Motivating Matrix-Free Finite Elements

Jeremy L Thompson

University of Colorado Boulder

jeremy@jeremylt.org

19 Mar 2026

Overview

Discretization of PDE for computation presents several choices

Matrix-free finite elements present benefits and challenges

We'll see how matrix-free finite elements works and the downstream choices and challenges that this discretization provides

Overview

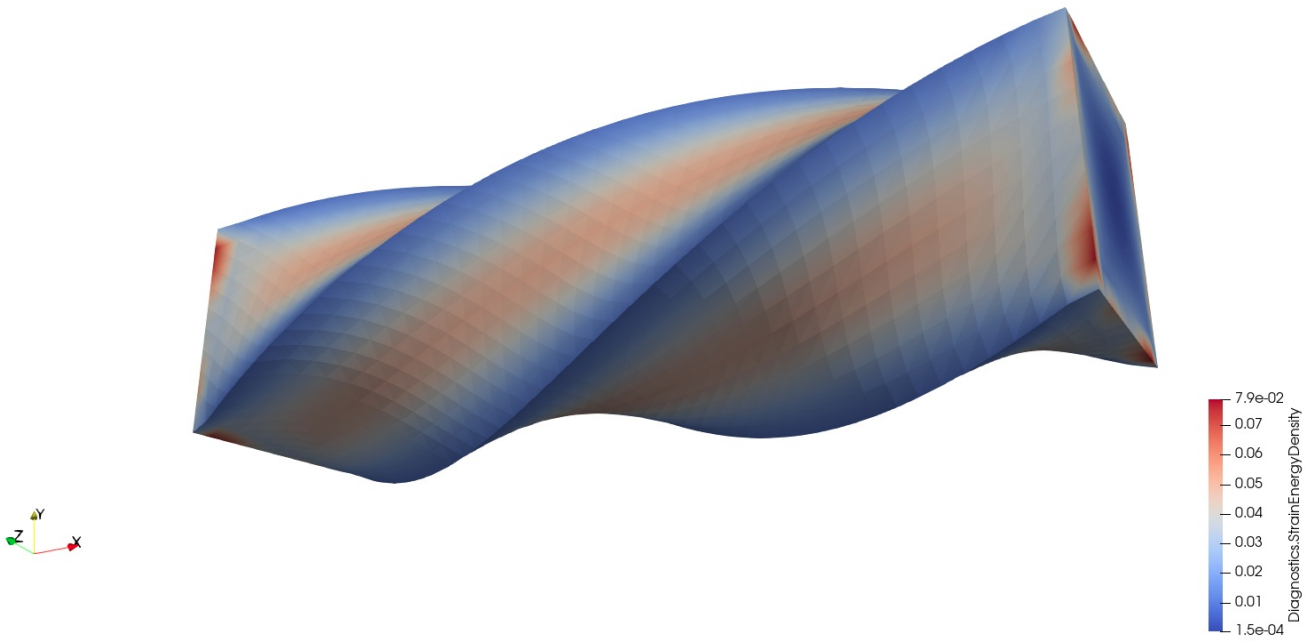
1. Finite Elements
2. Matrix-Free
3. Non-Linear Problem
4. Preconditioning
5. Cool Results
6. Questions

End Goal

We need to transform equations:

$$
\begin{aligned}
-\nabla \cdot \sigma - \rho g &= 0, \text{ in } \Omega \\
u &= \bar{u}, \text{ on } \partial\Omega^D \\
\sigma \cdot n &= \bar{t}, \text{ on } \partial\Omega^N
\end{aligned}
$$

into simulations:



Equation to Solve

Consider the general non-linear residual

$$F(u) - R = 0$$

where

- $F(u)$ represents our 'physics'
- R is some 'right hand side' or forcing term
- u is the unknown solution

Problem

Computers like discrete equations/operations

Our equation is continuous, and computers *hate* that

Another Problem

Solving these non-linear PDEs can be hard

Really, really hard

First Decision

With the L^2 inner product

$$\langle v, u \rangle = \int_\Omega vu \, dV$$

We can reformulate to the 'weak form':

$$\langle v, f(u) \rangle = \int_\Omega v \cdot f_0(u, \nabla u) + \nabla v : f_1(u, \nabla u) \; dV = 0$$

Note:

- boundary terms are omitted here (integration by parts)
- v is in some homogeneous space V_0

f_0 and f_1 Examples

Projection problem:

$$u = g$$

$$\int_\Omega v \cdot u \, dV = \int_\Omega vg \, dV$$

$$f_0\left(u, \nabla u\right) = u, \; f_1\left(u, \nabla u\right) = 0$$

Poisson problem:

$$-\nabla^2 u = g$$

$$\int_\Omega \nabla v : \nabla u \, dV - \int_{\partial\Omega} v \nabla u \cdot n \, dS = \int_\Omega vg \, dV$$

$$f_0\left(u, \nabla u\right) = 0, \; f_1\left(u, \nabla u\right) = \nabla u$$

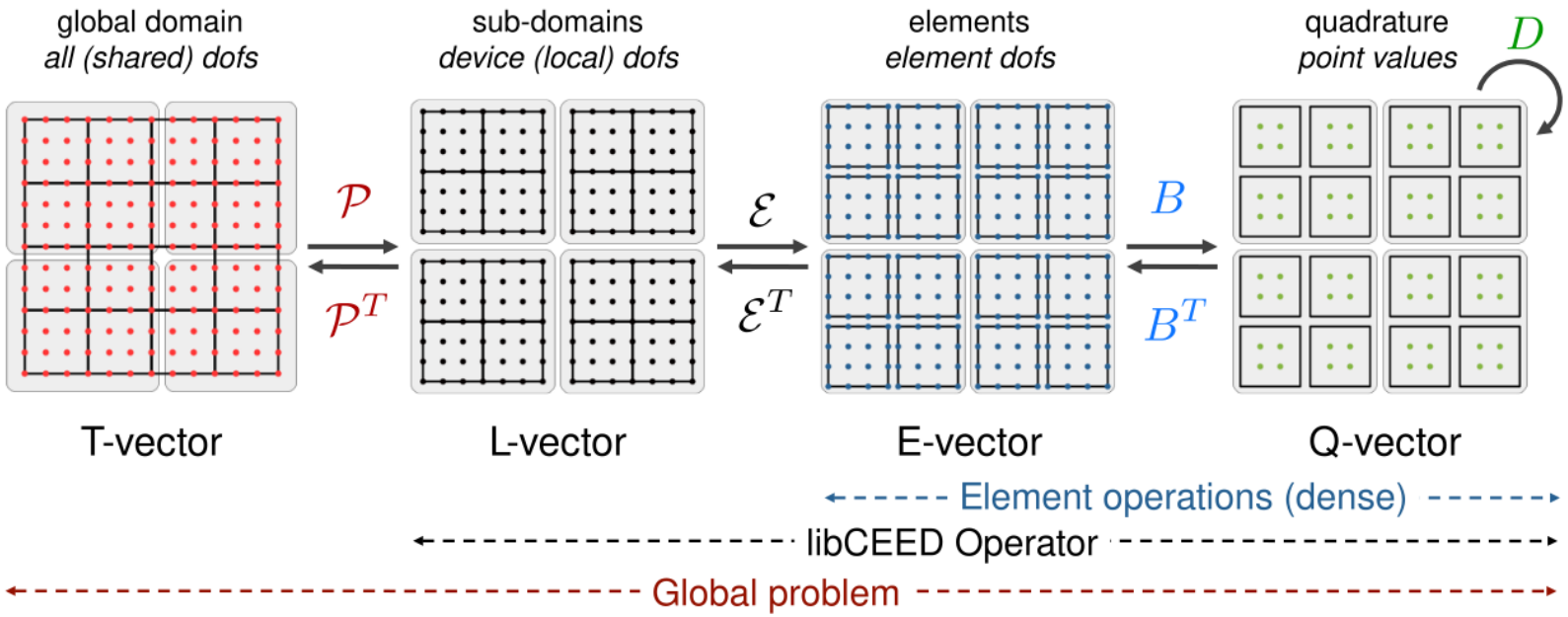
Second Decision

How to discretize?

- $u, v \in V$
- Different V give different properties
- We select V to be the space of piecewise continuous polynomials
- No continuity in derivatives (not splines)

Second Decision

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



T(otal)-vector represents discretization

Want solution values on red dots (Degrees of Freedom [DoF])

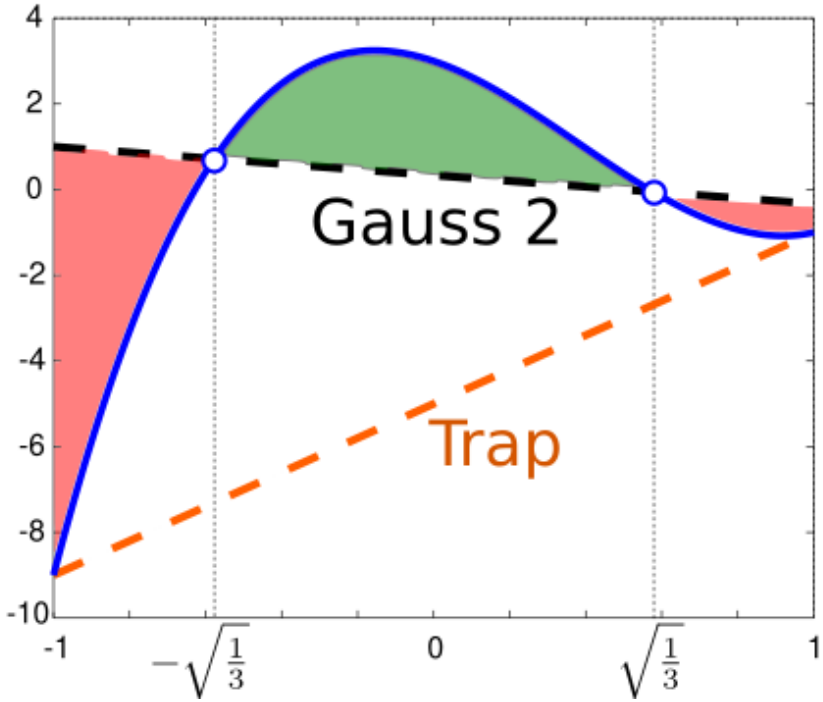
Fitting piecewise polynomial to red dots gives continuous solution

Third Decision

Need to convert integration to a discrete problem

- Quadrature rules take values at points to compute integral
- Piecewise nature means each element can be integrated separately
- Workflow:
  1. Map to interpolating polynomial
  2. Evaluate values/derivatives at quadrature points
  3. Evaluate f_0, f_1
  4. Multiply by quadrature weights
  5. Transpose process to return to DoFs

