivation of the sensitivity by deriving analytical expressions through chain rules is quite non-trivial, tedious and error prone. A possible approach is based on numerical differentiation such as finite-difference-based numerical (FDM) derivatives, which is a kind of numerical method for approximating the value of derivatives[78]. The FDM method is easier to implement. However, it is computationally expensive because it needs to execute the forward CPFEM simulation multiple times, especially when the size of the input parameters is larger than that of the objective function. Furthermore, in situations requiring precise sensitivities, such as in material bifurcation analysis, numerical approximations may introduce significant errors, potentially leading to incorrect conclusions[51]. In contrast, `JAX-CPFEM` computes these derivatives at once in a fully automatic manner, which is not only efficient but also feasible for general users.

For comparison, a case study with three Euler angles $[\alpha_i, \beta_i, \gamma_i] = [30^\circ, 40^\circ, 50^\circ]$ applied to each mesh/grain was used for verification, as shown in Fig. 8b, which also plots the z–z component of average Cauchy stress extracted from the corner cell versus applied strain. The sensitivity calculations for each mesh/cell (No. 1–8) using both AD and FDM-based methods are summarized in Table 1 for sanity check if the derivative is computed correctly. As shown in Table 1, the difference in sensitivity calculations between the two methods is less than 1%, confirming the accuracy of `JAX-CPFEM`'s differentiability and its ability to utilize AD for precise analytic derivatives even in highly non-linear problems.

While the primary benefit of AD in handling complex non-linear models was often highlighted, the computational cost comparison between AD and non-AD approaches is equally important, yet frequently overlooked. In this CPFEM problem, the number of crystal orientations in each cell typically exceeds that of the objective function, necessitating running forward CPFEM models twice as many times as the number of design parameters. For instance, in the case study in Table 1, 48 forward CPFEM runs ($2 \times 8 \times 3$) are required, taking ~4790 s with `JAX-CPFEM` on GPU. In contrast, AD with differentiable `JAX-CPFEM` required only 100 s to obtain the sensitivity vector, achieving a 48× acceleration over FDM even on GPU. This efficiency is promising for grain orientation design in CPFEM using gradient-based optimization, which will be elaborated in the next section.

Table. 1 | Comparison between AD and FDM sensitivity calculations for all cells with rotation $[\alpha_i, \beta_i, \gamma_i] = [30^\circ, 40^\circ, 50^\circ]$

		$\frac{\partial \hat{O}(\theta)}{\partial \alpha_i}$	$\frac{\partial \hat{O}(\theta)}{\partial \beta_i}$	$\frac{\partial \hat{O}(\theta)}{\partial \gamma_i}$
Cell No. 1	AD results	0.027515	−0.082391	−0.074161
	FDM results	0.027499	−0.082383	−0.074126
	Difference	0.58‰	0.10‰	0.47‰
Cell No. 2	AD results	0.027650	−0.079907	−0.070887
	FDM results	0.027635	−0.079898	−0.070850
	Difference	0.54‰	0.11‰	0.52‰
Cell No. 3	AD results	0.028880	−0.082469	−0.072670
	FDM results	0.028862	-0.082461	−0.072646
	Difference	0.62‰	0.10‰	0.33‰
Cell No. 4	AD results	0.025731	−0.081473	−0.074847
	FDM results	0.025690	−0.081466	-0.074823
	Difference	1.59‰	0.09‰	0.32‰
Cell No. 5	AD results	0.025732	−0.081473	−0.074847
	FDM results	0.025691	−0.081466	−0.074822
	Difference	1.59‰	0.09‰	0.33‰
Cell No. 6	AD results	0.028880	−0.082469	−0.072670
	FDM results	0.028863	−0.082462	−0.072646
	Difference	0.59‰	0.08‰	0.33‰
Cell No. 7	AD results	0.027650	−0.079907	−0.070887
	FDM results	0.027635	−0.079898	−0.070851
	Difference	0.54‰	0.11‰	0.51‰
Cell No. 8	AD results	0.027516	−0.082392	−0.074161
	FDM results	0.027498	−0.082383	−0.074126
	Difference	0.65‰	0.11‰	0.47‰

Note: Unit for AD/FDM results: MPa/degree; Difference $= \frac{|\text{AD results} - \text{FDM results}|}{\text{AD results}}$.

Inverse design of grain orientations via AD-based sensitivities

Microstructure optimization, such as considering the effect of grain size summarized by the traditional Hall-Petch relationship[79–81], is crucial for overcoming the well-known strength-ductility tradeoff. However, although simulations excel at mapping an input material microstructure to its resulting property, their direct application to inverse design has traditionally been limited by their high computing cost and lack of differentiability[35]. This section presents a pipeline for inverse design, combining our end-to-end differentiable `JAX-CPFEM` with gradient-based optimization. We illustrate the power of this approach with an example of the inverse design of initial grain orientations in a polycrystalline metal, aiming to achieve targeted mechanical properties after a specific manufacturing process.

The problem domain is a $0.1\,\text{mm} \times 0.1\,\text{mm} \times 0.1\,\text{mm}$ polycrystal copper, discreated with $8 \times 8 \times 8$ hexahedral mesh under the same boundary conditions as shown in Fig. 8a. The objective is to perform an inverse design of the initial crystal orientations of this 512 mesh, each representing different grains. The desired mechanical property is defined by a set of n points representing the local mechanical status (stress-strain curve), extracted from the corner cell adjacent to the origin (0, 0, 0). Let x_i and y_i represent the n points at different applied strain stages and the stress σ_{zz} extracted from the corner cell. The input parameters are summarized and flatten into a vector $\theta = [\theta_1, \theta_2, \ldots, \theta_{1536}]$, including three sequential rotations of Euler angles (α, β, γ) applied on all 512 mesh/grains around the Z, Y, and X axes from the starting position, representing the initial crystal orientation of each grain before deformation. The targeted mechanical properties y_i are the average stresses σ_{zz} of the corner cell at different applied displacement stages, as shown by the black points in Fig. 9a. In summary, the discretized PDE-CO problem is formulated as

$$\begin{aligned} &\min_{\theta \in \mathbb{R}^M} \hat{O}(\boldsymbol{\theta}) \\ &\text{s.t. } \boldsymbol{R}(\boldsymbol{U}, \boldsymbol{\theta}) = 0, \end{aligned} \tag{2}$$

where $\boldsymbol{R}(\boldsymbol{U}, \boldsymbol{\theta})$: $\mathbb{R}^N \times \mathbb{R}^M \to \mathbb{R}^N$ represents the direct consequence of discretizing the weak form and imposing Dirichlet boundary conditions, and $\boldsymbol{U} \in \mathbb{R}^N$ is the FEM solution vector of DOF. The objective function $\hat{O}(\boldsymbol{\theta})$ is the implicit function that arises from solving the PDE, representing the difference between the targeted and simulated mechanical property:

$$\hat{O}(\boldsymbol{\theta}) = w \times \sum_{i=1}^{n} (y_i - f_{sim,i}(\boldsymbol{\theta}))^2, \tag{3}$$

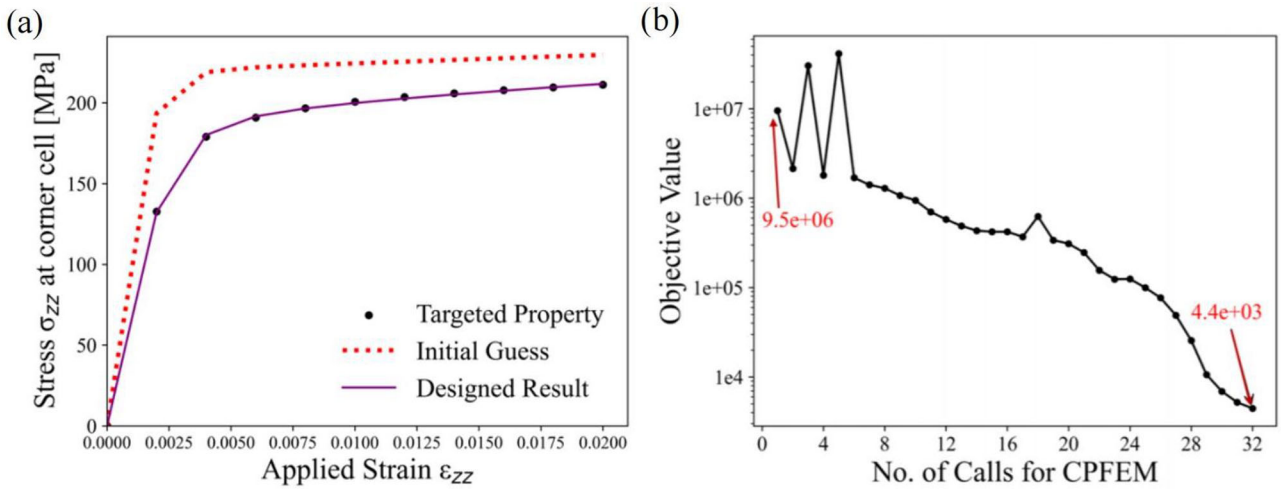

Fig. 9 | Inverse design of the crystal orientation in polycrystalline copper. Subfigure (**a**) shows the targeted local mechanical properties (ground truth) of σ_{zz} extract from the corner cell under different deformation stage x_i, represented by black dots. The red dashed line indicates the JAX-CPFEM simulation outputs based on the initial guess for optimization, which significantly deviates from the targeted properties. The purple line represents JAX-CPFEM simulation results based on the crystal orientations designed by gradient-based optimization. The purple line closely aligns with the targeted properties, demonstrating the robustness of our pipeline. Subfigure (**b**) illustrates the percentage reduction in the objective function value falls within 0.4% with 32 steps.

