Integrating Materials and Manufacturing Innovation (2025) 14:303–319
https://doi.org/10.1007/s40192-025-00405-6

TECHNICAL ARTICLE

MicroProcSim: A Software for Simulation of Microstructure Evolution

Md Maruf Billah[1] · Muhammed Nur Talha Kilic[2] · Md Mahmudul Hasan[1] · Zekeriya Ender Eger[1] · Yuwei Mao[2] · Kewei Wang[2] · Alok Choudhary[2] · Ankit Agrawal[2] · Veera Sundararaghavan[3] · Pınar Acar[1]

Received: 21 February 2025 / Accepted: 23 May 2025 / Published online: 23 June 2025

Abstract
Understanding the large deformation behavior of materials under external forces is crucial for reliable engineering applications. The mechanical properties of materials depend on their underlying microstructures, which change over time during deformation. Experimental observation of these processes is time-consuming and influenced by various conditions. Therefore, we developed `MicroProcSim`, a physics-based simulation tool to replicate the deformation process of cubic microstructures. `MicroProcSim` can predict the evolution of texture, represented by the orientation distribution function (ODF), over time under various loads and strain rates. This software package can be run on both Windows and Linux operating systems. Unlike conventional crystal plasticity finite element software, `MicroProcSim` offers a distinct advantage by rapidly generating deformed textures, as it bypasses incorporating grain morphology. Furthermore, comparisons with existing experimental and computational studies on texture evolution have demonstrated that this software seamlessly replicates real-world material processing conditions through a simple modification of a single input matrix.

Keywords Crystal plasticity modeling · Processing simulation · Strain rate · Microstructure · ODF

Introduction

The crystallographic texture of microstructures plays a pivotal role in determining the micro-scale characteristics of materials [1–3]. Controlling polycrystalline microstructures is essential in materials design. This is due to orientation-dependent material properties, such as stiffness and yield strength, which can vary as microstructures evolve during deformation [4–6]. Effective management of microstructures ensures the reliability of the desired material's performance. A significant amount of research, both experimental and theoretical, has been devoted to understanding and predicting the development of crystallographic textures in metallic microstructures. Textures that emerge due to specific monotonic deformations in cubic polycrystals have been studied extensively [7–9]. Particular focus has been given to textures arising from uniaxial compression or tension, plane strain compression, and simple shear [10, 11]. The significance

✉ Pınar Acar
pacar@vt.edu

Md Maruf Billah
mdmaruf@vt.edu

Muhammed Nur Talha Kilic
talha.kilic@u.northwestern.edu

Md Mahmudul Hasan
mdmahmudul@vt.edu

Zekeriya Ender Eger
endereger@vt.edu

Yuwei Mao
yuweimao2019@u.northwestern.edu

Kewei Wang
keweiwang2019@u.northwestern.edu

Alok Choudhary
a-choudhary@northwestern.edu

Ankit Agrawal
ankit-agrawal@northwestern.edu

Veera Sundararaghavan
veeras@umich.edu

[1] Department of Mechanical Engineering, Virginia Tech, Blacksburg, VA 24061, USA

[2] Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL 60208, USA

[3] Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, USA

of these deformations lies in their application in standard mechanical tests such as channel die compression, uniaxial compression, and torsion [12]. These tests are pivotal for assessing the mechanical behavior of materials under large strains. Furthermore, these deformation modes indicate the stresses encountered during various industrial forming processes, including extrusion, wire drawing, and rolling [13]. The ability to replicate these stress conditions in laboratory settings is crucial for predicting material performance, discovering new materials, and optimizing industrial manufacturing processes [14].

Various computational approaches have been developed to model how crystalline materials deform and how their textures evolve under different loading conditions. At the foundational level, there are simpler formulations, such as the original Taylor model [15] and its relaxed variants [16, 17]. These methods are straightforward and can provide valuable insights into the deformation behavior of crystalline aggregates under uniform strain assumptions. Intermediate complexity approaches include cluster-based frameworks such as Lamel model [18–20], advanced grain interaction (GIA) method [21–23], and relaxed grain cluster (RGC) schemes [24, 25]. These methods strike a balance between computational efficiency and capturing more realistic material behaviors, making them suitable for many practical applications. Moving toward more advanced methods, the self-consistent family of models integrates additional material behaviors. These include viscoplastic self-consistent (VPSC) [26, 27], elastic-rigid-plastic self-consistent (EPSC) [28, 29], and elastic-viscoplastic self-consistent (EVPSC) [30, 31] formulations. By considering interactions between individual grains and their surrounding matrix, these models offer improved accuracy in predicting texture evolution and anisotropic material responses. At the cutting edge of crystal plasticity modeling are full-field approaches, such as spectral methods leveraging fast Fourier transforms [32] and crystal plasticity finite element methods (CPFEM) [33]. These sophisticated techniques capture detailed microstructural responses, accommodating complex boundary conditions and local heterogeneities, albeit at a higher computational cost. Each method presents unique strengths, with their applicability determined by the specific phenomena of interest and the computational resources available. By selecting the appropriate approach, researchers can tailor their models to achieve the best compromise between accuracy and efficiency for their investigations.

In the field of texture development modeling, the orientation distribution function (ODF) has been established as an efficient and convenient method of representing microstructural texture because it serves as a one-point probability descriptor of grain orientation while simplifying complex analyses [34–41]. This technique has received attention because of its ability to provide a detailed quantitative description of textures, enhancing our understanding and application of ODFs [42–45]. The shift toward ODF-based techniques for microstructure modeling reflects their capability in capturing the complexity of texture evolution. In particular, we have utilized existing established finite element approaches in the development of `MicroProcSim`. This software utilizes the finite element-based ODF scheme with piecewise polynomial interpolation functions representing the texture over Rodrigues orientation space. This representation offers several key benefits. The simplicity and localized nature of these polynomial functions allow them to effectively model sharp textures. Furthermore, the finite element framework facilitates the construction of texture transformation analogs, such as interpolation, differencing, and projection for ODFs [12]. For example, the piecewise polynomial ODFs driven from deformation evolution are evaluated by employing well-established finite element methods to solve the ODF conservation equation for hyperbolic conservation laws. Furthermore, Rodrigues parameters uniquely map each orientation to a specific position within the Rodrigues fundamental region, ensuring a single, unambiguous representation [46].

The primary strength of CPFEM lies in its ability to explicitly capture the mechanical interactions among crystals within a polycrystal, without relying on homogenization assumptions. By incorporating constitutive formulations at the level of individual shear systems, CPFEM offers a framework capable of modeling physics-based, multiscale internal-variable plasticity, including various size-dependent effects and interface mechanisms [47, 48]. It also enables detailed access to both intra- and inter-grain deformation behaviors, making it particularly valuable for investigating grain boundary phenomena [49, 50]. In contrast, ODF-based methods offer a more efficient and simplified alternative, particularly suited for large-scale applications or cases where a statistical representation of texture suffices. This formulation cannot account for deformation mechanisms e.g., grain boundary sliding, non-crystallographic rotation, and twinning, which significantly influence reorientation in many crystal plasticity problems, limiting its applicability in capturing complex microstructural evolution accurately. The current ODF-based methods are particularly advantageous when global texture evolution rather than local stress–strain distributions is the primary focus. The statistical nature of ODFs also makes them well-suited for uncertainty quantification and inverse design problems where computational efficiency is paramount [51, 52].

In order to advance the deformation simulation of the materials, we have developed a software package called, '`MicroProcSim`,' which can capture the texture evolution of the cubic microstructures under different loading conditions. This software is capable of simulating a wide range of material processing conditions, where

various deformation modes contribute to altering the microstructural texture. Additionally, the strain rate and initial microstructural texture can be customized to replicate real experimental conditions. However, this article shows only three pure normal strain cases, three pure shear strain cases, and three plane strain deformation cases. The development of `MicroProcSim` revolutionizes materials science and engineering by enabling efficient simulation of microstructural texture evolution under various loading conditions. It overcomes experimental limitations, accelerates research, and reduces resource demands, allowing researchers to address advanced questions like optimizing material properties and process parameters. Notably, several studies [3, 10, 11, 53–55] have already published work utilizing `MicroProcSim`, demonstrating its widespread adoption and impact.

The article is structured as follows: the underlying physics behind the code development for the presented software will be discussed in section "Software Background", along with an illustrative guideline for operating the software. Section "Software Architecture" describes the architecture of this software. Following this, a few examples of microstructure evolution under different process conditions will be presented in section "Illustrative Examples". Then section "Extension of the Code to Different Operating Systems" delineates the extension of the software that runs on different operating systems. Later, section "Comparative Analysis" will present several experimental and computational studies that report texture evolution. These studies will be compared with the textures generated by `MicroProcSim` under similar deformation processes. Then, a summary of the computational costs associated with the example simulations will be provided in section 'Computational Costs". Finally, comprehensive conclusions will be drawn in section "Conclusion".

Software Background

Representation of Crystallographic Orientations

This software uses ODFs as a probability descriptor to represent the crystallographic textures of metallic microstructures. The ODFs essentially relate to the volume densities of the crystallographic orientations and are utilized to calculate the homogenized mechanical properties through local finite element discretization methods. In this work, the axis-angle parametrization of Rodrigues orientation space is employed, where the axis of rotation is scaled as $\mathbf{r} = \mathbf{n}\tan(\theta/2)$; here, $\mathbf{n}$ and θ represent the rotation axis and the angle of rotation, respectively [12]. The lattice orientation, $\mathbf{R}$, and the Rodrigues parameter, $\mathbf{r}$, can be expressed by the relationship in Eq. (1) [9].

$$\mathbf{R} = \frac{1}{1+\mathbf{r}.\mathbf{r}}(\mathbf{I}(1-\mathbf{r}.\mathbf{r}) + 2(\mathbf{r}\otimes\mathbf{r} + \mathbf{I}\times\mathbf{r})) \tag{1}$$

The vector ($\mathbf{r}$) can be represented as shown in Fig. 1a which operates in a three-dimensional space, offering advantages over two-dimensional stereographic projections. The geometry of the Rodrigues projective representation is influenced by both the symmetry of the object or lattice being studied and the specific characteristics of the Rodrigues space. Various researchers extensively documented the mathematical principles underlying Rodrigues space which can be accessed through the referred articles [56–58]. In this representation, orientation vectors are constrained by a maximum magnitude, which corresponds to the highest possible orientation achievable in a particular direction for a given symmetrical volume. When mapped in three dimensions, these vector endpoints form intricate polyhedral shapes as illustrated in Fig. 1b, contrasting with the spherical forms used in stereographic projections or the rectangular

