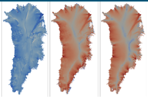
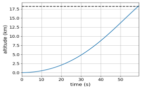
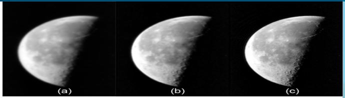
Sandia National Laboratories

Get ROL-ing

RAPID OPTIMIZATION LIBRARY







An Introduction to Sandia's Rapid Optimization Library

Drew Kouri   Denis Ridzal   Greg Von Winckel   **Aurya Javeed**

July 26, 2022

U.S. DEPARTMENT OF ENERGY   NNSA

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and 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.
SAND NO. 2014-00000

A Motivating Example

Rocket Dynamics

From the conservation of momentum,

$$\frac{dp}{dt} \approx \frac{\{(m-|\Delta m|)(u+\Delta u) + |\Delta m|(u-k)\} - mu}{\Delta t}$$
$$= \sum F = -mg$$
$$\implies \quad -m\frac{du}{dt} = k\frac{dm}{dt} + mg. \tag{1}$$

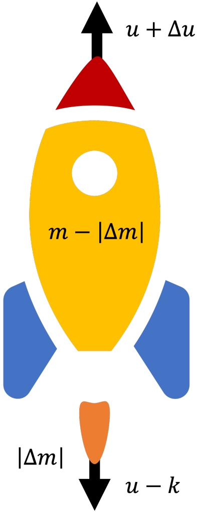
Here, we take g and the exhaust speed k to be constants but

$$\frac{dm}{dt} = -z < 0, \tag{2}$$

where $z = z(t)$ is a control of our choosing.

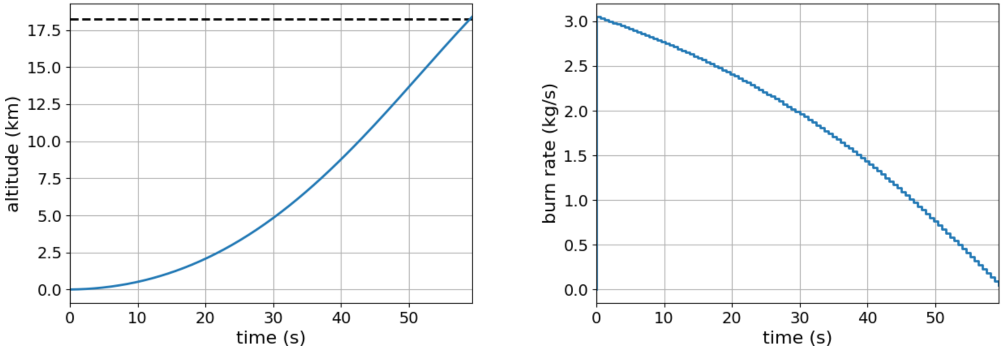
We want to solve the fuel efficiency problem

$$\underset{u,z}{\text{minimize}} \;\; \|z\|^2_{L^2(0,T)} + \lambda \left| y^* - \int_0^T u(t)\,dt \right|^2 \;\; \text{subject to (1) and (2).}$$



Solution

We discretize the fuel efficiency problem into a nonlinear program (NLP).



So why ROL?

Numerics

```
Composite-step trust-region solver
iter  fval           cnorm          gLnorm         snorm          delta     nnorm     tnorm     #fval   #grad  ...
0     5.333333e+03   2.027966e-13   2.666783e+00
1     5.223834e+03   2.933645e+00   3.555940e+00   1.000000e+02   2.00e+02  1.13e-14  1.00e+02  3       3
2     5.074484e+03   3.977936e+00   5.320566e+00   2.000000e+02   2.00e+02  1.06e-01  2.00e+02  5       5
3     4.936750e+03   1.929162e+00   6.883693e+00   1.657243e+02   1.16e+03  1.61e-01  1.66e+02  7       7
...
47    4.426957e+03   1.813330e-04   9.328418e-02   2.898613e+00   1.16e+03  7.35e-06  2.90e+00  95      95
48    4.426934e+03   6.805572e-05   4.641692e-02   1.479816e+00   1.16e+03  1.10e-05  1.48e+00  97      97
49    4.426917e+03   1.176645e-04   7.690407e-02   2.328988e+00   1.16e+03  4.24e-06  2.33e+00  99      99
50    4.426902e+03   4.457843e-05   3.584340e-02   1.192131e+00   1.16e+03  7.13e-06  1.19e+00  101     101
...
```

```
Composite-step trust-region solver
iter  fval           cnorm          gLnorm         snorm          delta     nnorm     tnorm     #fval   #grad  ...
0     5.333333e+03   1.570856e-15   1.803732e+02
1     4.976505e+03   7.464298e-01   1.380737e+02   2.175210e+01   1.00e+02  3.03e-15  2.18e+01  3       3
2     5.252000e+03   2.467093e-02   2.549998e+02   2.755372e+00   1.00e+02  2.75e+00  5.33e-02  5       5
3     4.473015e+03   7.617080e-02   2.595459e+01   7.041189e+00   1.00e+02  1.23e-01  7.04e+00  7       7
4     4.428484e+03   2.072535e-03   3.485754e+00   1.936220e+00   1.00e+02  3.08e-01  1.91e+00  9       9
5     4.426855e+03   3.830153e-06   7.137584e-01   8.183971e-02   1.00e+02  8.98e-03  8.13e-02  11      11
6     4.426841e+03   1.090076e-06   6.769629e-03   4.490118e-02   1.00e+02  1.87e-05  4.49e-02  13      13
7     4.426840e+03   8.296731e-12   5.966856e-04   1.035859e-04   1.00e+02  4.58e-06  1.03e-04  15      15
8     4.426840e+03   3.307995e-13   3.785700e-06   1.927025e-05   1.00e+02  2.37e-11  1.93e-05  17      17
Optimization Terminated with Status: Converged
```

Custom Linear Algebra – A Feature of ROL

ROL makes it easy to tailor inner products to problems.

For example, we can think of our control z as an element of a Hilbert space $\mathcal{H}$ with the inner product

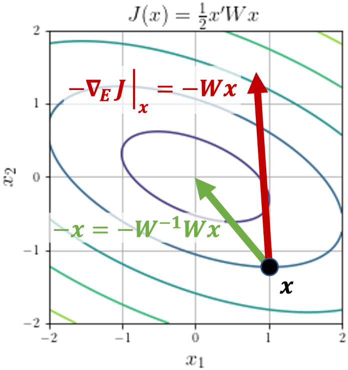
$$\langle f, g \rangle = \int_0^T f(t)g(t)dt.$$

The discretized analogue of $\mathcal{H}$ is a finite-dimensional space whose inner product is weighted by a quadrature matrix W – i.e., $\langle f, g \rangle = f' W g$.

A gradient with respect to a vector in the finite-dimensional space will be a function of W.

$$\lim_{h \to 0} \frac{|J(x+h) - J(x) - \langle \nabla J\big|_x, h \rangle|}{h} = 0$$

$$\implies \nabla J\big|_x = W^{-1} \nabla_E J\big|_x$$



- Málek, Josef, and Zdeněk Strakoš. *Preconditioning and the Conjugate Gradient Method in the Context of Solving PDEs*. SIAM, 2014.

ROL

Trilinos package for **large-scale optimization**. Uses: optimal design, optimal control and inverse problems in engineering applications; mesh optimization; image processing.

RAPID OPTIMIZATION LIBRARY

Numerical optimization made practical:
Any application, any hardware, any problem size.

- **Modern optimization algorithms.**
- **Maximum HPC hardware utilization.**
- **Special programming interfaces for simulation-based optimization.**
- **Optimization under uncertainty.**

- Hardened, production-ready algorithms for unconstrained, equality-constrained, inequality-constrained and nonsmooth optimization.
- Novel algorithms for optimization under uncertainty and risk-averse optimization.
- Unique capabilities for optimization-guided inexact and adaptive computations.
- Geared toward maximizing HPC hardware utilization through direct use of application data structures, memory spaces, linear solvers and nonlinear solvers.
- Special interfaces for engineering applications, for streamlined and efficient use.
- Rigorous implementation verification: finite difference and linear algebra checks.
- Hierarchical and custom (user-defined) algorithms and stopping criteria.

Formalism and Algorithms

Mathematical Formalism

ROL solves (smooth) nonlinear optimization problems numerically

$$\underset{x}{\text{minimize}} \;\; J(x) \;\; \text{subject to} \;\; \begin{cases} c(x) = 0 \\ \ell \le x \le u \\ Ax = b. \end{cases} \tag{G}$$

Here, x belongs to a Banach space $\mathcal{X}$ and

$$J : \mathcal{X} \to \mathbb{R}, \quad c : \mathcal{X} \to \mathcal{C}, \quad \text{and} \quad A : \mathcal{X} \to \mathcal{D},$$

where $\mathcal{C}$ and $\mathcal{D}$ are Banach spaces as well.