Third Decision



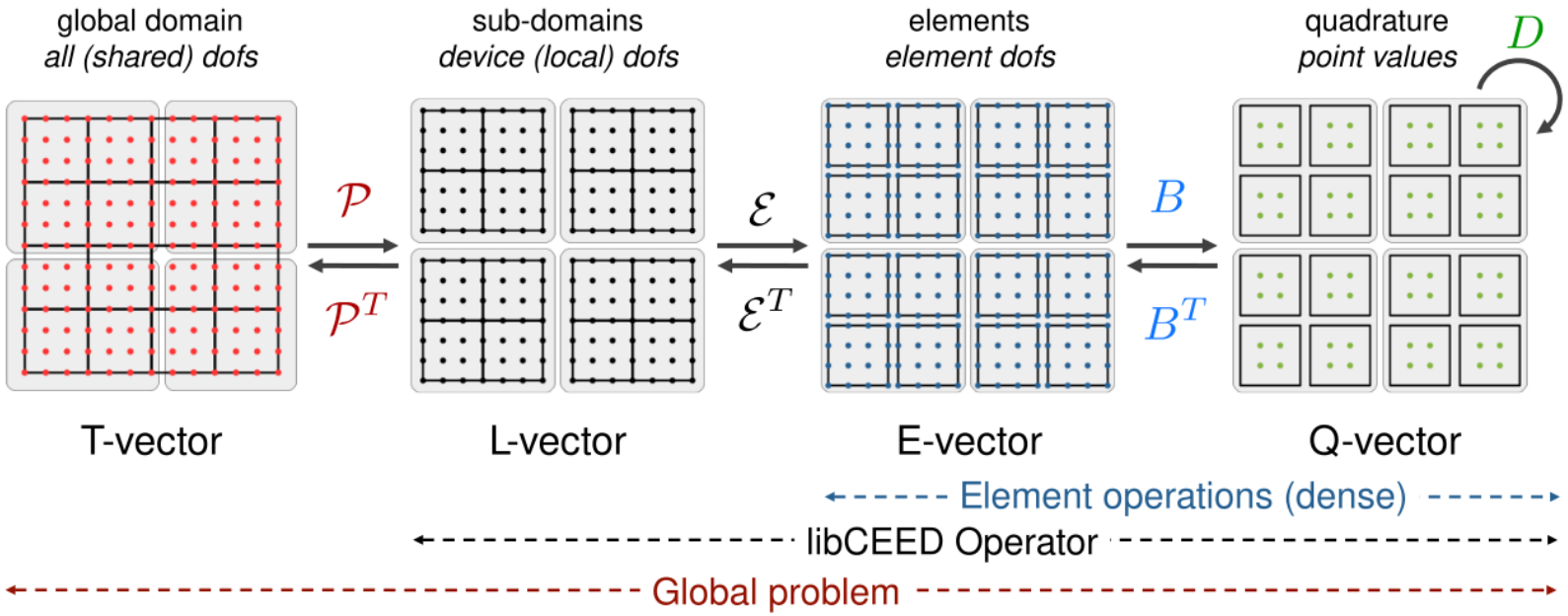
Why the extra work of interpolating to quadrature points?

Better accuracy in discrete integration

Gauss-Legendre accurate for polynomials up to $2n-1$

Third Decision

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



E(lement)-vector represents each 'piece' of our solution polynomial

DoFs on blue dots are red dots on a specific element

B maps to the quadrature points - green dots

D evaluates f_0, f_1, multiply by quadrature weights

Galerkin Form

Galerkin form for an arbitrary second order PDE:

$$\sum_e \mathcal{E}^T \left[(\mathbf{N}^e)^T \mathbf{W}^e \Lambda \left(f_0 \left(u^e, \nabla u^e \right) \right) + \sum_{i=0}^{d-1} (\mathbf{D}_i^e)^T \mathbf{W}^e \Lambda \left(f_1 \left(u^e, \nabla u^e \right) \right) \right] = 0$$

- $\mathcal{E}$ - element assembly/restriction operator
- $\mathbf{N}^e$ - interpolation to quadrature points
- $\mathbf{D}_i^e$ - derivatives at quadrature points
- $\mathbf{W}^e$ - quadrature weights
- Λ - pointwise multiplication at quadrature points
- $u^e = \mathbf{N}^e \mathcal{E}^e u$ and $\nabla u^e = \{\mathbf{D}_i^e \mathcal{E}^e u\}_{i=0}^{d-1}$

Wait, What?!?

Non-linear problem:

$$F\left(u\right) - R = 0$$

Weak form:

$$\left\langle v, f\left(u\right)\right\rangle = \int_{\Omega} v \cdot f_0\left(u, \nabla u\right) + \nabla v : f_1\left(u, \nabla u\right)\, dV = 0$$

Galerkin form:

$$\sum_e \mathcal{E}^T \left[\left(\mathbf{N}^e\right)^T \mathbf{W}^e \Lambda\left(f_0\left(u^e, \nabla u^e\right)\right) + \sum_{i=0}^{d-1} \left(\mathbf{D}_i^e\right)^T \mathbf{W}^e \Lambda\left(f_1\left(u^e, \nabla u^e\right)\right) \right] = 0$$

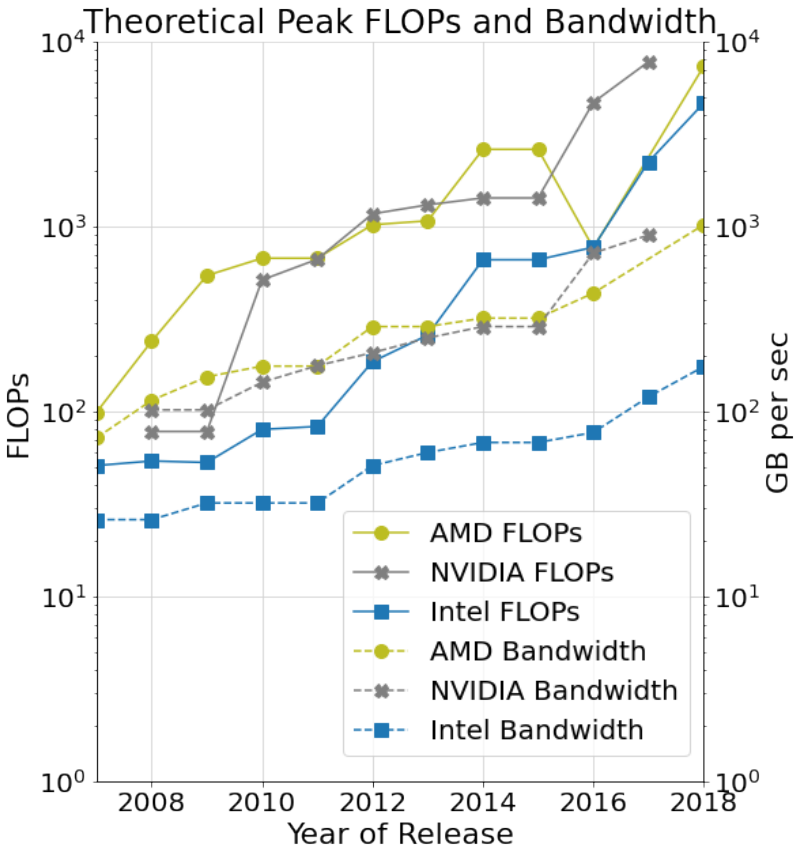
What I'm Handwaving

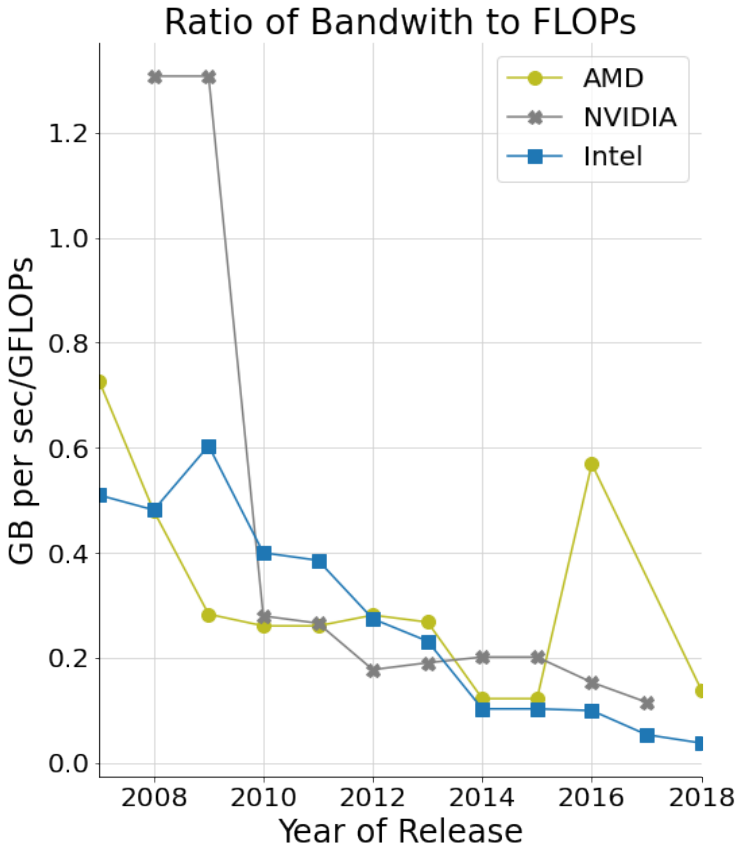
- Elements can have deformed shapes (unstructured meshes)
- Need 'geometric' factors for each individual element
  (Change of coordinate systems from physical to reference)
- $\mathcal{P}$ is parallel communication (supercomputers!)
- R (and its weak analogue) can be considered part of f_0 or separate
- Boundary terms are similar but 1 dimension lower (i.e. 3D $\rightarrow$ 2D)

Traditional Approach

- The Galerkin form naturally creates a sparse matrix for linear $F(u)$
- This sparse matrix can be solved with linear algebra
- Two problems:
  1. $F(u)$ is not usually linear
  2. Sparse matrices don't use hardware optimally
- Big Idea: perform the operations without forming a matrix

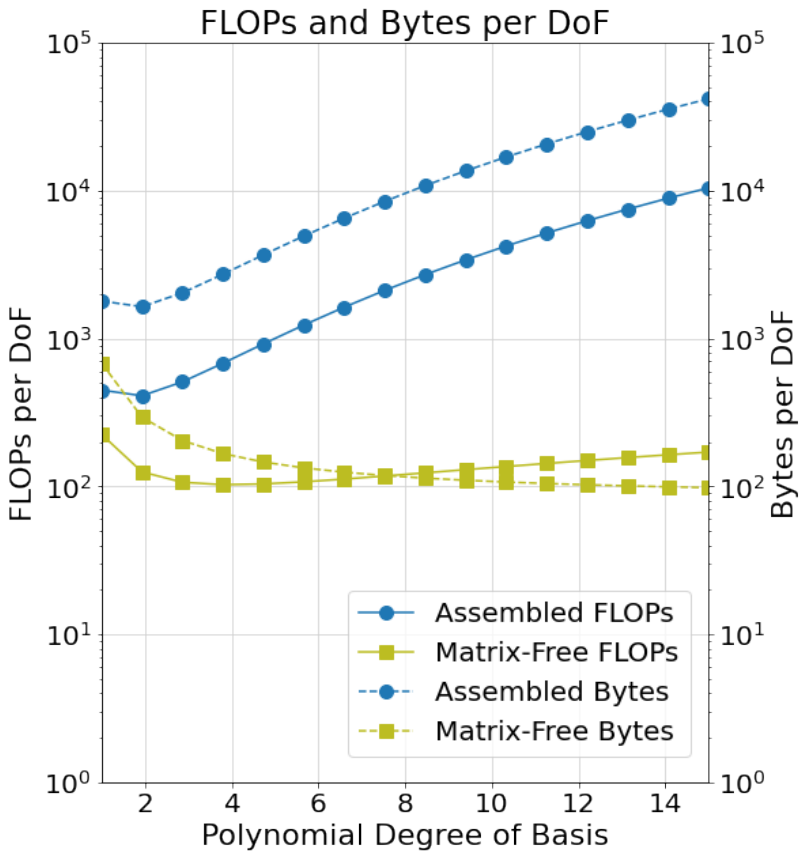
Modern Hardware

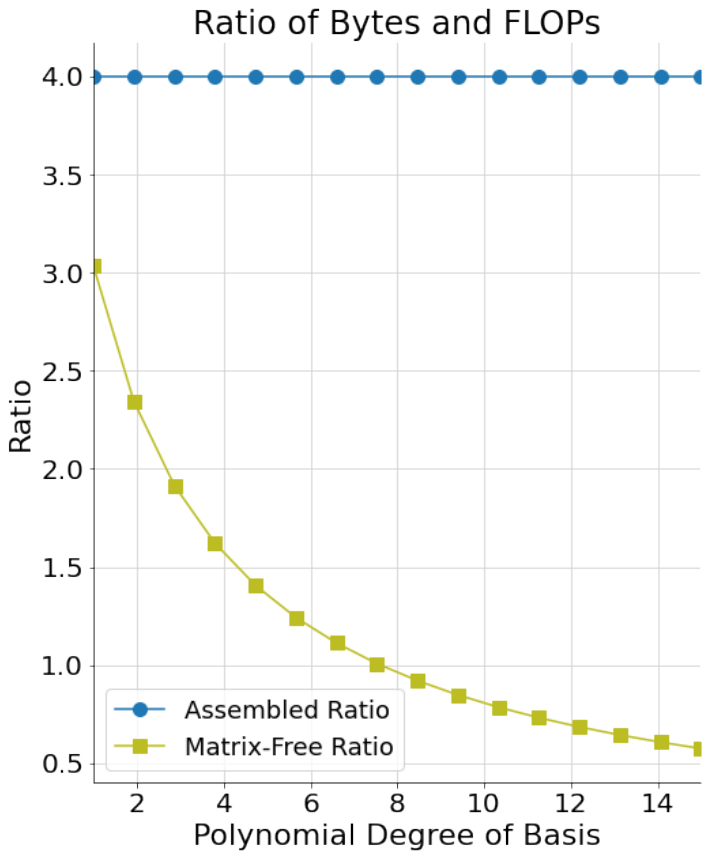




Modern hardware has lower memory bandwidth than FLOPs

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 limitations

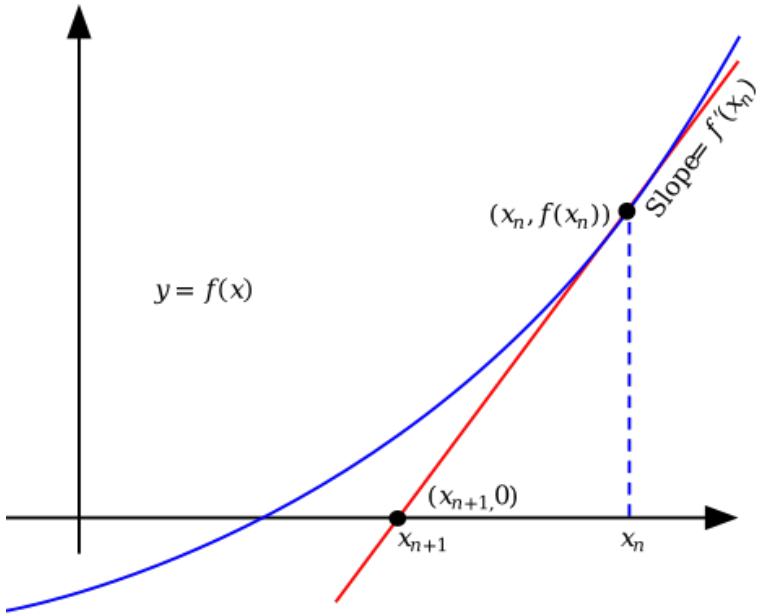
Non-linear Problem

We still have a non-linear problem

Linear algebra solves *linear* problems

Newton's Method

Many non-linear solvers fit into the 'Newton' family



In Newton's method, we form a tangent line

The *line*ar equation is easy to solve

In higher dimensions, the line becomes a (hyper)plane

Newton's Method

We want to solve

$$F(u) - R = 0$$

We can instead solve a series of 'tangent plane' problems

$$F(u_k) - R + J(u_k)(u_k - u_{k+1}) = 0$$

u_{k+1} provides closer estimates of the true solution u_*

We'll denote $w = u_k - u_{k+1}$

A New Problem

Ok, but we still need to form the tangent plane $J(u_k)$

This feels like we keep changing what the problem is!

Beauty

Weak form:

$$\langle v, f(u) \rangle = \int_\Omega v \cdot f_0(u, \nabla u) + \nabla v : f_1(u, \nabla u)\, dV = 0$$

We want to take the derivative of this weak form

Note:

- $\frac{d}{dx}(f(x) + g(x)) = f'(x) + g'(x)$
- $\frac{d}{dx} f(g(x)) = f'(g(x))\, g'(x)$

Beauty

Weak form derivative:

$$\langle v, J(u_k)\, w \rangle = \int_\Omega \begin{bmatrix} v^T & (\nabla v)^T \end{bmatrix} \begin{bmatrix} f_{0,0} & f_{0,1} \\ f_{1,0} & f_{1,1} \end{bmatrix} \begin{bmatrix} w \\ \nabla w \end{bmatrix}$$

Note:

- $f_{i,0} = \frac{\partial f_i}{\partial u}(u_k, \nabla u_k)$
- $f_{i,1} = \frac{\partial f_i}{\partial \nabla u}(u_k, \nabla u_k)$
- Can be discretized the same way as before

Wait, What?!? - Non-linear Equation

Non-linear problem:

$$F\left(u\right) - R = 0$$

Weak form:

$$\left\langle v, f\left(u\right)\right\rangle = \int_{\Omega} v \cdot f_0\left(u, \nabla u\right) + \nabla v : f_1\left(u, \nabla u\right)\, dV = 0$$

Galerkin form:

$$\sum_e \mathcal{E}^T \left[\left(\mathbf{N}^e\right)^T \mathbf{W}^e \Lambda\left(f_0\left(u^e, \nabla u^e\right)\right) + \sum_{i=0}^{d-1} \left(\mathbf{D}_i^e\right)^T \mathbf{W}^e \Lambda\left(f_1\left(u^e, \nabla u^e\right)\right) \right] = 0$$

Wait, What?!? - Linear Equation

Linearized problem:

$$F\left(u_k\right) - R + J\left(u_k\right)\left(u_k - u_{k+1}\right) = 0$$

Weak form:

$$\left\langle v, J\left(u_k\right) w \right\rangle = \int_\Omega \left[\begin{array}{cc} v^T & \left(\nabla v\right)^T \end{array} \right] \left[\begin{array}{cc} f_{0,0} & f_{0,1} \\ f_{1,0} & f_{1,1} \end{array} \right] \left[\begin{array}{c} w \\ \nabla w \end{array} \right]$$

Galerkin form:

$$\sum_e \mathcal{E}^T \left[\left(\mathbf{N}^e\right)^T \mathbf{W}^e \Lambda\left(f_0^*\left(w^e, \nabla w^e\right)\right) + \sum_{i=0}^{d-1} \left(\mathbf{D}_i^e\right)^T \mathbf{W}^e \Lambda\left(f_1^*\left(w^e, \nabla w^e\right)\right) \right] = 0$$

A New Problem

We have a discrete system we can solve efficiently

But we still have a problem

(Our problems are getting easier to tackle though!)

Condition Number

Matrix-free representation of the linear problem restricts solvers

Krylov subspace methods only need the matrix-vector product Ax,
not the assembled matrix A

But Krylov subspace methods are sensitive to condition number

High-order finite elements are more efficient matrix-free,
but high-order finite elements have a higher condition number

Preconditioning

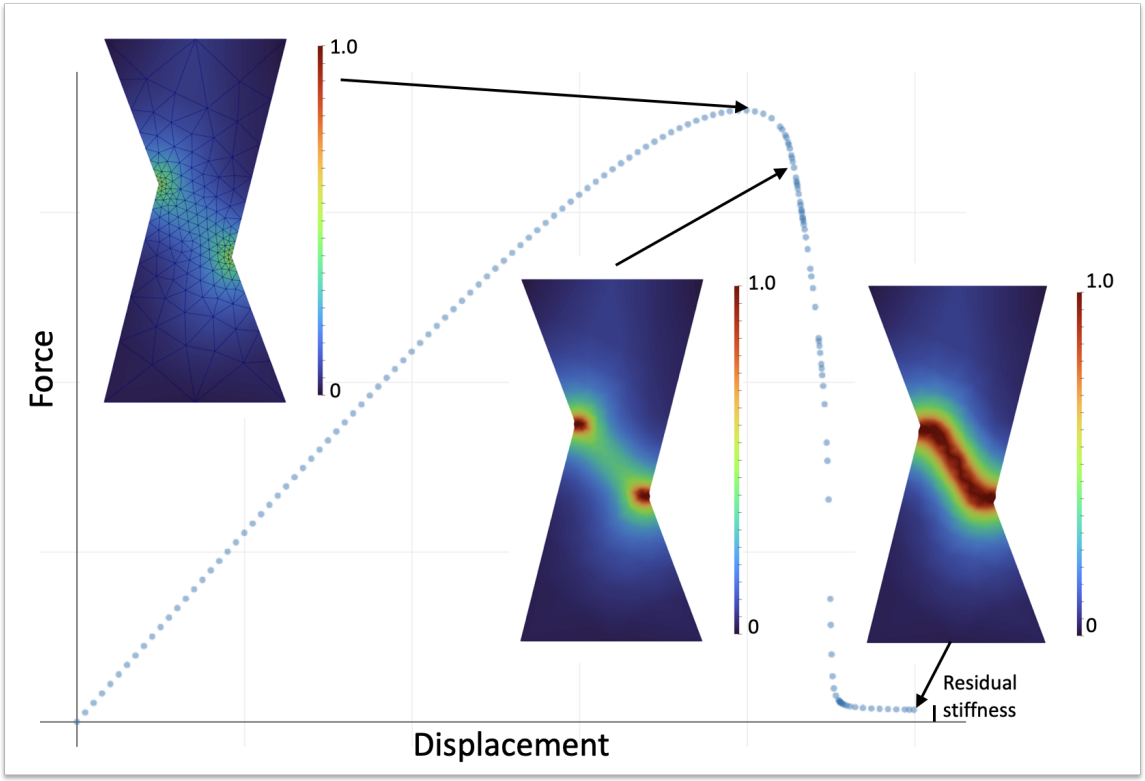
Preconditioning improves the condition number of A

Big Idea: $P^{-1}A \approx I$ has a better condition number

Challenge: Finding efficient P^{-1} can difficult

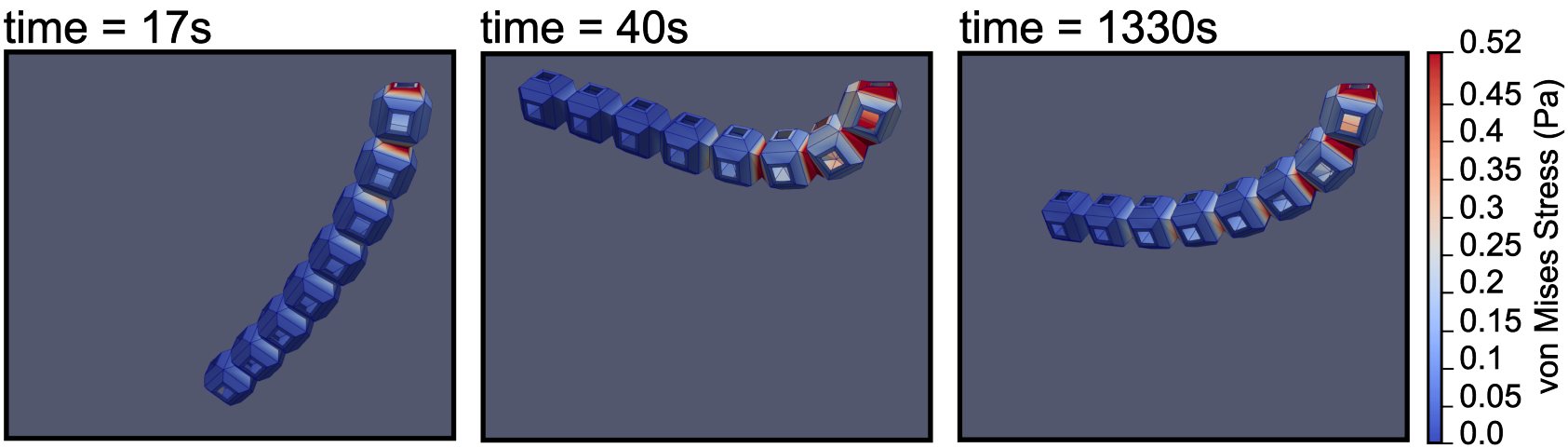
The Fix: This is a rich research area with lots of options

Example - Brittle Fracture



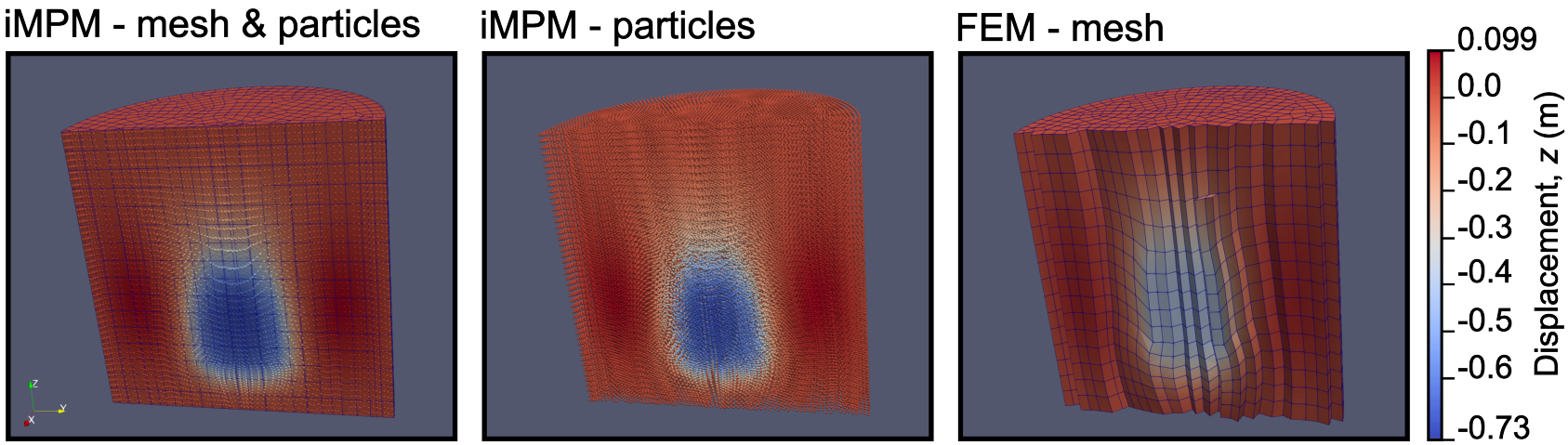
Quasistatic simulation of a compressive shear test for a generic brittle material

Example - Schwarz Pendulum



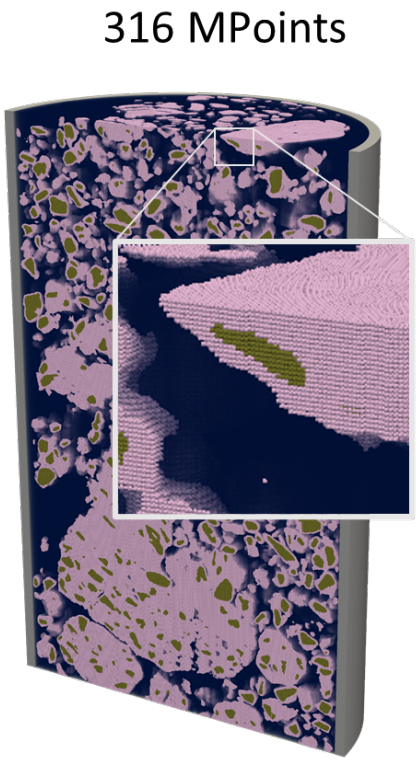
Dynamic simulation of 8 Schwarz-P cells with an applied traction force creating pendulum-like motion

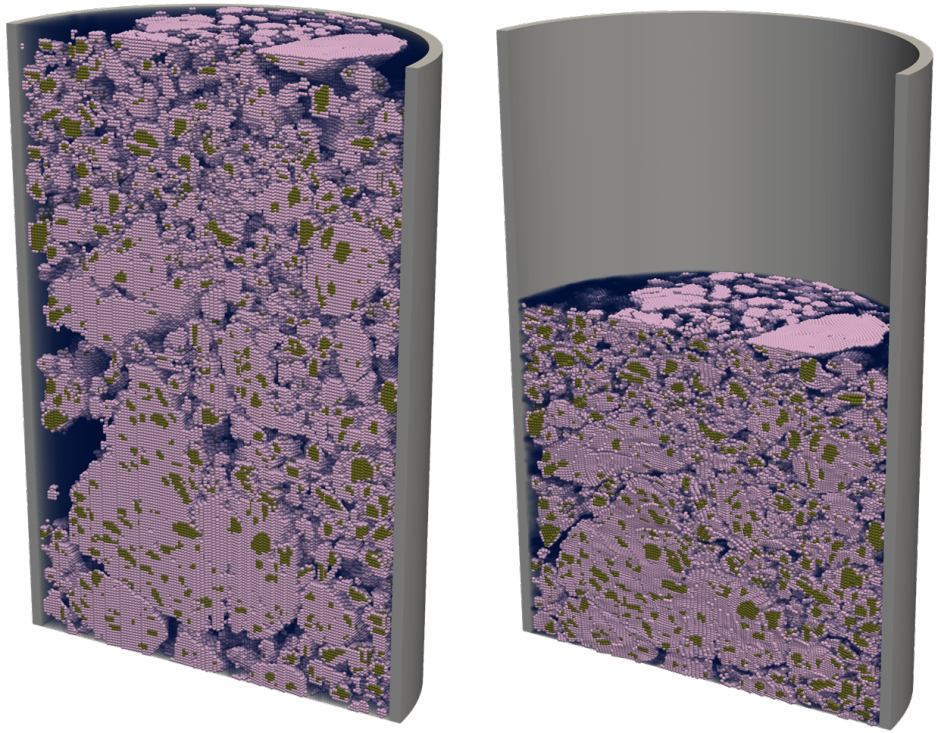
Example - Sinker Simulation



Quasistatic simulation of a cylinder with a dense high-modulus inclusion surrounded by a low-density low-modulus near-incompressible "foam"

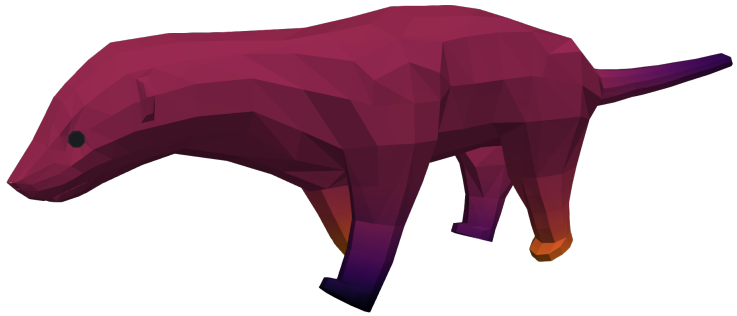
Example - Press Simulation





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

Questions?



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

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

Motivating Matrix-Free Finite Elements

Jeremy L Thompson

University of Colorado Boulder

jeremy@jeremylt.org

19 Mar 2026