Advanced Analysis, Sensitivity, and Optimization Capabilities in the Trilinos Collection

Roscoe A. Bartlett
Department of Optimization & Uncertainty Estimation

Sandia National Laboratories

Frontiers of Geophysical Simulation

August 20, 2009

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.
Overview of Trilinos
Sandia Physics Simulation Codes

- **Element-based**
  - Finite element, finite volume, finite difference, network, etc...

- **Large-scale**
  - Billions of unknowns

- **Parallel**
  - MPI-based SPMD
  - Distributed memory

- **C++**
  - Object oriented
  - Some coupling to legacy Fortran libraries
Motivation For Trilinos

• Sandia does LOTS of solver work.
• 11 years ago ...
  – Aztec was a mature package. Used in many codes.
  – FETI, PETSc, DSCPack, Spooles, ARPACK, DASPK, and many other codes were (and are) in use.
  – New projects were underway or planned in multi-level preconditioners, eigensolvers, non-linear solvers, etc…
• The challenges:
  – Little or no coordination was in place to:
    • Efficiently reuse existing solver technology.
    • Leverage new development across various projects.
    • Support solver software processes.
    • Provide consistent solver APIs for applications.
  – ASCI was forming software quality assurance/engineering (SQA/SQE) requirements:
    • Daunting requirements for any single solver effort to address alone.
• Trilinos\textsuperscript{1} is an evolving framework to address these challenges:
  – Follow a \textbf{TOOLKIT} approach.
  – Fundamental atomic unit is a \textit{package}.
  – Includes core set of vector, graph and matrix classes (Epetra/Tpetra packages).
  – Provides a common abstract solver API (Thyra package).
  – Provides a ready-made package infrastructure (\texttt{new\_package} package):
    • Source code management (cvs, bonsai \textRightarrow{} Moving to git).
    • Build tools (CMake [\texttt{New}]).
    • Automated regression testing (CTest/CDash [\texttt{New}]).
    • Communication tools (mailman mail lists).
  – Specifies requirements and suggested practices for package SQA.
• In general allows us to categorize efforts:
  – Efforts best done at the Trilinos framework level (useful to most or all packages).
  – Efforts best done at a package level (peculiar or important to a package).
  – Allows package developers to focus only on things that are unique to their package.

1. Trilinos loose translation: “A string of pearls”
• **Desktop: Development and more…**
  – Native MS Windows (Visual C++, CMake)
  – Native MAC (XCode, CMake)

• **Capability machines:**
  – Redstorm (XT3), Clusters
  – Roadrunner (Cell-based).
  – Multicore nodes.

• **Parallel software environments:**
  – MPI of course.
  – UPC, CAF, threads, vectors,…
  – Combinations of the above.

• **User “skins”:**
  – C++/C, Python
  – Fortran
  – Web, CCA
Trilinos Strategic Goals

- **Scalable Computations**: As problem size and processor counts increase, the cost of the computation will remain nearly fixed.

- **Hardened Computations**: Never fail unless problem essentially intractable, in which case we diagnose and inform the user why the problem fails and provide a reliable measure of error.

- **Full Vertical Coverage**: Provide leading edge enabling technologies through the entire technical application software stack: from problem construction, solution, analysis and optimization.

- **Grand Universal Interoperability**: All Trilinos packages will be interoperable, so that any combination of solver packages that makes sense algorithmically will be possible within Trilinos.

- **Universal Accessibility**: All Trilinos capabilities will be available to users of major computing environments: C++, Fortran, Python and the Web, and from the desktop to the latest scalable systems.

- **Universal Solver RAS**: Trilinos will be:
  - **Reliable**: Leading edge hardened, scalable solutions for each of these applications
  - **Available**: Integrated into every major application at Sandia
  - **Serviceable**: Easy to maintain and upgrade within the application environment.
# Trilinos Package Summary