All three of these maps are Fréchet differentiable. In addition, A is linear.

The bounds $\ell \le x \le u$ apply pointwise.

Algorithms

Type U
"Unconstrained"

$$\underset{x}{\text{minimize}} \;\; J(x)$$

$$\text{subject to} \;\; \begin{cases} \\ \\ Ax = b \end{cases}$$

Methods:

- trust region and line search globalization
- gradient descent, quasi and inexact Newton, nonlinear conjugate gradient.

Type B
"Bound Constrained"

$$\underset{x}{\text{minimize}} \;\; J(x)$$

$$\text{subject to} \;\; \begin{cases} \\ \ell \le x \le u \\ Ax = b \end{cases}$$

Methods:

- projected gradient and projected Newton, primal-dual active set.

Type E
"Equality Constrained"

$$\underset{x}{\text{minimize}} \;\; J(x)$$

$$\text{subject to} \;\; \begin{cases} c(x) = 0 \\ \\ Ax = b \end{cases}$$

Methods:

- composite step SQP and ...

Type G
"General Constraints"

$$\underset{x}{\text{minimize}} \;\; J(x)$$

$$\text{subject to} \;\; \begin{cases} c(x) = 0 \\ \ell \le x \le u \\ Ax = b \end{cases}$$

Methods:

- augmented Lagrangian, interior point, Moreau-Yosida, stabilized LCL.

API

ROL::Objective

$$\min_{x} \; J(x) \quad \text{subject to} \quad \begin{cases} c(x) = 0 \\ \ell \le x \le u \\ Ax = b \end{cases}$$

Member Functions

- `value` - $J(x)$
- `gradient` - $g = \nabla J(x)$
- `hessVec` - $Hv = [\nabla^2 J(x)]v$
- `update` - modify member data
- `invHessVec` - $H^{-1}v = [\nabla^2 J(x)]^{-1}v$
- `precond` - approximate $H^{-1}v$
- `dirDeriv` - $\frac{d}{dt}J(x+tv)|_{t=0}$

(pure virtual   virtual   optional)

- We do not need to specify linear operators with matrices – their action on vectors is enough.
- ROL works best with analytic derivatives. Without them, ROL defaults to finite difference approximations.
- Tools: `checkGradient`, `checkHessVec`, `checkHessSym`.

ROL::Objective

$$\min_x \; J(x) \quad \text{subject to} \quad \begin{cases} c(x) = 0 \\ \ell \le x \le u \\ Ax = b \end{cases}$$

Member Functions

- value - $J(x)$
- gradient - $g = \nabla J(x)$
- hessVec - $Hv = [\nabla^2 J(x)]v$
- update - modify member data
- invHessVec - $H^{-1}v = [\nabla^2 J(x)]^{-1}v$
- precond - approximate $H^{-1}v$
- dirDeriv - $\frac{d}{dt} J(x+tv)|_{t=0}$

(pure virtual virtual optional)

$$J(u,z) = \|z\|^2_{L^2(0,T)} + \lambda \left| y^* - \int_0^T u(t)\,dt \right|^2$$

```cpp
class RocketObjective : public ROL::Objective<double>
{
 ...

 public:

  Objective(double targetHeight_, double lambda_,
            const std::vector<double>& w_,) :
   targetHeight(targetHeight_), lambda(lambda_), w(w_)
  {
   N = w.size();
  }

  double value(const ROL::Vector<double>& x, double& tol)
  {
   const std::vector<double>& z = getControl(x);
   const std::vector<double>& u = getState(x);

   int i;

   double zIntegral = 0;
   for (i = 0; i < N; ++i)
     zIntegral += w[i]*z[i]*z[i];

   double uIntegral = 0;
   for (i = 0; i < N; ++i)
     uIntegral += w[i]*u[i];

   return zIntegral + lambda*std::pow(uIntegral - targetHeight, 2);
  }
  ...
}
```

ROL::Constraint

$$\underset{x}{\text{minimize}}\ J(x)\ \text{subject to}\ \begin{cases} c(x) = 0 \\ \ell \le x \le u \\ Ax = b \end{cases}$$

