Trilinos Tutorial

Nico Schlömer (TU Berlin)
for the PRACE summer school 2013, Ostrava.

Slides contributed by Mike Heroux & many Trilinos devs.
Goals of the Tutorial

- Present most packages
- Concept behind Trilinos
- Build, use Trilinos

The enlightenment.
Background/Motivation:

What can Trilinos do for you?
What is Trilinos?

Object-oriented software framework for...

- Solving big complex science & engineering problems
- A Project of Projects.
- More like Lego™ bricks than MATLAB™
Sandia National Labs (US Dept of Energy)

Sandia CSRI
Albuquerque, NM
(505) 845-7695
<table>
<thead>
<tr>
<th>Chris Baker</th>
<th>Jonathan Hu</th>
<th>Alan Williams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ross Bartlett</td>
<td>Joe Kotulski</td>
<td>Michael Wolf</td>
</tr>
<tr>
<td>Pavel Bochev</td>
<td>Rich Lehoucq</td>
<td></td>
</tr>
<tr>
<td>Paul Boggs</td>
<td>Kevin Long</td>
<td></td>
</tr>
<tr>
<td>Erik Boman</td>
<td>Karla Morris</td>
<td></td>
</tr>
<tr>
<td>Lee Buermann</td>
<td>Kurtis Nusbaum</td>
<td></td>
</tr>
<tr>
<td>Cedric Chevalier</td>
<td>Roger Pawlowski</td>
<td></td>
</tr>
<tr>
<td>Todd Coffey</td>
<td>Brent Perschbacher</td>
<td></td>
</tr>
<tr>
<td>Eric Cyr</td>
<td>Eric Phipps</td>
<td></td>
</tr>
<tr>
<td>David Day</td>
<td>Siva Rajamanickam</td>
<td></td>
</tr>
<tr>
<td>Karen Devine</td>
<td>Lee Ann Riesen</td>
<td></td>
</tr>
<tr>
<td>Clark Dohrmann</td>
<td>Marzio Sala</td>
<td></td>
</tr>
<tr>
<td>Kelly Fermoyle</td>
<td>Andrew Salinger</td>
<td></td>
</tr>
<tr>
<td>David Gay</td>
<td>Nico Schlömer</td>
<td></td>
</tr>
<tr>
<td>Jeremie Gaidamour</td>
<td>Chris Siefert</td>
<td></td>
</tr>
<tr>
<td>Mike Heroux</td>
<td>Bill Spotz</td>
<td></td>
</tr>
<tr>
<td>Ulrich Hetmaniuk</td>
<td>Heidi Thornquist</td>
<td></td>
</tr>
<tr>
<td>Mark Hoemmen</td>
<td>Ray Tuminaro</td>
<td></td>
</tr>
<tr>
<td>Russell Hooper</td>
<td>Jim Willenbring</td>
<td></td>
</tr>
<tr>
<td>Vicki Howle</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Past Contributors:*

<table>
<thead>
<tr>
<th>Jason Cross</th>
<th>Michael Gee</th>
<th>Esteban Guillen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob Heaphy</td>
<td>Robert Hoekstra</td>
<td>Kris Kampshoff</td>
</tr>
<tr>
<td>Karla Morris</td>
<td>Ian Karlin</td>
<td>Sarah Knepper</td>
</tr>
<tr>
<td>Kurtis Nusbaum</td>
<td>Marzio Sala</td>
<td>Tammy Kolda</td>
</tr>
<tr>
<td>Roger Pawlowski</td>
<td>Andrew Salinger</td>
<td>Joe Outzen</td>
</tr>
<tr>
<td>Brent Perschbacher</td>
<td>Nico Schlömer</td>
<td>Mike Phenow</td>
</tr>
<tr>
<td>Eric Phipps</td>
<td>Chris Siefert</td>
<td>Paul Sexton</td>
</tr>
<tr>
<td>Siva Rajamanickam</td>
<td>Bill Spotz</td>
<td>Bob Shuttleworth</td>
</tr>
<tr>
<td>Lee Ann Riesen</td>
<td>Heidi Thornquist</td>
<td>Ken Stanley</td>
</tr>
</tbody>
</table>
## Contributions by hour of the day