<table>
<thead>
<tr>
<th>Objective</th>
<th>Package(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Discretizations</strong></td>
<td></td>
</tr>
<tr>
<td>Meshing &amp; Spatial Discretizations</td>
<td>phdMesh, Intrepid, Phalanx, Shards, Pamgen, Sundance, FEI</td>
</tr>
<tr>
<td>Time Integration</td>
<td>Rythmos</td>
</tr>
<tr>
<td><strong>Methods</strong></td>
<td></td>
</tr>
<tr>
<td>Automatic Differentiation</td>
<td>Sacado</td>
</tr>
<tr>
<td>Mortar Methods</td>
<td>Moertel</td>
</tr>
<tr>
<td><strong>Core</strong></td>
<td></td>
</tr>
<tr>
<td>Linear algebra objects</td>
<td>Epetra, Jpetra, Tpetra</td>
</tr>
<tr>
<td>Abstract interfaces</td>
<td>Thyra, Stratimikos, RTop</td>
</tr>
<tr>
<td>Load Balancing</td>
<td>Zoltan, Isorropia</td>
</tr>
<tr>
<td>“Skins”</td>
<td>PyTrilinos, WebTrilinos, Star-P, ForTrilinos, CTrilinos</td>
</tr>
<tr>
<td>C++ utilities, I/O, thread API</td>
<td>Teuchos, EpetraExt, Kokkos, Triutils, TPI</td>
</tr>
<tr>
<td><strong>Solvers</strong></td>
<td></td>
</tr>
<tr>
<td>Iterative (Krylov) linear solvers</td>
<td>AztecOO, Belos, Komplex</td>
</tr>
<tr>
<td>Direct sparse linear solvers</td>
<td>Amesos</td>
</tr>
<tr>
<td>Direct dense linear solvers</td>
<td>Epetra, Teuchos, Pliris</td>
</tr>
<tr>
<td>Iterative eigenvalue solvers</td>
<td>Anasazi</td>
</tr>
<tr>
<td>ILU-type preconditioners</td>
<td>AztecOO, IFPACK</td>
</tr>
<tr>
<td>Multilevel preconditioners</td>
<td>ML, CLAPS</td>
</tr>
<tr>
<td>Block preconditioners</td>
<td>Meros</td>
</tr>
<tr>
<td>Nonlinear system solvers</td>
<td>NOX, LOCA</td>
</tr>
<tr>
<td>Optimization</td>
<td>MOOCHO, Aristos, GlobiPack, OptiPack, TriKota</td>
</tr>
<tr>
<td>Stochastic PDEs</td>
<td>Stokhos</td>
</tr>
</tbody>
</table>
### Automatic Intra-Package Dependency Handling

#### Trilinos/cmake/python/data/TrilinosPackageDependenciesTable.html

<table>
<thead>
<tr>
<th>Packages</th>
<th>P91</th>
<th>P92</th>
<th>P93</th>
<th>P94</th>
<th>P95</th>
<th>P96</th>
<th>P97</th>
<th>P98</th>
<th>P99</th>
<th>P100</th>
<th>P101</th>
<th>P102</th>
<th>P103</th>
<th>P104</th>
<th>P105</th>
<th>P106</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKL</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/Op</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/Kokkos</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/Spetra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/Smeg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/Paradiso</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/Telstra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/Selenite</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/Shade</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/Incopla</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/Tirano</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/ProgetLocation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/Pielme</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/Chips</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/AmcorC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/Conbel</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/Anonex</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/Ipamk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/Enronex</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/MOLES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/Beala</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/StratIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/StratIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/StratIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/StratIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/StratIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/StratIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/StratIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/StratIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/StratIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/StratIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/StratIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/StratIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/StratIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/StratIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/StratIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/StratIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/StratIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/StratIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/StratIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PKL/StratIn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(Partial) List of People Who Have Developed Trilinos

- Chris Baker
  Developer of Anasazi, RBGen, Tpetra

- Ross Bartlett
  Lead Developer of Thyra, Stratimikos and MOOCHO
  Developer of Rythmos, Teuchos

- Pavel Bochev
  Project Lead and Developer of Intrepid

- Paul Boggs
  Developer of Thyra

- Eric Boman
  Lead Developer of Isorropia
  Developer of Zoltan

- Todd Coffey
  Lead Developer of Rythmos

- David Day
  Developer of Komplex and Intrepid

- Karen Devine
  Lead Developer of Zoltan

- Clark Dohrmann
  Developer of CLAPS

- Michael Gee
  Developer of ML, NOX

- Bob Heaphy
  Lead Developer of Trilinos SQA

- Mike Heroux
  Trilinos Project Leader
  Lead Developer of Epetra, AztecOO, Kokkos, Komplex, IFPACK, Thyra, Tpetra
  Developer of Amesos, Belos, EpetraExt, Jpetra

