An overview of ExaConstit and its use in the ExaAM project

Robert Carson[1]

[1]Lawrence Livermore National Laboratory

Feb 1, 2022

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344, and supported by ECP (17-SC-20-SC), a collaborative effort of U.S. DOE Office of Science and NNSA. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

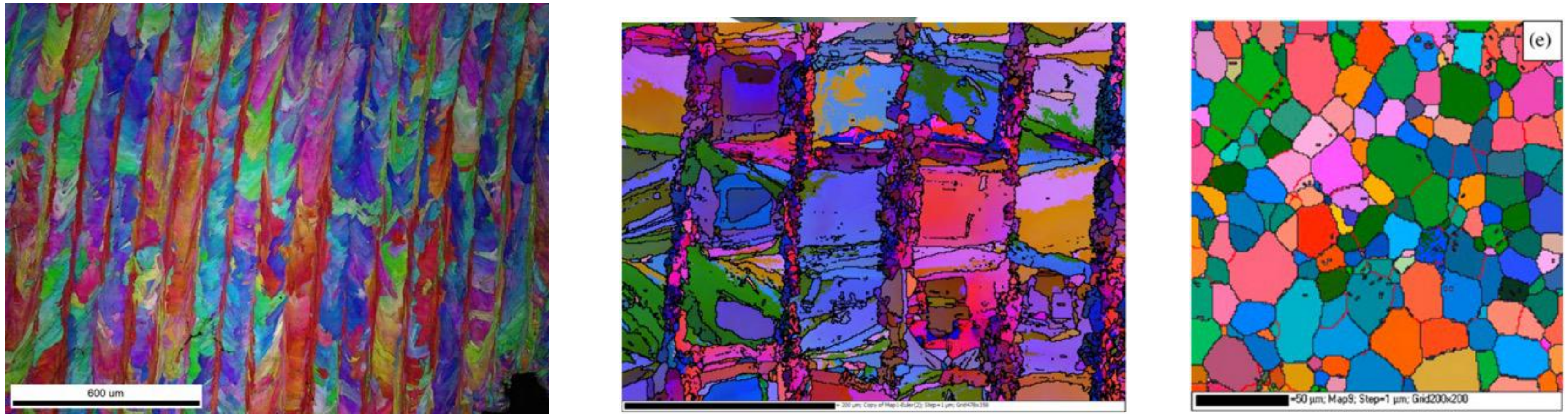
Outline

- Motivation and overview of project
- Overview of ExaConstit
  - FEM formulation
- Overview of GPU Porting Efforts
  - Element assembly formulation
  - Exascale readiness of ExaConstit
- Overview of local property calculation workflow
- ExaConstit's use in other areas
- Conclusion/On-going work

Motivation

- AM produces as-built metals that are characteristically quite different from typical manufacturing processes in both the microstructure and mechanical response [2]

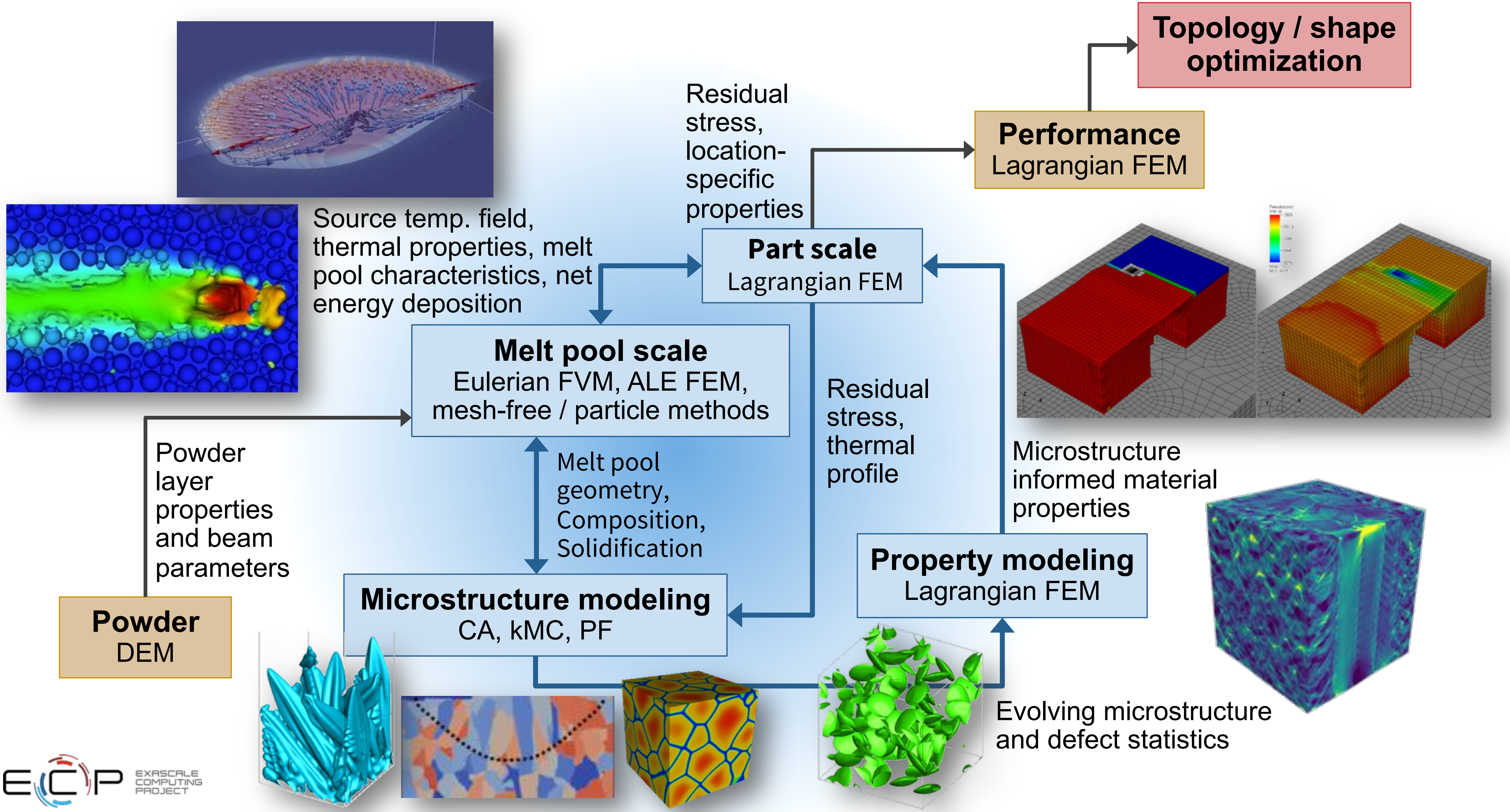
- Want to couple the microstructure development and the local macroscopic properties
  - Can be accomplished through crystal plasticity models that utilize finite elements


600 um
= 200 µm; Copy of Map1 Euler(2); Step=1 µm; Grid476x358
(e)
=50 µm; Map9; Step=1 µm; Grid200x200

Microstructure of an AM part at 2 different locations (Left and Middle) [3] and a traditional manufacturing microstructure [4]

[2] C. Bronkhorst *et al* 2019 *Int. J. Plasticity.* **118** 70-86, [3] *NIST* AMB2018-01-625-CBM-B2-P1-L7-TRANS-MS-EBSD,
[4] *J Alloys Compd* **volume 472, issue 1-2,** pages 127-132 (2009)

ExaAM focus (blue) fits within a larger modeling and design optimization context



Topology / shape optimization

Performance
Lagrangian FEM

Residual stress, location-specific properties

Part scale
Lagrangian FEM

Source temp. field, thermal properties, melt pool characteristics, net energy deposition

Melt pool scale
Eulerian FVM, ALE FEM, mesh-free / particle methods

Powder layer properties and beam parameters

Powder
DEM

Melt pool geometry, Composition, Solidification

Residual stress, thermal profile

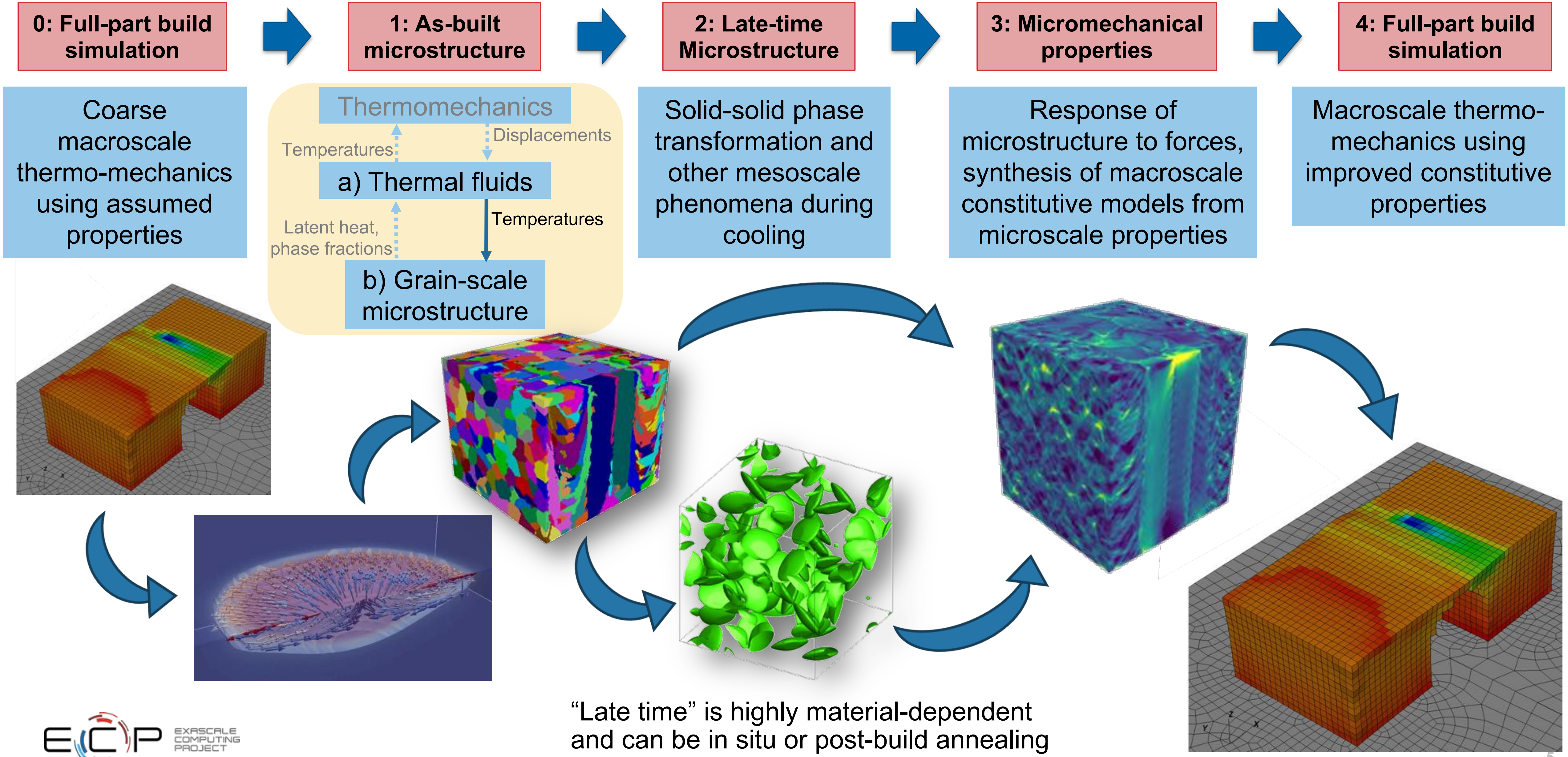
Microstructure informed material properties