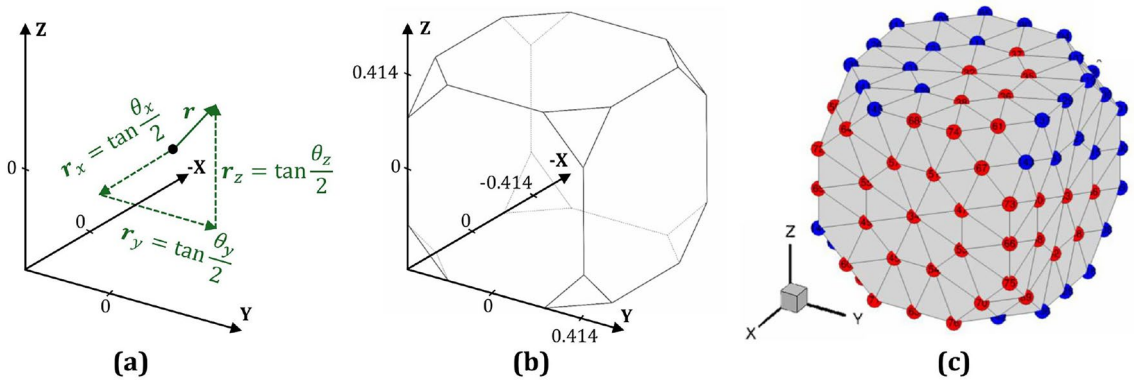

Fig. 1 **a** Schematic representation of the Rodrigues parameter ($\mathbf{r}$) defined in terms of orientation angles, **b** fundamental Rodrigues orientation space for an FCC crystal, and **c** finite element mesh representation of the Rodrigues orientation space, with ODFs specified at nodal points. Red points denote independent ODFs, while blue points indicate dependent ODFs

configurations seen in Euler angle coordinate systems [59]. The dimensions of the cubic Rodrigues fundamental space are determined by its constituent vectors. Along the axes, the shortest vectors that reach the surface have a magnitude of $\tan(45°/2) = \sqrt{2} - 1$. The vectors extending to the truncating triangles measure $\tan(60°/2) \approx 0.58$ in magnitude. For cubic symmetry, the maximum rotation angle occurs around a $(1, 1, \sqrt{2} - 1)$ axis, with its vectors having a magnitude of $\tan(62.8°/2) \approx 0.61$. Within the fundamental zone, each point corresponds to a unique orientation relative to a reference frame lacking symmetry, collectively forming what is known as the reduced orientation set.

The orientation spaces of polycrystalline face-centered cubic (FCC) structures are simplified due to reduced orientation set or crystal symmetries; among the 145 elements of the ODF, 76 remain independent representing 76 unique orientations, as shown in Fig. 1c [38, 60]. By using a local finite element method, the fundamental region is discretized, which involves N independent nodes and N_{elm} finite elements, each with N_{int} integration points. The unique ODF value at each nodal point in the mesh is intricately linked to the volume density of the corresponding crystallographic orientation. Subsequently, the set of ODFs (A) for any microstructure over the fundamental region must be normalized to unity, as expressed in Eq. (2) where w_m is the integration weight associated with the m-th integration point, and $|J_n|$ is the Jacobian determinant of the n-th element [52, 61, 62].

$$\int_{\Omega} A(\mathbf{r}, t)dv = \sum_{n=1}^{N_{elm}} \sum_{m=1}^{N_{int}} A(\mathbf{r_m}, t) w_m |J_n| \frac{1}{\left(1 + \mathbf{r_m}.\mathbf{r_m}\right)^2} = 1 \quad (2)$$

Microstructural Texture Evolution

The applied force can change the microstructural orientations, represented by the ODFs, during deformation. This evolution occurs within the constraints of volume normalization and the conservation of ODFs from the initial time ($t = 0$) to the final time ($t = t$) [10]. The conservation rule for crystallographic orientations can be described by the following Eq. (3), using the Eulerian rate form. In this equation, the reorientation velocity (υ) is crucial for the evolution of ODFs and can be expressed by Eq. (4) [53]. Here, the spin vector is denoted by ω, which is a vector form of the tensor $\dot{\boldsymbol{R}}^e \boldsymbol{R}^{e^T}$. According to the crystal plasticity model, elastic deformation gradient $\boldsymbol{F}^e$ assists to find $\boldsymbol{R}^e$ through $\boldsymbol{F}^e = \boldsymbol{R}^e \boldsymbol{U}^e$, where $\boldsymbol{U}^e$ represents the polar decomposition's unitary tensor [63].

$$\frac{\partial A(\mathbf{r}, t)}{\partial t} + \nabla A(\mathbf{r}, t) \cdot \upsilon(\mathbf{r}, t) + A(\mathbf{r}, t) \nabla \cdot \upsilon(\mathbf{r}, t) = 0 \quad (3)$$

$$\upsilon(\mathbf{r}, t) = \frac{1}{2}(\omega + (\omega \cdot r)r + \omega \times r) \quad (4)$$

Furthermore, the reorientation velocity (υ) is utilized to form the macro velocity gradient ($\mathbf{L}$) as formulated in Eq. (5). The microstructure constitutive model is used as the governing equation of ODF evolution, which can be simply expressed in terms of the macro velocity gradient. On the other hand, Taylor's macro-micro hypothesis [15] concludes that the reorientation velocity can be linked to the velocity gradient, where the macro velocity gradient should be equal to the crystal velocity gradient. Subsequently, the reorientation velocity is calculated from a constitutive model which is rate-independent. This means that the final texture $A(\mathbf{r}, t)$ is derived from the initial texture $A(\mathbf{r}, 0)$ by employing the previously mentioned finite element discretization method in Rodrigues space, along with this constitutive model.

$$\mathbf{L} = \mathbf{S} + \mathbf{R} \sum_{\alpha} \dot{\gamma}^{\alpha} \bar{\mathbf{T}}^{\alpha} \mathbf{R}^T \quad (5)$$

$$\dot{\gamma}^{\alpha} = \quad \dot{\gamma}^0 \left(\frac{\tau^{\alpha}}{s} \right)^{\frac{1}{m}} \operatorname{sgn}\left((\frac{\tau^{\alpha}}{s}) \right) \quad (6)$$

Every process condition results in a distinct deformation category, such as tension, compression, or shear, which can be defined with the timeframe of simulation in the input of the software. Ultimately, the macro velocity gradient controls the overall deformation process of crystal plasticity, which is used to evaluate the evolution of the ODF. However, to achieve a specific final texture from a given initial texture, the velocity gradient can be treated as an unknown variable. The relationship between lattice rotation ($\mathbf{R}$), lattice spin ($\mathbf{S}$), and macro velocity gradient ($\mathbf{L}$) can be formulated as shown in Eq. (5). The Schmid tensor and rate of shear of the α^{th} slip system are represented by $\bar{\mathbf{T}}^{\alpha}$ and $\dot{\gamma}^{\alpha}$, respectively. The shearing rate is defined by the Eq. (6), where s is the slip system hardness (considered uniform across all slip systems), m is the strain rate sensitivity, $\dot{\gamma}^0$ is a reference shearing rate, and τ^{α} is the resolved shearing rate on slip system α [9]. The material parameters $\dot{\gamma}^0$, m, and s were considered as 1s^{-1}, 0.05, and 27.17MPa, respectively. The macro velocity gradient expression essentially comprises two primary components: one is the lattice spin, regarding the deformation gradient ($\boldsymbol{F}$), which can be further decomposed into elastic ($\boldsymbol{F}^e$) and plastic ($\boldsymbol{F}^p$) deformation where $\boldsymbol{F} = \boldsymbol{F}^e \boldsymbol{F}^p$. The other term represents the rotated plastic velocity gradient resulting from the summation of shearing of all slip systems, expressed by $\mathbf{R} \sum_{\alpha} \dot{\gamma}^{\alpha} \bar{\mathbf{T}}^{\alpha} \mathbf{R}^T$. However, the macro velocity gradient can be expressed in a simple matrix form as shown in Eq. (7), where the decomposition of the velocity gradient tensor reveals several fundamental physical processes. When broken down mathematically, distinct terms

emerge that each correspond to different types of motion. The initial component describes tensile deformation along the x axis, followed by a term capturing rolling process along the y axis. The remaining three components represent pure shear deformation occurring in different spatial orientations. By combining these basic elements in varying proportions (adjustment of the process parameters α_1, α_2, α_3, α_4, and α_5), any incompressible deformation pattern and deformation strain rate can be mathematically represented. This framework provides a complete basis for describing how the material can distort while maintaining a constant volume. The detailed derivation can be accessed from the referred article [9].

$$\mathbf{L} = \alpha_1 \begin{bmatrix} 1 & 0 & 0 \\ 0 & -0.5 & 0 \\ 0 & 0 & -0.5 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix} + \alpha_3 \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \alpha_4 \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} + \alpha_5 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{7}$$

The entire deformation process is divided into n discrete steps, each with a duration of Δt. This autoregressive framework allows to model the deformation process through a series of steps, where the load at each step, F_i (which can be for compression/tension, plane strain compression, or shear), governs the deformation parameter (ϕ_{F_i}) via the macro velocity gradient ($\mathbf{L}$). The texture of the microstructure at any given step in this deformation process can be described using the ODF, as represented in Eq. (8).

$$\begin{aligned} A(\mathbf{r}, \Delta t) &= \phi_{F_0}(A(\mathbf{r}, 0)) \\ A(\mathbf{r}, 2\Delta t) &= \phi_{F_1}(A(\mathbf{r}, \Delta t)) \\ &\vdots \\ A(\mathbf{r}, n\Delta t) &= \phi_{F_{n-1}}(A(\mathbf{r}, (n-1)\Delta t)) \end{aligned} \tag{8}$$

A precisely defined set of forces, $\mathcal{F} := \{f_1, f_2, \ldots, f_k\}$, is employed to select a specific force, $F_i \in \mathcal{F}$, for each deformation process of this physics-based simulation. The optimal final ODF set is achieved through an optimal process path, P^*. This path is determined by the goal of obtaining desired material properties while adhering to the ODF normalization constraint as studied previously [10].

Software Architecture

This physics-based deformation simulation shows how ODF changes with time under different loading conditions and varying strain rates. Therefore, the inputs for the simulator include the definition of initial texture (which may preferably be defined as randomly oriented texture or any other texture if the initial texture data is available), loading scenario, and the corresponding strain rate. It also considers the necessary slip parameters of the corresponding materials for the deformation of the cubic microstructures. The total deformation time is 0.1 sec, and the code is developed to provide the deformed ODF in every 0.01 sec time step. This allows to report the evolution of the textures during the deformations. Fig. 2 shows the architecture of `MicroProcSim`.