- Ulrich Hetmaniuk
  Developer of Anasazi

- Robert Hoekstra
  Lead Developer of EpetraExt
  Developer of Epetra, Thyra, Tpetra

- Russell Hooper
  Developer of NOX

- Vicki Howle
  Lead Developer of Meros
  Developer of Belos and Thyra

- Jonathan Hu
  Developer of ML

- Sarah Knepper
  Developer of Komplex

- Tammy Kolda
  Lead Developer of NOX

- Joe Kotulska
  Lead Developer of Ifpack

- Rich Lehoucq
  Developer of Anasazi and Belos

- Kevin Long
  Lead Developer of Thyra, Sundance
  Developer of Teuchos

- Roger Pawlowski
  Lead Developer of NOX, Phalanx
  Developer of Shards, LOCA

- Michael Phenow
  Trilinos Webmaster
  Lead Developer of New Package

- Eric Phipps
  Lead Developer of Sacado
  Developer of LOCA, NOX

- Denis Rizkallah
  Lead Developer of Aristos and Intrepid

- Marzio Sala
  Lead Developer of Didasko and IFPACK
  Developer of ML, Amesos

- Andrew Salinger
  Lead Developer of LOCA

- Paul Sexton
  Developer of Epetra and Tpetra

- Bill Spotz
  Lead Developer of PyTrilinos
  Developer of Epetra, New Package

- Ken Stanley
  Lead Developer of Amesos and New Package

- Heidi Thornquist
  Lead Developer of Anasazi, Belos, RBGen, and Teuchos

- Ray Tuminaro
  Lead Developer of ML and Meros

- Jim Willenbring
  Developer of Epetra and New Package.
  Trilinos library manager

- Alan Williams
  Lead Developer of Isorropia
  Developer of Epetra, EpetraExt, AztecOO, Tpetra
Overview of Nonlinear Analysis, Sensitivity and Optimization Capabilities
Trilinos Nonlinear Solver and Analysis Packages

Nonlinear Problems: Given nonlinear operator $f(x, p) \in \mathbb{R}^{n+m} \rightarrow \mathbb{R}^{n}$

- Nonlinear equations: Solve $f(x) = 0$ for $x \in \mathbb{R}^{n}$
- Stability analysis: For $f(x, p) = 0$ find space $p \in \mathcal{P}$ such that $\frac{\partial f}{\partial x}$ is singular

Transient Nonlinear Problems:

- DAEs/ODEs 
  Solve $f(\dot{x}(t), x(t), t) = 0, t \in [0, T]$, $x(0) = x_0$, $\dot{x}(0) = x'_0$
  for $x(t) \in \mathbb{R}^{n}, t \in [0, T]$
- ODE/DAE Sensitivities ...

Optimization Problems:

