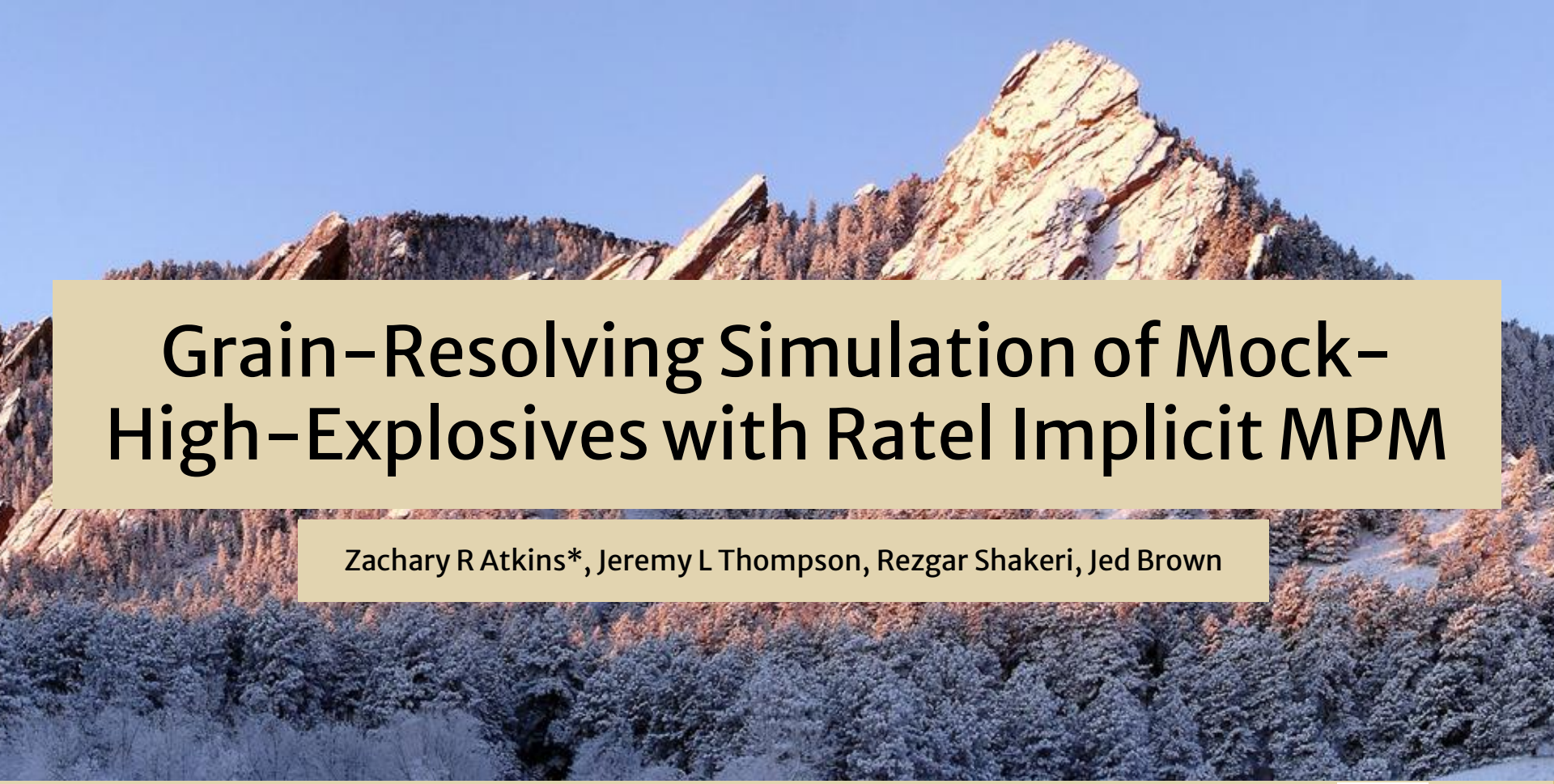


Grain-Resolving Simulation of Mock-High-Explosives with Ratel Implicit MPM

Zachary R Atkins*, Jeremy L Thompson, Rezgar Shakeri, Jed Brown

National Nuclear Security Administration

University of Colorado Boulder

COLORADO SCHOOL OF MINES

THE UNIVERSITY OF TENNESSEE KNOXVILLE

Outline

- Mock-HE Manufacturing Problem
- Implicit Material Point Method Formulation
- Challenges with Grain-Resolving Scale

Mock-High Explosive (HE) Manufacturing

- **Grains** of mock-HE crystals (IDOX) are coated with a nitro-plasticized polymer **binder** (Estane) to form **prills**
- By volume, each prill is 95% grain, 5% binder
  - This will be an issue in a few slides!

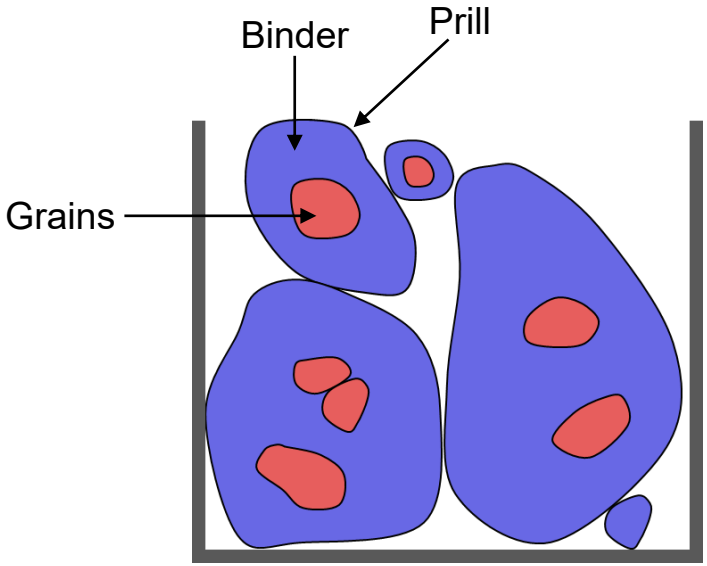


Image credits: Zachary White, CU Boulder

Mock-High Explosive (HE) Manufacturing

- **Grains** of mock-HE crystals (IDOX) are coated with a nitro-plasticized polymer **binder** (Estane) to form **prills**
- By volume, each prill is 95% grain, 5% binder
  - This will be an issue in a few slides!
- Prills are poured into a 5mm diameter cylindrical die, 8.5mm high unpressed
  - At this point, prills are *very* loosely packed – about 30% void
- Compressed to 3kPa loading force, ~5mm height pressed

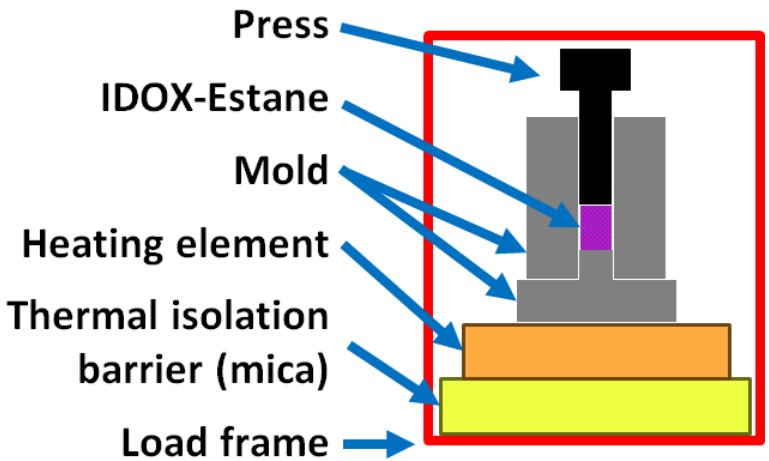
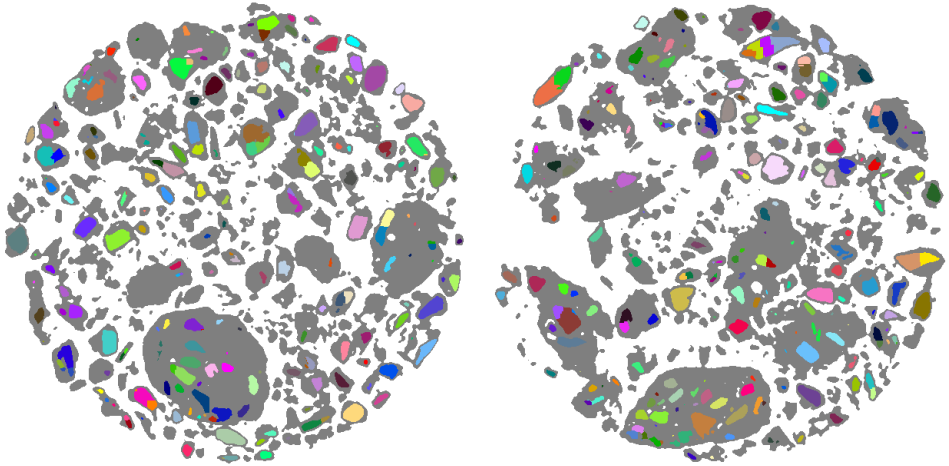


Image credits: Zachary White, CU Boulder; Trevor Lyons, Mines

Resolving Grains – Experimental Data

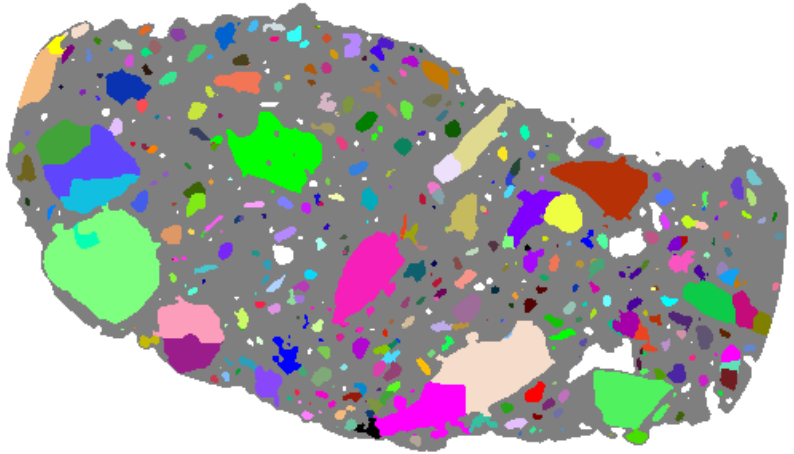
- Before pouring into the die, first pour into a plastic cylinder and perform computed tomography (CT) scan
- After post-processing, have a "stack" of TIFF image files
  - Each pixel is ~8.5um cubed
  - Value of each pixel is
    - 0: Void
    - 1: Binder
    - 2-N: Labeled grain (up to ~14000)
  - Roughly 400 million voxels total
  - We only care about ~290 million voxels



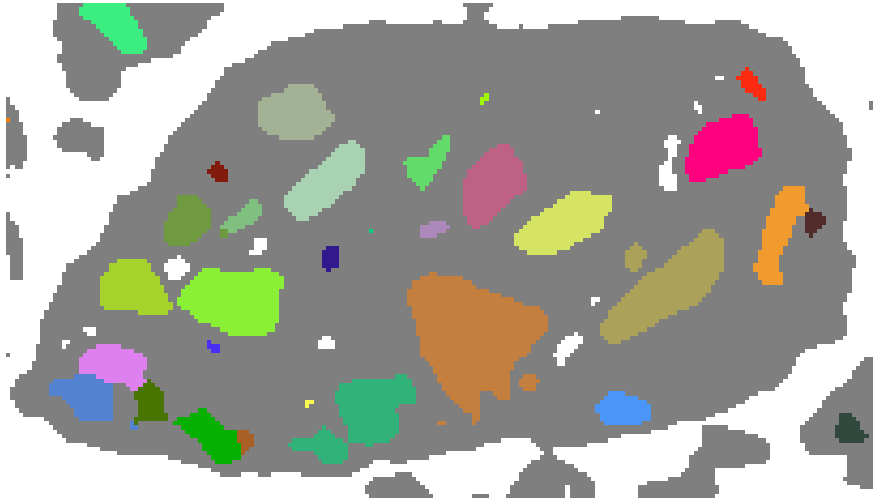
Sample horizontal slices of processed CT data
Gray: Binder, Colors: Grains
Credit: Nate Peterson (Mines), Gus Becker (CU Boulder)

Resolving Grains – Major Issue

- Problem: Where are all the grains?
  - Recall, each prill should be 95% grain, 5% binder



CT data of a real prill (grains still under-resolved)
Gray: Binder, Colors: Grains



Zoomed-in CT data of full press, single prill
Gray: Binder, Colors: Grains

Resolving Grains – Solutions

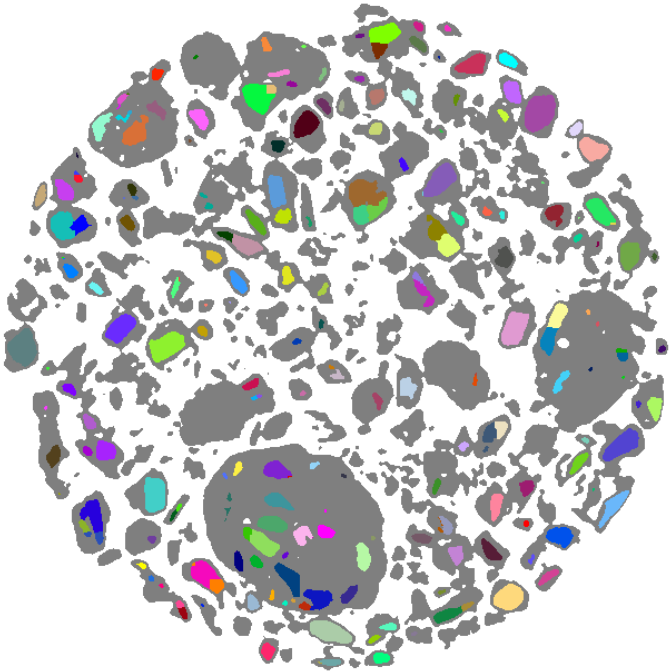
- Current solution: represent Estane binder + fine grains as homogenized material
  - Expect the material properties to be more like the grains
  - Limits our ability to accurately model the binder

- Ongoing work with collaborators:
  - CT scans of individual prills (Trevor Lyons, Mines)
  - Generate artificial prills (Jarett Poliner, Columbia)
  - Simulate a pour of the artificial prills (LAMMPS – Zachary White, CU Boulder)
  - Load simulated pour data into Ratel (Jeremy Thompson & me, CU Boulder)

Why MPM?

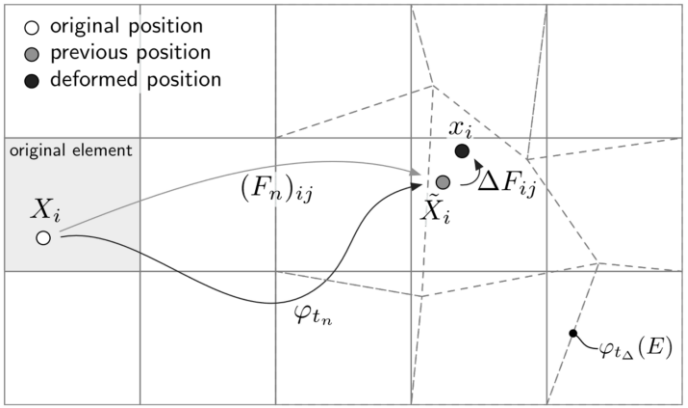
FEM just won't work!

- Highly heterogeneous material
  - Smaller grains cannot be meshed easily
  - Resulting mesh is poorly conditioned and must be smoothed
- Large void regions
  - Likely to cause overlap/inversion of cells
  - Modeling internal contact would be very difficult and expensive
- Very large deformations
  - Need to compress >40% of volume
  - High likelihood of extreme mesh distortion



Implicit MPM Formulation

- Previously-converged Lagrangian (Coombs et al. 2020)
- Currently, quasi-static time discretization



Reframe equilibrium equations to be with respect to $\widetilde{\Omega}$ at time t_n using stored deformation gradient $\boldsymbol{F}_n$

$$\boldsymbol{F} = \tilde{\boldsymbol{F}}\boldsymbol{F}_n, \quad J = \tilde{J}J_n, \quad \Delta\boldsymbol{F} = \tilde{\boldsymbol{F}},$$

$$J = \frac{d\mathrm{v}}{dV}, \quad \tilde{J} = \frac{d\mathrm{v}}{d\tilde{V}}, \quad J_n = \frac{d\tilde{V}}{dV},$$

$$\nabla\boldsymbol{u} = \nabla\boldsymbol{u}_n + \nabla\tilde{\boldsymbol{u}} + \nabla\tilde{\boldsymbol{u}} \cdot \nabla\boldsymbol{u}_n$$

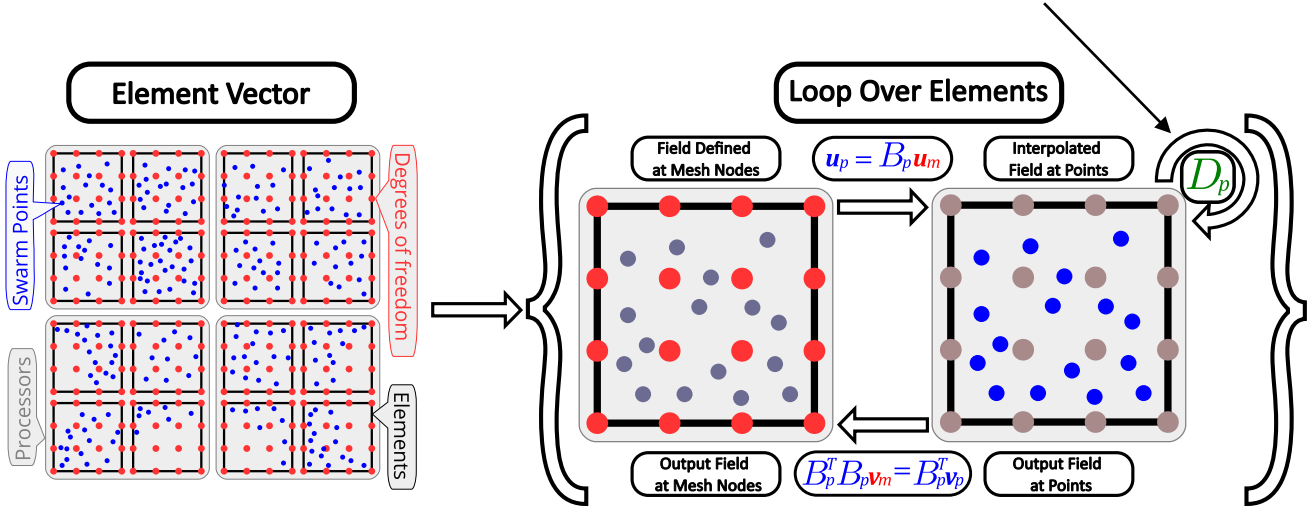
Solve for $\Delta\boldsymbol{u}$ such that

$$r(\Delta\boldsymbol{u}; \boldsymbol{u}_n) \coloneqq \int_{\widetilde{\Omega}} \nabla_{\tilde{X}}\boldsymbol{v} : \underbrace{\frac{1}{J_n}\boldsymbol{\tau}(\nabla\boldsymbol{u}^{\mathrm{p}})\tilde{\boldsymbol{F}}^{-T}}_{\tilde{\boldsymbol{P}}} \, \mathrm{d}\tilde{V} - \int_{\widetilde{\Omega}} \boldsymbol{v} \cdot \frac{1}{J_n}\rho_0\boldsymbol{g} \, \mathrm{d}\tilde{V} = \boldsymbol{0}$$

- We only use traditional material points, due largely to the parallelization strategy

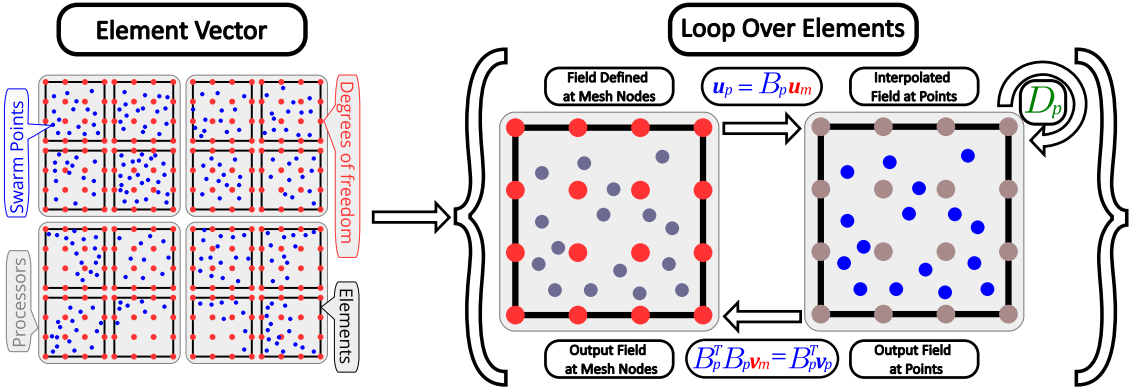
Implicit MPM Formulation

- Each material point must store $\nabla \boldsymbol{u}_n$ (9 components) as well as any state variables
- Constitutive properties, e.g., etc., are also stored per-point
  - E.g., density, bulk and shear moduli, fracture toughness
- Constitutive models are defined by action at a single material point (QFunction)



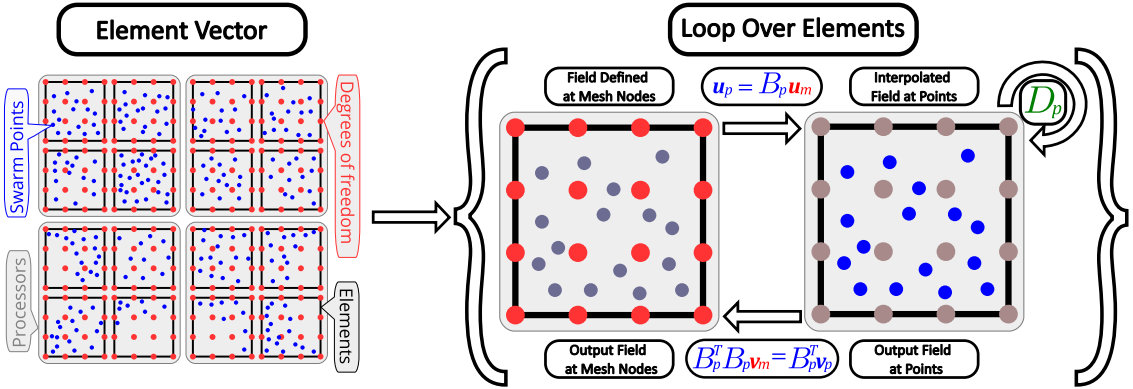
Matrix-Free Operators

- Strategy:
  - Group material points by cell
  - Use FEM basis functions to compute interpolation/gradient at nodes
  - Interpolate to points using Chebyshev interpolating polynomials
  - Apply constitutive model point-wise
  - Apply transpose interpolation operators to recover FEM output fields



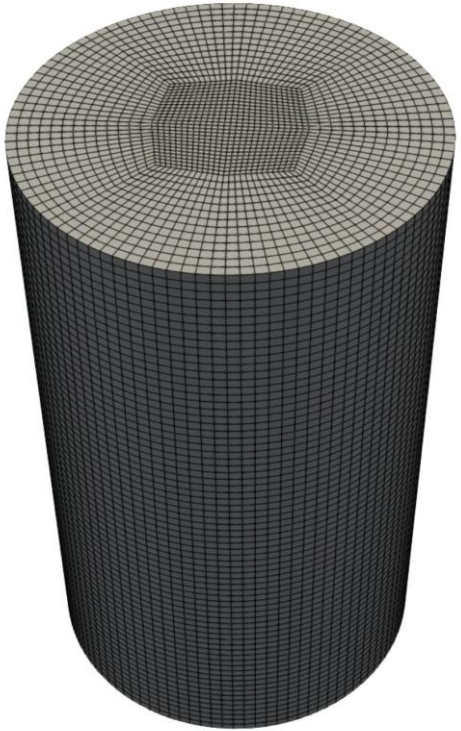
Matrix-Free Operators

- Advantages:
  - Utilizes the same matrix-free infrastructure used for FEM
  - Assembly for preconditioners is nearly identical as FEM
  - Take advantage of memory-limited, high-compute accelerators (i.e. GPUs)
  - Existing FEM constitutive models can be used with minor adjustments for extra stored state and material parameters



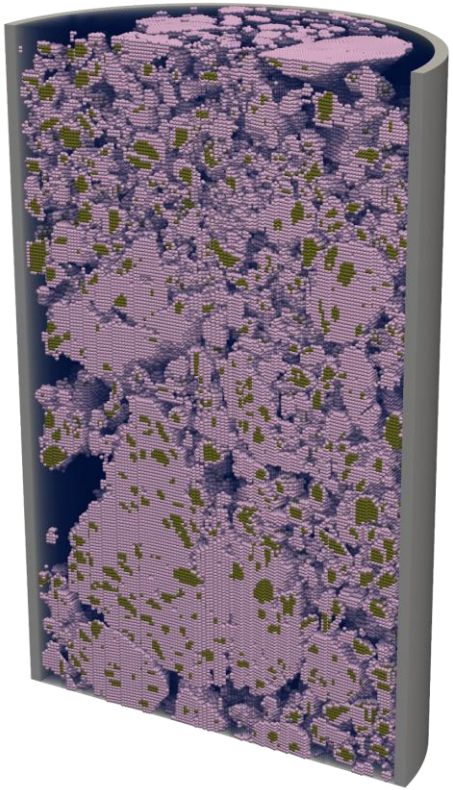
Simulation Setup – Meshing and Point Initialization

- Our MPM only supports hex meshes
  - Non-affine hexes are fine, though
- Externally generate a cylinder mesh, GMSH
  - Target an average characteristic length (here, 120 um)
- Place material points uniformly within each cell



Simulation Setup – Meshing and Point Initialization

- Our MPM only supports hex meshes
  - Non-affine hexes are fine, though
- Externally generate a cylinder mesh, GMSH
  - Target an average characteristic length (here, 120 um)
- Place material points uniformly within each cell
- Load voxel data into 3D array, one entry per pixel
- Locate each material point's enclosing voxel
  - The voxel size is used to rescale the mesh coordinates
- Set material properties based on the voxel value

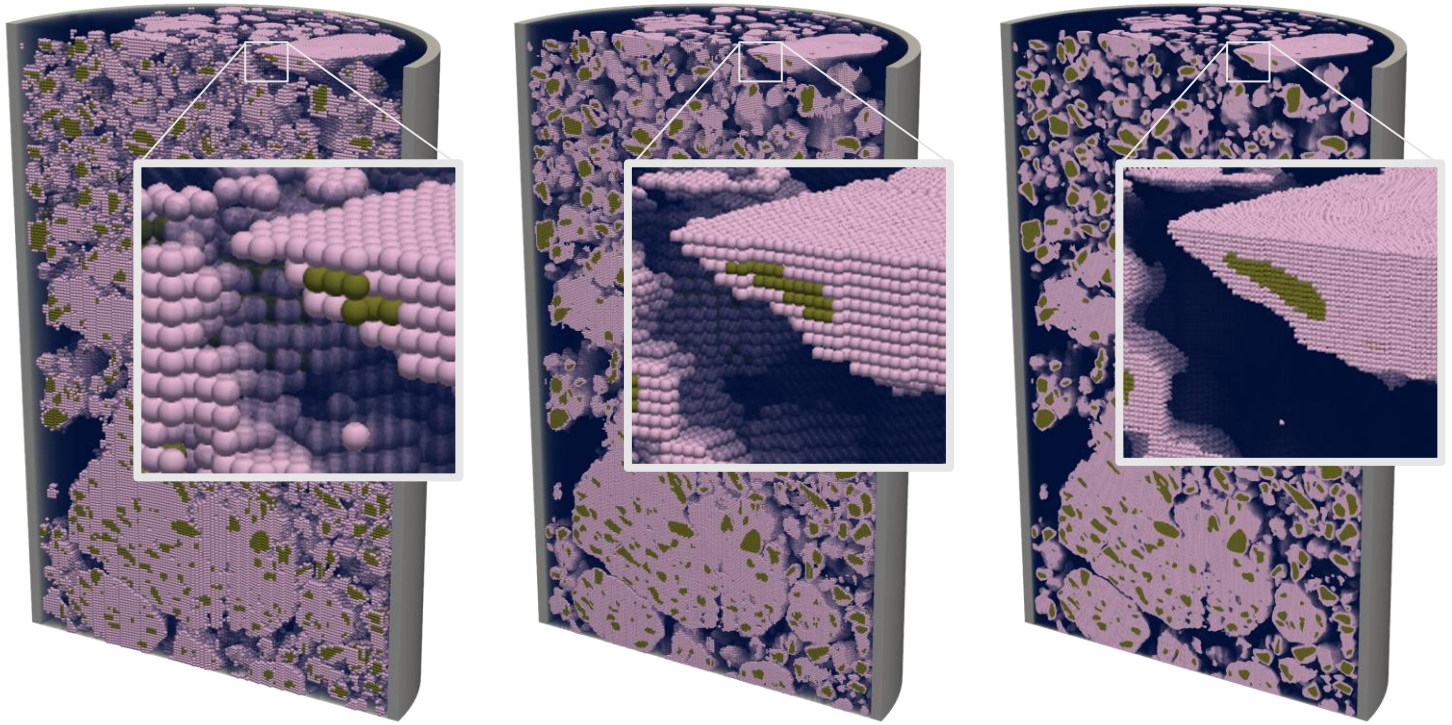


Simulation Setup – Meshing and Point Initialization

3.04 MPoints

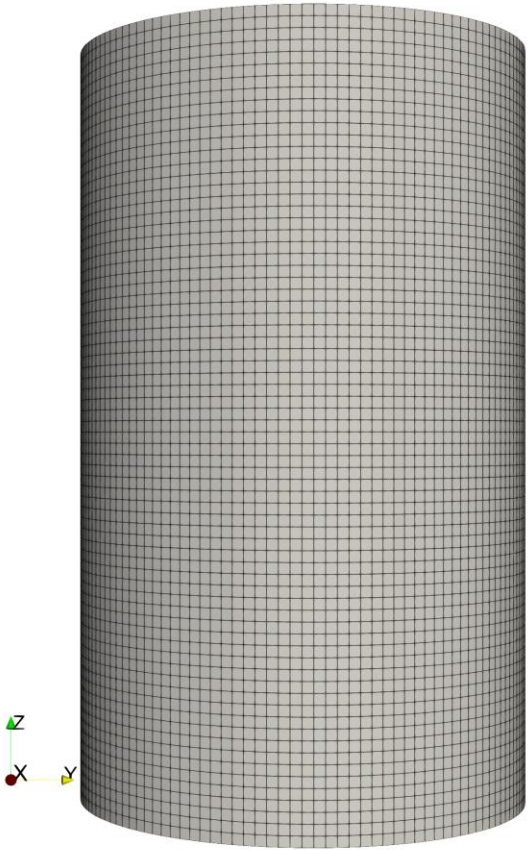
22.7 MPoints

316 MPoints



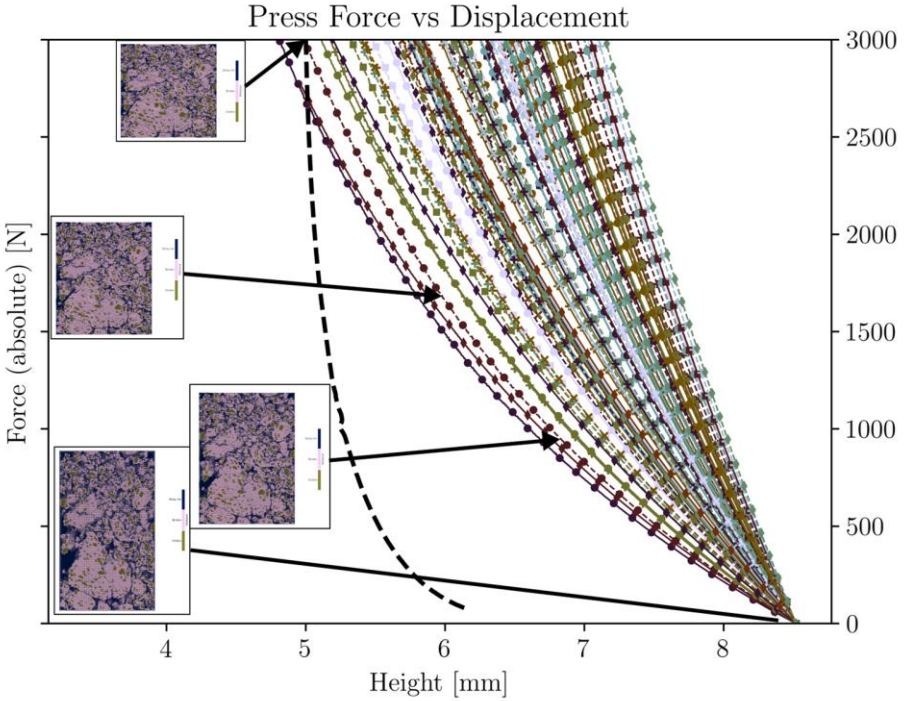
Boundary Conditions

- Ideally, we would use traction boundaries
  - This would ensure that we reach (and stop at) 3kPa
- But…traction boundaries are difficult to apply once points leave the boundary cells
  - Future work
- Instead, we use incrementally applied Dirichlet boundaries – outside and bottom are fixed
- Top is fixed in x-y directions and displaced by fixed increment in z
  - At the end of timestep, the mesh is transformed by the same increment



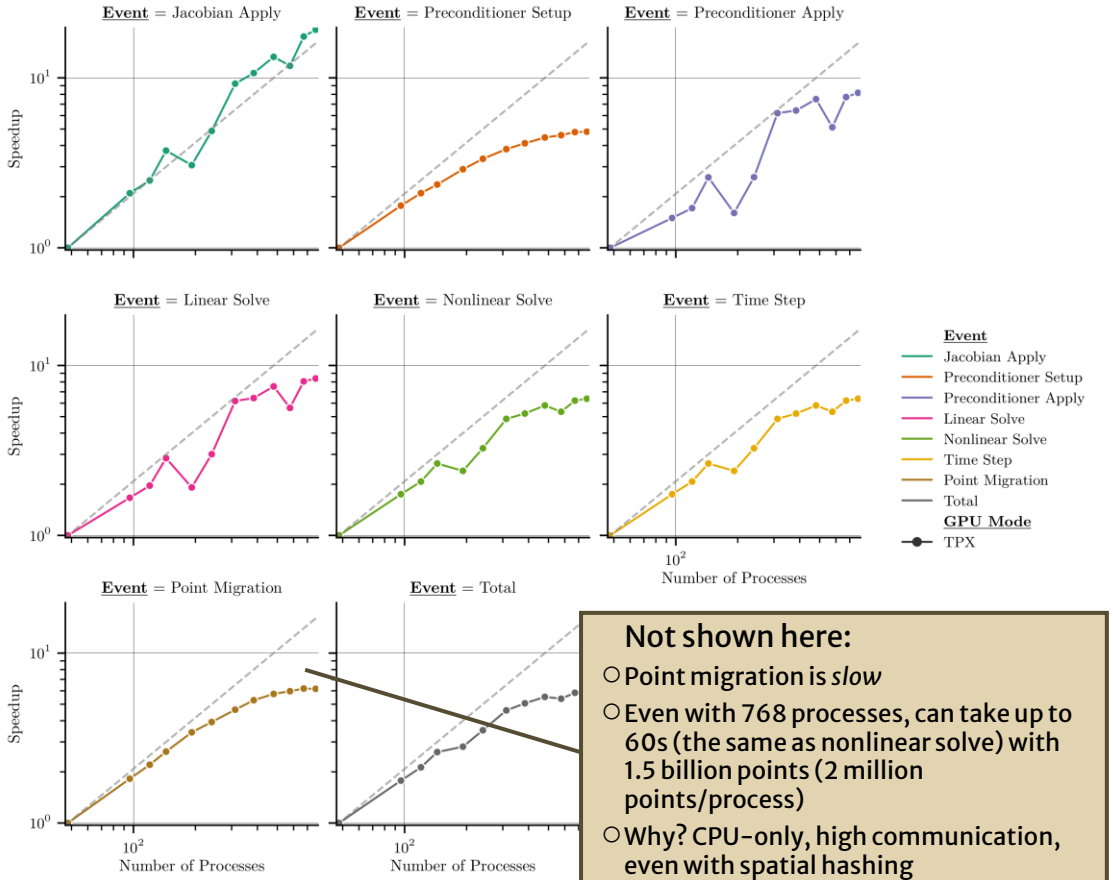
Quantities of Interest

- Force-Displacement, expect:
  - Compress ~40% to 5mm height
  - Peak loading force of 3kPa
- Determine:
  - Young's Modulus E for grains
  - Young's Modulus E for binder
- Want to add:
  - Viscoelastic parameters
  - Fracture toughness for phase-field damage
  - Plasticity (yield strength, etc.)



Performance

- Strong scaling
  - 400 million points
  - AMD MI300A APUs, 3 LGPUs/GPU, XNACK=1
  - 4–64 Tuolumne Nodes
- Solver configuration
  - PETSc GAMG multigrid preconditioner
  - PETSc BCGS(2) linear solver
  - PETSc Newton nonlinear solver, bisection line search
- Results
  - Matrix-free operators scale great
  - Preconditioner and point migration, not so much

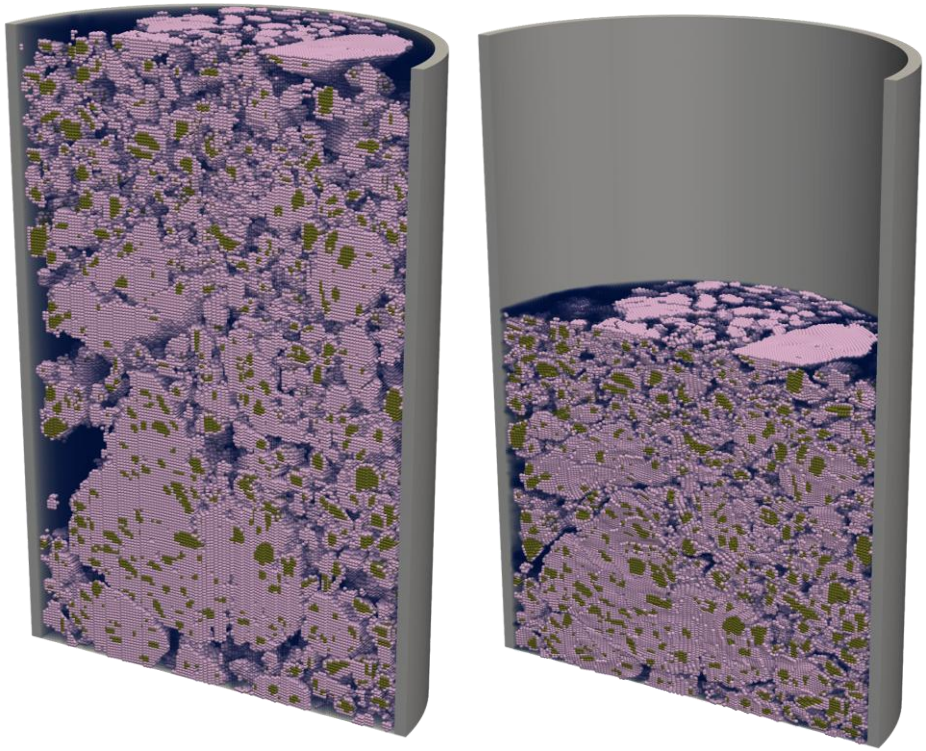


Not shown here:

- Point migration is *slow*
- Even with 768 processes, can take up to 60s (the same as nonlinear solve) with 1.5 billion points (2 million points/process)
- Why? CPU-only, high communication, even with spatial hashing

Challenges in Modeling

- Large voids
  - We don't currently support disabling cells – instead, use "sticky air": perfectly compressible, low stiffness material which does not retain state
- Heterogeneous materials on boundaries with fine details
  - Surface force decreases significantly under refinement
  - Only converges near grain-resolving scale*



What does “grain-resolving” mean?

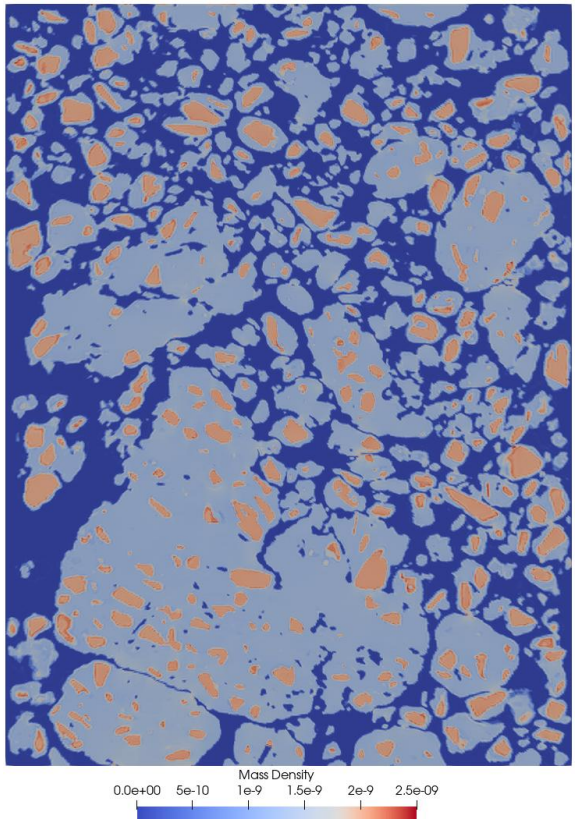
316 MPoints

- At least N points per grain
  - Relatively easy due to number/size of grains
  - Requires ~25 million points
- At least N points per voxel
  - Much harder, ~290 million voxels in domain
  - For approximately 1 cell/voxel, need ~8 billion material points
    - Checkpoint files at this resolution are about 1.5 TB each
    - Even projected mesh outputs are ~60GB
  - Even ~1 point per voxel is ~300 million points

- Unfortunately, it seems like we need the latter for surface forces to converge and to avoid numerical fracture

Challenges at Scale

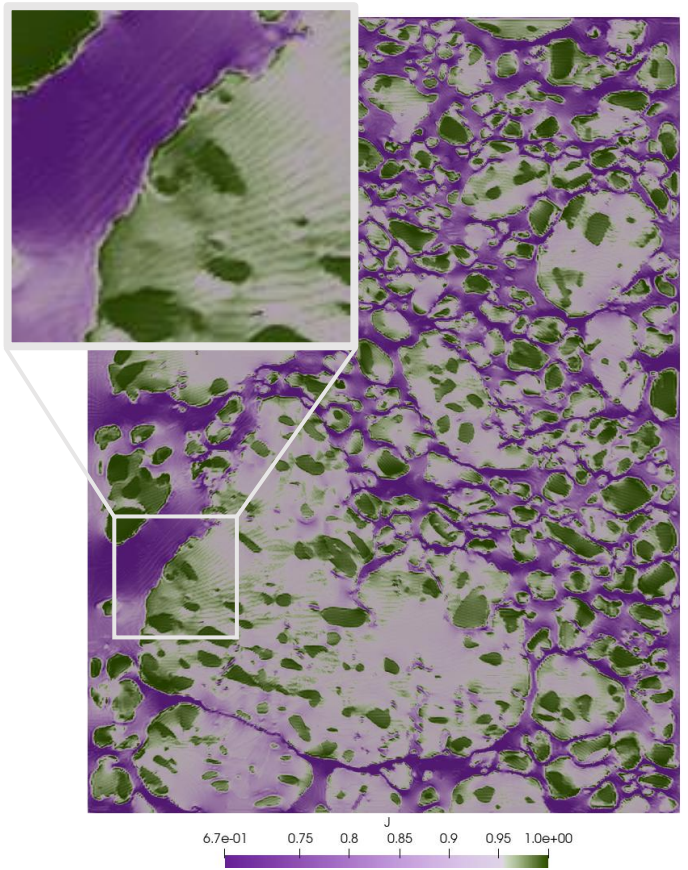
- Preconditioning/Linear Solver
  - More refined mesh ⇒ more linear iterations with Jacobi preconditioning
  - Solution: Use Algebraic Multigrid (AMG) preconditioner
- AMG Memory Requirements
  - Requires explicit assembly of libCEED matrix-free operators
  - Requires computing matrix triple products ($P^T AP$)
  - Just for memory, 1.5 billion points requires ~256 MI300A (128 GB)
- Point Migration/Location
  - Implemented in PETSc DMSwarm using spatial hashing
  - 1 spatial hash lookup + N cell/hash box pointwise nonlinear solves *per point* on process *and* for any points that left another process
  - Future work: Move this to GPU kernel and reduce communication



Mass Density – 1.5 billion points, lumped projection onto background mesh at ~20% deformation

Challenges due to MPM

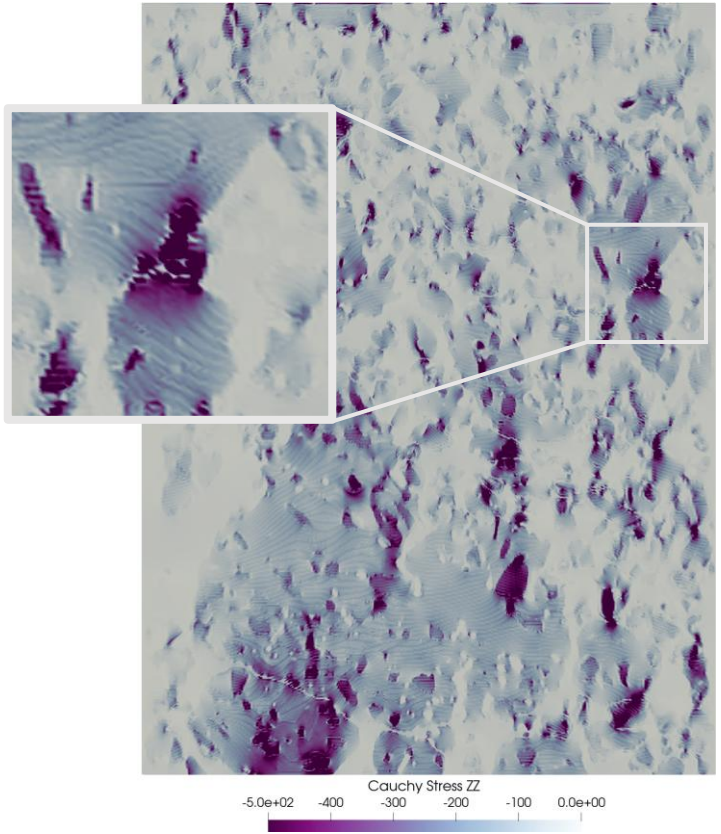
- Ringing/Cell-Crossing instability in projected output fields



$J = \det(\boldsymbol{F})$*: 1.5 billion points, lumped projection onto background mesh at ~20% deformation*

Challenges due to MPM

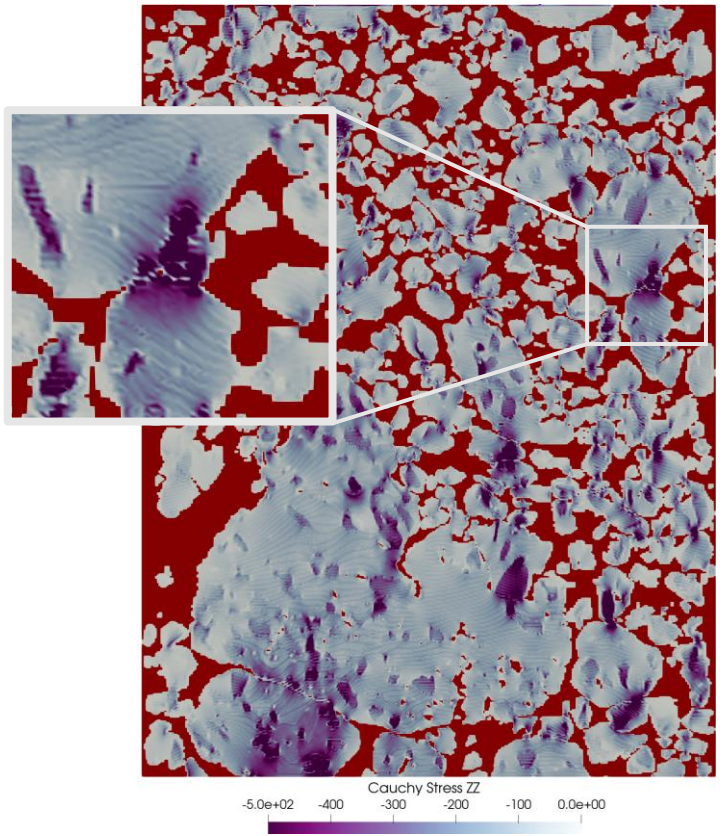
- Ringing/Cell-Crossing instability in projected output fields
- Stress singularities on prill boundaries
  - Prills effectively merge when they touch
  - Want prills to slide and rotate
  - Future work: inter-prill contact modeling



σ_{zz}: 1.5 billion points, lumped projection onto background mesh at ~20% deformation

Challenges due to MPM

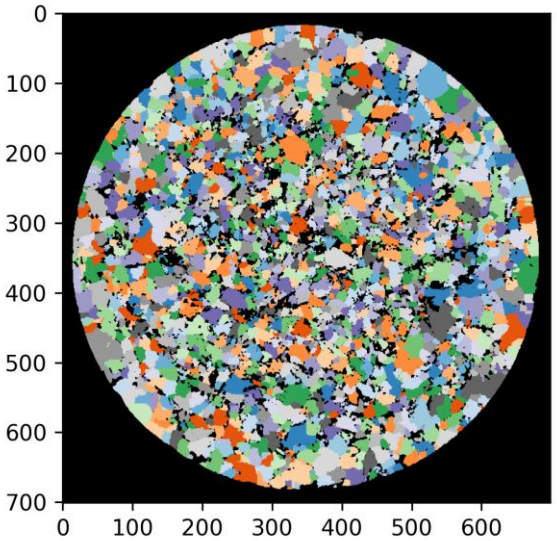
- Ringing/Cell-Crossing instability in projected output fields
- Stress singularities on prill boundaries
  - Prills effectively merge when they touch
  - Want prills to slide and rotate
  - Future work: inter-prill contact modeling



σ_{zz}: 1.5 billion points, lumped projection onto background mesh at ~20% deformation, air masked

Future Work

- Better input data
- Performance
  - GPU kernels for point location
- Material Modeling
  - Anisotropy
  - Viscoelasticity
  - Plasticity
- Simulation Capabilities
  - Traction boundaries
  - Inter-grain contact & friction



Hand-pressed sample (50N), credit: Trevor Lyons, Mines

Acknowledgements

The authors acknowledge support by the Department of Energy, National Nuclear Security Administration, Predictive Science Academic Alliance Program (PSAAP) under Award Number DE-NA0003962.

This research is supported by the Exascale Computing Project (17-SC-20-SC).

This research is supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under contract DE-AC02-06CH11357 and Award Number DE-SC0016140.

We thank Valeria Barra, Natalie Beams, Matthew Knepley, William Moses, and Junchao Zhang for their work on [3]. We thank PETSc, hypre, and libCEED developers, especially Mark Adams, Barry Smith, Stefano Zampini, Victor Paludetto Magri, Veselin Dobrev, Yohann Dudouit, and Tzanio Kolev, for collaboration on software development and algorithmic tuning. This research used Paraview, PyVista, Seaborn, Matplotlib, NumPy, and Pandas for data analysis and visualization.