The Neper/FEPX Project: Free / Open-source Polycrystal Generation, Deformation Simulation, and Post-processing

Romain Quey, Matthew Kasemer

The Neper/FEPX Project: Free / Open-source Polycrystal Generation, Deformation Simulation, and Post-processing

Romain Quey [1] **and Matthew Kasemer** [2]

[1] Mines Saint-Étienne, Univ Lyon, CNRS, UMR 5307 LGF, F – 42023 Saint-Etienne, France
[2] Department of Mechanical Engineering, University of Alabama, Tuscaloosa, AL 35487 United States

E-mail: romain.quey@mines-stetienne.fr

Abstract. In the past couple of years, efforts have lead toward the convergence of two established software packages: Neper, developed primarily at CNRS and Mines Saint-Étienne, and FEPX, developed historically at the Deformation Process laboratory at Cornell University and (since 2020) at the Advanced Computational Materials Engineering Laboratory at the University of Alabama. The objective was to create an extensive and homogeneous ensemble for polycrystal plasticity studies that includes polycrystal generation and meshing, parallel finite-element crystal-plasticity simulation, post-processing and visualization. The two programs were upgraded to common standards, both in terms of usage and resources. Collectively, the project is under active development and distributed under free / open-source license (see both: `https://neper.info`, `https://fepx.info`).

1. Introduction

The Neper project was started in 2003 and first released as free / open-source software in July 2009, and has been developed by Romain Quey, primarily at CNRS and Mines Saint-Étienne. The original intent of the effort was to generate random polycrystalline microstructures, as Voronoi tessellations, and to mesh them for further use in finite-element crystal-plasticity simulations [1], including at large strains [2]. The proposed set of methods can be efficiently applied to virtually any "CAD-type" polycrystal models (i.e. defined in a vectorial way, using vertices, edges, (planar) faces and polyhedra). Further efforts were made on the representation of more complex microstructures, such as those found in steels or titanium alloys [3], and on the generation of polycrystals of specified grain properties [4], be it from experiments, e.g. to replicate typical grain size and shape statistics, or from fundamental arguments, e.g. to isolate the influence of specific factors, in purely numerical studies [3]. For many years, Neper's development was mainly application-driven, but the program was sufficiently general to be of interest to others (see an exhaustive list of articles at `https://neper.info/applications.html`).

The FEPX project was started in the late 1990's in Paul Dawson's group, the Deformation Processes Laboratory (DPLab) at Cornell University, and since 2020 has been maintained and developed in Matthew Kasemer's group, the Advanced Computational Materials Engineering Laboratory (ACME Lab) at the University of Alabama. The original intent of the effort was to develop a code to support the simulation of polycrystalline deformation by explicitly

incorporating both the anisotropic elastic and plastic response. Early iterations of the framework focused on development for simulation of sheet forming operations [5] and texture evolution [6]. Eventually, the framework included the formulation of polycrystal models considering local homogenized response [7, 8] and robust parallelization for large-scale computation [9]. Further development saw the application of the framework to full-field simulations (for which grains are typically discretized into tens to thousands of elements) using regular-shaped grains [10], and finally using generalized, random-shaped grains via Voronoi tessellations in the first convergence with the Neper project [2].

After a decade of scientific collaboration and cross-usage of the two programs between the two research groups, and in a continuing effort to make FEPX publicly available, it was decided in April 2020 to make Neper and FEPX two "companion programs", to form an extensive and homogeneous ensemble dedicated to the analysis of polycrystal deformation, which lead to first common releases in July 2020. The original goal was to ease the concomitant use of the two programs, via the design of a shared file format, and to upgrade the programs to common standards in terms of usage (user interface, program inputs and outputs, etc.) and resources (website, documentation, source code management, user support, etc.). Since then, the Neper/FEPX project has continued to develop in all its aspects, and has started to integrate capabilities of other, "satellite" projects developed by the two groups and specifically dedicated to result post-processing, such as Orilib and Hermes (CNRS / Mines Saint-Étienne), two general programs for (EBSD-type) data analysis, and ODFPF (DPLab / ACME Lab), a program for orientation distribution function and pole figure related computations. The following provides an overview of the Neper/FEPX project, with a focus on recent advances.