- Unconstrained:
  Find $p \in \mathbb{R}^{m}$ that minimizes $g(p)$
- Constrained:
  Find $x \in \mathbb{R}^{n}$ and $p \in \mathbb{R}^{m}$ that:
  minimizes $g(x, p)$
  such that $f(x, p) = 0$
<table>
<thead>
<tr>
<th>Optimization</th>
<th>MOOCHO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unconstrained:</td>
<td></td>
</tr>
<tr>
<td>Constrained:</td>
<td></td>
</tr>
<tr>
<td>Find ( u \in \mathbb{R}^n ) that minimizes ( g(u) )</td>
<td></td>
</tr>
<tr>
<td>Find ( x \in \mathbb{R}^m ) and ( u \in \mathbb{R}^n ) that minimizes ( g(x, u) ) s.t. ( f(x, u) = 0 )</td>
<td></td>
</tr>
<tr>
<td>Bifurcation Analysis</td>
<td>LOCA</td>
</tr>
<tr>
<td>Given nonlinear operator ( F(x, u) \in \mathbb{R}^{n+m} )</td>
<td></td>
</tr>
<tr>
<td>For ( F(x, u) = 0 ) find space ( u \in U ) s.t. ( \frac{\partial F}{\partial x}(x, u) = 0 )</td>
<td></td>
</tr>
<tr>
<td>Transient Problems DAEs/ODEs:</td>
<td>Rythmos</td>
</tr>
<tr>
<td>Solve ( f(x(t), x(t), t) = 0 )</td>
<td></td>
</tr>
<tr>
<td>( t \in [0, T], x(0) = x_0 ), ( x(0) = x_0 )</td>
<td></td>
</tr>
<tr>
<td>for ( x(t) \in \mathbb{R}^n, t \in [0, T] )</td>
<td></td>
</tr>
<tr>
<td>Nonlinear Problems</td>
<td>NOX</td>
</tr>
<tr>
<td>Given nonlinear operator ( F(x) \in \mathbb{R}^m \to \mathbb{R}^n )</td>
<td></td>
</tr>
<tr>
<td>Solve ( F(x) = 0 ), ( x \in \mathbb{R}^n )</td>
<td></td>
</tr>
<tr>
<td>Linear Problems</td>
<td>AztecOO</td>
</tr>
<tr>
<td>Linear Equations:</td>
<td></td>
</tr>
<tr>
<td>Eigen Problems:</td>
<td>Belos</td>
</tr>
<tr>
<td>Given Linear Ops (Matrices) ( A, B \in \mathbb{R}^{m \times n} )</td>
<td></td>
</tr>
<tr>
<td>Solve ( Ax = b ) for ( x \in \mathbb{R}^n )</td>
<td>Ifpack, ML, etc...</td>
</tr>
<tr>
<td>Solve ( A\nu = \lambda B\nu ) for (all) ( \nu \in \mathbb{R}^n ), ( \lambda \in \mathbb{R} )</td>
<td>Anasazi</td>
</tr>
<tr>
<td>Distributed Linear Algebra</td>
<td></td>
</tr>
<tr>
<td>Matrix/Graph Equations</td>
<td>Epetra</td>
</tr>
<tr>
<td>Vector Problems:</td>
<td></td>
</tr>
<tr>
<td>Compute ( y = Ax; A = A(G); A \in \mathbb{R}^{m \times n}, G \in \mathbb{R}^{m \times n} )</td>
<td></td>
</tr>
<tr>
<td>Compute ( y = \alpha x + \beta w; \alpha = \langle x, y \rangle; \alpha, x, y \in \mathbb{R}^n )</td>
<td>Tpetra</td>
</tr>
</tbody>
</table>
Suite of time integration (discretization) methods

- Includes: backward Euler, forward Euler, explicit Runge-Kutta, and implicit BDF at this time
- Preliminary implicit Runge-Kutta methods (no local error control, other limitations)
- Binary checkpoints and restarting (new in Trilinos 10.0)
- Native support for operator split methods
- Highly modular
- Forward sensitivity computations
- Adjoint sensitivities being developed

Developers: Todd Coffey, Roscoe Bartlett
Why Sensitivities and Optimization?

The Standard Steady-State Forward Simulation Problem

For a given set of input parameters \( p \in \mathbb{R}^{n_p} \), solve the square state equations

\[
f(x, p) = 0
\]

for the state variables \( x \in \mathbb{R}^{n_x} \) then compute observation(s) \( y(x) \).

Example applications

- Discretized PDEs (e.g. finite element, finite volume, discontinuous Gelerkin, finite difference, …)
- Network problems (e.g. circuit simulation, power grids, …)
- …

Why is a forward solver not enough?

- A forward solve \( p \to g(x(p), p) \) can only give point-wise information, it can’t tell you what you ultimately want to know:
  - How to characterize the error in my model so that it can be improved? \( \rightarrow \) Error estimation
  - What is the uncertainty in \( x \) or \( g(x(p), p) \) given uncertainty in \( p \)? \( \rightarrow \) UQ
  - What is the “best” value of \( p \) so that my model \( f(x, p) = 0 \) fits exp. data? \( \rightarrow \) Param. Estimation
  - What is the “best” value for \( p \) to achieve some goal? \( \rightarrow \) Optimization

What are some of the tools that we need to answer these higher questions?