Member Functions

- `value` - $c(x)$
- `applyJacobian` - $[c'(x)]v$
- `applyAdjointJacobian` - $[c'(x)]^* v$
- `applyAdjointHessian` - $[c''(x)](v, \cdot)^* u$
- `update` - modify member data
- `applyPreconditioner`
- `solveAugmentedSystem`

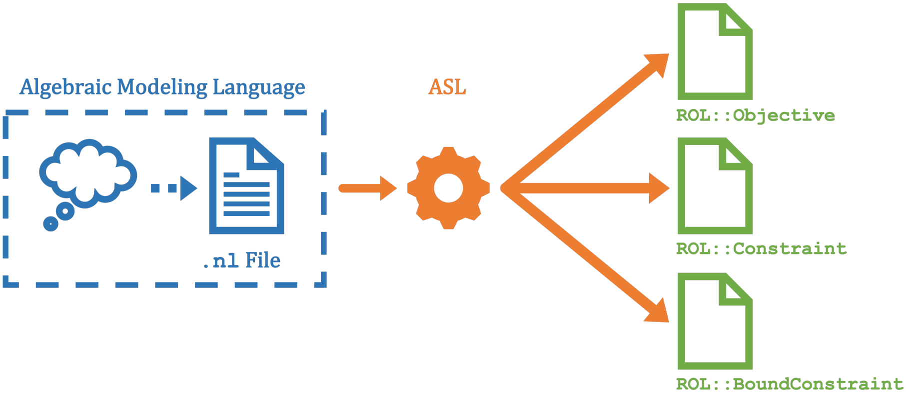
`ROL::BoundConstraint` implements $\ell \le x \le u$.

$$\frac{du}{dt} + k\frac{d\log m}{dt} + g = 0 \quad \text{and} \quad \frac{dm}{dt} = -z$$

```
class RocketConstraint : public ROL::Constraint<double>
{
 private:
  ...

  void computeMass(const std::vector<double>& z)
  {
    mass[0] = initialMass - dt*z[0];
    for (int i = 1; i < N; ++i)
      mass[i] = mass[i - 1] - dt*z[i];
  }

 public:
  ...

  void update(const ROL::Vector<Real> &x, UpdateType type, int iter = -1)
  {
    const std::vector<double>& z = getControl(x);
    computeMass(z);
  }

  void value(ROL::Vector<double>& c, const ROL::Vector<double>& x, double& tol)
  {
    std::vector<double>& cstd = getVector(c);

    const std::vector<double>& z = getControl(x);
    const std::vector<double>& u = getState(x);

    cstd[0] = u[0] + k*std::log(mass[0]/mInitial) + g*dt;
    for(int i = 1; i < N; ++i)
      cstd[i] = u[i] - u[i-1] + k*std::log(mass[i]/mass[i - 1]) + g*dt;
  }
  ...
}
```

AMPL-Solver Interface Library (ASL)

ROL can be a backend for algebraic modeling languages. We have an interface to AMPL.



- *Note*: Our current interface is matrix free, i.e., we do not yet precondition with the matrix information from ASL.

The `SimOpt` Interface

Our rocket example – and optimal control in general – is what we call a simulation-constrained optimization problem.

Full Space Formulation

The problem is *explicitly* constrained:

$$\underset{(u,z)\in\mathcal{U}\times\mathcal{Z}}{\text{minimize}} \quad J(u,z)$$
$$\text{subject to} \quad c(u,z)=0$$

Reduced Space Formulation

The problem is *implicitly* constrained:

$$\underset{z\in\mathcal{Z}}{\text{minimize}} \quad J(S(z),z),$$

where $u=S(z)$ solves $c(u,z)=0$.

- z = the vector being optimized (often a control or set of parameters)
- u = a state resulting from c (the simulation)

In engineering applications, c is often a differential equation.

ROL's `SimOpt` interface is "middleware":

- u and z are separated out of the optimization vector x
- converting full space formulations to reduced space ones (and vice-versa) is trivial.

The `SimOpt` Interface

ROL::Objective_SimOpt

- `value(u,z)`
- `gradient_1(g,u,z)`
- `gradient_2(g,u,z)`
- `hessVec_11(hv,v,u,z)`
- `hessVec_12(hv,v,u,z)`
- `hessVec_21(hv,v,u,z)`
- `hessVec_22(hv,v,u,z)`

A mnemonic:

- $1 =$ "sim" $= u$
- $2 =$ "opt" $= z$.

ROL::Constraint_SimOpt

- `value(u,z)`
- `applyJacobian_1(jv,v,u,z)`
- `applyJacobian_2(jv,v,u,z)`
- `applyInverseJacobian_1(ijv,v,u,z)`
- `applyAdjointJacobian_1(ajv,v,u,z)`
- `applyAdjointJacobian_2(ajv,v,u,z)`
- `applyInverseAdjointJacobian_1(iajv,v,u,z)`
- `applyAdjointHessian_11(ahwv,w,v,u,z)`
- `applyAdjointHessian_12(ahwv,w,v,u,z)`
- `applyAdjointHessian_21(ahwv,w,v,u,z)`
- `applyAdjointHessian_22(ahwv,w,v,u,z)`
- `solve(u,z)`

Stochastic Optimization

ROL also has middleware for stochastic problems:

$$\underset{x \in C}{\text{minimize}} \; \mathcal{R}(f(x, \xi)).$$

Here, x is a deterministic decision but ξ is a set of random parameters, i.e., $\xi = \xi(\omega)$.

For each x, $f(x, \xi)$ is a random variable $F_x(\omega)$.

$\mathcal{R}$ is a functional on these random variables that quantifies risk. $\mathcal{R}$ could be – for instance –

- an expectation: $\mathcal{R}(F_x) := \mathbb{E}[F_x]$,
- a quantile (the value at risk),
- a distributionally robust model

$$\mathcal{R}(F_x) = \sup_{P \in \mathcal{U}} \mathbb{E}_P[F_x].$$

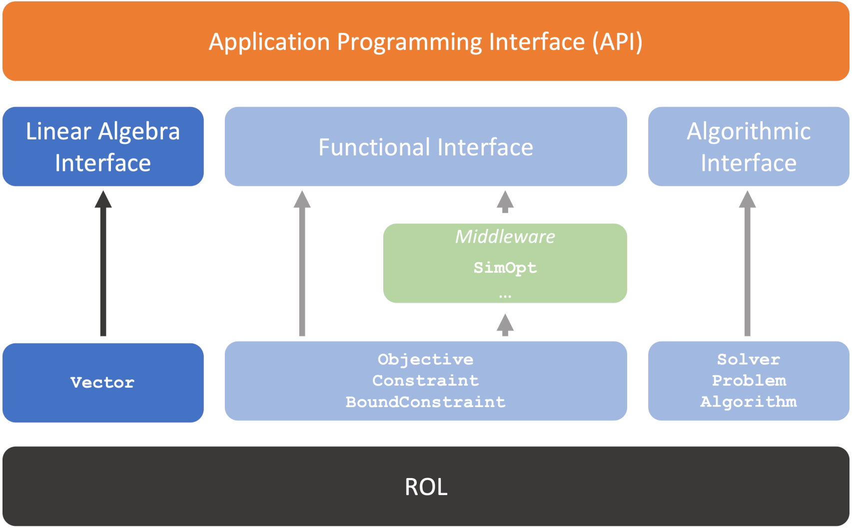
The set $\mathcal{C}$ can include both stochastic (e.g., $\ell \leq \tilde{\mathcal{R}}(G_x) \leq u$) and deterministic constraints.

ROL solves these problems in the usual way: $\mathcal{R}(F_x)$ and the stochastic constraints in C are replaced with approximations. For example, we might take

$$\mathbb{E}[F(x)] \approx \frac{1}{N} \sum_{k=1}^{N} f(x, \xi_k),$$

where the ξ_k are independent and identically distributed samples of ξ.

Design


Application Programming Interface (API)
Linear Algebra Interface
Functional Interface
Algorithmic Interface
Middleware
SimOpt
...
Vector
Objective
Constraint
BoundConstraint
Solver
Problem
Algorithm
ROL

`ROL::Vector` – A Linear Algebra Interface

Optimization algorithms manipulate vectors. But the *implementation* of these vectors do not affect what the algorithms do. (For example, the number of iterations before gradient descent reaches some stopping condition will be the same whether x – the vector being optimized – is stored on a laptop or distributed over a network.)

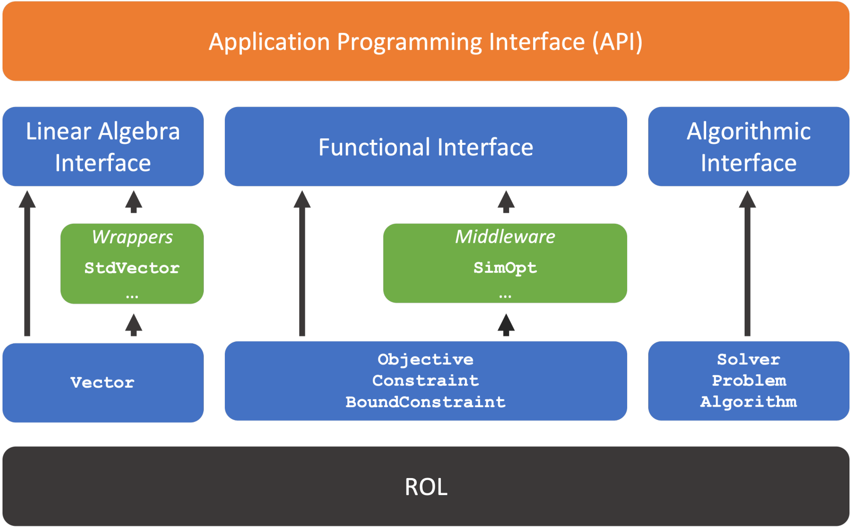
ROL similarly relegates the inner workings of vectors to users. As a result,

- ROL is hardware agnostic. Sandians run ROL on personal computers (in serial and MPI parallel), GPUs, and supercomputers too.
- Users can easily tune the linear algebra of a problem by inheriting from an instance of `ROL::Vector` (which we did in the rocket example).

Member Functions

- `dot`
- `plus`
- `norm`
- `scale`
- `clone`
- `axpy`
- `dual`
- `zero`
- `set`
- `basis`
- `reduce`
- `dimension`
- `applyUnary`
- `applyBinary`

Design


Application Programming Interface (API)
Linear Algebra Interface
Functional Interface
Algorithmic Interface
Wrappers
StdVector
...
Middleware
SimOpt
...
Vector
Objective
Constraint
BoundConstraint
Solver
Problem
Algorithm
ROL

Context

Related Software

- *Hilbert Class Library (HCL) - Rice University*

  An abstract linear algebra interface.

- *Trilinos - Sandia National Laboratories*

  Collection of linear and nonlinear solvers based on linear algebra abstractions.

  - *RTOp and Thyra*

    Packages for an extended set of algebraic abstractions.

  - *MOOCHO*

    Optimization package built on Thyra that solves reduced space formulations.

- *Rice Vector Library (RVL) - Rice University*

  A revamp of HCL.

- *Trilinos* (continued)

  - *Aristos*

    Optimization package with algebra abstractions and full space formulations.

  - *Optipack*

    A few special-purpose optimization routines using algebra abstractions.

- *PEOpt - Sandia National Laboratories*

  Optimization packages using an alternative implementation of algebra abstractions.

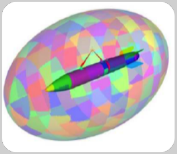
- *Optizelle - OptimoJoe*

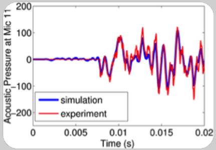
  Successor to PEOpt.

Applications

Inverse Problems in Acoustics/Elasticity

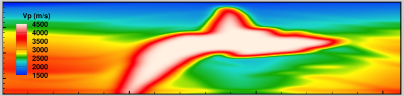
Sierra/SD – structural dynamics software

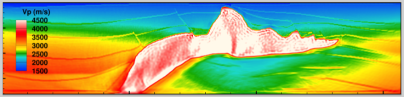




1M optimization + 1M state variables

DGM – a library of discontinuous Galerkin methods for solving partial differential equations

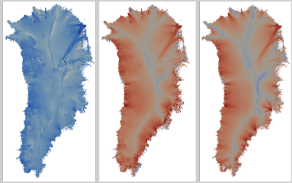




500K optimization + 2M×5K state variables

Estimating Basal Friction of Ice Sheets

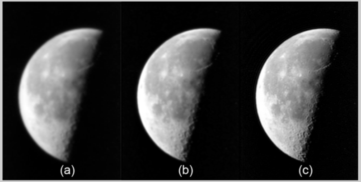
Albany – a multiphysics simulator



5M optimization + 20M state variables

Super-Resolution Imaging

GPU processing with *ArrayFire*



250K optimization variables on an NVIDIA Tesla

Conclusions

- ROL is C++ code for solving large optimization problems.
- It implements a variety of matrix-free algorithms and has been "battle-tested" on problems at Sandia.
- ROL has a flexible interface that can connect with algebraic modeling languages. And, importantly, ROL lets users implement their own vectors.