<table>
<thead>
<tr>
<th>Day</th>
<th>12am</th>
<th>1am</th>
<th>2am</th>
<th>3am</th>
<th>4am</th>
<th>5am</th>
<th>6am</th>
<th>7am</th>
<th>8am</th>
<th>9am</th>
<th>10am</th>
<th>11am</th>
<th>12pm</th>
<th>1pm</th>
<th>2pm</th>
<th>3pm</th>
<th>4pm</th>
<th>5pm</th>
<th>6pm</th>
<th>7pm</th>
<th>8pm</th>
<th>9pm</th>
<th>10pm</th>
<th>11pm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<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>Monday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<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>Tuesday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<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>Wednesday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<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>Thursday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<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>Friday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<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>Saturday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<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>

*libmesh*
## Contributions by hour of the day

<table>
<thead>
<tr>
<th>Day</th>
<th>12a</th>
<th>1a</th>
<th>2a</th>
<th>3a</th>
<th>4a</th>
<th>5a</th>
<th>6a</th>
<th>7a</th>
<th>8a</th>
<th>9a</th>
<th>10a</th>
<th>11a</th>
<th>12p</th>
<th>1p</th>
<th>2p</th>
<th>3p</th>
<th>4p</th>
<th>5p</th>
<th>6p</th>
<th>7p</th>
<th>8p</th>
<th>9p</th>
<th>10p</th>
<th>11p</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<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>Monday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<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>Tuesday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<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>Wednesday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<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>Thursday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<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>Friday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<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>Saturday</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<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>

*Trilinos*
Optimal Kernels to Optimal Solutions:
- Geometry, Meshing
- Discretizations, Load Balancing.
- Scalable Linear, Nonlinear, Eigen, Transient, Optimization, UQ solvers.
- Scalable I/O, GPU, Manycore

- 60 Packages.
- Binary distributions:
  - Cray LIBSCI
  - Debian, Ubuntu
  - Intel (in process)

R&D 100 Winner
- 9000 Registered Users.
- 30,000 Downloads.
- Open Source.

Transforming Computational Analysis To Support High Consequence Decisions

Each stage requires greater performance and error control of prior stages:
Always will need: more accurate and scalable methods.
more sophisticated tools.
Applications

- All kinds of physical simulations:
  - Structural mechanics (statics and dynamics)
  - Circuit simulations (physical models)
  - Electromagnetics, plasmas, and superconductors
  - Combustion and fluid flow (at macro- and nanoscales)

- Coupled / multiphysics models

- Data and graph analysis (2D distributions).
How Trilinos evolved

- Started as linear solvers and distributed objects
- Capabilities grew to satisfy application and research needs

### Numerical math
Convert to models that can be solved on digital computers

### Algorithms
Find faster and more efficient ways to solve numerical models

### Discretizations
- Time domain
- Space domain

### Methods
- Automatic diff.
- Domain dec.
- Mortar methods

### Solvers
- Linear
- Nonlinear
- Eigenvalues
- Optimization

### Core
- Petra
- Utilities
- Interfaces
- Load Balancing

- Discretizations in space and time
- Optimization and sensitivities
- Uncertainty quantification
Target platforms:
Any and all, current and future

- Laptops and workstations
- Clusters and supercomputers
  - Multicore CPU nodes
  - Hybrid CPU / GPU nodes
- Parallel programming environments
  - MPI, OpenMP
  - Intel TBB, Pthreads
  - Thrust, CUDA
  - Combinations of the above
- User “skins”
  - C++ (primary language)
  - C, Fortran, Python
  - Web (Hands-on demo!)
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, and important external packages, will be interoperable, so that any combination of packages and external software (e.g., PETSc, Hypre) 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.

- **TriBITS Lifecycle**: Trilinos will be:
  - **Reliable**: Leading edge hardened, scalable solutions for all apps
  - **Available**: Integrated into every major application at Sandia
  - **Serviceable**: “Self-sustaining”.

**Algorithmic Goals**

**Software Goals**
Unique features of Trilinos

• Huge library of algorithms
  – Linear and nonlinear solvers, preconditioners, …
  – Optimization, transients, sensitivities, uncertainty, …

• Growing support for multicore & hybrid CPU/GPU
  – Built into the new Tpetra linear algebra objects
    • Therefore into iterative solvers with zero effort!
  – Unified intranode programming model
  – Spreading into the whole stack:
    • Multigrid, sparse factorizations, element assembly…

• Growing support for mixed and arbitrary precisions
  – Don’t have to rebuild Trilinos to use it!