- Sensitivities and Optimization!
Derivatives and Sensitivity Computations
Steady-State Simulation-Constrained Sensitivities

Steady-State Simulation-Constrained Response

Compute \( g(x, p) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_s} \)

such that \( f(x, p) = 0 \)  \( (\text{where } f(x, p) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_s}) \)

Nonlinear elimination

\[ f(x, p) = 0 \quad p \rightarrow x(p) \quad p \rightarrow g(x(p), p) \rightarrow \hat{g}(p) \]

Reduced Response Function

Steady-State Sensitivities

State Sensitivity:

\[ \frac{\partial x}{\partial p} = \frac{\partial f^{-1}}{\partial x} \frac{\partial f}{\partial p} \]

Well suited for Newton Methods

Reduced Response Function Sensitivity:

\[ \frac{\partial \hat{g}}{\partial p} = \frac{\partial g}{\partial x} \frac{\partial x}{\partial p} + \frac{\partial g}{\partial p} \]

Forward (Direct) vs. Adjoint Sensitivities

Forward (Direct) Sensitivity Method:

\[ \frac{\partial \hat{g}}{\partial p} = \frac{\partial g}{\partial x} \left( -\frac{\partial f^{-1}}{\partial x} \frac{\partial f}{\partial p} \right) + \frac{\partial g}{\partial p} \]

Complexity \( O(n_g) \)

Adjoint Sensitivity Method:

\[ \frac{\partial \hat{g}^T}{\partial p} = \frac{\partial f^T}{\partial p} \left( -\frac{\partial f^{-1}}{\partial x} \frac{\partial g^T}{\partial x} \right) + \frac{\partial g^T}{\partial p} \]

Adjoint variables \( \lambda \)

Complexity \( O(n_g) \)

Uses for Sensitivities: Derivative-based optimization, UQ, error estimation etc.
Transient Sensitivities: State Equations with Responses

Fully Implicit ODE/DAE State Equations

\[ f(\dot{x}(t), x(t), p, t) = 0, \quad t \in [t_0, t_f] \]
\[ x(0) = x_0(p) \]
\[ \dot{x}(0) = \dot{x}_0(p) \]

Composite Response Function

\[ d(x, p) = \int_{t_0}^{t_f} g(\dot{x}(t), x(t), p, t) dt + h(\dot{x}(t_f), x(t_f), p) \]

Reduced Composite Response Function

\[ \tilde{d}(p, v) = \int_{t_0}^{t_f} \tilde{g}(p, t) dt + \tilde{h}(p) \]

where:

\[ \tilde{g}(p, t) = g(\dot{x}(p, t), x(p, t), p, t) \]
\[ \tilde{h}(p) = h(\dot{x}(p, t_f), x(p, t_f), p) \]

Forward Sensitivity Equations:

\[ \frac{\partial f}{\partial \dot{x}} \left( \frac{\partial \dot{x}}{\partial p} \right) + \frac{\partial f}{\partial x} \left( \frac{\partial x}{\partial p} \right) + \frac{\partial f}{\partial p} = 0, \quad t \in [t_0, t_f] \]

\[ \begin{align*}
\frac{\partial x(t_0)}{\partial p} &= \frac{\partial x_0}{\partial p} \\
\frac{\partial \dot{x}(t_0)}{\partial p} &= \frac{\partial \dot{x}_0}{\partial p}
\end{align*} \]

Forward Reduced Gradient

\[ \frac{\partial \hat{d}}{\partial p} = \int_{t_0}^{t_f} \left( \frac{\partial g}{\partial \dot{x}} \frac{\partial \dot{x}}{\partial p} + \frac{\partial g}{\partial x} \frac{\partial x}{\partial p} + \frac{\partial g}{\partial p} \right) dt + \left( \frac{\partial h}{\partial \dot{x}} \frac{\partial \dot{x}}{\partial p} + \frac{\partial h}{\partial x} \frac{\partial x}{\partial p} + \frac{\partial h}{\partial p} \right) \bigg|_{t=t_f} \]

Forward Sensitivity Methods:

- The forward sensitivities \( d(x)/d(p) \) are integrated right along with the forward state equation
- Can be solved using explicit or implicit time integration methods
- \( O(n_p) \) extra storage and computation per time step
- Reuse of forward solver integrator infrastructure, Jacobian/preconditioner storage

QASPR transient current sensitivities w.r.t. reaction parameters for an irradiated semiconductor device modeled with Charon

- Embedded sensitivities with AD/Sacado (Phipps) & Rythmos
- Finite differences (steplen=1e-2) (optimal steplen=1e-1)
- Embedded sensitivities vs. finite diff.
  - Much more accurate and robust!
  - 10x faster for 40 parameters!