where $f_{sim,i}(\theta)$ is the optimized simulation (sim) results at different applied displacement stages obtained from JAX-CPFEM, and w is the weight value. It is important to clarify that the sensitivity analysis and subsequent gradient-based optimization are not dependent on the specific choice of the cell as long as the location is consistent in the inverse design pipeline regardless of whether the cell is at the corner or the center cell.

The overall workflow for solving inverse design problems uses gradient-based optimization, utilizing search directions defined by the gradient of the function at the current point. The "optimizer" in the algorithm can use any off-the-shelf gradient-based optimization algorithms; for this problem, we used the limited-memory BFGS algorithm[82] provided by the SciPy package[75] as the optimizer, which is designed for efficiently handling large-scale optimization problems. Based on the chain rule, we can efficiently and automatically compute the tedious and error-prone derivatives $\frac{\partial \hat{O}(\theta)}{\partial \theta}$ using the AD function in JAX called "jax.vjp", which stands for vector-Jacobian product, which has been verified in the above section.

The gradient-based optimization begins with an initial guess with a same rotation for each mesh/grain, meaning the same Euler angles ($\alpha = 30^\circ, \beta = 30^\circ, \gamma = 30^\circ$) for each mesh, as shown in the first row of Fig. 10. The corresponding stress σ_{zz} of the corner cell for this initial guess is shown as the red dashed line in Fig. 9a, which is far from the targeted mechanical properties represented by black dots. Figure 9b shows the optimization iterations, where the objective value is defined in Eq. (3), and each optimization step corresponds to each time the gradient information is queried by the optimizer. Based on the AD-based derivatives calculated at each optimization step, the percentage reduction in the objective function value falls below 1% with only 26 steps, while is about 0.4% when the optimization is completed after 32 steps. Using the designed parameters obtained after 32 gradient-based iterations, we ran JAX-CPFEM to perform a sanity test, ensuring the inverse workflow was processed correctly. The stress-strain curve obtained based on designed parameters, represented by the purple line in Fig. 9a, closely matches the targeted mechanical properties, verifying our inverse design pipeline based on our differentiable JAX-CPFEM. The designed parameters, Euler angles $[\alpha_i, \beta_i, \gamma_i]$ for the 512 mesh/grains, are shown in the second row in Fig. 10. The corresponding inverse pole figures depicting the designed crystallographic orientation of grains are shown in the third row, while the initial guess of crystallographic orientation of grains is labeled by black boxes. This information can provide insights for future microstructure design via smart manufacturing.

In summary, this section proposed an inverse design of crystal orientations in polycrystalline materials as an example, demonstrating the robustness and efficiency of our pipeline, which integrates differentiable JAX-CPFEM with a general optimizer. We highlight three key contributions of our pipeline that distinguish it from others:

1. Scalability to high-dimensional inverse problems: Leveraging gradient-based optimization supported by the differentiable JAX-CPFEM, our pipeline effectively addresses high-dimensional inverse problems. For instance, in this section, the input variables' dimension is 1536, whose complexity is challenging for other gradient-free optimization methods like Bayesian approaches.
2. Handling ill-posed problems: While the example presented in this section is an ill-posed problem with non-unique and multiple solutions, our pipeline has the potential to incorporate various regularization techniques to stabilize and make the problem well-posed. These techniques include Tikhonov regularization[83], adding constraints, and smoothing[84], which are areas we intend to explore further in future research.
3. Versatility beyond crystal orientation design: Beyond crystal orientation designs, our gradient-based inverse design pipeline is adaptable to various manufacturing applications, including the design of initial material microstructures and the subsequent manufacturing process parameters. Also, efficient solution to forward CPFEM (involving complicated nonlinear relations) with GPU acceleration makes a wide range of applications of the inverse design pipeline possible.

Discussions

In the "Results" section, we presented JAX-CPFEM, an open-source, GPU-accelerated, and differentiable 3-D crystal plasticity finite element method (CPFEM) software package. Compared with existing codes of CPFEM, JAX-CPFEM is featured with its affordability, flexibility, and multi-purpose, making it accessible to a wider range of users. Specifically, JAX-CPFEM is fully vectorized and uses automatic differentiation (AD) to computationally evaluate the Jacobian matrix, ensuring constitutive laws and momentum balance can be solved within the Newton scheme. This flexibility allows general users to handle complex, non-linear constitutive materials laws without manually deriving the Jacobian matrix, for example, coupling different deformation mechanics and internal variables. We validated JAX-CPFEM across three CPFEM problems with different crystal structures and boundary conditions, by comparing it with the existing software MOOSE. Currently, JAX-CPFEM supports phenomenological CPFEM models, including Kalidindi's and Peirce's hardening laws. Work on integrating physics-based constitutive models is ongoing.