• Growing support for huge (> 2B unknowns) problems
Capability Leaders: Layer of Proactive Leadership

- Areas:
  - User Experience (W. Spotz) (May 2012).
  - Scalable I/O: (R. Oldfield) (Nov 2010).
  - Framework, Tools & Interfaces (J. Willenbring).
  - Software Engineering Technologies and Integration (R. Bartlett).
  - Discretizations (P. Bochev).
  - Scalable Linear Algebra (M. Heroux).
  - Linear & Eigen Solvers (J. Hu).
  - Nonlinear, Transient & Optimization Solvers (A. Salinger).

- Each leader provides strategic direction across all Trilinos packages within area.
Trilinos’ software organization
Trilinos is made of packages

- Not a monolithic piece of software
  - Like LEGO™ bricks, not Matlab™

- Each package:
  - Has its own development team and management
  - Makes its own decisions about algorithms, coding style, etc.
  - May or may not depend on other Trilinos packages

- Trilinos is not “indivisible”
  - You don’t need all of Trilinos to get things done
  - Any subset of packages can be combined and distributed
  - Current public release (11.2) contains 54 of the 60+ Trilinos packages

- Trilinos top layer framework
  - Not a large amount of source code: ~1.5%
  - Manages package dependencies
    - Like a GNU/Linux package manager
  - Runs packages’ tests nightly, and on every check-in

- Package model supports multifrontal development
# 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; Discretizations</td>
<td>STK, Intrepid, Pamgen, Sundance, ITAPS, Mesquite</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>Services</strong></td>
<td></td>
</tr>
<tr>
<td>Linear algebra objects</td>
<td>Epetra, Jpetra, Tpetra, Kokkos</td>
</tr>
<tr>
<td>Interfaces</td>
<td>Thyra, Stratimikos, RTOp, FEI,</td>
</tr>
<tr>
<td>Load Balancing</td>
<td>Zoltan, Isorropia, Zoltan2</td>
</tr>
<tr>
<td>“Skins”</td>
<td>PyTrilinos, WebTrilinos, ForTrilinos, Ctrilinos, Optika</td>
</tr>
<tr>
<td>C++ utilities, I/O, thread API</td>
<td>Teuchos, EpetraExt, Kokkos, Triutils, ThreadPool, Phalanx, Trios</td>
</tr>
<tr>
<td><strong>Solvers</strong></td>
<td></td>
</tr>
<tr>
<td>Iterative linear solvers</td>
<td>AztecOO, Belos, Komplex</td>
</tr>
<tr>
<td>Direct sparse linear solvers</td>
<td>Amesos, Amesos2</td>
</tr>
<tr>
<td>Direct dense linear solvers</td>
<td>Epetra, Teuchos, Pliris</td>
</tr>
<tr>
<td>Iterative eigenvalue solvers</td>
<td>Anasazi, Rbgen</td>
</tr>
<tr>
<td>ILU-type preconditioners</td>
<td>AztecOO, IFPACK, Ifpack2</td>
</tr>
<tr>
<td>Multilevel preconditioners</td>
<td>ML, CLAPS, Muelu</td>
</tr>
<tr>
<td>Block preconditioners</td>
<td>Meros, Teko</td>
</tr>
<tr>
<td>Nonlinear system solvers</td>
<td>NOX, LOCA, Piro</td>
</tr>
<tr>
<td>Optimization (SAND)</td>
<td>MOOCHO, Aristos, TriKota, Globipack, Optipack, ROI</td>
</tr>
<tr>
<td>Stochastic PDEs</td>
<td>Stokhos</td>
</tr>
</tbody>
</table>
Although most Trilinos packages have no explicit dependence, often packages must interact with some other packages:
- NOX needs operator, vector and linear solver objects.
- AztecOO needs preconditioner, matrix, operator and vector objects.
- Interoperability is enabled at configure time.
- Trilinos CMake system is vehicle for:
  - Establishing interoperability of Trilinos components…
  - Without compromising individual package autonomy.
  - Trilinos_ENABLE_ALL_OPTIONAL_PACKAGES option

Architecture supports simultaneous development on many fronts.
Software Development and Delivery
Compile-time Polymorphism
Templates and Sanity upon a shifting foundation

Software delivery:
  • Essential Activity

How can we:
  • Implement mixed precision algorithms?
  • Implement generic fine-grain parallelism?
  • Support hybrid CPU/GPU computations?
  • Support extended precision?
  • Explore redundant computations?
  • Prepare for both exascale “swim lanes”?

C++ templates only sane way:
  • Moving to completely templated Trilinos libraries.
  • Other important benefits.
  • A usable stack exists now in Trilinos.

Template Benefits:
  • Compile time polymorphism.
  • True generic programming.
  • No runtime performance hit.
  • Strong typing for mixed precision.
  • Support for extended precision.
  • Many more…

Template Drawbacks:
  − Huge compile-time performance hit:
    • But good use of multicore ;)
    • Eliminated for common data types.
  − Complex notation:
    • Esp. for Fortran & C programmers).
    • Can insulate to some extent.

“Are C++ templates safe? No, but they are good.”
## Solver Software Stack

<table>
<thead>
<tr>
<th>Optimization</th>
<th>Find $u \in \mathbb{R}^n$ that minimizes $g(u)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unconstrained:</td>
<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>
</tr>
<tr>
<td>Constrained:</td>
<td></td>
</tr>
</tbody>
</table>

### Bifurcation Analysis

Given nonlinear operator $F(x, u) \in \mathbb{R}^{n+m}$.

For $F(x, u) = 0$ find space $u \in U \ni \frac{\partial F}{\partial x}$.

### Transient 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 $x(t) \in \mathbb{R}^n$, $t \in [0, T]$.

### Nonlinear Problems

Given nonlinear operator $F(x) \in \mathbb{R}^m \rightarrow \mathbb{R}^n$.

Solve $F(x) = 0$ $x \in \mathbb{R}^n$.

### Linear Problems

**Linear Equations:**

Given Linear Ops (Matrices) $A, B \in \mathbb{R}^{m \times n}$.

Solve $Ax = b$ for $x \in \mathbb{R}^n$.

**Eigen Problems:**

Solve $Av = \lambda Bv$ for (all) $\nu \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$.

### Distributed Linear Algebra

**Matrix/Graph Equations:**

Compute $y = Ax$; $A = A(G)$; $A \in \mathbb{R}^{m \times n}$, $G \in \mathbb{S}^{m \times n}$.

**Vector Problems:**

Compute $y = \alpha x + \beta w$; $\alpha = \langle x, y \rangle$; $x, y \in \mathbb{R}^n$. 

---

### Phase I packages: SPMD, int/double

### Phase II packages: Templatred

**MDOCHO**

**LOCAC**

**Rythmos**

**NOX**

**Anasazi**

**Teuchos**

**Epetra**

**Sandia**

**Teuchos**

**Sacado**

**Teuchos**

**Teuchos**

**Teuchos**

**Teuchos**

---

22
## Solver Software Stack

### Phase I packages: SPMD, int/double

| Optimization      | Find $u \in \mathbb{R}^n$ that minimizes $g(u)$
|                   | Find $x \in \mathbb{R}^m$ and $u \in \mathbb{R}^n$ that minimizes $g(x, u)$ s.t. $f(x, u) = 0$

### Bifurcation Analysis

- Given nonlinear operator $F(x, u) \in \mathbb{R}^{n+m}$
- For $F(x, u) = 0$ find space $u \in U \ni \frac{\partial F}{\partial x}$

### Transient Problems

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

### Nonlinear Problems

- Given nonlinear operator $F(x) \in \mathbb{R}^m \rightarrow \mathbb{R}^n$
- Solve $F(x) = 0, x \in \mathbb{R}^n$

### Linear Problems

- **Linear Equations:**
  - Given Linear Ops (Matrices) $A, B \in \mathbb{R}^{m \times n}$
  - Solve $Ax = b$ for $x \in \mathbb{R}^n$

- **Eigen Problems:**
  - Solve $Av = \lambda Bv$ for (all) $v \in \mathbb{R}^n$, $\lambda \in \mathbb{C}$

### Distributed Linear Algebra

- **Matrix/Graph Equations:**
  - Compute $y = Ax$; $A = A(G)$; $A \in \mathbb{R}^{m \times n}, G \in \mathbb{R}^{m \times n}$

- **Vector Problems:**
  - Compute $y = \alpha x + \beta w$; $\alpha = \langle x, y \rangle$; $x, y \in \mathbb{R}^n$
Whirlwind Tour of Packages

Core Utilities
Discretizations Methods Solvers
Discretizations
Which packages?