Basic Steady-State Simulation-Constrained Optimization Problem:

Find \( x \in \mathbb{R}^{n_x} \) and \( p \in \mathbb{R}^{n_p} \) that:
- minimizes \( g(x, p) \)
- such that \( f(x, p) = 0 \)

Basic example optimization formations

- Parameter estimation / data reconciliation
- Optimal design
- Optimal control
- …

Define Lagrangian: \( L(x, p, \lambda) = g(x, p) + \lambda^T f(x, p) \)

First-order necessary optimality conditions

\[
\begin{align*}
\text{State equation:} & \quad \frac{\partial L}{\partial \lambda} = f(x, p) = 0 \\
\text{Adjoint equation:} & \quad \frac{\partial L}{\partial x} = \frac{\partial g}{\partial x} + \frac{\partial f}{\partial x} \lambda = 0 \\
\text{Gradient equation:} & \quad \frac{\partial L}{\partial p} = \frac{\partial g}{\partial p} + \frac{\partial f}{\partial p} \lambda = 0
\end{align*}
\]

\[
\frac{\partial g}{\partial p} = -\frac{\partial f}{\partial p} \frac{\partial f}{\partial x} \frac{\partial g}{\partial x} + \frac{\partial g}{\partial p} = 0
\]

Reduced gradient!

Trilinos Optimization Packages

- MOOCHO (R. Bartlett)
- Aristos (D. Ridzal)
Basic Steady-State Simulation-Constrained Optimization Problem:

Find $x \in \mathbb{R}^{n_x}$ and $p \in \mathbb{R}^{n_p}$ that:
- minimizes $g(x, p)$
- such that $f(x, p) = 0$

Two broad approaches for solving optimization problems:

- **Non-invasive (decoupled) approach (simulation constraints always satisfied):**

  Find $p \in \mathbb{R}^{n_p}$ that:
  - minimizes $\widehat{g}(p) = g(x(p), p)$
  - Optimization method never "sees" the state space!

- **Embedded (coupled) approach (converges optimality and feasibility together):**

  Find $x \in \mathbb{R}^{n_x}$ and $p \in \mathbb{R}^{n_p}$ that:
  - minimizes $g(x, p)$
  - such that $f(x, p) = 0$
  - Optimization method deals with the (parallel) state space and the parameter space together!
  - Requires special globalization methods to converge to a minimum!
A Spectrum of Optimization Methods from Decoupled to Coupled

**Fully Non-Invasive**

- Decreased impact to existing app code
- Ease of interfacing

**DAKOTA**

**MOOCHO**

**Aristos**

- Better scalability to large parameter spaces
- More accurate solutions
- Less computer time

**Fully Embedded**
Scalable Optimization Test Problem

Example: Parallel, Finite-Element, 2D, Diffusion + Reaction (GL) Model

\[
\begin{align*}
\min & \quad \frac{1}{2} \int_{\Omega} (x(y) - x^*(y))^2 dy \\
\text{s.t.} & \quad \nabla^2 x + \alpha (x - x^3) = r(y) \quad y \in \Omega \\
& \quad \frac{\partial x(y)}{\partial n} = q(p, y) \quad y \in \partial \Omega
\end{align*}
\]

- State PDE: Scalar Ginzburg-Landau equations (based on Denis Ridzal’s Ph.D. code)
- Discretization:
  - Second-order FE on triangles
  - \(n_x = 110,011\) state variables and equations
- Optimization variables:
  - Sine series basis
  - \(n_p = 8\) optimization variables
  - Note: \(df/dp\) is constant in this problem!!!
- Iterative Linear Solver: ILU (Ifpack), (GMRES) AztecOO

**Key Points**
- Simple physics but leads to very nonlinear state equations
- Inverse optimization problem is very ill posed in many instances
**Key Points**

- Finite differencing the underlying functions is much more efficient than finite differencing entire simulation!
- Finite differencing the underlying functions is more accurate!
- Coupled approach requires (almost) no extra application requirements!
ModelEvaluator Overview
Nonlinear Algorithms and Applications: Everyone for Themselves?