The code is designed for the cubic microstructures (FCC). According to the ODF definition, the coarse mesh for cubic microstructures involves 76 independent crystal orientations.

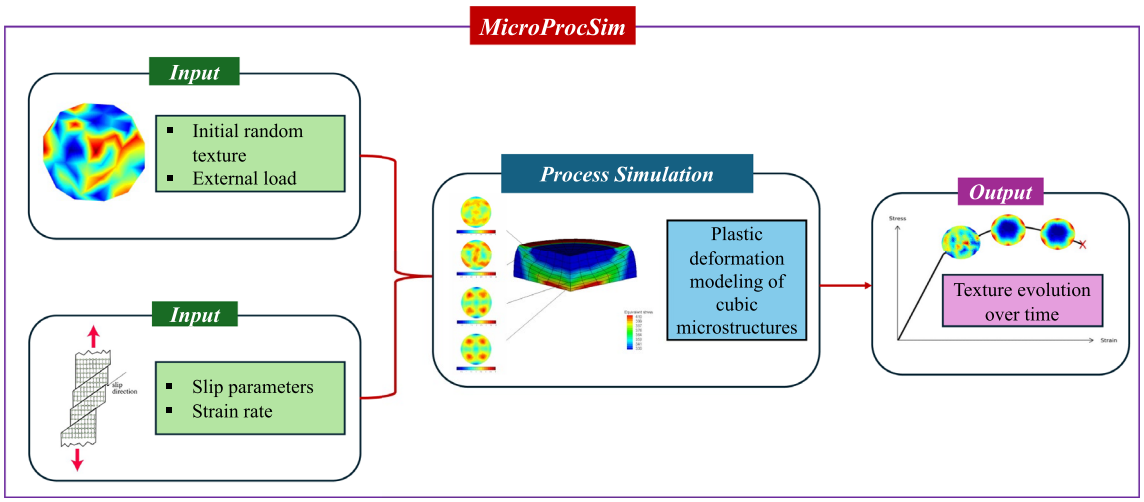

Fig. 2 Architecture of the `MicroProcSim` tool that includes the input, process simulation, and software output

However, this simulation provides 145 ODFs as output in each time step, including the ODFs for the dependent orientations arising due to crystallographic symmetries.

Software Functionalities

The simulation is executed using the `app.exe` file provided in the project folder. The execution of this file is necessary to obtain the simulation results. The loading conditions and initial input ODFs are managed through two specific text files: '`param.txt`' and '`Input_ODF.txt`,' respectively. In the project folder, a MATLAB script named '`process.m`' is written to automate the simulation for multiple runs, i.e., sequential processes. The simulation code is configured for a loading time of 0.1 s (single process). In order to run a process over 0.3 s (3 consecutive processes), the code must be executed three times sequentially, and the final ODFs from each run serve as the input for the subsequent run. The ODFs of independent crystal orientations must satisfy normalization constraints (Eq. (2)). Additionally, the ODF values cannot be negative. The MATLAB code also ensures that these conditions are met. It also saves the desired output, such as deformed ODFs and Cauchy stress tensor, at each step.

Input File Requirements

- The input file for loading conditions must be named '`param.txt`' to ensure that the `app.exe` runs correctly. Any deviation in naming will result in the application failing to execute. Therefore, the accompanying MATLAB script is designed to consistently produce a file named `param.txt` to accommodate any loading condition.
- For combined loading conditions (e.g., tension and shear in the *xy*-plane), two non-zero strain rates corresponding to the respective loading conditions must be specified in the `param.txt` file.
- The current provided '`input.txt`' file is only valid for randomly oriented microstructure. The user can modify this file as well to input any preferable microstructure texture.

Output and Analysis

- A MATLAB script, '`process.m`,' automates multiple runs and saves the necessary output files for analysis. The script requires two `.mat` files, '`newmesh.mat`' and '`FCC_volumefraction.mat`,' to be loaded before execution. The script accommodates single or combined loading conditions over multiple runs. However, for multiple runs, the first run must be completed, and its output must be used as input for the next run with a different loading condition.
- The normalization constraint mentioned in Eq. (2) is not fully satisfied or exactly equal to 1 due to minor numerical errors. Typically, this error is less than 0.1%. However, it can propagate if multiple loading steps are performed. To mitigate this issue, the final ODF used as the initial input for the next step is normalized before applying the subsequent loading. The `FCC_volume-fraction.mat` file helps to check the normalization constraint and update the output ODF accordingly, especially when performing multi-step loading simulations.
- This MATLAB script requires the '`mapping.txt`' file to map the software output ODF to the '`newmesh.mat`' ODF, as the coordinates of the ODF in the software system and MATLAB files are different. Furthermore, '`newmesh.mat`' is utilized to plot the output ODFs in Rodrigues space. The 'PlotFR' function used in the MATLAB script also requires a few additional functions, which are included in this folder as well.
- Each run of the simulation generates a '`stress-strain.out`' file containing the Cauchy Stress Tensor, which the MATLAB script also saves. Each run produces ten Cauchy Stress tensors; thus, three runs, for example, will result in thirty tensors. The Cauchy Stress tensor, originally a 3×3 matrix, is converted to a column matrix in MATLAB. The file `Cauchy.mat` contains columns representing the Cauchy Stress Tensors, which need to be reshaped back into 3×3 matrices for further analysis.

The process can also be executed directly via '`app.exe`' without using MATLAB for a single process. In this case, ensure that the required '`param.txt`' and '`Input_ODF.txt`' files are present alongside other necessary files. The folder includes five pre-configured '`param.txt`' files for different loading conditions: tension/compression along *x*-direction, plane strain compression along *y*-direction, *xy*-shear, *xz*-shear, and *yz*-shear. To run the process for tension, for example, rename '`param_tension.txt`' as '`param.txt.`' and then execute '`app.exe`.'

Illustrative Examples

This section presents example results along with the process conditions used to generate them. Three distinct cases are analyzed to explore the evolution of cubic microstructural textures: simple tension/compression, simple shear, and plane strain compression (as in a rolling process). These processes are summarized in Table 1, highlighting the three fundamental loading directions and their corresponding processing parameters (α_1 to α_5). A unit strain rate of $1\ s^{-1}$ is employed across all nine cases. The framework allows for simulating various processes and strain rates by modifying the processing parameter values. This can be achieved

Table 1 Loading types and corresponding parameters (in all cases, the strain rate is 1 s^{-1})

Loading type	Direction or plane	α_1	α_2	α_3	α_4	α_5
Tension or compression	x	1	0	0	0	0
	y	− 0.5	0.75	0	0	0
	z	− 0.5	− 0.75	0	0	0
Shear	xy	0	0	1	0	0
	xz	0	0	0	1	0
	yz	0	0	0	0	1
Rolling or plane strain compression	zy	0	1	0	0	0
	yx	1	− 0.5	0	0	0
	xz	− 1	− 0.5	0	0	0

by editing the `param.txt` file, either manually or using the MATLAB script `process.m`. The simulations were run for a total time of 0.1 seconds, divided into ten equal time steps of 0.01 seconds each, as inherently defined by the software. The initial microstructural texture was assumed to be randomly oriented, with all ODF values set equally ($A(\mathbf{r}, 0) \approx 2.42$), indicating an equal probability for all orientations.

The velocity gradient tensor, denoted as $\mathbf{L}$, is expressed in matrix form as a 3×3 square matrix. This matrix can be tailored to enable various loading conditions and strain rates by adjusting specific processing parameters. For applying normal stress (tension/compression), the diagonal components of $\mathbf{L}$ need to be modified appropriately. For tension or compression along the x-direction, perpendicular to the yz -plane, L_{11} is set to 1, L_{22} to −0.5, and L_{33} to −0.5, while all other components are set to zero, preserving volume incompressibility. Similarly, for normal stress along the y-direction, perpendicular to the xz-plane, L_{11} is set to −0.5, L_{22} to 1, and L_{33} to −0.5, with all other components zero. For normal stress along the z-direction, perpendicular to the xy-plane, L_{11} is set to −0.5, L_{22} to −0.5, and L_{33} to 1, again ensuring volume consistency. These loading conditions can be achieved by adjusting the values of α_1 and α_2 as shown in the accompanying Table 1. The output microstructure textures in Rodrigues orientation space are illustrated in Fig. 3, where the strain rate is 1 s^{-1}, the total time is 0.1 s, and the equivalent strain is 0.1 mm/mm. These simulations were performed using a single execution of the application (`app.exe`).

To apply simple shear strain along the three fundamental directions (γ_{xy}, γ_{xz}, and γ_{yz}), the diagonal elements of the $\mathbf{L}$-matrix must be set to zero, with only two off-diagonal elements set to 1. For γ_{xy}, which acts on the plane perpendicular to the x-axis and is directed along the y-axis, $L_{12} = L_{21} = 1$, while all other components are zero. This configuration ensures constant volume during the process. Similarly, for γ_{xz}, which acts on the plane perpendicular to the x-axis and is directed along the z-axis, $L_{13} = L_{31} = 1$, with the remaining components set to zero and for γ_{yz}, which acts on the plane perpendicular to the y-axis and is directed along the z-axis, $L_{23} = L_{32} = 1$, with all other components set to zero, preserving volume incompressibility. To achieve