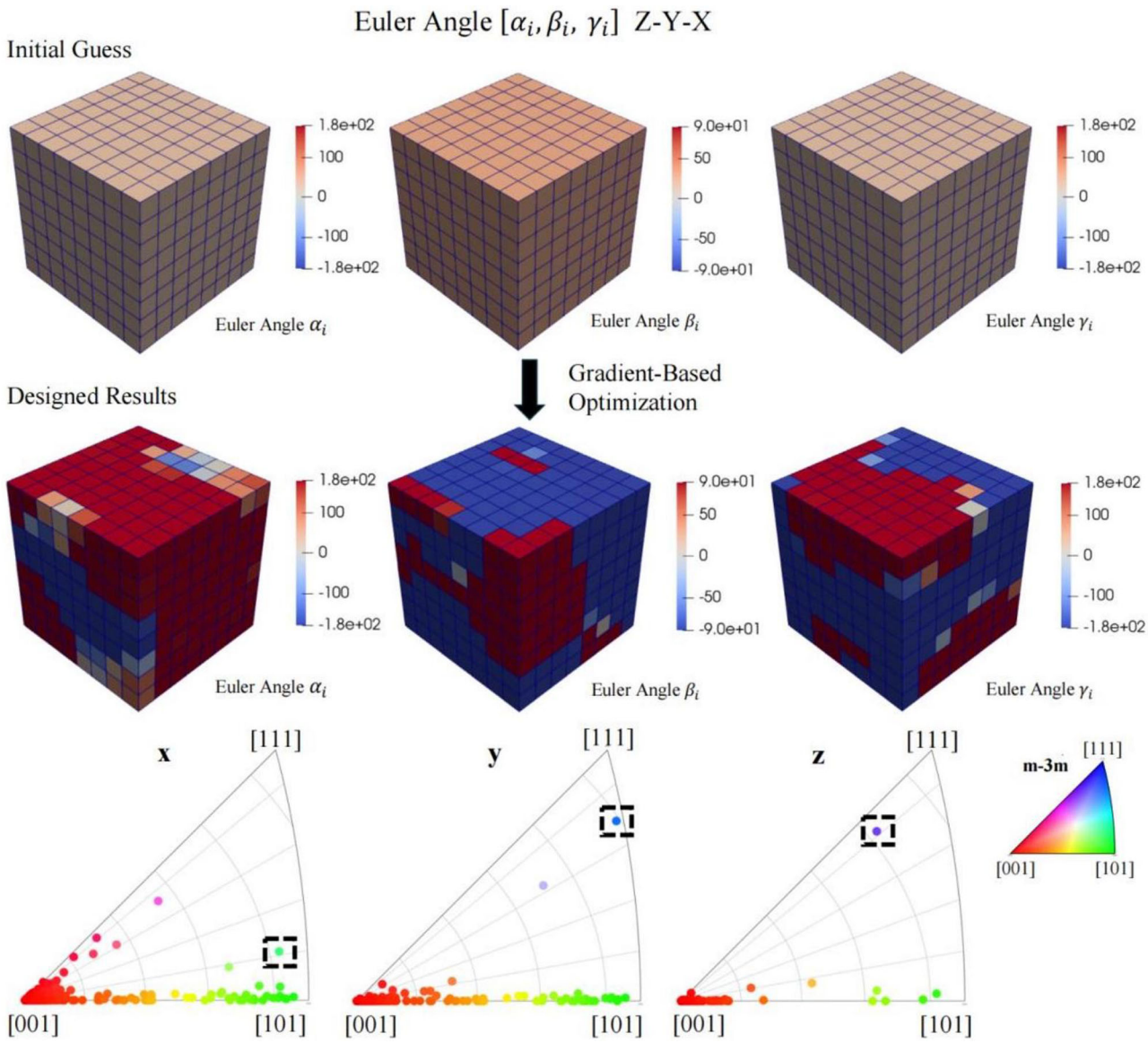

Fig. 10 | Inverse design of the initial crystal orientation of polycrystalline copper under deformations, involving three sequential rotations of Euler angles (α_i, β_i, γ_i) around the Z, Y, and X axes relative to their initial position, applied across all 512 mesh/grains using the differentiable `JAX-CPFEM`**.** The first row shows the initial guess for the gradient-based optimization, where each mesh is assigned the same Euler angles ($\alpha = 30^\circ$, $\beta = 30^\circ$, $\gamma = 30^\circ$). The second row displays the Euler angles designed for each cell after 32 iterations of gradient-based optimization. Inverse pole figures depicting the designed crystallographic orientation of grains for targeted properties, corresponding to a specific direction in the reference frame of crystal, are shown in the third row. The crystallographic orientations of initial guess are also identified on inverse pole figures by black dashed boxes.

Furthermore, we compared the performance of `JAX-CPFEM` (in both CPU-only and GPU modes) with `MOOSE` using MPI across various scenarios. Leveraging AD, array programming, and matrix formulation, `JAX-CPFEM` demonstrated significant computational efficiency, particularly with GPU acceleration. For larger-size single crystal problems, `JAX-CPFEM` (GPU mode) achieved about 1.4–4.5× speedup compared to `MOOSE` with MPI (8 cores). For larger-size polycrystal problems, `JAX-CPFEM` (GPU mode) achieved about 39× acceleration compared to `MOOSE` with MPI (8 cores), reducing computation time from days to under an hour. This greater acceleration in polycrystal simulations is due to the added complexity of polycrystalline materials, including anisotropy, grain boundary effects, and micromechanical interactions, which require more iterative steps and larger data management. For `MOOSE`, MPI's need for frequent synchronization and communication between cores slows down its performance, especially when managing large-scale data[85]. In contrast, GPUs efficiently handle these challenges with their parallel processing power, higher memory bandwidth, and optimization for matrix operations[86], making `JAX-CPFEM` on GPU has a clear speedup in solving polycrystalline problems. Currently, the largest problem solvable with a single 48 GB memory NVIDIA GPU is around 5 million DOF, we plan to leverage the capabilities of `JAX` for distributed computing to extend support for multi-GPU setups. These enhancements are part of our ongoing effort to make `JAX-CPFEM` more scalable for larger-scale problems.

Additionally, sensitivity analysis of a polycrystal copper targeting mechanical properties with respect to the initial crystal orientations of each grain used in CPFEM simulations demonstrated the differentiable capability of `JAX-CPFEM` for end-to-end sensitivity calculation. In the case of considering 8 mesh/cells, AD achieved a 48× acceleration over the FDM-based numerical method even on GPU, highlighting significant computational efficiencies compared to non-AD approaches. Through the AD-based sensitivity, `JAX-CPFEM` performed an inverse

design of initial grain orientations in a polycrystalline metal featuring targeted mechanical properties, based on a pipeline combining `JAX-CPFEM` with gradient-based optimization. This inverse case study illustrates the capability of differentiable `JAX-CPFEM` for seamlessly integrating with inverse design, which could facilitate research in the co-design of material types, initial microstructure and processing parameters. For instance, in directed energy deposition (DED) additive manufacturing (AM), pores are common microstructural features that influence deformation, damage, and failure[87,88]. What specific mechanical deformation processes after the AM process lead to certain regions having particular pore shapes, sizes, and distributions? Addressing this question can directly enhance the mechanical properties of AM products. To find a solution, by treating the process parameters as the design space, `JAX-CPFEM` could be used in combination with a gradient-based optimization algorithm to iteratively minimize the difference between predicted and target pore shapes. Additionally, the co-design of material types and initial microstructure through inverse design using `JAX-CPFEM` may face manufacturing challenges if, for example, the differences in rotation angles between adjacent parts are too large. However, we can further incorporate constraints into this pipeline to make the inverse-designed products more feasible for integration with smart manufacturing.

Methods

This section reviews the formulations of CPFEM, including the governing equation, constitutive laws, and numerical implementation. Then, we introduce several key features of `JAX-CPFEM` that are distinguished from the classic implementation of CPFEM, including automatic constitutive law, array programming style, matrix formulation, and automatic sensitivity.

Governing equation

The kinematics of isothermal finite deformation describes the process where a body originally in a reference configuration, $B \subset R^3$, is deformed to the current configuration, $S \subset R^3$, by a combination of externally applied forces and displacements over a period of time[22]. In this treatment, we choose the perfect single crystal as the reference state, which has the advantage of a constantly unchanged reference state. The balance of momentum in reference configuration (ignoring inertial term and body force) is expressed as follows:

$$\begin{aligned} \mathrm{Div}\, \boldsymbol{P} &= 0 \quad \text{in } \Omega, \\ \boldsymbol{u} &= \boldsymbol{u}_D \quad \text{on } \Gamma_D, \\ \boldsymbol{P} \cdot \boldsymbol{n} &= \boldsymbol{t} \quad \text{on } \Gamma_N, \end{aligned} \tag{4}$$

where $\boldsymbol{P}$ is the first Piola-Kirchhoff stress tensor, $\boldsymbol{u}$ is the displacement field to be solved, $\boldsymbol{u}_D$ is the boundary displacement, $\boldsymbol{t}$ is the traction and $\boldsymbol{n}$ is the outward normal vector.

Crystal plasticity constitutive model

The deformation gradient $\boldsymbol{F}$ is assumed to be multiplicatively decomposed in its elastic and plastic parts[89]:

$$\boldsymbol{F} = \boldsymbol{F}^{\mathrm{e}} \boldsymbol{F}^{\mathrm{p}}, \tag{5}$$

where $\boldsymbol{F}^{\mathrm{e}}$ is the elastic deformation gradient induced the reversible response of the lattice to external loads and displacements, and $\boldsymbol{F}^{\mathrm{p}}$ is the plastic deformation gradient, an irreversible permanent deformation that persists when all external forces and displacements that produce the deformation are removed.

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

$$\boldsymbol{L}^{\mathrm{p}} = \dot{\boldsymbol{F}}^{\mathrm{p}} (\boldsymbol{F}^{\mathrm{p}})^{-1}. \tag{6}$$

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

$$\boldsymbol{E}^{\mathrm{e}} = \frac{1}{2}(\boldsymbol{F}^{\mathrm{e}\mathrm{T}} \boldsymbol{F}^{\mathrm{e}} - \boldsymbol{I}). \tag{7}$$

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

$$\boldsymbol{S} = \mathbb{C} : \boldsymbol{E}^{\mathrm{e}}, \tag{8}$$

where $\mathbb{C}$ is the elastic modulus. The Cauchy stress $\boldsymbol{\sigma}$ is given by

$$\boldsymbol{\sigma} = \frac{1}{\det(\boldsymbol{F}^{\mathrm{e}})} \boldsymbol{F}^{\mathrm{e}} \boldsymbol{S} \boldsymbol{F}^{\mathrm{e}\mathrm{T}}. \tag{9}$$

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

$$\boldsymbol{P} = \det(\boldsymbol{F}) \boldsymbol{\sigma} \boldsymbol{F}^{-\mathrm{T}}. \tag{10}$$

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

$$\boldsymbol{L}^{\mathrm{p}} = \sum_{\alpha} \dot{\gamma}^{\alpha} \boldsymbol{s}_0^{\alpha} \otimes \boldsymbol{m}_0^{\alpha}, \tag{11}$$

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

$$\tau^{\alpha} = \boldsymbol{S} : \boldsymbol{s}_0^{\alpha} \otimes \boldsymbol{m}_0^{\alpha}. \tag{12}$$

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

$$\dot{\gamma}^{\alpha} = \dot{\gamma}_0 \left| \frac{\tau^{\alpha}}{g^{\alpha}} \right|^{1/m} \mathrm{sign}(\tau^{\alpha}), \tag{13}$$

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

$$\dot{g}^{\alpha} = \sum_{\beta} h^{\alpha\beta} |\dot{\gamma}^{\beta}|, \tag{14}$$

where $h^{\alpha\beta}$ is the rate of strain hardening on slip system α due to a shearing on the slip system β. Kalidindi et al. [65] self and latent hardening law gives

$$h^{\alpha\beta} = q^{\alpha\beta} h_0 \left| 1 - \frac{g^{\alpha}}{g_{sat}} \right|^{a} \mathrm{sign}\left(1 - \frac{g^{\beta}}{g_{sat}}\right). \tag{15}$$

Respectively, g_{ini} is the initial slip resistance and g_{sat} is the saturation slip resistance. a and h_0 are slip hardening parameters which are taken to be identical for all slip systems. $q^{\alpha\beta}$ is the matrix describing the latent hardening behavior of a crystallite. Here,

$$q^{\alpha\beta} = \begin{cases} 1 & \text{if } \alpha = \beta \\ r & \text{if } \alpha \neq \beta \end{cases}, \tag{16}$$

where r is the ratio of the latent hardening rate to the self-hardening rate.

Numerical aspects of CPFEM

The quasi-static process is described by Eq. (4). Based on array programming and AD, `JAX-FEM` can solve such second-order elliptic partial differential equation by Newton's method (see ref. 45 for details). For this section, we focus on introducing the constitutive model in its continuous

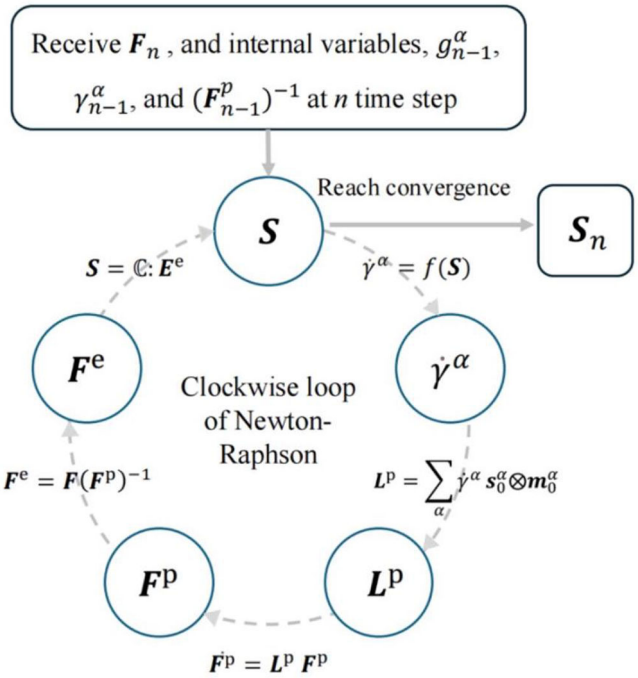

Fig. 11 | Clockwise loop of calculations during stress determination (S second Piola-Kirchhoff stress, $\dot{\gamma}^\alpha$ slip rate, L^p plastic velocity gradient, F^p plastic deformation gradient, F^e elastic deformation gradient).

form and discretizing the constitutive model in time for implementation in the `JAX-FEM` solver.

Since P_n and F_n are implicitly related in CPFEM, current studies often use a predictor-corrector scheme following the clockwise loop of calculations, as shown in Fig. 11. By this way, one could start predicting any of the quantities involved, follow the circle and compare the resulting quantity with the predicted one. Subsequently, the prediction will be updated using Newton's scheme.

Following the idea by Roters et al.[22] and `MOOSE`[60], we stick to the second Piola-Kirchhoff stress S based formulation. Based on S_n, solving P_n is straightforward. At current time step n, we are given the total deformation gradient F_n at any integration point and some internal variables from previous time step $(n-1)$, including g^α_{n-1}, γ^α_{n-1}, and $(F^p_{n-1})^{-1}$, we first solve the following nonlinear equations to get S_n:

$$R_C(F_n, S_n) = 0, \tag{17}$$

where $R_C : \mathbb{R}^{\dim \times \dim} \times \mathbb{R}^{\dim \times \dim} \to \mathbb{R}^{\dim \times \dim}$ is the residual function. The residual function in (17) is explicitly expressed as the following:

$$S_n - \mathbb{C}: \frac{1}{2}\left(F^{e\,\mathrm{T}}_n F^e_n - I\right) = 0, \tag{18}$$

where F^e_n is explicitly computed from S_n through the clockwise loop shown in Fig. 11. In the following description, we discretize the equations in time and show how to map from S_n to F^e_n.

Step 1: from S to $\dot{\gamma}^\alpha$. In this case, we use Kalidindi's hardening law as an example. Based on Eq. (12),

$$\tau^\alpha_n = S_n : s^\alpha_0 \otimes m^\alpha_0. \tag{19}$$

Discretize both Eqs. (13) and (14) in time as

$$\gamma^\alpha_n - \gamma^\alpha_{n-1} = \dot{\gamma}_0 \Delta t \left|\frac{\tau^\alpha_n}{g^\alpha_{n-1}}\right|^{1/m} \mathrm{sign}(\tau^\alpha_n), \tag{20}$$

$$g^\alpha_n - g^\alpha_{n-1} = \sum_\beta q^{\alpha\beta} h_0 \left|1 - \frac{g^\beta_{n-1}}{g_{sat}}\right|^a \mathrm{sign}\left(1 - \frac{g^\beta_{n-1}}{g_{sat}}\right)\left|\gamma^\beta_n - \gamma^\beta_{n-1}\right|. \tag{21}$$

As mentioned before, g^α_{n-1} and γ^α_{n-1} are stored internal variables from previous step.

Step 2: from $\dot{\gamma}^\alpha$ to L^p. Based on previous step's calculation, $\gamma^\alpha_n - \gamma^\alpha_{n-1}$ is available, and we discretize Eq. (11) as

$$L^p_n \Delta t = \sum_\alpha (\gamma^\alpha_n - \gamma^\alpha_{n-1}) s^\alpha_0 \otimes m^\alpha_0. \tag{22}$$

Step 3: from L^p to F^p. We continue to discretize Eq. (6) as

$$(F^p_n)^{-1} = \left(F^p_{n-1}\right)^{-1}(I - L^p_n \Delta t), \tag{23}$$

where I is the second order identity tensor and Δt is the time step size. $\left(F^p_{n-1}\right)^{-1}$ is also one of the stored internal variables from the previous step.

Step 4: from F^p to F^e. Based on Eq. (5), it follows

$$F^e_n = F_n (F^p_n)^{-1}. \tag{24}$$

Up to this point, we are able to evaluate the residual function R_C in Eq. (18) given S_n. Solving the equation requires Newton's method:

$$S^{(k+1)}_n = S^{(k)}_n + \theta \Delta S, \quad \left(\frac{\partial R_C}{\partial S^{(k)}_n}\right) \Delta S = -R_C\left(F_n, S^{(k)}_n\right), \tag{25}$$

where the superscript k denotes the iteration step in Newton's method. $\frac{\partial R_C}{\partial S^{(k)}_n} : \mathbb{R}^{\dim \times \dim \times \dim \times \dim}$ is the Jacobian matrix. Several examples of notations used in the chain rule, including Eq. (25)–(28), are given in Supplementary Note 1. To help the solver to converge, similar to `MOOSE`, we use the line search method[68] to determine a suitable value for the step size θ.

Automatic constitutive laws

As the common practice in nonlinear finite element methods, Newton's method is used to solve the governing equation in Eq. (4), as shown in Algorithm 1. At each integration point, the information of the first Piola-Kirchhoff stress tensor P_n and the tangent tensor $\frac{\partial P_n}{\partial F_n}$ are required. Due to the nonlinear CP constitutive relationship, P_n is obtained by a quadrature-level Newton's method based on the F_n and internal variables, while $\frac{\partial P_n}{\partial F_n}$ is obtained through implicit differentiation.

In practice, S_n is obtained first, based on which P_n is calculated (see Eq. (8)–(10)). Given F_n and the material parameters from the previous time step, the update of S_n during quadrature-level Newton iterations relies on the Jacobian matrix $\frac{\partial R_C}{\partial S_n}$, as shown in Eq. (25). For traditional treatment, e.g., Kalidindi's constitutive law, the CP Jacobian matrix must be derived manually, including the contributions from $\frac{\partial (F^p_n)^{-1}}{\partial (L^p_n)^{-1}}$, $\frac{\partial (L^p_n)^{-1}}{\partial \dot{\gamma}^\alpha_n}$, $\frac{\partial \dot{\gamma}^\alpha_n}{\partial \tau^\alpha_n}$, and $\frac{\partial \tau^\alpha_n}{\partial S_n}$:

$$\frac{\partial (F^p_n)^{-1}}{\partial S_n} = \sum_\alpha \frac{\partial (F^p_n)^{-1}}{\partial (L^p_n)^{-1}} \frac{\partial (L^p_n)^{-1}}{\partial \dot{\gamma}^\alpha_n} \frac{\partial \dot{\gamma}^\alpha_n}{\partial \tau^\alpha_n} \frac{\partial \tau^\alpha_n}{\partial S_n}, \frac{\partial R_C}{\partial S_n} = \mathbb{I} - \left[\mathbb{C}: \frac{\partial E^e_n}{\partial F^e_n} \frac{\partial F^e_n}{\partial (F^p_n)^{-1}} \frac{\partial (F^p_n)^{-1}}{\partial S_n}\right]. \tag{26}$$

Algorithm 1. Newton's Update

for n **in** 1 **to** N_L **do** // Loop over all load steps

  while $\|\boldsymbol{R}(\boldsymbol{u}_n)\| > \varepsilon$ **do** // Newton-Raphson iterations for momentum balance

    Initialize global stiffness matrix $\boldsymbol{K}_M$ and global residual $\boldsymbol{R}_M$ // Sparse matrix $\boldsymbol{K}_M$

    for e in 1 **to** N_e **do** // Do array programming on finite element cells

      Initialize element stiffness matrix $\boldsymbol{K}_e$ and element residual $\boldsymbol{R}_e$ // Dense matrix $\boldsymbol{K}_e$

      for i in 1 **to** N_i **do** // Do array programming on integration points

        Input: $\boldsymbol{F}_n$, g^{α}_{n-1}, γ^{α}_{n-1}, and $(\boldsymbol{F}^{p}_{n-1})^{-1}$

        while $\|\boldsymbol{R}_C(\boldsymbol{F}_n, \boldsymbol{S}_n)\| > \varepsilon_C$ **do** // Newton-Raphson iterations for constitutive laws

          $\boldsymbol{S}_n^{(k+1)} = \boldsymbol{S}_n^{(k)} + \theta\Delta\boldsymbol{S}, \Delta\boldsymbol{S} = -\left(\frac{\partial \boldsymbol{R}_C}{\partial \boldsymbol{S}_n^{(k)}}\right)^{-1} \boldsymbol{R}_C\left(\boldsymbol{F}_n, \boldsymbol{S}_n^{(k)}\right)$

          $k \leftarrow k+1$

        Return $\boldsymbol{S}_n, \frac{\partial \boldsymbol{S}_n}{\partial \boldsymbol{F}_n} = -\left(\frac{\partial \boldsymbol{R}_C}{\partial \boldsymbol{S}_n}\right)^{-1} \frac{\partial \boldsymbol{R}_C}{\partial \boldsymbol{F}_n}$ // Update $\boldsymbol{S}_n$, and get $\frac{\partial \boldsymbol{S}_n}{\partial \boldsymbol{F}_n}$ through implicit differentiation

      $\boldsymbol{P}_n = \det(\boldsymbol{F}_n)\boldsymbol{\sigma}_n \boldsymbol{F}_n^{-\mathrm{T}}, \frac{\partial \boldsymbol{P}_n}{\partial \boldsymbol{F}_n}$ // Get $\boldsymbol{P}_n$ and $\frac{\partial \boldsymbol{P}_n}{\partial \boldsymbol{F}_n}$ at all quad points

      Update $\boldsymbol{K}_e$ and $\boldsymbol{R}_e$ // Update component of $\boldsymbol{R}_e$ based on $\boldsymbol{P}_n$ and update $\boldsymbol{K}_e$ based on $\frac{\partial \boldsymbol{P}_n}{\partial \boldsymbol{F}_n}$

      Add $\boldsymbol{K}_e$ and $\boldsymbol{R}_e$ to $\boldsymbol{K}_M$ and $\boldsymbol{R}_M$ at all quad points. Assemble local components into global $\boldsymbol{R}_M$, $\boldsymbol{K}_M$

    Return $\boldsymbol{K}_M$, $\boldsymbol{R}_M$

    $\boldsymbol{u}_n^{(m+1)} = \boldsymbol{u}_n^{(m)} + \Delta\boldsymbol{u}, \boldsymbol{\Delta u} = -\boldsymbol{K}_M^{-1}\boldsymbol{R}_M$

    $m \leftarrow m+1$

  Return $\boldsymbol{u}_n$ // Finite element solution at load and time step n

Forming the exact analytical derivatives for each component in Eq. (26) is tedious due to the inherent non-linearity in the flow rule model. The exact CP Jacobian can even be intractable, especially when considering (1) multiple deformation mechanisms, e.g., slip, twinning and their interactions; (2) different constitutive relationships, e.g., Peirce's self-hardening law[24] and so on; (3) applied multi-physics conditions, e.g., heating and so on. Consequently, simplifications are often made, neglecting the derivative of the increment of normal vectors to slip planes and slip directions with respect to strain increments. However, this simplified form introduces errors of the same order as the elastic strain increments[47]. In contrast, `JAX-CPFEM` utilizes `jax.jacfwd`, a core function for AD, to automatically evaluate the Jacobian matrix $\frac{\partial \boldsymbol{R}_C}{\partial \boldsymbol{S}_n}$ mentioned in the red solid-line box, eliminating the need for users to manually consider these factors into CPFEM on a case-by-case basis. It should be noted that AD provided by `JAX` has its default rule to calculate derivatives of functions with singular points. The AD derivatives of functions written in `jax.numpy` are the same as those calculated by hand. For Kalidindi's constitutive law applied in our framework, there are two non-smooth equation, including Eq. (13) with absolute value calculation and Eq. (15) with signum functions. For the "signum" function, both left derivative and right derivative are the same, i.e., 0. For the "absolute" function, `JAX` takes the left derivative, which is −1 at the singularity point 0.

Similarly, $\frac{\partial \boldsymbol{P}_n}{\partial \boldsymbol{F}_n}$ will be available once $\frac{\partial \boldsymbol{S}_n}{\partial \boldsymbol{F}_n}$ is obtained. In CPFEM, the relationship between $\boldsymbol{S}_n$ and the strain $\boldsymbol{F}_n$ is implicit, as shown in the residual function mentioned in Eq. (17), linked by the non-linear constitutive relationship of slip hardening laws shown in Fig. 1. The calculation of $\frac{\partial \boldsymbol{S}_n}{\partial \boldsymbol{F}_n}$ hence falls into the framework of "implicit differentiation"[90,91]. Although `JAX-FEM` provides automatic sensitivity analysis functions, it is designed for simpler non-linear models, such as hyperelasticity[56], and cannot be directly applied to compute this derivative in CPFEM. To address this "implicit differentiation" problem of $\frac{\partial \boldsymbol{S}_n}{\partial \boldsymbol{F}_n}$ at each integration point of the mesh, we take the total derivative of Eq. (17) with respect to $\boldsymbol{F}_n$ and obtain

$$\frac{\partial \boldsymbol{R}_C}{\partial \boldsymbol{S}_n}\frac{\partial \boldsymbol{S}_n}{\partial \boldsymbol{F}_n} + \frac{\partial \boldsymbol{R}_C}{\partial \boldsymbol{F}_n} = 0. \tag{27}$$

Therefore,

$$\frac{\partial \boldsymbol{S}_n}{\partial \boldsymbol{F}_n} = -\left(\frac{\partial \boldsymbol{R}_C}{\partial \boldsymbol{S}_n}\right)^{-1} \frac{\partial \boldsymbol{R}_C}{\partial \boldsymbol{F}_n}. \tag{28}$$

In `JAX-CPFEM`, we use a function decorator, `jax.custom_jvp`, to introduce customized differentiation rules. Specifically, we set up a JAX-transformable function to get the value of $\boldsymbol{S}_n$ based on the results of clockwise loop of calculations, which serves as the forward function. Based on this forward function, we further define customized Jacobian-Vector Product (JVP) rules with $\frac{\partial \boldsymbol{R}_C}{\partial \boldsymbol{S}_n}$ and $\frac{\partial \boldsymbol{R}_C}{\partial \boldsymbol{F}_n}$ in Eq. (28) evaluated using normal AD, `jax.jacfwd`. This approach allows us to calculate the value of $\frac{\partial \boldsymbol{S}_n}{\partial \boldsymbol{F}_n}$ at each integration point, as shown in blue dotted-line box in Algorithm 1. A detailed introduction to implicit differentiation and its implementation in `JAX` function can be found in[92].

Array programming

Based on automatic linearization, `JAX-CPFEM` works directly with the weak form and performs the linearization based on AD for each load step. This method enables the assembly of the finite-dimensional linear system discretized by the Galerkin FEM, as shown in Algorithm 1, where N_e is the total number of elements and N_i is the number of integration points associated with each finite element cell. Instead of using for-loops as is common practice in `Fortran` or `C/C++` to go through all elements, `JAX-CPFEM` employs the array programming style (in the same spirit of NumPy[24]) for acceleration. This approach is used in solving Newton's linear problem of momentum balance. Additionally, a key feature in `JAX-CPFEM` is the use of array programming to map the computation at all integration point for all finite element cells at once. By utilizing `jax.vmap`, a core function of `JAX` for vectorized operations, the program can automatically vectorize and efficiently operate over arrays representing batches of inputs, fully utilizing GPU acceleration.

Matrix formulation

For CP, it is essential to update the characteristics of each slip system, including slip resistance, slip rate, and so on, based on the kinetics and interactions of each slip system. Typically, there are more than twelve slip systems[22], and traditional software implementations in `Fortran` or `C/C++` use for-loops to perform these computations. To fully utilize GPU acceleration, we transform constitutive laws from equation formulation to matrix formulation. For example, in Kalidindi's constitutive model, shown in Eq. (21), the update of slip resistance on each slip system is transformed from an equation-based computation to a matrix-based computation:

$$\begin{aligned} g^{\alpha}_{n+1} - g^{\alpha}_{n} = \textstyle\sum_{\beta} q^{\alpha\beta} h_0 \left|1 - \frac{g^{\beta}_{n}}{g_{sat}}\right|^{a} \operatorname{sign}\left(1 - \frac{g^{\beta}_{n}}{g_{sat}}\right) \left|\gamma^{\beta}_{n+1} - \gamma^{\beta}_{n}\right|. \\ \rightarrow \boldsymbol{g} = \boldsymbol{Q} \cdot \boldsymbol{c}, \end{aligned} \tag{29}$$

where $\boldsymbol{g} \in \mathbb{R}^{\alpha \times 1}$ is the vector of increment of slip resistance on each slip system, $\boldsymbol{Q} \in \mathbb{R}^{\alpha \times \alpha}$ is the matrix of coefficient matrix of interactions between each slip system, and $\boldsymbol{c} \in \mathbb{R}^{\alpha \times 1}$ is the remaining part on the right-hand side.

Automatic sensitivity

The successful computation of "sensitivity", i.e., the gradient of the objective function to design parameters, is crucial for understanding the effect of different materials/microstructure/processing parameters on the mechanical properties of either single crystal or polycrystal metals. We formulate the CPFEM forward simulations as follows

$$\boldsymbol{R}(\boldsymbol{U}, \boldsymbol{\theta}) = 0, \tag{30}$$

where $\boldsymbol{U} \in \mathbb{R}^N$ is the FEM solution vector of degrees of freedom (DOF), $\boldsymbol{\theta} \in \mathbb{R}^M$ is the materials/microstructure/processing parameter vector. $\boldsymbol{R}(\cdot, \cdot): \mathbb{R}^N \times \mathbb{R}^M \rightarrow \mathbb{R}^N$ is the constraint function, which represents the discretized governing partial difference equation (PDE). To conduct a sensitivity analysis of the parameter variables, we define the materials response as $O(\boldsymbol{U}, \boldsymbol{\theta}): \mathbb{R}^N \times \mathbb{R}^M \rightarrow \mathbb{R}$, which could be the sum of n points/stages defining simulation results (i.e., strain–stress curve of the polycrystal). A reduced formulation is used to embed the PDE constraint so that the materials response can be re-formulated as

$$\hat{O}(\boldsymbol{\theta}) := O(\boldsymbol{U}(\boldsymbol{\theta}), \boldsymbol{\theta}), \tag{31}$$

where $\boldsymbol{U}(\boldsymbol{\theta})$ is the implicit function that arises from solving the PDE. The total derivative of $\hat{O}(\boldsymbol{\theta})$ with respect to parameter variables $\boldsymbol{\theta}$ can be calculated through chain rules, see Supplementary Note 4 for details. Nevertheless, CPFEM entails complex nonlinear constitutive relations, and obtaining the sensitivity through analytical derivations using the chain rule is highly challenging, labor-intensive, and prone to errors.

A possible approach is based on numerical differentiation such as finite-difference-based numerical derivatives (FDM), which is a kind of numerical method for approximating the value of derivatives. Assuming a small perturbation of the model parameters, the model can be linearized with respect to the perturbation and the new material response as follows

$$\hat{O}(\boldsymbol{\theta} + \boldsymbol{\delta}) \approx \hat{O}(\boldsymbol{\theta}) + \boldsymbol{J}\boldsymbol{\delta}, \tag{32}$$

where

$$\boldsymbol{J} = \frac{\partial \hat{O}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}, \tag{33}$$

$\boldsymbol{J}$ is the gradient, namely, the derivatives of material response with respect to the set of parameter variables. To compute the Jacobian matrix, one of the parameters is perturbed by $\Delta\theta$ as follows

$$\boldsymbol{\theta}^{*j} = \boldsymbol{\theta} + [0, 0, \ldots, \Delta\theta_j, \ldots, 0]^{\mathrm{T}}, j = 1, 2, \ldots M \tag{34}$$

and the response of the perturbed model is determined by the FEM analysis of the RVE. This procedure is repeated for each parameter in the model and resulting result is given by,

$$J_j = \frac{\partial \hat{O}(\theta)}{\partial \theta_j} \approx \frac{\hat{O}(\theta^{*j}) - \hat{O}(\theta)}{\Delta\theta_j}. \tag{35}$$

The FDM is easy to implement. However, it is computationally expensive because it needs to execute the forward CPFEM simulation multiple times, especially when the size of the input parameters is larger than that of the objective function (e.g., $M \gg 1$ in this scenario). In addition, when accurate sensitivities are critical, such as in material bifurcation analysis, numerical approximations can lead to substantial errors, which may result in inaccurate conclusions[51].

`JAX-CPFEM` obtains these derivatives at once in a fully automatic manner, which is not only efficient but also feasible for general users. In addition, mathematically, inverse problems like calibration can be formulated as PDE-constrained optimization problems[77]. The successful computation of "sensitivity" is also the basis for gradient-based optimization algorithms. Namely, `JAX-CPFEM` also paves the way for gradient-based optimization algorithms to be applied to inverse problems.

Data availability

All data generated or analyzed during this study are included in this published article.

Code availability

`JAX-CPFEM` was constructed on top of our recent work `JAX-FEM`. They can be freely downloaded from the following link: https://github.com/SuperkakaSCU/JAX-CPFEM.

Received: 27 September 2024; Accepted: 17 January 2025;
Published online: 22 February 2025

References

1. Bernard, A. et al. Vision on metal additive manufacturing: developments, challenges and future trends. *CIRP J. Manuf. Sci. Technol.* **47**, 18–58 (2023).
2. Meng, X., Huang, Y., Cao, J., Shen, J. & dos Santos, J. F. Recent progress on control strategies for inherent issues in friction stir welding. *Prog. Mater. Sci.* **115**, 100706 (2021).
3. Dimiduk, D. M., Holm, E. A. & Niezgoda, S. R. Perspectives on the impact of machine learning, deep learning, and artificial intelligence on materials, processes, and structures engineering. *Integr. Mater. Manuf. Innov.* **7**, 157–172 (2018).
4. Mianroodi, J. R. et al. Modeling and simulation of microstructure in metallic systems based on multi-physics approaches. *npj Comput. Mater.* **8**, 93 (2022).
5. Allison, J. Integrated computational materials engineering: a perspective on progress and future steps. *J. Min. Metals Mater. Soc.* (TMS) **63** (2011).
6. Cao, J., Bambach, M., Merklein, M., Mozaffar, M., Xue, T. Artificial intelligence in metal forming, CIRP Ann. (2024).
7. Chen, L. Q. & Zhao, Y. H. From classical thermodynamics to phase-field method. *Prog. Mater. Sci.* **124**, 100868 (2022).
8. McNamara, K. et al. Predicting phase transformation kinetics during metal additive manufacturing using non-isothermal Johnson-Mehl-Avrami models: Application to Inconel 718 and Ti-6Al-4V. *Addit. Manuf.* **49**, 102478 (2022).
9. Hu, X. et al. Spectral phase-field model of deformation twinning and plastic deformation. *Int. J. Plast.* **143**, 103019 (2021).
10. Hu, F., Han, G., Fu, B., Shi, T., Huang, X. An elastoplastic phase-field study of the precipitation behaviors of Mg17Al12 phase in Mg-Al-based alloys: Part II. Precipitation under various loadings, J. Alloy. Compd. 171178 (2023).
11. Chakraborty, S. Patil, C. S. Niezgoda, S. R. Understanding the mechanisms behind the development of cube orientation in the non-cube grains during plane strain compression of medium to high stacking-fault energy FCC metals, Acta Mater. 119080 (2023).
12. Kang, J., Liu, X. & Niezgoda, S. R. Crystal plasticity modeling of ultrasonic softening effect considering anisotropy in the softening of slip systems. *Int. J. Plast.* **156**, 103343 (2022).
13. Patra, A. et al. P-cp: Open source dislocation density based crystal plasticity framework for simulating temperature-and strain rate-dependent deformation. *Comput. Mater. Sci.* **224**, 112182 (2023).
14. Pinz, M., Benzing, J., Pilchak, A. & Ghosh, S. A microstructure-based porous crystal plasticity FE model for additively manufactured Ti-6Al-4V alloys. *Int. J. Plast.* **153**, 103254 (2022).
15. Panchal, J. H., Kalidindi, S. R. & McDowell, D. L. Key computational modeling issues in integrated computational materials engineering. *Comput.-Aided Des.* **45**, 4–25 (2013).
16. Staroselsky, A. & Anand, L. A constitutive model for hcp materials deforming by slip and twinning: application to magnesium alloy AZ31B. *Int. J. Plast.* **19**, 1843–1864 (2003).
17. Zhu, K.-Y., Dai, S., Zou, S.-H., Yu, Y.-J., Deng, Z.-C. Experimental study and crystal plasticity modeling of additive manufacturing IN718 superalloy considering negative strain rate sensitivity behavior. Eur. J. Mech. A Solids 105304 (2024).
18. Zhang, W. et al. Quantitatively assessing the contributions of temperature-dependent deformation-induced martensitic transformation to uniform elongation and work hardening of TRIP-assisted duplex stainless steel via crystal plasticity, Mater. Sci. Eng. A 145758 (2023).
19. Agius, D. et al. A crystal plasticity model that accounts for grain size effects and slip system interactions on the deformation of austenitic stainless steels. *Int. J. Plast.* **152**, 103249 (2022).
20. Zhou, Y. et al. Multi-scale crystal plasticity finite element simulations of the microstructural evolution and formation mechanism of adiabatic shear bands in dual-phase Ti20C alloy under complex dynamic loading. *J. Mater. Sci. Technol.* **59**, 138–148 (2020).
21. Roters, F. et al. 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. *Comput. Mater. Sci.* **158**, 420–478 (2019).
22. Roters, F., Eisenlohr, P., Bieler, T., Raabe, D. *Crystal Plasticity Finite Element Methods: in Materials Science and Engineering* (WILEY-VCH Verlag, 2011).
23. Roters, F. et al. Overview of constitutive laws, kinematics, homogenization and multiscale methods in crystal plasticity finite-element modeling: Theory, experiments, applications. *Acta Mater.* **58**, 1152–1211 (2010).
24. Peirce, D., Asaro, R. & Needleman, A. An analysis of nonuniform and localized deformation in ductile single crystals. *Acta Met.* **30**, 1087–1119 (1982).
25. Daroju, S. et al. Experimental characterization and crystal plasticity modeling for predicting load reversals in AA6016-T4 and AA7021-T79. *Int. J. Plast.* **153**, 103292 (2022).
26. Wang, D. et al. Texture evolution and slip mode of a Ti-5.5 Mo-7.2 Al-4.5 Zr-2.6 Sn-2.1 Cr dual-phase alloy during cold rolling based on multiscale crystal plasticity finite element model. *J. Mater. Sci. Technol.* **111**, 76–87 (2022).
27. Zhao, J., Lv, L., Wang, K. & Liu, G. Effects of strain state and slip mode on the texture evolution of a near-α TA15 titanium alloy during hot deformation based on crystal plasticity method. *J. Mater. Sci. Technol.* **38**, 125–134 (2020).
28. Ghorbanpour, S. et al. 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**, 63–79 (2020).
29. Park, T. et al. Crystal plasticity modeling of 3rd generation multi-phase AHSS with martensitic transformation. *Int. J. Plast.* **120**, 1–46 (2019).
30. Xu, B. et al. Effect of pore on the deformation behaviors of NiTi shape memory alloys: a crystal-plasticity-based phase field modeling. *Int. J. Plast.* **175**, 103931 (2024).
31. Zhao, P., Shen, C., Niezgoda, S. R. & Wang, Y. Heterogeneous γ′ microstructures in nickel-base superalloys and their influence on tensile and creep performance. *Int. J. Plast.* **109**, 153–168 (2018).
32. Zhao, Q., Wahab, M. A., Ling, Y. & Liu, Z. Fatigue crack propagation within Al-Cu-Mg single crystals based on crystal plasticity and XFEM combined with cohesive zone model. *Mater. Des.* **210**, 110015 (2021).
33. Chakraborty, S. & Ghosh, S. A concurrent atomistic-crystal plasticity multiscale model for crack propagation in crystalline metallic materials. *Comput. Methods Appl. Mech. Eng.* **379**, 113748 (2021).
34. Mayeur, J., Beyerlein, I., Bronkhorst, C. & Mourad, H. Incorporating interface affected zones into crystal plasticity. *Int. J. Plast.* **65**, 206–225 (2015).
35. Liu, H. et al. End-to-end differentiability and tensor processing unit computing to accelerate materials' inverse design. *npj Comput. Mater.* **9**, 121 (2023).
36. Shiraiwa, T., Briffod, F., Enoki, M. & Yamazaki, K. Inverse analysis of the relationship between three-dimensional microstructures and tensile properties of dual-phase steels. *Mater. Today Commun.* **33**, 104958 (2022).
37. Mozaffar, M. et al. Mechanistic artificial intelligence (mechanistic-AI) for modeling, design, and control of advanced manufacturing processes: Current state and perspectives. *J. Mater. Process. Technol.* **302**, 117485 (2022).
38. Goldberg, D. E. *Genetic Algorithms* (Pearson Education India, 2013).
39. Savage, D. J., Feng, Z. & Knezevic, M. Identification of crystal plasticity model parameters by multi-objective optimization

