Automatic parameter sensitivities in Serac for engineering applications

Michael Tupek

Thanks to: Jamie Bramwell, Sam Mish, Brandon Talamini, Eric Chin, Chris White, Alex Chapman, LiDO team

Serac: HPC engineering software ecosystem

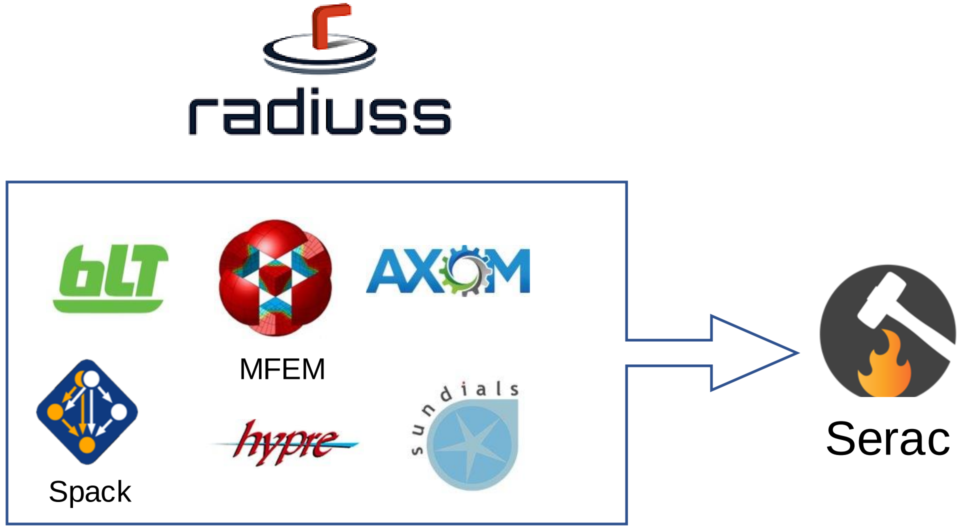
Modular physics solvers: composable simulation blocks

Modern simulation workflows: easy low-level APIs for data science applications

Differentiable by design: Sensitivity analysis via automatic differentiation for arbitrary parameterizations

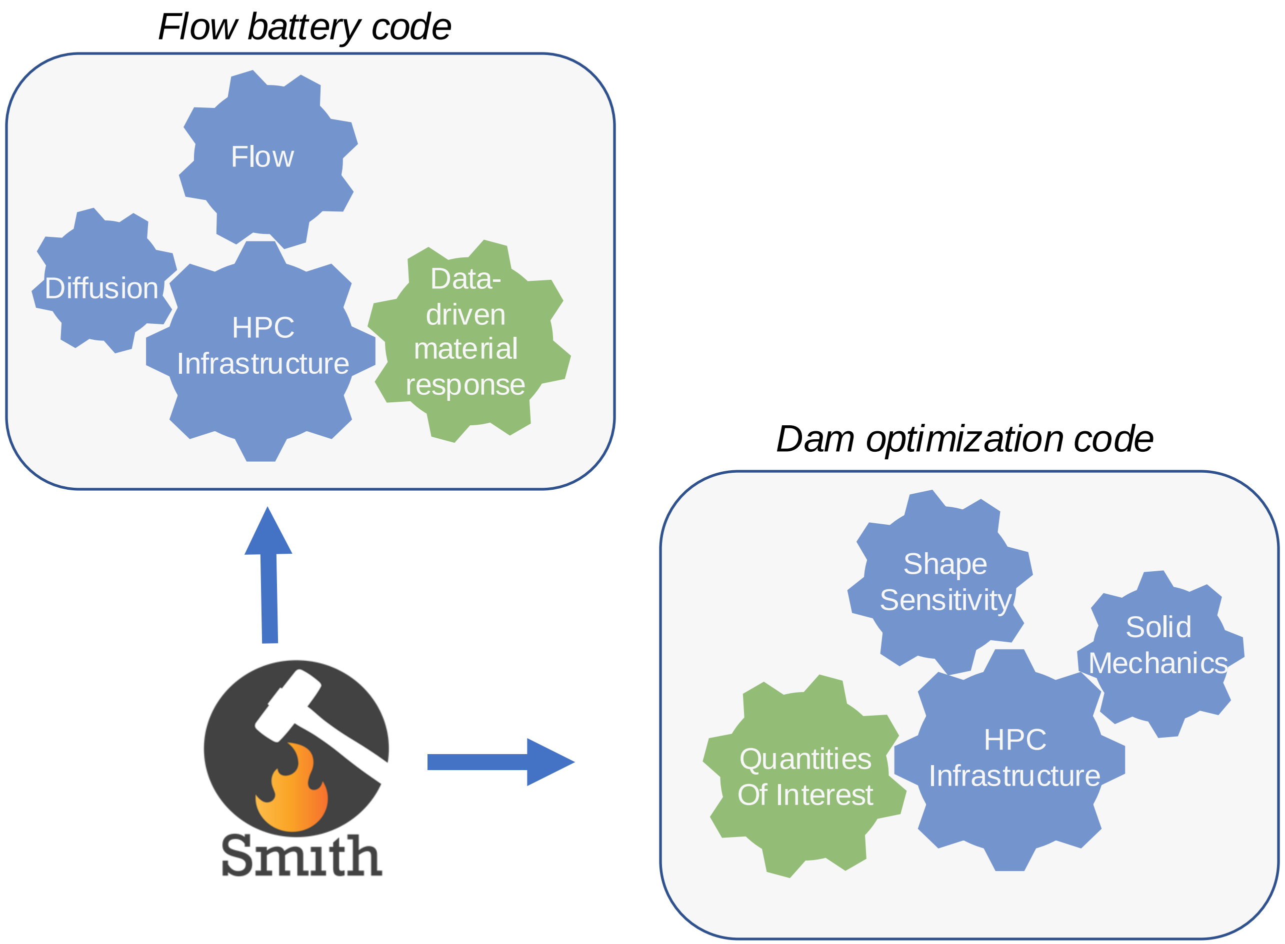
Rapid development: New capabilities in months, not years

Software quality: Modern software standards to ensure sustainability



Leverage and improve *LLNL institutional HPC software*

Modular physics capabilities allow agile development



Current physics (PDE) modules

- Large deformation solid mechanics
- Heat transfer
- Porous electrodes
- Helmholtz filters
- Steady-state incompressible flow
- Wave equation

Core discretization capabilities on GitHub

https://github.com/LLNL/serac

Serac

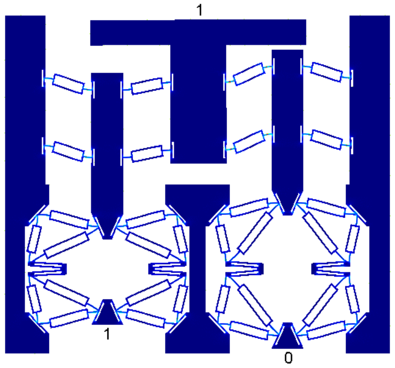
Reuse modular components to create specialized sustainable applications

Thermo-mechanical simulations

Applications

- Large deformations
- Buckling
- Nearly incompressible plasticity
- Fracture
- Contact
- Complex Geometry

Buckling of mechanical logic gates

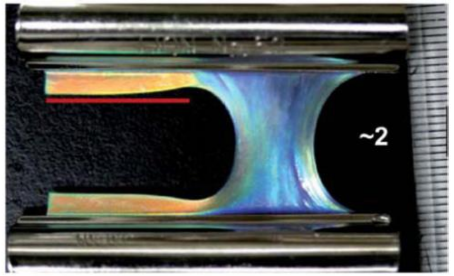


Y Song, et al, *Nature Communications* 10 (1), 882

Vehicle crash

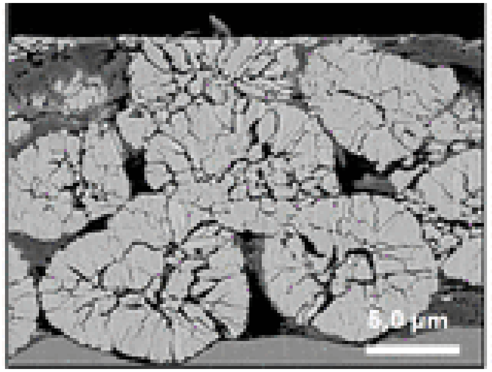
P. Wriggers, *Computational Contact Mechanics*, Springer 2002

Viscoelasticity/Plasticity



M Haque et al. *Soft Matter* (2012) 8 8008–16

Chemo-mechanical fracture of a battery electrode



T. Hiro et al. *Int J Impact Engrg* 35 (2008) 1578-1586

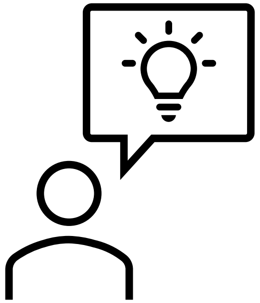
Traditional design loops can be expensive and nonoptimal, and the larger design spaces afforded by AM only increase complexity

Input: Material properties, loads, dimensions …

Simulation

Output: Mass, stress, strain, derived quantities …

- Trial and error
- Experience & intuition
- Time consuming
- Not optimal
- Incremental improvements



Output: Material properties, loads, dimensions …

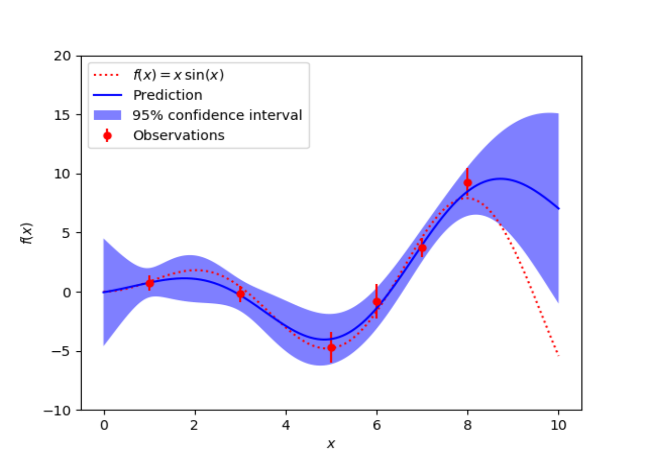
Design

Input: Mass, stress, strain, quantities of interest …

Gradient-enabled multiphysics simulation codes will allow much larger design space explorations

