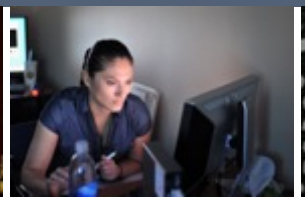
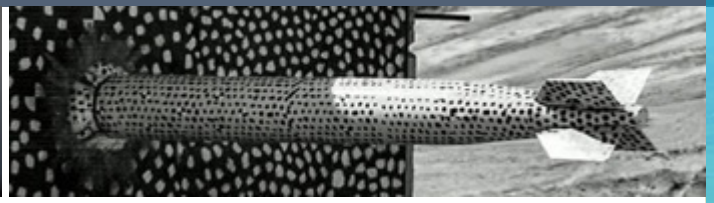
SAND2021-14942 C: UUR

SAND2022-14792C

Sandia National Laboratories

Nonlinear Analysis Product Area Update





PRESENTED BY

Roger Pawlowski

Current Package Owners: Bartlett, Ober, Pawlowski, Phipps, Perego, Ridzal and Tezaur

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.

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.

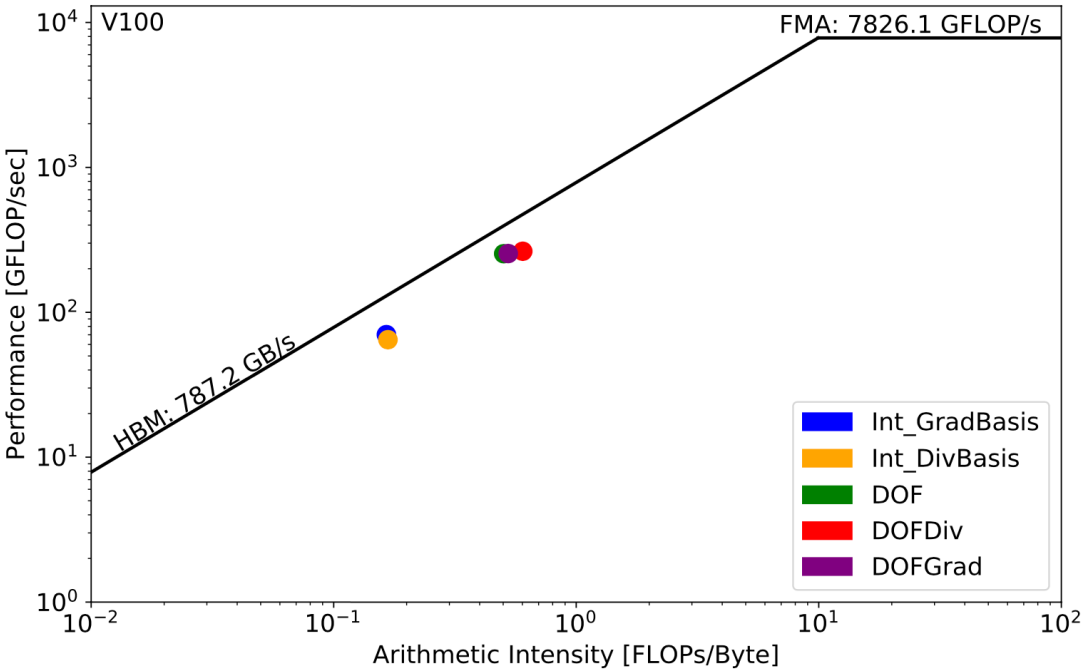
Nonlinear Analaysis Product Area

- Sacado: Automatic differentiation via operator overloading of scalar types
- NOX: Globalized Newton-based solvers
- LOCA: Stability, bifurcation and continuation algorithms
- ROL: Optimization
- Tempus: Time integration
- PIRO: Parameters-In-Response-Out: Utility layer
- Thyra: Abstaction Layer for linear algebra, linear solvers and nonlinear solvers
- Deprecated: Rythmos and PIKE

Nonlinear Product Area Updates

- Sacado (Phipps)
  - Library is UVM free
  - Tests refactored to be UVM free. Only certain DFad use-cases require UVM (and this won't change).
  - Paper on performance portability published in TOMS:
    - https://dl.acm.org/doi/10.1145/3560262
  - Ported to HIP in FY22/Q2
  - Porting Sacado to the new MDSpan (C++23) will substantially simplify the integration with Kokkos

Roofline for Panzer AD Kernels on V100 with DFAD



- NOX/LOCA (Pawlowski/Phipps)
  - UVM free
  - Householder constraint solver added for Tpetra stack
  - Ported to HIP
  - This FY: bordering algorithms for Tpetra

$$\begin{bmatrix} J & A \\ B^T & C \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} F \\ G \end{bmatrix}$$

Sacado DFAD and HIP

- Sacado DFAD objects do dynamic memory allocation at the point of use.
  - Initialization could happen on host. For CUDA, UVM simplified this.
  - Temporary DFADs in a device kernel could do allocation at runtime
  - HIP does not allow for new and delete on device!!!
  - For CUDA we had a memory pool alternative, but that is not supported for HIP and will probably be deprecated from Kokkos.
- If using DFads with HIP we have switched to allocating temporaries on host. Be careful to allocate temporaries for each thread.
  - Shared memory is a better option if you have enough
- Best to use SFAD and SLFAD (also more performant!)

```
-    ScalarT tmp;
     if (NUM_FIELD_MULT == 0)
     {
       // Loop over the quadrature points, scale the integrand by the
       // multiplier, and then perform the actual integration, looping over the
       // bases.
       for (int qp(0); qp < numQP; ++qp)
       {
-        tmp = multiplier_ * scalar_(cell, qp);
+        tmp_(cell) = multiplier_ * scalar_(cell, qp);
         for (int basis(0); basis < numBases; ++basis)
-          field_(cell, basis) += basis_(cell, basis, qp) * tmp;
+          field_(cell, basis) += basis_(cell, basis, qp) * tmp_(cell);
       } // end loop over the quadrature points
     }
```

```
+    /// For storing temporaries, one value per thread
+    PHX::View<ScalarT*> tmp_;
+    /// For storing temporaries, one value per thread
+    PHX::View<ScalarT*> tmp2_;
+
```

```
+
+    // Allocate temporary
+    if (Sacado::IsADType<ScalarT>::value) {
+      const auto fadSize = Kokkos::dimension_scalar(field_.get_view());
+      tmp_ = PHX::View<ScalarT*>("IntegratorBasisTimesScalar::tmp_",field_.extent(0),fadSize);
+      if (fieldMults_.size() > 1)
+        tmp2_ = PHX::View<ScalarT*>("IntegratorBasisTimesScalar::tmp_",field_.extent(0),fadSize);
+    } else {
+      tmp_ = PHX::View<ScalarT*>("IntegratorBasisTimesScalar::tmp_",field_.extent(0));
+      if (fieldMults_.size() > 1)
+        tmp2_ = PHX::View<ScalarT*>("IntegratorBasisTimesScalar::tmp_",field_.extent(0));
+    }
```

Tempus – Time-Integration Package (Ober)

- Provides time-integration methods for first and second-order ODEs
  - Explicit: $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{p}, t)$ $\quad\quad$ $\ddot{\mathbf{x}}(t) = \mathbf{f}(\dot{\mathbf{x}}(t), \mathbf{x}(t), \mathbf{p}(t), t)$
  - Implicit: $\mathbf{f}(\dot{\mathbf{x}}(t), \mathbf{x}(t), t) = 0$ $\quad\quad$ $\mathbf{f}(\ddot{\mathbf{x}}(t), \dot{\mathbf{x}}(t), \mathbf{x}(t), t) = 0$
- Developed to support advanced analysis techniques
  - Embedded transient sensitivity analysis and UQ capabilities
    - Forward and Adjoint Sensitivities
  - Couples with ROL to provide transient optimization capabilities
- Provides "out of the box" capabilities
  - Embedded error analysis for variable time steps
  - Temporal solution interpolation
  - Solution history management
  - *47+ Steppers*
- Provides customization capabilities
  - Ability to incorporate application-specific time steppers
  - *Application-specific time-step control* for variable time steps
  - Observers – ability to insert application-specific algorithms
  - *Time-event management*, e.g.,
    - Problem-specific events – switch flipped, x-ray impingement, …
    - Solution/diagnostic/debug/in-situ visualization output

Tempus Steppers

- Forward Euler
- Backward Euler
- Explicit Runge-Kutta (ERK) (*15+*)
- Diagonally Implicit Runge-Kutta (DIRK) (*20+*)
- Newmark, HHT-a, Leapfrog
- IMEX-RK (+Partitioned) (3+)
- BDF2
- Trapezoidal
- 1st Order Splitting
- *Subcycling*

- Other features
  - *Generate consistent initial conditions*
  - *FSAL when possible*
  - Extensive verification & unit testing
  - Documentation (Doxygen)

Italicized – latest development

Tempus Development

- Removed deprecated code for major Trilinos 14.0 release.
- Begun examples of Tempus usage.
- Removal of internal usage of ParameterLists (still can construction from them and provide one with current parameters).
- Improvements to forward and adjoint sensitivity analysis capabilities for L2M Embedded Components
    - Enable separate sensitivity ModelEvaluators (MEs) in Forward Sensitivity Analysis (FSA) integrators.
    - Allow 2nd adjoint ME for pseudo-transient adjoint integrator.

ROL (Ridzal)

- Reminder: ROL 2.0 released in April 2021.
- Update: ROL 1.0 interfaces will be maintained until October 2023, subject to the established Trilinos deprecation process.
- ROL 2.0 transition website: https://github.com/trilinos/Trilinos/blob/master/packages/rol/Version-2.0.md
- New functionality in 2022:
  - Module for **Optimal Experimental Design (OED)** with capabilities to solve A, C, I, R, and D-optimal design problems. Leverages ROL's existing API. Example: Optimal sensor placement.
  - **.NL file** solver for ROL. ROL can now be used as an optimization backend to **algebraic modeling languages** like AMPL, Pyomo, and JuMP since they export problems in this format.
- Functionality planned for 2023:
  - Methods to minimize the sum of smooth and **nonsmooth** functions. This class of problems arises in sparse estimation and control, imaging, risk-averse optimization and constrained optimization.
  - Easily maintainable, “self-generating” **Python interface** to ROL.
  - Several **usability** improvements: parameter list validation, better finite differencing, etc.

PIRO (Perego and Tezaur)

Provides driver classes for:

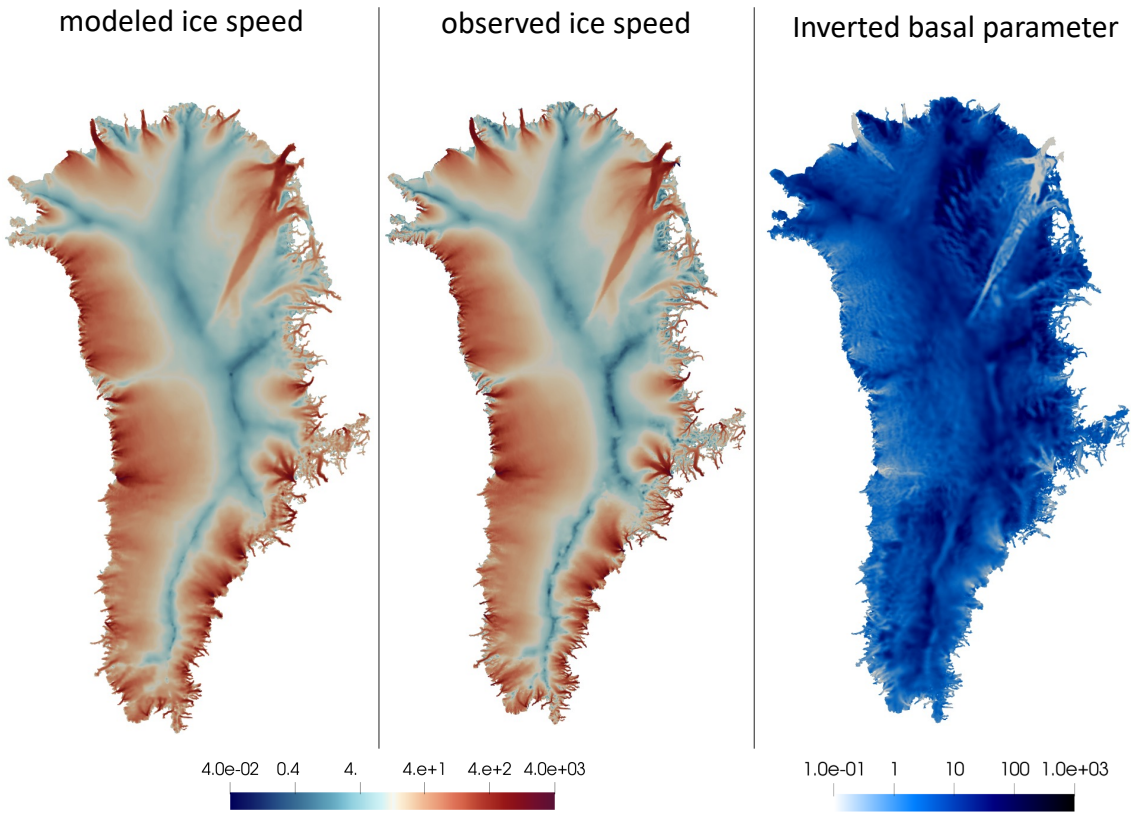
- solving nonlinear (time-dependent) problems (NOX, Tempus, Epetra/Tpetra solver stack),
- continuation problems and bifurcation analysis (LOCA),
- time-dependent problems (Tempus),
- computation of forward and adjoint sensitivities
- snapshot PDE-constrained optimization (ROL)

Updates:

- It can build if Epetra is disabled, although testing coverage should be improved.
- Added forward-only transient tests using Tempus

Plan for FY23:

- Switch to ROL2
- Enable *transient* PDE-constrained optimization



PDE-constrained optimization to calibrate an ice-sheet model of Greenland by matching observations of surface ice velocity (Perego, Siam News, 2022)