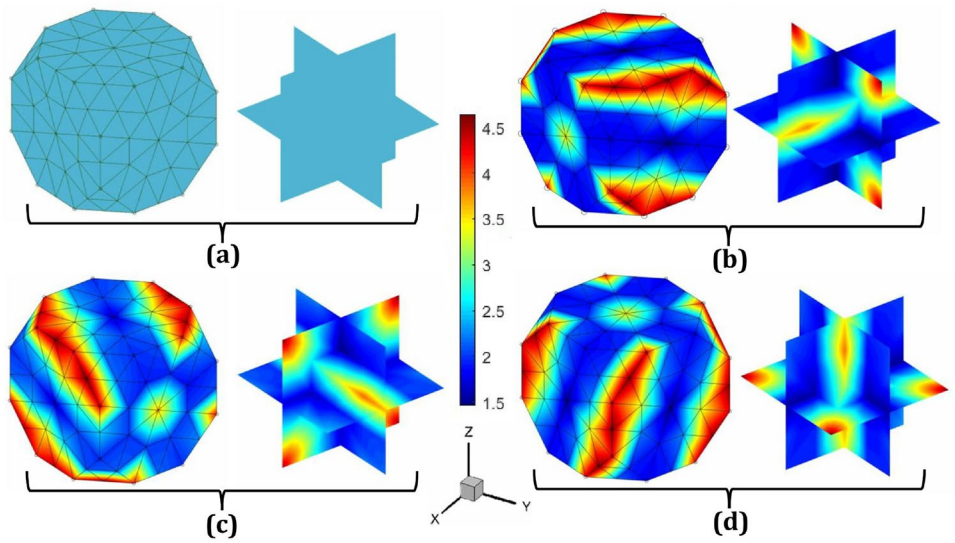

Fig. 3 Sample microstructures on Rodrigues orientation space: **a** Initial texture with random orientation, $A(\mathbf{r}, 0) \approx 2.42$; final texture after applying normal strain (tension/compression) along **b** the x-direction, perpendicular to the yz-plane, **c** the y-direction, perpendicular to the xz-plane, and **d** the z-direction, perpendicular to the xy-plane. In all cases, the strain rate is 1 s^{-1}, the total time is 0.1 s, and the equivalent strain is 0.1 mm/mm

these loading conditions, only the values of α_3, α_4, and α_5 need to be adjusted, as shown in the accompanying Table 1. The resulting microstructure textures in the Rodrigues orientation space are illustrated in Fig. 4, where the strain rate is 1 s^{-1}, the total time is 0.1 s, and the equivalent strain is 0.1 mm/mm. These simulations were conducted with a single execution of the `app.exe`. Furthermore, due to force balance in mechanics, γ_{yx}, γ_{zx}, and γ_{zy} are equivalent to γ_{xy}, γ_{xz}, and γ_{yz}, respectively. Thus, it is not necessary to perform separate simulations for these strain components.

In addition to simulating microstructure evolution during the rolling process, plane strain compression can be applied in various directions equivalent to rolling by modifying the elements of the deformation gradient tensor matrix **L**. This adjustment involves ensuring that all off-diagonal elements and one diagonal element are zero, depending on the rolling direction. For instance, rolling along the *zy*-plane, where *z* is the normal direction, *y* is the rolling direction, and *x* is the transverse direction, requires setting $L_{22} = 1$ and $L_{33} = -1$, which satisfies the incompressibility constraint. Similarly, for rolling along the *yx*-plane, with *y* as the normal direction, *x* as the rolling direction, and *z* as the transverse direction, $L_{11} = 1$ and $L_{22} = -1$ must be set. For rolling along the *xz*-plane, where *x* is the normal direction, *z* is the rolling direction, and *y* is the transverse direction, the conditions $L_{11} = -1$ and $L_{33} = 1$ ensure constant volume. These specific loading conditions can be achieved by adjusting the values of α_1 and α_2, as summarized in Table 1. The resulting microstructure textures on the Rodrigues orientation space are shown in Fig. 5, where the strain rate is 1 s^{-1}, the total

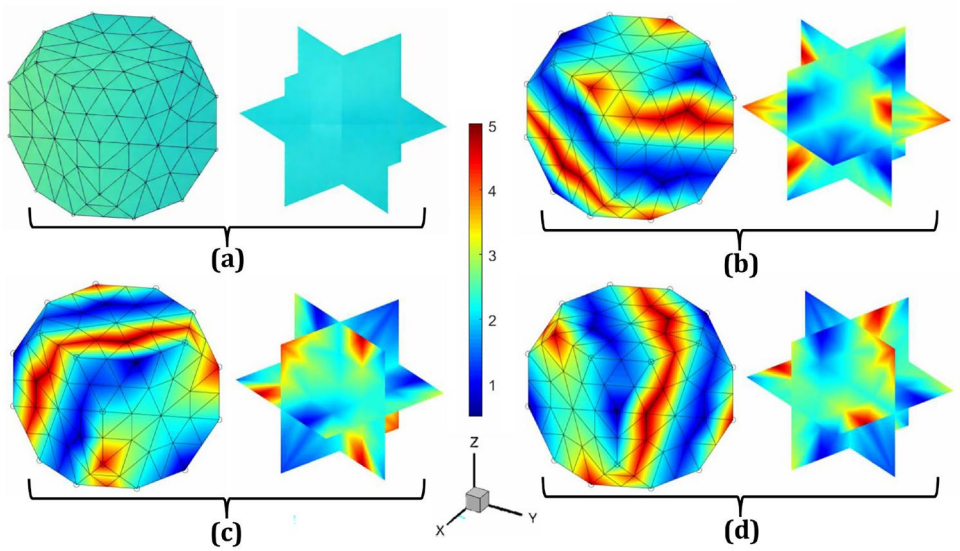

Fig. 4 Sample microstructures on Rodrigues orientation space: **a** Initial texture with random orientation, $A(\mathbf{r}, 0) \approx 2.42$; final texture after applying shear strain of **b** the *xy*-shear acted on the plane perpendicular to the *x*-axis and is directed along the *y*-axis, **c** the *xz*-shear acted on the plane perpendicular to the *x*-axis and is directed along the *z*-axis, and **d** the *yz*-shear acted on the plane perpendicular to the *y*-axis and is directed along the *z*-axis. In all cases, the strain rate is 1 s^{-1}, the total time is 0.1 s, and the equivalent strain is 0.1 mm/mm

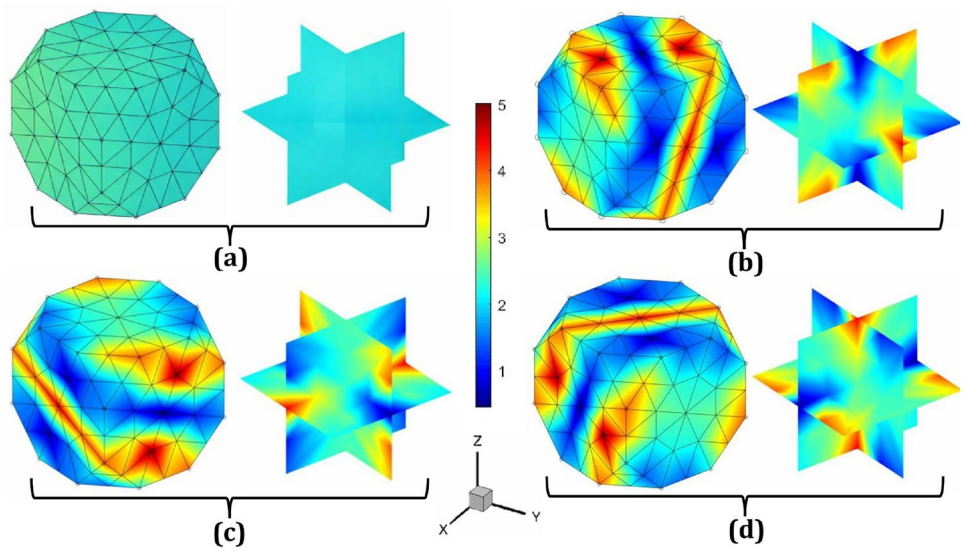

Fig. 5 Sample microstructures in Rodrigues orientation space: **a** Initial texture with random orientation, $A(\mathbf{r}, 0) \approx 2.42$; final textures after the plane strain compression or rolling process along **b** the *zy*-plane, where *z* is the normal direction, *y* is the rolling direction, and *x* is the transverse direction; **c** the *yx*-plane, where *y* is the normal direction, *x* is the rolling direction, and *z* is the transverse direction; and **d** the *xz*-plane, where *x* is the normal direction, *z* is the rolling direction, and *y* is the transverse direction. In all cases, the strain rate is 1 s^{-1}, the total time is 0.1 s, and the equivalent strain is 0.1 mm/mm

time is 0.1 s, and the equivalent strain is 0.1 mm/mm. Notably, these simulations were performed in a single execution of the application (`app.exe`).

Extension of the Code to Different Operating Systems

An operating system (OS) is a collection of programs that serves as an intermediary between application software and the computer hardware interface [64]. The OS is loaded by a bootloader, program, after which it facilitates the execution of hardware tasks. As illustrated in Fig. 6, communication between layers is bidirectional. By utilizing an OS, user programs have better interaction with the computer. The necessity for operating systems arises from the complexity of managing various hardware devices, including mouse, displays, and network interfaces. The OS additionally manages fundamental functions such as file systems, memory management, security, and multimedia execution. It also provides services to applications to prevent potential deadlocks and congestion [65].

In order to ensure that a program (e.g., `MicroProcSim`) can run on different operating systems, there are several strategies that can be used. The first option is recompiling the program with the specific libraries required for each operating system [66]. However, given the complexity of obtaining all necessary code files under present conditions, this approach may not always be feasible. An alternative solution involves creating a container [67], which provides a complete environment containing everything needed to run an application: code, runtime, system tools, system libraries, and settings. As illustrated in Fig. 6b, a program that was originally developed in a Windows operating system environment can be adapted to run in a Linux OS environment using a container. This method enables software, such as `app.exe`, to execute on Linux OS, even though it was originally designed for Windows.

Different operating systems utilize various system calls, which can be considered as the operating system's language. Examples include Windows API Calls [68] and POSIX Calls [69]. Third-party applications can facilitate the translation of these system calls from one OS to another, ensuring compatibility and functionality across diverse platforms.

In our research, we used the Windows application file of `MicroProcSim` called '`app.exe`,' which was originally designed to run on a Windows operating system and make Windows API calls. To run this application on a Linux environment, we utilized a tool called Wine [70]. Wine translates Windows API calls into POSIX calls. Given the diverse internal structures of various Linux distributions, we conducted this process on a server running Fedora. Detailed information about the Fedora setup is provided in Fig. 7.

In order to use the Wine application effectively, it is important to analyze the program to be run to determine its needs, such as graphical interface, sound, networking, and serial bus activity. During the installation of Wine, these requirements should be specified as arguments in the build command. Once Wine is built, a Windows application, such as `app.exe`, can run as an argument of the Wine application. However, because Wine introduces an additional layer, there will inevitably be a time difference compared to running an application built natively for Linux.

Comparative Analysis

In this section, several references will be utilized to compare the outputs from `MicroProcSim` with existing studies encompassing both computational and experimental investigations. Notably, most of the prior studies represent microstructures using pole figures rather than Rodrigues orientation space (ODF representation). To enable a meaningful

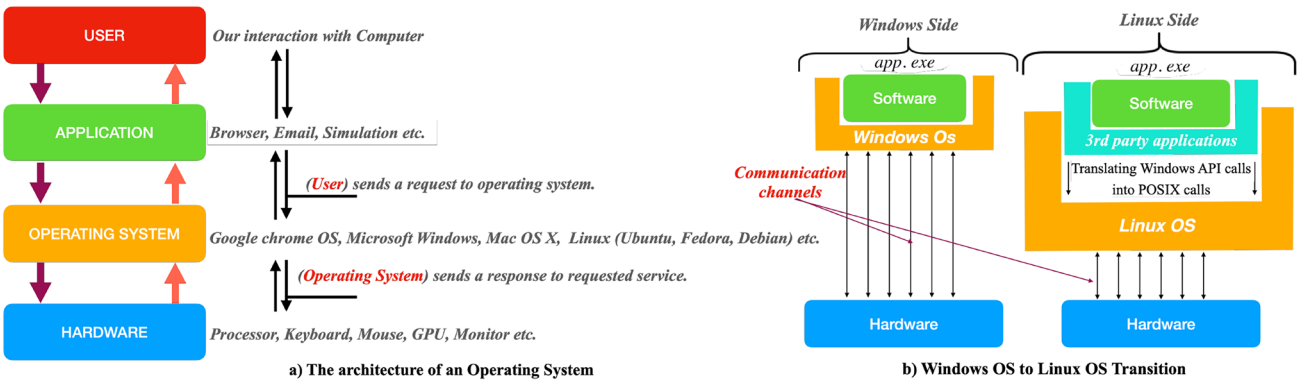

Fig. 6 **a** The general architecture of operating systems and their message-passing mechanisms through various layers and **b** additional components required for an operating system to execute external applications that are not runnable locally

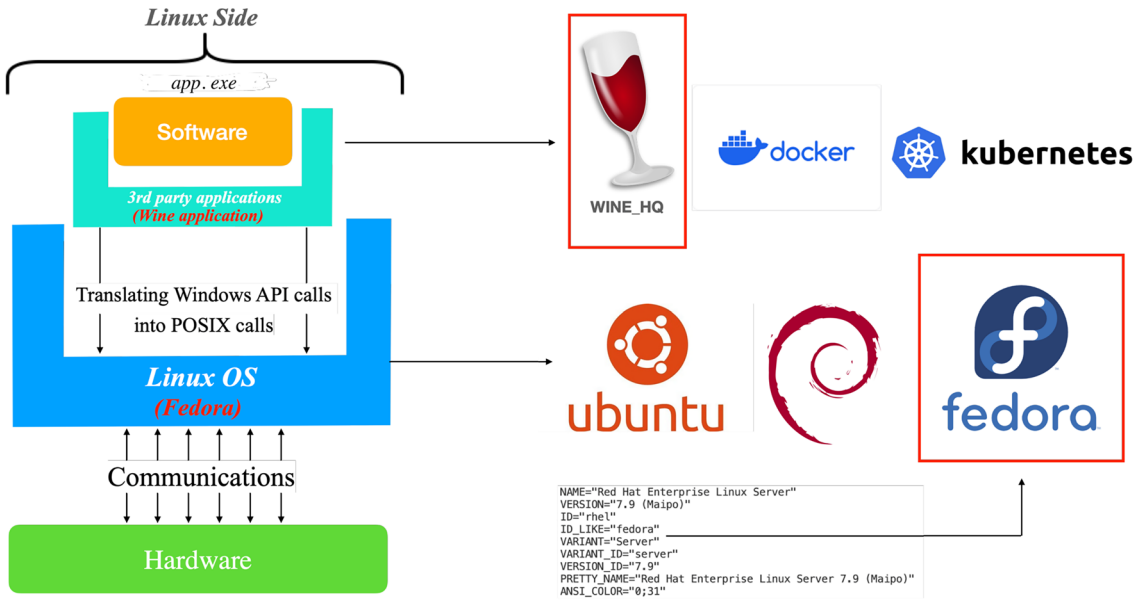

Fig. 7 A comprehensive description of the software utilized for specific layers, along with other relevant counterparts

comparison with these studies, the output ODFs have been converted into three pole figures, as required for the specific comparisons. The orientation distribution of crystals was transferred into pole figures generated for three distinct planes by utilizing the procedure of Barton et al. [71]. The pole density distribution, represented as $P(\mathbf{h}, y_i)$ describes the frequency of crystallographic orientations, where $\mathbf{h}$ indicates the plane normal vector and $y_1, y_2, \ldots, y_q$ represents specific positions on the unit sphere's surface for the measured diffraction planes. The mathematical relationship between ODF (A_j) and the pole density is established through a system matrix (M_{ij}) by $\sum_{j=1}^{k} M_{ij}A_j$. This relationship accounts for k independent ODFs determined in the analysis. To satisfy the physical constraint that the total volume fraction must be equal to unity, the modified pole density function $\left(P_i = P_i - M_{ik}/q_k\right)$ incorporates a normalization term, with the coefficients being adjusted accordingly for the first $(k-1)$ terms, $M_{ij} = M_{ij} - M_{ik}q_j/q_k$ for $j = 1, 2, \ldots, (k-1)$. However, the ODF can also be directly visualized through the pole figures using the MTEX software which is also a free and open-source toolbox widely utilized for texture analysis [72].

Bronkhorst et al. [73] conducted an experiment on oxygen-free high-conductivity (OFHC) copper, applying 37% true tensile strain to the randomly oriented microstructure of this FCC crystal, which exhibited isotropic properties. The final microstructure texture was documented using three distinct plane pole figures. To replicate their results, Yaghoobi et al. [74] employed the PRISMS-Plasticity TM modeling software. Instead of OFHC copper, they used the FCC 7075-T6 aluminum alloy microstructure while maintaining a similar strain level. In our study, we applied a 37% normal tensile strain to pure copper in three stages: an initial 13%, followed by 10%, and a final 10% strain $(1 - (1.13 \times 1.1 \times 1.1) \approx 37\%$ strain) along the z-direction, perpendicular to the xy-plane, as illustrated in Fig. 8a. To align with their methodology, we also considered an initially randomly oriented microstructure and used pole figures representing the same planes and directions, as shown in Fig. 8b–d. Analysis of the textural data reveals two primary orientational features. The presented pole figure analysis demonstrates that during tensile/compressive deformation, the polycrystalline grains undergo rotation, resulting in the alignment of either (1 1 1) or (1 0 0) crystallographic planes normal to the direction of applied stress. Furthermore, the application of simple normal strain results in an axisymmetric microstructure texture centered around the loading direction axis. The comprehensive observations derived from these pole figures indicate that the experimental and simulated microstructures were accurately captured using the presented `MicroProcSim` microstructure evolution software.

To compare shear textures, the high-pressure torsion (HPT) study by Duan et al. [75] and the additive friction stir deposition (AFSD) study by Griffiths et al. [76] have been selected. Shear strain plays a critical role in both the HPT and AFSD processes. In HPT, a combination of high compressive force and torsional rotation generates intense shear deformation in the material. The strain increases radially from the center to the edges due to varying tangential displacement, as shown in Fig. 9a. This extreme

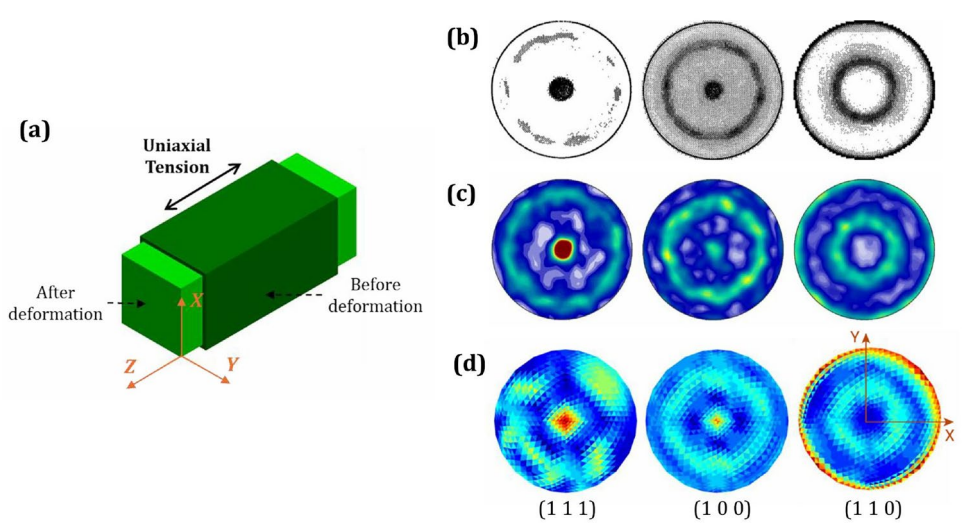

Fig. 8 Comparison of crystallographic textures of FCC crystal in terms of (1 1 1), (1 0 0) and (1 1 0) pole figures under uniaxial tensile loading at 37% true strain: **a** Schematic representation of a sample material during the deformation process, **b** experimental pole figures obtained from oxygen-free high-conductivity (OFHC) copper specimens, as reported by Bronkhorst et al. [73], **c** simulated texture evolution for 7075-T6 aluminum alloy using PRISMS-Plasticity TM modeling software [74], and **d** simulated texture of pure copper using `MicroProcSim` microstructure evolution software. All microstructures exhibited initially random textures prior to deformation. ((**b**) is reprinted from Ref. [73] with permission. **c** is reprinted from Ref. [74] with permission.)

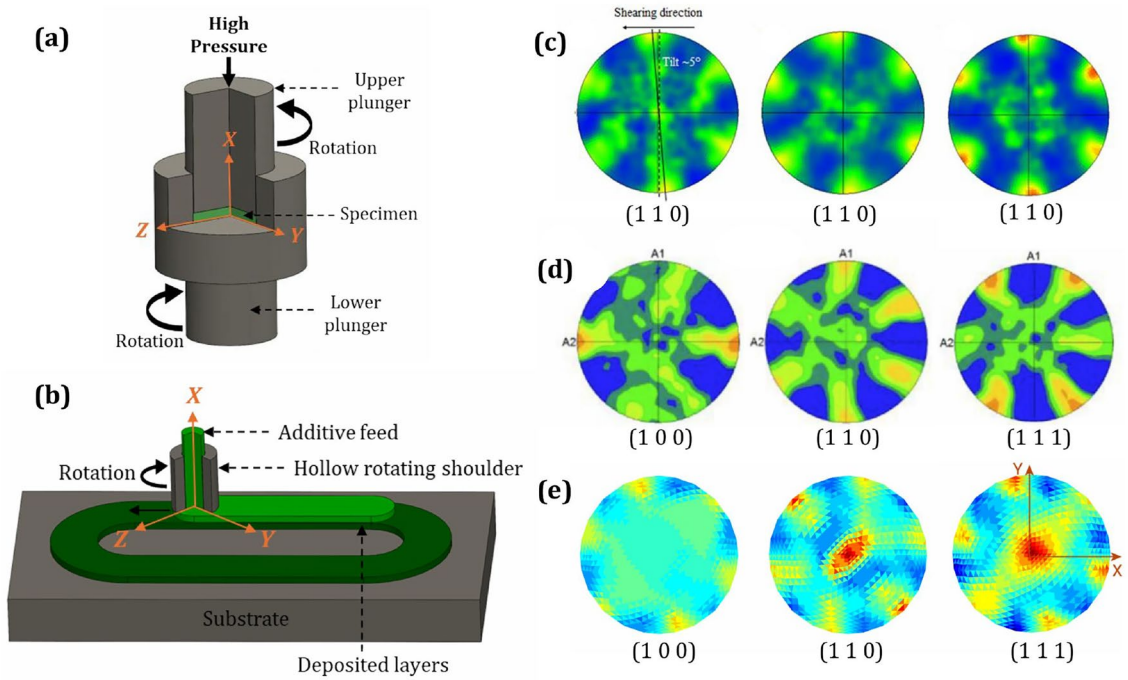

Fig. 9 Comparison of crystallographic textures after shear processing: **a** Schematic representation of a specimen during the high-pressure torsion (HPT) deformation process; **b** schematic representation of a specimen during the additive friction stir deposition (AFSD) process; **c** experimentally observed ideal torsion texture in terms of the (1 1 0) pole figure of G91 steel alloy, (left to right) processed by HPT, followed by annealing at 600 °C for 6 h and 24 h, as reported by Duan et al. [75]; **d** experimentally observed texture of deposited copper in terms of (1 0 0), (1 1 0), and (1 1 1) pole figures after the AFSD process, as reported by Griffiths et al. [76]; **e** simulated texture in terms of (1 0 0), (1 1 0), and (1 1 1) pole figures after simple shear on pure copper using `MicroProcSim` microstructure evolution software, starting from a randomly oriented initial microstructure and deformed to 0.1 mm/mm strain. ((**c**) is reprinted from Ref. [75] with permission. **d** is reprinted from Ref. [76] with permission)

deformation modifies the grain structure and texture, significantly enhancing material properties. In contrast, AFSD involves a hollow rotating tool that generates frictional heat to plasticize an additive feed material, which is then deposited onto a substrate, as illustrated in Fig. 9b. Shear strain in AFSD arises from the rotational and translational motion of the tool, causing localized deformation in the deposition zone. While HPT induces shear across the entire sample under uniform high pressure, AFSD produces localized shear to enable material flow and strong metallurgical bonding. Both processes utilize shear strain to achieve preferred grain texture refinement and improved mechanical properties. Although both shear and compressive strains exist in these processes, the shear strain magnitude is significantly higher than the compressive strain. This dominance of shear strain results in textures that resemble pure shear textures.

The HPT examination [75] analyzed both the microstructural features and textural characteristics of a Grade 91 steel containing 9% chromium. The material underwent high-pressure torsion processing followed by thermal treatment at 600°C. The pole figures of the resulting texture are represented in Fig. 9c. On the other hand, the AFSD research [76] examined how processing conditions affect microstructural development by comparing two metals with different responses to thermomechanical processing: an aluminum-magnesium-silicon alloy and pure copper. Both materials exhibit pronounced shear texture patterns. However, the study illustrates the pure copper texture in Fig. 9d for comparison with the simulated shear texture shown in Fig. 9e. To match the experimental setup, *xy*-shear was applied on the plane perpendicular to the *x*-axis and directed along the *y*-axis, as shown in Fig. 9a and b, where angular or rotational motion was applied around the *x*-axis. In the `MicroProcSim` simulation cases, the strain rate was set to 1 s^{-1}, the total simulation time to 0.1 s, and the equivalent strain to 0.1 mm/mm. The initial texture was a randomly oriented microstructure. In all the shear texture pole figures, approximately six equally spaced hotspots were observed along the circumference, as expected for shear textures if the pole figures are drawn in a manner consistent with the applied shear strain notation. The simulated texture pole figure shows reasonable agreement with the experimentally observed cubic texture after the simple shear process.

Another shear deformation process is Equal Channel Angular Extrusion (ECAE), as reported by Gazder et al. [77]. This study analyzed the resulting deformed textures using an alternative axis representation which leads to differed pole figure from those in previous cases. In the ECAE process depicted in Fig. 10a, shear strain arises as the material is forced through a die with intersecting channels of equal cross section. The severe deformation occurs at the intersection of the two channels where the material must change direction sharply. As the material flows through this region under high pressure, it undergoes simple shear deformation due to the abrupt change in velocity gradient along the shear plane. This creates large plastic deformation while preserving the overall shape and cross-sectional dimensions of the billet. In this experimental investigation [77], the textures of interstitial-free (IF) steel and copper were analyzed after varying numbers of passes, as shown in Fig. 10b, c, and d. Although *xy*-shear strain occurs during the ECAE process, similar to the previous case, the reported pole figures were constructed based on the *Y* and *Z* axes instead of the *X* and *Y* axes. This shift alters the hotspot locations on the pole figures. However, texture simulation using `MicroProcSim` after

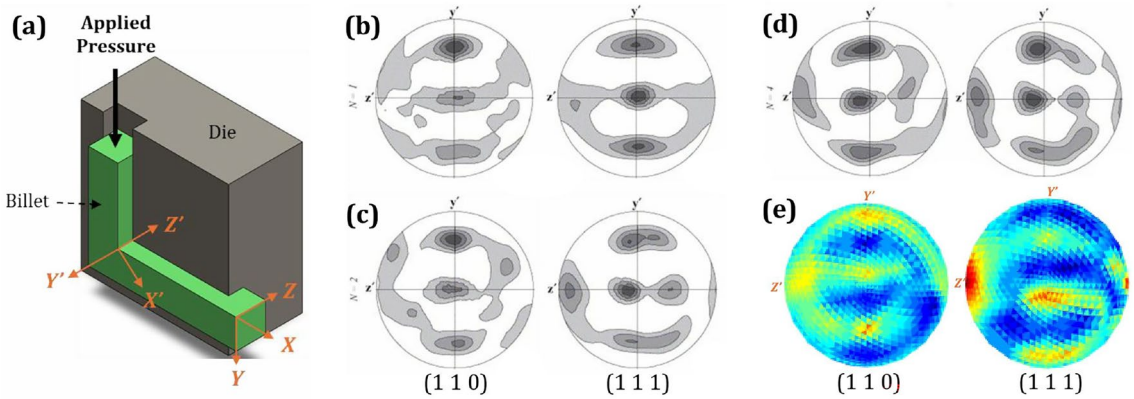

Fig. 10 Comparison of crystallographic textures after the shear process: **a** Schematic representation of a specimen undergoing the equal channel angular extrusion (ECAE) process; **b–d** textures rotated by 45° parallel to the shear plane of the 90° ECAE die intersection, representing simple shear for IF-steel on the (1 1 0) pole figure and copper on the (1 1 1) pole figure after **b** 1 pass, **c** 2 passes, and **d** 4 passes, as reported by Gazder et al. [77]; and **e** simulated texture after simple shear on pure copper using the `MicroProcSim` microstructure evolution software, starting with a randomly oriented initial microstructure and deformed to 0.1 mm/mm strain. ((**b**), (**c**), and (**d**) are reprinted from Ref. [77] with permission)

an *xy*-shear process with a shear strain of 0.1 mm/mm, as illustrated in Fig. 10e, closely matches the experimentally observed texture when the pole figure notation is consistent with them.

Another example texture [78] after the rolling process has been compared with our `MicroProcSim`-simulated plane strain compression texture. In the rolling process shown in Fig. 11a, plane strain compression occurs as a metal sheet is fed through two rotating rollers. The rollers exert compressive forces in the vertical (*Z*) direction, reducing the thickness of the sheet. Since the width of the sheet (*Y* direction) remains constant due to frictional and geometric constraints, deformation primarily occurs in the thickness (*Z*) and length (*X*) directions. This restriction creates a plane strain condition where strain in the *Y* direction is negligible, resulting in a two-dimensional deformation state. The material elongates in the *X* direction while being compressed in the *Z* direction, exemplifying plane strain compression. Tomé and Lebensohn [78] simulated an FCC aggregate texture by a rolling process that involved 500 orientations, which are shown in Fig. 11b and c. In our study, plane strain compression was applied to a randomly oriented texture in four stages, where each stage had 10% strain, resulting in a total strain of approximately 47% that matches the previously reported study. The texture simulated using `MicroProcSim`, as shown in Fig. 11d, reasonably matches the results of the earlier work. However, the deformed texture varies among different materials due to several fundamental factors: their distinct crystal structures, available slip systems, stacking fault energies, and characteristic deformation mechanisms [79].

Computational Costs

To demonstrate the computational cost of `MicroProcSim`, we have provided the execution times of the `app.exe` file under various loading conditions, as detailed in the illustrative examples shown in Figs. 3, 4, 5. These results are summarized in Table 2, which also includes the average utilization of memory, CPU, and GPU during the simulations. The computational analyses were conducted on a system equipped with 15.8 GB of RAM and an Intel(R) Core(TM) i7-10750 H CPU operating at a base frequency of 2.60 GHz. The system featured dual GPUs: an Intel(R) UHD Graphics processor for integrated graphics and an NVIDIA GeForce GTX 1650 Ti for high-performance tasks. For these simulations, only the Intel(R) UHD Graphics processor was utilized. Storage was provided by a PC611 NVMe SK hynix 512GB SSD, which operates on the NVMe (non-volatile memory express) protocol via a PCIe interface, enabling fast read and write operations. The machine ran on the Windows 11 Education operating system.

Table 2 lists execution times for the mentioned example simulations, which approximately range from half a minute to one minute for the given computer configurations. Eventually, these execution times mostly depend on the number of iterations required to solve the ODF, which in turn

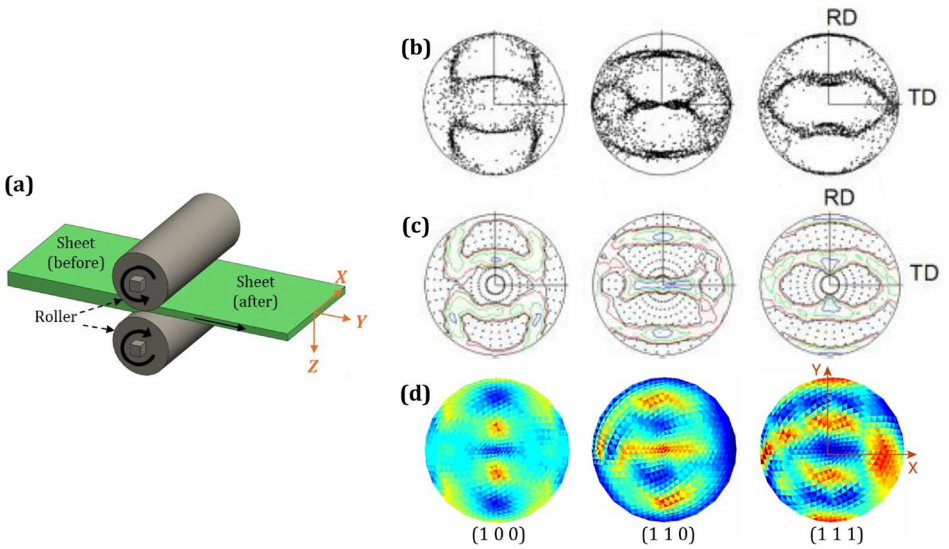

Fig. 11 Comparison of crystallographic textures in terms of (1 0 0), (1 1 0), and (1 1 1) pole figures after plane strain compression or a rolling process with 47% reduction: **a** Schematic representation of the specimen during the rolling process; **b–c** equal-area pole figures of simulated rolling of an FCC aggregate, showing (**b**) dots and (**c**) intensity lines, as reported by Tomé and Lebensohn [78]; **d** simulated texture of pure copper after plane strain compression using `MicroProcSim` microstructure evolution software, starting from an initially random orientation. ((**b**) and (**c**) are reprinted from Ref. [78] with permission.)

Table 2 Summary of computational costs for example simulations as illustrated in section "Illustrative Examples"

Deformation	Figs.	Execution time (s)	Average memory usage (MB)	Average CPU usage (%)	Average GPU usage (%)
x-tension	3 (b)	24.1836	13.8799	10.8636	2.2894
y-tension	3 (c)	45.0563	14.5887	21.1369	1.2874
z-tension	3 (d)	36.5485	14.2369	16.8455	1.4735
xy-shear	4 (b)	34.8552	14.0270	15.9062	2.2426
xz-shear	4 (c)	47.4494	14.0236	22.2812	2.1780
yz-shear	4 (d)	54.8004	14.0159	25.9956	1.6420
zy-rolling	5 (b)	30.1065	13.9054	13.5446	2.0764
yx-rolling	5 (c)	28.1823	14.0880	12.6850	2.4780
xz-rolling	5 (d)	38.9201	14.1903	18.0972	2.2065

depends on the initial textures of the microstructure, type of applied load, and strain rate. The execution times for these mentioned cases were also observed when running `app.exe` on a Linux OS using the `Wine` application. In this scenario, an AMD EPYC-7702 processor with a base frequency of 2 GHz was used. The execution times vary almost linearly; for example, using `Wine` results in approximately 62% longer execution time for the fastest case (x-tension) and up to 123% longer for the slowest case (yz-shear). On average, the additional Wine layer causes the execution time to nearly double compared to running natively on Windows.

When comparing `MicroProcSim` with recent microstructure texture evolution methods, notable differences in computational efficiency emerge. For instance, the viscoplastic self-consistent generalized material model (`VPSC-GMM`) [80] coupled with a Lagrangian hydrodynamics finite element code exhibits run times exceeding one minute in non-vectorized scenarios, though vectorization significantly enhances performance. These simulations addressed dynamic deformation conditions and incorporated the initial crystallographic texture of a tantalum cylinder. In contrast, `PRISMS-Plasticity TM` [74] demonstrates scalability advantages: weak-scaling tests for a polycrystalline copper sample with 400–102400 grains on 256 processors achieved a wall time of ~400 s, while strong-scaling analyses of a 400 grain sample under 100% compressive strain showed wall times of 8 s on 64 processors and 100–200 s when using fewer processors (e.g., 4 or 16). Notably, `MicroProcSim` demonstrates lower simulation costs compared to both existing crystal plasticity texture evolution software and other texture evolution modeling approaches.

Conclusion

The development of `MicroProcSim` marks a significant advancement in the simulation of metallic microstructures under deformation processes. `MicroProcSim` effectively predicts the evolution of microstructural textures in terms of ODFs under various loads and strain rates. This tool, originally designed for Windows and now extended to Linux, offers a robust solution for replicating the deformation behavior of cubic microstructures. It saves significant time and resources, which are otherwise typically spent on experimental observations. A MATLAB code is also included in the software package to automate the process for consecutive processing and save the desired output. In this study, sample results are reported for different loading conditions. In contrast to conventional crystal plasticity finite element software, `MicroProcSim` stands out by swiftly generating deformed textures without accounting for grain morphology, focusing solely on grain texture. Additionally, comparisons with experimental and computational studies on texture evolution confirm that the software effectively mimics real-world material processing conditions with just a simple adjustment to a single input matrix. This simulation tool will provide engineers and researchers with a reliable method for understanding and predicting the large deformation behavior of materials, with the potential to contribute to more informed decision-making and the development of more resilient materials. The future work on `MicroProcSim` will include the utilization of GPU resources to further improve its computational efficiency, development of a user-friendly graphical interface, as well as the extension of the microstructure formulation to different crystallographic systems (Table 3).

Table 3 Code metadata

Code metadata	Description
Current code version	*v1.0*
Permanent link to code/repository used for this code version	https://github.com/NU-CUCIS/MicroProcSim
Legal code license	GNU General Public License (GPL)
Software code languages, tools, and services used	C++, Matlab
Compilation requirements, operating environments, and dependencies	Windows Linux (with Wine application)
If available, link to developer documentation/manual	https://github.com/NU-CUCIS/MicroProcSim/blob/main/README.md
Support email for questions	pacar@vt.edu

Editor's Video Summary The online version of this article (https://doi.org/10.1007/s40192-025-00405-6) contains an Editor's Video Summary, which is available to authorized users.

Funding This research was supported by the National Science Foundation (NSF) CMMI Division Grant CMMI-2053840/2053929. Partial support from NIST award 70NANB19H005 and Northwestern Center for Nanocombinatorics is also acknowledged.

Data Availability The GitHub access link of `MicroProcSim` along with other code metadata can be found in the table above.

Declaration

Conflict of interest The authors declare that they have no conflicts of interest related to this work.

References

1. Li H, Chen T, Li W, Zhang H, Han S, Zhou C, Chen Z, Flores-Johnson EA, Shen L, Lian J et al (2021) Grain size dependent microstructure and texture evolution during dynamic deformation of nanocrystalline face-centered cubic materials. Acta Mater 216:117088
2. Fujii H, Cui L, Maeda M, Nogi K (2006) Effect of tool shape on mechanical properties and microstructure of friction stir welded aluminum alloys. Mater Sci Eng A 419(1–2):25–31
3. Mao Y, Hasan M, Paul A, Gupta V, Choudhary K, Tavazza F, Liao WK, Choudhary A, Acar P, Agrawal A (2023) An ai-driven microstructure optimization framework for elastic properties of titanium beyond cubic crystal systems. NPJ Comput Mater 9(1):111
4. Sundararaghavan V, Zabaras N (2008) A multi-length scale sensitivity analysis for the control of texture-dependent properties in deformation processing. Int J Plast 24(9):1581–1605
5. Agius D, Kourousis KI, Wallbrink C, Song T (2017) Cyclic plasticity and microstructure of as-built SLM Ti-6Al-4V: the effect of build orientation. Mater Sci Eng A 701:85–100
6. Kim DK, Kim JM, Park WW, Lee HW, Im YT, Lee YS (2015) Three-dimensional crystal plasticity finite element analysis of microstructure and texture evolution during channel die compression of if steel. Comput Mater Sci 100:52–60
7. Hirsch J, Lücke K (1988) Overview no. 76: Mechanism of deformation and development of rolling textures in polycrystalline fcc metals-i. description of rolling texture development in homogeneous cuzn alloys. Acta Metallurgica 36(11):2863–2882
8. Cho CH, Kim DO, Son K, Park HS (2023) Relationship between hot workability and texture evolution in an Al-Zn-Mg-Cu alloy under hot compressive stress mode. J Mater Sci 58(42):16537–16549
9. Sundararaghavan V, Zabaras N (2005) On the synergy between texture classification and deformation process sequence selection for the control of texture-dependent properties. Acta Mater 53(4):1015–1027
10. Lin J, Hasan M, Acar P, Blanchet J, Tarokh V (2023) Neural network accelerated process design of polycrystalline microstructures. Mater Today Commun 36:106884
11. Hasan M, Ender Eger Z, Senthilnathan A, Acar P (2024) Microstructure-sensitive deformation modeling and materials design with physics-informed neural networks. AIAA J 62(5):1864–1874
12. Kumar A, Dawson P (2000) Computational modeling of FCC deformation textures over Rodrigues' space. Acta Mater 48(10):2719–2736
13. Hingole RS (2016) Advances in metal forming. Springer, New York
14. Blakey-Milner B, Gradl P, Snedden G, Brooks M, Pitot J, Lopez E, Leary M, Berto F, Du Plessis A (2021) Metal additive manufacturing in aerospace: a review. Mater Des 209:110008
15. Taylor GI (1938) Plastic strain in metals. J Inst Metals 62:307–324
16. Van Houtte P (1982) On the equivalence of the relaxed Taylor theory and the bishop-hill theory for partially constrained plastic deformation of crystals. Mater Sci Eng 55(1):69–77
17. Kocks U, Chandra H (1982) Slip geometry in partially constrained deformation. Acta Metall 30(3):695–709
18. Van Houtte P, Delannay L, Samajdar I (1999) Quantitative prediction of cold rolling textures in low-carbon steel by means of the LAMEL model. Texture Stress Microstruct 31(3):109–149
19. Liu Y, Delannay L, Van Houtte P (2002) Application of the LAMEL model for simulating cold rolling texture in molybdenum sheet. Acta Mater 50(7):1849–1856

20. Van Houtte P, Kanjarla AK, Van Bael A, Seefeldt M, Delannay L (2006) Multiscale modelling of the plastic anisotropy and deformation texture of polycrystalline materials. Eur J Mech A/Solids 25(4):634–648
21. Crumbach M, Goerdeler M, Gottstein G, Neumann L, Aretz H, Kopp R (2003) Through-process texture modelling of aluminium alloys. Modell Simul Mater Sci Eng 12(1):1
22. Engler O, Crumbach M, Li S (2005) Alloy-dependent rolling texture simulation of aluminium alloys with a grain-interaction model. Acta Mater 53(8):2241–2257
23. Mu S, Al-Samman T, Mohles V, Gottstein G (2011) Cluster type grain interaction model including twinning for texture prediction: application to magnesium alloys. Acta Mater 59(18):6938–6948
24. Tjahjanto D, Eisenlohr P, Roters F (2009) A novel grain cluster-based homogenization scheme. Modell Simul Mater Sci Eng 18(1):015006
25. Eisenlohr P, Tjahjanto D, Hochrainer T, Roters F, Raabe D (2009) Texture prediction from a novel grain cluster-based homogenization scheme. Springer, New York
26. Lebensohn RA, Tomé C (1993) A self-consistent anisotropic approach for the simulation of plastic deformation and texture development of polycrystals: application to zirconium alloys. Acta Metall Mater 41(9):2611–2624
27. Tomé C, Lebensohn R (2007) Visco-plastic self-consistent (vpsc). Los Alamos National Laboratory (USA) and Universidad Nacional de Rosario (Argentina) vol 6
28. Turner P, Tomé C (1994) A study of residual stresses in zircaloy-2 with rod texture. Acta Metall Mater 42(12):4143–4153
29. Shao Y, Tang T, Li D, Tang W, Peng Y (2015) Crystal plasticity finite element modelling of the extrusion texture of a magnesium alloy. Modell Simul Mater Sci Eng 23(5):055011
30. Wang H, Wu P, Tomé C, Huang Y (2010) A finite strain elastic-viscoplastic self-consistent model for polycrystalline materials. J Mech Phys Solids 58(4):594–612
31. Kronsteiner J, Theil E, Ott AC, Arnoldt AR, Papenberg NP (2024) Modeling of texture development during metal forming using finite element visco-plastic self-consistent model. Curr Comput-Aided Drug Des 14(6):533
32. Eghtesad A, Germaschewski K, Knezevic M (2022) Coupling of a multi-GPU accelerated elasto-visco-plastic fast Fourier transform constitutive model with the implicit finite element method. Comput Mater Sci 208:111348
33. Adams BL, Kalidindi SR, Fullwood DT (2012) Microstructure sensitive design for performance optimization. Butterworth-Heinemann
34. Kumar A, Dawson PR (2009) Dynamics of texture evolution in face-centered cubic polycrystals. J Mech Phys Solids 57(3):422–445
35. Sundararaghavan V (2007) Multi-scale computational techniques for design of polycrystalline materials
36. Acar P (2020) Machine learning reinforced crystal plasticity modeling under experimental uncertainty. AIAA J 58(8):3569–3576
37. Acar P, Sundararaghavan V (2016) Linear solution scheme for microstructure design with process constraints. AIAA J 54(12):4022–4031
38. Acar P, Sundararaghavan V, De Graef M (2018) Computational modeling of crystallographic texture evolution over cubochoric space. Modell Simul Mater Sci Eng 26(6):065012
39. Kalidindi SR, Knezevic M, Niezgoda S, Shaffer J (2009) Representation of the orientation distribution function and computation of first-order elastic properties closures using discrete fourier transforms. Acta Mater 57(13):3916–3923
40. Suwas S, Ray RK, Suwas S, Ray RK (2014) Representation of texture. Crystallographic texture of materials, 11–38.
41. Inoue H (2015) Prediction of in-plane anisotropy of bendability based on orientation distribution function for polycrystalline face-centered cubic metal sheets with various textures. Mater Trans 56(1):61–69
42. Sundararaghavan V, Zabaras N (2004) A dynamic material library for the representation of single-phase polyhedral microstructures. Acta Mater 52(14):4111–4119
43. Acar P, Sundararaghavan V (2016) Utilization of a linear solver for multiscale design and optimization of microstructures in an airframe panel buckling problem. In: 57th AIAA/ASCE/AHS/ASC structures, structural dynamics, and materials conference, p 0156
44. Acar P, Srivastava S, Sundararaghavan V (2017) Stochastic design optimization of microstructures with utilization of a linear solver. AIAA J 55(9):3161–3168
45. Acar P, Ramazani A, Sundararaghavan V (2017) Crystal plasticity modeling and experimental validation with an orientation distribution function for Ti-7Al alloy. Metals 7(11):459
46. Cho JH (2007) Determination of volume fraction of the texture components in the Rodrigues fundamental region. Mater Sci Eng A 465(1–2):228–237
47. Raabe D, Sachtleber M, Zhao Z, Roters F, Zaefferer S (2001) Micromechanical and macromechanical effects in grain scale polycrystal plasticity experimentation and simulation. Acta Mater 49(17):3433–3441
48. Raabe D, Klose P, Engl B, Imlau KP, Friedel F, Roters F (2002) Concepts for integrating plastic anisotropy into metal forming simulations. Adv Eng Mater 4(4):169–180
49. Zhao Z, Ramesh M, Raabe D, Cuitino A, Radovitzky R (2008) Investigation of three-dimensional aspects of grain-scale plastic surface deformation of an aluminum oligocrystal. Int J Plast 24(12):2278–2297
50. Roters F, Eisenlohr P, Hantcherli L, Tjahjanto DD, Bieler TR, Raabe D (2010) Overview of constitutive laws, kinematics, homogenization and multiscale methods in crystal plasticity finite-element modeling: theory, experiments, applications. Acta Mater 58(4):1152–1211
51. Acar P, Sundararaghavan V (2017) Uncertainty quantification of microstructural properties due to variability in measured pole figures. Acta Mater 124:100–108
52. Billah MM, Acar P (2024) Design of polycrystalline metallic alloys under multi-scale uncertainty by connecting atomistic to meso-scale properties. Acta Mater 270:119879
53. Acar P (2019) Machine learning approach for identification of microstructure-process linkages. AIAA J 57(8):3608–3614
54. Wang K, Mao Y, Hasan M, Billah MM, Kilic MNT, Gupta V, Liao WK, Choudhary A, Acar P, Agrawal A (2014) Deep learning based inverse modeling for materials design: From microstructure and property to processing. In: 2024 international conference on machine learning and applications (ICMLA), pp 236–241. IEEE
55. Billah MM, Acar P Process-informed property evolutions of microstructures. Available at SSRN 5198175
56. Frank C (1988) Orientation mapping: 1987 MRS fall meeting von hippel award lecture. MRS Bull 13(3):24–31
57. Neumann P (1991) Representation of orientations of symmetrical objects by Rodrigues vectors. Texture Stress Microstruct 14(1):53–58
58. Neumann P (1991) Graphical representations of orientations and ODFS by Rodrigues vectors. Steel research 62(12):560–566
59. Schwartz A (2009) Electron backscatter diffraction in materials science. Springer, New York
60. Sundararaghavan V, Zabaras N (2007) Linear analysis of texture-property relationships using process-based representations of Rodrigues space. Acta Mater 55(5):1573–1587
61. Acar P (2017) Multi-scale design and optimization of microstructures under uncertainties. Ph.D. dissertation, University of Michigan. https://deepblue.lib.umich.edu/handle/2027.42/138534

62. Billah MM, Acar P (2025) Design of functionally graded inconel 718 alloy structures by developing material property closures. In: AIAA SCITECH 2025 Forum, p 1746
63. Ganapathysubramanian S, Zabaras N (2004) Design across length scales: a reduced-order model of polycrystal plasticity for the control of microstructure-sensitive material properties. Comput Methods Appl Mech Eng 193(45–47):5017–5034
64. Hansen PB (1973) Operating system principles. Prentice-Hall, Inc
65. Tanenbaum A (2009) Modern operating systems. Pearson Education, Inc
66. Yuan P, Guo Y, Zhang L, Chen X, Mei H (2018) Building application-specific operating systems: a profile-guided approach. Sci China Inf Sci 61:1–17
67. Fink J (2014) Docker: a software as a service, operating system-level virtualization framework. Code4Lib J (25)
68. Microsoft: Windows API Index (2023) https://learn.microsoft.com/en-us/windows/win32/apiindex/windows-api-list. Accessed: 2024-06-19
69. Koopman P, DeVale J (1999) Comparing the robustness of posix operating systems. In: Digest of papers. Twenty-ninth annual international symposium on fault-tolerant computing (Cat. No. 99CB36352), pp 30–37. IEEE
70. WineHQ: About Wine (2023). https://www.winehq.org/about. Accessed: 2024-06-19
71. Barton NR, Boyce DE, Dawson PR (2002) Pole figure inversion using finite elements over Rodrigues space. Textures Microstruct 35(2):113–144
72. Bachmann F, Hielscher R, Schaeben H (2010) Texture analysis with MTEX-free and open source software toolbox. Solid State Phenom 160:63–68
73. Bronkhorst CA, Kalidindi SR (1992) Anand L (1992) Polycrystalline plasticity and the evolution of crystallographic texture in FCC metals. Philos Trans R Soc London. Series A Phys Eng Sci 341(1662):443–477
74. Yaghoobi M, Allison JE, Sundararaghavan V (2022) Prisms-plasticity TM: an open-source rapid texture evolution analysis pipeline. Integr Mater Manuf Innov 11(4):479–496
75. Duan J, Wen H, Zhou C, Islamgaliev R, Li X (2019) Evolution of microstructure and texture during annealing in a high-pressure torsion processed Fe-9cr alloy. Materialia 6:100349
76. Griffiths RJ, Garcia D, Song J, Vasudevan VK, Steiner MA, Cai W, Hang ZY (2021) Solid-state additive manufacturing of aluminum and copper using additive friction stir deposition: process-microstructure linkages. Materialia 15:100967
77. Gazder AA, Dalla Torre F, Gu C, Davies CH, Pereloma EV (2006) Microstructure and texture evolution of BCC and FCC metals subjected to equal channel angular extrusion. Mater Sci Eng A 415(1–2):126–139
78. Tome CN, Lebensohn RA (2023) Material modeling with the visco-plastic self-consistent (VPSC) approach: theory and practical applications. Elsevier
79. Kestens L, Pirgazi H (2016) Texture formation in metal alloys with cubic crystal structures. SAGE Publications, London
80. Zecevic M, Lebensohn RA, Rogers M, Moore J, Chiravalle V, Lieberman E, Dunning D, Shipman G, Knezevic M, Morgan N (2021) Viscoplastic self-consistent formulation as generalized material model for solid mechanics applications. Appl Eng Sci 6:100040