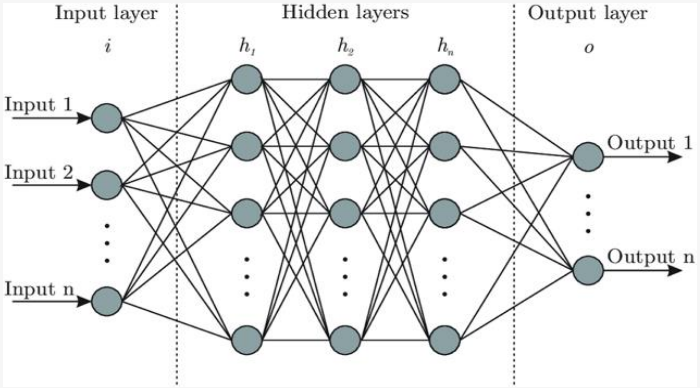
Black Box (Gradient-Free)

- ✓Can use existing simulation tools
- ✓Non-intrusive for code
- ✓Good for exploration
- Requires many simulation samples
- Limited to O(10) design parameters



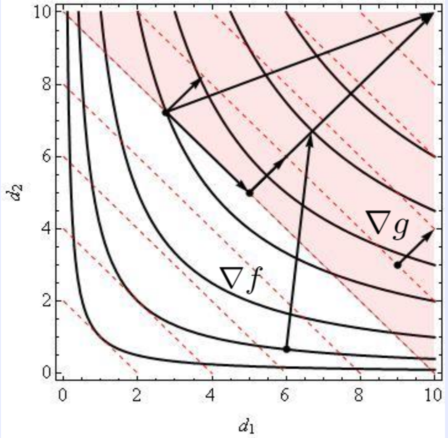
ML Surrogate

- ✓Can use existing simulation tools
- ✓Non-intrusive for code
- ✓Surrogate has gradients
- ML training can be expensive
- Hard to modify design space
- Limited to O(10) design parameters

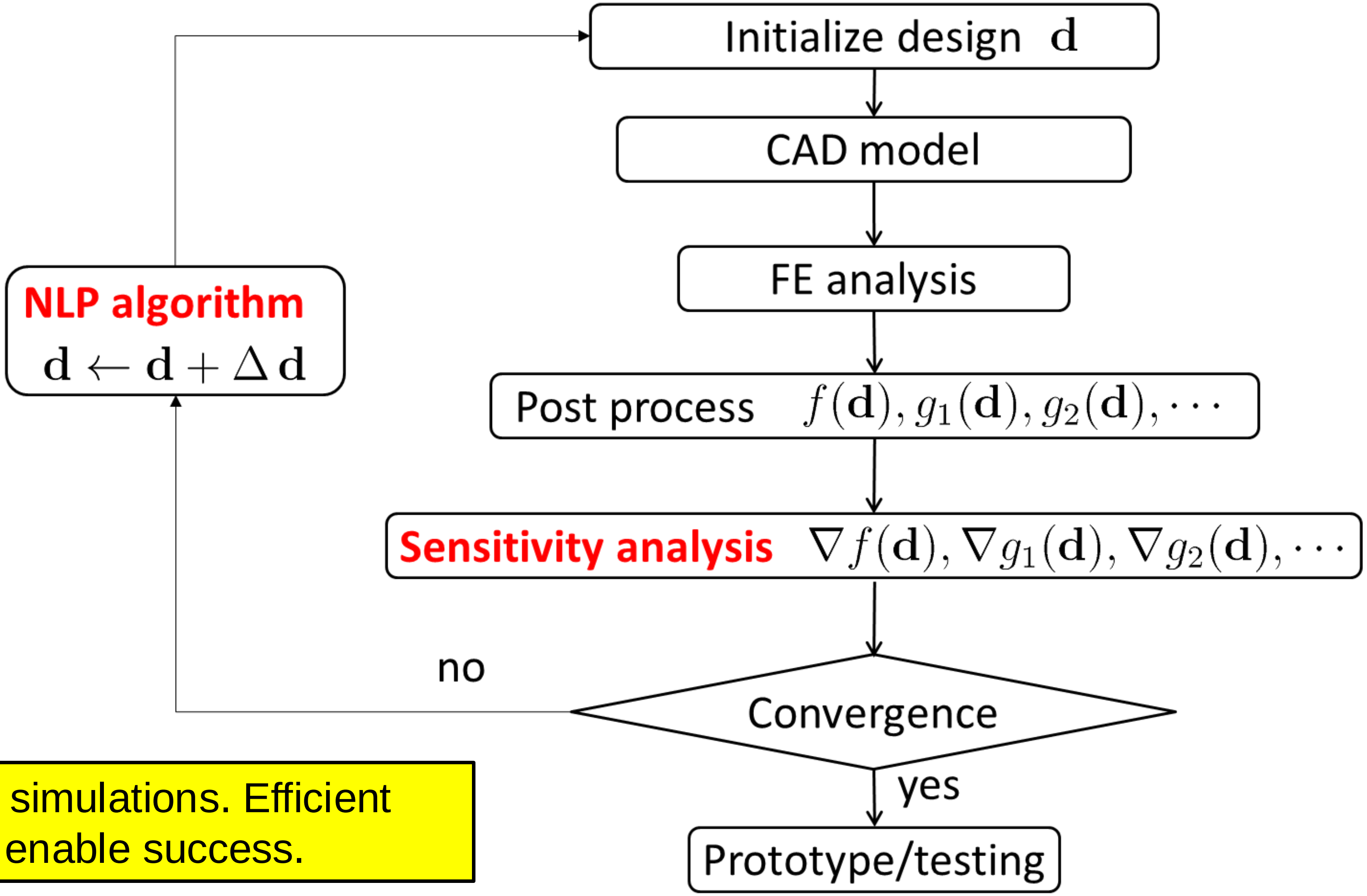


Gradient-Based

- ✓Large parameter space, O(1M)
- ✓Fast convergence
- ✓Agile parameterizations
- ✓Provable local optimality
- Requires gradients
- Code Intrusive

