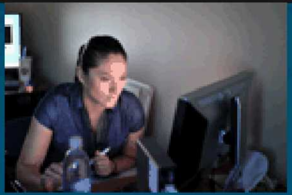
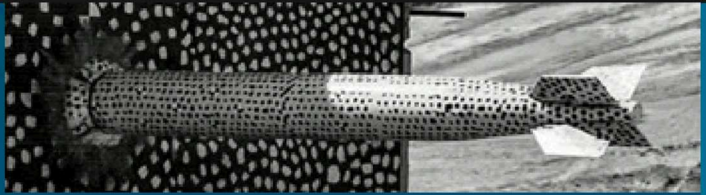
SAND2019-2989C

Optimization-based Design for Manufacturing

CONTRIBUTORS

Miguel Aguilo, Brett Clark, Joshua Robbins

PRESENTED BY

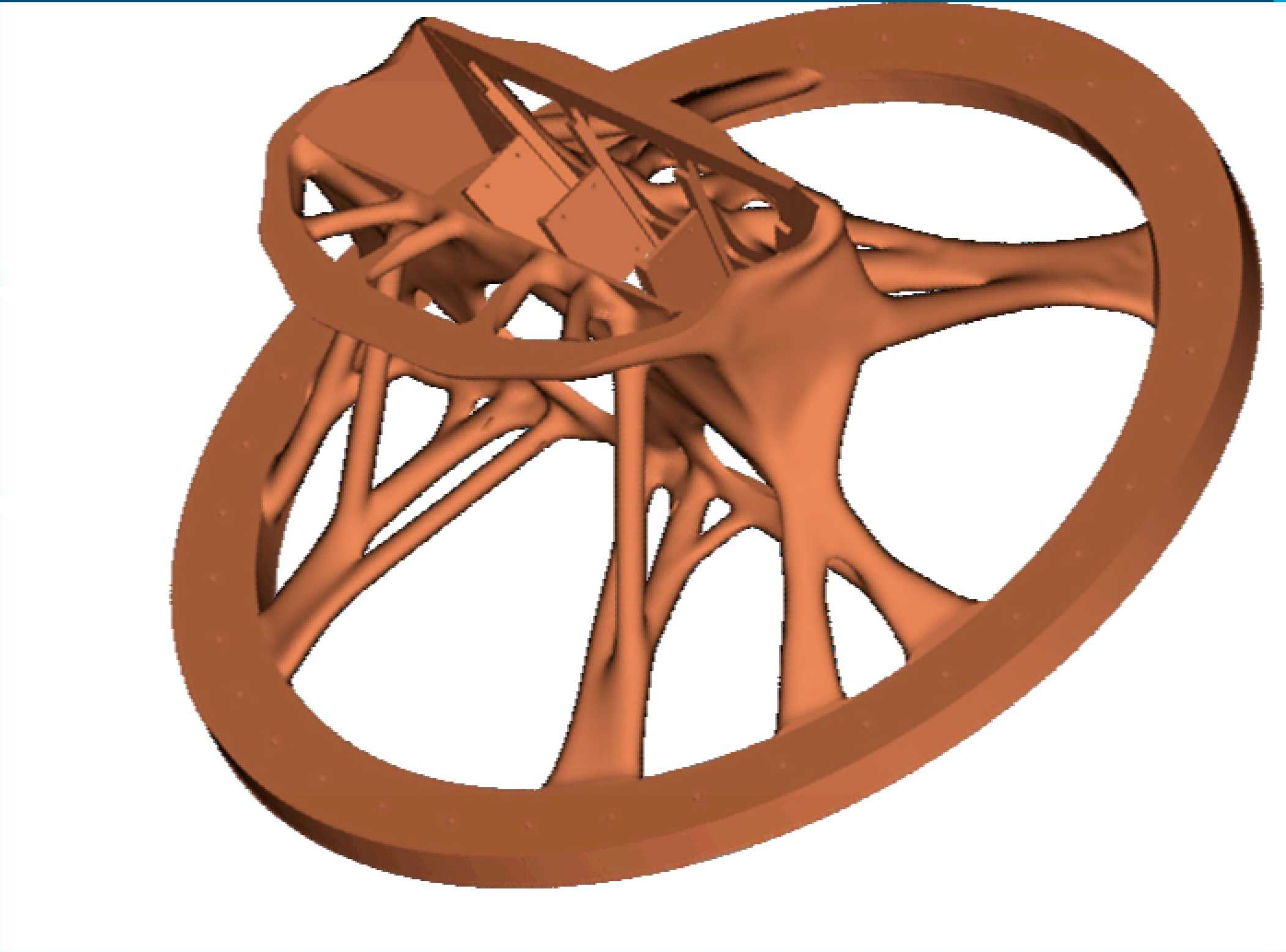
Joshua Robbins

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Motivation:

- Compute manufacturable designs*
  - Low scrap rates (high success rates
  - Cheaper print processes (?)

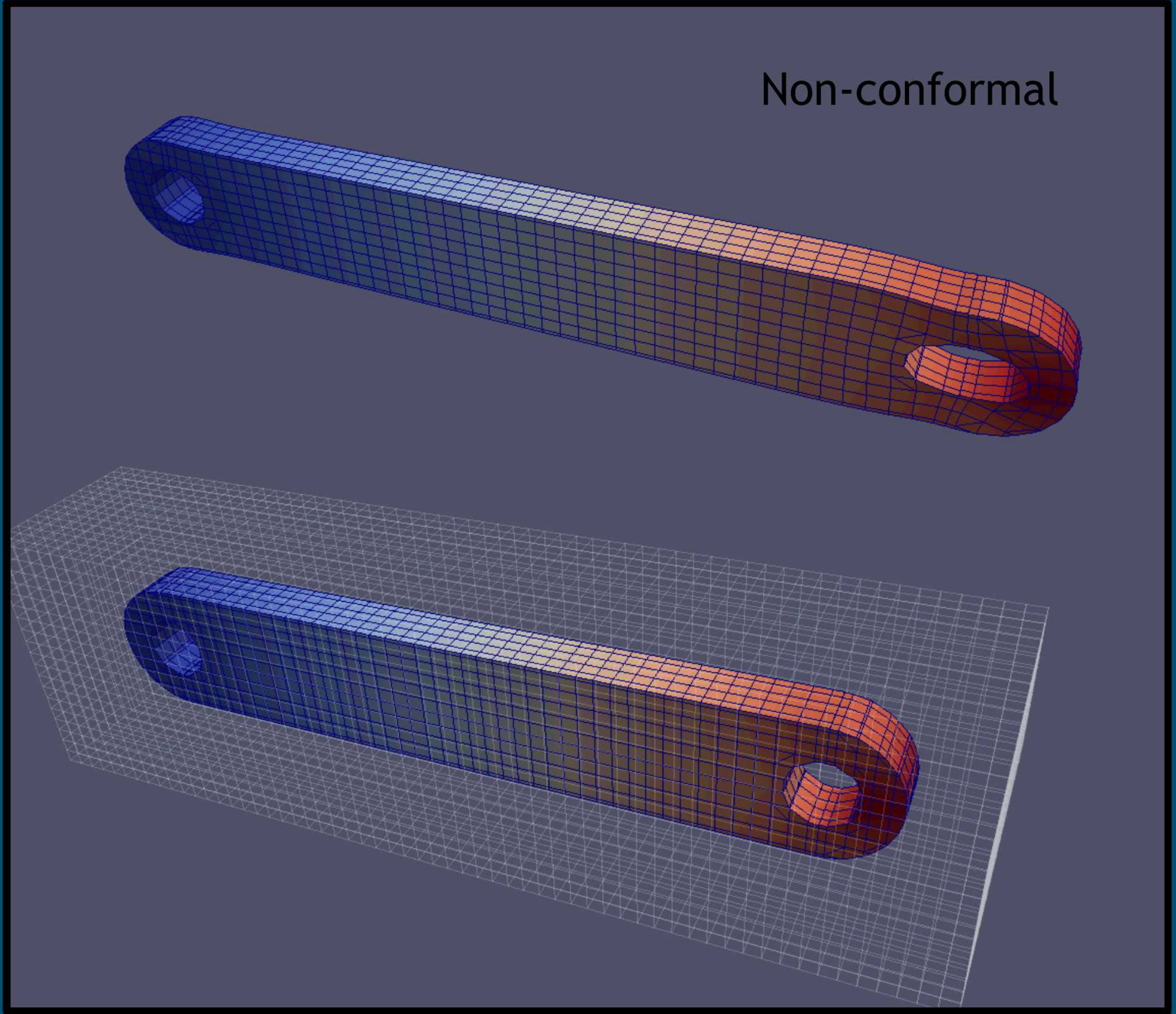
- Apply accurate stress constraints (Acc
  for residual stresses)

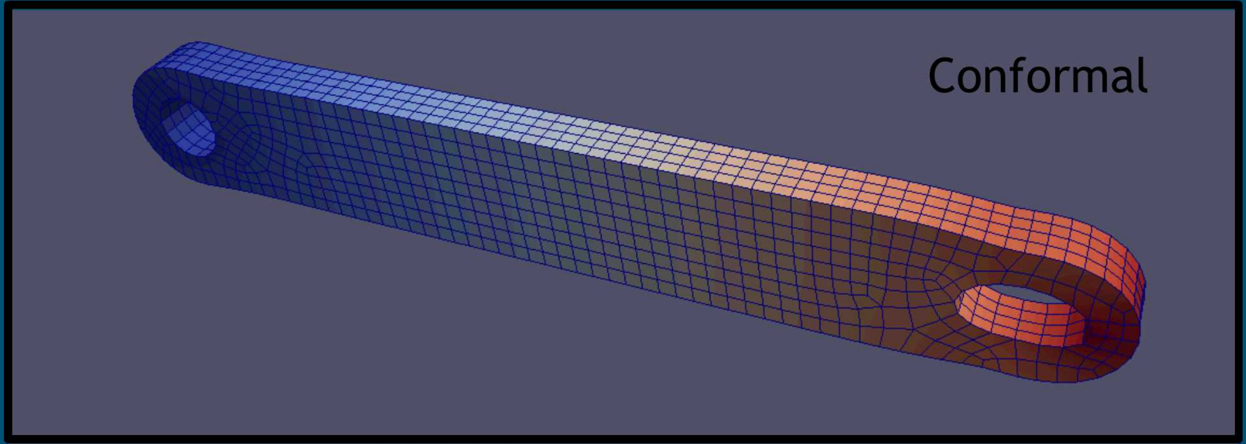


* B. Jared et al., Additive Manufacturing: Toward holistic design, Scripta Materialia. 135 (2017)

Optimization-based Design for Manufacturing Project:

- ‘Meshless’ design – concurrent SO/TO
- Process-aware design

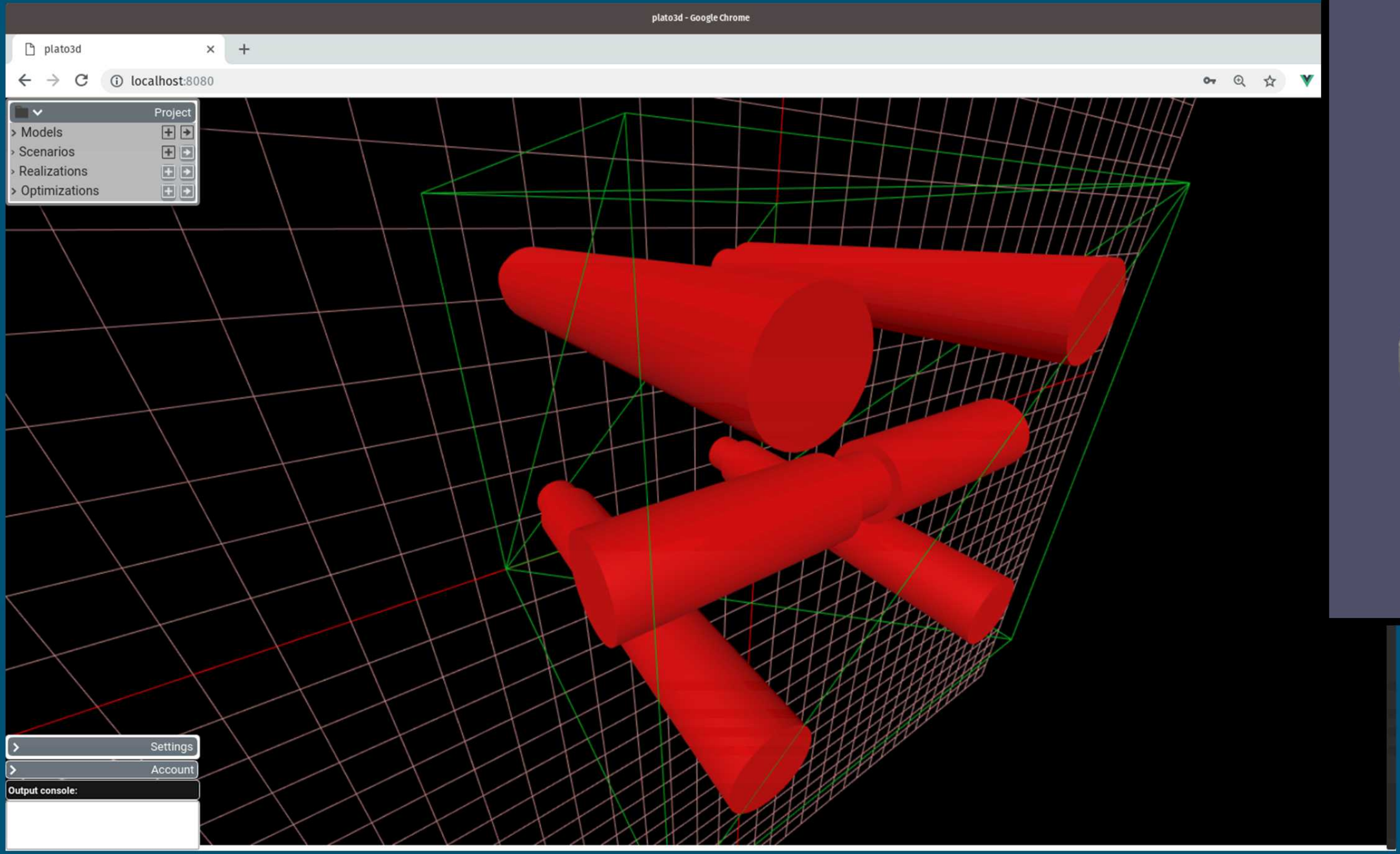


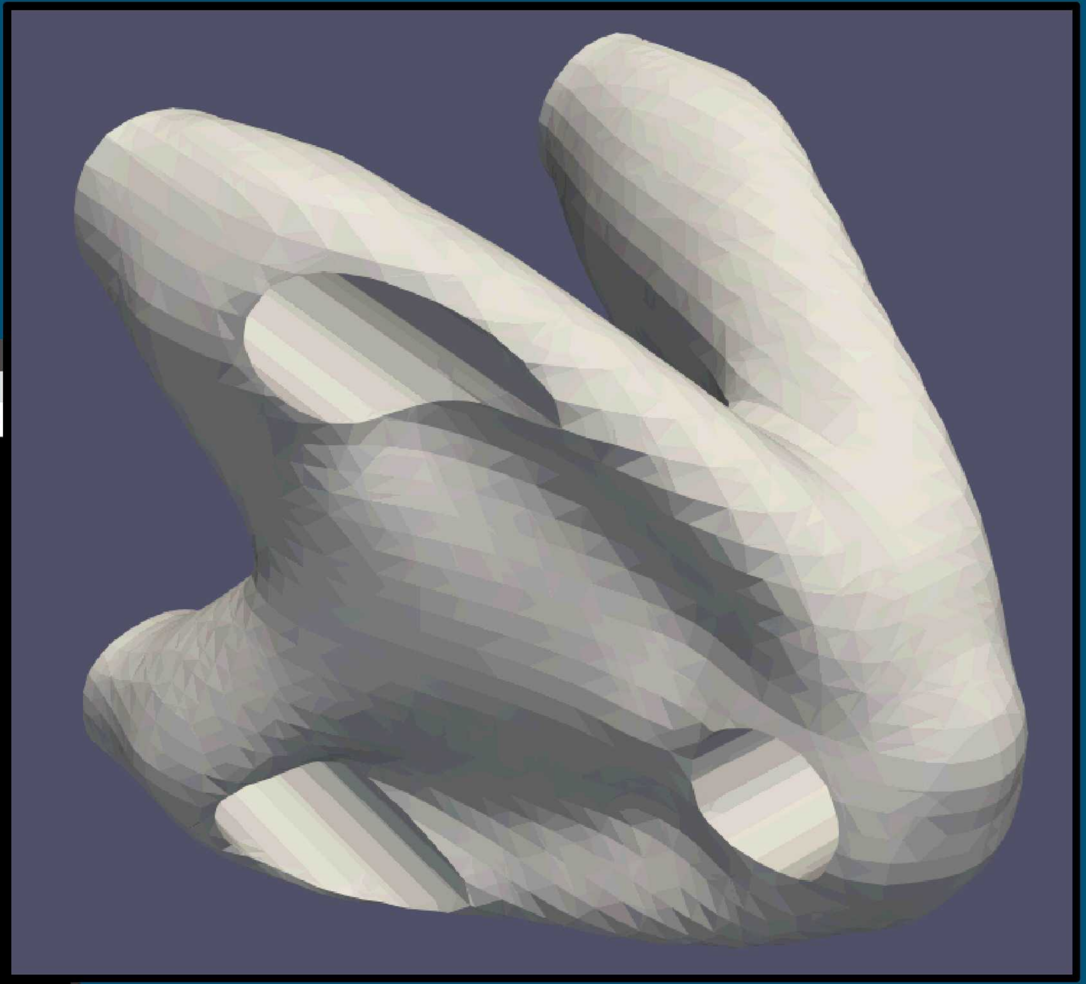


‘Meshless’ Design

Capture geometry on a background mesh non-conformally:

- Simplifies geometry construction and meshing
- Permits movement of fixed geometry within a design domain
- Permits optimization of assemblies





Process-aware Design

Objective: $$\min_{\boldsymbol{\varphi},\boldsymbol{X}} \sum_{\boldsymbol{i}} \alpha_i f_i(\boldsymbol{S}_i, \boldsymbol{\varphi}, \boldsymbol{X})$$

PDE Constraint: $$\boldsymbol{g}_i(\boldsymbol{S}_i, \boldsymbol{\varphi}, \boldsymbol{X}) = \boldsymbol{0}$$