Microstructure modeling
CA, kMC, PF

Property modeling
Lagrangian FEM

Evolving microstructure and defect statistics

A full ExaAM simulation consists of five stages



0: Full-part build simulation

Coarse macroscale thermo-mechanics using assumed properties

1: As-built microstructure

Thermomechanics

Temperatures

Displacements

a) Thermal fluids

Latent heat, phase fractions

Temperatures

b) Grain-scale microstructure

2: Late-time Microstructure

Solid-solid phase transformation and other mesoscale phenomena during cooling

3: Micromechanical properties

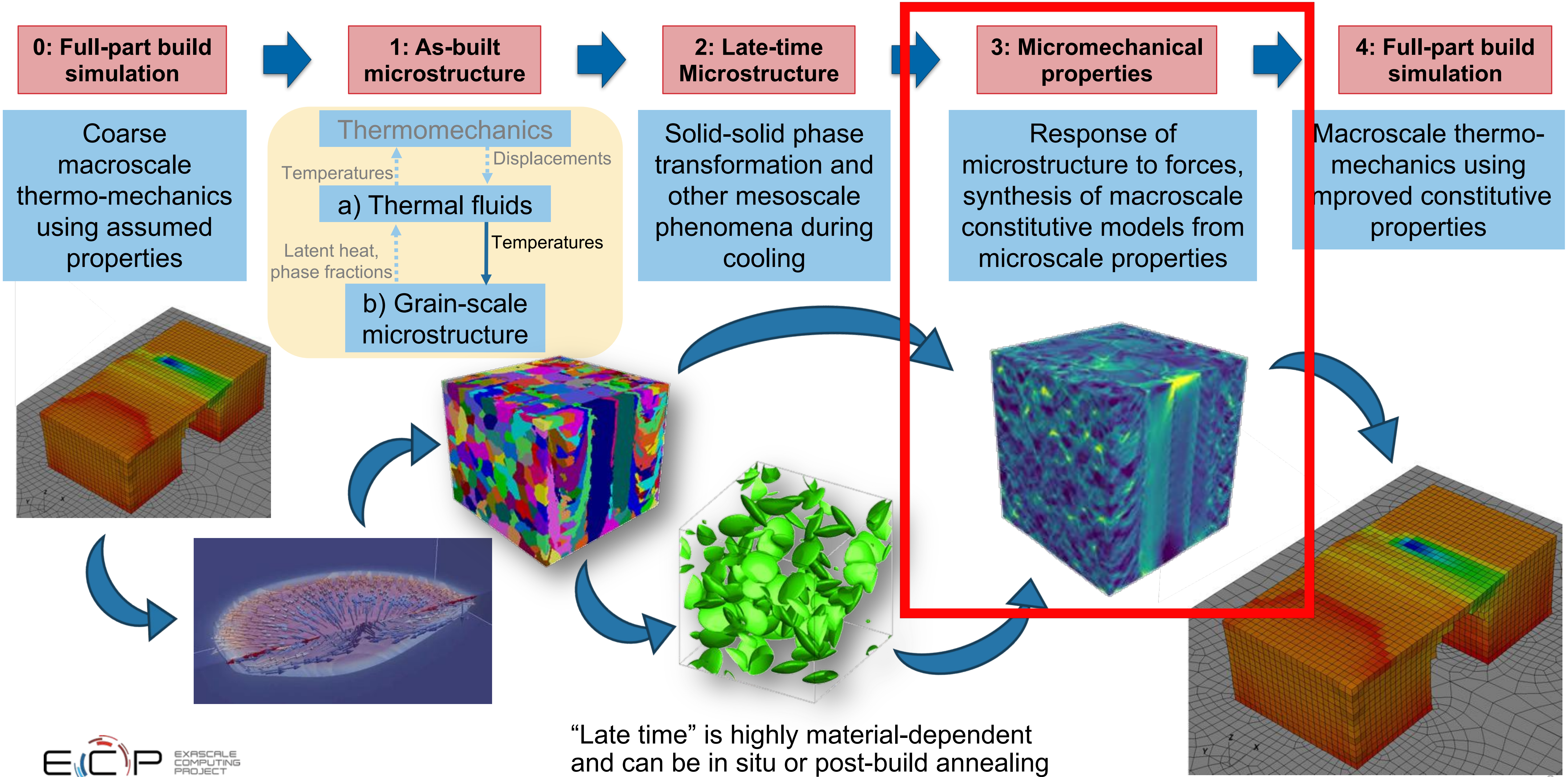
Response of microstructure to forces, synthesis of macroscale constitutive models from microscale properties

4: Full-part build simulation

Macroscale thermo-mechanics using improved constitutive properties

“Late time” is highly material-dependent and can be in situ or post-build annealing

A full ExaAM simulation consists of five stages



0: Full-part build simulation

Coarse macroscale thermo-mechanics using assumed properties

1: As-built microstructure

Thermomechanics

Temperatures

Displacements

a) Thermal fluids

Latent heat, phase fractions

Temperatures

b) Grain-scale microstructure

2: Late-time Microstructure

Solid-solid phase transformation and other mesoscale phenomena during cooling

3: Micromechanical properties

Response of microstructure to forces, synthesis of macroscale constitutive models from microscale properties

4: Full-part build simulation

Macroscale thermo-mechanics using improved constitutive properties

“Late time” is highly material-dependent and can be in situ or post-build annealing

EXASCALE COMPUTING PROJECT

Overview of ExaConstit

- A new nonlinear quasi-static, implicit FEM solid mechanics code built on the MFEM library
  - Development began in 2018 for ExaAM project
  - Updated Lagrangian formulation (velocity based)
  - Natively supports a wide range of element types
  - Supports complex parts and non-trivial/changing Dirichlet BCs
  - Available at https://github.com/LLNL/ExaConstit

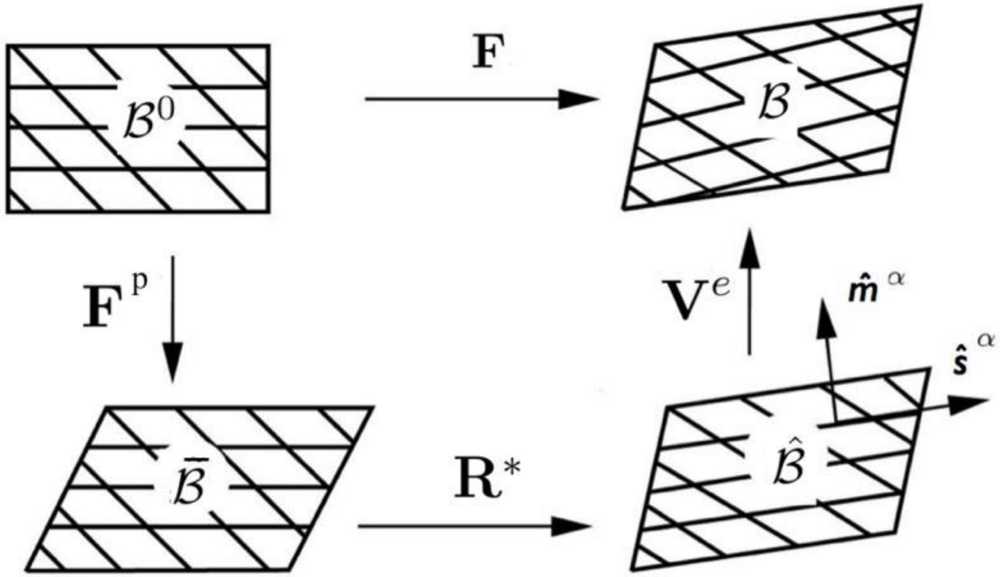
Overview of ExaConstit

- Designed with exascale computers in mind but performant even on desktop machines
- New features, workflows, and post-processing tools are constantly being added
- Bread and butter is crystal plasticity type problems
  - Crystal plasticity models are provided through ExaCMech library
    - ExaCMech is available at https://github.com/LLNL/ExaCMech
    - Models can be upwards of 75-90x faster than Abaqus UMAT runs
  - Users can also run UMAT problems as well
    - UMAT framework enables us to run a wide range of industry and research models

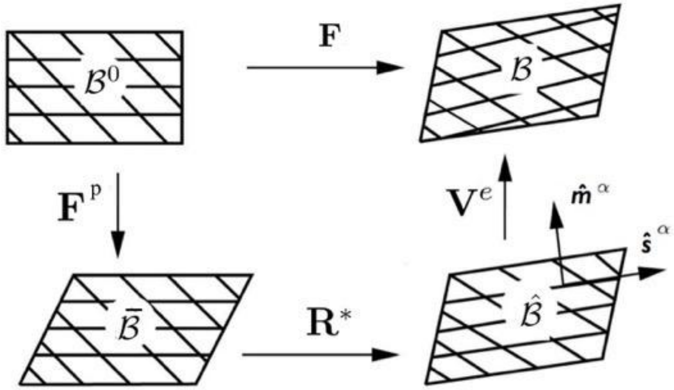
Model Overview

- A large strain single crystal elasto-viscoplastic model is being used
  - Small deviatoric elastic strain assumption is made

$$\mathbf{F} = \mathbf{F}^e\mathbf{F}^p$$



Model Overview



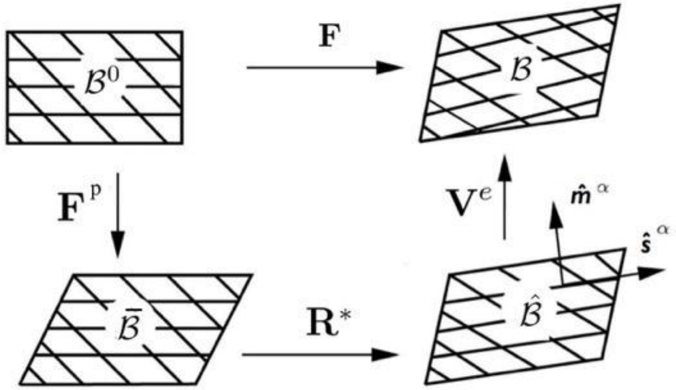
$$\mathbf{L} = \dot{\mathbf{V}}^e\mathbf{V}^{e-1} + \mathbf{V}^e\hat{\mathbf{L}}^p\mathbf{V}^{e-1}$$

$$\hat{\mathbf{L}}^p = \dot{\mathbf{R}}^*\mathbf{R}^{*T} + \mathbf{R}^*\dot{\mathbf{F}}^p\mathbf{F}^{p-1}\mathbf{R}^{*T}$$

$$\bar{\mathbf{L}}^p \;=\; \dot{\mathbf{F}}^p\mathbf{F}^{p-1} \qquad \bar{\mathbf{L}}^p = \sum_{\alpha=1}^{n} \dot{\gamma}^\alpha \bar{\mathbf{s}}^\alpha \otimes \bar{\mathbf{m}}^\alpha$$

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

Model Overview



$$\mathbf{L} = \dot{\mathbf{V}}^e \mathbf{V}^{e-1} + \mathbf{V}^e \hat{\mathbf{L}}^p \mathbf{V}^{e-1}$$

$$\hat{\mathbf{L}}^p = \dot{\mathbf{R}}^* \boxed{\mathbf{R}^*}^T + \mathbf{R}^* \dot{\mathbf{F}}^p \mathbf{F}^{p-1} \mathbf{R}^{*T}$$

$$\bar{\mathbf{L}}^p = \dot{\mathbf{F}}^p \mathbf{F}^{p-1} \qquad \bar{\mathbf{L}}^p = \sum_{\alpha=1}^{n} \dot{\gamma}^\alpha \bar{\mathbf{s}}^\alpha \otimes \bar{\mathbf{m}}^\alpha$$

Small elastic strain assumption

$$\mathbf{V}^\mathbf{e} = \mathbf{I} + \boxed{\boldsymbol{\varepsilon}^\mathbf{e}}$$

$$\boxed{\dot{\gamma}} = \dot{\gamma}_0 \left(\frac{\tau^\alpha}{g^\alpha} \right)^{\frac{1}{m}} \operatorname{sgn}(\tau^\alpha)$$

FEM Formulation

- We’re solving for the conservation of linear momentum using a Newton Raphson scheme
  - Our finite element formulation takes on a form very similar to linear elasticity
  - We make use of an updated Lagrangian formulation here

$$[K_{tan}]\{-\Delta\mathbf{V}\} = \{f_{int}\}$$

$$[K_{tan}] = \int_{\Omega} [\mathbf{B}]^{\mathrm{T}}[\boldsymbol{C}^{\boldsymbol{\sigma\tau}}][\mathbf{B}]\, d\Omega \qquad [\boldsymbol{C}^{\boldsymbol{\sigma\tau}}] = \frac{\mathrm{d}\boldsymbol{\sigma}}{\mathrm{d}\boldsymbol{d}}$$

$$\{\mathbf{V}\}^{i+1} = \{\mathbf{V}\}^{i} + \{\Delta\mathbf{V}\}$$

$$\{f_{int}\} = \int_{\Omega} [\mathbf{B}]^{\mathrm{T}}\{\boldsymbol{\sigma}\} d\Omega$$

$$\{\mathbf{x}\}^{t+\Delta t} = \{\mathbf{x}\}^{t} + \Delta t\{\mathbf{V}\}^{i+1}$$

Note: This formulation uses the velocity rather than the typical displacement

Transitioning over to the GPU: Separate Material Model and FEM code

- ExaConstit originally had the material model calculation tied in with the linearized RHS calculation and compute stiffness matrix
  - This strategy does not scale well as different assembly operations are added
- Refactored code into a set-up phase, pre-processing step before assembly, RHS calculations, and gradient calculation / operation

Set-up phase

- Calculate necessary parameters to use within our material model
  - Pre-processing stage for material kernel
- Material kernel stage
- Post-processing material kernel stage
  - Perform what-ever steps are necessary for material kernel’s data to be used by the rest of the code
- MFEM_FORALL loops are used for pre & post processing steps
- Material model uses its own parallelization strategy

An Element Assembly Formulation

- We’re making use of efficient formulations developed back in the 80s [5]
- $\{f_{int}\}$ is just calculated using partial assembly formulation
- This formulation allows us an efficient way to compute sub-blocks of $[K_{tan}]$
  - Could further specialize this for tensor-type elements for further performance gains

$$[K_{tan}] = \int_{\Omega} [\mathbf{B}]^{\mathrm{T}} [\boldsymbol{C}^{\boldsymbol{\sigma\tau}}][\mathbf{B}]\, d\Omega$$

$$[K_{tan}] = \int_{\Omega} [\boldsymbol{b_1 b_2} \dots \boldsymbol{b_N}]^T [\boldsymbol{C}^{\boldsymbol{\sigma\tau}}][\boldsymbol{b_1 b_2} \dots \boldsymbol{b_N}] d\Omega$$

$$[\boldsymbol{b_1 b_2} \dots \boldsymbol{b_N}] = [\mathbf{B}]$$

$$\{f_{int}\} = \int_{\Omega} [\mathbf{B}]^{\mathrm{T}} \{\boldsymbol{\sigma}\} d\Omega$$

$$\det(J) w_{qpt} \nabla_{ij} \phi_{\xi}^{T} J_{jk}^{-T} \sigma_{kl}$$

[5] AK. Gupta 1983 *Int J Numer Meth Eng.* **19** 1410-1413

Incompressible Material Support within ExaConstit

- An initial integration formulation for incompressible has been implemented
  - Based on the work presented in [6]
  - New integrators are not as simple to port over to the element assembly formulation

- This formulation brings linear hexahedron response much more in line with response from higher order elements
  - Runtime is comparable to full integration approach

Contains a mean dilation and full deviatoric response

$$\mathbf{B}_a = \left[\begin{array}{ccc} B_1 & 0 & 0 \\ 0 & B_2 & 0 \\ 0 & 0 & B_3 \\ \hline B_2 & B_1 & 0 \\ 0 & B_3 & B_2 \\ B_3 & 0 & B_1 \end{array}\right] \longrightarrow \bar{\mathbf{B}}_a = \left[\begin{array}{ccc} B_5 & B_6 & B_8 \\ B_4 & B_7 & B_8 \\ B_4 & B_6 & B_9 \\ \hline B_2 & B_1 & 0 \\ 0 & B_3 & B_2 \\ B_3 & 0 & B_1 \end{array}\right]$$

Full integration gradient matrix

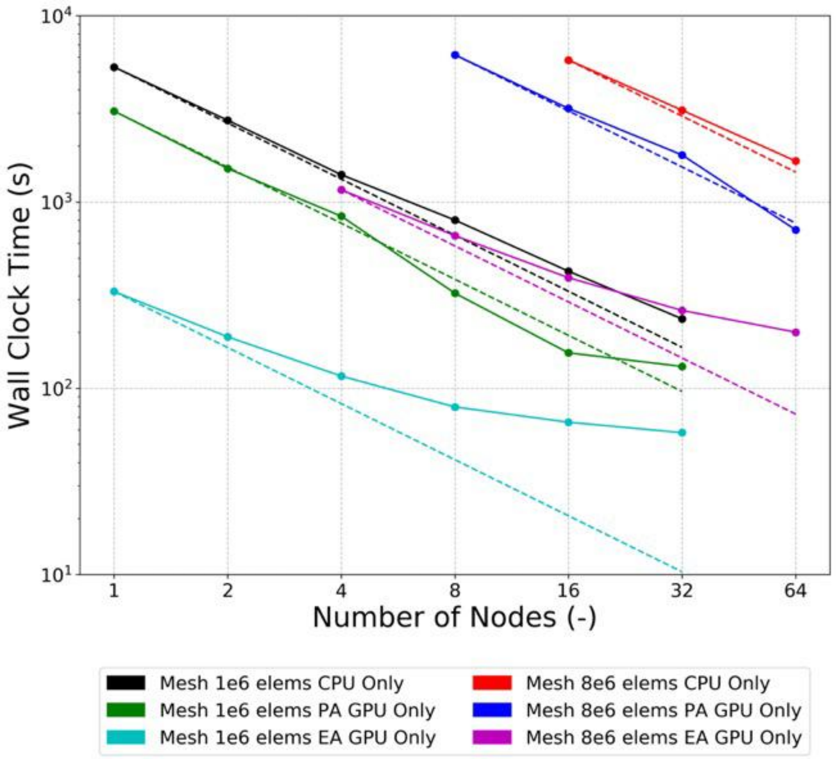
Incompressible integration gradient matrix

[6] TRJ. Hughes 1980 *Int J Numer Meth Eng.* **15** 1413-1418

Exascale Readiness Status: ExaConstit

Summit performance

- 15x speed-up with GPU implementation over CPU
- GPU strong and weak scaling for most performant assembly method (element assembly)
- Poor strong and weak scaling is due to start-up cost of reading in the mesh and partitioning it
  - Recent work-in-progress within MFEM is expected to drastically improve this area (Thanks Veselin!)
- Ported ExaCMech over to HIP this past year
  - ExaConstit port will follow this year

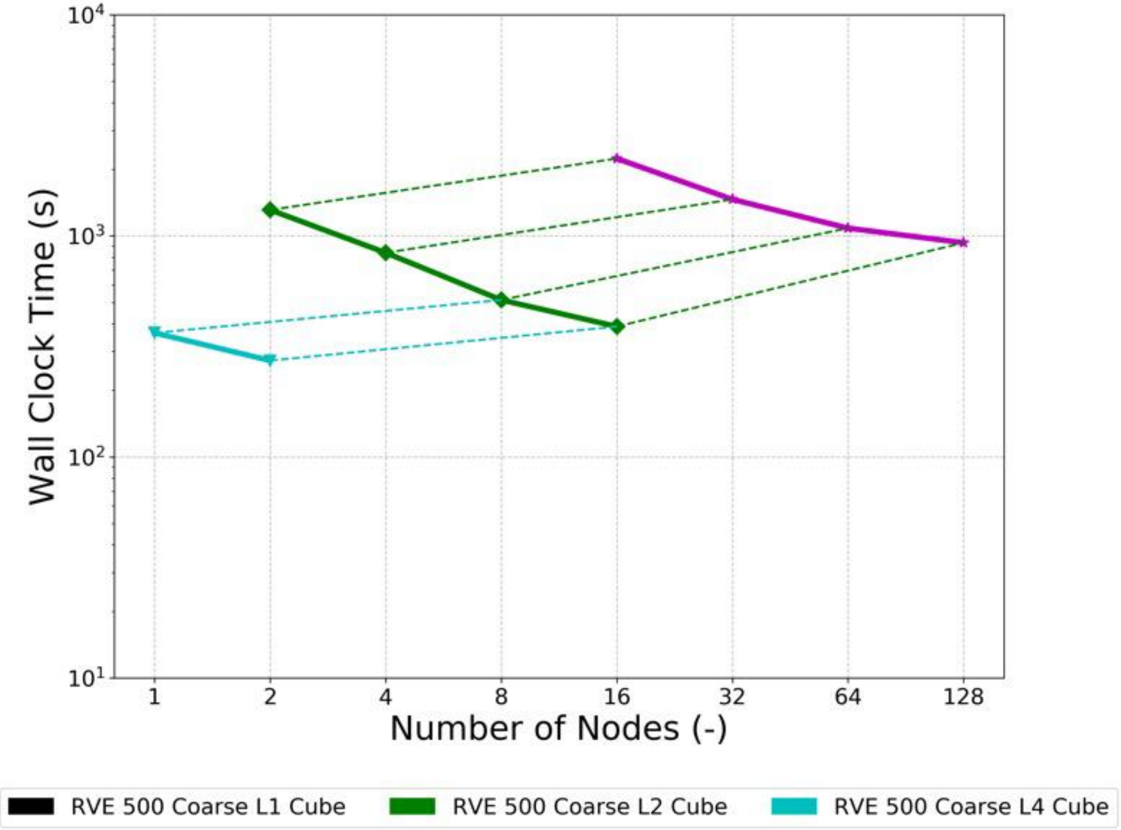


Strong scaling across different assembly methods

Exascale Readiness Status: ExaConstit

Summit performance

- 15x speed-up with GPU implementation over CPU
- GPU strong and weak scaling for most performant assembly method (element assembly)
- Poor strong and weak scaling is due to start-up cost of reading in the mesh and partitioning it
  - Recent work-in-progress within MFEM is expected to drastically improve this area (Thanks Veselin!)
- Ported ExaCMech over to HIP this past year
  - ExaConstit port will follow this year



Strong and weak scaling of a typical AM microstructure

EXASCALE COMPUTING PROJECT

Computational shifts between GPU and Host Runs

Summit performance

- How do computational costs shift when moving from CPU to GPU?
  - Looking at a 450k element mesh run with Caliper
- Krylov solver still dominates run time
- What's going on with the MPI calls?
  - MPI is taking up a larger % of our runtime now
    - Partially due to issues with MPI D2D calls not being as performant as possible
  - MPI times are 40% less on the GPU though
- Improvements could be made by looking at communication hiding Krylov solvers

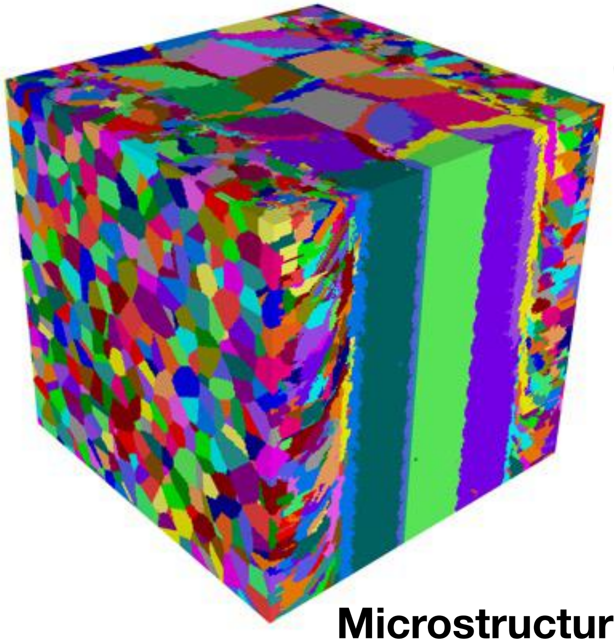
CPU Kernel Name	Time % (total)
Krylov Solver	84.534
Material Model	7.091
MPI Calls	6.778
Element Assembly	0.577
Simulation Initialization	0.295
Material Model Setup	0.283
Integrator Setup	0.137
Total	99.69

GPU Kernel Name	Time % (total)
Krylov Solver	55.16
MPI Calls	18.83
Element Assembly	15.80
Material Model	6.28
Simulation Initialization	1.39
Integrator Setup	0.93
Main Driver	0.88
Assemble Diagonal	0.47
Post-processing step	0.14
Total	99.87

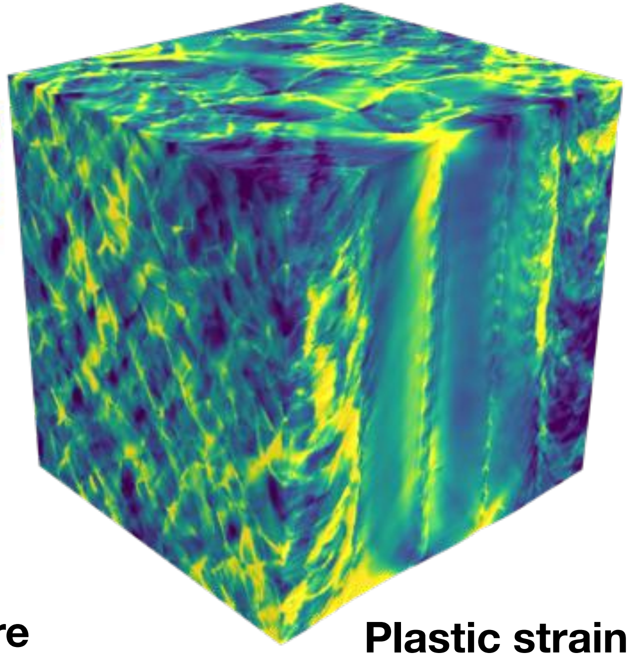
Stage 3: Micromechanical properties (ExaConstit)

- We now have a performant code but what do we need that for?
- ExaAM requires local property to be calculated from microstructures obtained through out an AM part
- Local properties are calculated by running tons of crystal plasticity simulations under varying loading conditions and temperatures
  - Simply running 1 simulation per microstructure is not adequate for AM parts
  - Complex macroscopic models are needed for part scale simulations (not computed by ExaConstit)

AMB2018-01 L8 microstructure generated by ExaCA
0.5^3 mm domain, 3475 grains



Microstructure from ExaCA

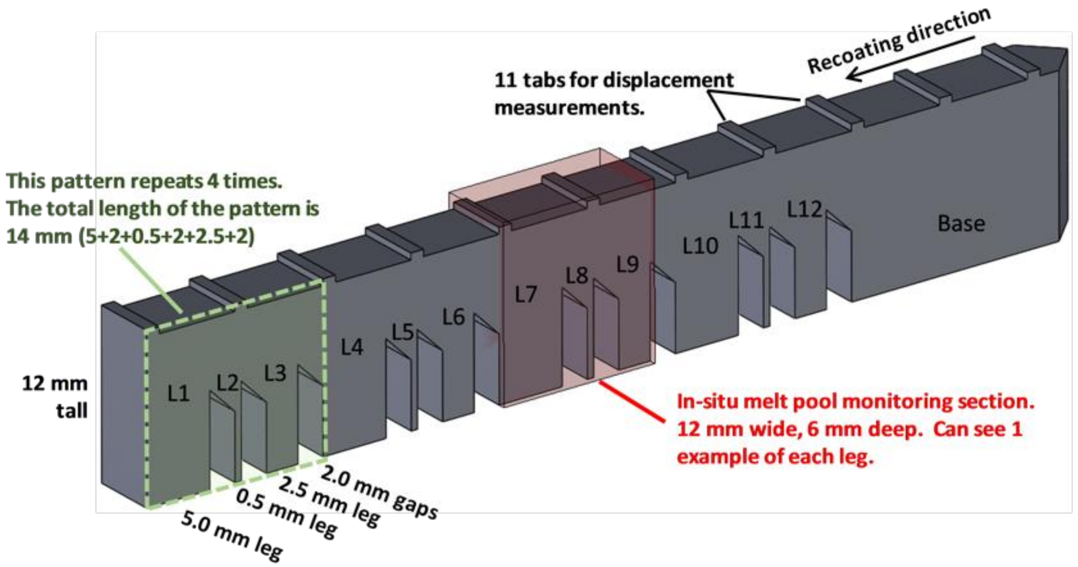


Plastic strain rate from ExaConstit

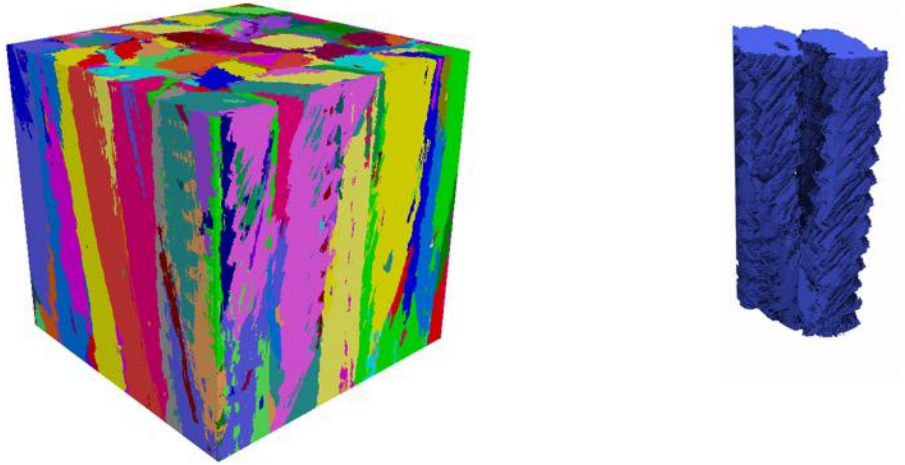
- Localization of plastic strain is non-trivial for these complex AM microstructures - What is an appropriate RVE?
- Hydrostatic stress is commonly used to drive porosity models - How can we appropriately homogenize?

Representative Volume Element for Macroscopic Calculations

- Microstructure was obtained from a cellular automata simulation based on scan paths of the NIST AM benchmark problem AMB2018-01. Thermal histories provided by either TruchasPBF or OpenFOAM simulations of the scan path
  - Inconel 625 material
    - Voce hardening model parameterized against AFRL MIDAS challenge 3 data [7]
  - Scan pattern mimics one of the larger legs of the AMB2018-01
  - Microstructure used is away from edge boundaries
  - 500^3 microns sample with 300^3 voxel size


Recoating direction
11 tabs for displacement measurements.
This pattern repeats 4 times. The total length of the pattern is 14 mm (5+2+0.5+2+2.5+2)
12 mm tall
L1
L2
L3
L4
L5
L6
L7
L8
L9
L10
L11
L12
Base
In-situ melt pool monitoring section. 12 mm wide, 6 mm deep. Can see 1 example of each leg.
2.0 mm gaps
2.5 mm leg
0.5 mm leg
5.0 mm leg

AMB2018-01 part

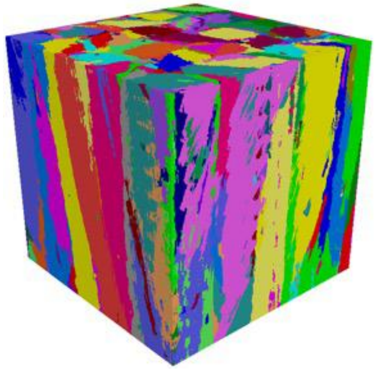


Initial CA microstructure

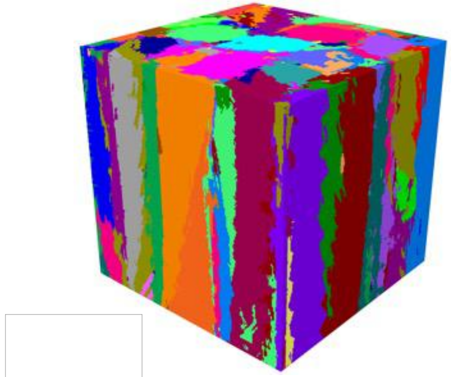
[7] https://materials-data-facility.github.io/MID3AS-AM-Challenge/

Domain Size Microstructures

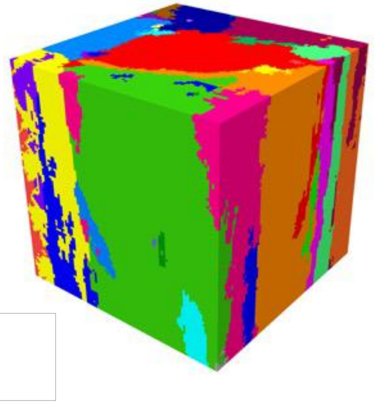
- Microstructures are all taken from the middle of the initial 500^3 microns sample
- Microstructures have dimensions: 500^3 microns, 333.3^3 microns, and 166.7^3 microns
  - 27, 8, and 1 million linear hexahedron elements respectively
- Uniaxial tension tests out to 5% were taken on these samples
  - Symmetry boundary conditions were applied to all samples
  - Large strains are not of great interest to our AM applications



Large Microstructure



Medium Microstructure

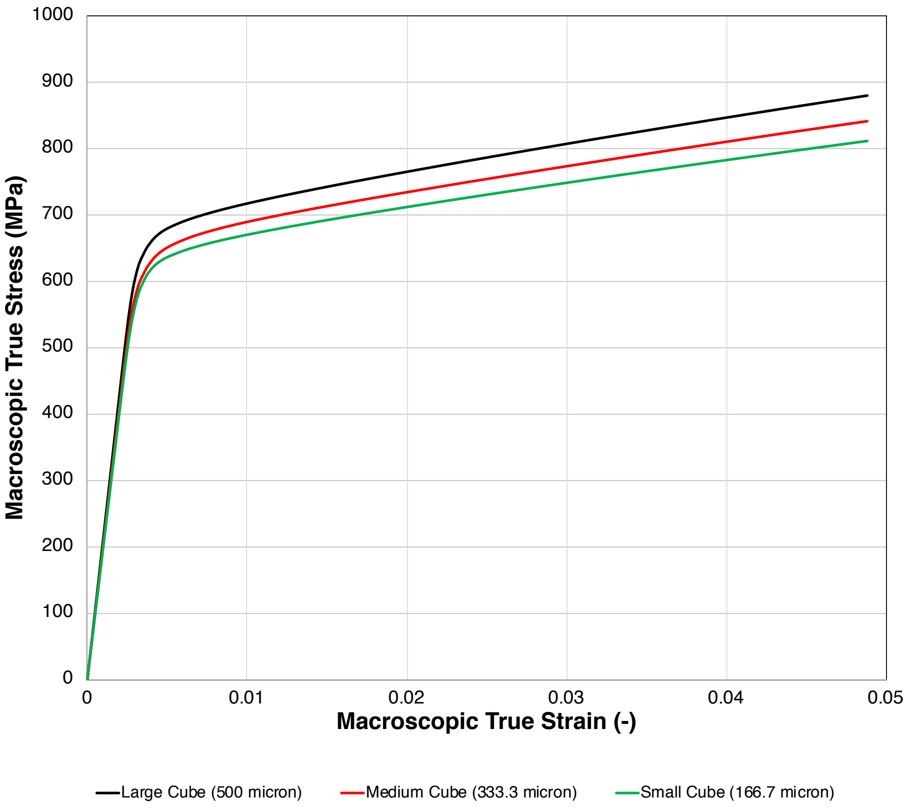


Small Microstructure

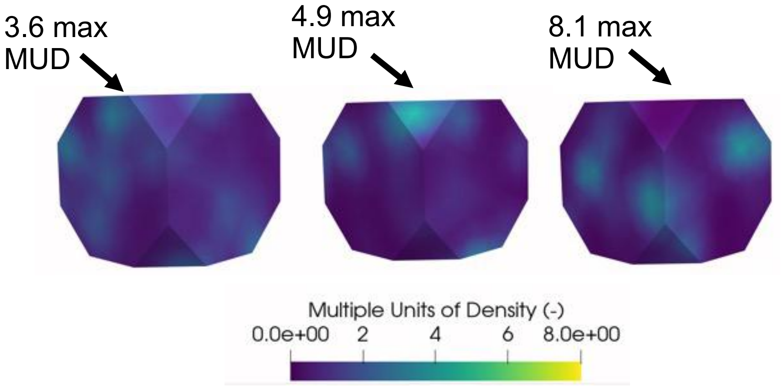
[8] TRJ. Hughes 1980 *Int J Numer Meth Eng.* **15** 1413-1418

Macroscopic Stress Strain Response

- The medium cube was below 5% error for the entire loading history
  - Error continues to grow as sample enters fully developed plastic flow
- The small cube started around 6% error and continued to grow as simulation progressed
- Differences in responses was largely driven by differences in crystallographic texture
  - Follows historical trends already observed



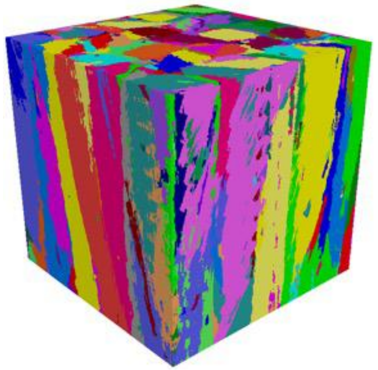
Macroscopic stress strain curves for different size microstructures



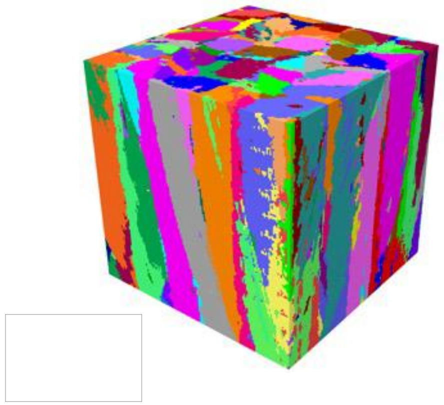
Orientation distribution plots of large, medium, and small cube

A Need for Mesh Coarsening

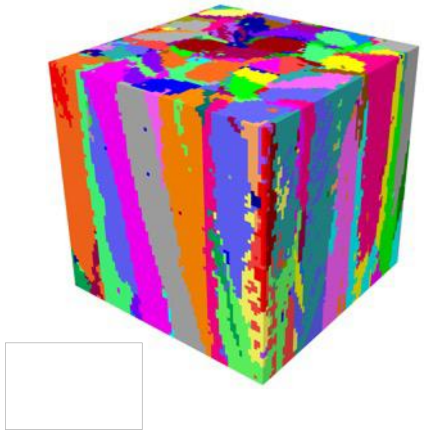
- Cellular Automata simulations produce microstructures with fine voxel sizes (1.667 microns)
  - Results in large number of elements for our simulations
    - 500^3 micron sample had 27 million elements
  - Similar issues could come from either near-field high energy x-ray diffraction (nf-HEXD) or 3D-serial sectioned EBSD mapped microstructures
- Coarsening voxel data set reduces this issue
  - But what effects does this have on property calculations and intragrain heterogeneity responses?
- We’re examining four levels of coarsening
  - Coarsen large microstructure 2, 3, 4, and 6 levels
  - Levels refer here to number of neighbor voxels averaged into 1 voxel in coarser mesh



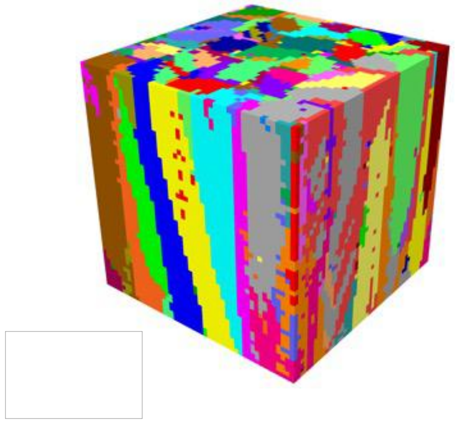
Original Microstructure



Coarsen L2 Microstructure



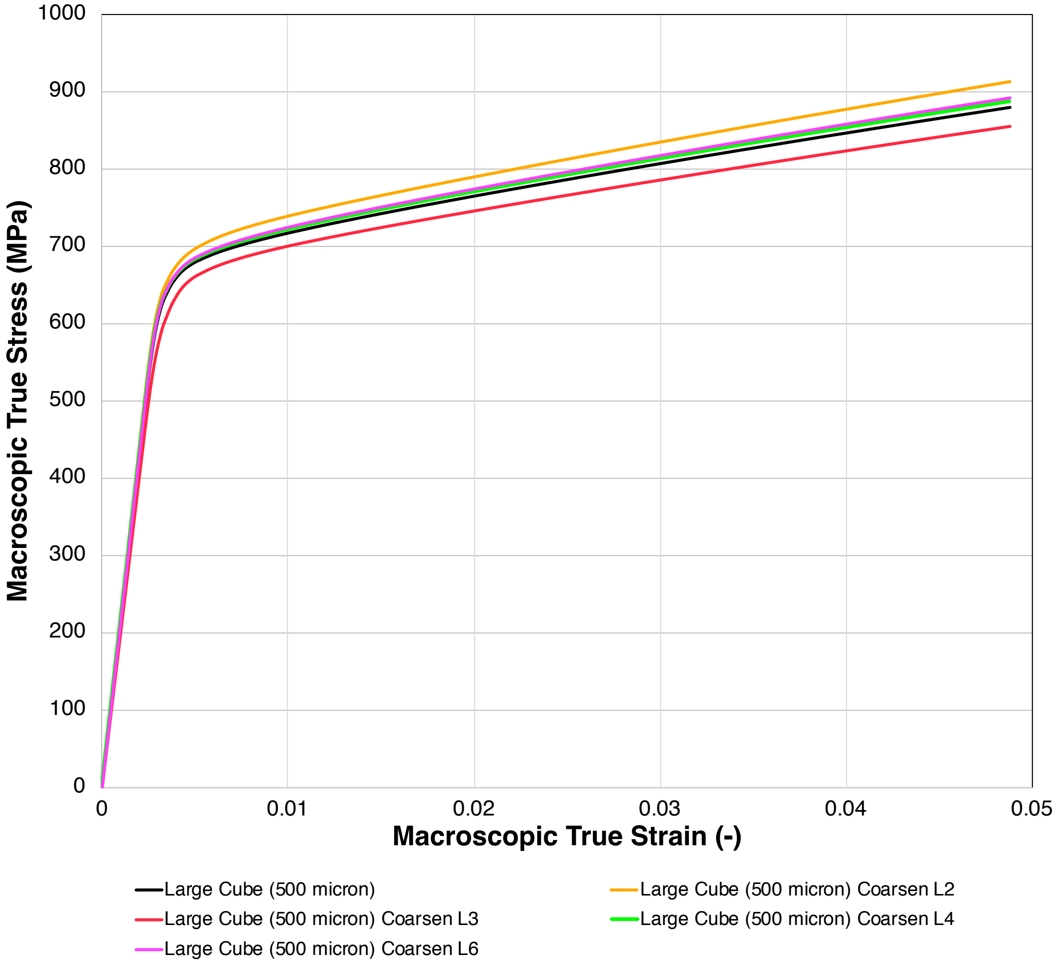
Coarsen L4 Microstructure



Coarsen L6 Microstructure

Macroscopic Stress-Strain Response

- Uniaxial tension tests out to 5% were taken on these samples
  - Symmetry boundary conditions were applied to all samples
- Stress-strain responses differed by at most 3% from the original mesh for all cases
- Crystallographic texture driving similarities in response
  - Voce hardening model lacks size dependence which could affect the trends seen here

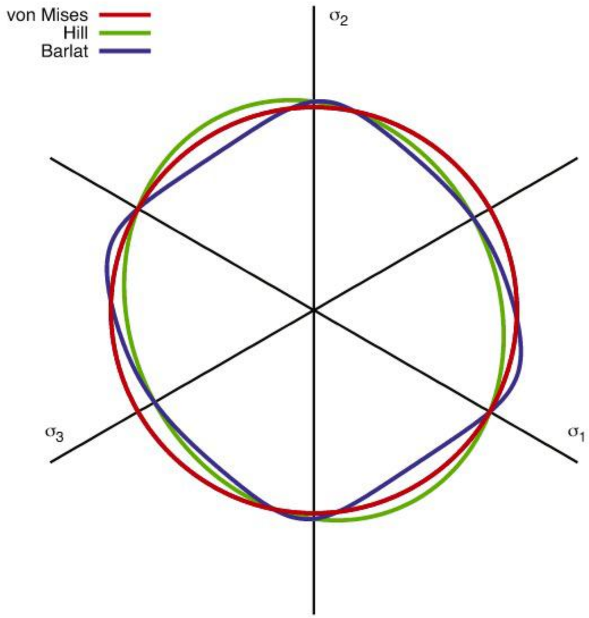


Macroscopic stress strain curves for different size microstructures

EXASCALE COMPUTING PROJECT

Why an anisotropic yield surface?

- If you're dealing with a strongly textured material a J2-type model probably isn't going to cut it
  - Rolling process to create sheet metal is well known for creating a textured materials that have orthotropic like behavior
- As-built AM materials can develop strong textures in the build direction
  - Post-build heat treatments and manufacturing processes such as HIP (high isostatic pressure) can reduce this issue
- Examine how varying the anisotropic properties of the material affects desired performance of a part



Yield surface comparison in the π-plane for 2090-T4 Al parameterization given in [9,10]

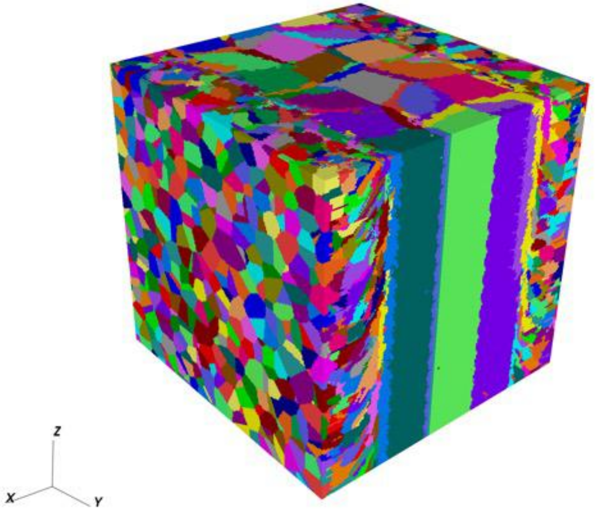
[9] F. Barlat *et al* 2005 *Int. J. Plasticity* **21** 1009-1039 [10] W.M. Schrezinger 2017 *Comput. Methods Appl. Mech. Eng.* **317** 526-553

Barlat Yield Surface Model (Yld2004-18p)

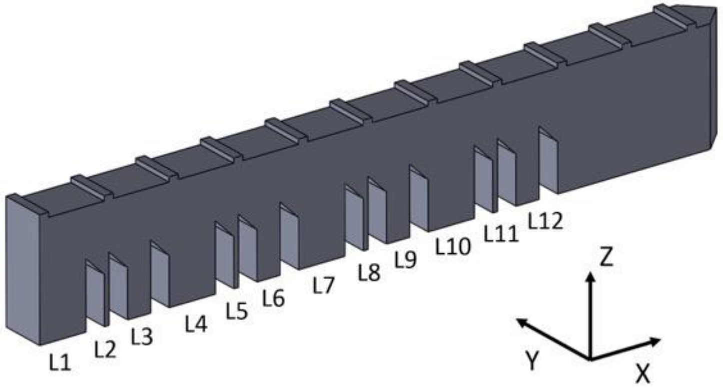
- Barlat and his associates have created several yield surfaces over the year to improve modelling of manufacturing processes
  - Sheet metal processes was one of the largest drivers in this process
- The Yld2004-18p model is one of his more famous ones
  - 18 parameter model that’s largely orthotropic with some out of plane behavior
  - Nice properties in that it can reduce to several famous models such as J2, Hill, and Tresca yield surface model
  - As of end of 2021, we have this implemented in one of LLNL’s in-house material library
    - An open-source version will be coming to ExaCMech within a year or so

Implementation into an AM workflow

- Barlat model has traditionally been parameterized using experimental data
  - Data is not always easy to obtain nor cheap but simulations are “cheap”
  - Crystal plasticity simulations to the rescue
- ExaConstit is going to drive our simulations along with some workflow tools and python
  - Workflow tools will eventually be released as part of ExaConstit repo (contact if interested in them)
- Microstructure provided by an outside code, ExaCA
- Simple SciPy optimization function used to reduce error between the yield function predicted by Barlat model and outputted simulation response
  - Certain load conditions also make use of r-factor

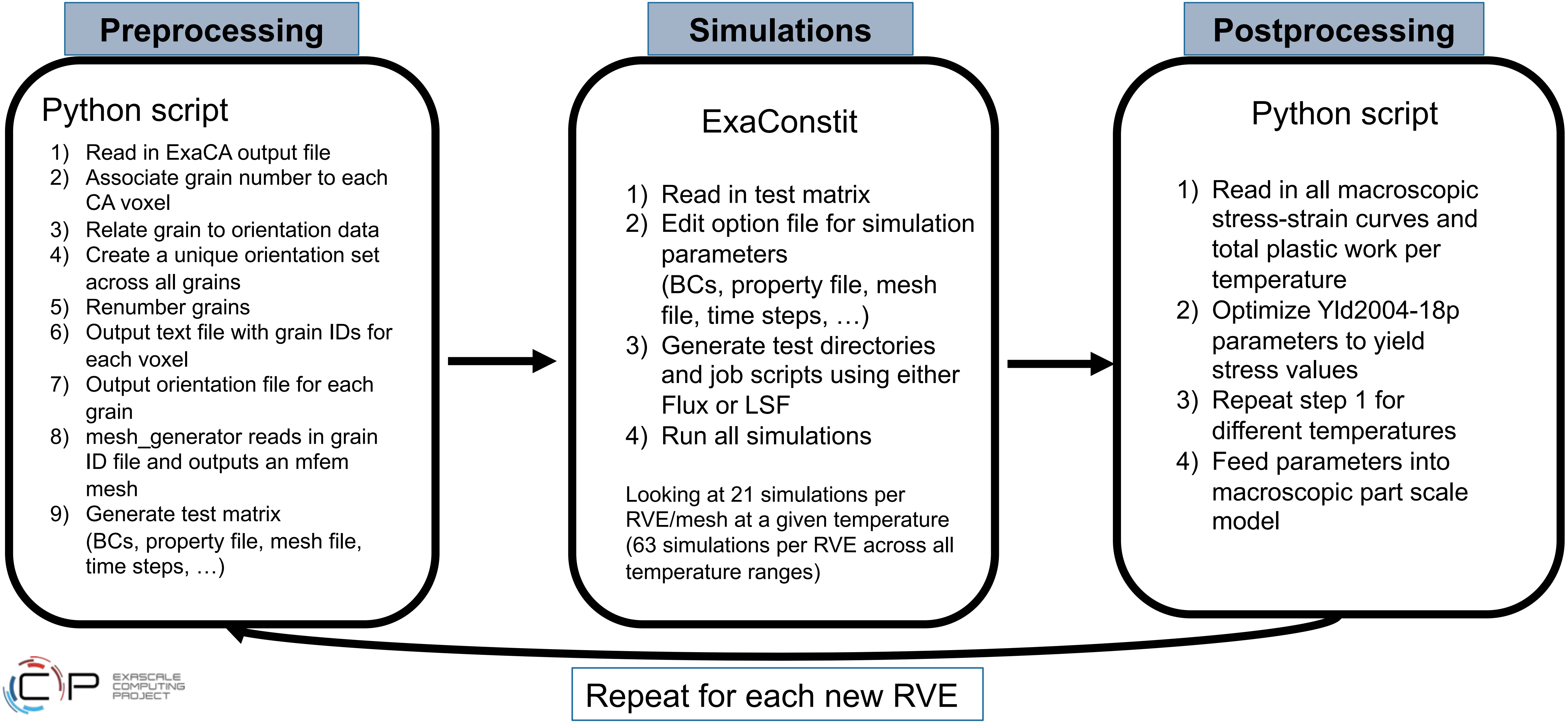


Microstructure of interest
(L8 leg down below*)



We are collaborating with the ExaWorks team to enable execution of ExaAM workflow on Exascale platforms.

- Flux-based implementation of ExaCA-ExaConstit workflow resulted in a 2x improvement



Preprocessing

Python script

1) Read in ExaCA output file
2) Associate grain number to each CA voxel
3) Relate grain to orientation data
4) Create a unique orientation set across all grains
5) Renumber grains
6) Output text file with grain IDs for each voxel
7) Output orientation file for each grain
8) mesh_generator reads in grain ID file and outputs an mfem mesh
9) Generate test matrix (BCs, property file, mesh file, time steps, …)

Simulations

ExaConstit

1) Read in test matrix
2) Edit option file for simulation parameters (BCs, property file, mesh file, time steps, …)
3) Generate test directories and job scripts using either Flux or LSF
4) Run all simulations

Looking at 21 simulations per RVE/mesh at a given temperature (63 simulations per RVE across all temperature ranges)

Postprocessing

Python script

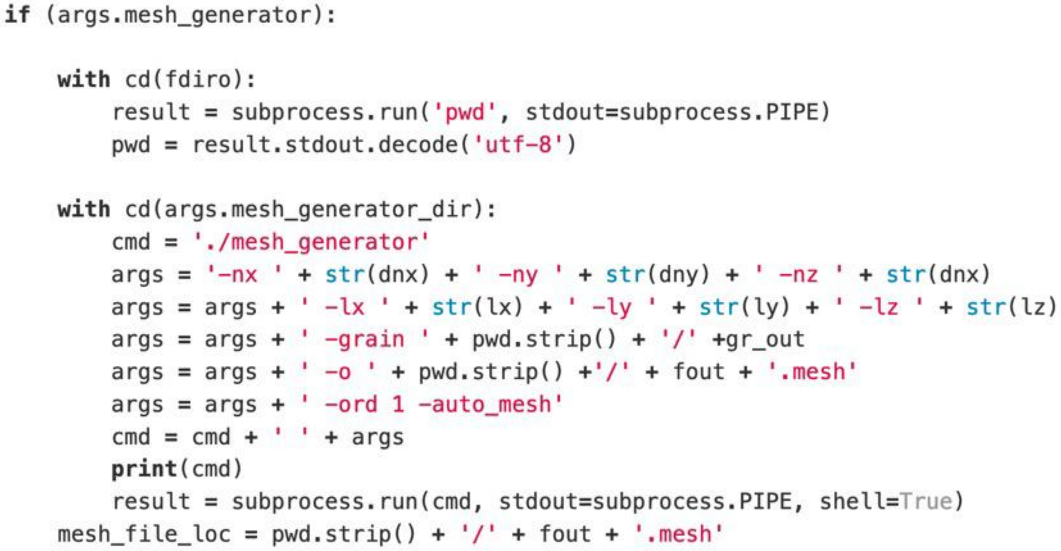
1) Read in all macroscopic stress-strain curves and total plastic work per temperature
2) Optimize Yld2004-18p parameters to yield stress values
3) Repeat step 1 for different temperatures
4) Feed parameters into macroscopic part scale model

Repeat for each new RVE

Preprocessing Step

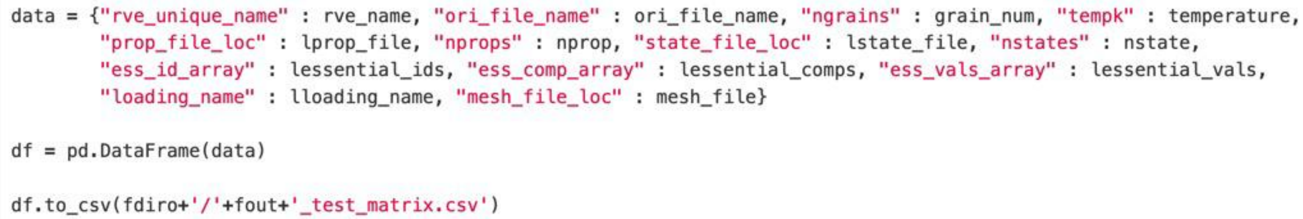
- Python CLI script takes in a microstructure, generates a mesh, and creates a test matrix
  - Provided multiple temperatures and their corresponding property files
  - Can coarsen microstructure if needed
  - Test matrix generated will be later slurped up by job generation script

```
python3 ./exaconstit_cli_preprocessing.py -ifdir ./ -ifile exaca.csv -ofdir ./output_dir/ -runame
super_cool_microstructure -c 1 -mg -mgdir ./ -t 298.0 -fprops ./props_cp_voce_in625.txt -nprops 17 -fstate
./state_cp_voce.txt -nstates 24
```



```
if (args.mesh_generator):

    with cd(fdiro):
        result = subprocess.run('pwd', stdout=subprocess.PIPE)
        pwd = result.stdout.decode('utf-8')

    with cd(args.mesh_generator_dir):
        cmd = './mesh_generator'
        args = '-nx ' + str(dnx) + ' -ny ' + str(dny) + ' -nz ' + str(dnx)
        args = args + ' -lx ' + str(lx) + ' -ly ' + str(ly) + ' -lz ' + str(lz)
        args = args + ' -grain ' + pwd.strip() + '/' +gr_out
        args = args + ' -o ' + pwd.strip() +'/' + fout + '.mesh'
        args = args + ' -ord 1 -auto_mesh'
        cmd = cmd + ' ' + args
        print(cmd)
        result = subprocess.run(cmd, stdout=subprocess.PIPE, shell=True)
    mesh_file_loc = pwd.strip() + '/' + fout + '.mesh'
```

Mesh generation code



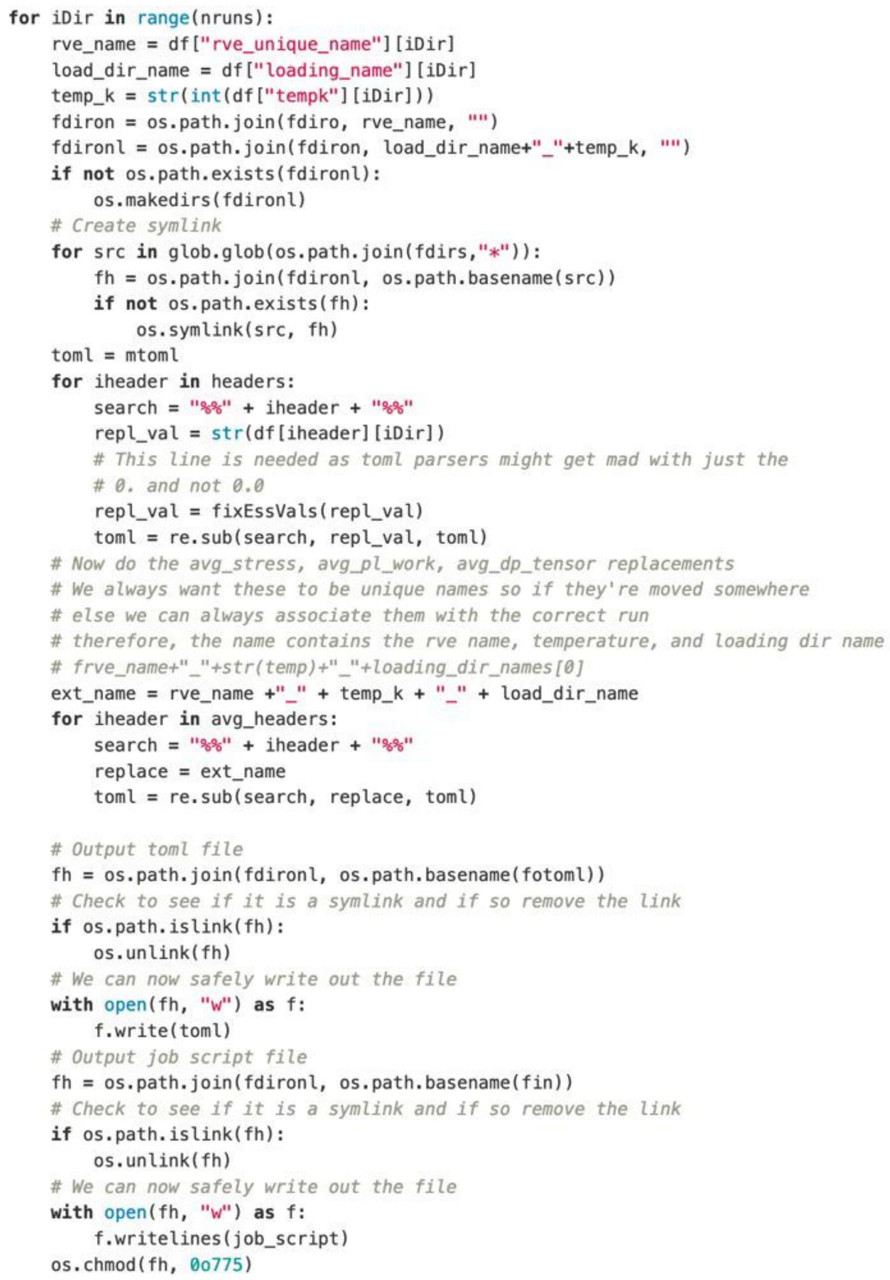
```
data = {"rve_unique_name" : rve_name, "ori_file_name" : ori_file_name, "ngrains" : grain_num, "tempk" : temperature,
        "prop_file_loc" : lprop_file, "nprops" : nprop, "state_file_loc" : lstate_file, "nstates" : nstate,
        "ess_id_array" : lessential_ids, "ess_comp_array" : lessential_comps, "ess_vals_array" : lessential_vals,
        "loading_name" : lloading_name, "mesh_file_loc" : mesh_file}

df = pd.DataFrame(data)

df.to_csv(fdiro+'/'+fout+'_test_matrix.csv')
```

Test matrix creation

Job Generation Step

- Python CLI script takes in a test matrix, master option file, and a job submission file
  - Each simulation gets a unique job directory that has symlinks to original shared data
  - Master option file has regex searchable expressions in it that are replaced with test matrix values
  - Supports LSF or Flux job submission systems
    - Flux only required an additional 5 lines of code…
  - Generates a master job script that will submit all jobs for you
- Job generation could easily be extended to parametric studies

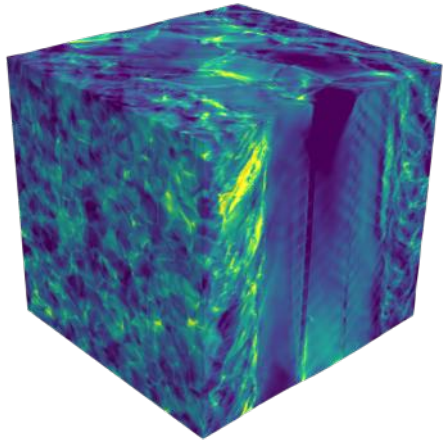
```
python3 ./job_cli.py –sdir ./ –odir ./../workflow_runs/ –
imtfile options_master.toml –iotfile options.toml –ijfile
hip_mechanics.flux –ijfd ./ –iofile options.csv
```



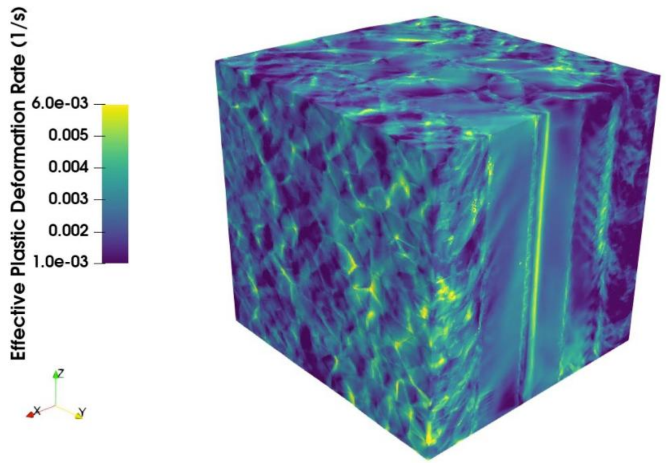
```python
for iDir in range(nruns):
    rve_name = df["rve_unique_name"][iDir]
    load_dir_name = df["loading_name"][iDir]
    temp_k = str(int(df["tempk"][iDir]))
    fdiron = os.path.join(fdiro, rve_name, "")
    fdironl = os.path.join(fdiron, load_dir_name+"_"+temp_k, "")
    if not os.path.exists(fdironl):
        os.makedirs(fdironl)
    # Create symlink
    for src in glob.glob(os.path.join(fdirs,"*")):
        fh = os.path.join(fdironl, os.path.basename(src))
        if not os.path.exists(fh):
            os.symlink(src, fh)
    toml = mtoml
    for iheader in headers:
        search = "%%" + iheader + "%%"
        repl_val = str(df[iheader][iDir])
        # This line is needed as toml parsers might get mad with just the
        # 0. and not 0.0
        repl_val = fixEssVals(repl_val)
        toml = re.sub(search, repl_val, toml)
    # Now do the avg_stress, avg_pl_work, avg_dp_tensor replacements
    # We always want these to be unique names so if they're moved somewhere
    # else we can always associate them with the correct run
    # therefore, the name contains the rve name, temperature, and loading dir name
    # frve_name+"_"+str(temp)+"_"+loading_dir_names[0]
    ext_name = rve_name +"_" + temp_k + "_" + load_dir_name
    for iheader in avg_headers:
        search = "%%" + iheader + "%%"
        replace = ext_name
        toml = re.sub(search, replace, toml)

    # Output toml file
    fh = os.path.join(fdironl, os.path.basename(fotoml))
    # Check to see if it is a symlink and if so remove the link
    if os.path.islink(fh):
        os.unlink(fh)
    # We can now safely write out the file
    with open(fh, "w") as f:
        f.write(toml)
    # Output job script file
    fh = os.path.join(fdironl, os.path.basename(fin))
    # Check to see if it is a symlink and if so remove the link
    if os.path.islink(fh):
        os.unlink(fh)
    # We can now safely write out the file
    with open(fh, "w") as f:
        f.writelines(job_script)
    os.chmod(fh, 0o775)
```

Main code logic

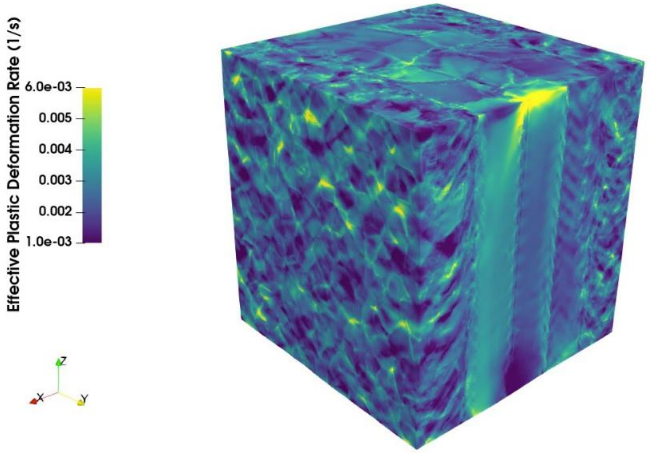
(Aside) AM microstructures have some crazy heterogeneous deformation



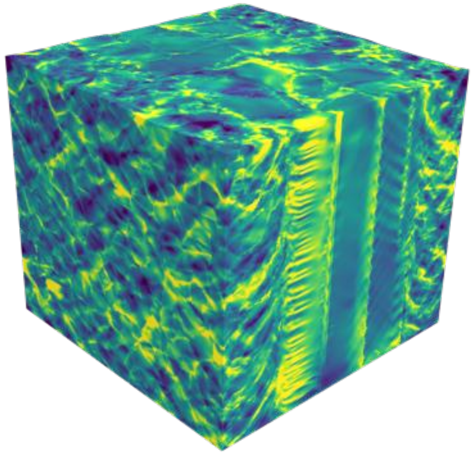
Loading x direction



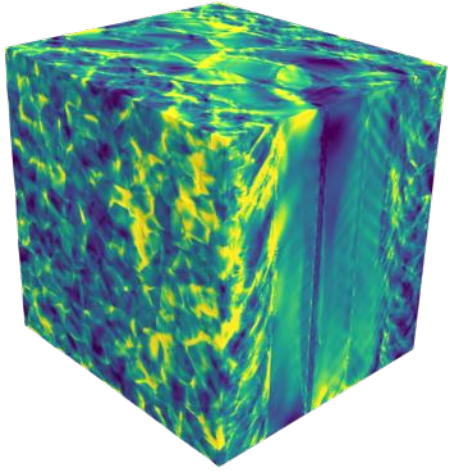
Loading y direction



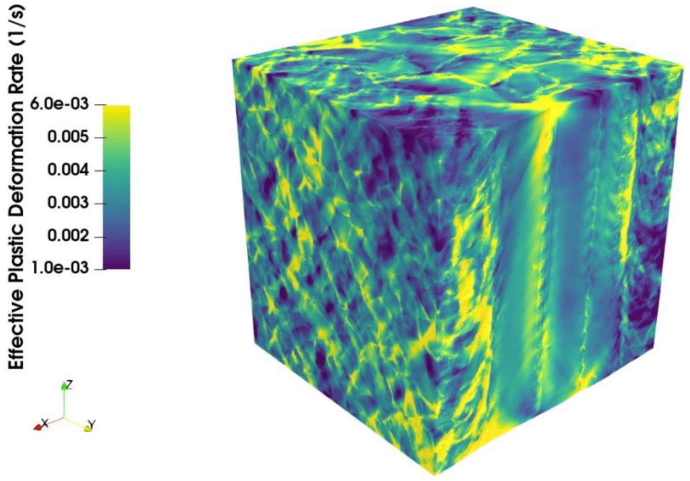
Loading z direction



Equal biaxial loading
x-y direction

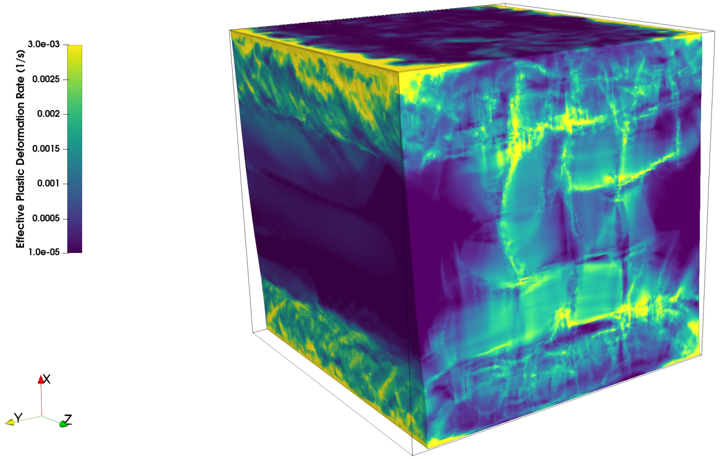


Equal biaxial loading
y-z direction

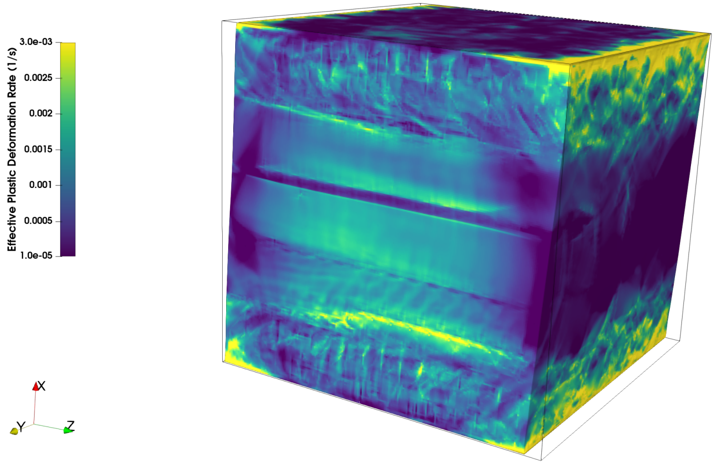


Equal biaxial loading
x-z direction

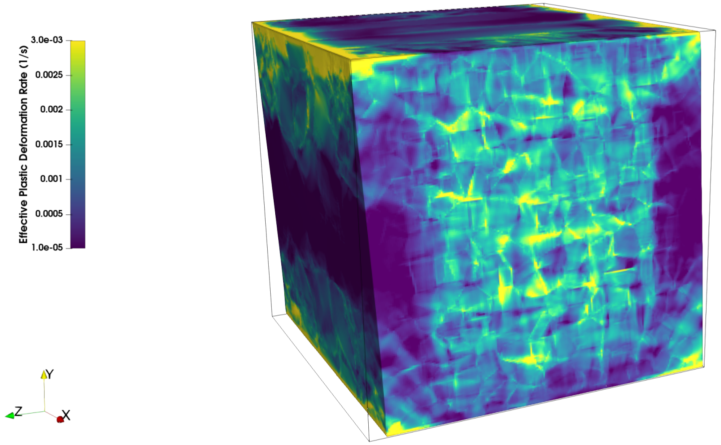
(Aside) AM microstructures have some crazy heterogeneous deformation



Simple shear xy



Simple shear xz



Simple shear yz

Optimization Procedure

- For each temperature load up all volume average stress-strain responses and total volume plastic work values
  - Loading in the build direction (z-axis) is considered ground-truth for optimization purposes
  - Calculate 0.2% off-set yield and the plastic work associated with that
  - For all simulations find strain step closest to plastic work produce above and use associated stress value to calculate von Mises stress to be used in optimization procedure
  - Let optimization function do its thing (go grab some coffee/do other work)
    - Later temperatures use previous temperature parameters as initial guess to get within ball park

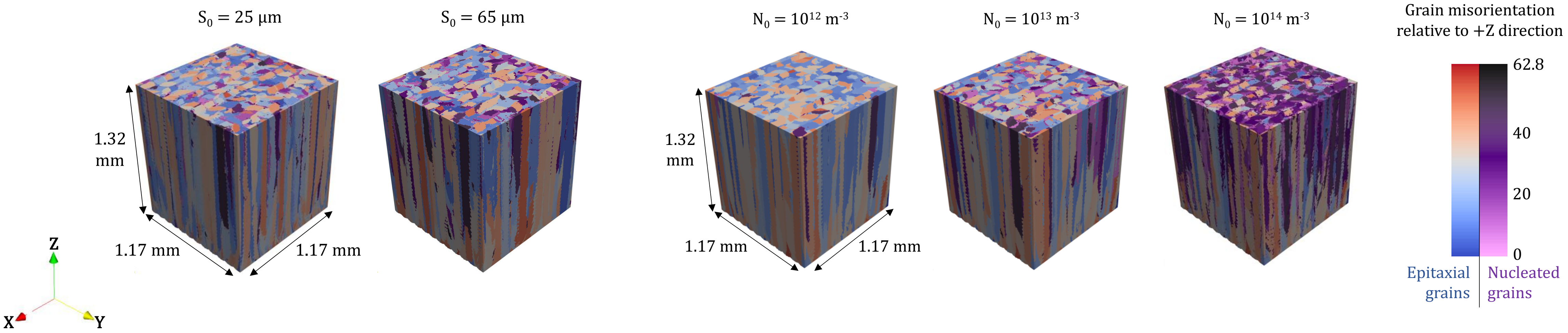
ExaConstit in other areas

- ExaConstit can be used in areas outside of just the AM field
  - Quantifying variability in microstructure effects on material response
  - Fatigue applications and understanding deformation mechanisms at micro-scale is a large research area
  - Creating better material models to capture single crystal behavior
  - Modelling geological materials to understand how Earth’s lower mantle deforms
  - Coupling experiments and simulations by using forward diffraction techniques
  - Modelling texture evolution of materials over large deformations

Variability in processing conditions and effects on mechanical response

- Varying processing conditions can lead to different microstructures which can effect the micro-mechanical response of a material
  - Wide parameter space so usually best to focus on only a few aspects

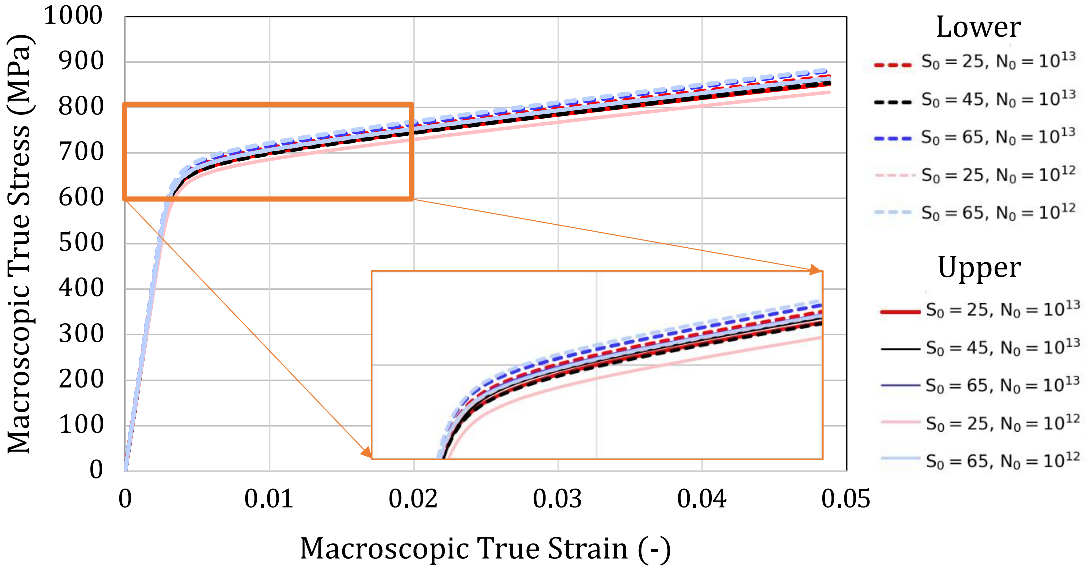
- ExaAM team conducted a study on varying parameters that are inputted into ExaCA, microstructure generation code, and effects on micro-mechanical response for AM processes



Variation of AM substrate diameter and nucleation rate effects on microstructure

Variability in processing conditions and effects on mechanical response

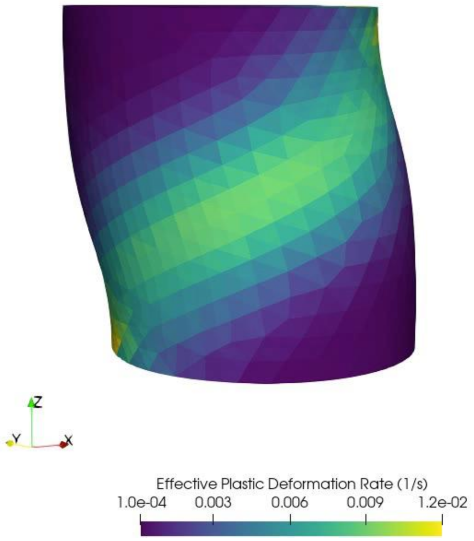
- Varying the substrate diameter and nucleation rate does affect the macroscopic response of the material
  - In cases examined, a difference of ~6% was noted between the min and max stress-strain responses
- Variation within the intragrain response was also noted and a larger variability was noted here as well

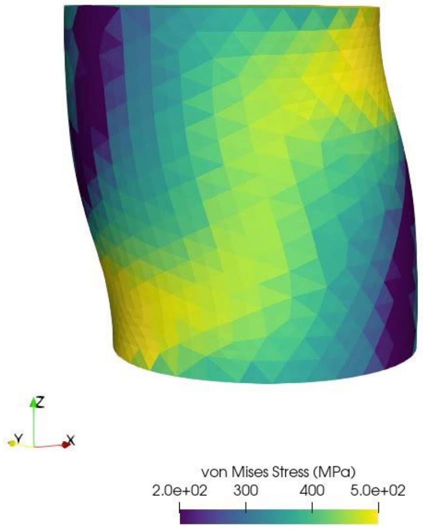


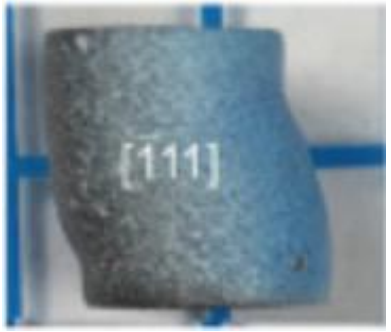
Variation of AM substrate diameter and nucleation rate effects on macroscopic response

Simulating single crystal responses

- Capturing single crystal response of metals is often challenging as traditional crystal plasticity models were designed to capture response of polycrystalline materials
- We have an LDRD set-up to better examine this area and our working on some promising new model formulations
  - Capture buckling behavior as seen in experimental results as well as stress-strain response







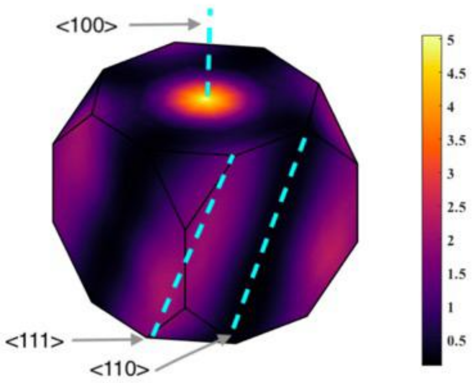
Crystallographic Texture Evolution

- Crystallographic texture informs of us the orientation dependence of a part in a given direction
- Texture plays a large role in determining how a part will deforms
- Manufacturers use texture to their advantage
  - Gas turbine blades are oriented in the <100> direction to improve fatigue life [11]
  - Accounting for texture allows deep forming operations to reduce waste [12]
- Large texture development occurs over large applied strains
  - ExaAM is not concerned with strains typically needed for texture development

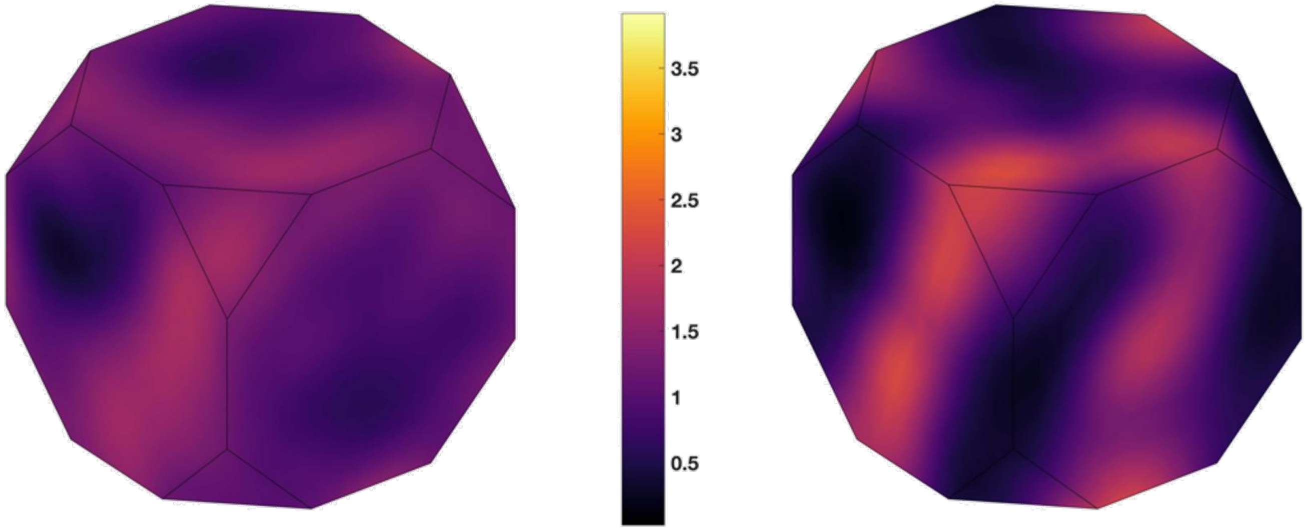
[11] G. Swanson and NK. Arakere 2000 *NASA/TP-2000-210074*
[12] P.R. Dawson et al. 2003 *Int. Mater. Rev.* **48** 86-122

Crystallographic Texture Evolution of AM parts

- A CA microstructure sample was monotonically compressed to 30% strain in order to develop a strong texture in the <110> direction



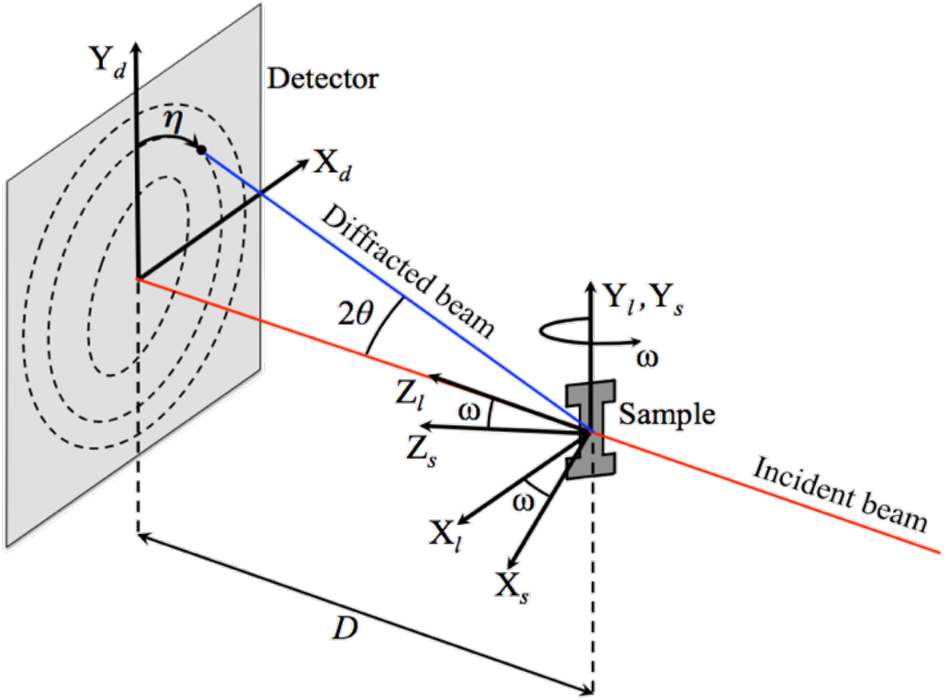
Example of strongly textured material with various fibers labelled [13]



Initial (Left) and final (Right) crystallographic texture of the CA sample

[13] R. Carson and P. Dawson 2019 *JMPS* **126** 1-19

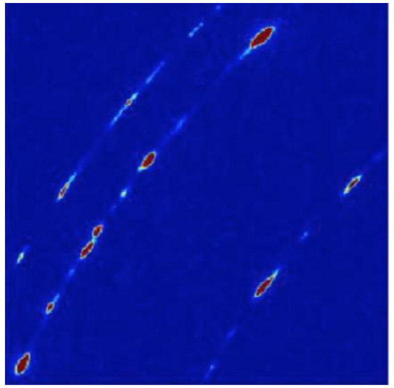
Simulating High Energy X-ray Diffraction Experiments

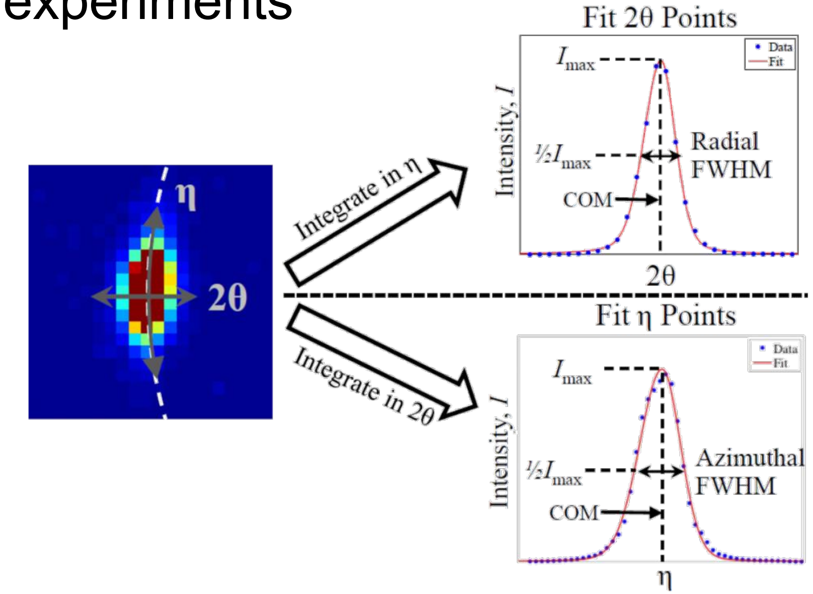


The peak data includes information about strain and lattice orientation of a grain

Simulating High Energy X-ray Diffraction Experiments

- Peak data is integrated in ω direction and either θ or η direction
  - Provides us with info about relative amounts of intragrain heterogeneity in either orientation, η spread, or strain, θ spread.
  - Provides direct comparison to experiments





Spread in peak data tells us relative amounts of intragrain heterogeneity in the elastic deformation

[14] M. Obstalecki *et al* 2014 *Acta Materialia*, **75** pp 259 – 272.

42

Summary

- ExaConstit is an open-source crystal-plasticity FEM code built on MFEM
  - Highly performant on systems ranging from desktops all the way to systems such as Summit
  - Implemented several different integrators for GPU support and different materials models
- Created a workflow in-collaboration with ExaWorks team to efficiently run large number of simulations needed to calculate local properties used in ExaAM’s part scale simulations
  - Led to anisotropic yield surface model being added to one of LLNL’s material modelling libraries
- ExaConstit is well poised to tackle a wide range of research topics within the crystal plasticity community

Questions?

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

Lawrence Livermore
National Laboratory