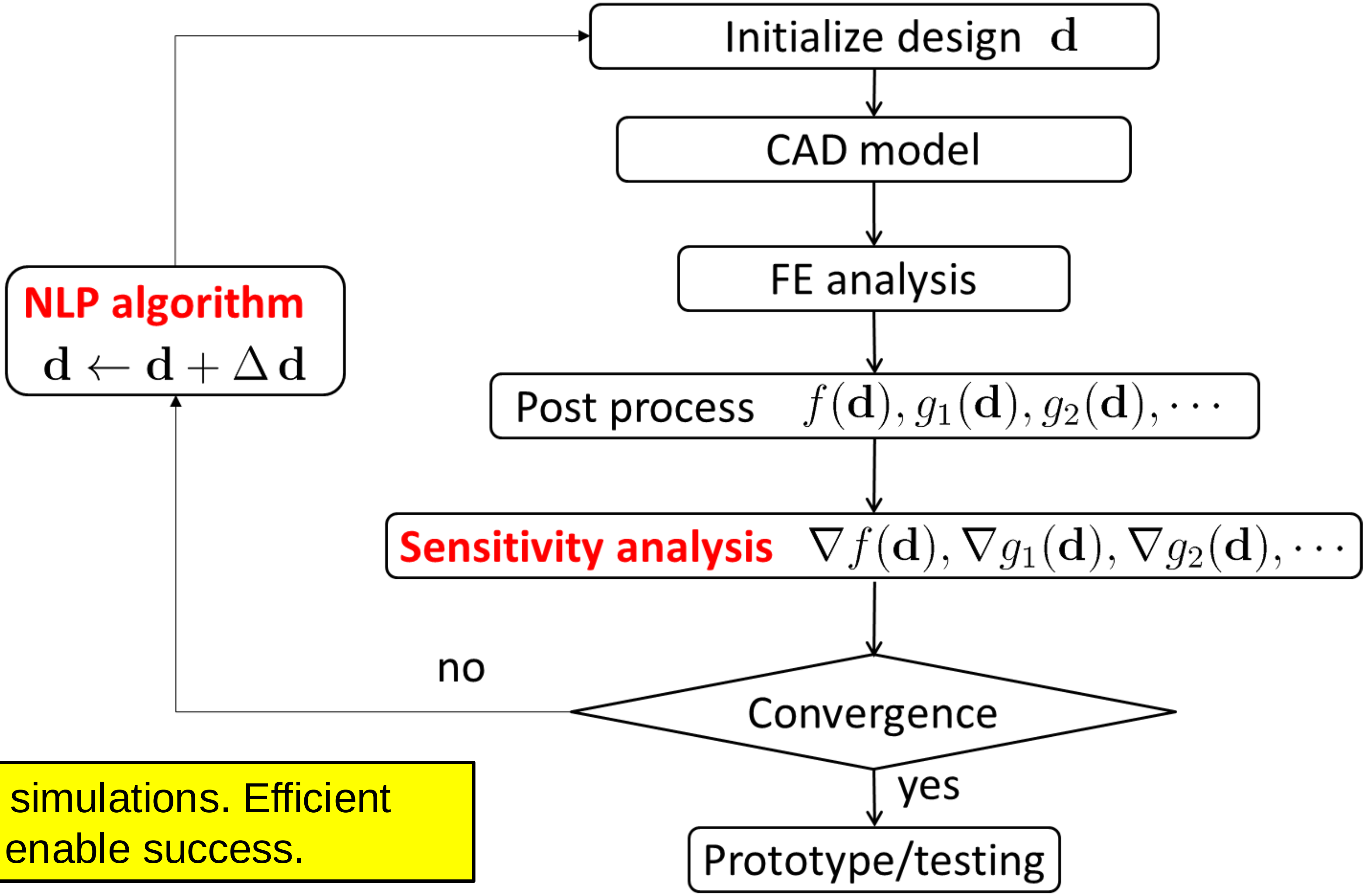


Our focus is *gradient-enabled* design optimization



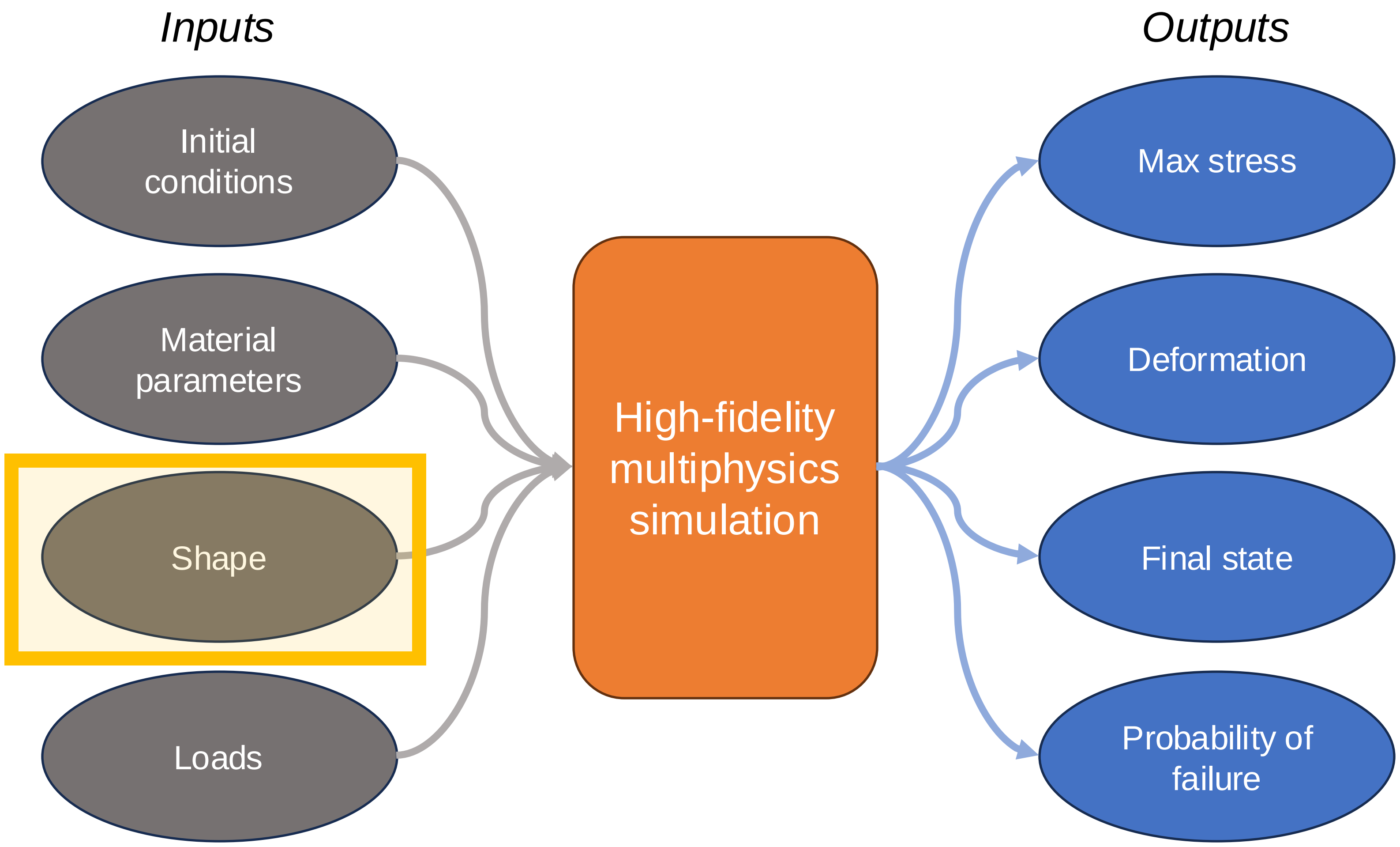
Each loop requires costly simulations. Efficient algorithms are needed to enable success.

Our focus is *gradient-enabled* design optimization



Each loop requires costly simulations. Efficient algorithms are needed to enable success.

Derivatives of simulation codes greatly expand their usefulness

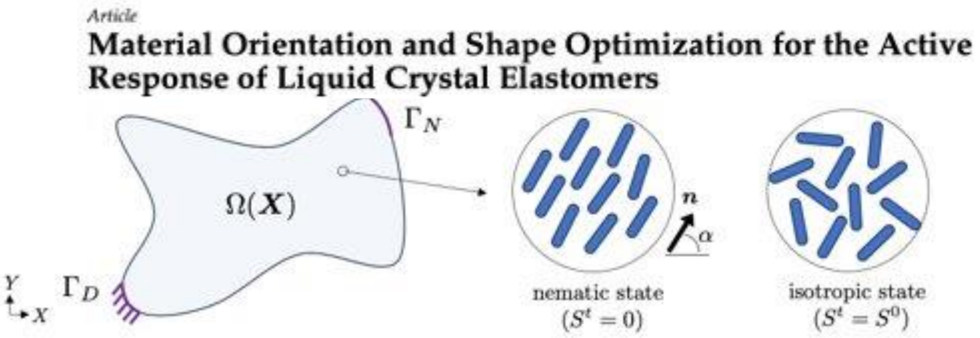


Want $\dfrac{\partial\ \text{input}}{\partial\ \text{output}}$ for:

- Generative design optimization
- Robust uncertainty quantification
- Automated model calibration
- Inverse problems

Consider the derivatives required for design optimization

- Nonlinear solid mechanics
  - Material nonlinearity
  - Geometric nonlinearity
  - Implicit dynamics
- Liquid crystal elastomer material model
  - Designed anisotropy from additive manufacturing
- Shape and parameter derivatives



$$\int_{\Omega_0} \sigma(\mathbf{u}) \cdot \left(\nabla_{\mathbf{X}} \delta \mathbf{v} \mathbf{F}^{-1}\right) \det \mathbf{F} \, dV_0 - \int_{\Omega_0} \rho_0 \mathbf{b} \cdot \delta \mathbf{v} dV_0$$

$$- \int_{\Gamma_{N_0}} \delta \mathbf{v} \cdot \mathbf{t} \, ||\mathbf{F}^{-T} \mathbf{n}_0|| \det \mathbf{F} \, dA_0 + \int_{\Omega_0} \rho_0 \ddot{\mathbf{u}} \cdot \delta \mathbf{v} \, dV_0 = 0, \quad \forall \delta \mathbf{v} \in \hat{\mathbf{U}}$$

Derivatives are also needed for adjoint methods

Consider the PDE-constrained optimization problem:

$$\text{minimize } \mathcal{Q}(u, p) \text{ such that } \mathcal{A}(u, p) = 0$$

$\mathcal{Q}$	Quantity of interest
$\mathcal{A}$	Nonlinear partial differential operator
u	Primal state variables
p	Parameters

For gradient-based optimization, we need

$$\frac{d\bar{Q}}{dp} = \frac{\partial Q}{\partial p} + \frac{\partial Q}{\partial u}\boxed{\frac{\partial u}{\partial p}}$$

HARD TO CALCULATE!

Derivatives are also needed for adjoint methods

Form the Lagrangian:

$$\mathcal{L}(u,p,\lambda) = \mathcal{Q}(u,p) + \lambda^* \mathcal{A}(u,p)$$

- One adjoint solve per QOI
- Adjoint solve always linear
- Linearization often needed for state

Compute the sensitivity by finding a stationary point:

$$\mathcal{L}_\lambda = 0 \quad \Longrightarrow \quad \mathcal{A}(u,p) = 0 \qquad \text{state equation (solve for u)}$$

$$\mathcal{L}_u = 0 \quad \Longrightarrow \quad \boxed{\mathcal{Q}_u} + \boxed{\mathcal{A}_u^*(u,p)}\lambda = 0 \qquad \text{adjoint equation (solve for } \lambda)$$

$$\mathcal{L}_p = \frac{d\mathcal{Q}}{dp} \quad \Longrightarrow \quad \boxed{\mathcal{Q}_p} + \lambda^* \boxed{\mathcal{A}_p(u,p)} = \frac{d\mathcal{Q}}{dp} \qquad \text{sensitivity calculation (compute for each } p)$$

Use forward mode auto diff for discrete sensitivities

Options for differentiation of algorithms

Finite Difference (Numerical Derivatives)

- **Pros**
  - Simple
  - Works for existing implementations
- **Cons**
  - Catastrophic cancellation
  - Bad performance

$$\frac{\partial y}{\partial x} \approx \frac{y(x+\epsilon) - y(x-\epsilon)}{2\epsilon}$$

Analytical Derivation (Symbolic Derivatives)

- **Pros**
  - Great performance
  - Accurate derivatives
- **Cons**
  - A lot of work to implement
  - Easy to make subtle mistakes

$$f'(x) = \frac{\exp(x)(x+x^3)\cos(2-\exp(x)) - (x^2-1)\sin(2-\exp(x))}{(1+x^2)^2}$$

Automatic Differentiation

- **Pros**
  - Accurate derivatives
  - Easy to implement and use
  - Harder to make mistakes
- **Cons**
  - Not as performant as the manual option
  - Requires source changes

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial f}\frac{\partial f}{\partial g}\frac{\partial g}{\partial h}\frac{\partial h}{\partial x}$$

Use finite element interpolation to discretize parameters

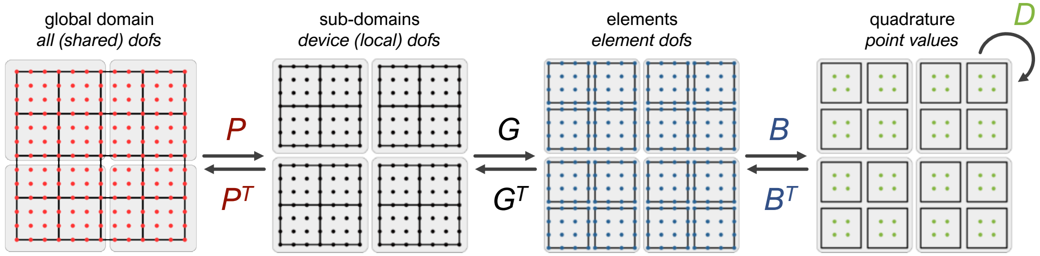
Parameters

$\hat{\rho} = B_\rho G_\rho P_\rho \, \rho$

Parametric nonlinear residual

$A(u;\rho) = P^T G^T B^T \, D(\hat{u},\hat{\rho})$

Differentiate at Quadrature points only via autodiff!



$$\nabla_u A(u;\rho) = P^T G^T B^T \, \nabla_{\hat{u}} D(\hat{u},\hat{\rho})$$

```
template <typename gradient_type>
struct dual {
  double        value;
  gradient_type gradient;
};
```

Smith

MFEM + Enzyme –
Coming soon!

Functional: A core enabling technology for differentiable finite element kernels

Consider an arbitrary nonlinear finite element residual operator:

$$r(u_1, u_2) = \int_\Omega \left(\mathrm{source}(u_1, u_2)\,\phi + \mathrm{flux}(u_1, u_2) \cdot \nabla\phi\right)\, dV$$

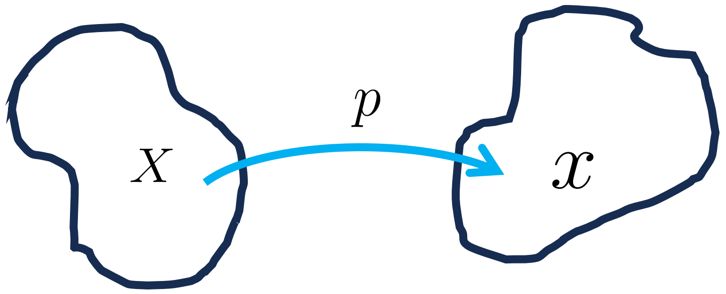
```
residual.AddVolumeIntegral(
  [f, s](auto x, auto arg1, auto arg2){
    auto source = s(x, arg1, arg2);
    auto flux = f(x, arg1, arg2);
    return serac::tuple{source, flux};
  },
  domain_of_integration
);

auto [result, grad_u_1] = residual(differentiate_wrt(u_1), u_2);
```

- Handles quadrature state
- Scalar QOI capable
- Requires C++17

✓ Readable
✓ Scalable
✓ Flexible
✓ Differentiable

Shape-Aware Functional: A wrapper for calculating conformal shape derivatives

H1 vector-valued *shape displacement* parameter $p = \sum_{i=0}^{N} p_i \boldsymbol{\phi}_\mathbf{i}$



Transformations handled automatically by **`ShapeAwareFunctional`**

$$x = X + p$$

$$r(p,u) = \int_{\boxed{\Omega_p}} \left(\psi \cdot s(x, u, \nabla_x u) + \nabla_x \psi : f(x, u, \nabla_x u)\right)\, dx$$

$$= \int_{\boxed{\Omega}} \left(\psi \cdot s\left(X + p, u, \frac{\partial u}{\partial X}\left(I + \frac{\partial p}{\partial X}\right)^{-1}\right) + \left(\frac{\partial \psi}{\partial X}\left(I + \frac{\partial p}{\partial X}\right)^{-1}\right) : f\left(X + p, u, \frac{\partial u}{\partial X}\left(I + \frac{\partial p}{\partial X}\right)^{-1}\right)\right) \det\left(I + \frac{\partial p}{\partial X}\right) dX$$

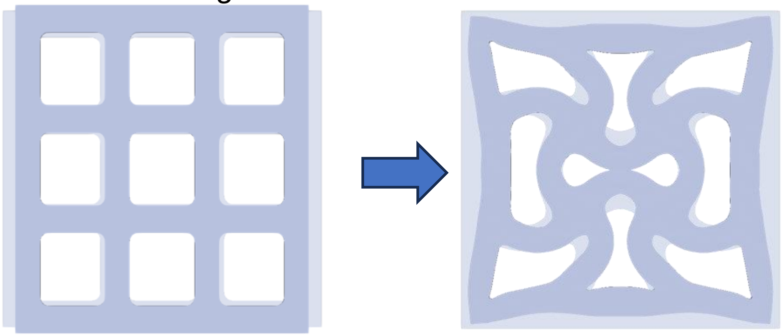
Every mesh node is now a design parameter!

Example: Parameter integral via shape-aware functional

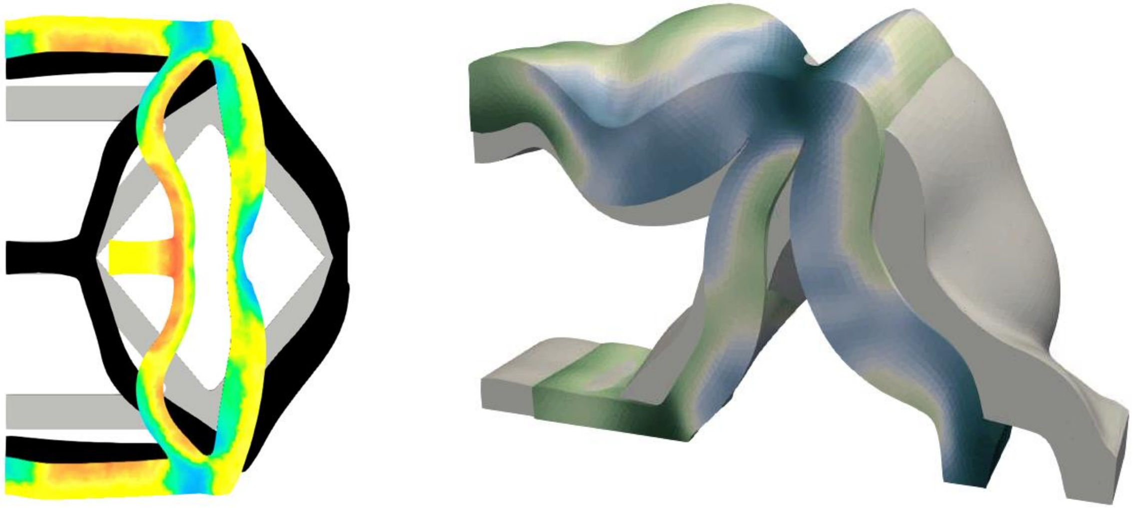
```cpp
// Define the shape-aware QOI object
serac::ShapeAwareFunctional<shape_space, double(parameter_space)> serac_qoi(shape_fes, parameter_fes);

// Note that the integral does not have a shape parameter field. The transformations are handled under the hood
// so the user only sees the modified x = X + p input arguments
serac_qoi.AddDomainIntegral(
    serac::Dimension<dim>{}, serac::DependsOn<0>{},
    [](double /*t*/, auto /*x*/, auto param) { return serac::get<0>(param); }, whole_mesh);

// Note that the first argument after time is always the shape displacement field
auto [result, grad] = serac_qoi(t, differentiate_wrt(shape_displacement), parameter);
```

We are rapidly delivering novel results to new customers

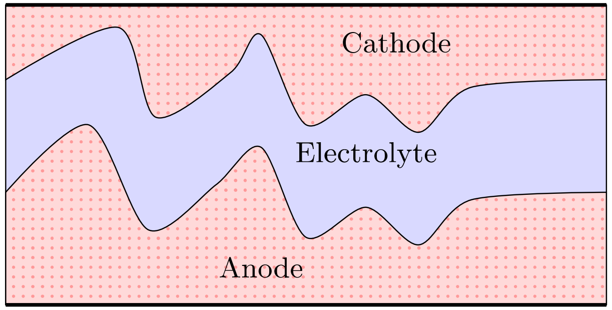
Jorge-Luis Barrera

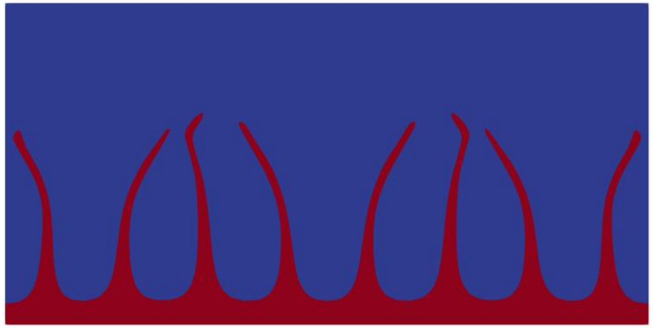


Design of 3D printed liquid crystal responsive elastomer structures



We are currently leveraging this software stack to design porous electrodes via shape optimization





Hanyu Li

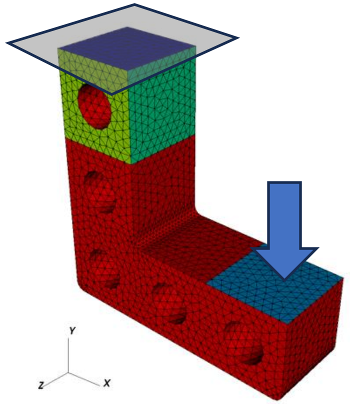
Optimal porous electrode design

$$-\nabla \cdot \left(\sigma^{\mathrm{eff}} \nabla \Phi_1\right) = -a(i_r + i_c)$$

$$-\nabla \cdot \left(\kappa^{\mathrm{eff}} \nabla \Phi_2\right) - z_+ \nu_+ F \nabla \cdot \left(\left(D_+^{\mathrm{eff}} - D_-^{\mathrm{eff}}\right) \nabla c\right) = a(i_r + i_c)$$

$$\frac{\partial(\epsilon c)}{\partial t} - \nabla \cdot (D \nabla c) = \frac{s_+}{F n \nu_+} t_- a i_r + \frac{1}{F z_+ \nu_+}\left(t_- \frac{d q_+}{d q} - t_+ \frac{d q_-}{d q}\right) a i_c$$

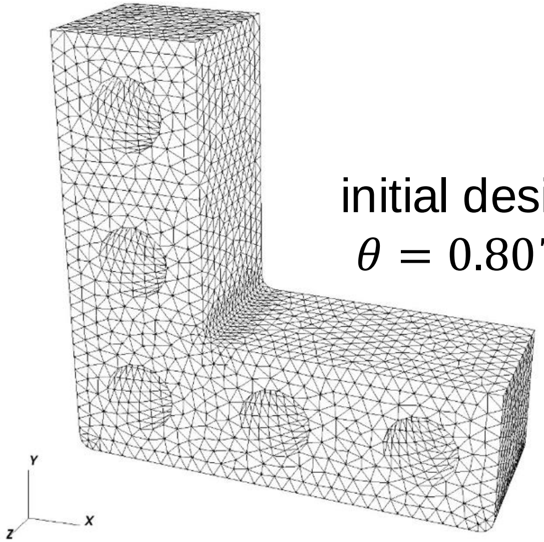
Free-form shape optimization with design control



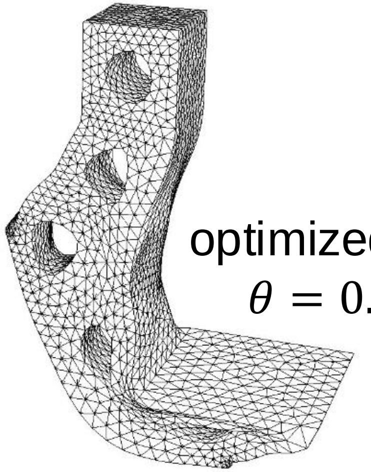
- Use shape (topology-preserving) optimization to *minimize mass subject to maximum stress constraint*

- Demonstrate use of design constraints
  - Rectangular top mount
  - Min (+) feature = 0.1
  - Min (-) feature = 0.2
  - Top circular hole (variable position/radius)