2. Description

Neper and FEPX are programs designed to be used in succession. Neper itself is made of several modules ("-T" for tessellation, "-M" for meshing, "-S" for simulation and post-processing, and "-V" for visualization). Neper is written in the C language and multithreaded using OpenMP, and is typically used on a personal computer or workstation (i.e., a single computational node, $\mathcal{O}(10)$ of processing units). It uses several external libraries, for various tasks ranging for distance computation to meshing and image generation [11, 12, 13, 14, 15, 16, 17, 18, 19]. FEPX is written in the Fortran language and parallelized using Open MPI, and is typically used on computer clusters (i.e., multiple computational nodes, $\mathcal{O}(10-1000)$ of processing units). A typical Neper/FEPX workflow is illustrated in figure 1 and detailed in the following.

2.1. Tessellation Generation (Neper -T)

Polycrystals are most often represented as *tessellations* defined under a vectorial format, i.e., using vertices, edges, faces and polyhedra (or "cells"), and cells are generally convex. Such a "CAD-type" geometrical representation allows for the modelling of a wide variety of polycrystalline microstructures while still being prone to meshing by standard methods. The tessellations include *single-scale tessellations*, which are able to represent simple polycrystals, and *multi-scale tessellations*, which are more adapted to represent polycrystalline microstructures for which subdivisions (i.e., intra-cell or intra-grain features) associated with phase transformations occur during the material processing, as is the case in most steels, titanium alloys, etc.

2.1.1. Single-scale Tessellations. Any convex-cell tessellation (or, more specifically, *normal* tessellation [20]) can be generated, independently of the sizes, shapes and arrangements of its cells. This is made possible using Laguerre tessellations, which can intrinsically generate any such tessellations. The cells of a Laguerre tessellation, C_i, are defined from a set of seeds, S_i, of

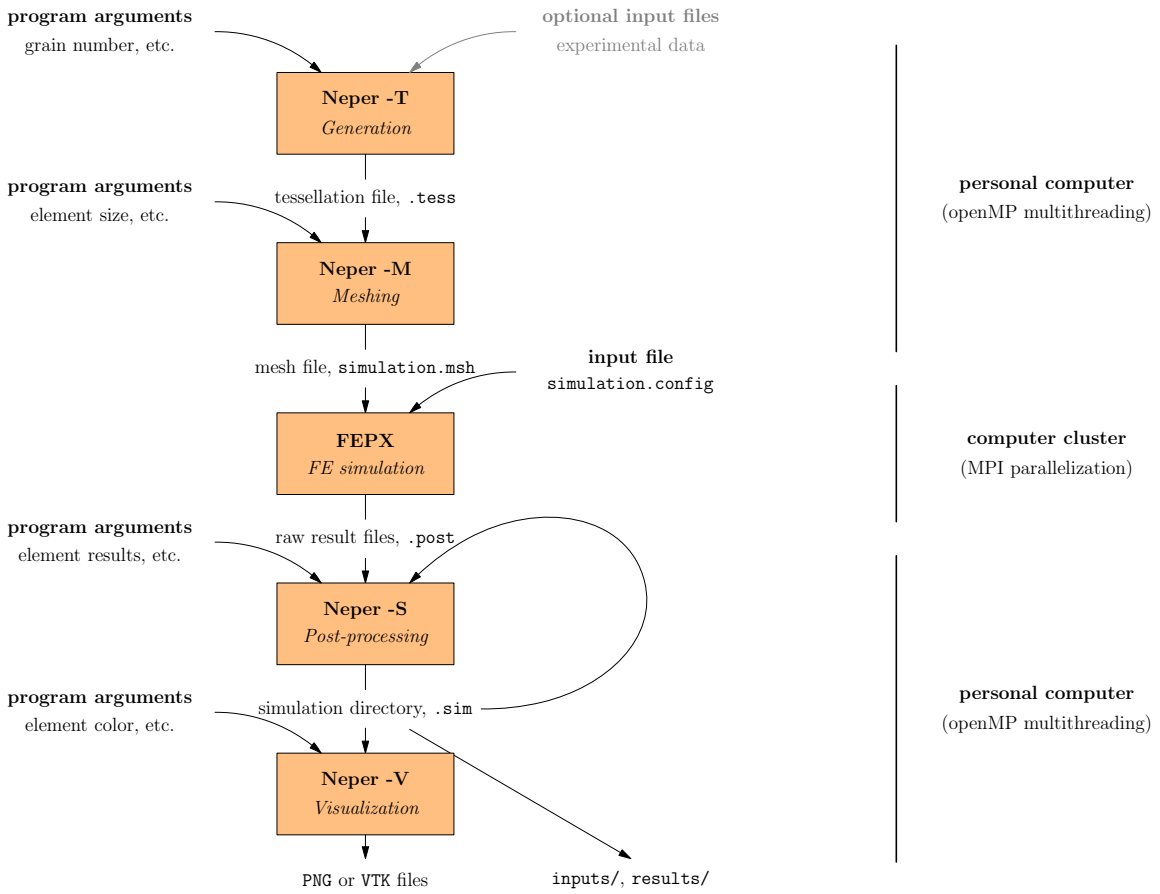

Figure 1. A typical Neper/FEPX workflow, including the successive steps, and input and output files. The different Neper modules and FEPX are run in succession, take inputs as arguments (for Neper) or external files (optionally for Neper, exclusively for FEPX), and exchange result files and directories.

given positions, $\boldsymbol{x_i}$, and weights, w_i:

$$C_i = \left\{ P(\boldsymbol{x}) \in D \mid d(P,\, S_i)^2 - w_i < d(P,\, S_j)^2 - w_j \;\; \forall j \neq i \right\}, \tag{1}$$

where $d(\bullet,\, \bullet)$ is the Euclidean distance. The seed positions and weights are determined by a constraint-free optimization. This is different from more standard approaches, for which seed attributes are pre-determined, e.g. from a grain size distribution and a dense sphere packing algorithm or from individual grain properties [4]. Optimization is carried out so as to obtain specified cell properties (grain size or shape distributions, morphological texture, etc.), or grain positions, sizes or shapes, or even a rasterized image of the polycrystal, as may be provided by electron backscatter diffraction or synchrotron X-ray diffraction experiments [21, 4]. Examples of single-scale tessellations are provided in figure 2a-c.

2.1.2. Multi-scale Tessellations. Several scales are obtained by generating tessellations in the individual cells of the previous, upper-scale tessellation(s), mimicking the actual grain subdivisions occurring during material processing. Arbitrarily complex microstructures can be generated, where the cell properties of the tessellation are specified at each scale, as in the case of a single-scale tessellation. For instance, subcells can be used to represent bainitic packets

in steel or, equivalently, colonies in titanium alloys. Lamellar structures can also be generated. Different phases can be applied to the different cell subdivisions. Orientation relationships between successive scales can also be taken into account [3]. An example of a multi-scale tessellation is provided in figure 2d.

2.2. Meshing (Neper -M)

Tessellations can be discretized via unstructured meshing into tetrahedral elements, which provides a uniform element size distribution and maximizes the element shape quality factors. Meshing is usually preceded by a tessellation regularization procedure that removes the smallest features of the tessellation [2], which is essential to obtain good-quality meshes (both in terms of element size and shape factor), facilitates the numerical convergence of the simulation and allows for large strain simulations (up to about 40%). To reach larger strains, re-meshing can be applied, by which a new mesh is constructed over a deformed mesh and the state variables are mapped to the new mesh [2]. Remeshing does not significantly affect the simulation results and, using all methods combined, polycrystals have been deformed to strains of the order of 1.2–1.4 for aluminium in plane strain compression [2, 22, 23], and larger strains can be attained. Example polycrystal meshes are provided in figure 2e,f.

2.3. Crystal-Plasticity Simulation (FEPX)

Broadly, a polycrystal is defined via the mesh and crystal orientations (defined at elements), while the crystal behavior is defined via its elastic response (Hooke's law) and plastic response (slip systems, slip kinetics, and hardening), and the associated modeling parameters. Loading is applied as a function of time by specifying velocities or forces as boundary conditions.

2.3.1. Crystal Response. A truncated description of the base models employed in FEPX is given below (see `https://fepx.info` for a description of all available models). The stress, σ, is related to the elastic strain, ε, via Hooke's law:

$$\sigma = \mathcal{C}\varepsilon, \tag{2}$$

where $\mathcal{C}$ is the stiffness tensor. Cubic, hexagonal and tetragonal symmetries can be considered.

The kinematics of slip are described by a power law:

$$\dot{\gamma}^{\alpha} = \dot{\gamma}_0 \left(\frac{|\tau^{\alpha}|}{g^{\alpha}} \right)^{1/m} \operatorname{sgn}(\tau^{\alpha}), \tag{3}$$

where $\dot{\gamma}_0$ is the fixed-rate strain rate scaling coefficient, τ^{α} is the resolved shear stress on a slip system, α, g^{α} is the current critical resolved shear stress, and m is the rate sensitivity exponent.

For an isotropic hardening assumption, slip system strength evolution (hardening) is modeled by:

$$\dot{g^{\alpha}} = h_0 \left(\frac{g_{s0} - g^{\alpha}}{g_{s0} - g_0} \right)^{n} \dot{\gamma}, \tag{4}$$

where h_0 is the fixed-state hardening rate scaling coefficient, g_{s0} is the initial slip system saturation strength, g_0 is the initial slip system strength, and n is the non-linear Voce hardening exponent. In the above equation, $\dot{\gamma}$ is calculated as

$$\dot{\gamma} = \sum_{\alpha} |\dot{\gamma}^{\alpha}| \,. \tag{5}$$

FEPX has the ability to handle other crystal behaviors, such as cyclic hardening, saturation strength evolution, latent hardening, etc., further described in FEPX's documentation.

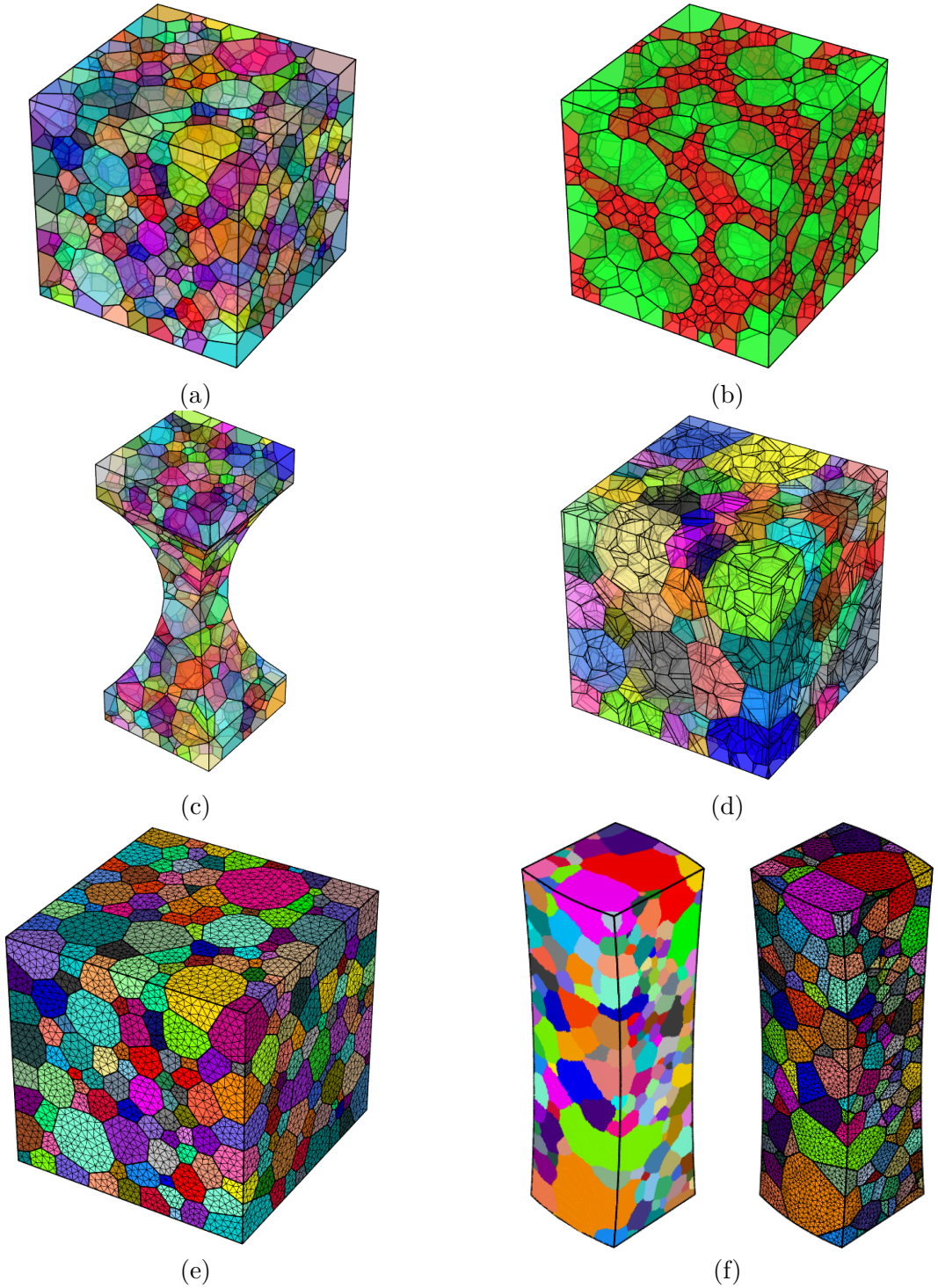

Figure 2. Typical microstructures and meshes generated by Neper. (a) Typical grain-growth polycrystal, (b) 2-phase polycrystal, (d) non-convex polycrystalline specimen, (d) 3-scale polycrystal representative of a (bainitic) steel microstructure, (e) meshed grain-growth polycrystal, and (f) meshed DCT polycrystal.

A complete description of the base constitutive model and the finite element method implementation can be found in Refs. [7, 8] and are also documented in a theory manual, which describes FEPX's pre-2020 state [24].

2.3.2. Deformation History. A simulation is defined as a set of steps through time, in which (generally) either a user-specified strain or load state is targeted at the end of each time step. FEPX considers either uniaxial deformation conditions or principal-triaxial deformation conditions. Simulations may also be performed considering either constant loading rates or constant strain rates.

2.3.3. Boundary Conditions. Boundary conditions must be applied to control the deformation response. FEPX allows for generalized boundary conditions, though comes pre-loaded with a variety of conditions suited to the base deformation histories (uniaxial and principal-triaxial) described above. Generally, boundary conditions are defined at a set of nodes and are specified as the prescribed velocities in three orthogonal directions at each node (aligned with the sample coordinate system).

2.3.4. Running Conditions. Simulation runtime is dependent on both the size and difficulty of the simulation, and the size of computer/cluster available to perform the simulations.

Concerning the former, many aspects will influence simulation runtime, including the difficulty/non-linearity of the problem, the computational expense of the models employed, and other factors. However, simulation runtime will be primarily affected by the size of the mesh and the number of load steps. Generally speaking, the larger the mesh (i.e., the larger the number of elements/nodes), or the higher number of load steps, the longer the simulation will take to complete on a given number of computational units.

Concerning the latter, simulations are typically performed in parallel on multiple processing cores—either on typical computers with multi-core processors ($\mathcal{O}(10)$ processing units), computational workstations with multiple multi-core processors ($\mathcal{O}(10-100)$ processing units), or computational clusters with potentially many computational nodes each with multiple multi-core processors ($\mathcal{O}(10-1000)$ total processing units).

Approximate runtimes for a simulation with 10 equally-spaced load steps to 5% strain utilizing isotropic hardening, utilizing a computational workstation with 24 processing units are summarized in Table 1. Note that these values do not imply scaling behavior, nor are they meant to demonstrate optimal running conditions, but instead are meant to give a rough estimate of how long simulations of a certain size will take to complete given a number of processing units, and one would expect more efficient computation of larger simulations performed on more processing units.

Number of Nodes	Approximate Runtime (min)
10,000	1
30,000	5
100,000	60
400,000	300

Table 1. Estimates of representative simulation runtimes

2.4. Post-processing (Neper -S)

Post-processing involves the archiving of a simulation (inputs and results) and further processing of the results. Typically, this is done using a custom file and directory structure, and custom scripts and program written in a general language. Neper aims to simplify and standardize this process via the definition of a *simulation directory* and the ability to modify or complete it using simple procedures, with a focus on operations specific to polycrystal computations.

A *simulation directory* has a simple and adaptable hierarchical file and directory structure, as shown in figure 3, which includes the simulation inputs and results, at successive deformation steps. A *simulation directory* allows for post-processing operations such as direct computation of mesh properties (grain centers and volumes, etc.), new simulations results from the mesh or existing simulation results (standard computations such as that of the equivalent stress from the full stress tensor, or arbitrary computations via generalized mathematical expressions), grain-averaged or sample-averaged results, results over pre-defined sets or elements, etc. The new results are in turn included into the *simulation directory*. Practically, the new results become attributes of the entities to which they relate, which allows for the reverse approach consisting of defining mesh regions from the simulation results, as is useful for fine analyses or to identify microstructure–property relationships.

A *simulation directory* can also organize experimental data as input (optionally results), such as 2D or 3D orientation maps provided by EBSD or synchrotron X-ray diffraction techniques. This makes it possible to use the same tools and procedures to process data obtained by different methods, which is particularly interesting in the context of experiment-simulation comparisons [25].

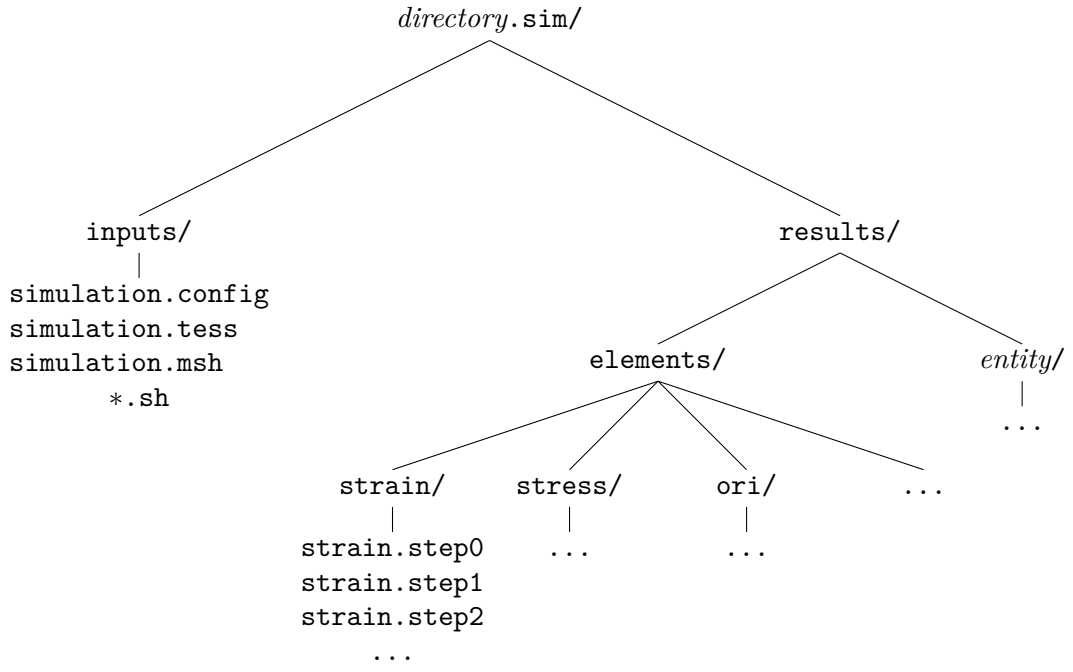

Figure 3. Typical structure of a *simulation directory*. A *simulation directory* contains two self-defined subdirectories, `inputs` and `results`. The `results` directory itself is divided into one directory for each entity, which can be either the built-in `elements`, `elsets` (for grains), `nodes` and `mesh` (for the full sample), or custom entities, and each entity directory is itself divided into result directories, themselves containing the actual data files.

2.5. Visualization (Neper -V)

Visualization mainly operates in real, physical space, to show a mesh and its simulation results, but can also be done as pole figures or in orientation space, to analyse the results. Images are generated non-interactively, at the PNG or PDF formats, or at the VTK format for interactive visualization using, e.g., Paraview. 3D scenes (in real space and orientation space) are rendered as rasterized images, by ray-tracing [18], while 2D scenes (in pole figure space) are rendered as vectorial images [19]. Example visualizations are provided in figure 4. Results can be visualized at successive simulation steps or at different view angles, and animations can be created accordingly, as MP4 files. In real space, a displacement field can typically be applied at nodes, and any result field (stress, strain, orientation, etc.) can be visualized at elements, as shown in figure 4a. Different data sets (such as those obtained at successive steps of a simulation) can be superimposed onto the same image to produce advanced plots, as shown for a pole figure in figure 4b. Finally, an example of plot in Rodrigues orientation space is provided in figure 4c.

3. Conclusions

The Neper/FEPX project aims to built an extensive and homogeneous ensemble for polycrystal plasticity studies. Both Neper and FEPX are products of over 40 collective years of active development and use, for various applications. Recent efforts have upgraded both programs to common standards, which lead to simplified usage, increased robustness and improved resources (websites, documentations, source code management, user support, etc.). The two programs can be used efficiently to simulate polycrystal deformation in a fully-automated way. Current efforts are focused on the improvement and extension of post-processing, mostly by integrating otherwise-existing capabilities, and on strengthening further synergies between the two programs.

References

[1] Barbe F, Quey R, Musienko A and Cailletaud G 2009 *Mech. Res. Com.* **36** 762–768
[2] Quey R, Dawson P and Barbe F 2011 *Comput. Methods Appl. Mech. Eng.* **200** 1729–45
[3] Kasemer M, Quey R and Dawson P 2017 *J. Mech. Phys. Solids* **103** 179–98
[4] Quey R and Renversade L 2018 *Comp. Methods Appl. Mech. Eng.* **330** 308–33
[5] Mathur K 1995 *Simulation of Materials Processing: Theory, Methods and Applications - NUMIFORM 95*
[6] Dawson P and Beaudoin A 1997 *JOM* **49** 34–41
[7] Marin E and Dawson P 1998 *Comput. Methods Appl. Mech. Eng.* **165** 1–21
[8] Marin E and Dawson P 1998 *Comput. Methods Appl. Mech. Eng.* **165** 23–41
[9] Mika D and Dawson P 1999 *Acta Mater.* **47** 1355–69
[10] Ritz H and Dawson P R 2009 *Model. Simul. Mater. Sci. Eng.* **17** 1–21
[11] GNU Scientific Library URL `http://www.gnu.org/software/gsl/`
[12] muparser - fast math parser library URL `https://github.com/beltoforion/muparser`
[13] nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees URL `https://github.com/jlblancoc/nanoflann`
[14] Montanari M and Petrinic N 2018 *SoftwareX* **7** 352–5
[15] Geuzaine C and Remacle J F 2009 *International Journal for Numerical Methods in Engineering* **79** 1309–1331
[16] The NLopt nonlinear-optimization package URL `http://ab-initio.mit.edu/wiki/index.php/NLopt`
[17] Scotch URL `https://www.labri.fr/perso/pelegrin/scotch`
[18] *http://www.povray.org*
[19] Asymptote: the Vector Graphics Language URL `http://asymptote.sourceforge.net`
[20] Lautensack C 2008 *J. Appl. Stat.* **35** 985–995
[21] Kasemer M, Echlin M, Stinville J, Pollock T and Dawson P 2017 *Acta Mater.* **136** 288–302
[22] Quey R, Driver J and Dawson P 2015 *J. Mech. Phys. Solids* **84** 506–27
[23] Quey R, Dawson P and Driver J 2012 *J. Mech. Phys. Solids* **60** 509–24
[24] Dawson P and Boyce D 2015 *ArXiv* arXiv:1504.03296 [cond–mat.mtrl–sci]
[25] Renversade L and Quey R 2019 *9 IOP Conf. Ser.: Mater. Sci. Eng.* **580** 012022
[26] Li R, Quey R, Zhang Y, Kobayashi M, Oddershede J and Juul Jensen D to appear *IOP Conf. Ser.: Mater. Sci. Eng.*

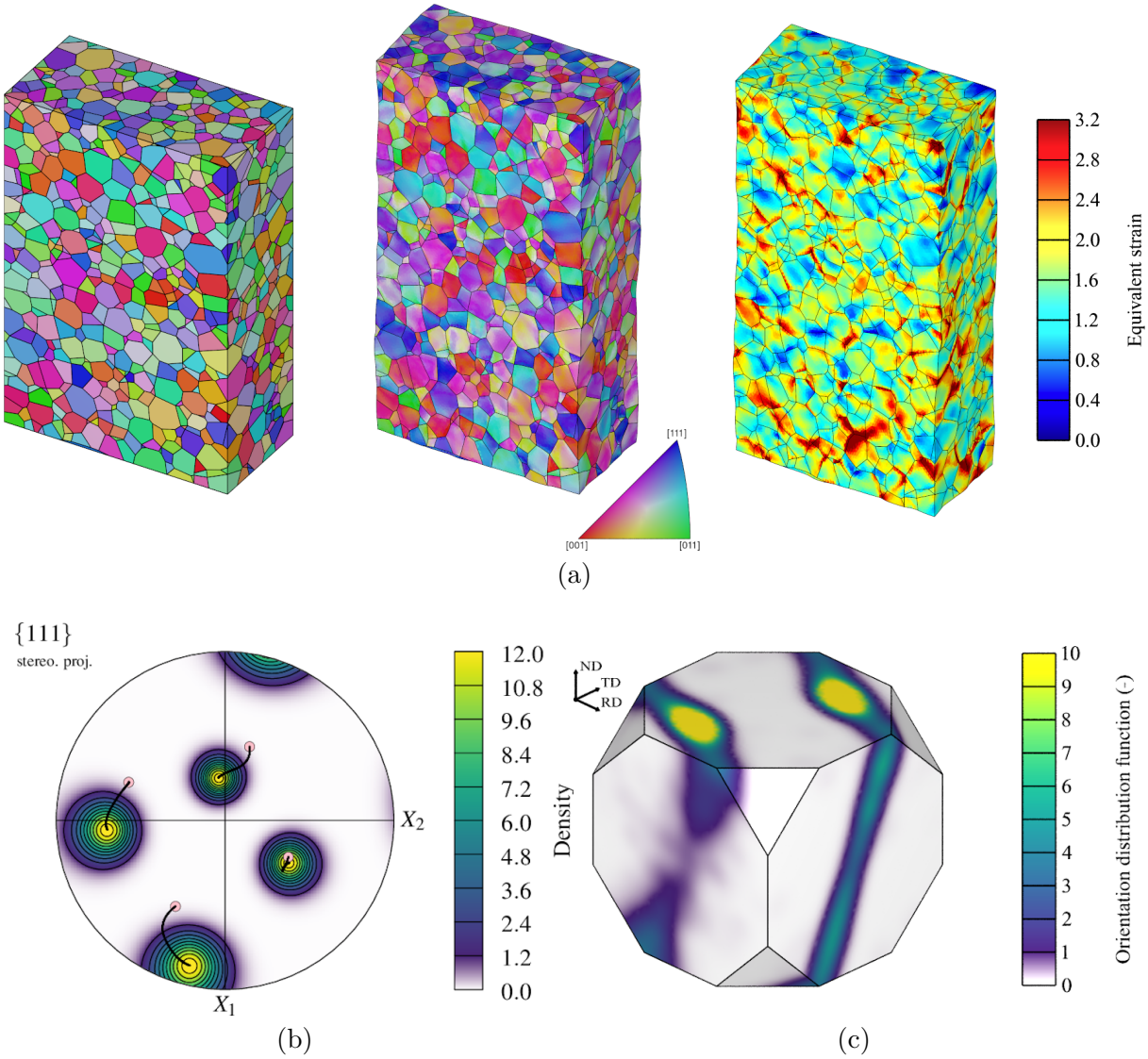

Figure 4. Typical simulation results obtained using Neper and FEPX. (a) Al–4%Cu polycrystal deformed to 16% in uniaxial tension (left: undeformed polycrystal, middle: deformed polycrystal colored by orientation, right: deformed polycrystal colored by equivalent strain) [26], (b) grain lattice rotation on a pole figure, (c) rolling orientation distribution function over Rodrigues orientation space.