Inequality Constraint: $$h(\boldsymbol{\varphi}, \boldsymbol{X}) \leq 0$$

Optimization of Process Objectives

$$\min_{\boldsymbol{\phi}} f\left(\boldsymbol{\phi}\right)$$

$$f\left(\boldsymbol{\phi}\right) = F\left(\boldsymbol{\phi}, \boldsymbol{T}^1(\boldsymbol{\phi}), \boldsymbol{T}^2(\boldsymbol{\phi}), ..., \boldsymbol{T}^n(\boldsymbol{\phi}), \boldsymbol{U}^1(\boldsymbol{\phi}), \boldsymbol{U}^2(\boldsymbol{\phi}), ..., \boldsymbol{U}^n(\boldsymbol{\phi}), \boldsymbol{c}^1(\boldsymbol{\phi}), \boldsymbol{c}^2(\boldsymbol{\phi}), ..., \boldsymbol{c}^n(\boldsymbol{\phi})\right)$$

$$\boldsymbol{Q}^k\left(\boldsymbol{\phi}, \boldsymbol{T}^k, \boldsymbol{T}^{k-1}\right) = 0$$: Heat Equation

$$\boldsymbol{R}^k\left(\boldsymbol{\phi}, \boldsymbol{T}^k, \boldsymbol{T}^{k-1}, \boldsymbol{U}^k, \boldsymbol{U}^{k-1}, \boldsymbol{c}^k, \boldsymbol{c}^{k-1}\right) = 0$$: Mechanical Equilibrium

$$\boldsymbol{H}^k\left(\boldsymbol{\phi}, \boldsymbol{T}^k, \boldsymbol{T}^{k-1}, \boldsymbol{U}^k, \boldsymbol{U}^{k-1}, \boldsymbol{c}^k, \boldsymbol{c}^{k-1}\right) = 0$$: Mechanical State Eqns

$$k = 1, 2, ..., n$$

Langrange Function

$$\hat{f} = f(\phi) + \sum_{k=1}^{n} \boldsymbol{\gamma}^{k^T} \boldsymbol{Q}^k + \sum_{k=1}^{n} \boldsymbol{\lambda}^{n^T} \boldsymbol{R}^k + \sum_{k=1}^{n} \boldsymbol{\mu}^{n^T} \boldsymbol{H}^k$$

Final Step

$$\frac{\partial F}{\partial \boldsymbol{U}^n} + \boldsymbol{\lambda}^{nT} \frac{\partial \boldsymbol{R}^n}{\partial \boldsymbol{U}^n} + \boldsymbol{\mu}^{nT} \frac{\partial \boldsymbol{H}^n}{\partial \boldsymbol{U}^n} = 0$$

$$\frac{\partial F}{\partial \boldsymbol{c}^n} + \boldsymbol{\lambda}^{nT} \frac{\partial \boldsymbol{R}^n}{\partial \boldsymbol{c}^n} + \boldsymbol{\mu}^{nT} \frac{\partial \boldsymbol{H}^n}{\partial \boldsymbol{c}^n} = 0$$

$$\frac{\partial F}{\partial \boldsymbol{T}^n} + \boldsymbol{\gamma}^{nT} \frac{\partial \boldsymbol{Q}^n}{\partial \boldsymbol{T}^n} + \boldsymbol{\lambda}^{nT} \frac{\partial \boldsymbol{R}^n}{\partial \boldsymbol{T}^n} + \boldsymbol{\mu}^{nT} \frac{\partial \boldsymbol{H}^n}{\partial \boldsymbol{T}^n} = 0$$

kth Step

$$\frac{\partial F}{\partial \boldsymbol{U}^k} + \boldsymbol{\lambda}^{k+1^T} \frac{\partial \boldsymbol{R}^{k+1}}{\partial \boldsymbol{U}^k} + \boldsymbol{\mu}^{k+1^T} \frac{\partial \boldsymbol{H}^{k+1}}{\partial \boldsymbol{U}^k} + \boldsymbol{\lambda}^{kT} \frac{\partial \boldsymbol{R}^k}{\partial \boldsymbol{U}^k} + \boldsymbol{\mu}^{nT} \frac{\partial \boldsymbol{H}^k}{\partial \boldsymbol{U}^k} = 0$$

$$\frac{\partial F}{\partial \boldsymbol{c}^k} + \boldsymbol{\lambda}^{k+1^T} \frac{\partial \boldsymbol{R}^{k+1}}{\partial \boldsymbol{c}^k} + \boldsymbol{\mu}^{k+1^T} \frac{\partial \boldsymbol{H}^{k+1}}{\partial \boldsymbol{c}^k} + \boldsymbol{\lambda}^{kT} \frac{\partial \boldsymbol{R}^k}{\partial \boldsymbol{c}^k} + \boldsymbol{\mu}^{nT} \frac{\partial \boldsymbol{H}^k}{\partial \boldsymbol{c}^k} = 0$$

$$\frac{\partial F}{\partial \boldsymbol{T}^k} + \boldsymbol{\gamma}^{k+1^T} \frac{\partial \boldsymbol{Q}^{k+1}}{\partial \boldsymbol{T}^k} + \boldsymbol{\lambda}^{k+1^T} \frac{\partial \boldsymbol{R}^{k+1}}{\partial \boldsymbol{T}^k} + \boldsymbol{\mu}^{k+1^T} \frac{\partial \boldsymbol{H}^{k+1}}{\partial \boldsymbol{T}^k}$$

$$+ \boldsymbol{\gamma}^{kT} \frac{\partial \boldsymbol{Q}^k}{\partial \boldsymbol{T}^k} + \boldsymbol{\lambda}^{kT} \frac{\partial \boldsymbol{R}^k}{\partial \boldsymbol{T}^k} + \boldsymbol{\mu}^{nT} \frac{\partial \boldsymbol{H}^k}{\partial \boldsymbol{T}^k} = 0$$

$k = n-1, ..., 1, 2$

Total Derivative

$$\frac{d\hat{f}}{d\phi} = \frac{\partial F}{\partial \phi} + \sum_{k=1}^{n} \left(\boldsymbol{\gamma}^{kT} \frac{\partial \boldsymbol{Q}^k}{\partial \phi} + \boldsymbol{\lambda}^{kT} \frac{\partial \boldsymbol{R}^k}{\partial \phi} + \boldsymbol{\mu}^{kT} \frac{\partial \boldsymbol{H}^k}{\partial \phi} \right)$$

To solve, we need:

$$\frac{\partial F}{\partial \boldsymbol{\varphi}}, \frac{\partial F}{\partial \boldsymbol{X}}, \frac{\partial F}{\partial \boldsymbol{U}^k}, \frac{\partial F}{\partial \boldsymbol{T}^k}, \frac{\partial F}{\partial \boldsymbol{c}^k},$$

$$\frac{\partial \boldsymbol{Q}^k}{\partial \boldsymbol{\varphi}}, \frac{\partial \boldsymbol{Q}^k}{\partial \boldsymbol{X}}, \frac{\partial \boldsymbol{Q}^k}{\partial \boldsymbol{T}^k}, \frac{\partial \boldsymbol{Q}^k}{\partial \boldsymbol{T}^{k-1}},$$

$$\frac{\partial \boldsymbol{R}^k}{\partial \boldsymbol{\varphi}}, \frac{\partial \boldsymbol{R}^k}{\partial \boldsymbol{X}}, \frac{\partial \boldsymbol{R}^k}{\partial \boldsymbol{T}^k}, \frac{\partial \boldsymbol{R}^k}{\partial \boldsymbol{T}^{k-1}}, \frac{\partial \boldsymbol{R}^k}{\partial \boldsymbol{U}^k}, \frac{\partial \boldsymbol{R}^k}{\partial \boldsymbol{U}^{k-1}}, \frac{\partial \boldsymbol{R}^k}{\partial \boldsymbol{c}^k}, \frac{\partial \boldsymbol{R}^k}{\partial \boldsymbol{c}^{k-1}},$$

$$\frac{\partial \boldsymbol{H}^k}{\partial \boldsymbol{\varphi}}, \frac{\partial \boldsymbol{H}^k}{\partial \boldsymbol{X}}, \frac{\partial \boldsymbol{H}^k}{\partial \boldsymbol{T}^k}, \frac{\partial \boldsymbol{H}^k}{\partial \boldsymbol{T}^{k-1}}, \frac{\partial \boldsymbol{H}^k}{\partial \boldsymbol{U}^k}, \frac{\partial \boldsymbol{H}^k}{\partial \boldsymbol{U}^{k-1}}, \frac{\partial \boldsymbol{H}^k}{\partial \boldsymbol{c}^k}, \frac{\partial \boldsymbol{H}^k}{\partial \boldsymbol{c}^{k-1}}$$

Approach

Plato Analyze is built on Sacado for automatic differentiation.
AD is straightforward to use.

Templatize

```
template <typename ScalarT>
ScalarT func(const ScalarT& a, const ScalarT& b, const ScalarT& c) {
  return c*std::log(b+1.)/std::sin(a);
}
```

Create AD types, and call template function

```
int main(){
  typedef Sacado::Fad::Sfad<double,2> SFadType;
  SFadType a(2, 0, 1.0), b(2, 1, 2.0), c(3.0);
  auto r = func(a,b,c);
  cout << "Value: " << r.val() << endl;
  cout << "dr/da: " << r.dx(0) << endl;
  cout << "dr/db: " << r.dx(1) << endl;
}
```

Sacado is available at github.com/trilinos/trilinos

VectorFunctionInc: $Q^k(T^k, T^{k-1}, \varphi, X)$

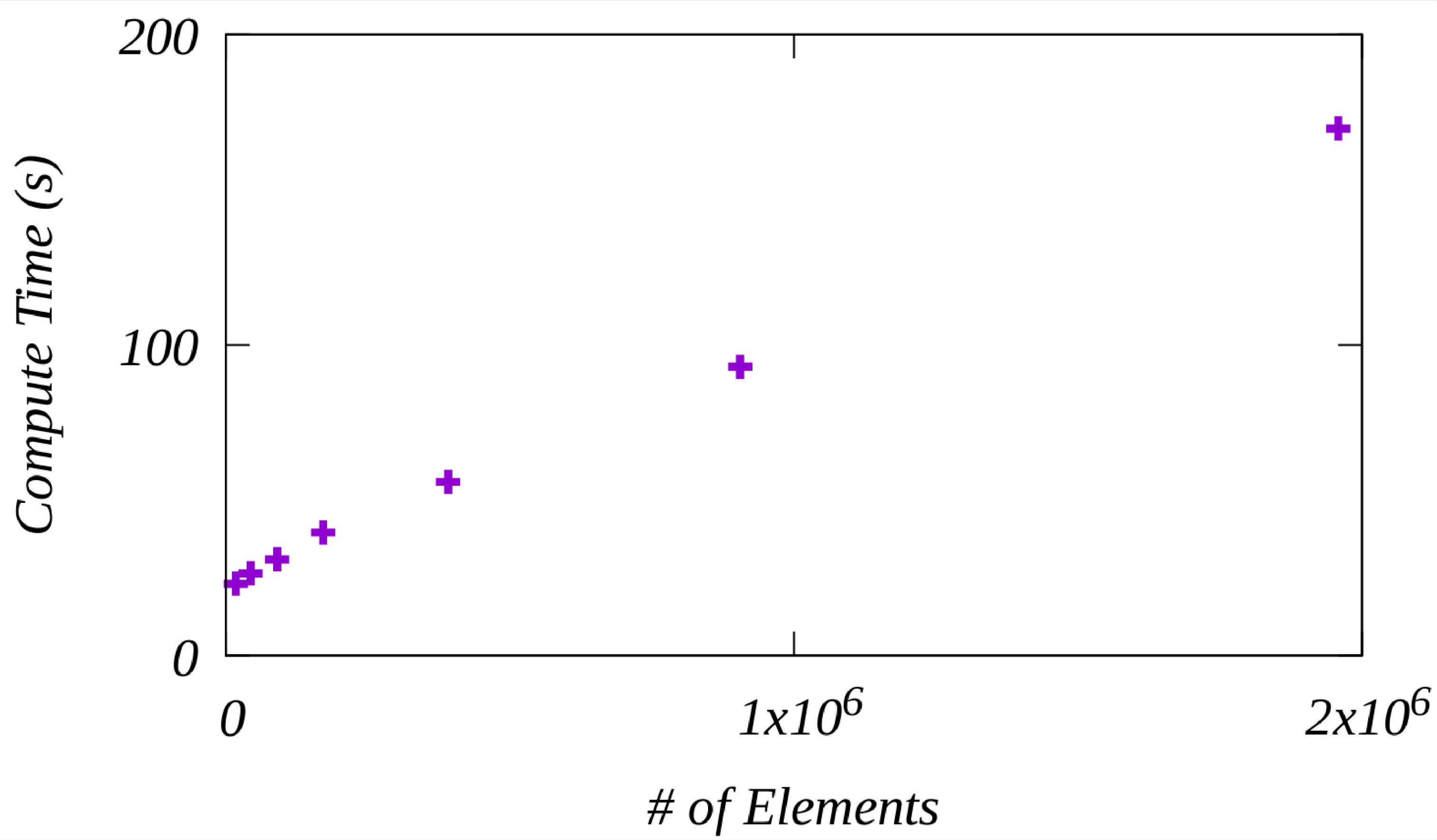
```
virtual void
evaluate( Alexa::ScalarMultiVectorT<typename EvaluationType::StateScalarType    > state,
          Alexa::ScalarMultiVectorT<typename EvaluationType::PrevStateScalarType> prevState,
          Alexa::ScalarMultiVectorT<typename EvaluationType::ControlScalarType  > control,
          Alexa::ScalarArray3DT    <typename EvaluationType::ConfigScalarType   > config,
          Alexa::ScalarMultiVectorT<typename EvaluationType::ResultScalarType   > result);
```

Heat Equation:

```
Kokkos::parallel_for(Kokkos::RangePolicy<int>(0,numCells), LAMBDA_EXPRESSION(int cellOrdinal)
     {
       computeGradient(cellOrdinal, gradient, config, cellVolume);
       cellVolume(cellOrdinal) *= quadratureWeight;

       scalarGrad(cellOrdinal, tGrad,     state,     gradient);
       scalarGrad(cellOrdinal, tPrevGrad, prevState, gradient);

       thermalFlux(cellOrdinal, tFlux,     tGrad);
       thermalFlux(cellOrdinal, tPrevFlux, tPrevGrad);

       applyFluxWeighting(cellOrdinal, tFlux,     control);
       applyFluxWeighting(cellOrdinal, tPrevFlux, control);

       fluxDivergence(cellOrdinal, result, tFlux,     gradient, cellVolume, timestep/2.0);
       fluxDivergence(cellOrdinal, result, tPrevFlux, gradient, cellVolume, timestep/2.0);

       ...

     },"Heat Equation Residual");
```

https://github.com/SNLComputation/lgrtk/blob/master/src/plato/HeatEquationResidual.hpp

Objective Gradient (100 time steps)



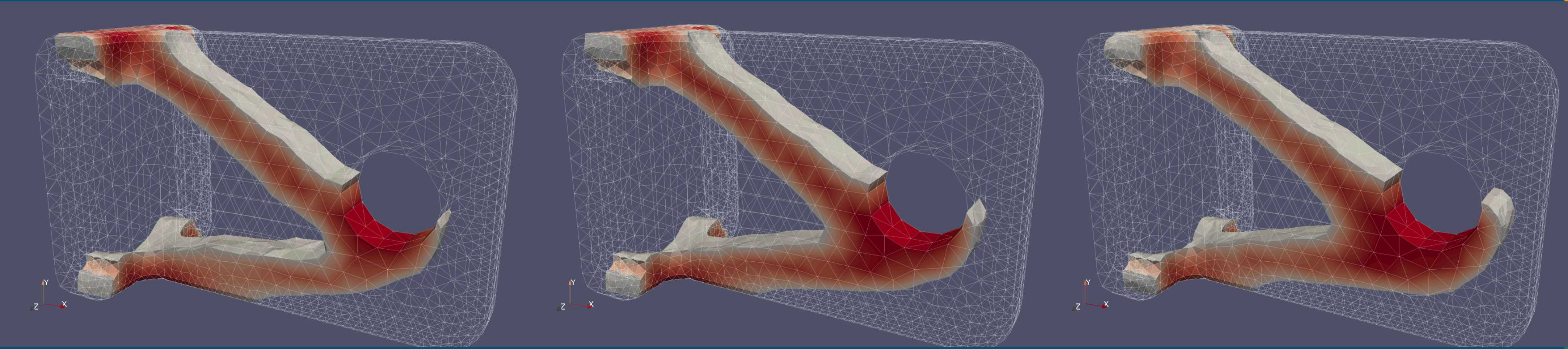
Example:

- Objectives: Mechanical stiffness, thermal stiffness over time
- Physics: Mechanical equilibrium, transient heat equation

Thermal: 0.05

Thermal: 0.25

Thermal: 0.50



Define coarse bases

$$\boldsymbol{\mathcal{N}} = \{\mathcal{N}_i \mid \nabla \cdot \nabla \mathcal{N}_i = 0 \;\forall\; \boldsymbol{x} \in \Omega_{i-1} \cup \Omega_i, \\ \mathcal{N}_i(\boldsymbol{x}) = 0 \;\forall\; \boldsymbol{x} \in \partial\Omega_{i+1}, \\ \mathcal{N}_i(\boldsymbol{x}) = 1 \;\forall\; \boldsymbol{x} \in \partial\Omega_i, \\ \mathcal{N}_i(\boldsymbol{x}) = 0 \;\forall\; \boldsymbol{x} \in \partial\Omega_{i-1},\; i \in I_{sn} \}$$

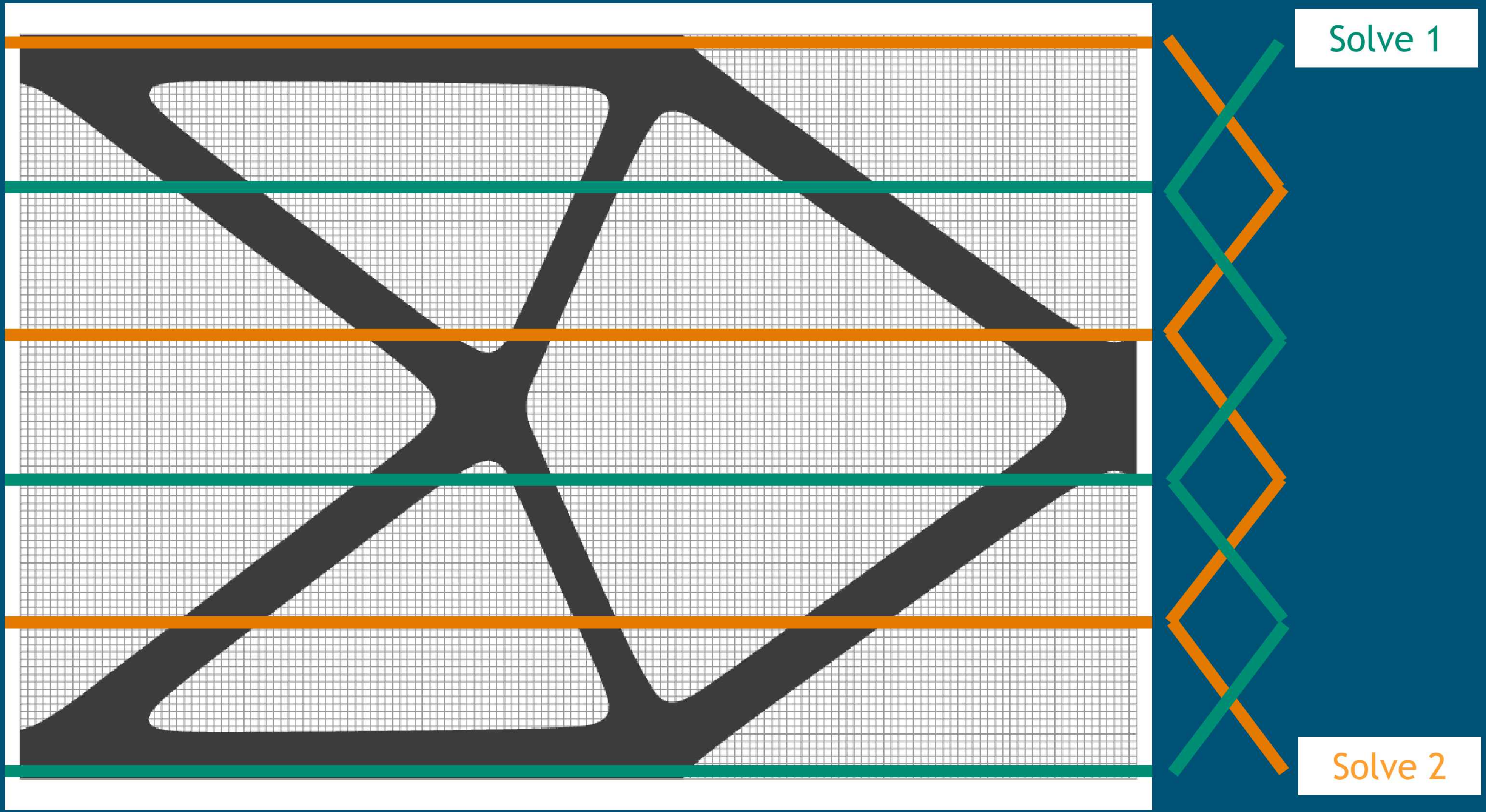
Partition of Unity

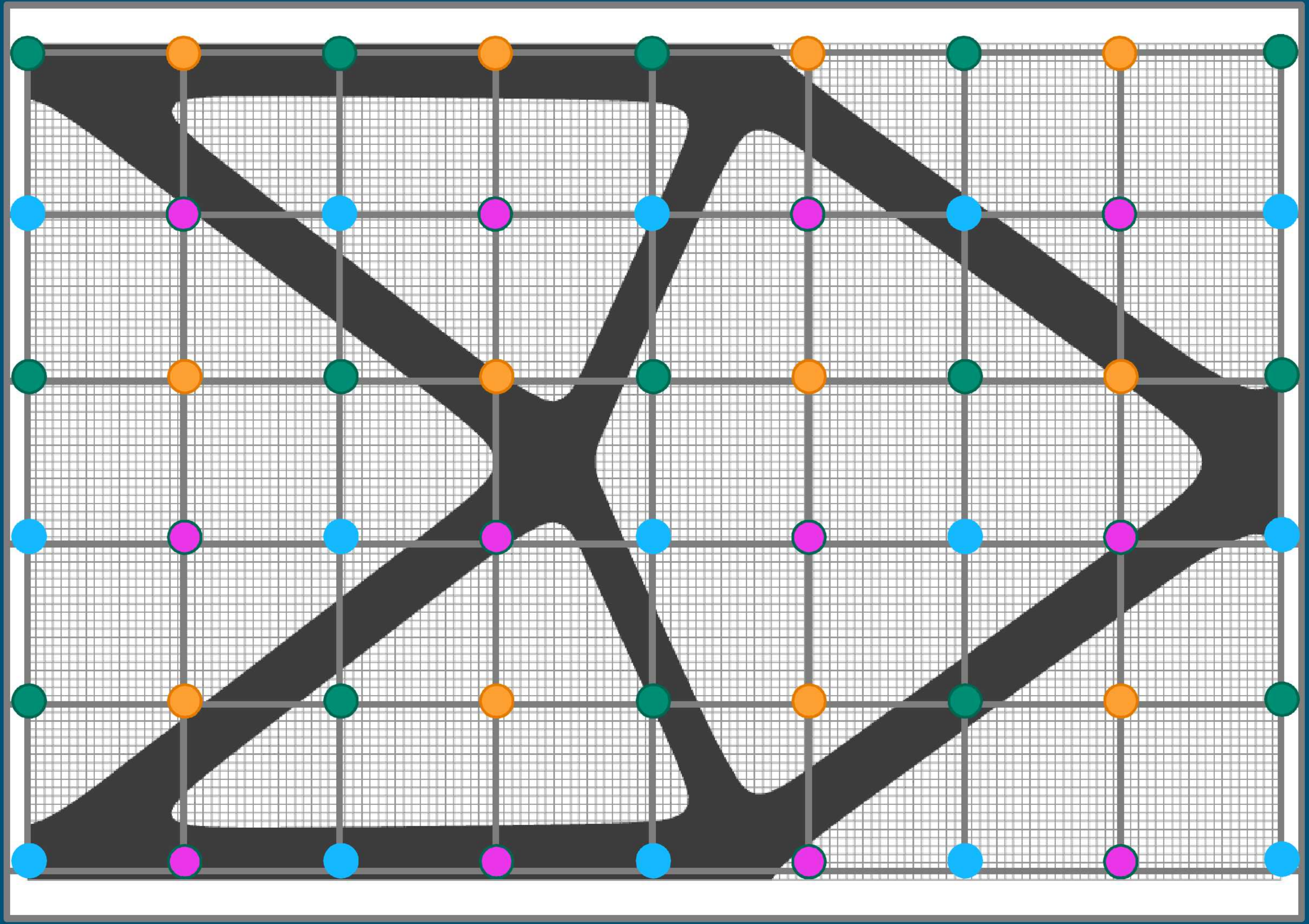
$$\sum_{\alpha \in I_{sn}} \mathcal{N}_\alpha(\boldsymbol{x}) = 1 \;\forall\; \boldsymbol{x} \in \Omega$$

$$T(\boldsymbol{x}, t) = \sum_{\alpha \in I_{sn}} \mathcal{N}_\alpha(\boldsymbol{x}) T_\alpha(t)$$

Coarse Model

$$\boldsymbol{\mathcal{M}} \dot{\boldsymbol{T}}(t) + \boldsymbol{\mathcal{K}} \boldsymbol{T}(t) + \boldsymbol{\mathcal{F}} - \boldsymbol{\mathcal{Q}} = 0$$


Solve 1
Solve 2



Solve 1

Solve 2

Solve 3

Solve 4

Progress toward process-aware optimization:

Coupled thermo-mechanics in Plato Analyze (residual function, objective functions, optimization problem added to regression tests, integration tests)

Thermal

$$\nabla \cdot \boldsymbol{q} - s = 0$$

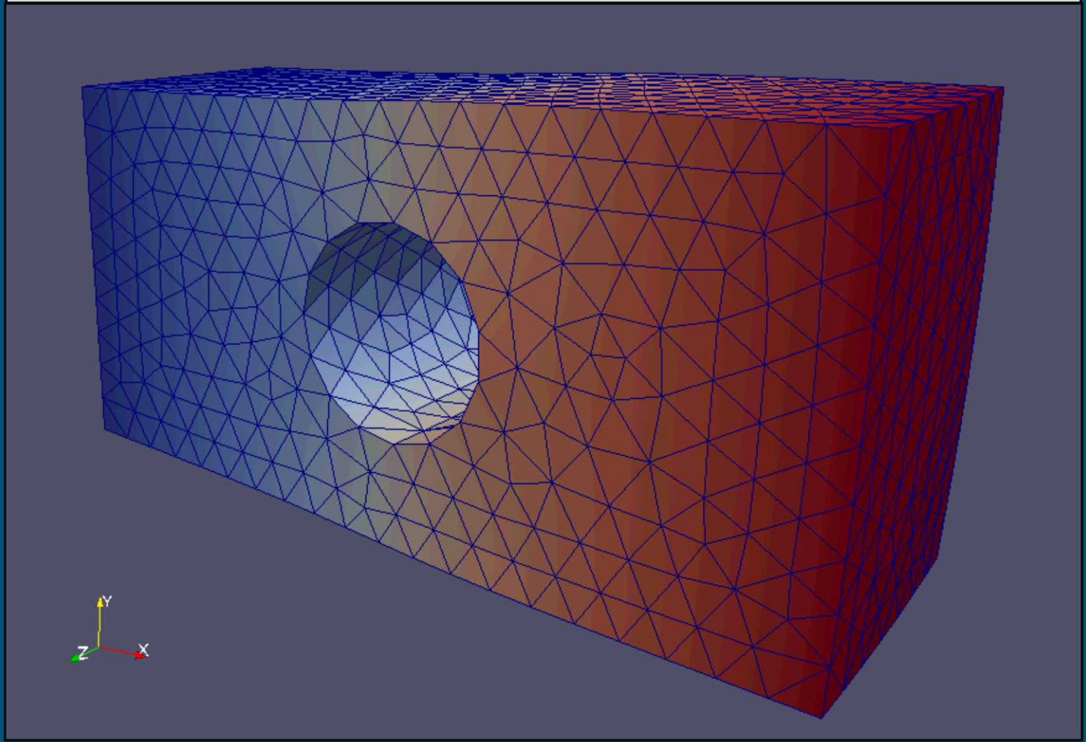
Mechanical

$$\nabla \cdot \boldsymbol{\sigma} + \boldsymbol{b} = 0$$

Coupling

$$\begin{Bmatrix} \boldsymbol{\sigma} \\ \boldsymbol{q} \end{Bmatrix} = f(\boldsymbol{\varepsilon}, \varphi, \nabla\varphi)$$

Example: thermal flux load



Progress toward process-aware optimization:

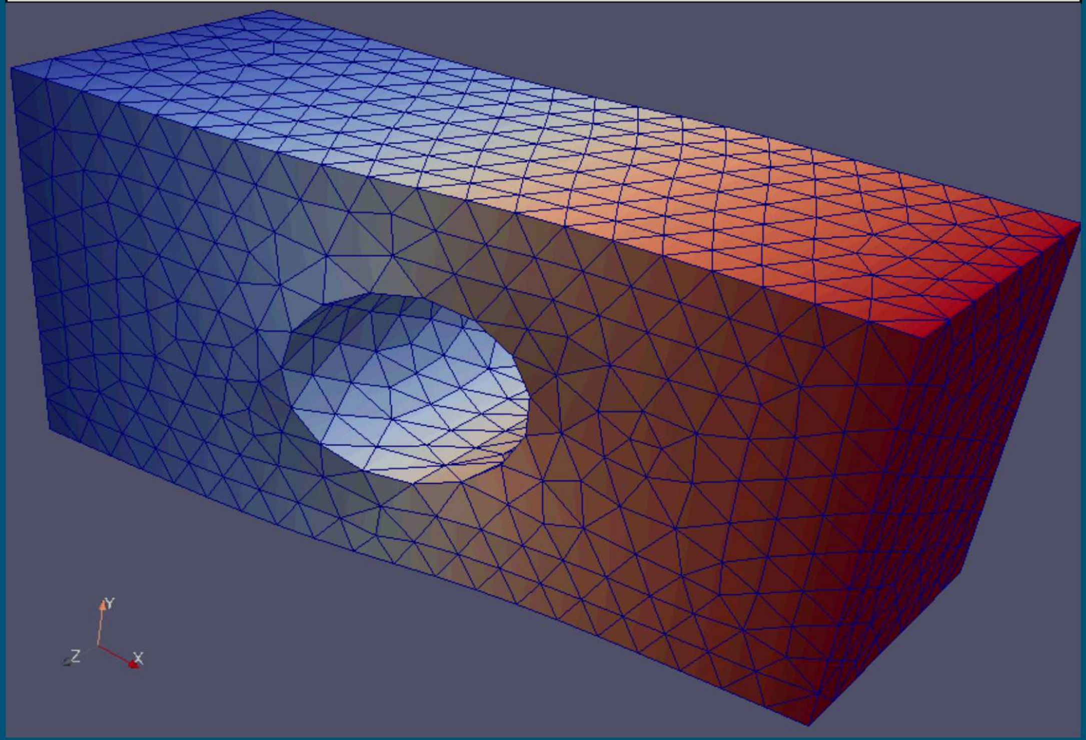
Coupled electro-mechanics in Plato Analyze (residual function, objective functions, optimization problem added to regression tests, integration tests)

Electrical
$\nabla \cdot \boldsymbol{D} - q = 0$

Mechanical
$\nabla \cdot \boldsymbol{\sigma} + \boldsymbol{b} = 0$

Coupling
$\begin{Bmatrix} \boldsymbol{\sigma} \\ \boldsymbol{D} \end{Bmatrix} = f(\boldsymbol{\varepsilon}, \nabla\varphi)$

Example: electric charge load



Progress

Physics:	Elastostatics, Poisson (thermal, electrostatic), coupled thermo-mechanics, coupled electro-mechanics, heat equation
Objectives:	Compliance, p-norm
Constraints:	Volume/mass

More physics, objectives, constraints coming soon.