Nonlinear ANA Solvers in Trilinos

NOX / LOCA  Rythmos  MOOCHO

Xyce  Charon  Tramonto  Aria  Aleph

Trilinos and non-Trilinos Preconditioner and Linear Solver Capability

Key Point
- BAD
Overview of Nonlinear Model Evaluator Interface

Motivation: An interface for nonlinear problems is needed that will support a variety of different types of problems

- Nonlinear equations (and sensitivities)
- Stability analysis and continuation
- Explicit ODEs (and sensitivities)
- DAEs and implicit ODEs (and sensitivities)
- Unconstrained optimization
- Constrained optimization
- Uncertainty quantification
- ...

as well as different combinations of problem types such as:

- Uncertainty in transient simulations
- Stability of an optimum under uncertainty of a transient problem

Approach: Develop a single, scalable interface to address all of these problems

- (Some) Input arguments:
  - State and differential state: 
    \[ x \in \mathcal{X} \text{ and } \dot{x} = \frac{dx}{dt} \in \mathcal{X} \]
  - Parameter sub-vectors: 
    \[ p_l \in \mathcal{P}_l \text{ for } l = 0 \ldots N_p - 1 \]
  - Time (differential):
    \[ t \in \mathbb{R} \]

- (Some) Output functions:
  - State function:
    \[ (\dot{x}, x, \{p_l\}, t) \Rightarrow f \in \mathcal{F} \]
  - Auxiliary response functions:
    \[ (\dot{x}, x, \{p_l\}, t) \Rightarrow g_j \in \mathcal{G}_j, \text{ for } j = 0 \ldots N_g - 1 \]
  - State/state derivative operator (LinearOpWithSolve):
    \[ (\dot{x}, x, \{p_l\}, t) \Rightarrow W = A \frac{\partial f}{\partial x} + B \frac{\partial f}{\partial x} \]
Some Nonlinear Problems Supported by the ModelEvaluator

<table>
<thead>
<tr>
<th><strong>Nonlinear equations:</strong></th>
<th>Solve $f(x) = 0$ for $x \in \mathbb{R}^n$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Stability analysis:</strong></td>
<td>For $f(x, p) = 0$ find space $p \in \mathcal{P}$ such that $\frac{\partial f}{\partial x}$ is singular</td>
</tr>
<tr>
<td><strong>Explicit ODEs:</strong></td>
<td>Solve $\dot{x} = f(x, t) = 0, t \in [0, T], x(0) = x_0$, for $x(t) \in \mathbb{R}^n, t \in [0, T]$</td>
</tr>
<tr>
<td><strong>DAEs/Implicit ODEs:</strong></td>
<td>Solve $f(\dot{x}(t), x(t), t) = 0, t \in [0, T], x(0) = x_0, \dot{x}(0) = x'_0$, for $x(t) \in \mathbb{R}^n, t \in [0, T]$</td>
</tr>
<tr>
<td><strong>Explicit ODE Forward Sensitivities:</strong></td>
<td>Find $\frac{\partial x}{\partial p}(t)$ such that: $\dot{x} = f(x, p, t) = 0, t \in [0, T], x(0) = x_0$, for $x(t) \in \mathbb{R}^n, t \in [0, T]$</td>
</tr>
<tr>
<td><strong>DAE/Implicit ODE Forward Sensitivities:</strong></td>
<td>Find $\frac{\partial x}{\partial p}(t)$ such that: $f(\dot{x}(t), x(t), p, t) = 0, t \in [0, T], x(0) = x_0, \dot{x}(0) = x'_0$, for $x(t) \in \mathbb{R}^n, t \in [0, T]$</td>
</tr>
<tr>
<td><strong>Unconstrained Optimization:</strong></td>
<td>Find $p \in \mathbb{R}^m$ that minimizes $g(p)$</td>
</tr>
</tbody>
</table>
| **Constrained Optimization:** | Find $x \in \mathbb{R}^n$ and $p \in \mathbb{R}^m$ that:
  - minimizes $g(x, p)$
  - such that $f(x, p) = 0$ |
| **ODE Constrained Optimization:** | Find $x(t) \in \mathbb{R}^n$ in $t \in [0, T]$ and $p \in \mathbb{R}^m$ that:
  - minimizes $\int_0^T g(x(t), p)$
  - such that $\dot{x} = f(x(t), p, t) = 0$, on $t \in [0, T]$
  - where $x(0) = x_0$ |
Nonlinear Algorithms and Applications: Everyone for Themselves?

Nonlinear ANA Solvers in Trilinos

Key Point

- BAD
Nonlinear ANA Solvers in Trilinos

Key Points

- Provide single interface from nonlinear ANAs to applications
- Provide single interface for applications to implement to access nonlinear ANAs
- Provides shared, uniform access to linear solver capabilities
- Once an application implements support for one ANA, support for other ANAs can quickly follow
Trilinos “Skins” and Interoperability with Other Software
PyTrilinos provides Python access to Trilinos packages
• Uses SWIG to generate bindings.
• Epetra, AztecOO, IFPACK, ML, NOX, LOCA, Amesos and NewPackage are supported.
• ModelEvaluator wrapper is being developed
• Developers: Bill Spotz

CTrilinos provides C and Fortran 77 compatible wrappers to Trilinos
• Currently wraps just part of Epetra
• More wrappers to come => ModelEvaluator …
• Provides basic C++/Fortran interoperability for ForTrilinos interfaces
• Developers: ???

ForTrilinos developing full object-oriented interfaces to Trilinos
• Based on basic wrappers in CTrilinos
• Uses new OO features of Fortran 2003
• Developers: Damian Rousan
• Not in Trilinos 10.0
Trilinos / PETSc Interoperability

- **Epetra_PETScAIJMatrix class**
  - Derives from Epetra_RowMatrix
  - Wrapper for serial/parallel PETSc aij matrices
  - Utilizes callbacks for matrix-vector product, getrow
  - No deep copies

- Enables PETSc application to construct and call virtually any Trilinos preconditioner

- **ML accepts fully constructed PETSc KSP solvers as smoothers**
  - Fine grid only
  - Assumes fine grid matrix is really PETSc aij matrix

- **Complements Epetra_PETScAIJMatrix class**
  - For any smoother with getrow kernel, PETSc implementation should be *much* faster than Trilinos
  - For any smoother with matrix-vector product kernel, PETSc and Trilinos implementations should be comparable
Summary Wrap-up
External Visibility

• Awards: R&D 100, HPC SW Challenge (04).
• www.cfd-online.com:

<table>
<thead>
<tr>
<th>Trilinos 😊</th>
</tr>
</thead>
<tbody>
<tr>
<td>A project led by Sandia to develop an object-oriented software framework for scientific computations. This is an active project which includes several state-of-the-art solvers and lots of other nice things a software engineer writing CFD codes would find useful. Everything is freely available for download once you have registered. Very good!</td>
</tr>
</tbody>
</table>

• Industry Collaborations: Boeing, Goodyear, ExxonMobil, others.
• Linux distros: Debian, Mandriva, Ubuntu, Fedora.
• SciDAC TOPS-2 partner, IAA Algorithms (with ORNL).
• Over 8000 downloads since March 2005.
• Occasional unsolicited external endorsements such as the following two-person exchange on mathforum.org:
  > The consensus seems to be that OO has little, if anything, to offer
  > (except bloat) to numerical computing.
  I would completely disagree. A good example of using OO in numerics is Trilinos: http://software.sandia.gov/trilinos/ |
• Trilinos and related packages are available via LGPL.

• Current release (9.0) is “click release”. Unlimited availability.

• Trilinos Release 10.0: September 2009.
  – CMake is now the supported build system
  – Autotools is no longer supported
  – Alpha release currently available to try CMake build system

• Trilinos Awards:
  – 2004 R&D 100 Award.
  – SC2004 HPC Software Challenge Award.
  – Sandia Team Employee Recognition Award.
  – Lockheed-Martin Nova Award Nominee.

• More information:

• 6th Annual Trilinos User Group Meeting in October 2008 @ SNL
  – talks available for download

• 7th Annual Trilinos User Group Meeting November 3-5, 2009
Dependencies and Reusability

Using externally developed software can be as risk!
- External software can be hard to learn
- External software may not do what you need
- Upgrades of external software can be risky:
  - Breaks in backward compatibility?
  - Regressions in capability?
- External software may not be well supported
- External software may not be support over long term

What can reduce the risk of depending on external software?
- Strong software engineering skill and processes (high quality, low defects, frequent releases)
- Strong organizational relationships
- Regulated backward compatibility and smooth upgrading
- Long term commitment (i.e. 10-30 years) to actively support the software

Trilinos leaders and stakeholders recognize these issues and are committed to continual improvement!
Useful Links


Trilinos mailing lists:  [http://trilinos.sandia.gov/mail_lists.html](http://trilinos.sandia.gov/mail_lists.html)

Trilinos User Group (TUG) meetings: