Ratel - Solid Mechanics for the Exascale Era

Jeremy L Thompson

University of Colorado Boulder

jeremy@jeremylt.org

6 August 2025

Overview

1. libCEED
2. Ratel
3. GPU and AD
4. Contact and MPM
5. Questions

Top 500

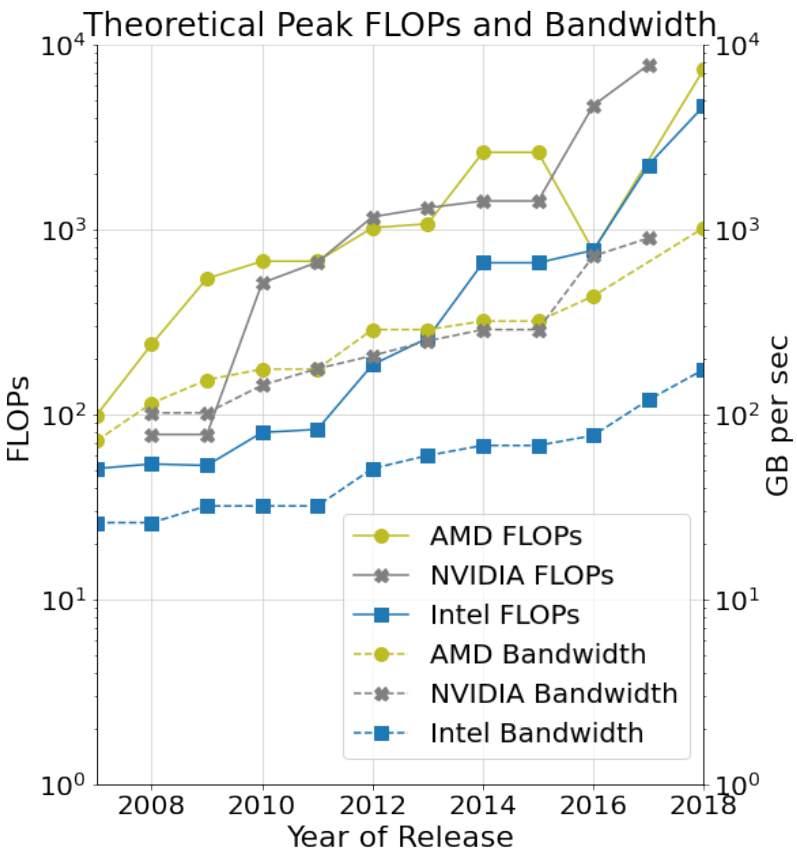
Machine	HPL	HPCG
El Capitan	1,742.00 PFLOPs	17.41 PFLOPS
Fugaku	442.01 PFLOPs	16.00 PFLOPs
Frontier	1,353.00 PFLOPs	14.05 PFLOPs
Aurora	1,012.00 PFLOPs	5.61 PFLOPs
LUMI	379.70 PFLOPs	4.59 PFLOPs

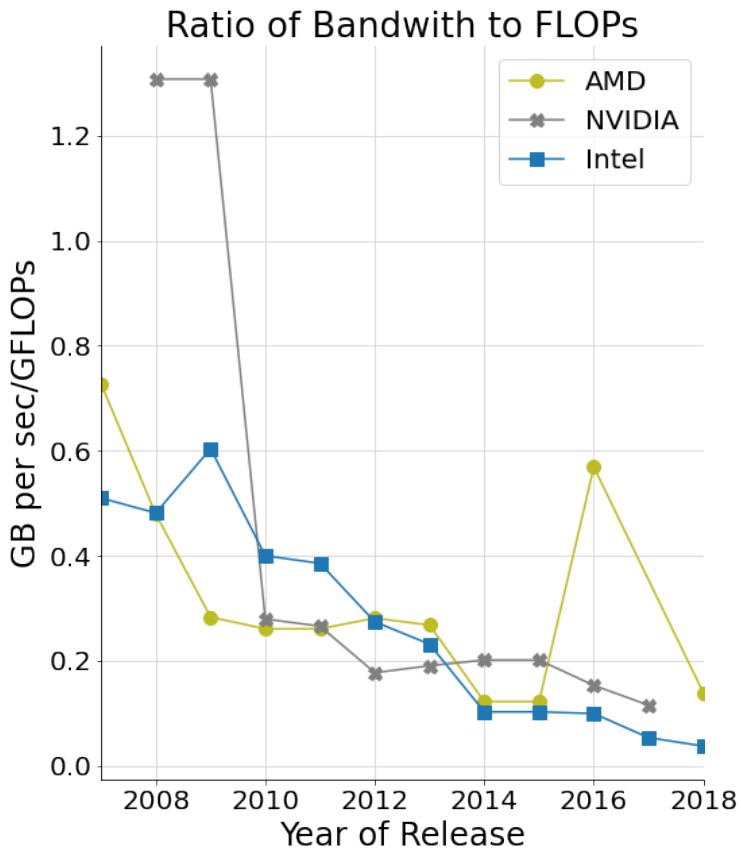
Top 500 Machines for HPCG with HPL peak FLOPs

HPCG closer to representative FLOPs for simulations

Difficult to realize peak FLOPs with CG on modern machines

Modern Hardware

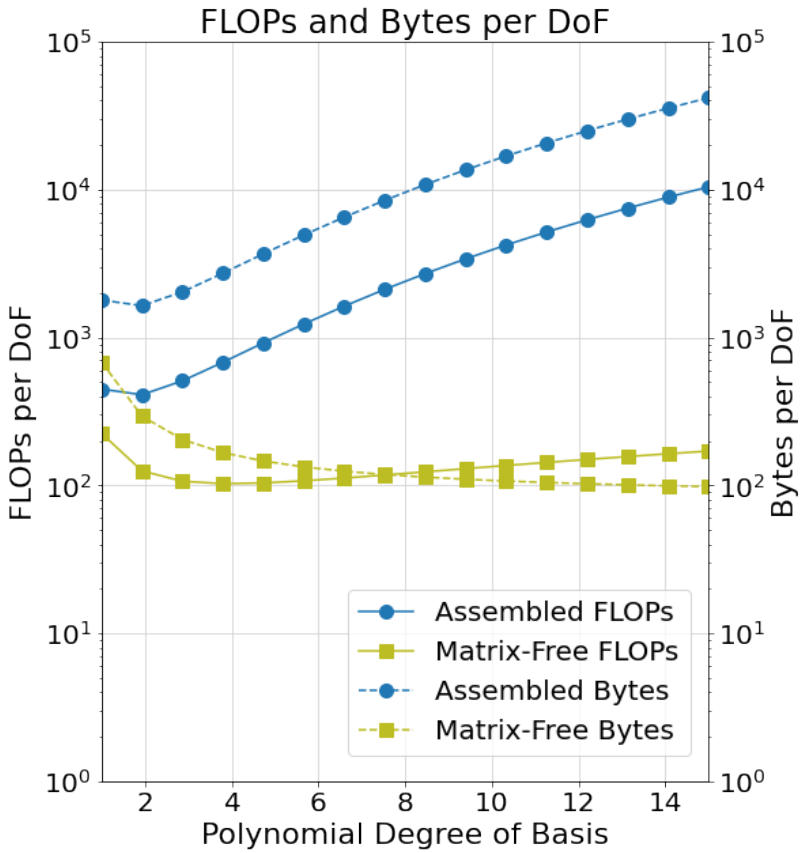


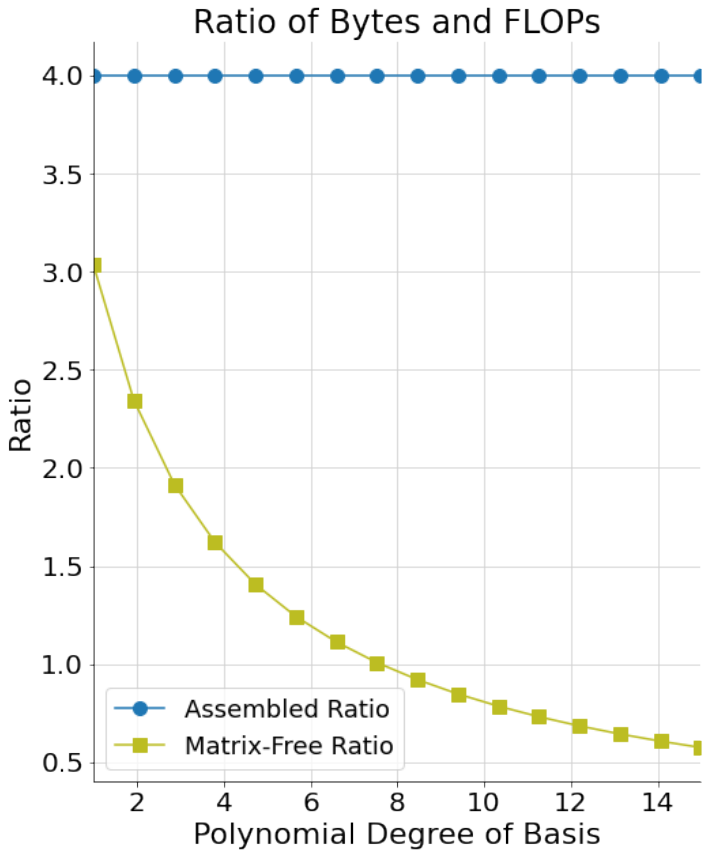


Memory bandwidth is improving slower than FLOPs

Mirrors difference between Top 500 HPL vs HPCG benchmarks

Benefits of Matrix-Free





Requirements for matrix-vector product with sparse matrix vs matrix-free for screened Poisson $\nabla^2 u - \alpha^2 u = f$ in 3D

Matrix-free representations using tensor product bases better match modern hardware

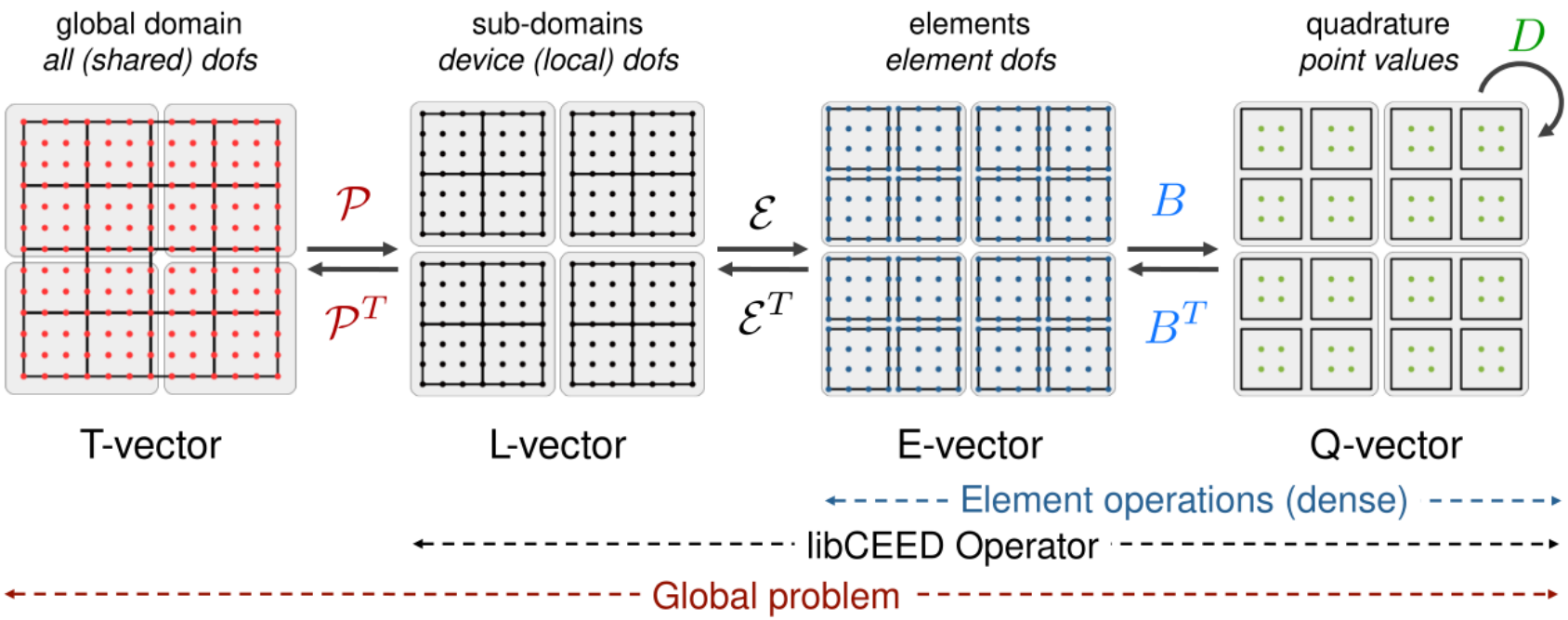
libCEED Team

libCEED Repo: https://github.com/CEED/libCEED

Developers: Ahmad Abdelfattah, Zach R. Atkins, Valeria Barra, Natalie Beams, Jed Brown, Jean-Sylvain Camier, Veselin Dobrev, Yohann Dudouit, Leila Ghaffari, Sebastian Grimberg, Tzanio Kolev, David Medina, Will Paznel, Thilina Ratnayaka, Rezgar Shakeri, Stan Tomov, James Wright III, Jeremy L Thompson

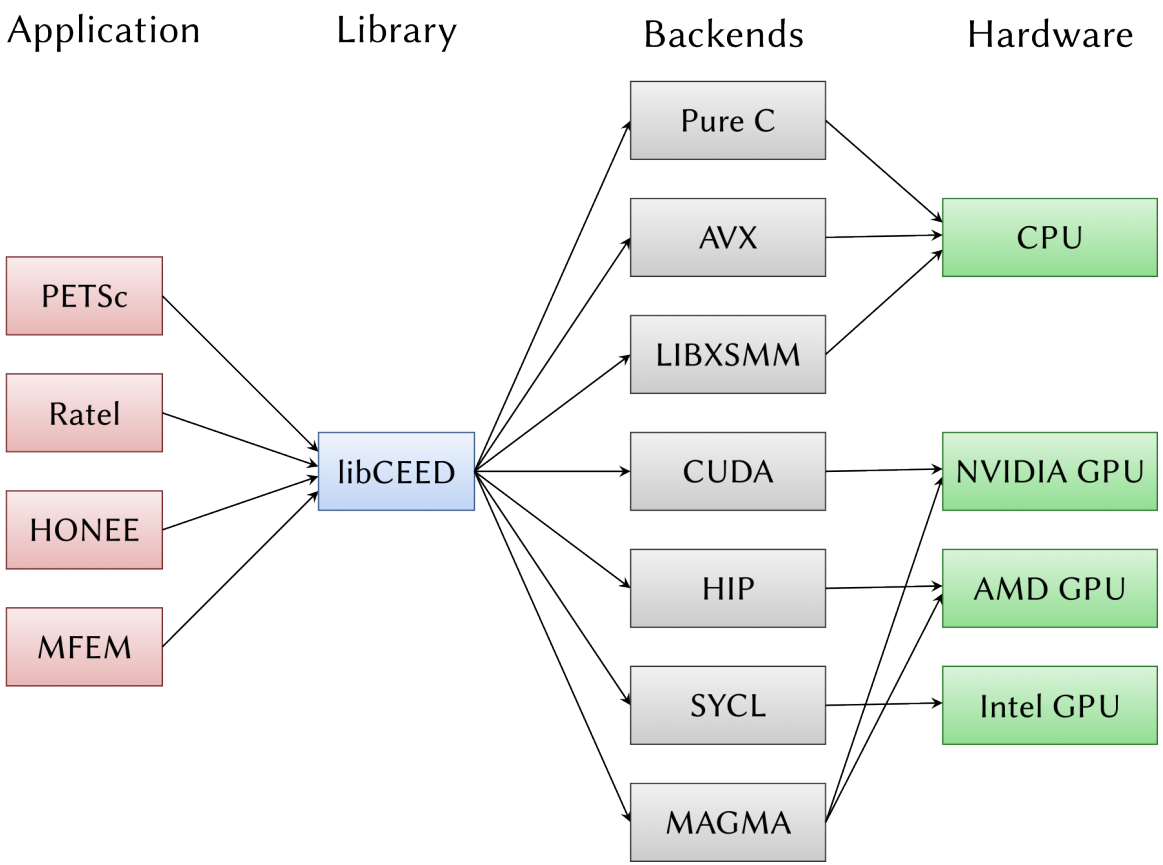
Matrix-Free Operators with libCEED

$$A = \mathcal{P}^T \mathcal{E}^T B^T D B \mathcal{E} \mathcal{P}$$



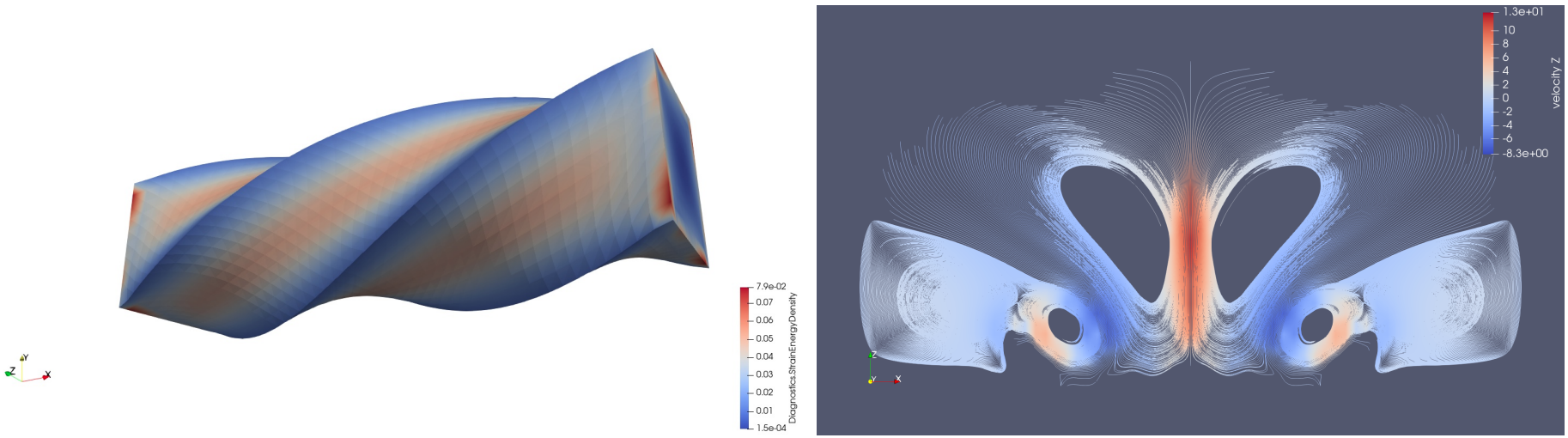
libCEED provides arbitrary order matrix-free operator evaluation

Performance Portability from libCEED



Performance portability with libCEED's matrix-free operators

Mini-Apps



libCEED solid mechanics (left) and fluid dynamics (right) mini-apps

- libCEED supports FEM-like simulations on modern hardware
- Mini-apps have been expanded into independent libraries

libCEED Projects

Several projects built using libCEED

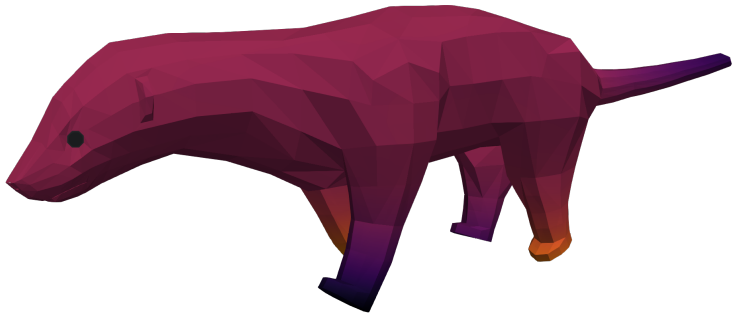
- Ratel - solid mechanics FEM and iMPM (PSAAP)
- HONEE - fluid dynamics FEM & differential filtering (PHASTA)
- MFEM - various applications, libCEED integrators (LLNL)
- Palace - quantum circuit design, MFEM + libCEED (Amazon)
- RDycore - FV river dynamical core, PETSc + libCEED (SciDAC)