integrating microstructural evolution and mechanical data. *Comput. Methods Appl. Mech. Eng.* **379**, 113747 (2021).
40. Tran, A., H. Lim, H. An asynchronous parallel high-throughput model calibration framework for crystal plasticity finite element constitutive models, Comput. Mech. 1–14. (2023).
41. Tran, A. et al. aphBO-2GP-3B: a budgeted asynchronous parallel multi-acquisition functions for constrained Bayesian optimization on high-performing computing architecture. *Struct. Multidiscip. Optim.* **65**, 132 (2022).
42. Bernardo, J. M., Smith, A. F. *Bayesian Theory* (John Wiley & Sons, 2009).
43. Mozaffar, M., Liao, S., Jeong, J., Xue, T. & Cao, J. Differentiable simulation for material thermal response design in additive manufacturing processes. *Addit. Manuf.* **61**, 103337 (2023).
44. Mozaffar, M., Cao, J. Additive manufacturing process design with differentiable simulations, arXiv preprint arXiv:2107.10919 (2021).
45. Xue, T. et al. JAX-FEM: a differentiable GPU-accelerated 3D finite element solver for automatic inverse design and mechanistic data science. *Comput. Phys. Commun.* **291**, 108802 (2023).
46. Chockalingam, K. et al. Crystal plasticity with Jacobian-free Newton–Krylov. *Comput. Mech.* **51**, 617–627 (2013).
47. Huang, Y. *A User-Material Subroutine Incroporating Single Crystal Plasticity in the ABAQUS Finite Element Program* (Harvard University, 1991).
48. Bradbury, J. et al, JAX: Composable Transformations of Python+ NumPy programs (2018).
49. Seidl, D. T. & Granzow, B. N. Calibration of elastoplastic constitutive model parameters from full-field data with automatic differentiation-based sensitivities. *Int. J. Numer. Methods Eng.* **123**, 69–100 (2022).
50. Li, Z., Bloomfield, M. O. & Oberai, A. A. Simulation of finite-strain inelastic phenomena governed by creep and plasticity. *Comput. Mech.* **62**, 323–345 (2018).
51. Chen, Q., Ostien, J.T., Hansen, G. Automatic differentiation for numerically exact computation of tangent operators in small-and large-deformation computational inelasticity. In: *Proc. 143rd Annual Meeting & Exhibition: Annual Meeting Supplemental Proceedings*, 289–296 (Springer, 2016).
52. Rothe, S. & Hartmann, S. Automatic differentiation for stress and consistent tangent computation. *Arch. Appl. Mech.* **85**, 1103–1125 (2015).
53. Fullwood, D. T., Niezgoda, S. R., Adams, B. L. & Kalidindi, S. R. Microstructure sensitive design for performance optimization. *Prog. Mater. Sci.* **55**, 477–562 (2010).
54. Kulkarni, S. S., Venkatraman, A., Senor, D. J. & Devanathan, R. A sensitivity analysis of twinning crystal plasticity finite element model using single crystal and poly crystal Zircaloy. *Comput. Mater. Sci.* **230**, 112425 (2023).
55. Kotha, S., Ozturk, D. & Ghosh, S. Parametrically homogenized constitutive models (PHCMs) from micromechanical crystal plasticity FE simulations, part I: Sensitivity analysis and parameter identification for Titanium alloys. *Int. J. Plast.* **120**, 296–319 (2019).
56. Ogden, R. W. *Non-Linear Elastic Deformations* (Courier Corporation, 1997).
57. Kuhn, J., Spitz, J., Sonnweber-Ribic, P., Schneider, M., Böhlke, T. Identifying material parameters in crystal plasticity by Bayesian optimization. *Optim. Eng.* **23**,1489–1523 (2021).
58. Veasna, K., Feng, Z., Zhang, Q. & Knezevic, M. Machine learning-based multi-objective optimization for efficient identification of crystal plasticity model parameters. *Comput. Methods Appl. Mech. Eng.* **403**, 115740 (2023).
59. Sedighiani, K. et al. Determination and analysis of the constitutive parameters of temperature-dependent dislocation-density-based crystal plasticity models. *Mech. Mater.* **164**, 104117 (2022).
60. Permann, C. J. et al. MOOSE: enabling massively parallel multiphysics simulation. *SoftwareX* **11**, 100430 (2020).
61. Yaghoobi, M. et al, PRISMS-plasticity: an open-source crystal plasticity finite element software. Comput. Mater. Sci. **169**, 109708 (2019).
62. Sun, W., Yan, R., Zhang, Y., Dong, H. & Jing, T. GPU-accelerated three-dimensional large-scale simulation of dendrite growth for Ti6Al4V alloy based on multi-component phase-field model. *Comput. Mater. Sci.* **160**, 149–158 (2019).
63. Asaro, R. J. & Rice, J. Strain localization in ductile single crystals. *J. Mech. Phys. Solids* **25**, 309–338 (1977).
64. Kalidindi, S. R. Incorporation of deformation twinning in crystal plasticity models. *J. Mech. Phys. Solids* **46**, 267–290 (1998).
65. Kalidindi, S. R., Bronkhorst, C. A. & Anand, L. Crystallographic texture evolution in bulk deformation processing of FCC metals. *J. Mech. Phys. Solids* **40**, 537–569 (1992).
66. Wong, S. L., Madivala, M., Prahl, U., Roters, F. & Raabe, D. A crystal plasticity model for twinning-and transformation-induced plasticity. *Acta Mater.* **118**, 140–151 (2016).
67. Raabe, D. Roters, F. Barlat, F. Chen, L.-Q. *Continuum Scale Simulation of Engineering Materials: Fundamentals-Microstructures-process Applications* (John Wiley & Sons, 2004).
68. Nocedal, J., Wright, S. J. *Numerical Optimization* (Springer, 1999).
69. Frostig, R. Johnson, M. J. Leary, C. Compiling machine learning programs via high-level tracing. Syst. Mach. Learn. **4** (2018).
70. Knoll, D. A. & Keyes, D. E. Jacobian-free Newton–Krylov methods: a survey of approaches and applications. *J. Comput. Phys.* **193**, 357–397 (2004).
71. Mánik, T., Asadkandi, H. & Holmedal, B. A robust algorithm for rate-independent crystal plasticity. *Comput. Methods Appl. Mech. Eng.* **393**, 114831 (2022).
72. Vieira de Carvalho, M., Cardoso Coelho, R. P. & Pires, F. M. A. On the computational treatment of fully coupled crystal plasticity slip and martensitic transformation constitutive models at finite strains. *Int. J. Numer. Methods Eng.* **123**, 5155–5200 (2022).
73. Lim, H., Battaile, C. C., Bishop, J. E. & Foulk, J. W. III, Investigating mesh sensitivity and polycrystalline RVEs in crystal plasticity finite element simulations. *Int. J. Plast.* **121**, 101–115 (2019).
74. Quey, R., Dawson, P. R. & Barbe, F. Large-scale 3D random polycrystals for the finite element method: generation, meshing and remeshing. *Comput. Methods Appl. Mech. Eng.* **200**, 1729–1745 (2011).
75. Virtanen, P. et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Methods* **17**, 261–272 (2020).
76. Henderson, A., Ahrens, J., Law, C. *The ParaView Guide* (ParaView Developers2004).
77. Rees, T., Dollar, H. S. & Wathen, A. J. Optimal solvers for PDE-constrained optimization. *SIAM J. Sci. Comput.* **32**, 271–298 (2010).
78. Herrera-Solaz, V., LLorca, J., Dogan, E., Karaman, I. & Segurado, J. An inverse optimization strategy to determine single crystal mechanical behavior from polycrystal tests: application to AZ31 Mg alloy. *Int. J. Plast.* **57**, 1–15 (2014).
79. Petch, N. J. The cleavage strength of polycrystals. *J. Iron Steel Inst.* **174**, 25–28 (1953).
80. Lakshmanan, A. et al. A combined experimental and crystal plasticity study of grain size effects in magnesium alloys. *J. Magnes. Alloy.* **11**, 4445–4467 (2023).
81. Gu, Y., Stiles, C. D. & El-Awady, J. A. A statistical perspective for predicting the strength of metals: Revisiting the Hall–Petch relationship using machine learning. *Acta Mater.* **266**, 119631 (2024).
82. Byrd, R. H., Lu, P., Nocedal, J. & Zhu, C. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.* **16**, 1190–1208 (1995).
83. Golub, G. H., Hansen, P. C. & O'Leary, D. P. Tikhonov regularization and total least squares. *SIAM J. Matrix Anal. Appl.* **21**, 185–194 (1999).
84. Louis, A. A unified approach to regularization methods for linear ill-posed problems. *Inverse Probl.* **15**, 489 (1999).

85. Wilkinson, P. *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers*, 2 Ed. (Pearson Education India, 2006).
86. Cook, S. *CUDA Programming: A Developer's Guide to Parallel Computing with GPUs* (Newnes, 2012).
87. Kafka, O. L. et al. X-ray computed tomography analysis of pore deformation in IN718 made with directed energy deposition via in-situ tensile testing. *Int. J. Solids Struct.* **256**, 111943 (2022).
88. Hu, G., Zha, R., Wang, Y., Cao, J., Guo, P. Digital fringe projection for interlayer print defect characterization in directed energy deposition. In: *Proc. International Symposium on Flexible Automation*, p. V001T01A002 (American Society of Mechanical Engineers, 2024).
89. Asaro, R. J. Crystal plasticity. *J. Appl. Mech.* **50**, 921–934 (1983).
90. Griewank, A. Walther, A. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation* (SIAM Publications, 20080.
91. Bonnans, J. F. Shapiro, A. *Perturbation Analysis of Optimization Problems* (Springer Science & Business Media, 2013).
92. Blondel, M. et al. Efficient and modular implicit differentiation. *Adv. Neural Inf. Process. Syst.* **35**, 5230–5242 (2022).

Acknowledgements

The authors acknowledge helpful discussions of Dr. Nicolo Grilli on the realization of polycrystal simulations on MOOSE, Dr. Fan Chen, Jiachen Guo, and Jin Choi for discussions. The authors would like to acknowledge support from the Department of Defense Vannevar Bush Faculty Fellowship, USA N00014-19-1-2642, and from the NSF Engineering Research Center for Hybrid Autonomous Manufacturing Moving from Evolution to Revolution (ERC-HAMMER) under Award Number EEC-2133630.

Author contributions

Fanglei Hu: Conceptualization, Methodology, Software, Investigation, Writing - Original Draft; Steve Niezgoda: Validation, Writing - Review & Editing; Tianju Xue: Conceptualization, Methodology, Software, Resources, Supervision, Writing - Review & Editing; Jian Cao: Conceptualization, Methodology, Resources, Supervision, Writing - Review & Editing.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41524-025-01528-2.

Correspondence and requests for materials should be addressed to Tianju Xue or Jian Cao.