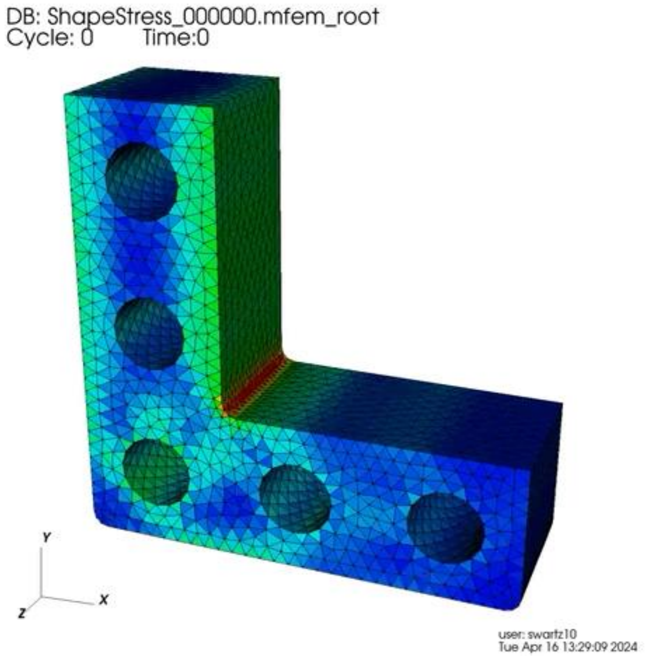
Kenny Swartz



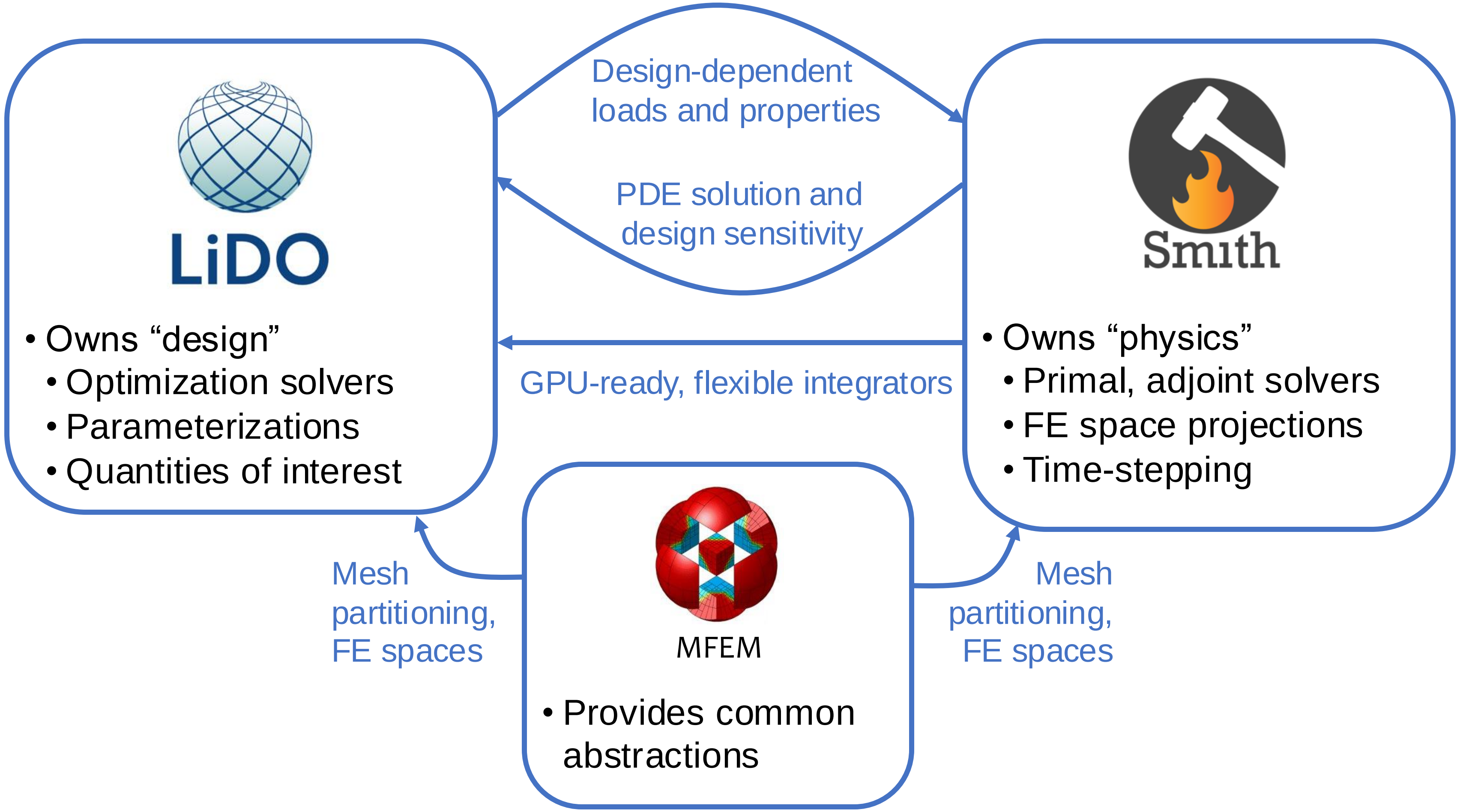
initial design
$\theta = 0.807$



optimized design
$\theta = 0.154$



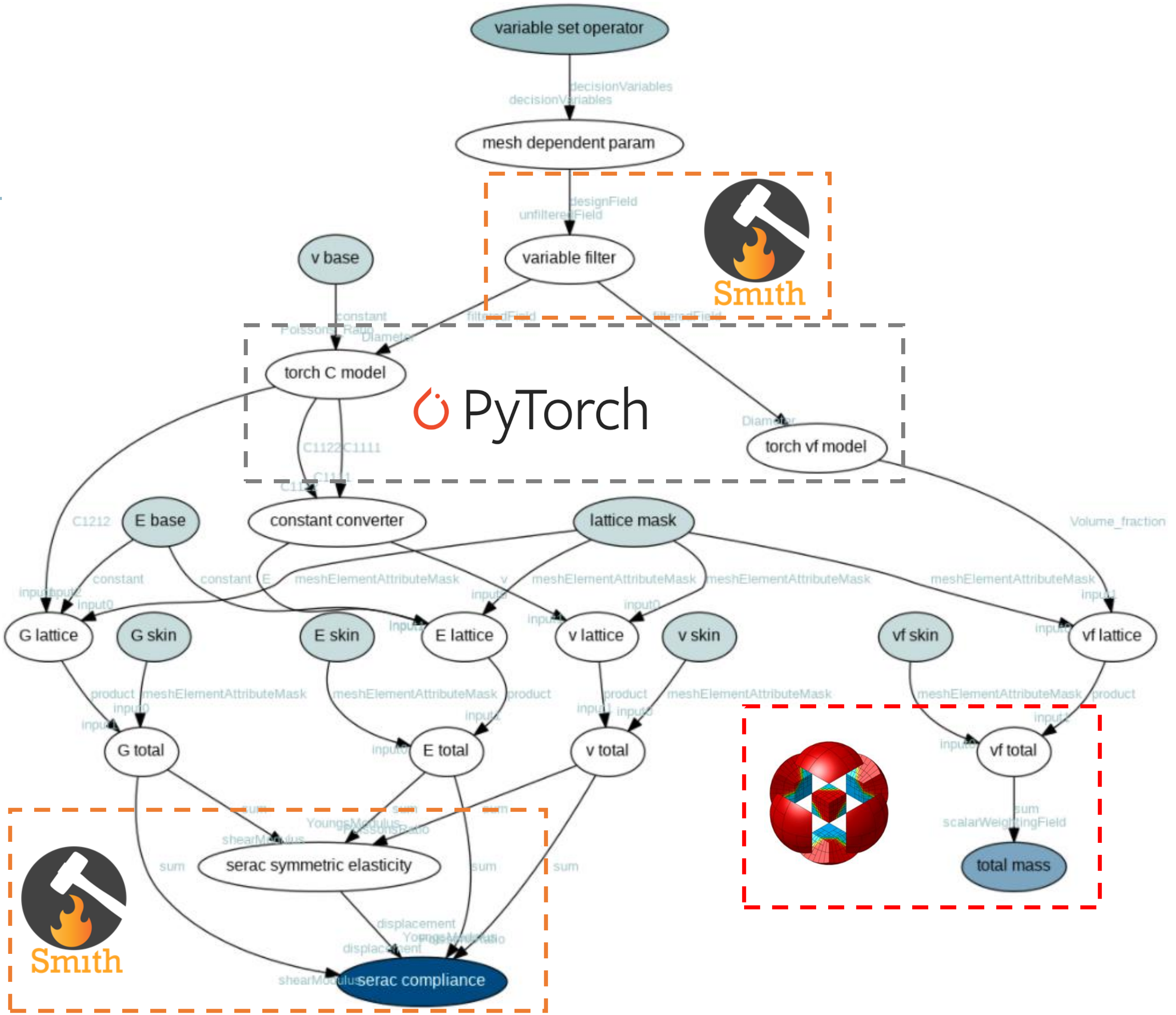
LiDO + Smith: a suite of tools for automated gradient-based design and optimization



LiDO

Automated discrete adjoint analysis via LiDO

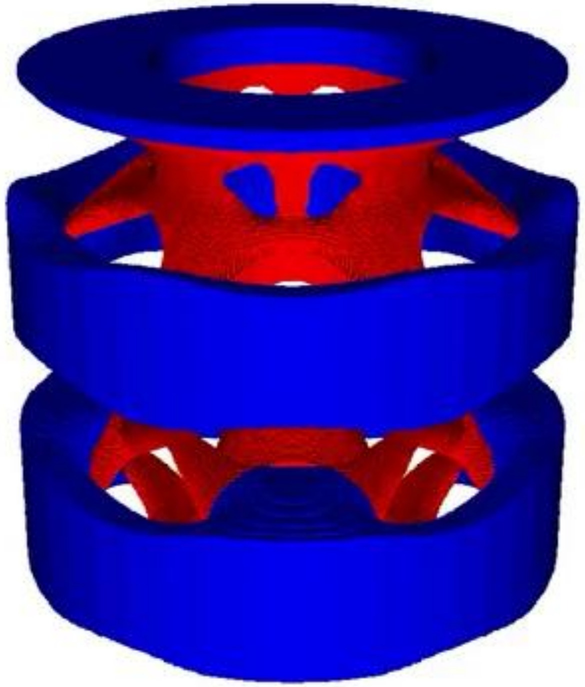
Reverse mode autodiff via LiDO!



This framework can be rapidly adapted for cutting-edge research and applications

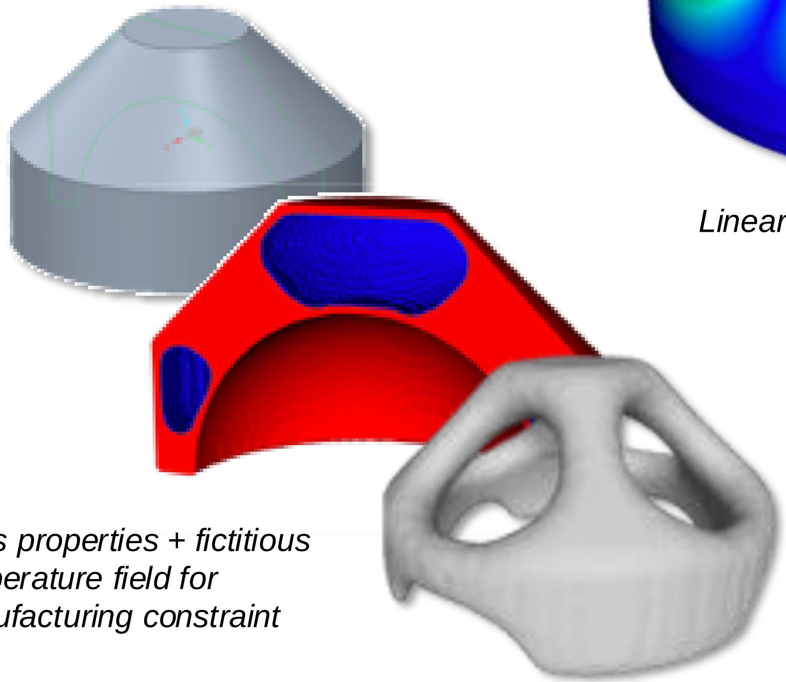
Optical sensor housing with negative thermal expansion

Smith + LiDO

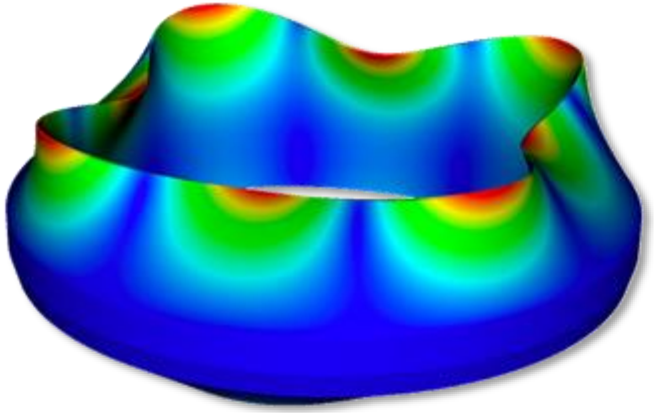


Whiteboard sketch to delivered design in one week!

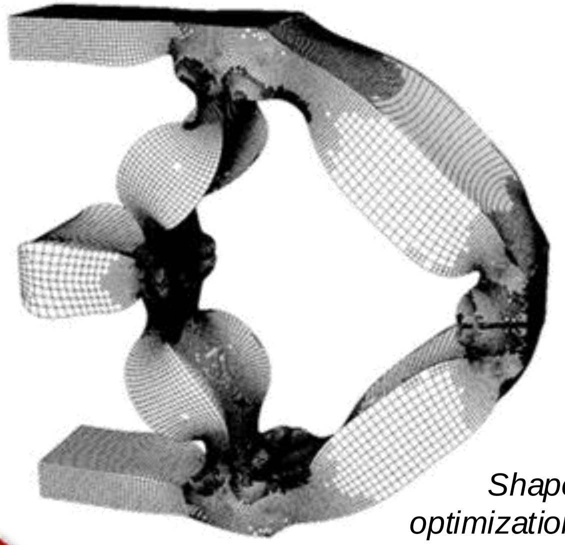
LiDO/Smith has now been used in a variety of static problems, including multiple materials, nonlinear materials, manufacturing constraints…



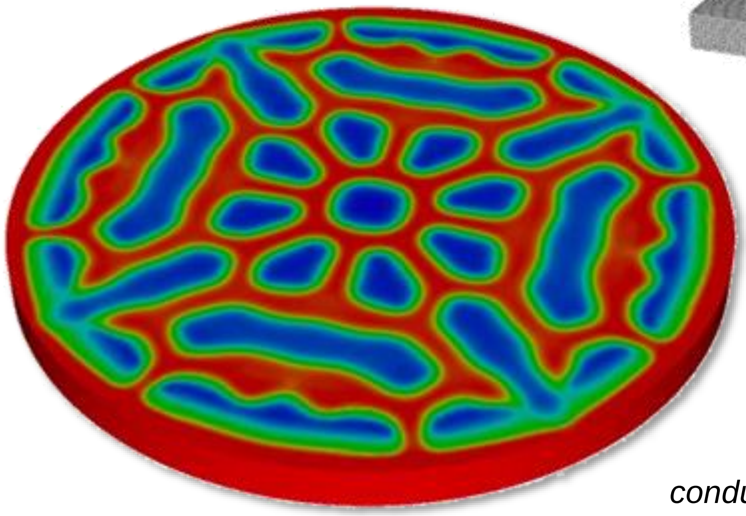
Mass properties + fictitious temperature field for manufacturing constraint



Linear resonance



Shape optimization with stress constraint



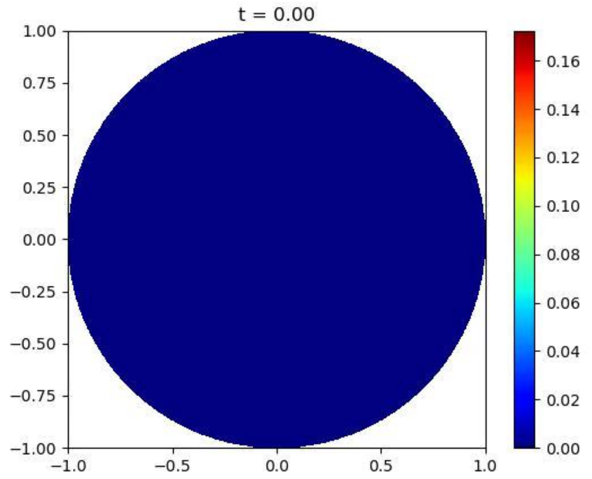
Weakly coupled conduction and elasticity with stress constraint

Transient sensitivities

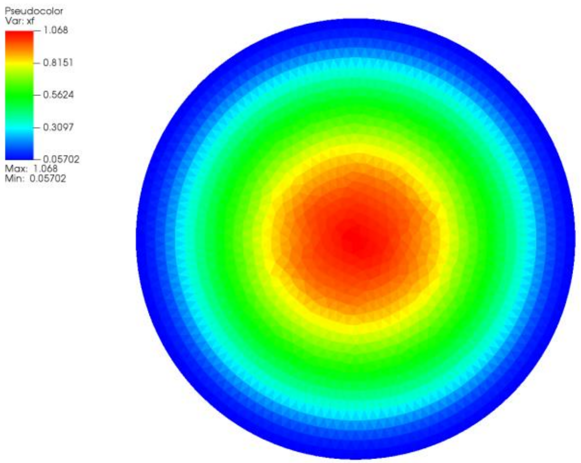
- We have analytical solution $T^*(\boldsymbol{x}, t)$, assuming $Q^*(\boldsymbol{x})$
- Solve inverse problem for $Q(\boldsymbol{x})$ that produces $T(x, t) = T^*(x, t)$

$$\min_{Q} \int_{T} \int_{\Omega} (T(\boldsymbol{x}, t) - T^*(\boldsymbol{x}, t))^2 + |\nabla T(\boldsymbol{x}, t) - \nabla T^*(\boldsymbol{x}, t)|^2 \, d\Omega \, dt$$

$T^*(\boldsymbol{x}, t)$

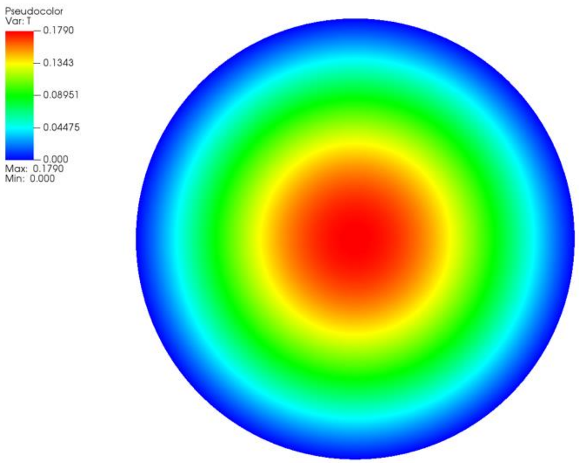


$Q(\boldsymbol{x})$

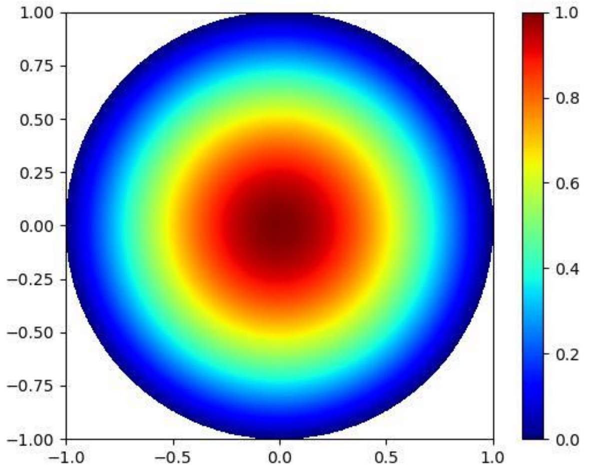


L2<0> design parameters

$T(\boldsymbol{x}, 1.0)$



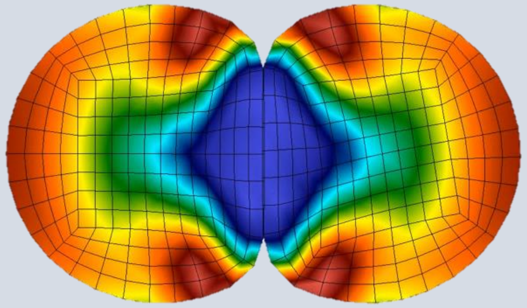
$Q^*(\boldsymbol{x})$



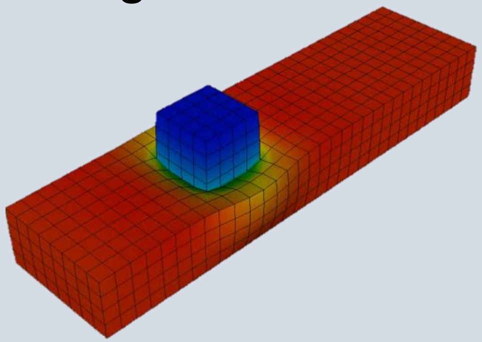
https://github.com/LLNL/Tribol

Serac interfaces with Tribol contact library.

Contact methodologies

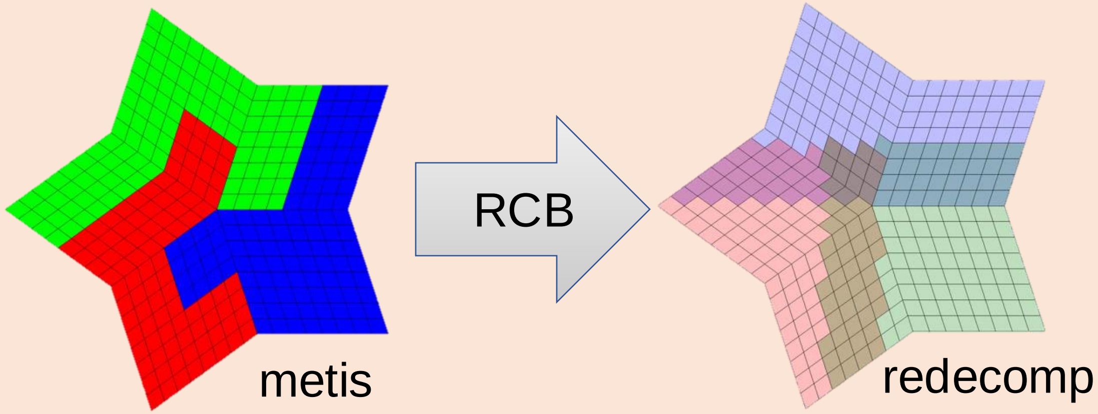


Common plane



Mortar

Domain redecomposer (redecomp)



MFEM interface

Submesh

- Create a surface ParMesh from boundary attributes
- Elements/dofs do not change rank

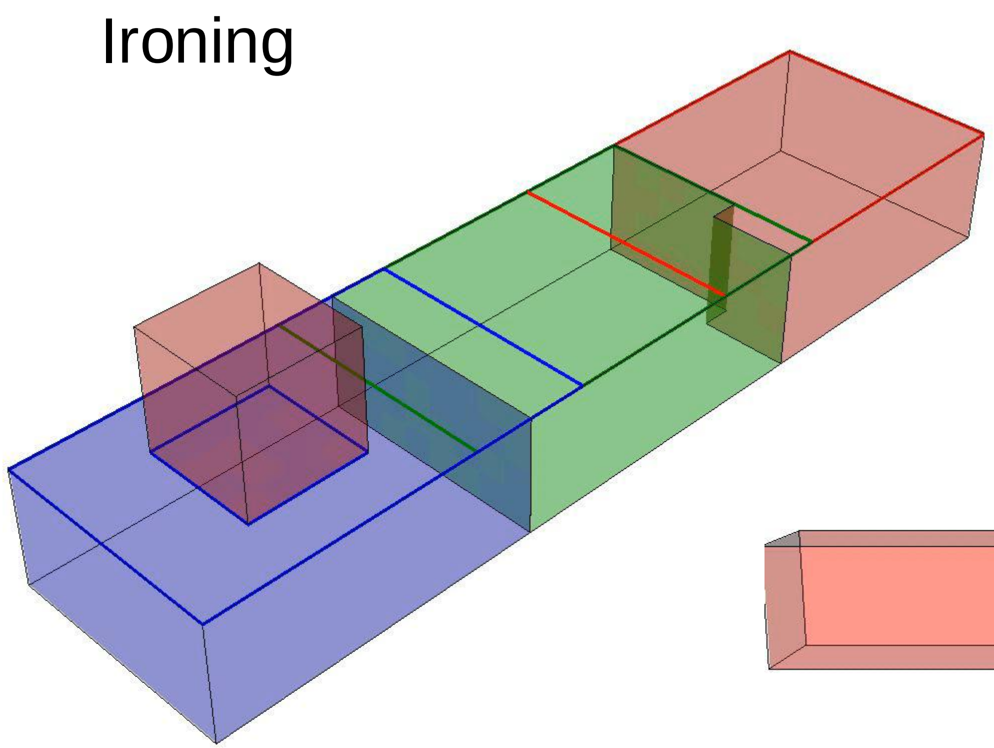
Low-order refined transfers

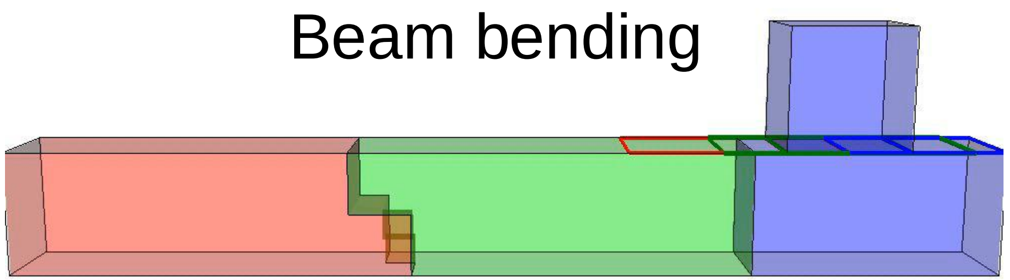
- Transfer higher-order mesh data to a refined low-order mesh
- Mass-conservative mapping

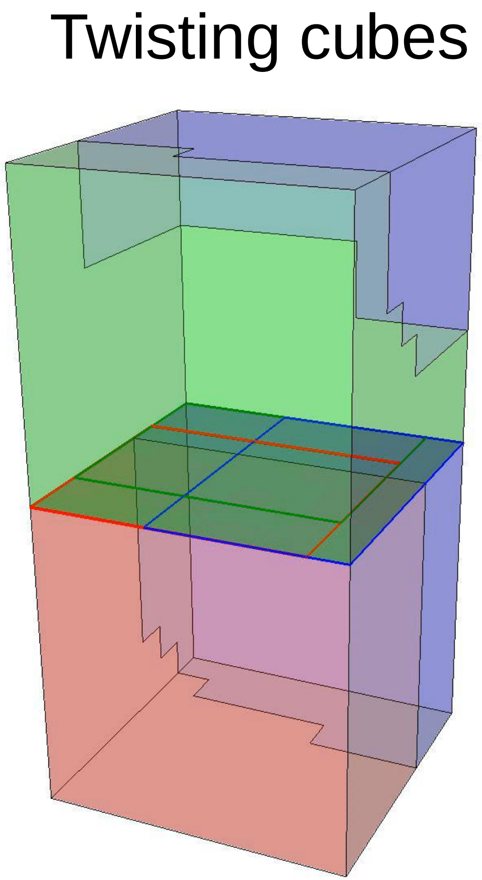
More example problems

Surface redecomposition on-rank domains

MFEM on-rank domains

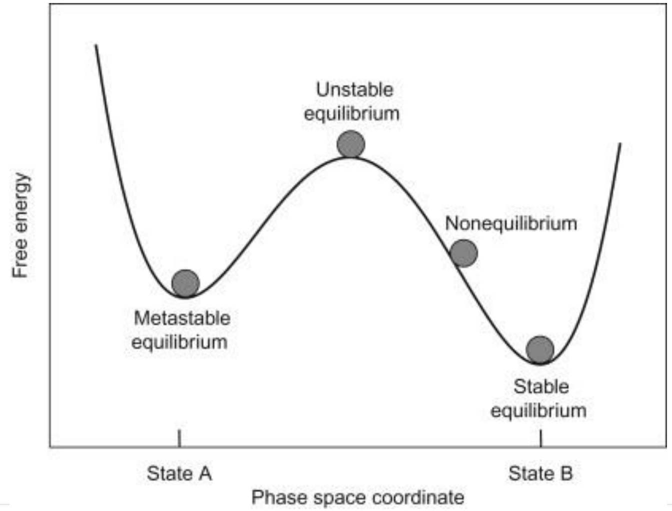


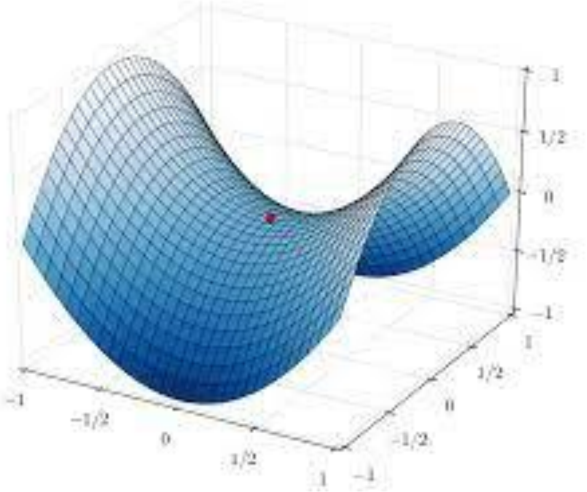


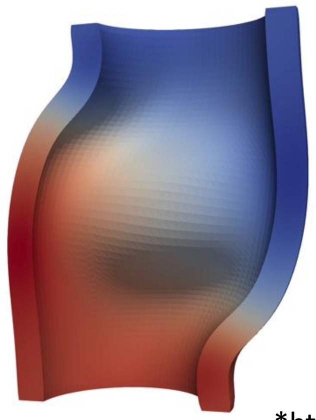


‘Optimization-based’ nonlinear solver via MFEM interface

- Supports a variety of preconditioners through HYPRE and PETSc
- Solves even when energy is unknown or incomputable
- Handles common system asymmetries
- Solves for stable equilibrium $\mathbf{g}(\mathbf{u}) = 0 \quad \mathbf{K} \succeq 0$ (stiffness eigenvalues are non-negative)







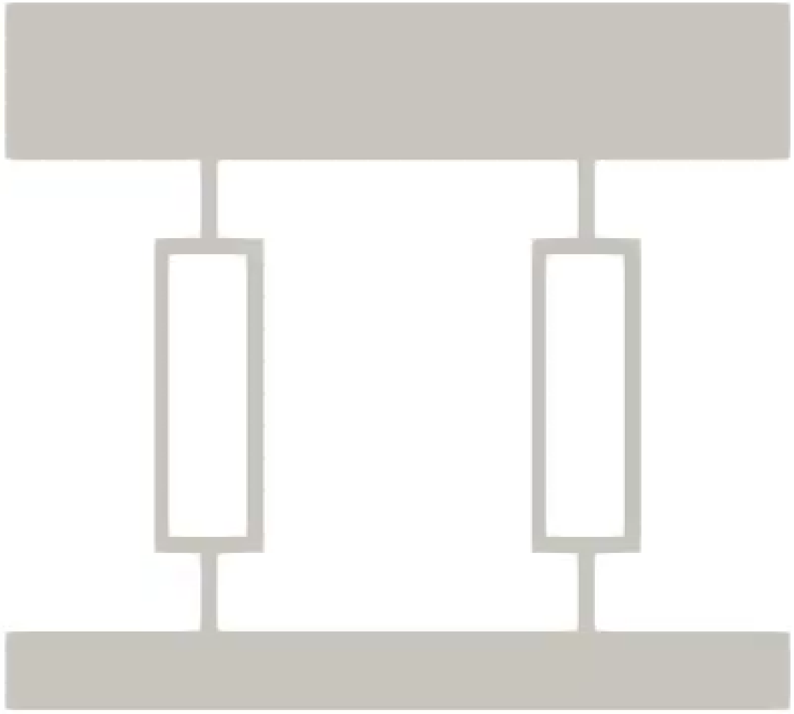
*https://github.com/llnl/serac

Example: Euler-beam buckling

Fastest solver found:

Nonlinear: Trust-region
Preconditioner: LU

Example: Mechanical logic gate



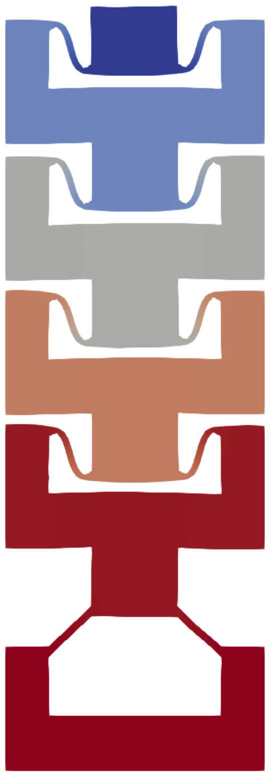
Fastest solver found:

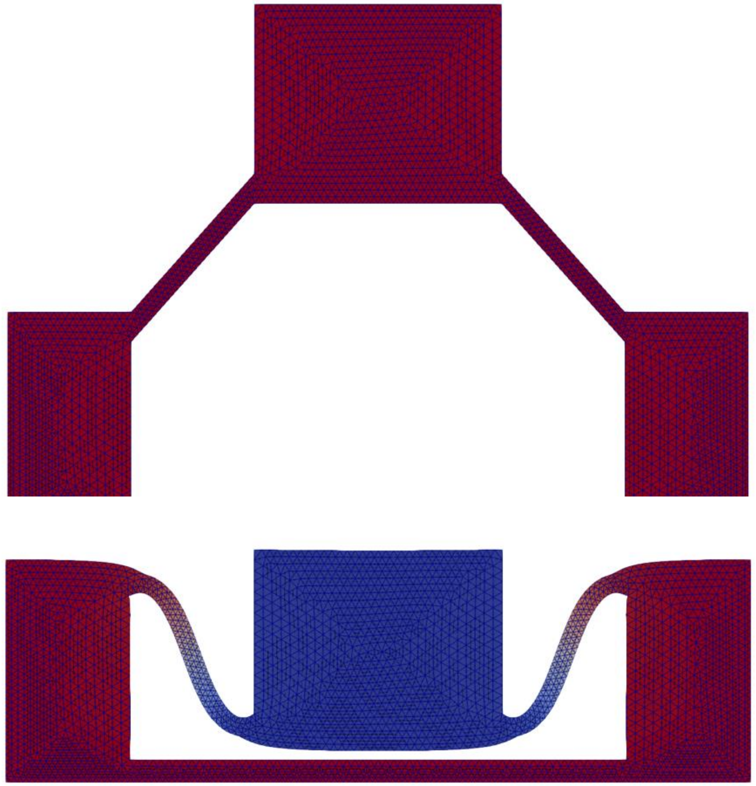
Nonlinear: Trust-region
Preconditioner: LU

Geometry and setup courtesy of Hilary Johnson and Katie Riley, LLNL

Example: energy dissipating buckling structures

- Snap-through (and sometimes snap-back) cellular structures



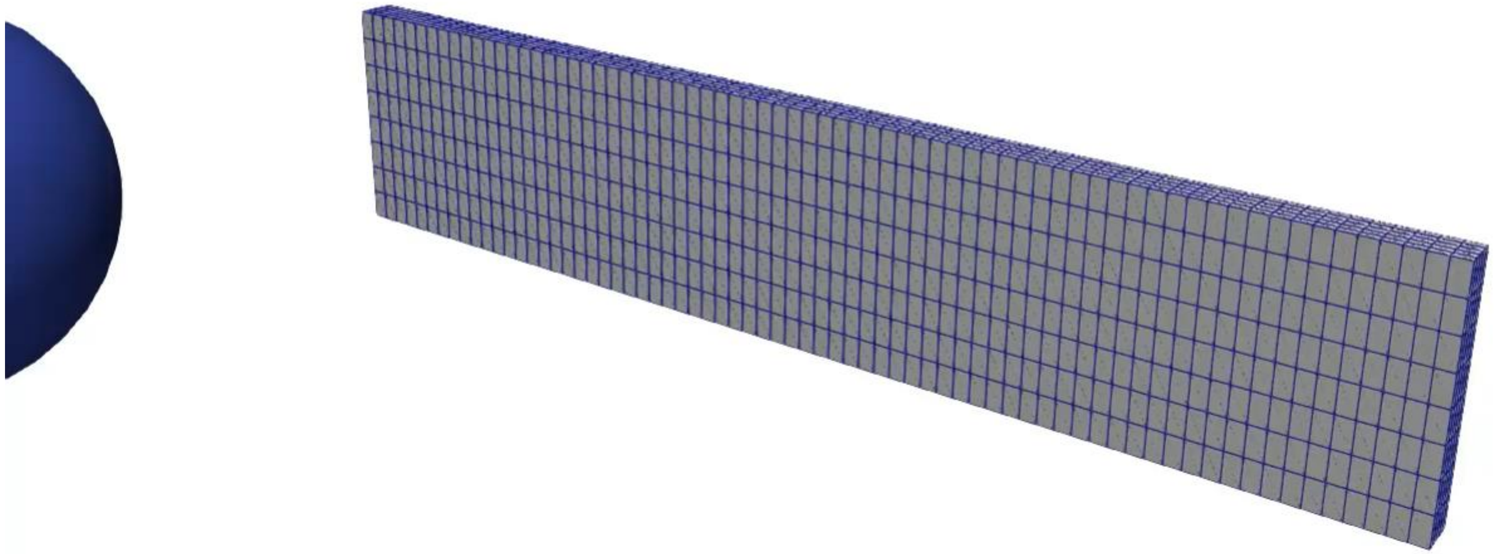


Fastest solver found:

Nonlinear: Trust-region
Preconditioner: LU

Geometry and setup courtesy of Ryan Alberdi, SNL

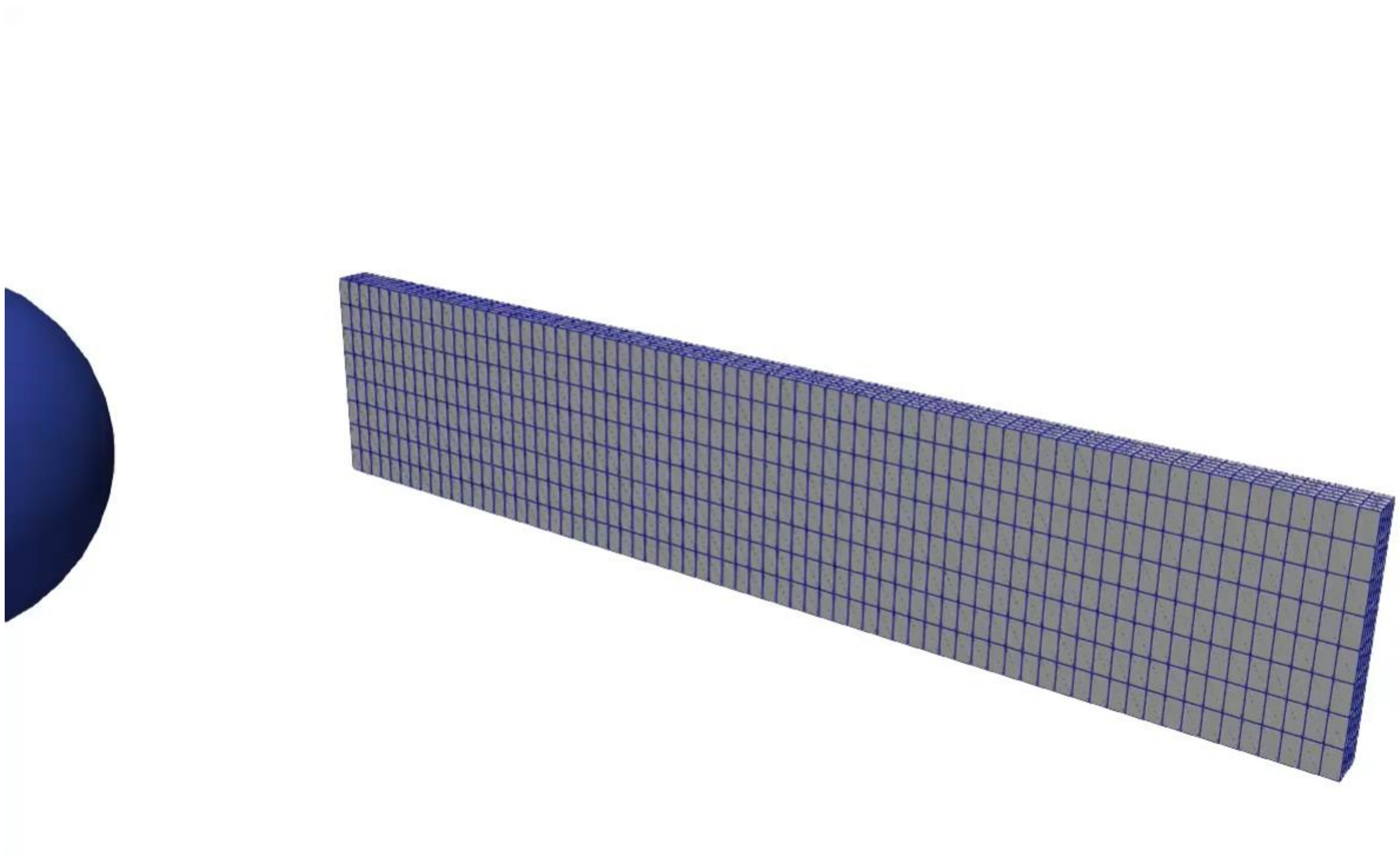
Contacting sphere



Fastest solver found:

Nonlinear: Trust-region
Preconditioner: Multigrid

Contacting sphere with friction

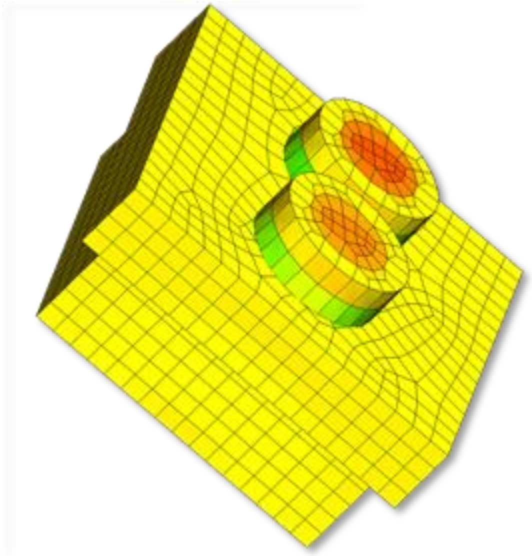


Fastest solver found:

Nonlinear: Trust-region
Preconditioner: Jacobi

Future directions

- Differentiable contact
- Enzyme for AD
- Matrix-free solvers
- Manufacturing simulation
- Dynamic checkpointing
- Adaptive schemes
- Differentiable integrated codes
- Gradient-based UQ
- Integration of ML techniques



THANK YOU

DISCLAIMER

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.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC