ON THE DEVELOPMENT OF AN OPEN-SORUCE PREPROCESSING FRAMEWORK FOR FINITE ELEMENT SIMULATIONS

by

Alexandra Danielle Arellano Mallory

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science in Aeronautics and Astronautics

School of Aeronautics & Astronautics

West Lafayette, Indiana

May 2019

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Michael Sangid, Chair

Department of Aeronautics and Astronautics

Dr. Weinong Chen

Department of Aeronautics and Astronautics

Dr. Thomas Siegmund

Department of Mechanical Engineering

Approved by:

Dr. Weinong Chen

Head of the Graduate Program

Ad astra per aspera.

ACKNOWLEDGMENTS

First and foremost, thank you to Dr. Sangid for his guidance and support. Your support has meant so much to getting through the program.

Additional thank you to my committee members, Dr. Weinong Chen and Dr. Thomas Siegmund for serving on my advisor committee. Support from the Indiana Consortium for Simulation-Based Engineering of Materials and Structures (ICSEMS) for funding this project is appreciated. In particular, thank you to Dr. Siegmund for his efforts in leading ICSEMS.

Thank you as well to ACME^2 labs for all of the help and support over the last two years. A special thank you to Andrea Nicolas, Michael Waddell, and Ritwik Bandyopadhyay for help with coding and analyzing the results.

Finally, thank you my parents for supporting me in every way they could. Thank you for raising me to be the person I am. Even though you are far away, the support you have me was absolutely immeasurable and means everything to me.

The support of everyone – Dr. Sangid, my committee, my labmates, and all the friends I have made – means everything to me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Author: Mallory, Alexandra D. MSAAE
Institution: Purdue University
Degree Received: May 2019
Title: On the Development of an open-source preprocessing framework for finite element simulations
Committee Chair: Michael Sangid

Computational modeling is essential for material and structural analyses for a multitude of reasons, including for the improvement of design and reducing manufacturing costs. However, the cost of commercial finite element packages prevent companies with limited financial resources from accessing them. Free finite element solvers, such as Warp3D, exist as robust alternatives to commercial finite element analysis (FEA) packages. This and other open-source finite element solvers are not necessarily easy to use. This is mainly due to a lack of a preprocessing framework, where users can generate meshes, apply boundary conditions and forces, or define materials. We developed a preprocessor for Warp3d, which is referred to as *W3DInput*, to generate input files for the processor. *W3DInput* creates a general framework, at no cost, to go from CAD models to structural analysis. With this preprocessor, the user can import a mesh from a mesh generator software – for this project, Gmsh was utilized – and the preprocessor will step the user through the necessary inputs for a Warp3D file. By using this preprocessor, the input file is guaranteed to be in the correct order and format that is readable by the solver, and makes it more accessible for users of all levels. With this preprocessor, five use cases were created: a cantilever beam, a displacement control test, a displacement control test with a material defined by a user-defined stress-strain curve, a crystal plasticity model, and pallet. Results were outputted to Exodus II files for viewing in Paraview, and the results were verified by checking the stress-strain curves. Results from these use cases show that the input files generated from the preprocessor functions were correct.

1. INTRODUCTION

Computational modeling is essential for material and structural analyses: to provide preliminary insight into material and structural reactions, aid in product improvement, and reduce manufacturing costs. However, many commercial finite element packages can be prohibitively expensive for the small- to medium-sized engineering companies. Free finite element solvers, such as Warp3D [1], exist as robust alternatives to commercial finite element analysis (FEA) packages. Despite this, the lack of a built-in preprocessor limits the accessibility of such solvers. As part of the ICSEMS[1] efforts, we developed a preprocessor for Warp3d, which is referred to as *W3DInput*, to generate input files for the processor. *W3DInput* creates a general framework, at no cost, to go from CAD models to structural analysis.

At its crux, *W3DInput* is a method by which CAD models can be converted into input for finite element analysis. This is accomplished by reading mesh data generated from a mesh generator, such as Gmsh [2], and stepping the user through the information input required of a Warp3D input file. Each step in the preprocessor is designed to translate the needs of each component of a Warp3D input file in an easy-to-follow format that requires only a basic knowledge of finite element analysis. As a whole, the preprocessor acts as an easily accessible framework by which Warp3D input files can be generated.

Five use cases have been successfully completed using the preprocessor: a simple cantilever beam, a cube using displacement control, a cube using a stress-strain curve as the material definition, and a plastic pallet, and a crystal plasticity model. Each use case tests the preprocessor's ability to generate an input file for common types of analyses and elements. The cantilever beam tests the preprocessor's ability to create a simple model with linear tetrahedral elements. The cube model was generated with more complex boundary conditions and is a displacement control model, which verified the preprocessor's ability to correctly generate multiple unique boundary conditions. The stress-strain curve tests the ability to use stress-strain

[1] ICSEMS: Indiana Consortium for Simulation-Based Engineering of Materials and Structures. A Purdue consortium for digital analysis of materials and structures where members can collaborate with experts at Purdue in these fields on research projects.

curves in the material definition. The pallet is a test for handling large amount of data. Relative to the beam and cube, this model has relatively a large number of elements with a nonlinear formulation. It was modeled with simple boundary condition as applied loads in order to prevent them from interfering with solution convergence. Finally, a crystal plasticity model was created to test the use of more complex material definitions.

The final output commands of the preprocessor are to convert the generated input file into a Patran-formatted flat model file and generate results files for stress, strain, displacement, and reaction forces. Simple calculations of stress and strain include their six components. At the user's discretion, additional micromechanical fields can be output.

To visualize results, *W3DInput* automatically includes a command to write a Patran-formatted *.text* file, as well as to write the stress, strain, displacement, and reactions forces to individual files. The stress and strain will include only the components specified during the element definition. These files are read into a built-in program in the Warp3D software package that converts this data into a Paraview-compatible Exodus II file [3].

While the capabilities of the preprocessor does not represent the full capabilities of the Warp3d solver, the preprocessor is still a reliable method of generating input files for many solid 3D analysis applications. These applications can be expanded, upon the customer's request. The preprocessor is not limited by geometry, and thus can be represented by linear or quadratic solid tetrahedral of hexahedral elements. The material models defined in the preprocessor are applicable to a wide range of engineering materials for macro-scale analysis. These definitions allow for several different types of analysis, including elastic, plastic (rate-independent and rate-dependent), and fatigue. Further expansions of *W3DInput* will allow for creep, temperature- and time-dependent analyses, and fracture analysis.

2. LITERATURE REVIEW

Warp3D was chosen as the solver to use for this project because of its robustness: it is not limited to any specific type of analysis and is capable of handling linear and nonlinear analyses. Additionally, it is not limited by scale. In regard to meeting the needs of ICSEMS, this solver has several specialized functions related to material and structural analysis and are relevant to several fields of engineering. Some of these functions include crystal plasticity, fracture mechanics, void nucleation, and fatigue.

As stated previously, for small- to medium-sized companies, commercial finite element packages can be prohibitively expensive despite its vital need, and thus open-source software are an excellent alternative. However, most finite element packages come as solvers only without a preprocessing or postprocessing framework built around it. This makes them extremely difficult to use for less advanced finite element analysts. Most preprocessors are built for a single solver, or do not encompass the entirety of the needs of finite element preprocessing. Software such as Gmsh [2], and Cubit [4] are mesh generators and preprocessing apparatuses that do not serve any particular finite element solver, but are each lacking in some way that does not meet the needs of ICSEMS.

2.1 Gmsh

Gmsh is a mesh-generation tool that is comprised of four modules: geometry, meshing, solver import, and postprocessing. Within the geometry module, users can import models in translational formats such as STEP, BREP, and IGES. While Gmsh has a wide array of geometry generation options, it is not intended as a replacement for CAD software [2], and more complex geometries, such as those shown in Figure 1, should be imported. In this regard, it is a useful tool for generating simple geometries that may be used in initial analyses and reducing the number of steps to translate a CAD model to an FEA model. This is not, however, a direct CAD to FEA tool.

Figure 1. Propeller and its mesh in Gmsh [4].

2.1.1 Meshing in Gmsh

Additionally, Gmsh can automatically generate one-, two- and three-dimensional unstructured meshes [2]. In 2D and 3D, these are triangular and tetrahedral meshes, respectively, generated using a Delaunay triangulation based meshing kernel [5]. Order can also be changed automatically. In Gmsh, each element has a characteristic length, a^{char}, defined by Equation 1, that is then used to determine the efficiency of the mesh, κ, shown in Equation 2. Each element has its own efficiency index, κ^e, given by Equation 3, and are summed to determine the overall efficiency of the mesh. The unstructured mesh is generated with the aim of κ being as close to one as possible.

$$a^{char} = \int \frac{1}{\delta(x, y, z)} da$$

Equation 1. characteristic length of an element [2]

$$\kappa = \exp\left(\frac{1}{ne}\sum_{e=1}^{ne} \kappa^e\right)$$

Equation 2. Efficiency index of mesh [2]

$$\kappa^e = a^{char} - 1 \quad a^{char} < 1$$
$$\kappa^e = \frac{1}{a^{char}} - 1 \quad a^{char} \geq 1$$

Equation 3. Element efficiencies given the characteristic length [2]

Delaunay triangulation [6] generates points on a plane in such a way that has a minimum number of colinear points and satisfies the empty circle property. That is, for the small circle drawn around any triangle, that circle contains only one triangle (Figure 2). This ensures a minimum size element, but does not ensure a quality mesh. Instead, Gmsh also has its own optimization procedure to increase the efficiency index [2] and improve mesh quality. This is accomplished by moving node, edges, and faces of triangles. The user can implement the optimization function and the Netgen optimization tool [7] sequentially to improve mesh quality more than implementing them separately [2].

A structured mesh, however, requires the user to manually define the mesh. Users must a transfinite meshing technique, which uses tessellated faces to create pseudo-edges derived from the mid-surface topology of the CAD model [8]. This meshing technique is not a simple option for more complex geometries, such as those that cannot be easily decomposed into prismatic shapes. In Gmsh, the user must apply this individually on edges, surfaces, and volumes. On edges, a user must specify the number of nodes on the edge, and can choose to generate a "biased mesh," or a mesh where one region has a finer mesh than the rest of the model. The surface must then be generated by choosing the transfinite lines. The transfinite volume can then be generated by choosing its boundary with five or six faces [2]. These geometries are called transfinite because the values chosen to define the mesh are values corresponding to an infinite set of numbers from which can be chosen. For structured meshing, this is especially important because the user can control the initial size and sweep of the elements with the use of transfinite lines. A similar function would be the *seed edge* function in Abaqus [9]. This, however, only defines it in one direction. The transfinite surface combines the transfinite lines and matches the node locations to generate a two-dimensional mesh. The transfinite volume can then be used to extrude that mesh into three-dimensions.

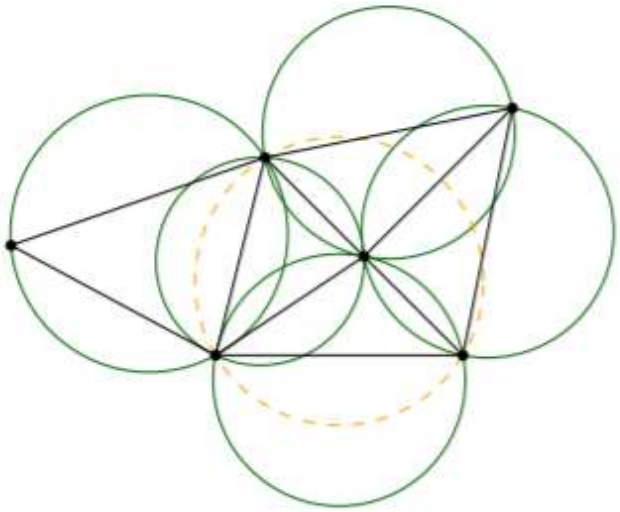

Figure 2. Empty Circle Property as applied to Delaunay Triangulation

2.1.2 Solver

The solver import option in Gmsh allows the user to import finite element solvers for post processing and mesh generation specific to the solver. However, this solver import is expressly for loading the other three modules (geometry, mesh, post-processing) in reference to the solver [2]. This is not a method by which a full input file can be generated. Additionally, the solvers must be imported through a Unix or TCP/IP socket. For use with Warp3D, Gmsh does not find an correct IP address and terminates import, as shown in Figure 3, and thus for preprocessing, is not a viable option either.

```
Info    : Calling '"D:\warp3d_17.9.2\run_windows_64\warp3d.exe"  -onelab "GetDP" 127.0.0.1:49961'
Error   : Abnormal server termination (Socket listening timeout on socket 127.0.0.1:0)
Info    : Done
```

Figure 3. Warp3D solver import error message

2.2 Cubit

Cubit is a “full-featured” software for mesh generation in which, unlike Gmsh, most of the functions and tools, including structured mesh generation, are fully automatic. Like Gmsh, users can import CAD geometries in translational formats using the ACIS solid modeling kernel. This kernel maintains the geometric representation of the model upon import or export, and allows the user to partition the geometry further to aid in the generation of a hexahedral mesh. Additionally, with the use of this kernel in Cubit, poorly defined models can be redefined to improve, or “heal”

the models [4]. The locations of boundary conditions and applied forces can also be explicitly defined.

2.2.1 Meshing

With this software, one of the biggest advantages over Gmsh is that automatic mesh generation for a wider variety of element types. In addition to the tetrahedral and hexahedral 3D solid element types, Cubit also has pyramid and wedge element types. The 2D element types are the same. For these element types, a heuristic element size is established between 1 and 10, where 10 represents the coarsest mesh, and 1 represents the finest [4].

There are three options by which a mesh can be defined: a default scheme, surface auto selection, and volume selection. The default scheme selection allows Cubit to determine the type of element based on the geometry (and any decompositions or partitions). In general, Cubit will attempt to apply a quadrilateral or hexahedral mesh. An example of a hexahedral mesh in Cubit is shown in Figure 4. If this fails, the user must add further partitions or define the mesh explicitly. Note that this default element definition will not generate tetrahedral elements first. If they are specifically desired, the user can use the command *Set Default Element* [TET/TRI] in the command line [4]. The surface and volume scheme selection are methods by which the user explicitly defines the mesh in 2D and 3D, respectively.

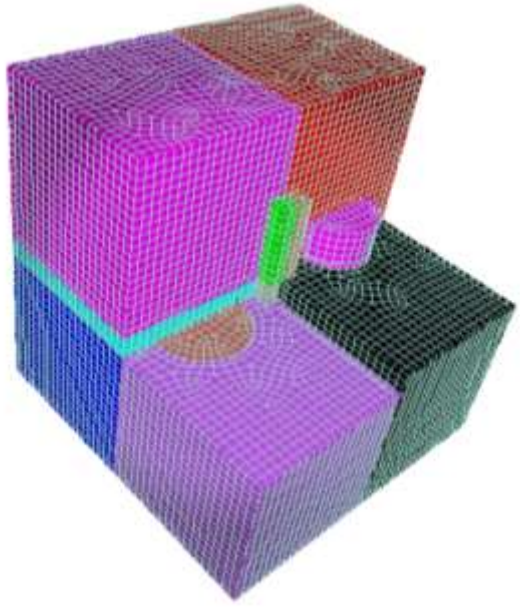

Figure 4. Hexahedral mesh in Cubit [4]

2.2.2 Limitations

Despite the ease of use of Cubit, there are several limitations in its use as a preprocessor for Warp3D. First, Cubit does not allow the user to enter values of the applied forces or boundary conditions, or input material conditions. These are necessary parts of the input file to run an analysis in Warp3D. Essentially, Cubit is capable of providing a mesh and specifying the locations, and easily generating multiple element types on a single model, which is vital to the preprocessing of an finite element solver, but provides no other major functions.

In addition, Cubit is available as a free software to a very limited number of individuals. As it is a software developed at Sandia National Laboratory, the software is only available to U.S. government officials and those with U.S. government contracts. While there is a public version of the software, it is not open-source and comes at a cost, thus does not fit with the open-source preprocessor.

2.3 Other meshing softwares

Many other meshing softwares, such as OpenMesh [10] and MeshLab [11] have similar capabilities as Gmsh and Cubit that allows them to process and generates meshes, but also like Gmsh and Cubit, these softwares are lacking the additional framework tools of full preprocessing suite. Unlike Gmsh and Cubit, however, these tools lack the ability to import CAD models from programs like Solidworks [12] or Creo [13].

Despite this, many of these softwares are able to effectively generate the solid element meshes necessary for input in Warp3D. As such, there is no need to entirely reinvent a meshing software. Rather, to effectively develop an open-source preprocessing framework for Warp3D, it is much more conducive to import the mesh into a preprocessor that can handle the remaining preprocessing needs for FEA in Warp3D. Considering the meshing softwares above, Gmsh is the most viable option for use in this preprocessing framework as is most closely matches its needs.

2.4 Crystal Plasticity

Crystal plasticity models provide a model through which highly anisotropic polycrystalline engineering materials can be analyzed by incorporating macroscopic deformation states and grain-level plastic behaviors. On the macro-scale, materials such as metals behave isotropically, but this assumption does not capture grain-level plasticity behavior vital to understanding the failure mechanisms of the material. Crystal plasticity models allow the effect of texture to be included in mechanical analyses [14].

Classical crystal plasticity models that do not account for grain interaction are based on the work of Bishop and Hill [15], [16] to relate microscopic and macroscopic stresses. To incorporate texture evolution, a Taylor function [17] approximated is used to iterate towards the stresses on each crystal using an initial guess from the yield surface. These models are limited by several assumptions, such as excluding grain interactions or enforcing microscopic stress and strain to macroscopic stress and strain.

For crystal plasticity finite element (CPFE) models, the dislocation motion is homogenized over the slip system [18]. Dislocation motion itself is on the order of Angstroms, whereas a single grain in a polycrystalline engineering material is approximately tens to several hundred microns wide. This makes it intractable to capture individual dislocations with a CPFE model. Slip activity, however, is large enough to capture in a CPFE model where grains are modeled individually and the interactions between grains is captured. On larger length scales, such as when using a Taylor homogenization [1], [15], [16], the microscale and macroscale behavior is captured [19].

Finite element crystal plasticity models have recently become more widely used as they are able to incorporate kinematic effects and grain interactions and are able to solve crystal plasticity models with complex boundary conditions [17]. These methods can be computationally expensive, but closer matches between experimental data and simulations can be generated, allowing for better predictions of material behavior [20]. Simplified models can be used to approximate the material behavior, but the exclusion does not improve the use of simulations to predict materials behavior.

Finite element models are built on constitutive modeling, and begin with the definition of dislocation slip. The deformation gradient is comprised of elastic and plastic portions (Equation 1), and the plastic deformation rate is defined as shown in Equation 5.

$$F = F_e F_p$$

Equation 4. Deformation gradient decomposed into elastic and plastic components [21]

$$\dot{F} = L_p F_p$$

Equation 5. Deformation rate as a function of the plastic velocity gradient and the plastic deformation gradient [21]

Where L_p is the velocity gradient and is defined as

$$L_p = \sum_{a=1}^{N} \dot{\gamma}^a \boldsymbol{m^a} \otimes \boldsymbol{n}^a$$

Equation 6. Velocity gradient [22]

Equation 6 describes the velocity gradient as a function of the shear strain rate, and the slip directions and normal to the active slip planes. The resolved shear stress on the plane is then defined as

$$\tau^\alpha = \boldsymbol{S} * (\boldsymbol{m^a} \otimes \boldsymbol{n}^a)$$

Equation 7. Resolved shear stress on a slip system [21]

The flow rule in Equation 8 relates the shear stress τ^α on the slip system to the shear strain rate $\dot{\gamma}^a$ [23], [24]. In this equation, $\dot{\gamma}_0^\alpha$ is a reference strain rate, τ^α is the stress on the slip system, and τ_c^α is the critical resolved shear stress. Slip on a system occurs when the shear stress on the slip system is greater than the critical resolved shear stress.

$$\dot{\gamma}^a = \dot{\gamma}_0^\alpha \left| \frac{\tau^\alpha}{\tau_c^\alpha} \right|^{\frac{1}{m}} sgn(\tau^\alpha)$$

Equation 8. Flow rule relating stress and shear strain rate on a slip system [21]

The effect of the critical resolved shear stress on hardening behavior is described by Equations 7 and 8. The critical resolved shear stress can be defined as a function of the hardening matrix, $h_{\alpha\beta}$ that a slip system β induces on a system α. The parameters $h_0, \tau_s,$ and α define the slip hardening, and $q_{\alpha\beta}$ is a measure of latent hardening. This value is generally 1.0 for coplanar slip systems. For other slip systems, the value is 1.4, and indicates that the hardening model is anisotropic [21].

$$\tau_c^{\alpha} = h_{\alpha\beta}\left|\dot{\gamma}^{\beta}\right|$$

Equation 9. Hardening behavior on a slip system [21]

$$h_{\alpha\beta} = q_{\alpha\beta}\left[h_0\left(1 - \frac{\tau_c^{\beta}}{\tau_s}\right)^{\alpha}\right]$$

Equation 10. Hardening matrix as a function of resolved shear stress [21]

Many hardening behavior models exist, including Mechanical Threshold Stress (MTS) [25] and Voce [24], [26], [27]. The MTS hardening function described how the strength of a pristine crystal creates obstacles to dislocation motion, and thus the strength of the crystal increases due to work hardening[25]. Similar to the MTS model, the Voce model uses a power-law hardening model to relate the slip rate a reference strain rate, the critical resolved shear stress, and the resolved shear stress on the slip plane [24], [26], as shown in Equation 9. This equation holds for both MTS and Voce [1].

Warp3D uses these equations in addition to Green-Nagdhi stress rate formulation to incorporate the kinematics of crystal plasticity [1]. The general framework for this is shown in Figure 5.

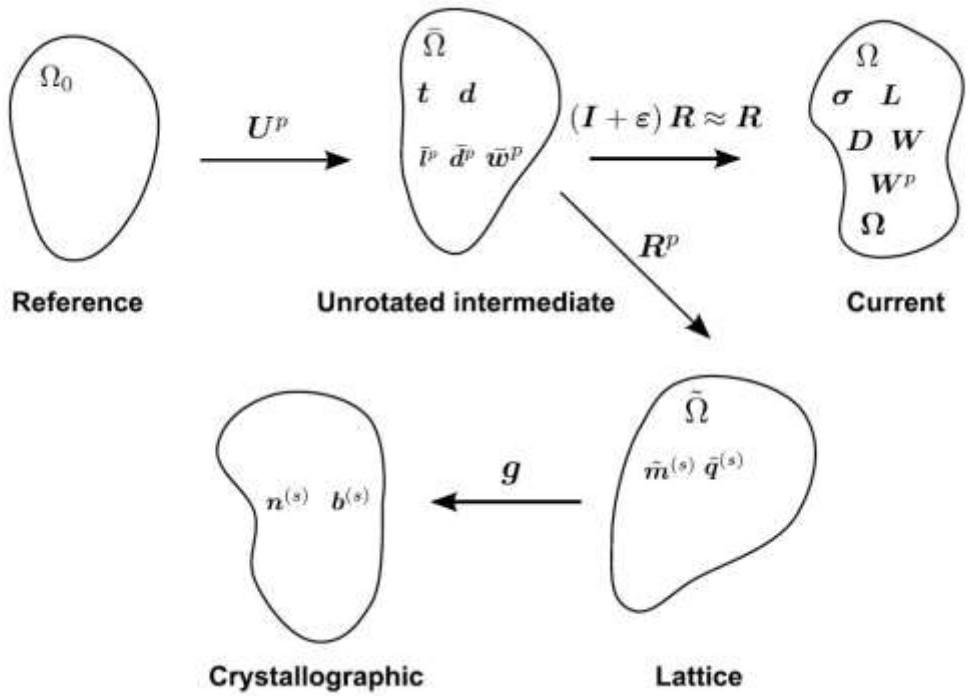

Figure 5. Crystal plasticity framework for Warp3D [1].

The deformation gradient is defined using Equation 4 and Equation 5, where the velocity gradient becomes the plastic velocity gradient, and is composed of symmetric and skew-symmetric components, *d*, and *w*. The symmetric and skew symmetric components in the reference frame, *D*, and *W*, are related to *d* and *w* by Equation 11 and Equation 12, respectively [1].

$$\boldsymbol{D} = \frac{1}{2}(\boldsymbol{L} + \boldsymbol{L^T}) = \dot{\varepsilon} + \varepsilon\dot{\boldsymbol{R}}\boldsymbol{R}^T - \dot{\boldsymbol{R}}\boldsymbol{R}^T\varepsilon + \boldsymbol{R}\bar{\boldsymbol{d}}^{\boldsymbol{p}}\boldsymbol{R}^T + \varepsilon\boldsymbol{R}\bar{\boldsymbol{w}}^{\boldsymbol{p}}\boldsymbol{R}^T - R\bar{\boldsymbol{w}}^{\boldsymbol{p}}\boldsymbol{R}^T\varepsilon$$

Equation 11. Symmetric component of velocity gradient [1]

$$\boldsymbol{W} = \frac{1}{2}(\boldsymbol{L} - \boldsymbol{L^T}) = \dot{\boldsymbol{R}}\boldsymbol{R}^T + \boldsymbol{R}\bar{\boldsymbol{w}}^{\boldsymbol{p}}\boldsymbol{R}^T + \varepsilon\boldsymbol{R}\bar{\boldsymbol{d}}^{\boldsymbol{p}}\boldsymbol{R}^T - R\bar{\boldsymbol{d}}^{\boldsymbol{p}}\boldsymbol{R}^T\varepsilon$$

Equation 12. Skew-symmetric component of velocity gradient [1]

The Green-Naghdi stress rate $\check{\sigma}$ is defined as the dot product of the stiffness matrix with the strain rate as a function of the velocity gradient, added to the stress rate:

$$\boldsymbol{C}: \dot{\varepsilon}^{tot} = \dot{\boldsymbol{\sigma}}$$

Equation 13. Stress-rate in the current frame [1]

$$\boldsymbol{C}: \boldsymbol{D} = \dot{\boldsymbol{\sigma}} + \boldsymbol{\sigma}\dot{\boldsymbol{R}}\boldsymbol{R}^T - \dot{\boldsymbol{R}}\boldsymbol{R}^T\boldsymbol{\sigma} + \boldsymbol{C}: \left(\boldsymbol{R}\bar{\boldsymbol{d}}^{\boldsymbol{p}}\boldsymbol{R}^T\right) + \boldsymbol{\sigma}\boldsymbol{R}\bar{\boldsymbol{w}}^{\boldsymbol{p}}\boldsymbol{R}^T - \boldsymbol{R}\bar{\boldsymbol{w}}^{\boldsymbol{p}}\boldsymbol{R}^T\boldsymbol{\sigma}$$

Equation 14. Tensor dot product of stiffness matrix and symmetric velocity gradient [1]

$$\overset{\circ}{\boldsymbol{\sigma}} = \boldsymbol{C}: \left(\boldsymbol{D} - \boldsymbol{R}\bar{\boldsymbol{d}}^{\boldsymbol{p}}\boldsymbol{R}^T\right) - \boldsymbol{\sigma}\boldsymbol{R}\bar{\boldsymbol{w}}^{\boldsymbol{p}}\boldsymbol{R}^T + \boldsymbol{R}\bar{\boldsymbol{w}}^{\boldsymbol{p}}\boldsymbol{R}^T\boldsymbol{\sigma}$$

Equation 15. Green-Nagdhi stess rate [1]

The inclusion of the microscopic vorticity, $\bar{\boldsymbol{w}}^{\boldsymbol{p}}$**,** enables the texture evolution to be included in the plastic velocity gradient in the intermediate frame [1].

On the lattice frame, the plastic deformation is defined as shown in Equation 6, which is converted from the crystallographic frame by the rotation tensor $\boldsymbol{g}$, defined in Equation 16. The constitutive equations are then used to find the shear stress and strain in the crystallographic frame. The stress updates using a backward Euler integration, where the rotation matrix updates as defined in Equation 17 [1].

$$\boldsymbol{g}^T = \begin{bmatrix} -sin\psi sin\phi - cos\psi cos\phi cos\theta & sin\psi cos\phi - cos\psi sin\phi cos\theta & cos\psi sin\theta \\ cos\psi sin\phi - sin\psi cos\phi cos\theta & -cos\psi cos - sin\psi sin\phi cos\theta & sin\psi sin\theta \\ cos\phi sin\theta & sin\phi sin\theta & cos\theta \end{bmatrix}$$

Equation 16. Transformation matrix from the reference to the crystal frame

$$\boldsymbol{R}^{\boldsymbol{p}}_{n+1} = \exp\left(\bar{\boldsymbol{w}}^{\boldsymbol{p}}_{n+1}\right)\boldsymbol{R}^{\boldsymbol{p}}_{n}$$

Equation 17. Rotation matrix update [1]

Where *n* is the time step. Additionally, the Jacobian is calculated by Equation 20, and residual stress are found using a Newton-Raphson solver. Non-zero values of the rotation tensor are treated as the residual stress, and are derived by Equation 21, which relates the Cauchy stress, $\boldsymbol{t}$, on the unrotated intermediate frame, the resolved shear stress on the slip plane in the crystal frame, $\tilde{\boldsymbol{\tau}}^{\boldsymbol{\alpha}}$ and the Jacobian to the residuals.

$$\tilde{\boldsymbol{\tau}}^{\boldsymbol{\alpha}} = \boldsymbol{g}^T\boldsymbol{\tau}\boldsymbol{g}$$

Equation 18. Resolved shear stress on a slip plane, rotated from the current frame

$$\boldsymbol{t} = \boldsymbol{R}^T \boldsymbol{\sigma} \boldsymbol{R}$$

Equation 19. Cauchy stress on the unrotated intermediate frame

$$J = \frac{\partial f_\beta}{\partial \sigma_\delta^{'c}} = \sum_\alpha n \frac{\dot{\gamma}_0}{\bar{\tau}} \left| \frac{\tau^\alpha}{\bar{\tau}} \right|^{n-1} m_\beta^\alpha m_\delta^\alpha$$

Equation 20. Jacobian function as a function of stress on the slip system

$$\boldsymbol{J} \begin{bmatrix} \boldsymbol{t} \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{R}_2 \end{bmatrix}$$

Equation 21. Rotation tensors for the Cauchy stress on the unrotated intermediate frame and the shear stress on the crystal frame [1]

The Jacobian is a 2x2 matrix, which, when multiplied by the stress gives $\boldsymbol{R_1}$, which is the plastic rotation tensor to update the stress in the unrotated intermediate frame. When the Jacobian is multiplied by the shear stress in the crystal frame, it gives the plastic rotation tensor, $\boldsymbol{R}_2$, to update the shear stress in the crystal frame.

As the hardening model employed has an effect on the slip rate and the hardening behavior, several models are included in the Warp3D package: Voce, Mechanical Threshold Stress, Ma, Roters, and Raabe, power-law hardening for titanium, and a user-defined hardening model. The simplest, Voce, utilizes Equation 8 to define the slip rate and Equation 22 for the hardening, where τ_w is the hardening resistance, t_v is the work hardening saturation strength, and θ_0 is the hardening slip for stage III hardening, as shown in Figure 6 [1]. Stress updates are accomplished with a forward Euler integration, and is shown in Equation 23.

$$\dot{\tau}_w = \theta_0 \left(1 - \frac{\tau_w}{t_v}\right)^m \sum_{\alpha=1}^{n_{slip}} |\dot{\gamma}^\alpha|$$

Equation 22. Hardening behavior utilized in Voce model [1]

$$\tilde{\tau}_{n+1} = \tilde{\tau}_n + \theta_0 \left(1 - \frac{\tilde{\tau}_{n+1} - \tau_y}{t_v}\right)^m \sum_{\alpha=1}^{n_{slip}} |\Delta\gamma_{n+1}^\alpha|$$

Equation 23. Forward Euler integration of stress updates [1]

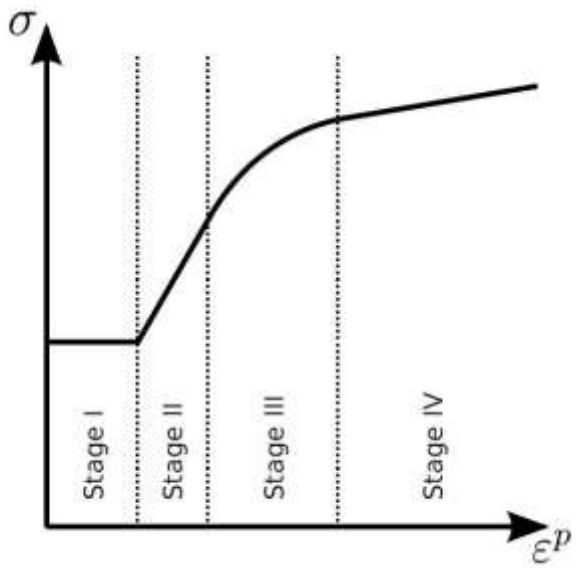

Figure 6. Hardening Stages for Voce hardening model [1].

As this crystal plasticity model can be extremely complex, its implementation in Warp3D requires a preprocessor to aid in input. For simpler models such as the Voce model, which has several options for simplification in Warp3D, has a very limited number of parameters that the user is required to input. A preprocessor will help aid in understanding the implications of the simplified models and what the values will entail.

3. PREPROCESSOR

The preprocessor, w3dInput, written entirely in the open-source programming language Python v. 2.7 [28], works by stepping the user through a series of input options within 5 modules, shown in Figure 7. The initial input mainly focuses on importing the mesh and extracting the information. The formatting is removed, and the data is moved to a subdirectory to the main input file. Materials are then defined in three parts: stress-strain curves, the material model, and the functionally graded material definition. The stress-strain curves and the functionally graded material definition are optional. Next, element parameters are defined from four three-dimensional element types: linear tetrahedral, quadratic tetrahedral, linear hexahedral, and quadratic hexahedral. Boundary conditions and applied forces are defined by specifying a node or element set, and then the values are assigned. The solution controls contain a basic set of controls of a mechanical test, including the thread type and iteration limits. Finally, the output controls dictate which values will be recorded to the analysis log and for which steps. A command is included in the input file to save the results to flat text files for export to Paraview [29].

As this preprocessing framework is meant to be an open-source method in which CAD models can be converted to FEA models, the first step is to generate a mesh. In some of the use cases discussed in the following chapter, the open-source mesh generator Gmsh was used. The CAD models can be imported as *.step* or *.stl* files, and meshed using the 3D mesh option. Quadratic elements can be generated using the "order 2" option. Node and element sets can also be created by generating physical groups in the geometry tab. The user can export the mesh in an Abaqus [9] or Nastran [30] format.

Hex-dominated meshes must be generated manually in Gmsh. The user can still import a CAD file to generate the geometry, but the grid lines for the model must be defined manually. Then, the transfinite line function must be defined in terms of the physical lines of the geometry and the grid lines defined. These lines must then be extruded through the model, and the transfinite lines need to be joined to form a transfinite surface. Each transfinite surface needs to be

recombined and extruded up to the next gridline. The physical volume can then be generated from the surfaces. This set-up will allow the user open the file again in Gmsh, and generate a mesh with brick elements. Figure 8 shows a tetrahedral mesh generated through Gmsh.

In general, the user will be asked to enter a number corresponding to a choice. For example, to choose from the node and element sets to keep, the user will be prompted to enter 1 to do so. The preprocessor will then display the names of all sets found, and then the user will enter the indices of the sets they want to keep. Exceptions to this type of input are when a name, file, or list with multiple components is requested.

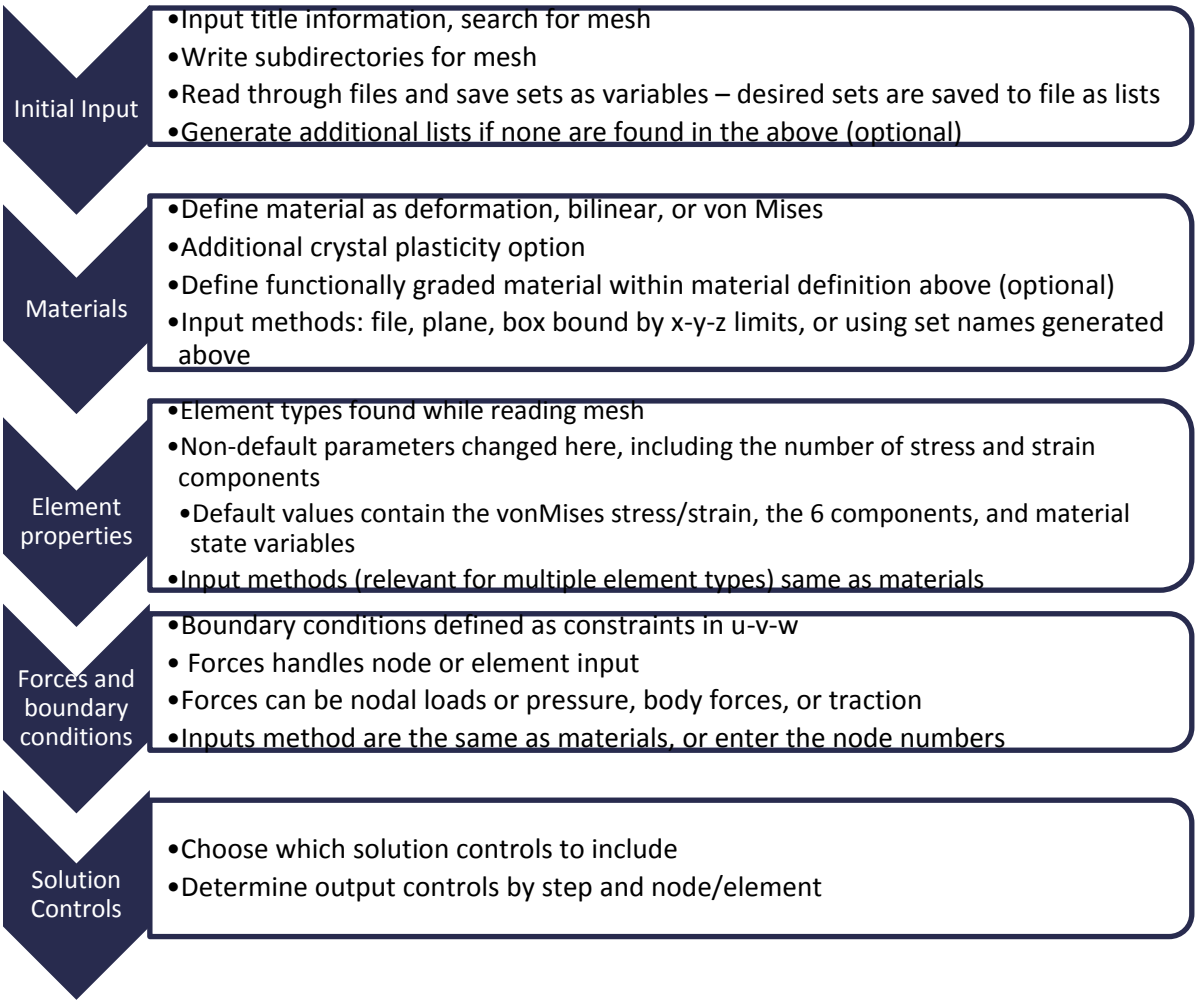

Figure 7. Work flow chart of Preprocessor modules, including descriptions of the main functions.

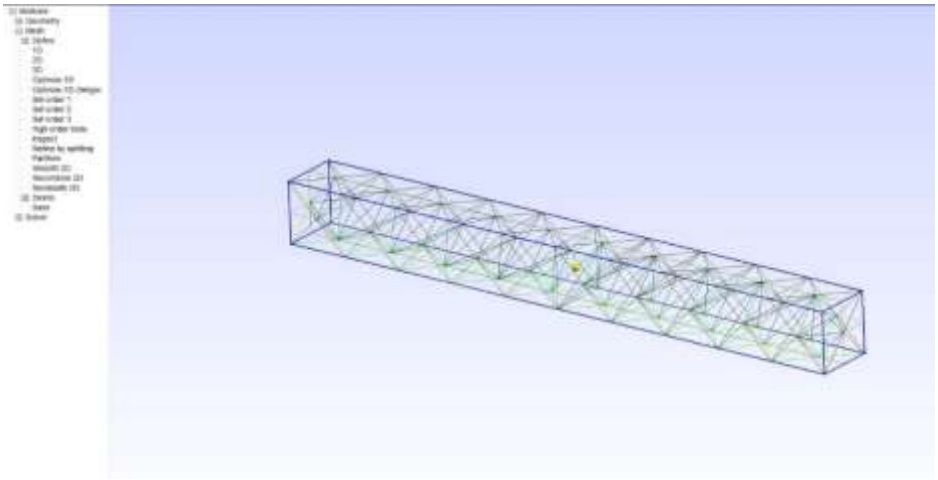

Figure 8. Gmsh UI and example of a tetrahedral mesh on a cantilever beam

3.1 Initial inputs

Upon initializing the preprocessor, the user will be directed to specify a directory where the mesh is located, to give a name to the input file, and a name to the model. The preprocessor will first look for a file containing the word mesh. If the number of files containing the word is not one, the user will be prompted to enter a file name. This file will be parsed for node and element information. The formatting is removed, and the data is saved to two subdirectory input files, mesh_coord.inp and mesh_incid.inp, respectively. Additionally, the mesh file will be searched for node and element sets. The names will have _n or _e appended to the end of the name to aid in differentiating between the set types. The user can then choose whether any are kept. Aside from the appended suffix to the name, there is no differentiation in the lists between nodes or elements. These are written to the input file as number lists that can be referenced by the name in Warp3D and in the preprocessor. Figure 9 shows the parsing process of the mesh file.

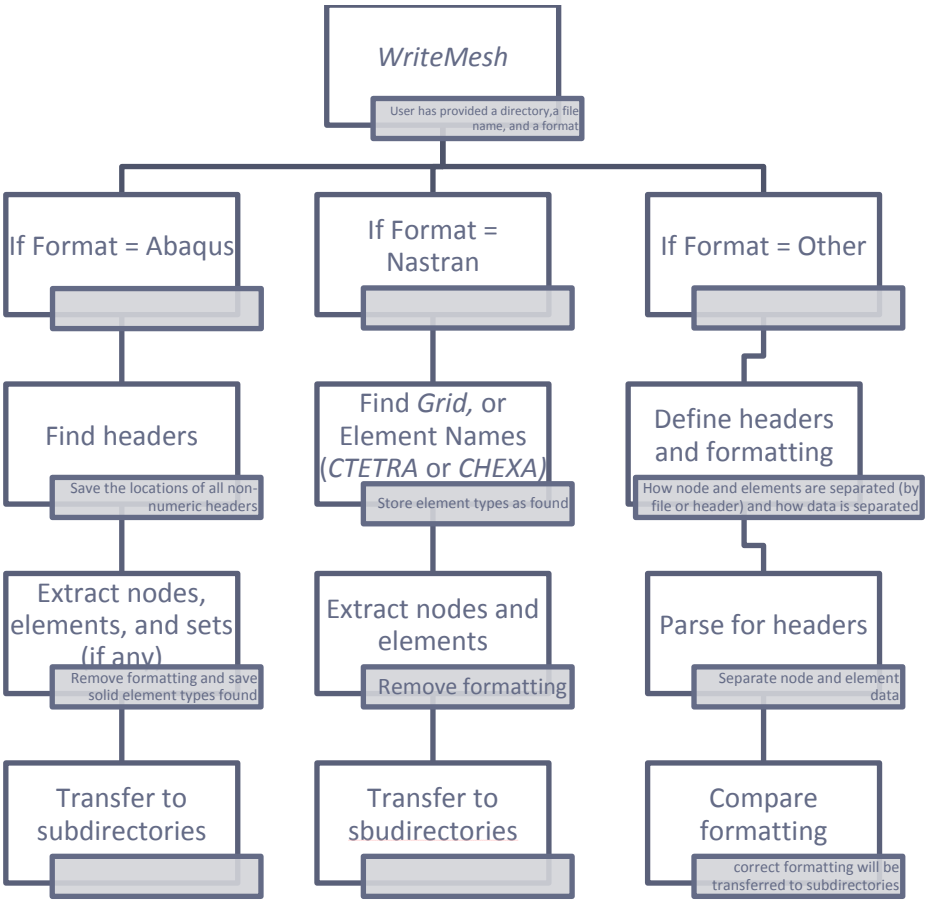

Figure 9. Parsing Process of code writeMesh to generate subdirectories for nodes and elements; optional generation of node and element lists.

For each format type, *writeMesh* opens the file and writes each line to array index. For the Abaqus format, the searches for the headers *NODES and *ELEMENTS. The nodes are considered to be all lines between these two headers and are saved to an array. Each index in the array then has the formatting removed and the data is copied to mesh_coords.inp. When parsing for elements, the code searches for element type codes C3D4, C3D10, C3D8, C3D20, C3D8R, and C3D20R. Any subsequent headers without any of these codes in them are treated as the end point of the elements. Between the header *ELEMENTS and the designated end point, the code will save any code types found, and save the code types to an array for later use in the element module. If none are found, the code will check the number of connected nodes for each element to determine the element type. Again, these are saved to an array for later use. All numerical data is saved to an array, the formatting is removed, and the data copied to mesh_incid.inp.

If among the remaining headers are titles NSET= or ELSET=, the code will save the name next to the titles and assign the lines between headers to the preceding name. If the user chooses to save the set to the input file, the user can rename it as they choose. The data is converted from string to float, and abbreviated. If the abbreviated list is longer than 25 indices, the list is saved to a subdirectory file with the same name.

For the Nastran format, *writeMesh* looks for three line-titles: GRID, CTETRA, and CHEXA [30]. Each line starting with GRID is a node, and each line starting with CTETRA or CHEXA is a tetrahedral or hexahedral element, respectively. For lines beginning with GRID, all numbers are copied to the node subdirectory input file. For element lines, the first number is the element ID number. The second number, usually a 1, is removed. Next, the last number on the line is checked to see if it begins with +E. If so, the element is quadratic, and the remaining nodes are on the subsequent line. This is verified by checking that the next line also begins with +E. The numbers beginning with +E are also removed and the element data is copied to a single line in mesh_incid.inp. The element types are verified by checking the number of nodes per element, and the element types are saved to an array for use in the element module. The Nastran format does not generate sets; therefore, if the user wishes to generate node or element sets, it must be done by methods described on the next page.

If the user specifies “other” as the format, the user can specify no headers, or input the header names for the nodes, elements, and sets (if any). Next, the user will be asked to specify whether the data is separated with commas or spaces. The code will search for the headers to separate the data. If any formatting is present, it is removed and copied to the subdirectory files. Again, if sets are present, the user can choose which to save and the data will be written to the main input file.

Additional node and element sets can be generated other files, by defining a plane, a bounding box, or by direct input. Node and element sets from file are parsed in the same fashion as from the mesh file. The names and the associated lists are saved, and the names are printed to the user to choose which sets to keep. The plane is defined by the equation $ax + by + cz + d = 0$. The user will be prompted to enter the coefficients, and then the preprocessor will find the nodes that lie on the plane. For simple planes, such as x=1, the plane will be written to the main input file as

such, as this is a legible format to Warp3D. Elements on the plane are found by using the connectivity to determine the face number. Figure 1Figure 10 shows the process by which nodes and elements are found on a plane. For the bounding box, the user is directed to input the upper and lower bounds in *x*, *y*, and *z*. The user can decide whether the elements must lie completely within the box, or only a single face. This process is outlined in Figure 11. Lastly, the user can directly enter the nodes or elements. For this input type, there will be no differentiation between a node or element set unless the user specifies one. Except for the file input, the user must name the lists. The numbers in each list are abbreviated and written to the main input file in a single line. This function in shown in Appendix A.

For plane input, combinations of three nodes corresponds to linear tetrahedral elements, four to linear hexahedral elements, six to quadratic tetrahedral elements, and eight to quadratic hexahedral elements. These numbering also correspond to the bounding box input where the user requires only one face to lie within the bounds. For bounding box input where the entire element must lie within the box, combinations of four nodes corresponds to linear tetrahedral elements, eight to linear hexahedral elements, ten to quadratic tetrahedral elements, and twenty to quadratic hexahedral elements.

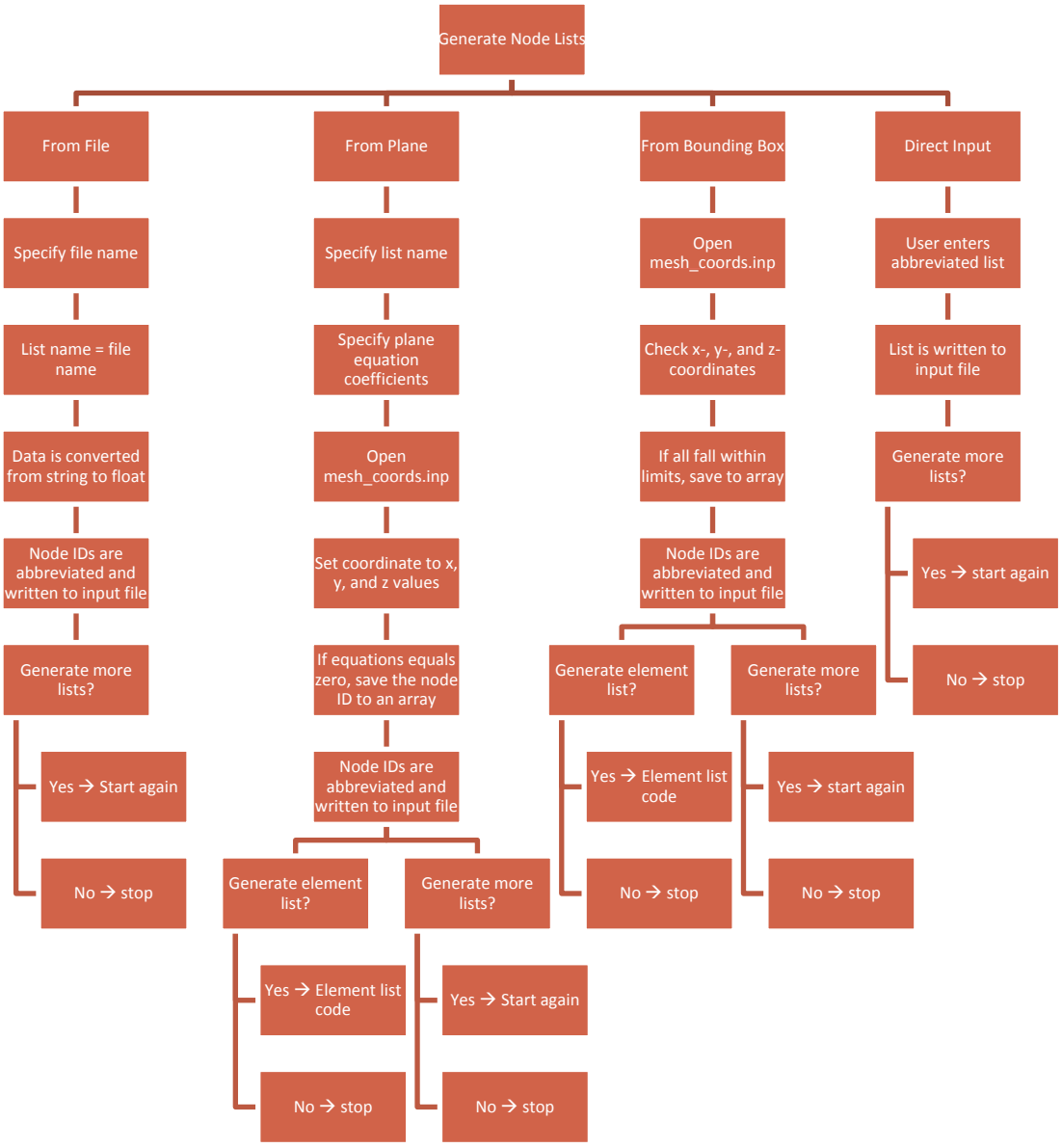

Figure 10. List generation process for each input type.

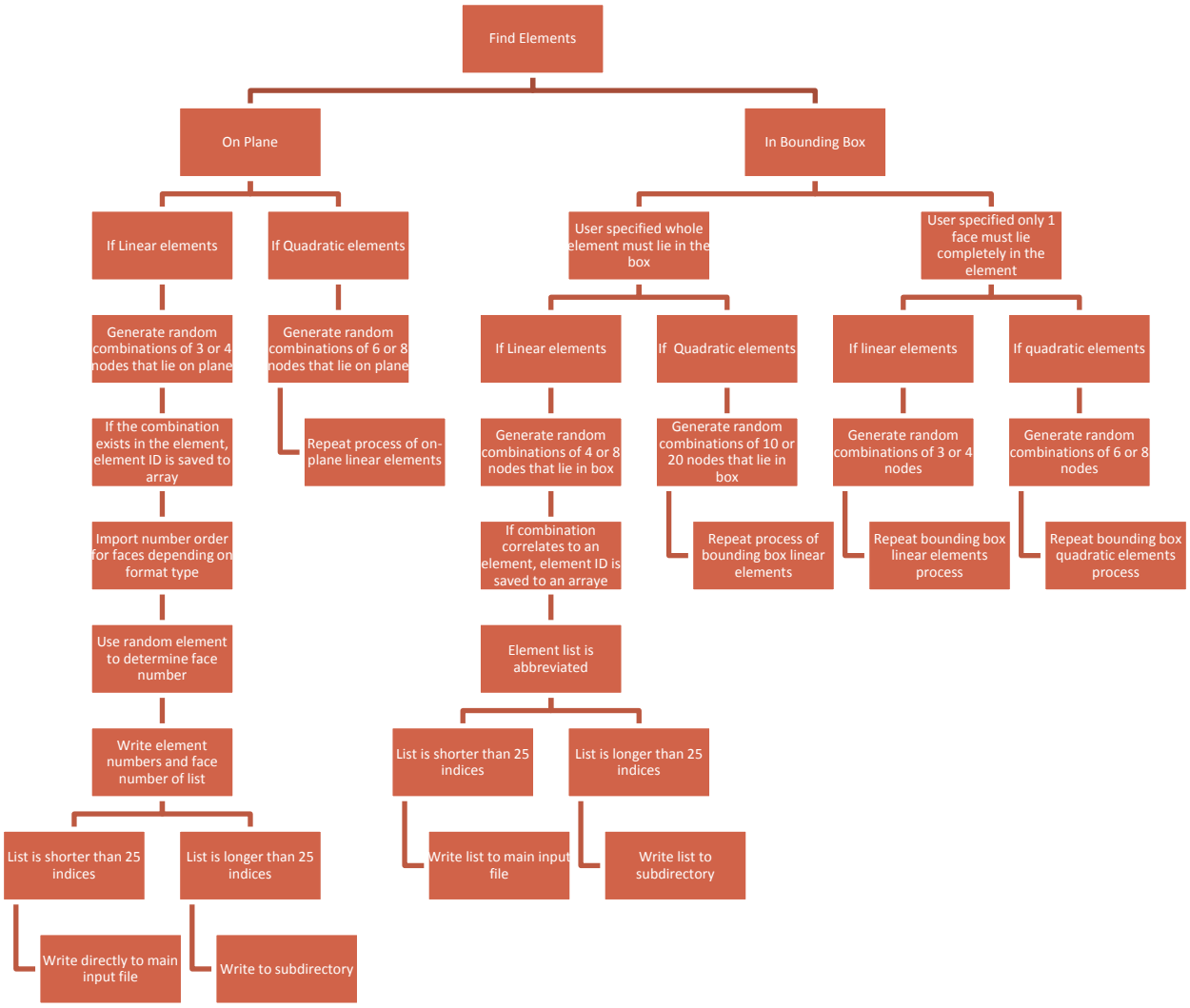

Figure 11. Processes to find elements and face numbers from a plane or bounding box input.

3.2 Materials

There are three sections of the material model, two of which are options only available with certain material models and constants. The first section is inputting user-defined stress-strain curves, which can be used in two of the four material models included in the preprocessor: deformation, bilinear, von Mises, and crystal plasticity. These will be discussed further in this chapter. This input is optional. In the material module, stress-strain curves must be defined first, then the material model. The last section is the input of functionally graded material definitions. This option allows the user to vary the certain properties under a single material definition, and

define properties by location. This too, is optional, and will be discussed further in this chapter as well.

3.2.1 Stress-strain curves

Stress strain curves are identified by a curve number and can be inputted by either a file or direct input; these two inputs cannot be combined for a single stress-strain curve definition. Warp3D has several rules that are for formatting the curve points, which the preprocessor checks through. Instructions for the stress-strain curves also include recommendations for the type of data (i.e., using logarithmic strain-Cauchy true stress vs. engineering strain-stress) and a list of the formatting rules:

1. The data points should be ordered strain-stress
2. The number of points used to define a curve is unlimited, but only 20 can be read on a single line.
3. All strain-stress values must be positive
4. Strain must increase monotonically, stress does not
5. Strain should be total strain
6. The first point on the curve should be the first non-zero strain-stress point.
7. After the last point in the curve, Warp3D assumes a perfectly plastic response.
8. The elastic modulus defined in the material model (bilinear or von Mises) must match the modulus of the linear region of the curve.

The preprocessor checks for and corrects for rules one through three and rule six. Rule eight is verified by Warp3D. Figure 12 details correction process for the remaining steps as well as the process to generating the stress-strain curves. As the user is generating stress-strain curves, the code is keeping track of the number and assigning the count number as the number ID to the curve. The user can generate a single curve from either a file or by directly inputting the data points. The data from either input option will be converted from string to floating point numbers. If the first point in the data is (0,0), the point is removed. The code then checks that the order is strain-stress by verifying that the first index in the data point is less than the second. If not, the order is switched. All data is saved as the absolute value. Lastly, the code limits the number of

data points per line to 10 for input from a file, and 20 for direct input. Excess is written to the subsequent lines.

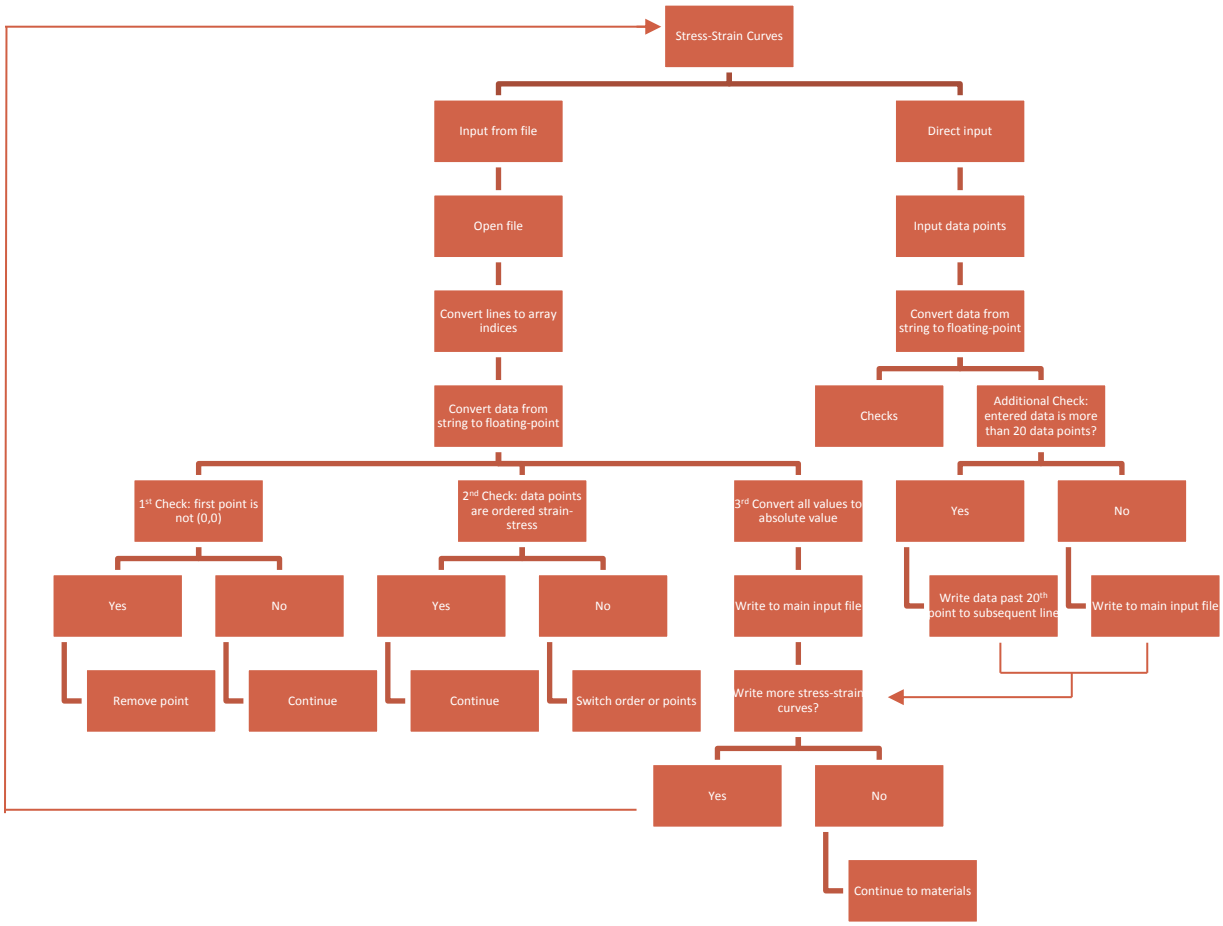

Figure 12. Stress-strain curve input and format verification process

3.2.2 Material models

Four of the ten material models in Warp3D are included in the preprocessor: deformation, bilinear, von Mises, and crystal plasticity. The crystal plasticity model will be described in Section 3.2.4. For each model, the user will be asked whether the material model is functionally graded. If the user affirms this, the user will be asked to identify the constants that are part of a functionally graded material (FGM) model definition. This portion of the material definition is described in Figure 13. The remaining constants, if there are any, will be defined within the material model. For each material model, the user is required to assign it a name, which will later

be used as part of the element parameters. In the instructions, the variable names for the material constants will be provided, and the user can choose which variables to define. The remaining values will be set to the default values. Table 1 lists the material constants and the default values for the deformation model and Table 2 lists the constants for the bilinear model. Note that the deformation model does not support the use of stress-strain curves. The deformation, bilinear, and von Mises models must have at least the elastic modulus, the Poisson's ratio, and the hardening modulus or power-law exponent defined, as the default value is invalid. The preprocessor conducts an error check to ensure these values are defined. If the hardening modulus is not given, the preprocessor will define it as the tangent of the elastic modulus. Figure 14 describes the process by which the deformation, von Mises and Bilinear models are written.

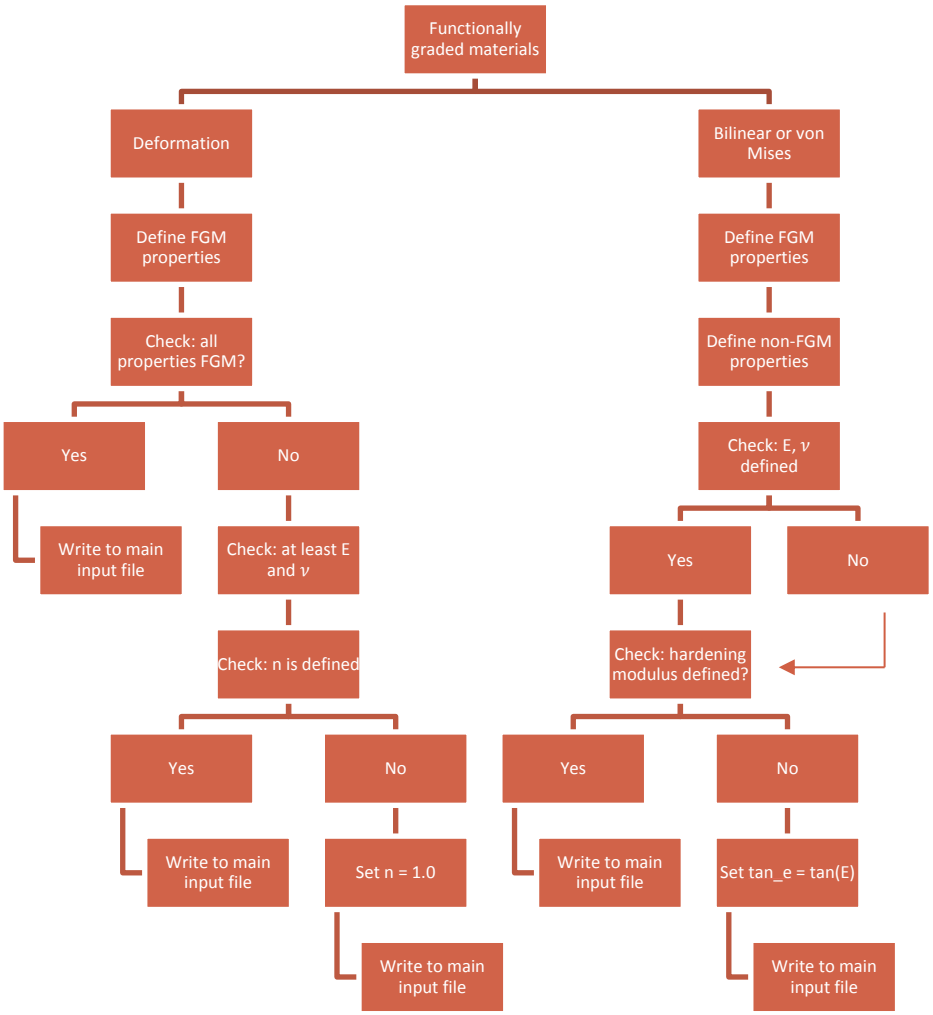

Figure 13. Functionally graded material definition process within the material definition.

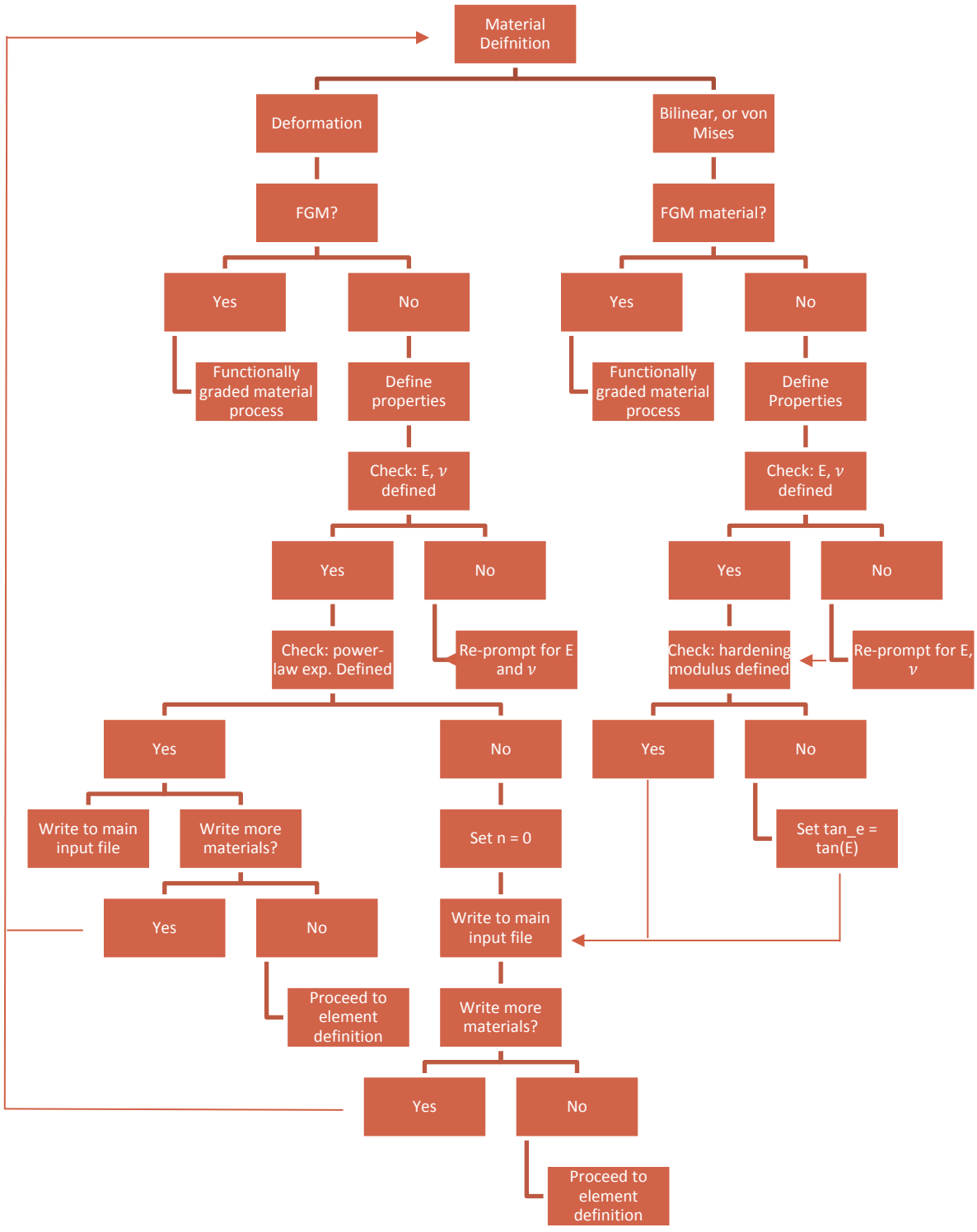

Figure 14. Material definition process for Deformation, Bilinear, and von Mises material models.

3.2.2.1 Deformation model

The stress-strain curve of the deformation model has three parts: the elastic region, a circular elastic-plastic transition region, and a power-law region (Figure 15) [1], and is based on the dissertation of Yongyi Wang [31]. In this model, ϵ_0 and σ_0 represent the yield strain and stress, respectively. K_1 and K_2 are based the upper and lower limits of the circular transition region, and K_1 defines the limit of the elastic region, which is defined by Equation 24. The stress-strain relation in the circular region is defined by Equation 25, and the power-law region is defined by Equation 26. In this region, the stress at the given location on the model is higher than the upper stress limit of the transition region, K_2 [1].

$$\frac{\epsilon}{\epsilon_0} = \frac{\sigma}{\sigma_0}, \qquad \frac{\sigma}{\sigma_0} \leq K_1$$

Equation 24. Stress-strain relation in the elastic region of the deformation stress-strain curve [1]

$$\frac{\epsilon}{\epsilon_0} = \epsilon_{Nc} - \sqrt{r_{Nc}^2 - \left(\frac{\sigma}{\sigma_0} - \sigma_{Nc}\right)^2}, \qquad K_1 \leq \frac{\sigma}{\sigma_0} \leq K_2$$

Equation 25. Stress-strain relation in the circular transition region of the deformation stress-strain curve [1]

$$\frac{\epsilon}{\epsilon_0} = \left(\frac{\sigma}{\sigma_0}\right)^n, \qquad \frac{\sigma}{\sigma_0} \geq K_2$$

Equation 26. Stress-strain relation in the power-law region of the deformation stress-strain curve [1]

Table 1 Material constants and default values for the deformation material model. In parentheses is the variable name used in w3dInput and Warp3D [1].

Constant	Default Value
Elastic modulus (e)	30000
Poisson's ratio (nu)	0.3
Yield Strength (yld_pt)	0.0
Density (rho)	0.0
Power-law exponent (n_power)	0.0

Material constants for the model are given in Table 1, with the default values and the variable name in Warp3D and the preprocessor, *W3DInput*. Note that the default value for the power-law

exponent is zero. This is not a valid value for the power-law exponent, and will result in an error when the input file runs. Instead, if a value is not assigned to the power-law exponent, the preprocessor will automatically assign it a value of 1.0. Additionally, a linear analysis can be defined with this model by setting the yield strength equal to the elastic modulus. This will keep the stress in the model below the lower stress limit K_1, and thus constrain the stress to the elastic region. Effective stress and strain for this model are defined by the von Mises yield function [1].

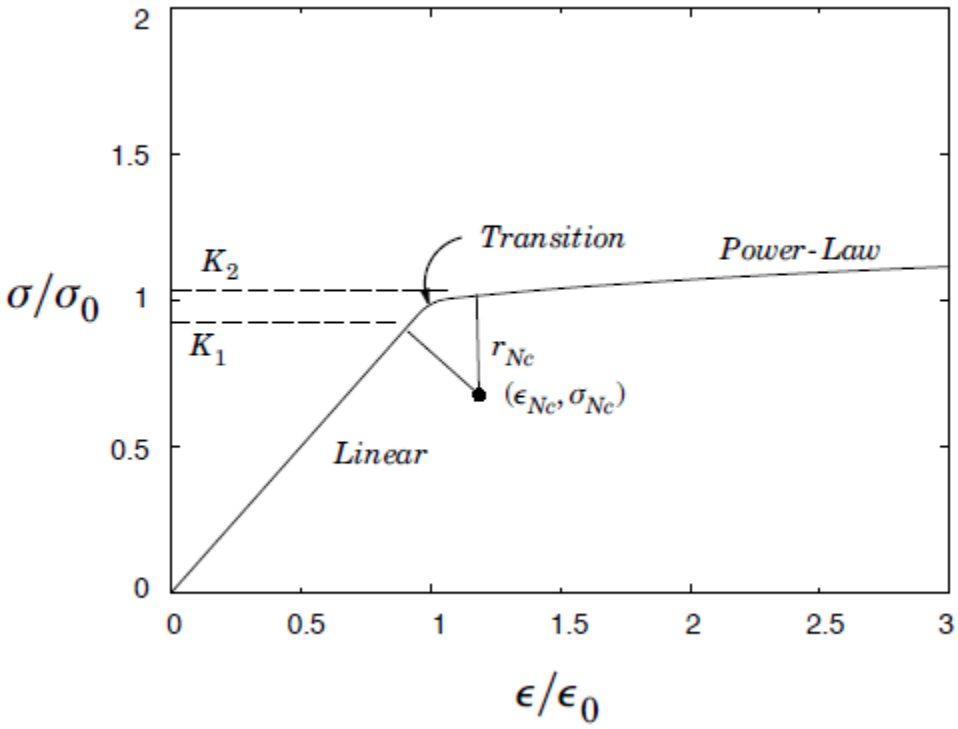

Figure 15. Deformation stress-strain curve [1].

3.2.2.2 Bilinear Model

The bilinear model in Warp3D is a computationally efficient substitute for using the von Mises model [1]. The stress-strain curve, shown in Figure 16, is divided into two parts: an elastic region and a plastic region. The yield strength represents the point at which the curve immediately transitions from the elastic region to plastic region. Figure 16 shows Cauchy true stress, σ, versus logarithmic strain, ϵ, which are related to the engineering stress (σ_E) and strain (ϵ_E) by Equation 27 and Equation 28, respectively.

$$\sigma = \sigma_E(1 + \epsilon_E)$$

Equation 27. Relation of Cauchy true stress to engineering stress and strain [1]

$$\epsilon = \ln(1 + \epsilon_E)$$

Equation 28. Logarithmic strain related to engineering strain [1]

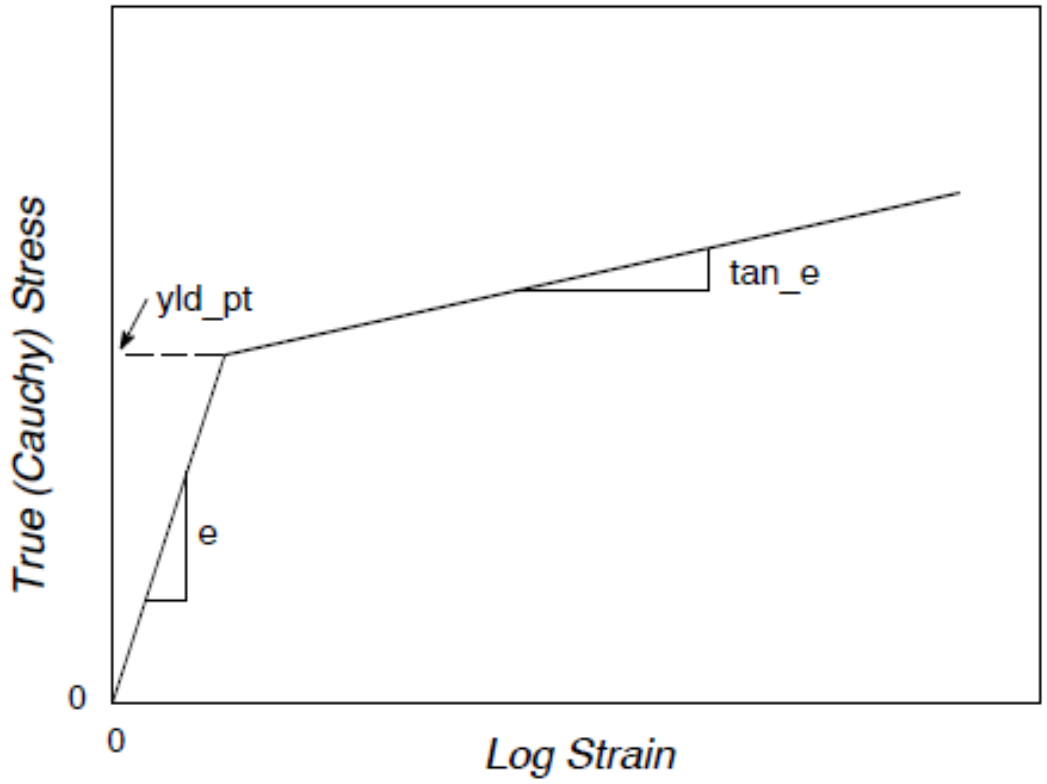

Figure 16. Bilinear stress-strain curve showing Cauchy trues tress vs. logarithmic strain [1]

The slope of the plastic region is referred to as the plastic modulus, H', which is related to the elastic modulus by Equation 29. E_T is a user-defined tangent of the elastic modulus, E, and is referred to as the hardening modulus [1]. The default value for the hardening modulus is 0.0, which is valid for the material model and would indicate an elastic-perfectly plastic model (Table 2). This is noted in the preprocessor, and thus will be left undefined if the user does not define the modulus.

$$H' = \frac{EE_T}{E - E_T}$$

Equation 29. Plastic modulus as a function of the elastic modulus and its tangent.

Table 2. Material constants and default values for the bilinear model. In parentheses are the Warp3D variable names [1].

Constant	Default Value
Elastic Modulus (e)	30000
Poisson's Ratio (nu)	0.3
Yield Strength (yld_pt)	0.0
Density (rho)	0.0
Hardening Modulus (tan_e)	0.0
Hardening Mixity (beta)	1.0
Stress-Strain Curves (curve)	0

The plastic region also incorporates strain hardening, of which there are three types: isotropic, kinematic, and mixed hardening. The value of the hardening mixity, β, can be between 0.0 and 1.0, where $\beta = 1.0$ indicated isotropic strain hardening, $\beta = 0.0$ indicates kinematic strain hardening, and any value in between represents the percentage of the hardening behavior that is kinematic versus isotropic. This hardening is in reference to the radius of the von Mises yield surface, as shown in Figure 17. Isotropic hardening refers to a proportional increase in the plastic modulus, and is the default hardening type. Kinematic hardening refers to the radius of the yield surface translating normal to the surface [1].

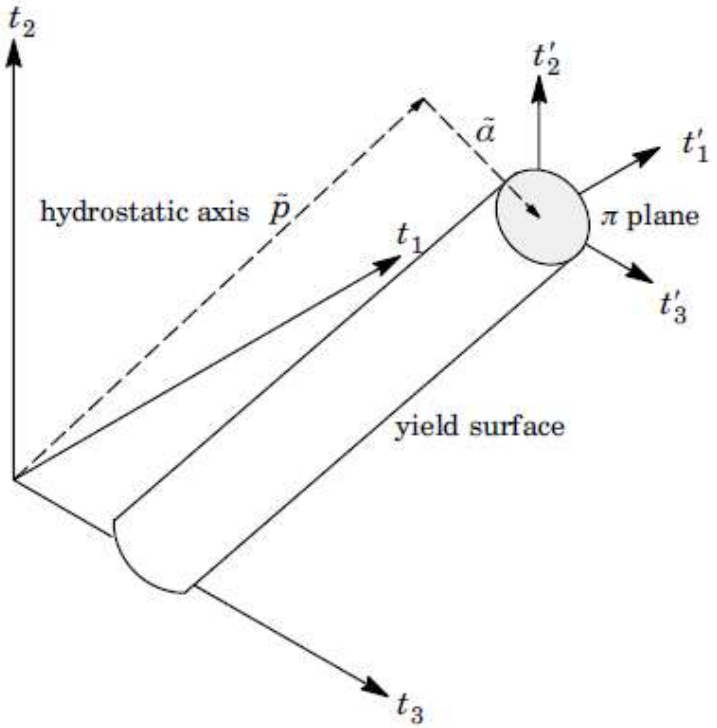

Figure 17. von Mises yield surface [1]

3.2.2.3 von Mises Model

There are three approach options to the von Mises model: linear hardening (option 1), power-law hardening (option 2), or a stress-strain curve definition (option 3). In the linear hardening model, plastic region is defined by the hardening modulus, or the tangent of the elastic modulus. The power-law hardening model is a strain-rate dependent model, and the plastic region has additional dependencies on a reference strain rate, a viscosity exponent, and a power-law exponent. The stress-strain approach is defined by entering a curve number, the elastic modulus, and the Poisson's ratio. All available constants for the von Mises model are listed in Table 3 with the approach option.

The linear hardening model is defined similarly to the bilinear model, with the exception that there is no strain hardening. The stress-strain curve for this approach is the same as in Figure 16, and the stress-strain relations are given by Equation 27 and Equation 28.

The power-law hardening model is a power-law viscoplastic relation as described by Equation 30, where $\dot{\epsilon}^{vp}$ is the viscoplastic strain rate, D reference strain rate, $1/\eta$, q is the rate-dependent stress, σ^e is the inviscid stress, and m is the viscosity exponent. For this approach, D and m are user-defined, as shown in Table 3 with their default values [1].

$$\dot{\epsilon}^{vp} = D\left[\left(\frac{q}{\sigma^e}\right)^m - 1\right]$$

Equation 30. Power-law viscoplastic relation for the power-law hardening von Mises model [1]

Table 3. von Mises material constants and options

Constant	Default Value	Approach Option(s)
Elastic modulus (e)	30000	1, 2, 3
Poisson's Ratio (nu)	0.3	1, 2, 3
Yield Strength (yld_pt)	0.0	1, 2
Density (rho)	0.0	1, 2, 3
Hardening Modulus (tan_e)	0.0	1, 2
Power-law exponent (n_power)	0.0	2
Reference Strain Rate (ref_eps)	0.0	2

Table 3. von Mises material constants and options

Viscosity Exponent (m_power)	0.0	2
Stress-Strain Curves (curve)	0	3

The third approach uses temperature independent stress-strain curves to define the material. As stated in Section 3.2.1, the stress-strain curves can be inputted manually, or inputted with a file. This curve must have an elastic modulus that matches the value entered. For this approach, only the elastic modulus, the Poisson's Ratio, and the density must have defined constants [1].

3.2.3 Functionally Graded Material properties

If a functionally graded material was defined during the material model section of this module, then the user will be directed to this final section of the module: defining the functionally graded material (FGM) properties. This section of the module does not start unless a FGM material was defined in the previous section. In this section, the user must define a region of nodes using either the file input option, referring to a list name, or the bounding box option as defined in Section 3.2 and Figure 10. The preprocessor has already taken note of the properties which were defined as part of the FGM property, and will direct the user to input the values for the material constants. After two regions have been defined, the user will be prompted to indicate whether the FGM definition is complete.

If there is more than one material in the model that is defined as functionally graded, there is no need to refer to the material on the region. Rather, if the two materials have different constants which are functionally graded, Warp3D will be able to differentiate between the two. The tracking of the material constants in the FGM definition helps maintain this consistency. Material constants that have been designated as part of the FGM properties, and will recite them back to the user to define them. The definition process for the FGM properties is outlined in Figure 18.

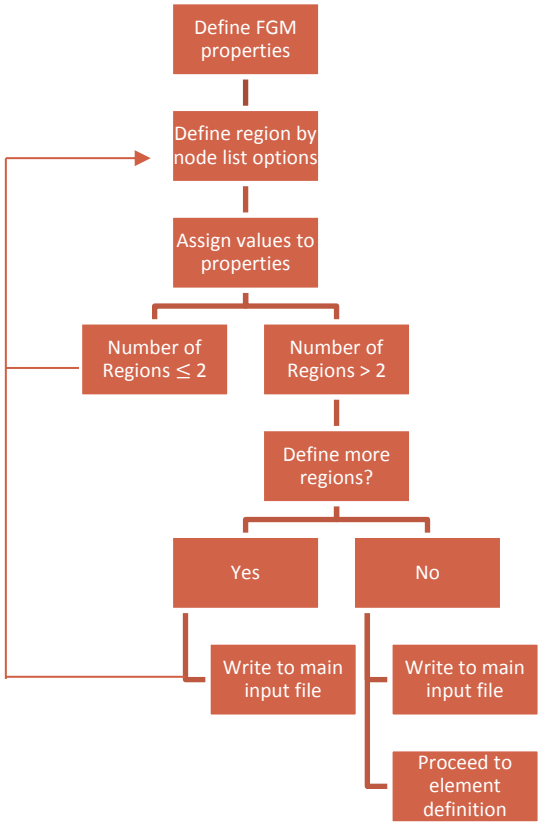

Figure 18. Process to writing the functionally graded material properties by region.

3.2.4 Crystal plasticity model

The crystal plasticity model is made of three required components: a macroscopic material model, the crystallographic properties, and the crystal plasticity model that links the angle inputs and conventions to a crystal property. In the preprocessor, any of the other three material models can be used to define the macroscopic material model. The most important part of this macroscopic material model is that the elastic modulus and Poisson's ratio is defined.

Each set of crystallographic properties are assigned to a crystal number, and contains the type of slip system, the type of hardening model (voce or user-defined), the level of anisotropy of the stiffness matrix, and a reference strain rate. Table 4 lists all of the crystallographic properties and the default values. The crystal properties do not support an FGM definition. The final component has three input types: a single grain (Table 5), multiple grains of the same crystallographic properties (Table 6), or multiple grains of multiple crystallographic properties (Table 7).

Depending on the option chosen, the user will be directed to input either a file containing the location of each grain with reference to the elements and the Euler angles, or a single set of angles. The process to fully defining the crystal plasticity model is outlines in Figure 19.

Table 4. Crystallographic properties and hardening types

Property	Default Value	Hardening Type
Slip System (slip_type)	'fcc'	-
Elasticity Type (elas_type)	Isotropic	-
Elastic Modulus (e)	69000	-
Stiffness Matrix (C11,C12,C22,C33,C44, C55)	_	-
Shear Modulus (mu)	-	-
Poisson's Ratio (nu)	0.33	-
Reference strain rate (gamma_bar)	1E10 1/s	Voce
Hardening exponent (voce_m)	1.0	Voce
Yield Strength (tau_y)	0.0	Voce
Work Hardening Saturation Strength (tau_v)	0.0	Voce
Slope of hardening rate (theta_0)	100	Voce
User-defined hardening properties (u_1-u_6)	-	User Defined
User defined crystallographic properties (cp_001-cp_100)	-	User Defined

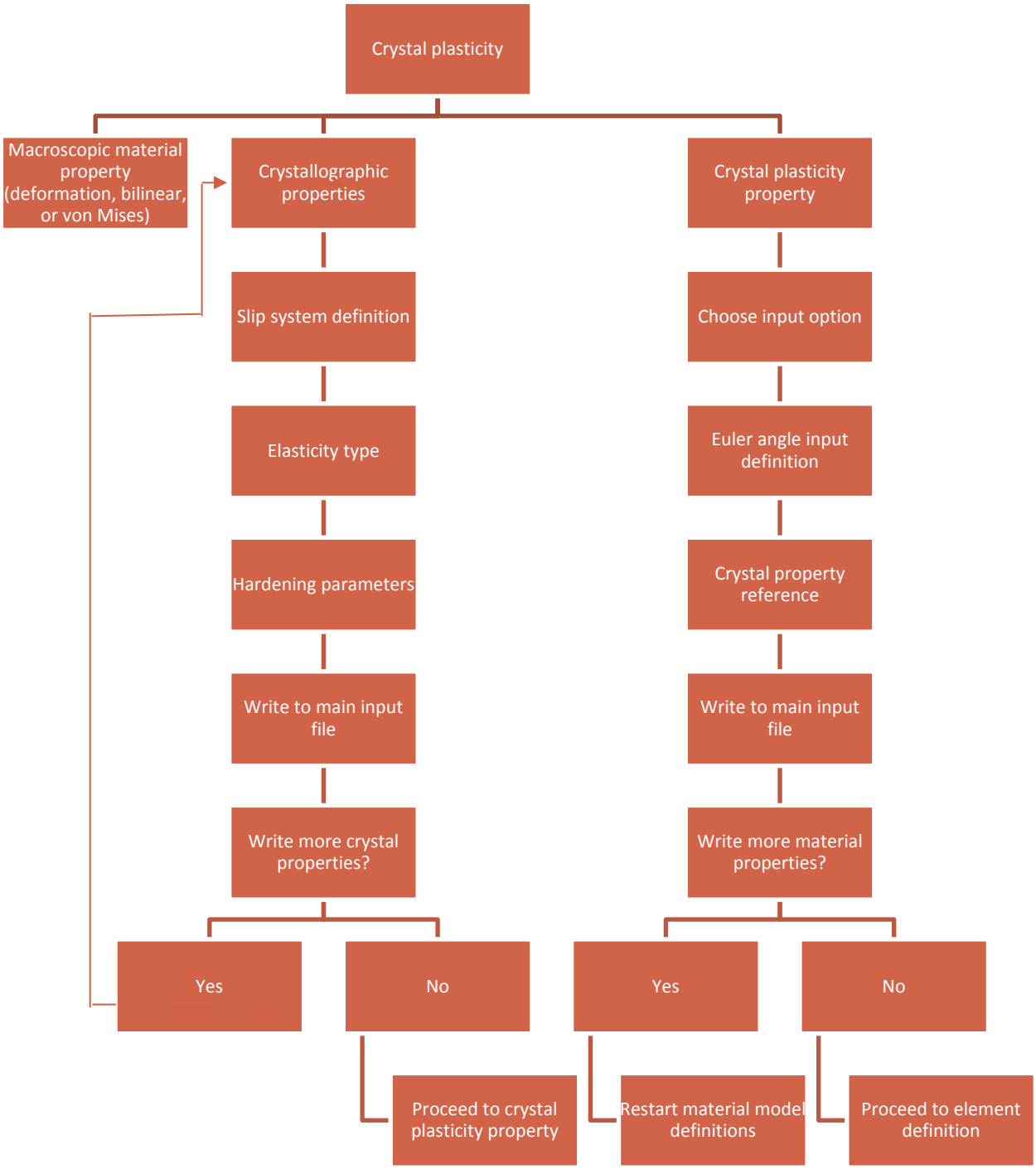

Figure 19. Crystal plasticity input process

When using Voce hardening, the user can specify an alternative mode where the reference strain rate is available to define. The user will be prompted to turn it off. If so, the reference strain rate

does not need to be defined. Otherwise, the user will be prompted to do so. This hardening type can be simplified to two hardening types: constant linear hardening and no hardening. In the constant linear hardening, the work hardening saturation strength (tau_v) must be significantly large such that the ratio of the shear stress on the crystal (tau_w) to tau_v approaches zero.

For user-defined crystal plasticity models, the user simply needs to enter the user properties and the user crystal properties. It is not required that the user use all of the properties. The user-defined hardening properties are not for a UMAT. Both sets of values belong to the crystal definition. This function is not recommended for users without existing knowledge on the implementation of UMATs.

Table 5. Crystal plasticity model input options for a single crystal

Property	Default
Angle convention	kocks
Associated crystal properties	1
Number of TBH aggregate crystals	1
Angle Type	Degrees
Orientation input	Single <Euler angles>
Density	0.0

Table 6. Crystal plasticity model input options for a polycrystal of a single crystal definition

Property	Default
Angle convention	kocks
Associated Crystal properties	1
Number of TBH crystal aggregates	1
Angle Type	Degrees
Orientation Input	File <filename>
Density	0.0

Table 7. Crystal plasticity model input options or a polycrystal of a multiple crystal definitions

Property	Default
Angle Convention	kocks
Associated crystal properties	File
Number of TBH crystal aggregates	1
Angle Type	Degreees
Orientation input	File <filename>
Density	0.0

3.2.5 Conclusion

Users can define an up to 500 materials in a single input file [1]. At the completion of each section of the material module, the user will be prompted to indicate whether all definitions belonging to the section have been written. In other words, all stress-strain curves must be defined before a material model, and all functionally graded material properties must be defined after the material model. In any other order, Warp3D does not recognize the input and will cause read errors to the main input file.

3.3 Elements

The parameters displayed to the user are determined from the element types found from parsing the mesh file. The default parameters are displayed to the user, along with the non-default values, and then the user can choose to keep the default values or change them. Each parameter, displayed in Table 8, can be change individually. The stress and strain components calculated during the analysis are determined here. The values can be used to split a single type of elements into multiple groups. For this option, the user must specify the number of groups to split the elements into. This option is useful for a model with multiple materials or different stress-strain output needs. If a single set of element parameters is defined, all element will automatically be assigned to those parameters. Otherwise, the user must define the element set that the parameters applies to in the same fashion as the functionally graded regions. Figure 20 details the process to generating the element parameters.

Table 8. Element Parameters

Parameter	Default	Element Type
Geometric formulation	Linear	All
Material	-	All
Order of integration	-	All
Stress-strain output	Minimal	All
Output location on element	Gaussian points	All
$\bar{B}$ formulation	On	8-node brick

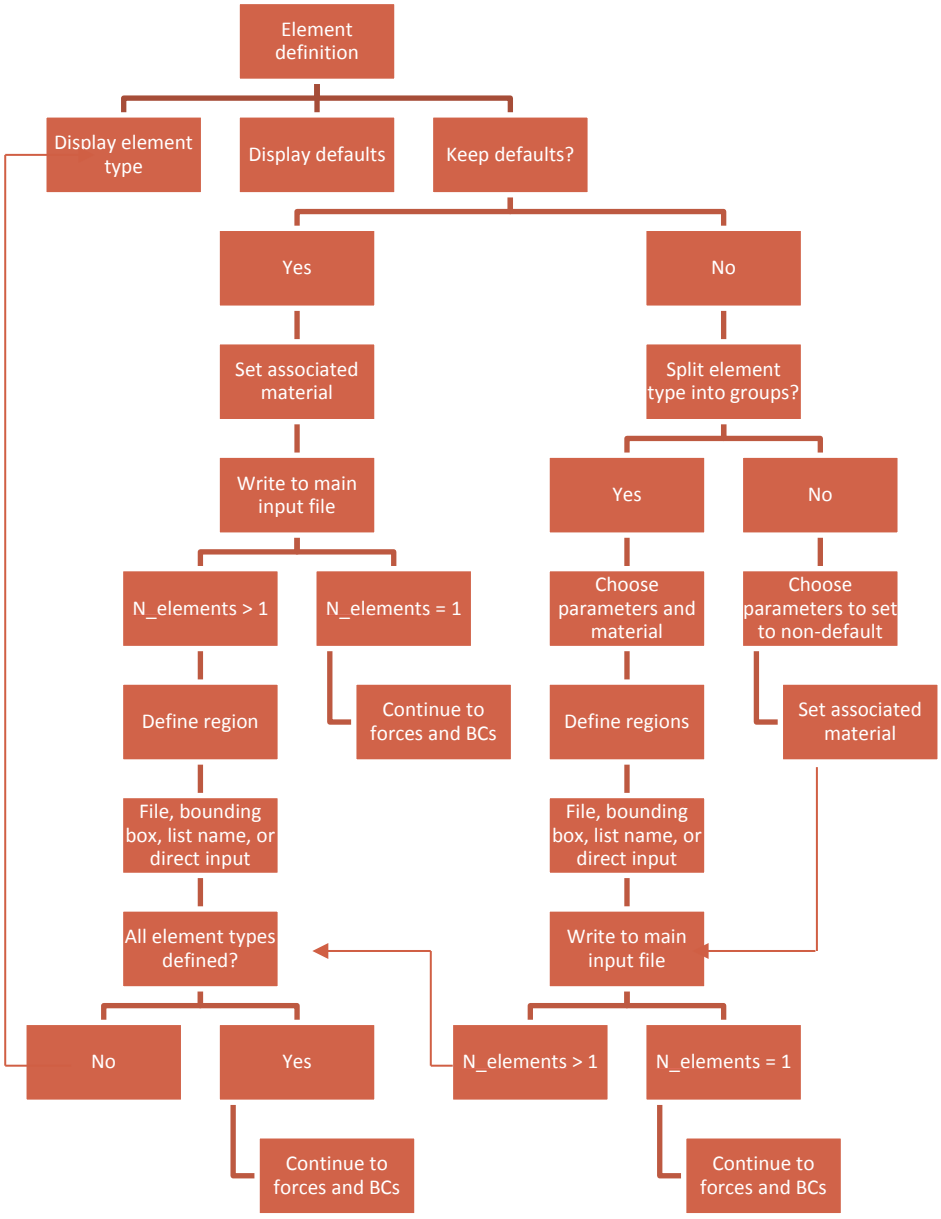

Figure 20. Element parameter definition process

Note that some properties are only available for certain element types. The $\bar{B}$ formulation is only available for the 8-node brick elements. This $\bar{B}$ formulation is the first derivative of the shape function of isoparametric elements (Equation 29), as shown in Equation 30, and represents dilatational strain. In Equation 29, ξ, η, and ζ are the isoparametric axes of the element, these axes are converted back to x, y, and z, and the partial derivatives on these axes make up the $\bar{B}$

vector. A stabilization factor can be introduced in the solution controls in order to reduce hourglass modes in the elements.

$$\boldsymbol{N} = N_i(\xi, \eta, \zeta)u_j$$

Equation 31. Shape function for isoparametric elements [32]

$$\boldsymbol{B} = \frac{\delta \boldsymbol{N}}{\delta \boldsymbol{x}}$$

Equation 32. **B** matrix as the derivative of the shape function [32]

The default value for the order of integration is also different for each element type. For the 8-node brick elements, the default order is 2x2x2, which generates 8 gaussian points on the element at each corner. For quadratic hexahedral elements, the default is a 9-point rule, which is similar to the 2x2x2 formulation, but includes an additional gaussian point at the origin. A 14-point rule is available for the 20-node elements which places gaussian points at each corner and on the faces. For linear tetrahedral elements, a 1-point rule is used where the only gaussian point is at the element center. For the quadratic, a 4-point rule is the default with the points at the vertices. An additional 5-point rule is available for the quadratic tetrahedral elements that combines the points from the 1-point and 4-point rules [1].

Additional non-default options exist for the geometric formulation, the stress-strain output location, and the stress-strain values. For each element type, the user can specify a nonlinear geometric formulation. The default output location for all elements is at the gaussian points. Additional output locations are for the element centers and at the nodes. For the stress-strain outputs, the minimal set contains 6 components, the von Mises stress and strain, the work density, and the material state variables. The full output includes the stress and strain invariants, the principal stresses and strains, and the direction cosines for both.

3.4 Boundary conditions and Applied forces

Boundary conditions, forces (or loading patterns), and loading step definitions are compiled into a single module (Figure 21). For the boundary condition and force sections of the module, the user can enter an unlimited number for each. For both, the user will need to specify the surface or

region where the force or boundary condition lies, and then the total value of the force or boundary condition. In general, these values will be zero for the boundary conditions unless the user is implementing a displacement control analysis, which will be discussed in more depth later. For the loading step definitions, all steps will fall under a single header, but an unlimited number of individual steps can be underneath. Each boundary condition and loading pattern is purely mechanical. There are no thermal components to this preprocessor.

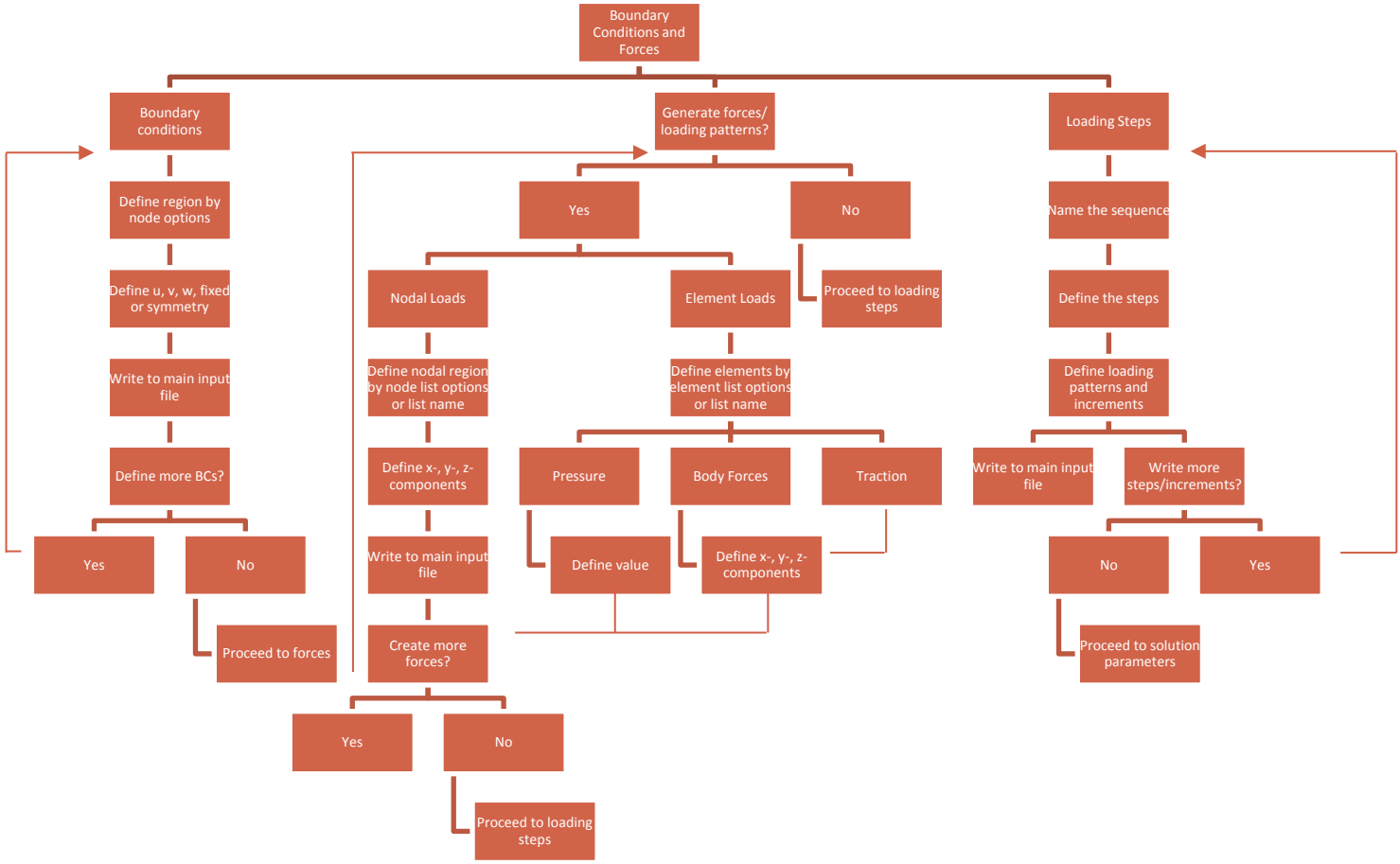

Figure 21. Boundary Conditions and Forces module process

3.4.1 Boundary conditions

Boundary conditions are input as the normalized displacement components in u, v, and w. Fixed and symmetry boundary conditions can also be specified. In Warp3D, boundary conditions are applied directly to nodes. The user will be first asked to define a surface on which to apply a

boundary condition, and then to define the displacement components. Surfaces can be defined using a plane, a list name, or a file containing the node set. All three components do not need to be included, as the there is no default boundary condition value.

If the user is creating a displacement control analysis model, the boundary condition at the surface should be 1.0 in the desired direction. The zero-displacement components do not need to be specified on the surface used to define the displacement control. The actual total displacement value will be defined by creating a loading increment where the applied load is defined as the constraints. The total displacement should be equal to the sum of the values at the increments.

3.4.2 Applied Forces

Forces are optional (if the user is conducting a displacement control analysis), and are called loading patterns in Warp3D. Each applied force must be assigned a name and a location – this can be accomplished by referring to a list name, using a file containing a node or element set, or a surface, which are processed in the same fashion as previously described. The user must then define the non-zero x, y, and z components of the applied load.

Applied forces can be nodal or element forces. Nodal forces must be defined as the value on a single node. This value is likely unknown to the user; therefore, to ease input, the user will be prompted for the total force value, and then the preprocessor will calculate the per-node value, assuming a uniform stress. Element loads come in three forms: pressure, traction, and body forces. Input options are the same as nodal forces, and the elements are found in the same fashion as described for element sets. Pressure is a value per element that is normal to the defined surface, while body forces and tractions are defined by the x, y, and z components. A face number will be calculated by the preprocessor is it is unknown by the user.

Non-uniform loading of any kind is not supported by the preprocessor or by Warp3D in a single force definition. This kind of load would require the user to do a Reimann sum approximation over the elements or nodes using as many loading patterns as desired. The incrementation can be done together.

3.4.3 Loading increments

Loading increments are generated by specifying the step(s), the loading pattern(s), and the increment value(s). The user will be directed to enter the steps for which the user would like to define an increment, then to indicate the loading parameters and the increment for the steps. The loading increments, or the sequence, is defined by a single name, and the steps will have loading patterns and increments are listed below the header. For example, for a fatigue sequence with an initial overload applied, the first step to the peak load refer to the loading pattern and increment than the unloading and subsequent loading and unloading of the fatigue sequence.

To simplify this type of loading sequence, a unique abbreviation form can be used. If total number of loading cycles is 1000 (excluding the overload), the overload and unload can be defined at steps 1 and 2, respectively. Then, the loading portions of the cycles can be defined as steps 3-2002 by 2, and the unloading portions as steps 4-2002 by 2. This sort of abbreviation will define all of the loading portions of the cycles as the odd-numbered steps, and the unloading portions as the even-numbered steps. Step definitions do not have to be perfectly sequential. If the overload was in the middle of the loading sequence, the loading portion of the cycles could be defined as steps 1-1000 by 2 1003-2002 by 2. An example of the loading sequence input is shown in Figure 22.

```
loading creep
    nonlinear
    step 1-10 constraints 0.0003
    step 11-100 constraints 1.0e-15
```

Figure 22. Example of loading increments for a displacement control test from the Warp3D crystal plasticity example [1]

3.5 Solution Controls and Output Commands

The solution controls, listed in Table 9 with the default values, is a short list of common parameters for a Warp3D analysis. The solution technique, however, does not have a default parameter. There are 7 solution techniques outlined in **Error! Reference source not found.**. E ach solution technique is defined by three parts: the thread type, the solution method, and whether the solution is symmetric or not. For the convergence tests, there are six types of tolerances related to displacement tolerances and residual force tolerances (outlined in Table 10).

This is described by Equation 31, where **P** is the known force vector, M is the mass matrix, $\ddot{\boldsymbol{u}}$ is the acceleration, and **I** is the nodal forces calculated from the stress [1]. For a model implementing crystal plasticity, the displacement extrapolation function is automatically turned off, and if for linear brick elements are used for the model, the $\bar{B}$ stabilization factor is set to zero.

$$\boldsymbol{Res} = (\boldsymbol{P} - \boldsymbol{M}\ddot{\boldsymbol{u}}) - \boldsymbol{I} \quad (31)$$

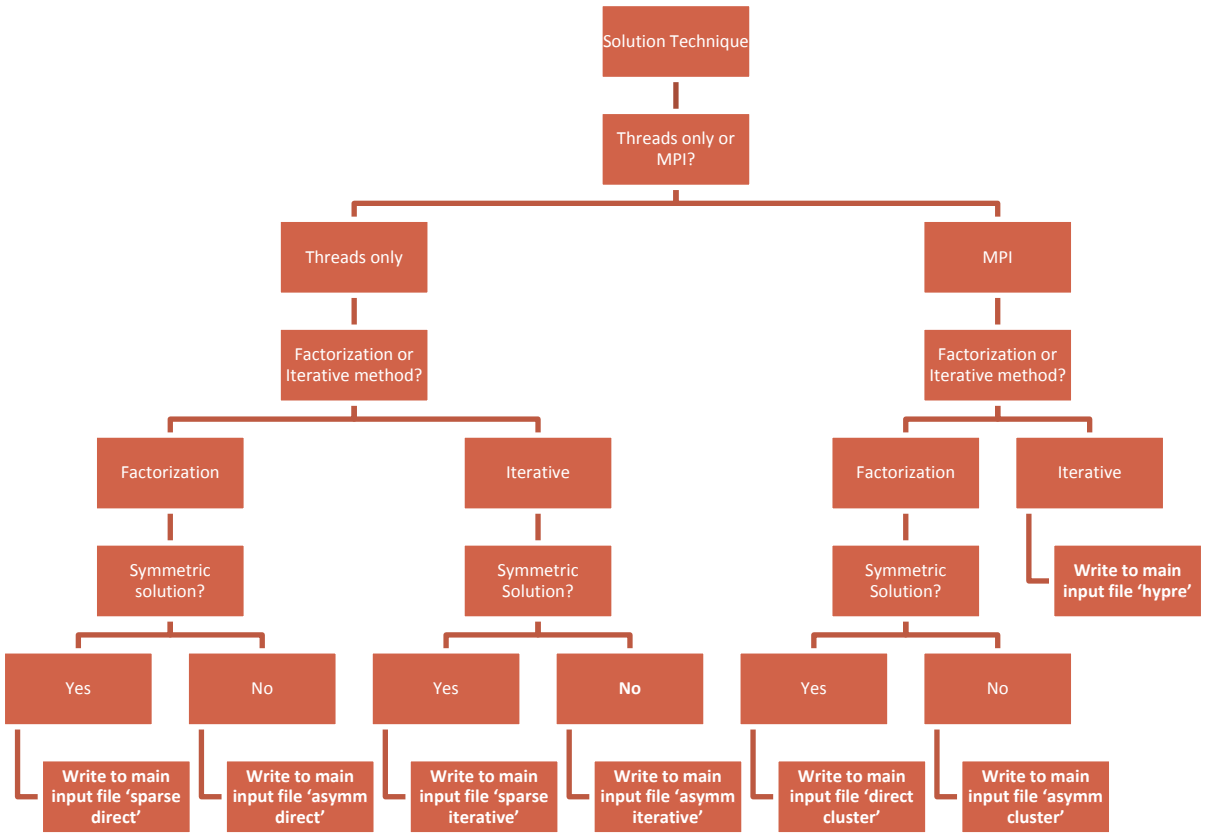

Figure 23. Flow chart for solution technique

Table 9. Solution parameters with descriptions and defaults

Parameter	Description	Default
Thread Type	Using only processor threads or threads + MPI	-
Convergence Test	Convergence tolerances for displacement and residual force	Off
Divergence Check	Check for diverging solution are 3 iterations (2 for strict check)	Off
$\bar{B}$ stabilization factor	Suppression of hourglass modes in 8-node elements	0.0

Table 9. Solution parameters with description and defaults (continued)

Extrapolation function	Extrapolation of displacement at iteration n+1	On
Batch Messages	Displays messages about solution progress	Off
Material Messages	Stress update status messages	Off
Maximum Iterations	Maximum number of iterations per step	5
Minimum Iterations	Minimum number of iterations per step	1
Trace Solution	All messages about solution progress, including strain and stress updates, and convergence test results	Off
Continue for Non-converging solution	Whether to continue analysis for a solution that appears to be diverging	off

Table 10. Convergence test options

Convergence Tests	Default
Normal displacement	0.01
Normal residual stress	0.01
Absolute maximum displacement	1E-6
Absolute maximum residual stress	1E-6
Maximum displacement	0.01
Maximum residual stress	0.01

All parameters except for the convergence test can be used only once. The user can specify all six convergence tests and assign individual tolerance values to each. The divergence check has two options: the first uses Warp3D's default check wherein if three iterations result in a diverging solution, and a stricter version where the solution is assumed to be diverging after two consecutive iterations with a non-converging solution.

In order to initiate an analysis, the user must specify the loading sequence and steps for which displacements should be calculated. Additionally, the user can specify that stress, strain, forces, and displacements can be recorded to the analysis history file. Each of these comes six format types (Table 11). The user can specify which for which steps the values should be recorded and which elements or nodes to record them for. If the numbers are unknown but a region is known, the user can use the same input methods as the stated previously.

Table 11. Output format types and descriptions

Type	Description
Default	Data will fit on a standard piece of paper
Wide	Data is on lines with 132 columns
Eformat	Data is printed in f12.5 format
Precision	Data is printed in f26.16 format; can be used with eformat and default
No header	Removes identifying labels
Totals only	Summed reactions and displacements rather than components

By default, the preprocessor outputs seven flat text files for conversion to Exodus-II files, which are compatible with Paraview: a patran-formatted flat text file of the model, flat text files for the displacement and forces at the nodes, and flat text files for the stresses and strain at the nodes and elements. For crystal plasticity models, there are seven types of output options outlines in Table 12. Each of these can be written in binary or formatted files. The crystal number from the crystallographic property definition can be used to pick different outputs for different crystal types, or the user can enter zero to apply the output options to all crystal types. At this point, the preprocessor adds a stop command to the end of the main input file and the file is ready to run in Warp3D.

Table 12. Output options for crystal plasticity models.

Type	Description
1	Euler angles only
2	Euler angles and creep constants
3	Euler Angles, lattice strains, slip activity
4	Nye tensor, Euler angles, slip activity
5	All values not tied to the crystal
6	Euler angles, creep constants, hardening values
7	All of the above

3.6 Running in Warp3d and Visualizing in Paraview

The input file can be run using the command prompt in Windows or the terminal in Linux. If using Windows, the user must assign the solver location to a variable, the directory of the input file to be run, and the number of MKL and OMP threads to use. Detailed directions are provided in the Warp3D package under run_windows_64. When the analysis is complete, at least 8 new files will be written to the directory specified: the analysis log file and the 7 flat text files.

If only the analysis log file was been written, that indicated there is an error in the input file that the user must correct. If there are several errors that require the user to completely rewrite the section, there is an additional correction code *correctInput* to fix the sections. This function will read through the input file, and the user can choose from the headers materials, elements, constraints, forces, or solution controls. A list of the features under those headers will be displayed and the user can select which features to change or add. Information is added using the modules from the preprocessor. The code will then rewrite the corrected information to a new input file with the old name plus a suffix of the user's choosing. The code's process is described in Figure 24.

If the analysis does complete, then the flat files can be used to generate a Paraview-compatible Exodus II file using a python code included in the Warp3D package, *warp3d2exii*. The exodus II file format is a format for pre- and postprocessing of data. This file format is made up to the geometric information and the nodal and element results, which are stored by time step [3]. This code will prompt the user to choose a name for the .exo file, the type of file and location of the flat text model file, and the directory of the results files. If time stamps are included in the data, those can be exported with the files. This code will convert the information into a single Exodus II file that is ready to view in Paraview.

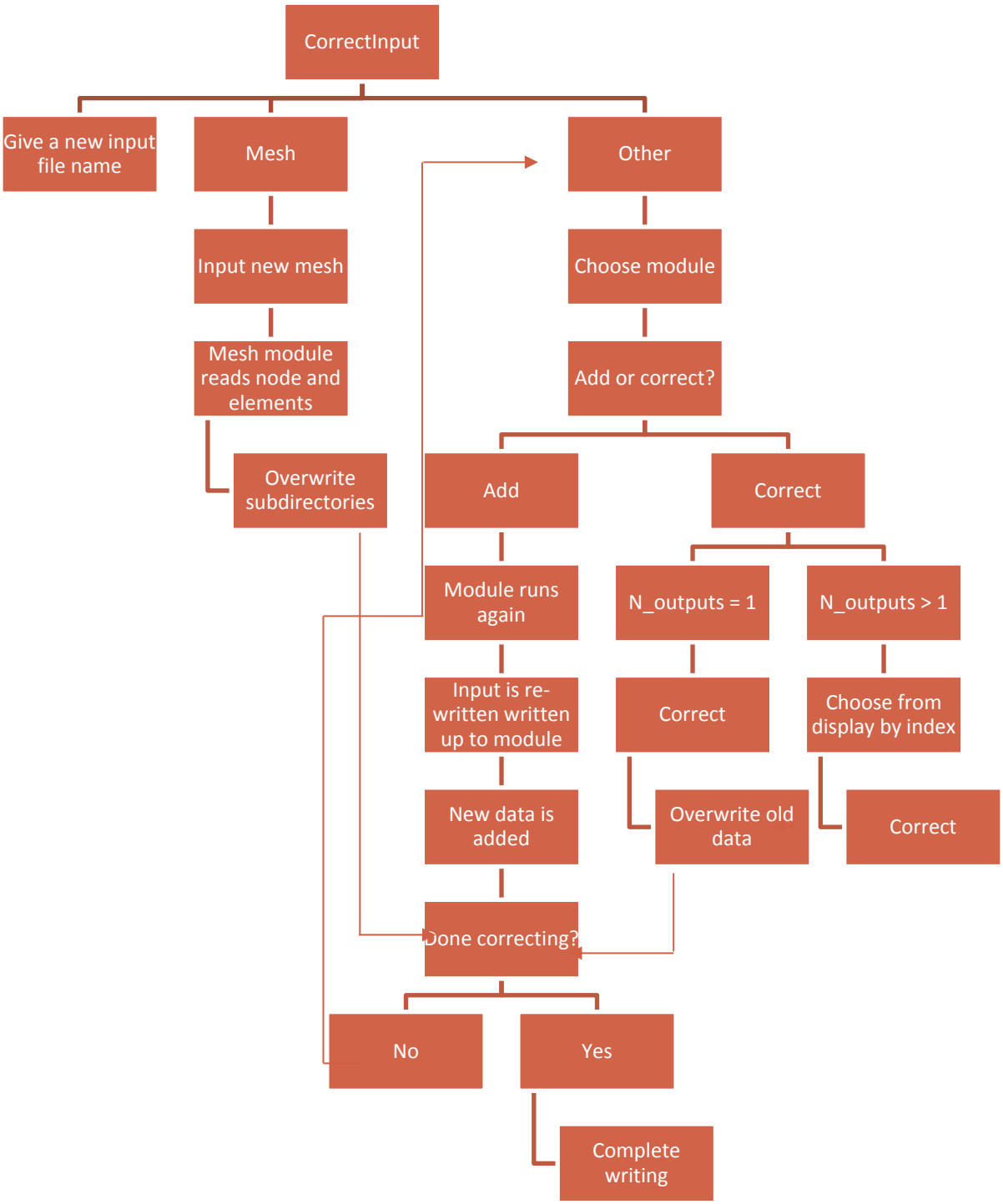

Figure 24. Process for correction an input file with errors

4. USE CASES

Five use cases were generated using the preprocessing framework using three geometric models: a cantilever beam, a cube, and a pallet (Figure 25). Using the cantilever beam and the pallet, only one use case was generated. The remaining three were generated using the cube: a displacement control test, another displacement control test with the material defined with a user-defined stress-strain curve, and a crystal plasticity model. Each mesh was generated with an Abaqus format. For all use cases, four processor threads were allocated to the analysis. Force, displacement, stress, and strain was outputted as flat-text from the analysis. For visualization in Paraview [29], the flat text files were converted to an Exodus II file [3] using the code from the Warp3D package, *warp3d2exii* [1]. Each model was run on a Dell precision 7810 desktop with an Intel Xeon E5-2623 processor and 32GB or RAM.

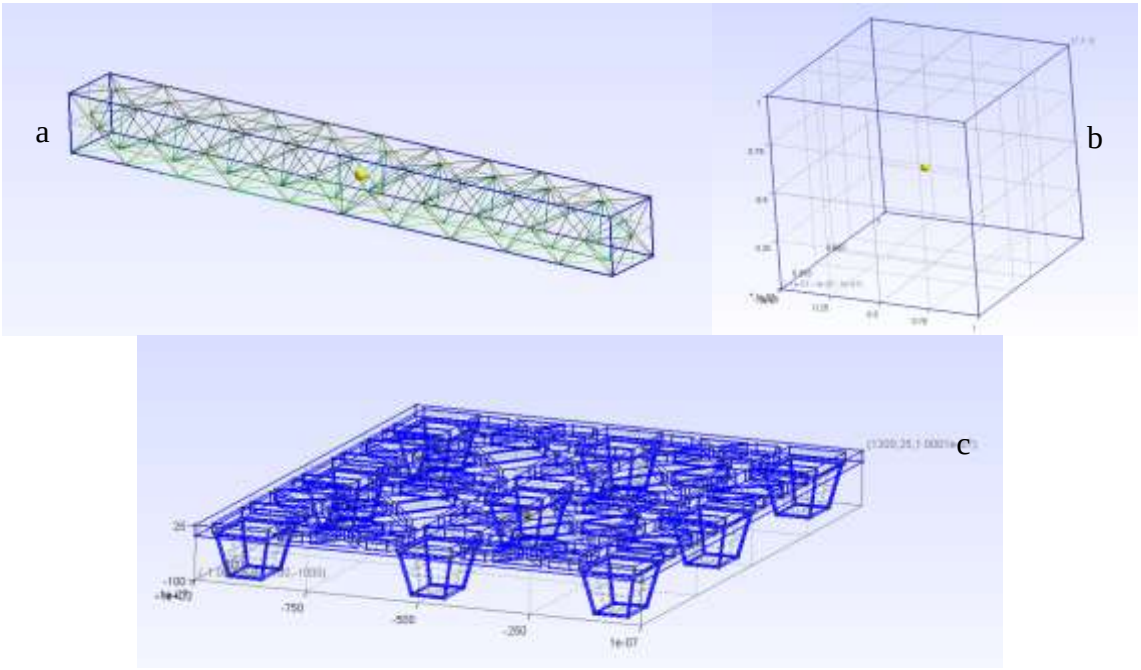

Figure 25. Gmsh-generated or imported geometric models. The pallet (c) is imported from a *.step* file

4.1 Cantilever Beam

The following functions were used to generate the cantilever beam model:

- Linear tetrahedral elements generated from Gmsh
- Deformation material model as described in Section 3.2.2.1
- Plane surface definition as described in Section 3.1 for boundary condition with fixed value
- Nodal forces on an edge inputted with direct input as described in Section 3.1

For this model, the geometry was generated and mesh directly in Gmsh. The cross-section of the beam is 1x1 mm, with a length of 10mm, thus is sufficiently long for the results to be verified using Euler beam theory in Equation 32 (Figure 1a). The unit system for this use case is mm-kN-MPa. For this model, the left end is fixed, and a total load of 2 kN along the upper right edge in the positive y-direction (Figure 26).

$$\sigma = \frac{My}{I_x} \qquad (32)$$

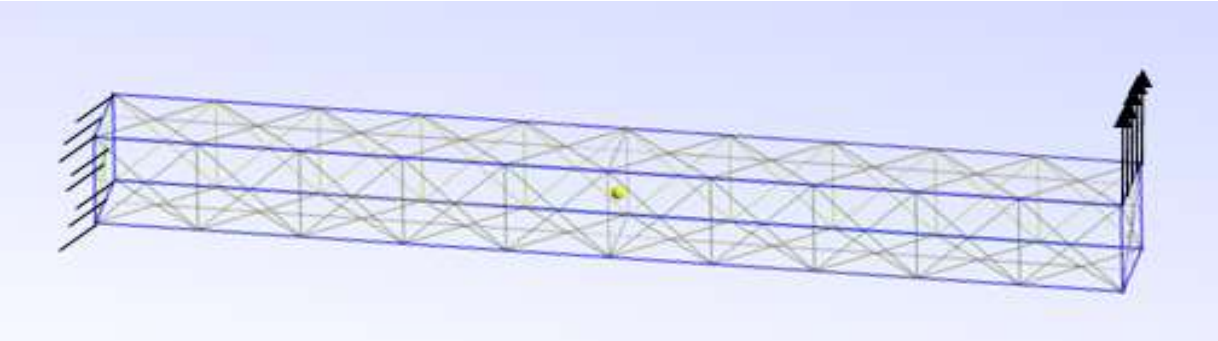

Figure 26. Cantilever beam mesh with linear tetrahedral elements; boundary conditions shown and applied force shown.

Linear tetrahedral elements were used; a total of 86 nodes and 204 elements were created. When importing the mesh into the preprocessor, no node or element sets from the mesh file were kept as none were created in Gmsh. Because the mesh is relatively coarse, on the right face there are only two nodes at the upper edge. These node numbers were located manually. In other words, since the number of elements was small, the mesh file exported from Gmsh was opened, and the node IDs on the upper right edge were noted and input as a list. In the preprocessor, the default parameters for the element type were used. That is, the geometric formulation is linear, the order of integration is the 1-point rule, and the stress and strain output is the minimal set at gaussian points. This input example is shown in Figure 27.

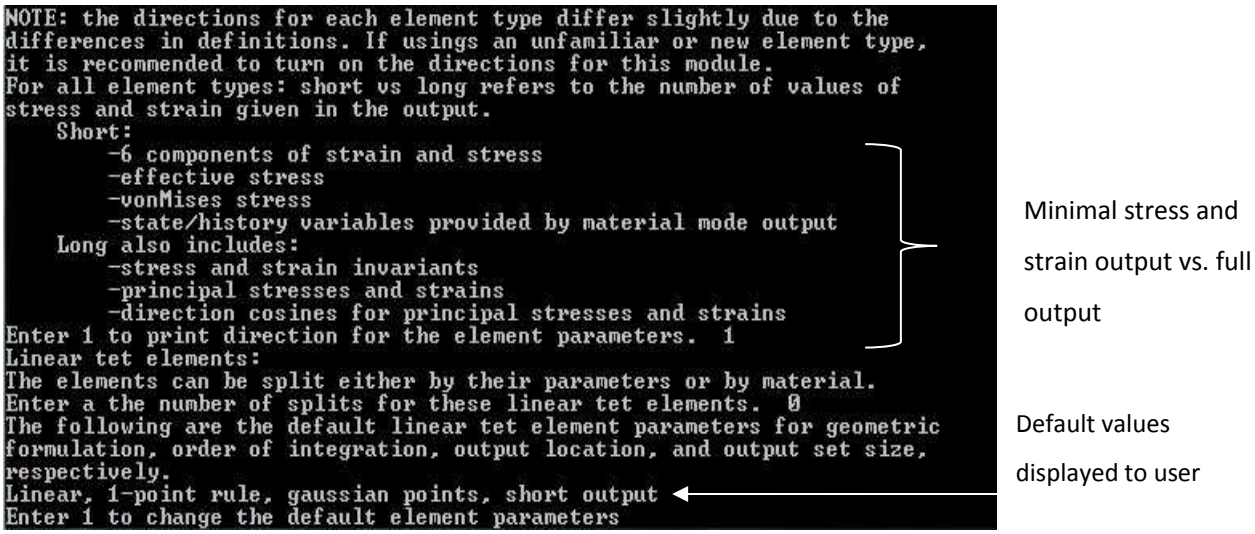

Figure 27. Preprocessor interface during the element module

The material model used for this use case is the deformation model with 300WA steel properties used [33]. The elastic modulus is 205x103 MPa, the Poisson's ratio is 0.33, and the yield stress is 450 MPa. The material density is set to zero, and a power-law exponent of 5.0 is used.

As stated earlier, this boundary condition for this model is a fixed condition at the left side of the beam. In the preprocessor, a plane definition is used, where the plane is $x = 0$. The coefficients are inputted as 1 0 0 0, which corresponds to a plane equation of $1x + 0y + 0z + 0 = 0$, which simplified to $x = 0$. For the applied forces, nodal forces were used. As this mesh is relatively coarse, only two nodes are located on the upper right edge that belong to the same element, this is equivalent to a uniform load over the edge. These node ID numbers were inputted directly to the preprocessor and translated to the main input file.

For the cantilever model, the sparse direct solution technique was used – this uses the factorization method on the threads for a symmetric solution. The standard divergence check in Warp3D was used, and a normal displacement convergence test is a relative tolerance of 1% was used. Given the simplicity of the model, is was expected that this strict convergence test will maintain accuracy of the solution without interfering with the model's ability to converge. The maximum number of iterations per steps was set to 20, while the minimum is one. Batch messages, material messages,

and the trace solution function were turned off. Finally, the output location was chosen to be the lower left edge, as this location is expected to have the highest tensile stress.

Analysis for this model completes in approximately 4 seconds; the conversion to the Exodus II file format takes approximately the same amount of time. The stress-strain curve is shown in Figure 28 and the contour plot in Figure 29 show the results of the analysis. The results of the stress show that the stress remained significantly lower than the yield stress, thus this analysis remained well within the elastic region of the curve. This is also seen in the contour plot, which has the highest stress in the loading direction is at the bottom left, and the maximum compressive stress on the upper left. The stress strain curve shows a maximum effective stress of 129 MPa, which Euler beam theory gives a maximum stress of 120 MPa, resulting in a difference of 7%.

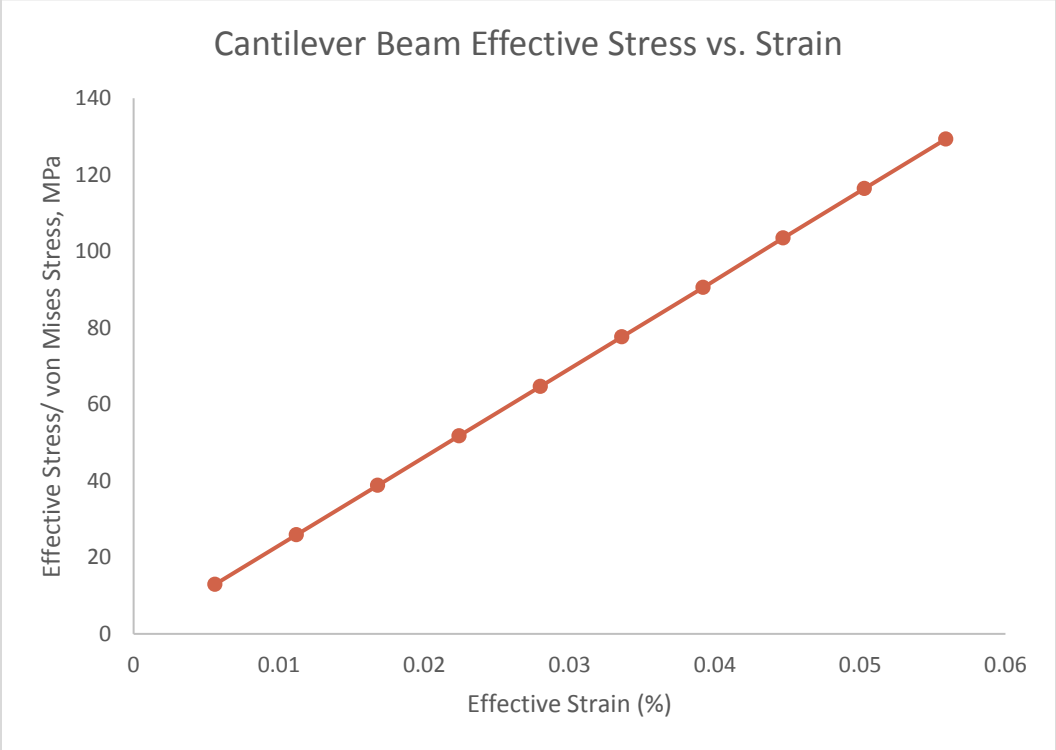

Figure 28. Stress-strain curve derived from analysis of the cantilever beam

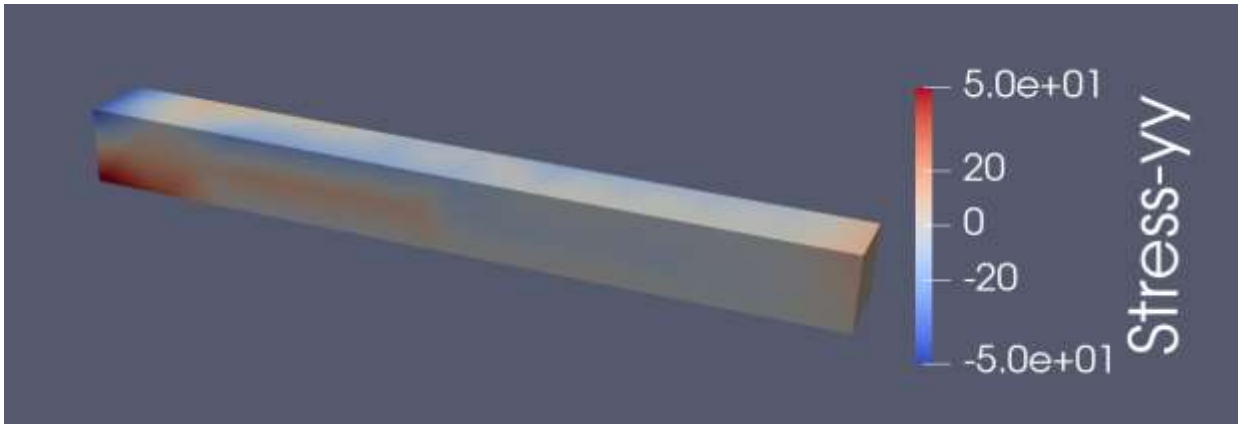

Figure 29. Stress contour of the cantilever beam in the loading direction in MPa.

4.2 Cube

Three use cases were generated using the cube: displacement control, stress-strain curve material model, and crystal plasticity model. Each model is a 1x1x1 mm cube and implements a linear hexahedral mesh; it has a total of 27 elements and 64 nodes. To test convergence, the mesh was refined to a size of 273 elements. In the preprocessor, the output location of the stress and strain was changed from the gaussian points to the nodes. The remaining parameters – linear geometric formulation, the order of integration as 2x2x2, and a minimal output of stress and strain values. Each model uses an applied displacement in lieu of an applied force.

4.2.1 Displacement control

The displacement control test uses the bilinear material model to model 300 WA steel [33]. The hardening modulus was calculated directly from the hardening modulus to be 0.36, and the model uses isotropic hardening, as described in Section 3.2.2.2.

For this use case, a fixed boundary condition was used on the plane $z = 0$, and roller constraints were applied to planes $x = 0$ and $y = 0$. The displacement was applied for the plane $z = 1$ (Figure 30). The loading sequence was set such that the total displacement was 0.025 mm, which for this model, is equivalent to 2.5% strain.

For this model, as with the cantilever beam, the sparse direct solution technique was used. The maximum iterations per step was set to 10, and the minimum to 1. The divergence check was turned off, and normal residual force convergence test was used with a tolerance of 1%. Batch messages and material messages were turned off, and the trace solution function remained on. The analysis for this solution completes in less than 1 second; the conversion to the Exodus II file format takes approximately 5 seconds. Stresses and strains were calculated at element 1 on the $x = 0.33$ face and plotted on Figure 31. This face is internal to the cube and away from the boundary conditions. The contour stress plot for stresses in the loading direction are shown in Figure 32. The high stresses around the bottom edge of the cube appear to be artifacts. Given that there are multiple boundary conditions at those locations, the results there may not be reliable. Excluding those locations, the maximum stress on the contour plot appears to be near 500 MPa. This is shown in the stress-strain curve as well. As expected, the curve extends just beyond the yield into the plastic region.

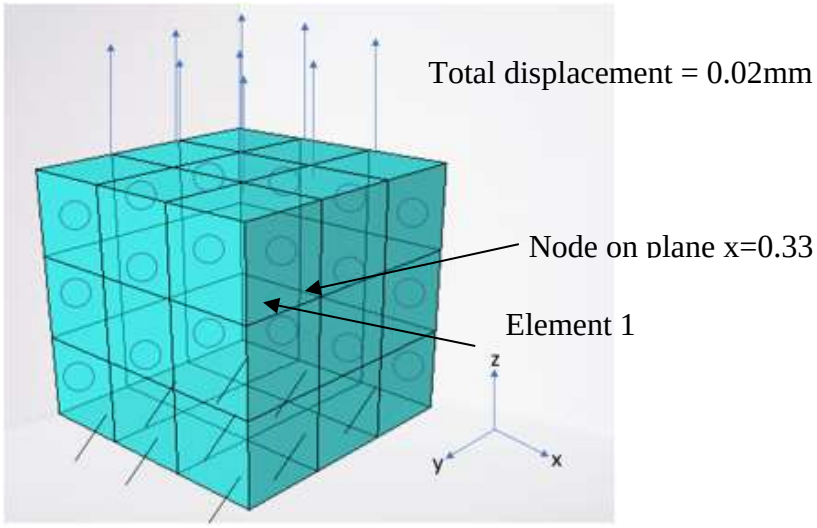

Figure 30. Displacement control boundary conditions and applied displacement

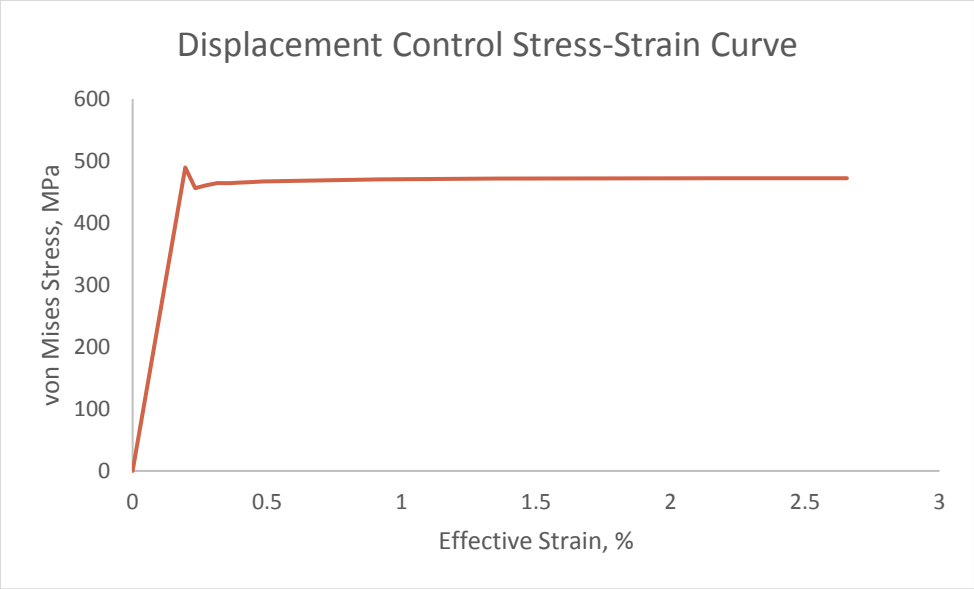

Figure 31. Stress-strain results from displacement control use case

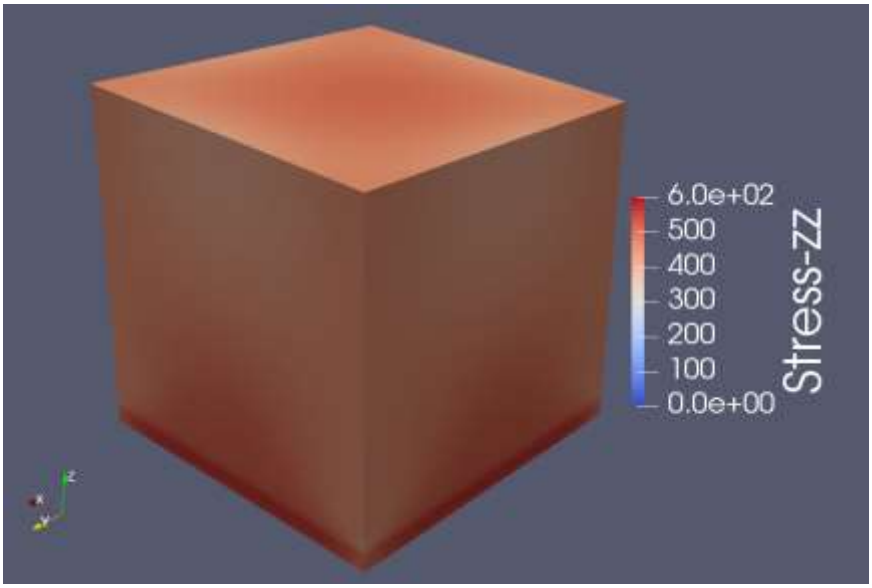

Figure 32. Stress contour plot in the loading direction for the displacement use case in MPa

4.2.2 Stress-strain Curve material definition

This model uses a user-defined stress-strain curve, along with the elastic modulus and Poisson's ratio to define the material model. This was accomplished using the von Mises material model approach 3, as described in Section 3.2.2.3. The stress-strain curve, shown in Figure 33, extends

to 2.5% with a proportional limit of approximately 0.6% strain. The values inputted are arbitrary and is expressly used to test the usage of stress-strain curves to define a material. It doesn't intentionally match the behavior of any material. The elastic modulus is defined at 30 GPa and the Poisson's ratio as 0.33. The density of the material is set to zero.

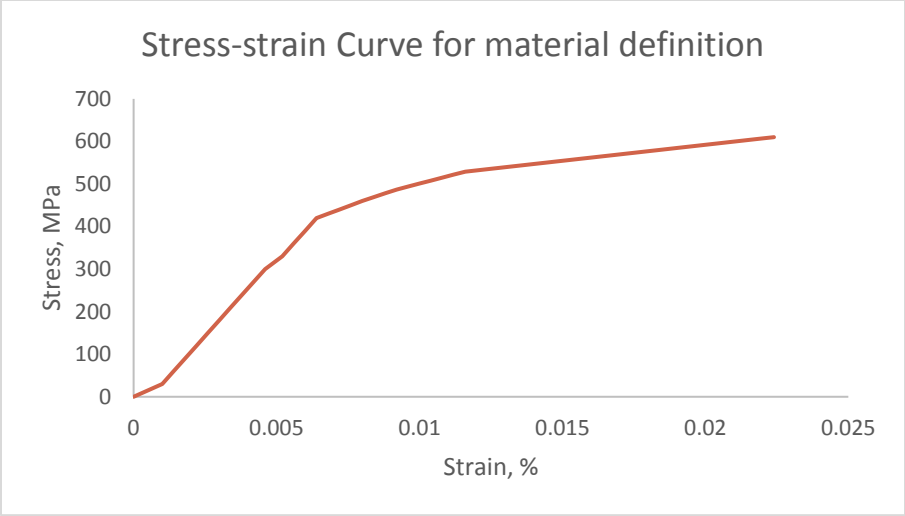

Figure 33. Stress-strain curve used to define the von Mises material model

This model uses the refined mesh with 273 elements and the default parameters for the linear hexahedral elements: linear geometric formulation, 2x2x2 order of integration, stress and strain output at the gaussian points, the minimum output of stress and strain, and the $\bar{B}$ formulation is turned on. This allows the model to consider the dilatational strain and along with the stabilization factor set to zero, reduces hourglass modes in the element.

For this use case, only the plane $z = 0$ is fixed (Figure 34). The displacement control was set to a total strain of 4%, to elicit a perfectly plastic response beyond the stress-strain curve definition. The contour plot, shown in Figure 35, shows a maximum stress of 700 MPa, which may be an artifact due to the boundary conditions. Away from the boundary condition, the maximum stress appears to be approximate 600 MPa.

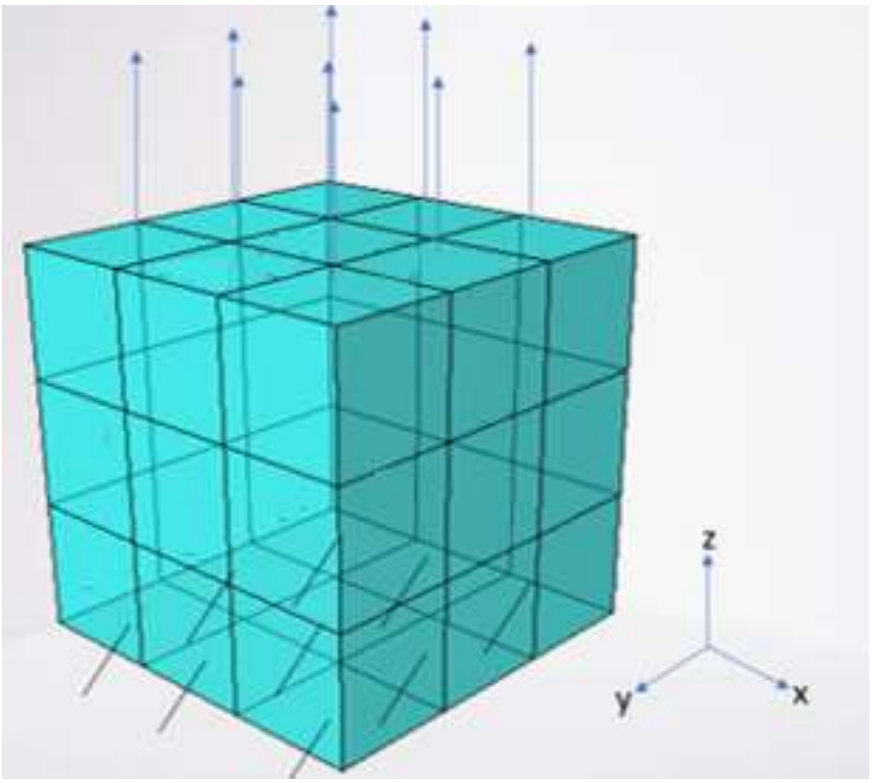

Figure 34. Boundary conditions and applied displacement for stress-strain curve use case

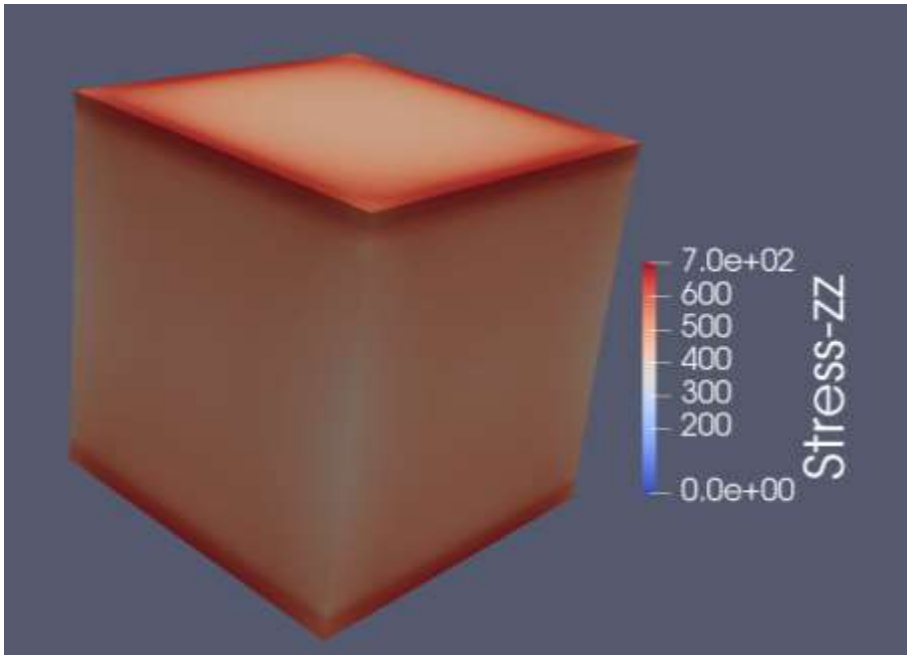

Figure 35. Stress contour plot in loading direction for stress-strain use case

4.2.3 Crystal Plasticity

With this use case, the mesh size returns to that of the displacement control model, with boundary conditions as described in Figure 30. For this model, the element parameters are all default except for the output location, which is changed to the center of the elements. The macroscopic material model is a deformation model with an elastic modulus of 205 GPa, a Poisson's ratio 0.33, and a power-law exponent of 20, as described in Section 3.2.2.1. The total strain is set to 10%.

In the crystal definition, the alternative mode of the Voce model for constant linear hardening. This alternate mode allows the user to set the reference strain rate to a desired value, whereas the general use Voce model does not [1]. This model uses a work hardening saturation strength of 5500 MPa, a hardening exponent of 20, and a voce parameter of 1.0, as described in Section 3.2.4 and Equation 8 in Section 2.4. An isotropic stiffness matrix was defined with an elastic modulus of 205 GPa and a Poisson's ratio of 0.333. The slip system is an FCC crystal. The yield strength of the crystal was defined at 900 MPa. The reference strain rate was set to 1, and the Newton-Raphson tolerance was set to 1E-10. The yield strength and elastic modulus were verified by the stress-strain results in Figure 36. This data is the average von Mises stress over all nodal data, which was extrapolated from the element centers, and the effective strain, also averaged over nodal data.

A total of 999 random orientations were divided evenly over the 27 elements to create 27 homogenized polycrystal aggregates in the model, all related to the FCC crystal defined above. Each element contains 37 orientations. The Euler angles follow a Bunge notation, and contained in a *.ang* file that assigns the random orientations to the elements. The displacement was divided into 1000 uniform steps. Stress and strain measurements were taken every 50 steps.

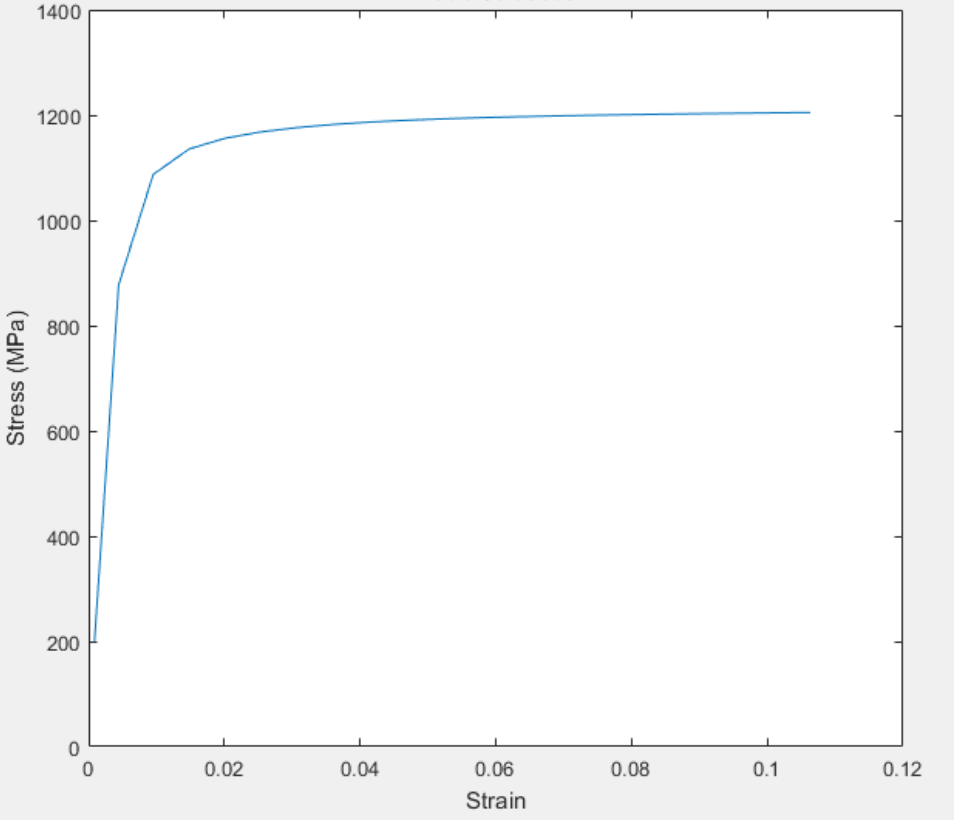

Figure 36. Stress-strain curve averaged over all nodal data.

This use case also uses the sparse direct solution technique; the maximum number of iterations per step is set to 20, and the minimum is set to 2. The iterative displacement relative tolerance is set to 1%. This strict displacement tolerance ensures that the solution is converging to a reasonable solution. Batch messages and material messages are turned off, the solution extrapolation function is turned off, and the $\bar{B}$ stabilization factor is set to 1.0 to reduce hourglass modes and volumetric changes. In addition to the force, displacement, stress, and strain flat text files, a flat text file was also generated for the Euler angles to plot the texture evolution.

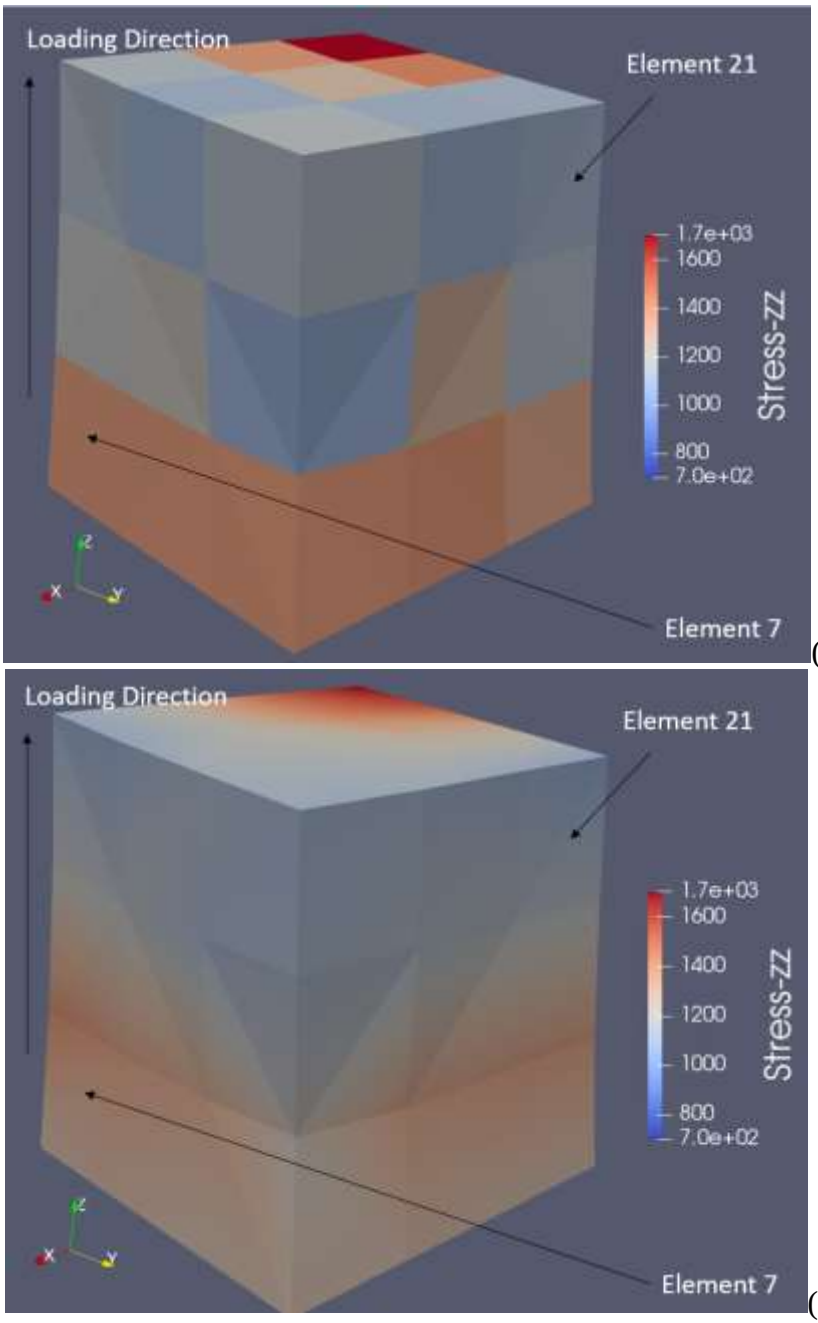

Figure 37. Resultant stress (a) from element centers and (b) extrapolated from the element centers to the nodal points

A Taylor factor was calculated for each element and the entire model using a Matlab code. The Taylor factor for the entire model was calculated to 3.061, with a maximum of 3.22. Two elements in particular, elements 7 and 21, were noted for their difference in stress, as shown in Figure 37. These two elements showed some of the greatest differences from the expected value for a random

texture of 3.06: element 7 had a Taylor factor of 3.22 and element 21 had a Taylor factor of 2.88 in the deformed state. Additionally, these elements had the greatest change in the Taylor factor from the undeformed state. In the undeformed state, the Taylor factors were 3.11 and 2.89, respectively.

The maximum stress shown in the contour plots in Figure 39 is 1.7 GPa, and the minimum as approximately 1.1GPa. The maximum stress appears to be an outlier: the majority of the stress values appear to be between 1GPa and 1400 GPa. Additionally, the 10% strain shows severe deformation on the unconstrained faces, x=1 and y=1. Given that the mesh of this model is relatively coarse, this may have influenced the stress results to be excessively high around the yield point. A more refined mesh is required to verify this, without discretizing the grains, so as to maintain the model formulation.

4.3 Pallet

The following functions were used to generate the pallet model:

- Quadratic tetrahedral elements generated from Gmsh
- Deformation material model
- plane surface definition for boundary condition with fixed value
- nodal forces inputted on a plane using a plane surface input

For this model, this pallet design was pulled from the GradCad [35] library as a Solidworks [12] part, and converted into .step file. This file was imported into Gmsh, meshed using the 3D mesh option to generate tetrahedral elements, and then the element order was changed to a second order, or quadratic element. The maximum element size was set to 60mm. This mesh was exported as an Abaqus file Because of the complexity of the geometry, the mesh generation in Gmsh takes over one hour, and thus it is recommended to use another mesh generator software. The mesh is shown in Figure 38. The model uses a mm-kN-MPa unit system, and has dimensions of approximately 1200x1200x125 mm.

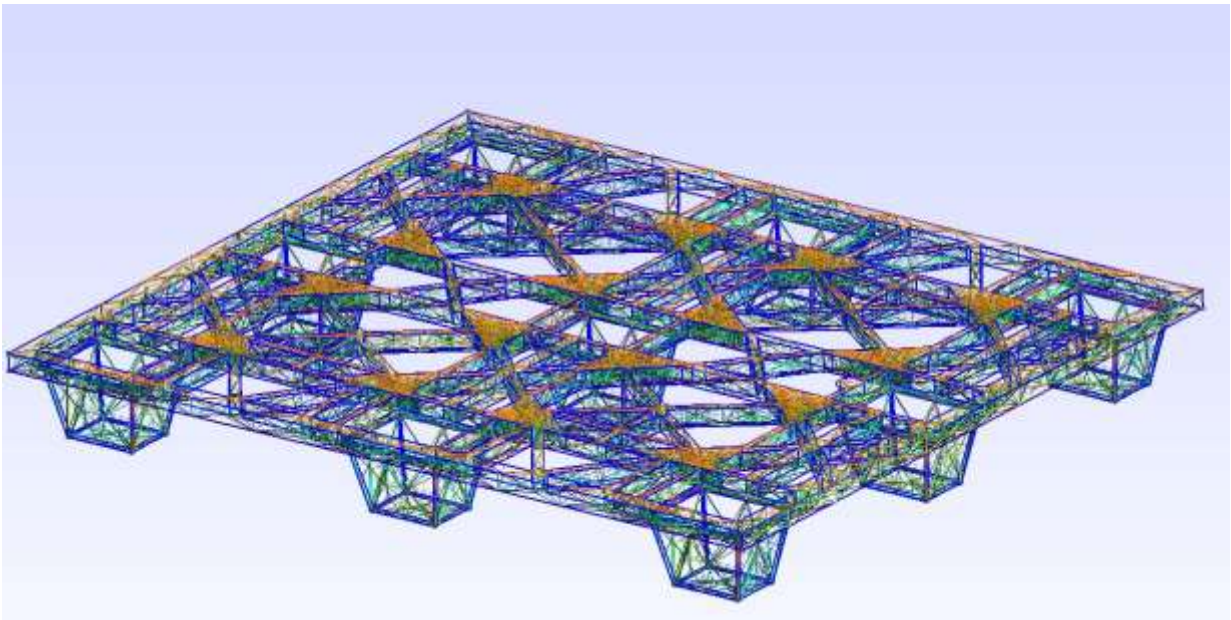

Figure 38. Quadratic mesh generated in Gmsh

The material used in this model is a linear low-density polyethylene with a nano-SiO2 weight percentage of 10% with an elastic modulus of 65 MPa [36], a Poisson's ratio of 0.35, and a density of .935g/cm3. For this model, none of the node of element sets generated by Gmsh were kept. A node set, however, was generated for the top surface using a plane definition, which was used to reference the location for the applied for. A plane definition was used for boundary condition directly in the boundary condition module of the preprocessor. The applied force was a total of 16kN, shown in Figure 39, divided over 100 steps.

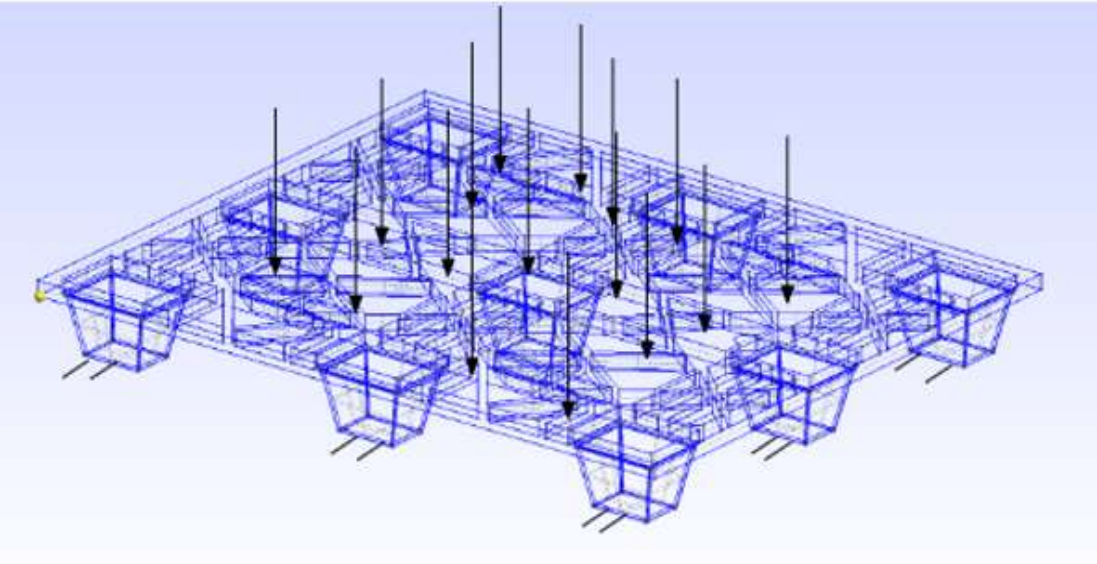

Figure 39. Applied loading and boundary conditions for pallet

The solution technique for this model is also the sparse direct technique with a maximum number of iterations set to 20. Batch messages, material messages, and trace solution function are turned off. The analysis was also set to continue analysis when the solution starts to diverge, but a convergence test for the residual forces was set to a tolerance of 0.0005%. Because this tolerance is extremely strict, the solution is more likely to be referred to as diverging, and thus allows the nonconvergent solution continuance function to be tested. Despite the strict convergence tolerance, all steps in the solution converged. A contour of the stress in the loading direction is shown in Figure 40.

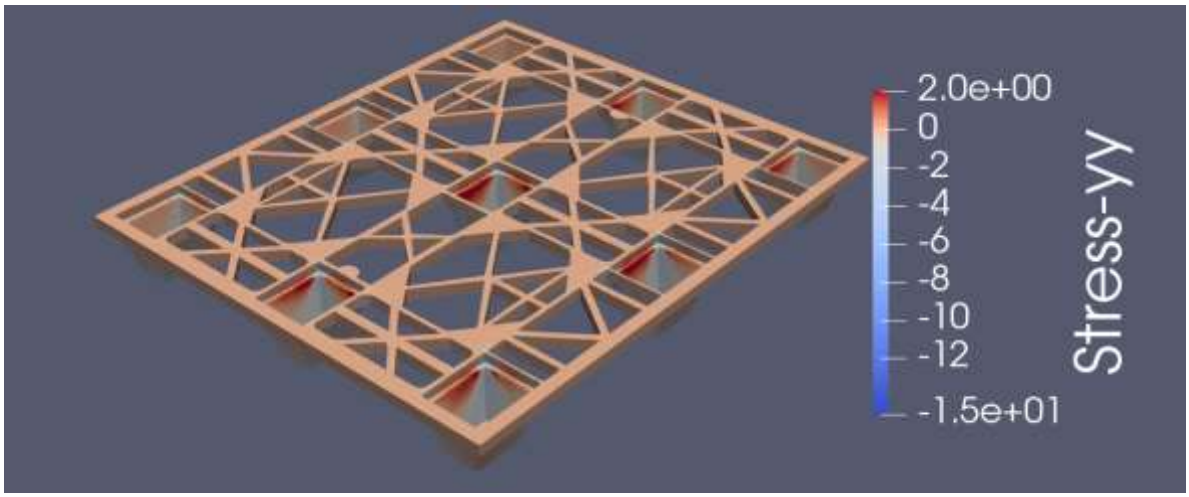

Figure 40. Stress contour of pallet in the loading direction in MPa

5. CONCLUSION

In summary, this project developed a preprocessing code, *W3DInput,* as an intermediate between the user and the finite element software Warp3D. This code is part of a larger framework that integrates the usage of Gmsh for mesh generation for first- and second-order solid elements. These meshes can be exported in Abaqus or Nastran formats, and read into the preprocessor. In the preprocessor, the mesh is read to find nodes and elements. These are separated, reformatted, and written to subdirectories. The remaining needs of an input file – material definitions, application of forces and boundary conditions, element definitions, solution controls, and output controls, are defined in *W3DInput*. The output of the code is an input file that is ready to run in the Warp3D solver. To test the functions of the code, five use cases were generated: a cantilever beam, a displacement control cube, a displacement control test with the material defined by a user-defined stress-strain curve, a crystal plasticity model, and a pallet with a complex geometry.

5.1 Limitations

While this preprocessor is robust enough to use for purely mechanical analysis and crystal plasticity, the preprocessor does not handle any thermal or thermomechanical behavior. Additionally, other complex material behaviors such as void nucleation or fracture modeling with the use of cohesive elements.

Additionally, this preprocessor does not process contact. Contact constraints are available in Warp3D as static or dynamic rigid bodies. However, this is limited to prismatic shapes like plates for cylinders. Additionally, contact in Warp3D is cannot be generated between two deformable bodies [1].

5.2 Future Applications

As mentioned in the above section, there are several analysis limitations in this preprocessor and in Warp3D. The inclusion of thermal properties and thermal loading, other material models, including the void nucleation and growth model, as the introduction of other element types such as cohesive elements, can be accomplished at the customer’s request.

Aside from thermal properties, there are an additional six material models in Warp3D that each serve purposes unique from the four included in the preprocessor. These models are:

- Gurson void nucleation
- Hydrogen-material interations
- Creep
- User-defined materials
- Cohesive material models
- Cyclic plasticity

Additionally, contact constraints can also be included. In Warp3D, contact constraints require the user to defined them in a similar order to the applied forces and boundary conditions. First, the user must assign it a reference name, and choose the shape of the rigid body. Second, the user but define the surface on the deformable body on which the contact will be applied. Lastly, the user must assign a direction of motion a speed at which it will. This is most useful for manufacturing processes.

In regard to the preprocessor's functionality, future work would include the development of a user interface. A user interface will not only aid in the ease of use of the preprocessor, but will greatly aid in making Warp3D a more useful alternative to commercial FEA software packages. With a user interface, this will also allow the preprocessor and the correction codes to become a single robust application.

APPENDIX A. CODING SAMPLES

LIST INPUT

This list input calls a file and a folder in which the file contains a set of nodes or elements. Several sets may be contained in a single file. These are read and separated, and then the user is prompted to choose which ones to keep.

```
setArr,setNames,lineAt = msh.sets(sfile,folder)
#find the lines to save to a variable - this is for later when writing lists

for i in range(0,len(lineAt)):
    ind1 = lineAt[i]
    thisName = setNames[i].rstrip()
    if i != len(lineAt):
        j = i + 1
        ind2 = lineAt[j]
        thisSet = list(setArr[ind1:ind2])
    else:
        thisSet = list(setArr[ind1:])
    exec('%s = %r' % (thisName,thisSet))

print('The following setNames were found:')
print(setNames)
print('Enter the index/indices of the sets you want to keep. The indices start at 0.')
saveList = raw_input('Separate the indices with a space.  ')
saveList = saveList.split(' ')

for i in range(0,len(saveList)):
    ind = saveList[i]
    a = eval(setNames[ind])
    abvA = cons.abvArr(a)
    inp.write('list "'+setNames[ind]+'" ')
    for j in range(0,len(abvA)):
        inp.write(abvA[j]+' ')
    inp.write('\n')
    inp.write('!'+'\n')
```

MATERIAL MODEL – DEFORMATION

An example of the input for the deformation material model

```
import numpy as np
import time
from preprocessor import inp

def writeMat(modelType,fgm,matDir):

    matName = raw_input("Give the material a name.  ")
    inp.write('material '+matName+'\n')
    inp.write(' properties ')

    if modelType == 1:
        inp.write('deformation')
        if matDir == 1:
            print("The following properties are included in the deformation model:")
            print("-elastic modulus (property name: e)")
            print("-poisson's ratio (property name: nu)")
            print("-density (property name: rho)")
            print("-yield stress (property name: yld_pt)")
            print("-power-law exponent (property name: n_power)")
            print('all properties must be included in the property definition.')
            print('If it is not desired to define density, enter zero. For a  ')
            print('linear-elastic analysis, enter a high yield stress and 1 for')
            print('the power-law exponent')
            print("...")
            time.sleep(5)

        if fgm == 1:
            if matDir == 1:
                print("Use the variable names given above - the code will not recognize them otherwise.")
                print("Use spaces to separate names and values.")
                print("...")
                time.sleep(5)

            #identifying which properties are functionally graded
            print('First, the properties that are part of the FGM property must be identified.')
            allProps = ['e','nu','rho','yld_pt','n_power']
            print('Enter which properties are part of the functionally graded propery DO NOT enter')
            fgmProps = raw_input('the values. Do not enter properties that do not vary. Use spaces to separate.  ')
            readProps = fgmProps.split(' ') #this will be used to verify that the properties for successive sections match.

            #arrays for writing the material definition
            fgmWrite = np.array([])

            #finding the properties that aren't functionally graded
            print('Next, the remaning properties that are not part of the functionally graded')
            print('property definition must identified and defined.')
            for i in range(0,len(allProps)):
                if allProps[i] not in readProps:
                    print(allProps[i]+' was not found in the functionally graded properties.')
                    val = raw_input('Enter the value for the property: '+allProps[i]+'.  ')
                    fgmWrite = np.append(fgmWrite,[allProps[i],val])
                else:
                    fgmWrite = np.append(fgmWrite,[allProps[i],'fgm'])
            print('non-FGM properties completed.')

            #write the material definition
            firstHalf = len(fgmWrite)/2
            if (len(fgmWrite)%2 != 0):
                firstHalf = firstHalf-1
            commaInd = firstHalf-1
            for i in range(0,len(fgmWrite)):
                inp.write(' '+fgmWrite[i])
                if i == commaInd:
                    inp.write(','+'\n')
                    inp.write('    ')
            return readProps
        else:
            print('Enter the properties using the format shown below:')
            print("e 200000 nu 0.5 rho 2250 yld_pt 305 n_power -0.12")
            properties = raw_input('The property name should precede the value.  ')
            if 'n_power' not in properties:
```

```
    properties = properties + 'n_power 1.0'
props = properties.split(' ')
firstHalf = len(props)/2
if (len(props)%2 != 0):
    firstHalf = firstHalf-1
commaInd = firstHalf-1
for i in range(0,len(props)):
    inp.write(' '+props[i])
    if i == commaInd:
        inp.write(','+'\n')
        inp.write('    ')
```

REFERENCES

[1] B. Healy *et al.*, "Civil Engineering Studies Warp3D," no. 607, 2013.

[2] G. Christophe and R. Jean-FranÁois, "Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities," *Int. J. Numer. Methods Eng.*, vol. 79, no. 11, pp. 1309–1331, 2009.

[3] L. A. Schoof and V. R. Yarberry, "EXODUS II: A finite element data model," no. September, 1994.

[4] T. Blacker *et al.*, "CUBIT 15.2 User Documentation," no. May, p. 990, 2016.

[5] H. Resk, L. Delannay, M. Bernacki, T. Coupez, and R. Logé, "Adaptive mesh refinement and automatic remeshing in crystal plasticity finite element simulations," *Model. Simul. Mater. Sci. Eng.*, vol. 17, no. 7, 2009.

[6] B. Gärtner and M. Hoffmann, "Chapter 6 Delaunay Triangulations," *Comput. Geom. Lect. Notes*, no. c, pp. 55–71, 2012.

[7] J. Schöberl, "An advancing front 2D/3D-mesh generator based on abstract rules," *Comput. Vis. Sci.*, vol. 1, no. 1, pp. 41–52, 1997.

[8] K. Beatty and N. Mukherjee, "A transfinite meshing approach for body-in-white analyses," *Proc. 19th Int. Meshing Roundtable, IMR 2010*, pp. 49–65, 2010.

[9] "Abaqus." Dassault Systemes, Velizy-Villacoublay, France, 2013.

[10] "OpenMesh." Rwth Aachen University Visual Computing Institute, Aachen, Germany, 2019.

[11] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "MeshLab: an Open-Source Mesh Processing Tool," *Eurographics Ital. Chapter Conf. 2008 Salerno, Italy*, pp. 129–136, 2008.

[12] "SolidWorks." Dassault Systemes, Velizy-Villacoublay, France, 2016.

[13] "PTC Creo." PTC, Boston, MA, 2018.

[14] A. F. Bower, "Cnstitutive Models - Relations between Stress and Strain." [Online]. Available: http://solidmechanics.org/Text/Chapter3_12/Chapter3_12.php. [Accessed: 23-Mar-2018].

[15] G. I. Taylor, "[Lecture Notes] Plastic Strain in Metals," *Journal of the Institute of Metals*, vol. 62. pp. 307–325, 1938.

[16] J. F. W. Bishop and R. Hill, “XLVI. A theory of the plastic distortion of a polycrystalline aggregate under combined stresses.,” *London, Edinburgh, Dublin Philos. Mag. J. Sci.*, vol. 42, no. 327, pp. 414–427, 2014.

[17] F. Roters, P. Eisenlohr, L. Hantcherli, D. D. Tjahjanto, T. R. Bieler, and D. Raabe, “Overview of constitutive laws, kinematics, homogenization and multiscale methods in crystal plasticity finite-element modeling: Theory, experiments, applications,” *Acta Mater.*, vol. 58, no. 4, pp. 1152–1211, 2010.

[18] F. Roters, P. Eisenlohr, T. R. Bieler, and D. Raabe, *Crystal Plasticity Finite Element Methods*, vol. 4, no. 4. 2011.

[19] D. Raabe, P. Klose, B. Engl, K. P. Imlau, F. Friedel, and F. Roters, “Concepts for integrating plastic anisotropy into metal forming simulations,” *Adv. Eng. Mater.*, vol. 4, no. 4, pp. 169–180, 2002.

[20] P. R. Dawson, “Dawson00a,” vol. 37, p. 16, 1999.

[21] F. Roters, P. Eisenlohr, T. R. Bieler, and D. Raabe, *Crystal Plasticity Finite Element Methods: In Materials Science and Engineering*. 2010.

[22] Rice~J.R., “Inelastic constitutive relations for solids: an internal-variable theory and its application to metal plasticity,” *J. Mech. Phys. Solids*, vol. 19, no. 6, pp. 433–455, 1971.

[23] J. W. Hutchinson, “Bounds and Self-Consistent Estimates for Creep of Polycrystalline Materials,” *Proc. R. Soc. A Math. Phys. Eng. Sci.*, vol. 348, no. 1652, pp. 101–127, 2006.

[24] E. Voce, “The relationship between stress and strain for homogenous deformations,” *J. Inst. Met.*, vol. 74, pp. 537–562, 1948.

[25] P. S. Follansbee and U. F. Kocks, “A constitutive description of the deformation of copper based on the use of the mechanical threshold stress as an internal state variable,” *Acta Metall.*, vol. 36, no. 1, pp. 81–93, 1988.

[26] VOCE and E., “A practical strain hardening function,” *Metallurgia*, vol. 51, pp. 219–226, 1955.

[27] U. F. Kocks, “A statistical theory of flow stress and work-hardening,” *Philos. Mag.*, vol. 13, no. 123, pp. 541–566, 1966.

[28] G. van Rossum, “Python.” 2017.

[29] “Paraview.” Kitware, Sandia National Laboratory, Los Alamos National Laboratory, Albuquerque, NM; Clifton Park, NY; Los Alamos, NM, 2017.

[30] Siemens Industry Software, “Element Library Reference,” pp. 1–272, 2014.

[31] Y. Wang, “A two-parameter characterization of elastic-plastic crack tip fields and applications to cleavage fracture.” 1991.

[32] F. Jacob and B. Ted, *A first course in finite elements*. 2007.

[33] “Matweb.” [Online]. Available: http://www.matweb.com/.

[34] M. Sangid, “AAE648: Modeling Damage and Strength Mechanisms in Materials.”

[35] “Tarima Plastica - GrabCad,” 2019. [Online]. Available: grabcad.com.

[36] E. Kontou and M. Niaounakis, “Thermo-mechanical properties of LLDPE/SiO2 nanocomposites,” *Polymer (Guildf).*, vol. 47, no. 4, pp. 1267–1280, 2006.