Design Implications

Using matrix-free operators drives design decisions

- Direct solvers are out (assembled matrices, $\mathcal{O}\left(p^6\right)$)
- Iterative solvers are in (Krylov methods, etc)
- High(er) order = high accuracy & bad condition numbers
- Preconditioning is needed for fast convergence

Ratel Team

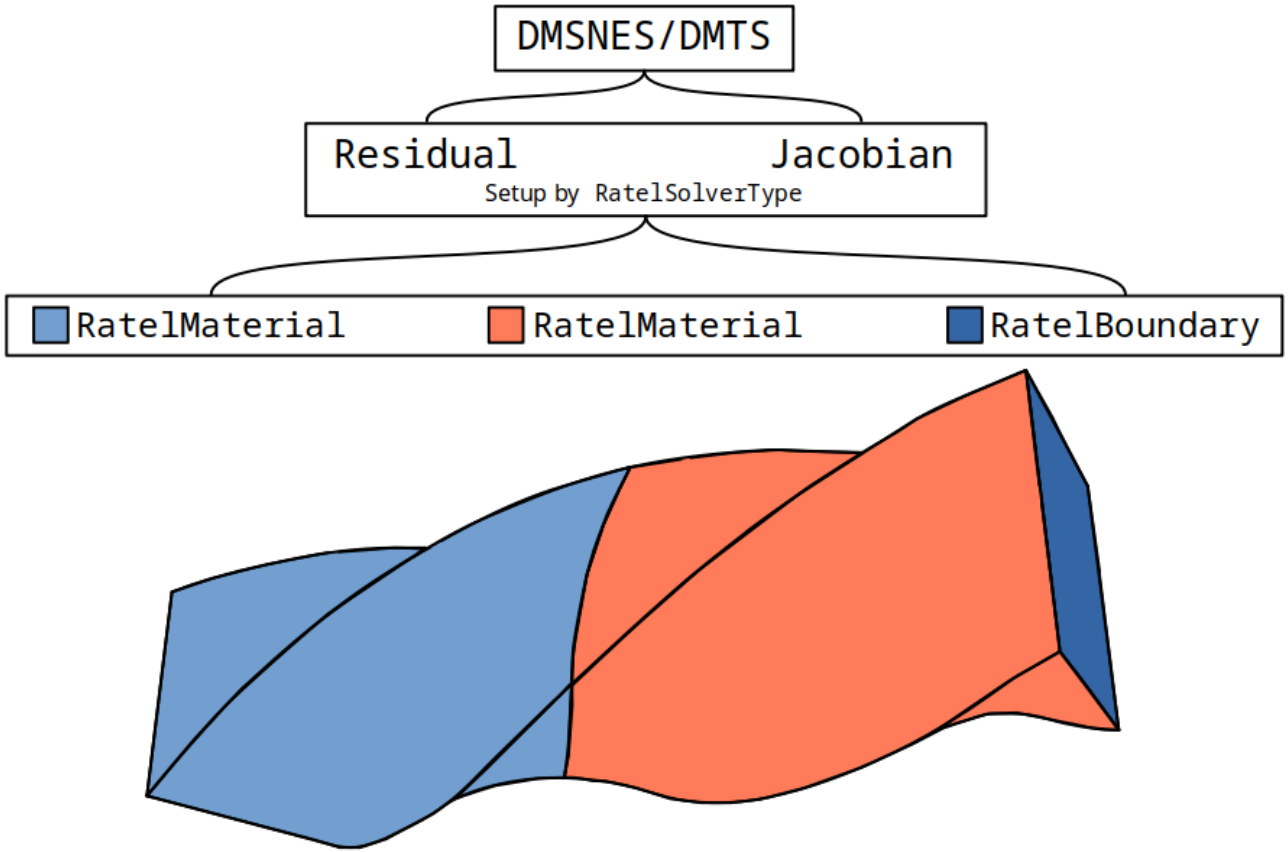


Ratel Repo: https://gitlab.com/micromorph/ratel

Developers: Zach R. Atkins, Jed Brown, Fabio Di Gioacchino, Leila Ghaffari, Zach Irwin, Rezgar Shakeri, Ren Stengel, Jeremy L Thompson

PSAAP Micromorph Center & iMPM

Basic Design



Each material region sets up part of the non-linear and linear equations

Material Models

Multiple models are supported in Ratel

- Elasticity - Linear, Neo-Hookean, Mooney Rivlin, Ogden, Hencky
- Mixed Elasticity - Linear, Neo-Hookean, Odgen
- Plasticity - Hencky, various types
- Anisostropy - Neo-Hookean
- Brittle Fracture - Linear damage, Neo Hookean damage
- Poromechanics - Linear, Neo-Hookean
- Viscoelasticity - Hencky
- iMPM - Neo-Hookean, Neo-Hookean damage

Current and initial configuration for many models
(Rezgar Shakeri, Fabio Di Gioacchino, Zach Irwin, PSAAP postdocs)

Boundary Conditions

Multiple boundary conditions are supported in Ratel

- Clamp (Dirichlet)
- Slip (partial Dirichlet)
- Traction (Neumann)
- Pressure
- Rigid contact - Nitsche, penalty (Coulomb or Threlfall friction)

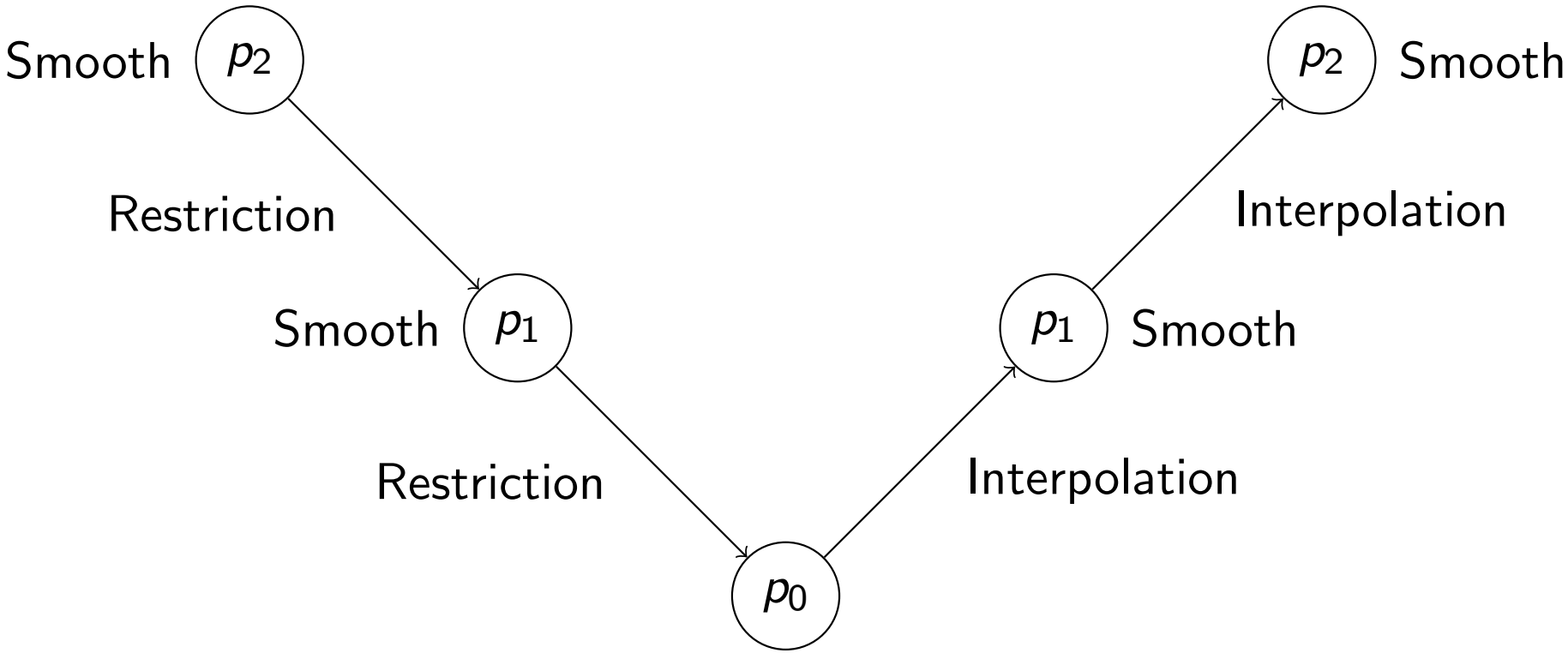
All BCs are time dependent
(Zach R. Atkins, PhD student and Rezgar Shakeri, PSAAP postdoc)

Preconditioning Support

Iterative solvers need preconditioning

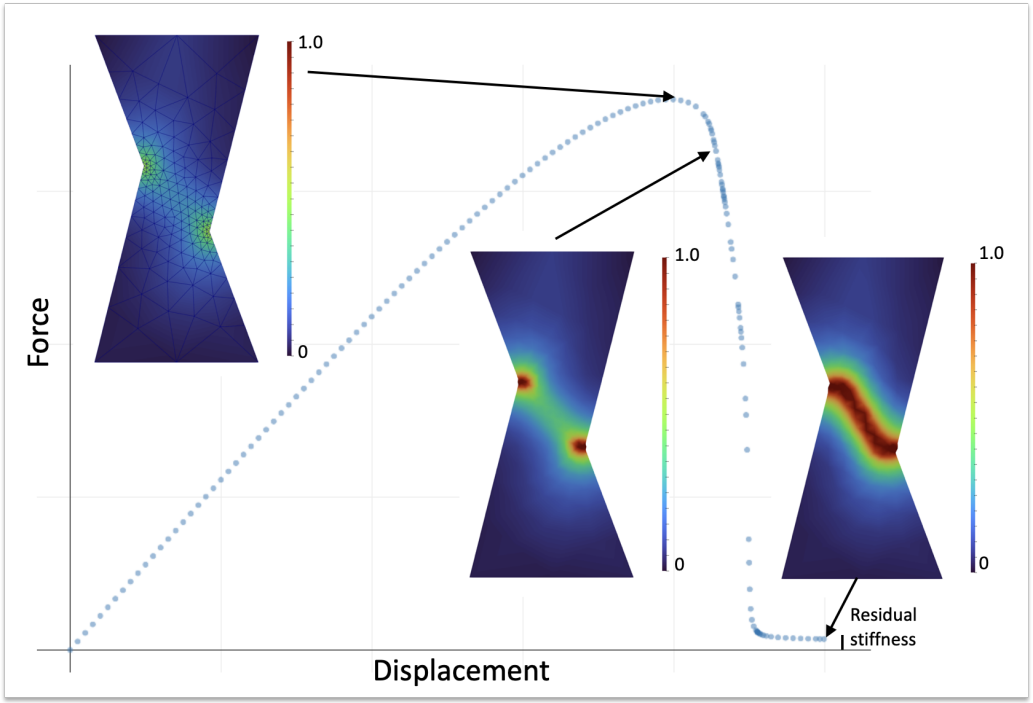
- $Ax = b$, slow for high-order (ill-conditioned)
- libCEED supports various preconditioner ingredients
- Most PETSc preconditioners fully supported
- Multigrid prolong/restrict also supported in libCEED

p-multigrid



Ratel uses matrix-free p-multigrid, AMG coarse solve

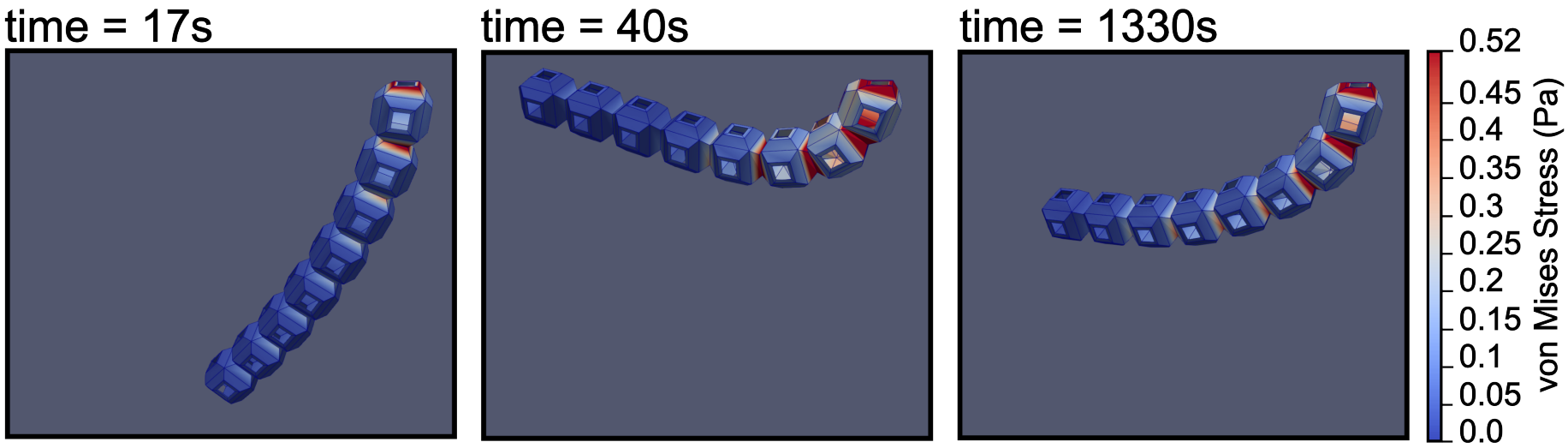
Example - Linear Damage




```
$ bin/ratel-quasistatic -options_file examples/ymls/ex02-
    quasistatic-elasticity-linear-damage-compressiveshear-
    AT2-face-forces.yml
```

Quasistatic simulation of compressive shear for generic brittle material
(Fabio Di Gioacchino, PSAAP postdoc)

Example - Dynamic Pendulum




```
$ bin/ratel-dynamic -options_file examples/ymls/ex03-dynamic
    -elasticity-schwarz-pendulum-enzyme.yml
```

Dynamic simulation of Neo-Hookean Schwarz-P 'pendulum' with Enzyme
(Layla Ghaffari, recent PhD graduate)

CEED Benchmark Problems

Performance on CEED BPs

- BP1 - Scalar projection problem
- BP2 - 3 component projection problem
- BP3 - Scalar Poisson problem
- BP4 - 3 component Poisson problem
- Ogden - representative production problem

Bulk of FLOPs are in basis evaluation for BPs

CEED Benchmark Problems

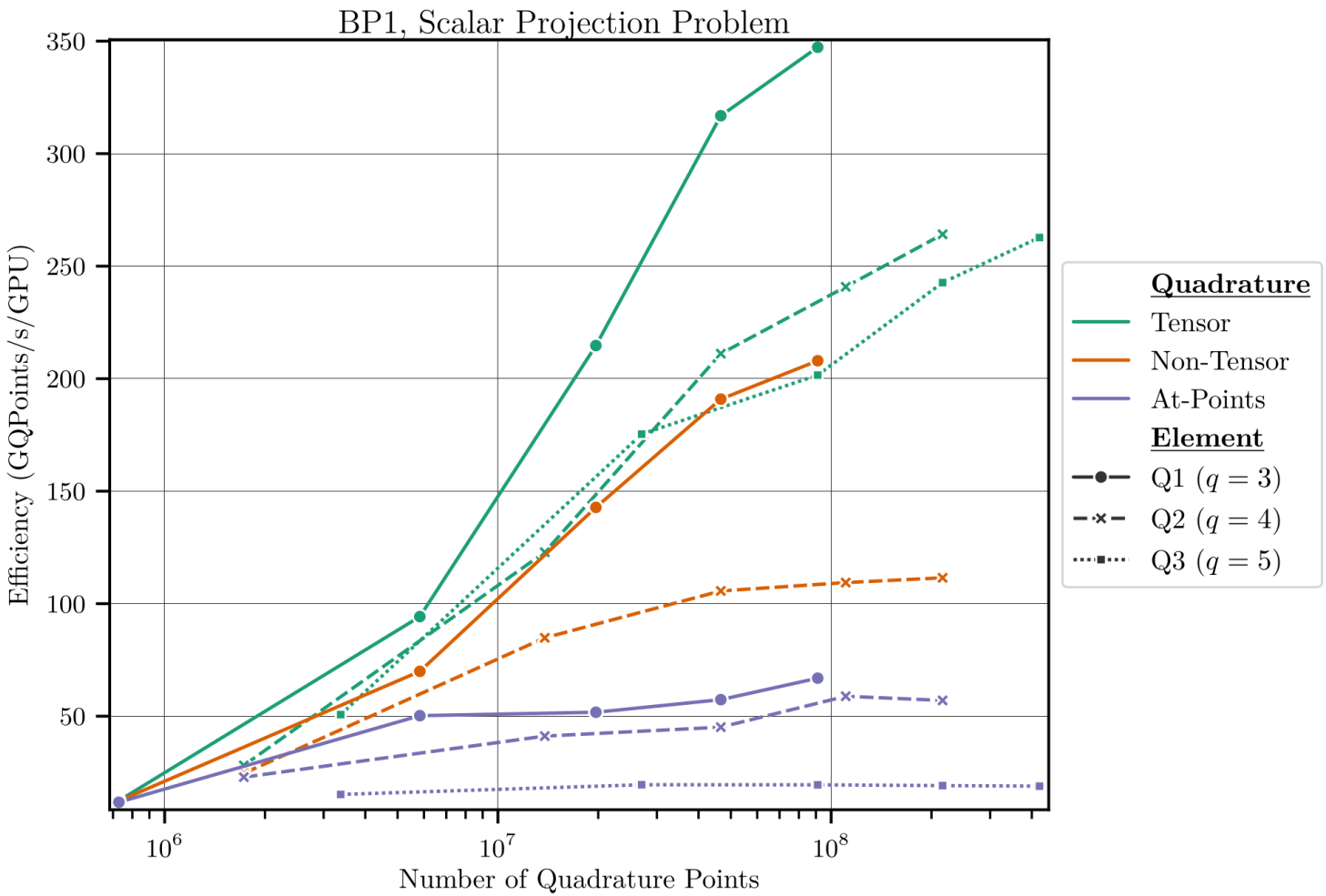
Performance on CEED BPs

- $p = 2, 3, 4$ and $q = p + 1$
- Units cube with 30^3, 60^3, 90^3, 120^3, and 150^3 elements
- Compare tensor, non-tensor, and at-points basis evaluation
- MMS w/ partial sum of Weierstrass function, $a = 0.5$, $b = 1.5$, $N = 2$

Using 4x AMD Instinct™MI300A Accelerated Processing Units (APUs)

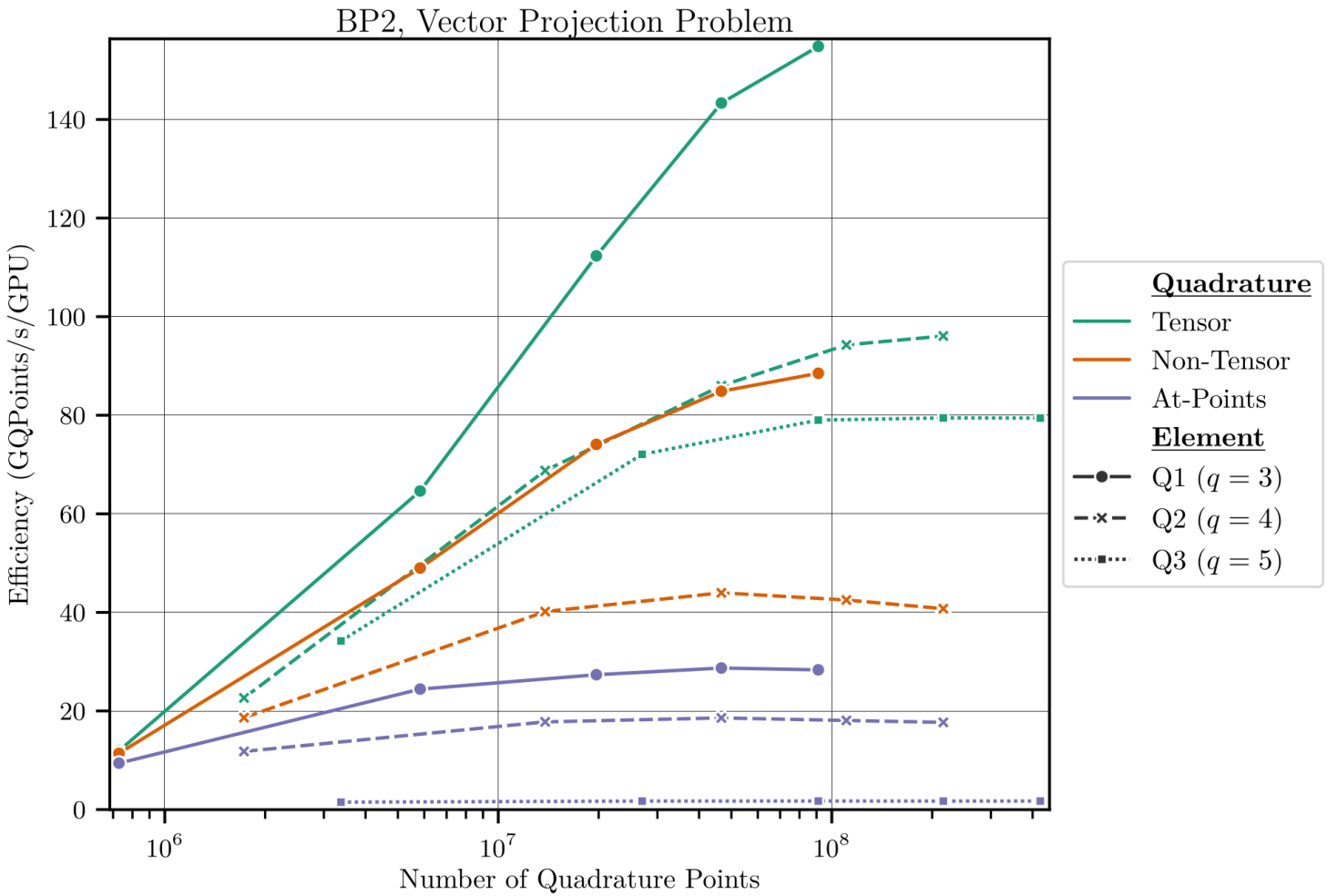
See also Performance Portable Solid mechanics via Matrix-Free p-Multigrid
(Also, dissertation of Ren Stengel, recent PhD graduate)

BP1



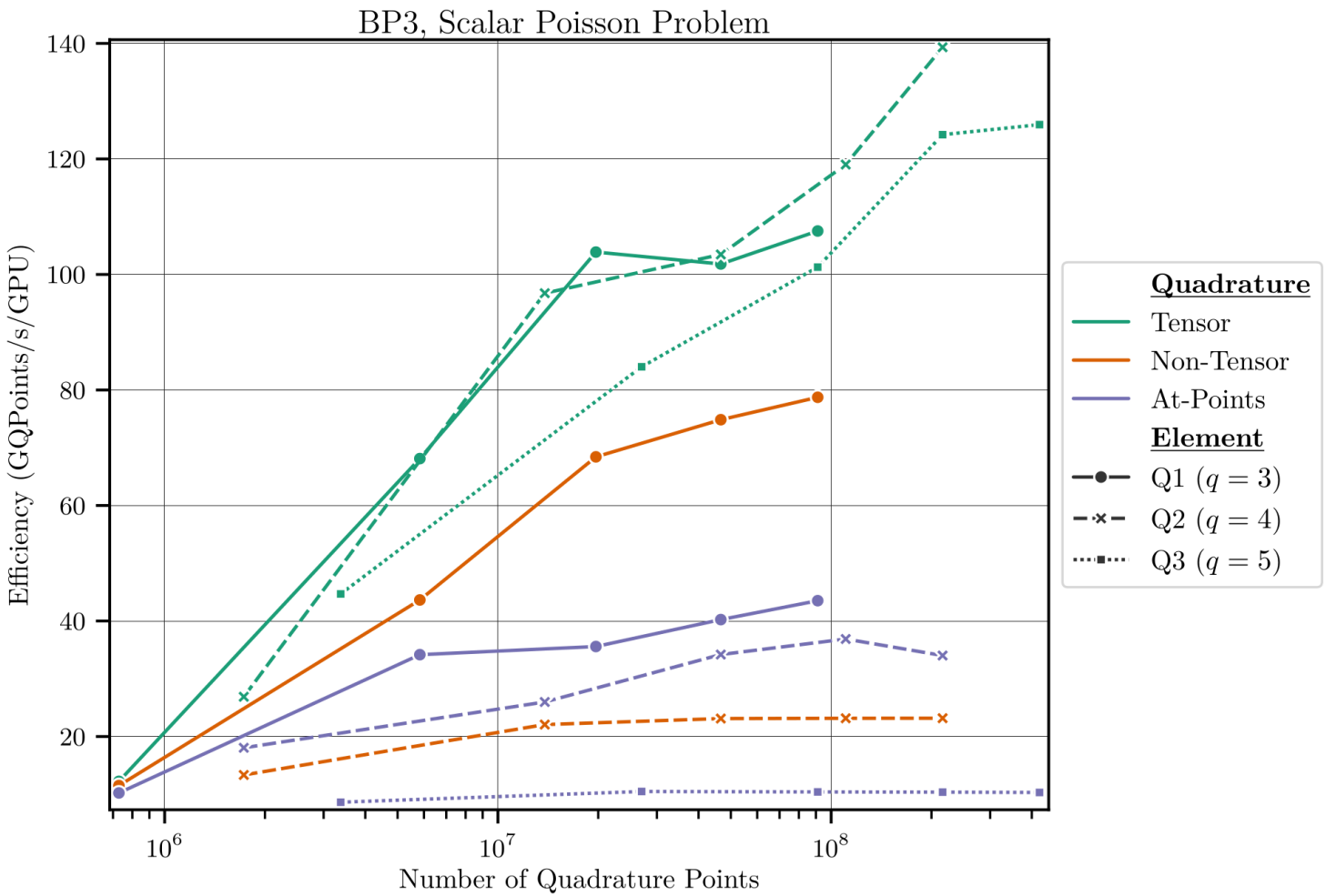
More FLOPs to do leads to lower efficiency

BP2



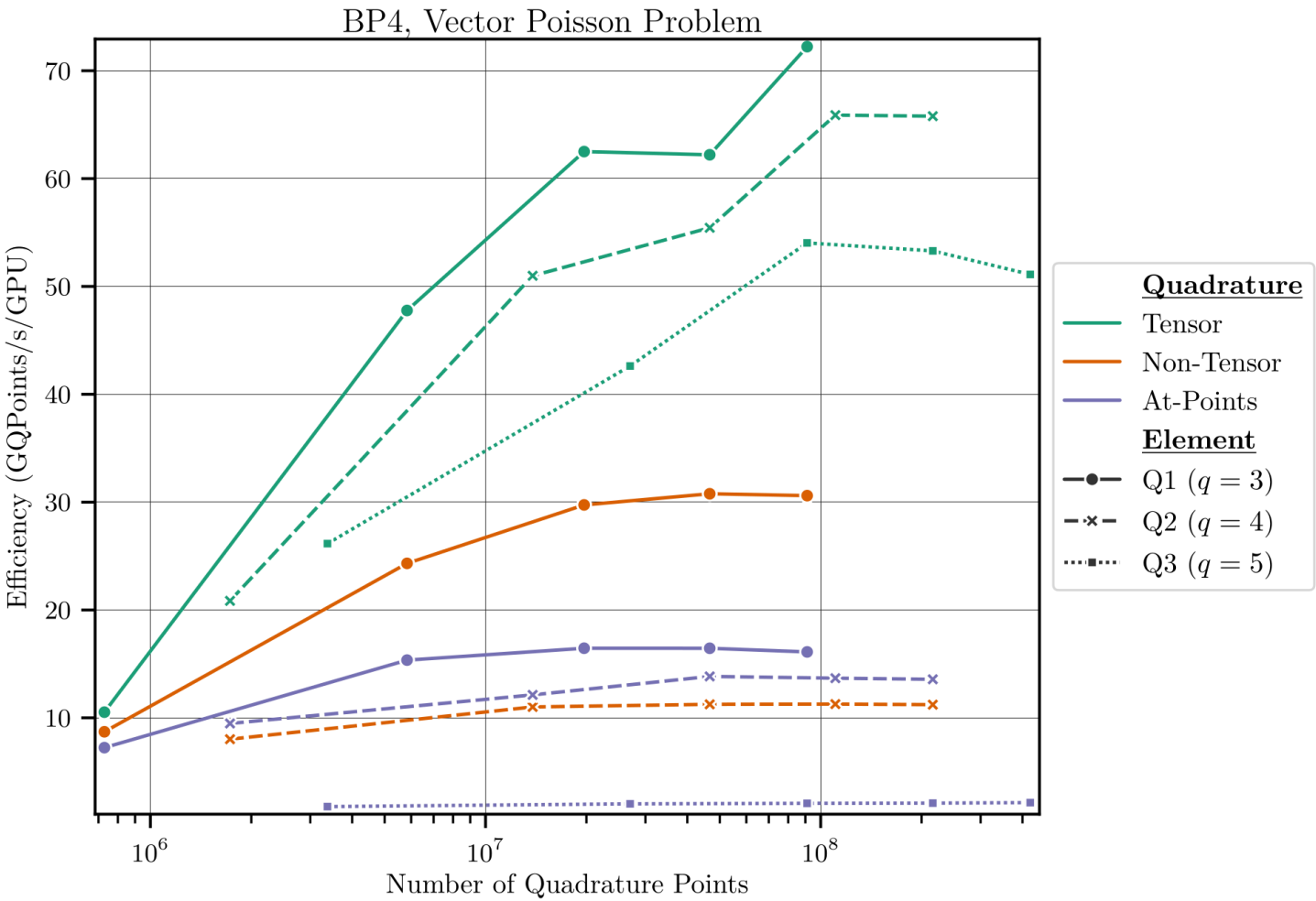
More components better represents production workloads

BP3



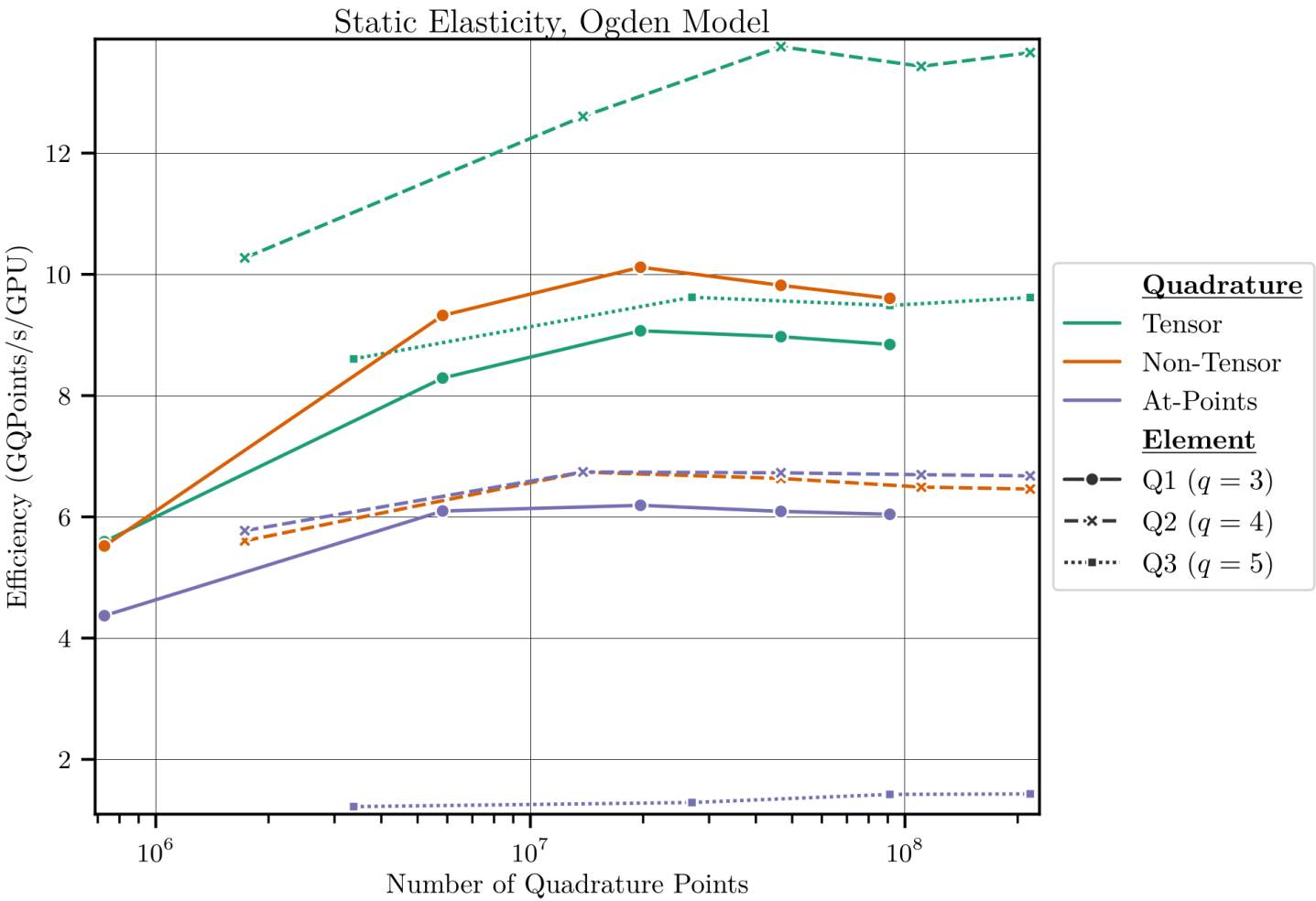
With derivatives, at-points closer to non-tensor

BP4



Closest CEED BP to production workload

Ogden



Basis cost less important with heavier QFunctions

Two Families of Approaches

Three libCEED backends with two approaches to operator application

- Separate kernels
  - `/gpu/*/ref` and `/gpu/*/shared`
  - $\mathcal{E}$, B, and D all separate kernels
  - Higher overall memory usage, multiple kernel launches
- Fused kernel
  - `/gpu/*/gen`
  - Single kernel JiTed with data from $\mathcal{E}$, B, and D
  - Lower overall memory usage, single kernel launch

Ref Operator Application

`/gpu/*/ref` and `/gpu/*/shared` use largely the same code

$A_L = \mathcal{E}^T B^T D B \mathcal{E}$ use separate kernels

$\mathcal{E}$ source comes from the `/gpu/*/ref`

`/gpu/*/ref` uses basic kernels for B

`/gpu/*/shared` uses shared memory for B

D source is given by the user

Gen Operator Application

`/gpu/*/gen` generates a single kernel for the operator

$A_L = \mathcal{E}^T B^T D B \mathcal{E}$ uses a single kernel

$\mathcal{E}$ source comes from the `/gpu/*/ref`

B source comes from `/gpu/*/shared`

D source is given by the user

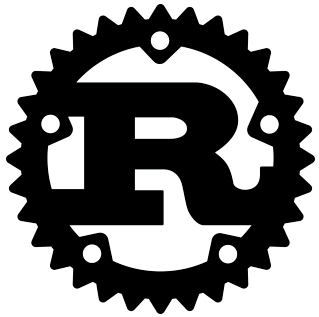
(Original gen backend by Yohann Dudouit)

Generated Operator Kernel

```
1  extern "C" __global__ void CeedKernelCudaGenOperator_mass(CeedInt num_elem,
2      void* ctx, FieldsInt_Cuda indices, Fields_Cuda fields, Fields_Cuda B,
3      Fields_Cuda G, CeedScalar *W, Points_Cuda points) {
4    // Setup kernel data
5
6    // Input and Output field constants and basis data
7
8    // Element loop
9    __syncthreads();
10   for (CeedInt elem = blockIdx.x*blockDim.z + threadIdx.z; elem < num_elem;
11        elem += gridDim.x*blockDim.z) {
12     // -- Input field restrictions (E) and basis actions (B)
13
14     // -- Output field setup
15     {
16       // -- Apply QFunction (D)
17       mass(ctx, 1, inputs, outputs);
18     }
19
20     // -- Output field basis actions (B^T) and restrictions (E^T)
21   }
22 }
23 // -----------------------------------------------------------------------------
```

$A_L = \mathcal{E}^T B^T D B \mathcal{E}$ in a single kernel

QFunctions in Rust



- libCEED can compile *D* in Rust for CUDA
- Lower Rust and `/gpu/cuda/gen` kernel to LLVM IR
- Link, optimize, and compile LLVM IR to PTX, similar final perf
- https://github.com/CEED/libCEED/pull/1881

(Summer undergrad Allen MacFarland)

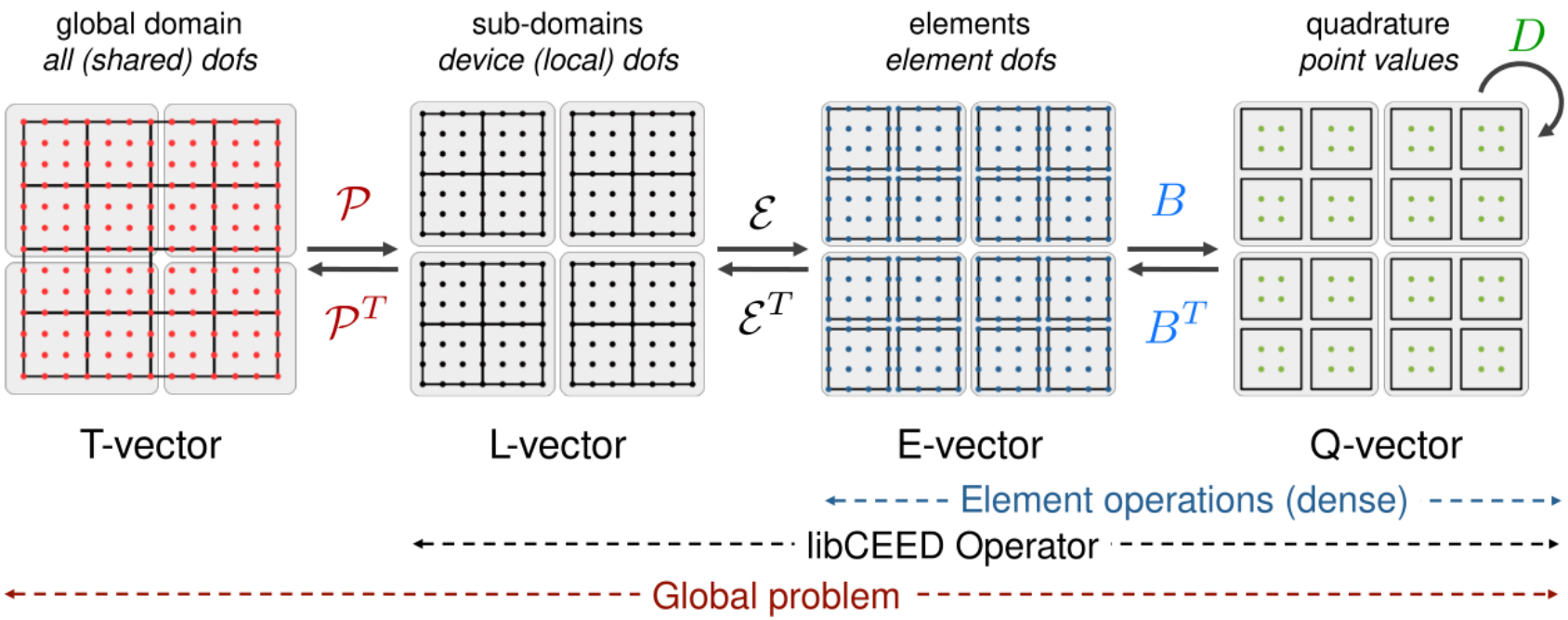
UHYPER Integration

- Ongoing work to wrap UHyper for Ratel
- CPU implementation working with Flang
- Hope to use LLVM IR for future CUDA support
- https://gitlab.com/micromorph/ratel/-/merge_requests/1136

(Summer undergrad Adonay Mezgebe)

AD Roadmap

$$A = \mathcal{P}^T \mathcal{E}^T B^T D B \mathcal{E} \mathcal{P}$$



AD can be implemented in D

Enzyme AD

- Computes gradients of source code
- Uses optimized LLVM IR
- Performance similar to hand derivatives w/o algebraic simplification*

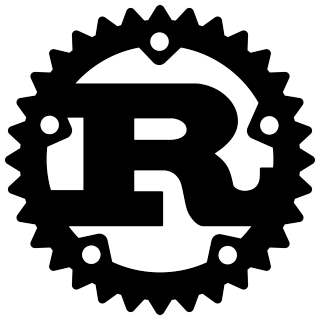
(See dissertation of Layla Ghaffari, recent PhD graduate)

Enzyme Neo-Hookean Elasticity

```
1  CEED_QFUNCTION_HELPER void RatelKirchhofftau_sym_NeoHookean_AD(const CeedScalar
       lambda, const CeedScalar mu, CeedScalar e_sym[6], CeedScalar tau_sym[6]) {
2    CeedScalar dPsi_sym[6] = {0.}, b_sym[6], dPsi[3][3], b[3][3], tau[3][3];
3
4    // dPsi / de
5    __enzyme_autodiff((void *)RatelStrainEnergy_NeoHookeanCurrentAD_Enzyme, e_sym,
         dPsi_sym, enzyme_const, lambda, enzyme_const, mu);
6    for (CeedInt i = 3; i < 6; i++) dPsi_sym[i] /= 2.;
7
8    // b = 2 e + I
9    for (CeedInt j = 0; j < 6; j++) b_sym[j] = 2 * e_sym[j] + (j < 3);
10
11   // tau = (dPsi / de) b
12   RatelSymmetricMatUnpack(dPsi_sym, dPsi);
13   RatelSymmetricMatUnpack(b_sym, b);
14   RatelMatMatMult(1., dPsi, b, tau);
15   RatelSymmetricMatPack(tau, tau_sym);
16 }
17
18 CEED_QFUNCTION_HELPER void Rateldtau_fwd(const CeedScalar lambda, const CeedScalar mu
       , CeedScalar e_sym[6], CeedScalar de_sym[6], CeedScalar tau_sym[6], CeedScalar
       dtau_sym[6]) {
19   __enzyme_fwddiff((void *)RatelKirchhofftau_sym_NeoHookean_AD, enzyme_const, lambda,
         enzyme_const, mu, e_sym, de_sym, tau_sym, dtau_sym);
20 }
```

Enzyme computing Jacobian from Residual, via energy

Better Enzyme AD



- Enzyme AD available in Rust
- Simplifies user experience, route to easier GPU impl
- Goal of similar perf to hand derivatives of D in C
- https://github.com/EnzymeAD/rust

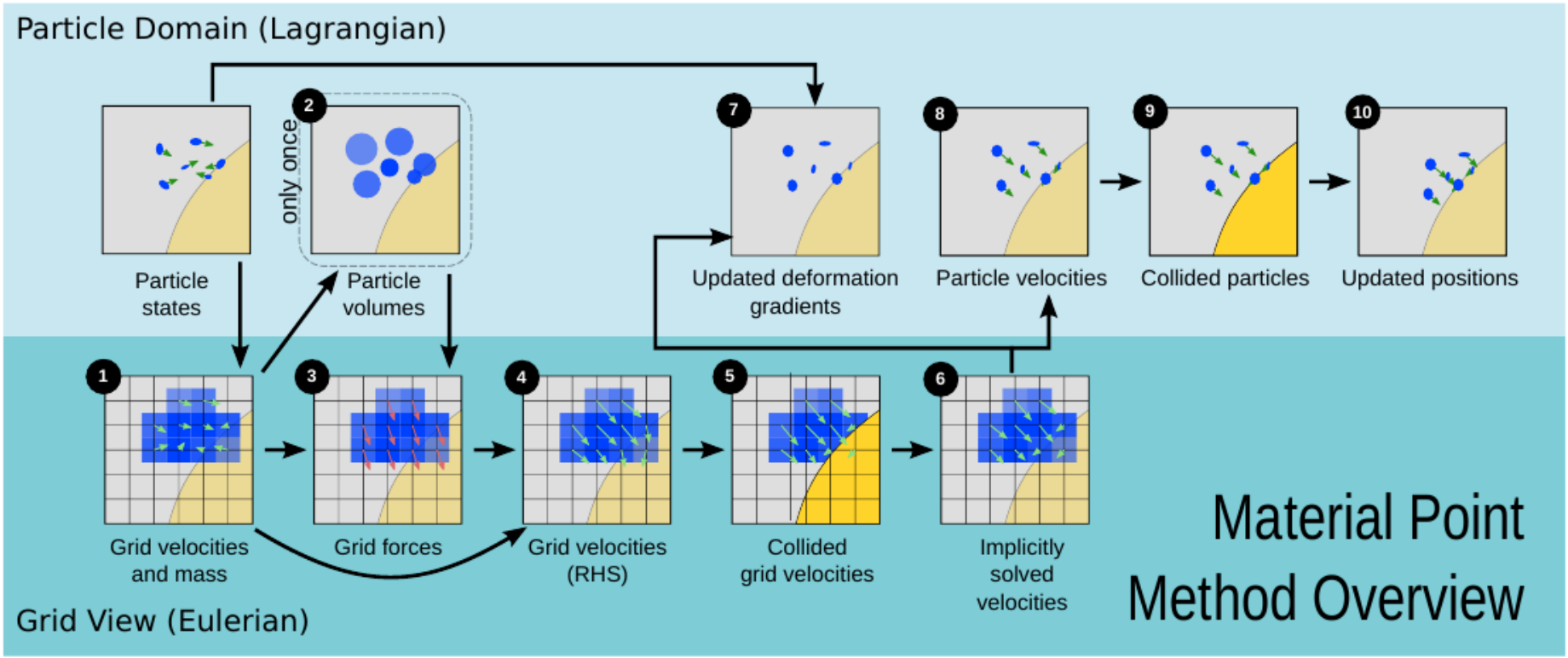
Contact Models

Nitsche method with rigid platens

- Support both Nitsche and penalty method
- Contact shape either platen or cylinder
- Support time dependent translation of surface
- Coulomb or Threlfall friction
- Deformable mesh-to-mesh contact in progress

(Zach R. Atkins, PhD student)

What is MPM?



- Continuum based particle method with background mesh for gradients
- Extension of FLIP (which is an extension of PIC)
- Used in rendering for the movie *Frozen*

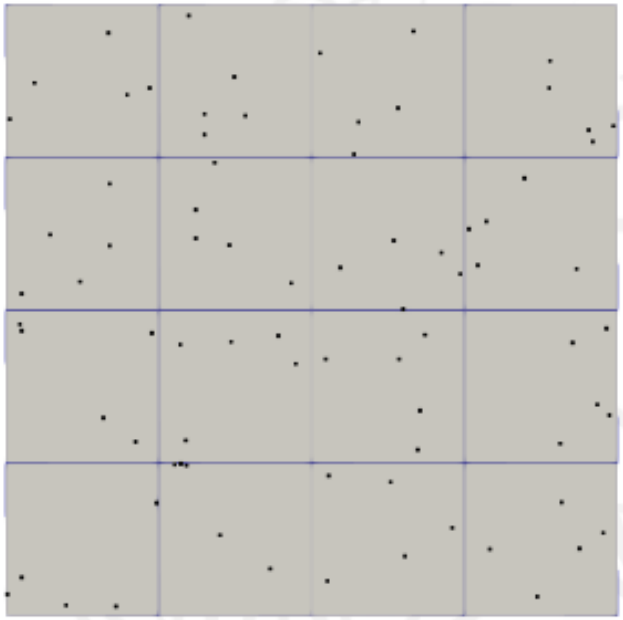
MPM vs FEM

MPM can be formulated as very similar to FEM

- Problem on background mesh changes when material points move
- Can be viewed as FEM with arbitrary quadrature point locations
- Natural fit for libCEED matrix-free representation
- Ratel FEM infrastructure provides fast background mesh solves

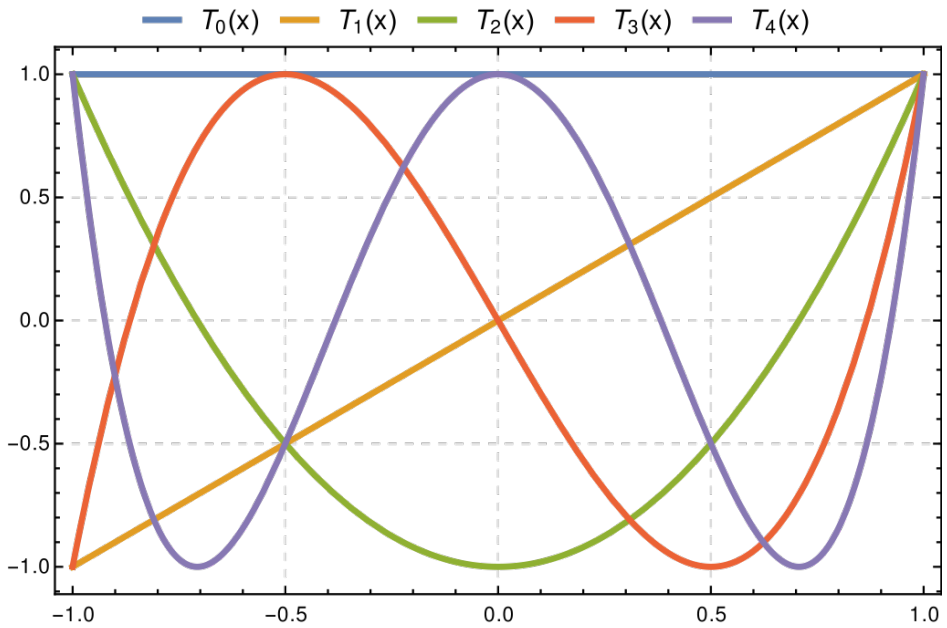
(Zach R. Atkins, PhD student)

DMSwarm for Material Points



- PETSc DMSwarm manages material points
- Point migration on CPU only (Toulumne/El Capitan helps here!)
- PETSc DMPlex manages background mesh

libCEED Basis Evaluation to Points



- Interpolate from primal to dual (quadrature) space
- Fit Chebyshev polynomials to values at quadrature points
- Evaluate Chebyshev polynomials at arbitrary points

libCEED Basis Evaluation to Points

Interpolation to Chebyshev has same FLOPs as FEM $\mathcal{O}\left(q^4\right)$

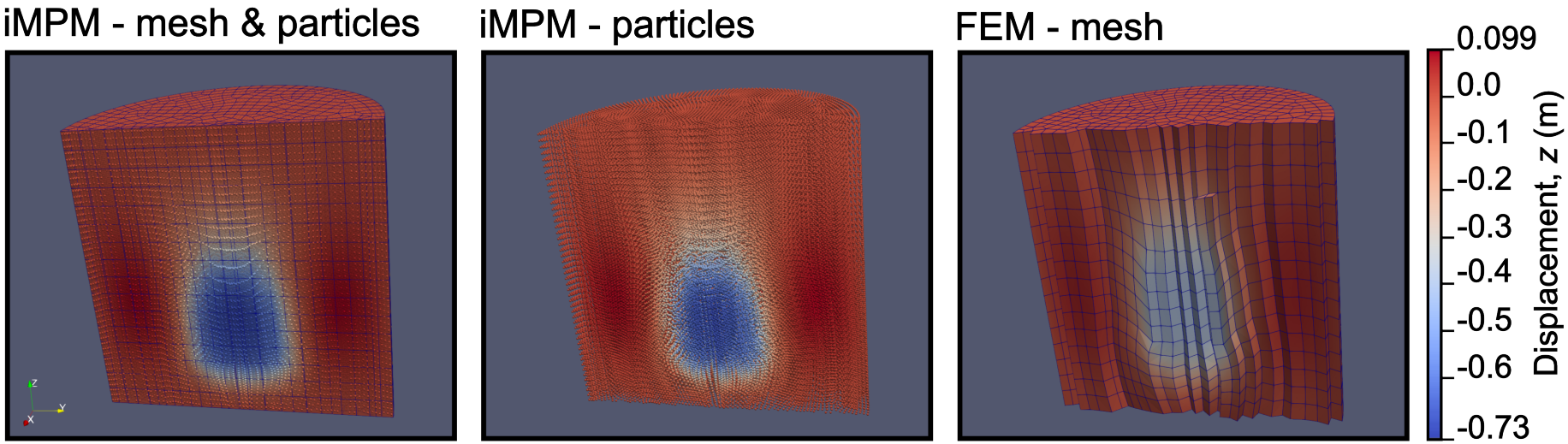
- Invert map C^{-1} from quadrature points to Chebyshev coeffs
- Create 1D interpolation matrix $B = CN$
- Tensor product:
  $B = (C \otimes C \otimes C)(N \otimes N \otimes N) = (CN) \otimes (CN) \otimes (CN)$
- Additional cost from evaluation to arbitrary points

libCEED Basis Evaluation to Points

Per point evaluation is expensive $\mathcal{O}\left(q^3\right)$

- Recurrence for Chebyshev values at point
  $f_0 = 1,\ f_1 = 2x,\ f_n = 2xf_{n-1} - f_{n-2}$
  $f_0' = 0,\ f_1' = 2,\quad f_n' = 2xf_{n-1}' + 2f_{n-1} - f_{n-2}'$
- Contract pencil of values with element coefficients
- Evaluation is independent per material point
- $\mathcal{O}\left(q^3\right)$ FLOPs at $\mathcal{O}\left(\hat{q}^3\right)$ points
- Using $p = q,\ \hat{q} = q + 1$ in current work

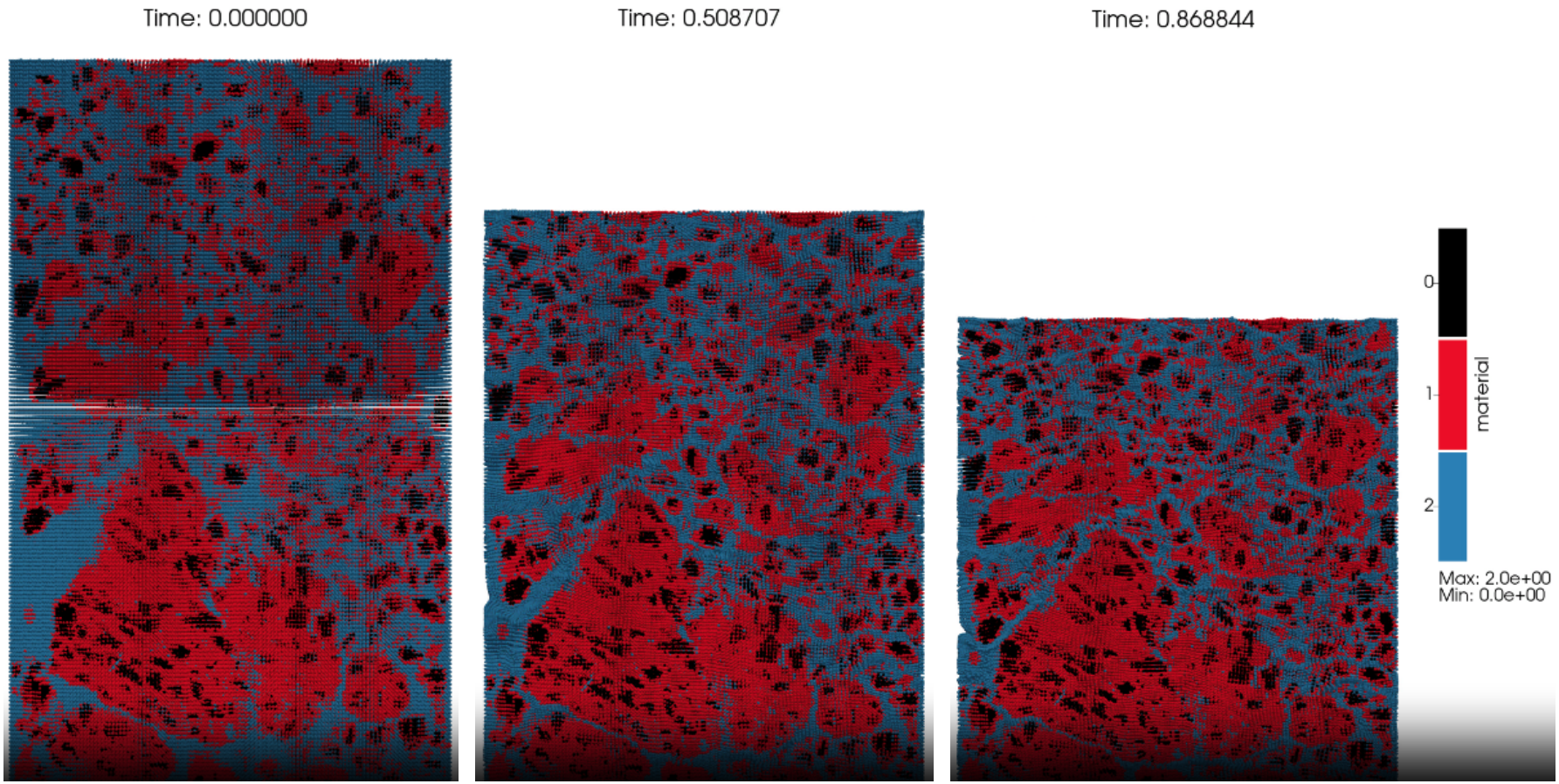
Example - MPM Sinker




```
$ bin/ratel-quasistatic -options_file examples/ymls/ex02-
    quasistatic-elasticity-mpm-neo-hookean-damage-current-
    sinker-cylinder.yml
```

FEM, iMPM simulations of dense sinker in "foam" validation problem
(Mesh distortion limits FEM simulation)

Example - Press Simulation



Compression of mock HE grains (black) and binder (red) mixture
(Reset background mesh to computational region on each timestep)

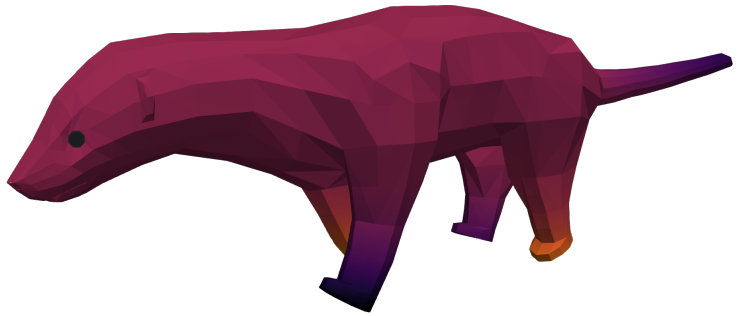
Ratel PCpMG

pMG giving promising initial results with GPU impl

- Neo-Hookean finite strain elasticity with damage
- Confined press of grain/binder with "sticky air" voids
- Jacobi iterations tend to double with 2x refinement
- pMG iteration counts robust with refinement

	Material Points	Jacobi its	pMG its
Coarse	388,800	900-1000	35-45
Fine	7,372,800	-	25-40

Questions?



libCEED Repo: https://github.com/CEED/libCEED
Ratel Repo: https://gitlab.com/micromorph/ratel

Grant: Predictive Science Academic Alliance Program (DE-NA0003962)

Ratel - Solid Mechanics for the Exascale Era

Jeremy L Thompson

University of Colorado Boulder

jeremy@jeremylt.org

6 August 2025