FEI, Panzer
user-defined assignment and management of global degrees of freedom; assembly of local PDE discretization data into distributed linear systems; etc.

Phalanx
decomposition of complex PDE systems into a number of elementary user-defined expressions; efficient management of expression dependencies; hooks to embedded tools, etc.

Intrepid
local (cell-based) FE/FV/FD basis definition; numerical integration; cell geometry; etc.

Shards
definition of cell topology

Developers: Pavel Bochev, Denis Ridzal, Alan Williams, Roger Pawlowski, Eric, Cyr, others.
Shards
- a suite of common tools for topological data that facilitate interoperability between PDE software
- cell definitions (e.g., triangle, hexahedron, etc.)
- methods to manage and access information about cell topologies:
  - (1) query adjacencies of subcells
  - (2) find subcell permutation w. r. to global cell
  - (3) create user-defined custom cell topologies
Discretization Capability Area
Which packages?

**Intrepid**
- physics-compatible cell-local PDE discretizations
- streamlined access to finite element, finite volume and finite difference methods
- support for a wide range of cell topologies
- compatible finite element spaces of arbitrary degree for $H(\text{grad})$, $H(\text{curl})$, $H(\text{div})$ and $L^2$ spaces
- Lagrange-interpolating and modal FE bases
- prototype for polyhedral FE bases
- prototype for control-volume (CV) FEM
- numerical integration: spatial and stochastic
- cell geometry tools: volumes, normals, tangents, reference-to-physical maps

**Shards**
- cell topology

**Intrepid**
- local PDE discretization

**Phalanx**
- PDE expression trees

**FEI, Panzer**
- global DOFs; linear systems
Discretization Capability Area
Which packages?

**Phalanx**
- cell-local field (variable, data) evaluation kernel specifically designed for general PDE solvers
- decomposition of complex PDE systems into a number of elementary user-defined expressions
- management of expression dependencies
- enables rapid development of large PDEs
- user-defined data types and evaluation types offer unprecedented flexibility for direct integration with user applications
- they also enable embedded technology such as automatic differentiation for sensitivity analysis, optimization and uncertainty quantification

**Shards**
- cell topology

**Intrepid**
- local PDE discretization

**Phalanx**
- PDE expression trees

**FEI, Panzer**
- global DOFs; linear systems
Discretization Capability Area
Which packages?

FEI, Panzer
- user-defined assignment and management of global degrees of freedom (DOFs)
- assembly of cell-local PDE discretization data into global, distributed linear systems
- insulate PDE application codes from linear-algebra issues such as sparse matrix storage and mappings of DOFs to distributed linear equations
- support multi-physics problems, allowing for arbitrarily complicated PDE discretizations with multiple DOFs per subcell (edge-based, face-based, node-based and mixed)
Rythmos

- Suite of time integration (discretization) methods
  - Includes: backward Euler, forward Euler, explicit Runge-Kutta, and implicit BDF at this time.
  - Native support for operator split methods.
  - Highly modular.
  - Forward sensitivity computations will be included in the first release with adjoint sensitivities coming in near future.

Developers: Curt Ober, Todd Coffey, Roscoe Bartlett
Whirlwind Tour of Packages

Discretizations  Methods  Core  Solvers
Sacado: Automatic Differentiation

- Efficient OO based AD tools optimized for element-level computations

- Applies AD at “element”-level computation
  - “Element” means finite element, finite volume, network device,…

- Template application’s element-computation code
  - Developers only need to maintain one templated code base

- Provides three forms of AD
  - **Forward Mode:** \((x, V) \rightarrow (f, \frac{\partial f}{\partial x} V)\)
    - Propagate derivatives of intermediate variables w.r.t. independent variables forward
    - Directional derivatives, tangent vectors, square Jacobians, \(\frac{\partial f}{\partial x}\) when \(m \geq n\).

  - **Reverse Mode:** \((x, w) \rightarrow (f, w^T \frac{\partial f}{\partial x})\)
    - Propagate derivatives of dependent variables w.r.t. independent variables backwards
    - Gradients, Jacobian-transpose products (adjoints), when \(n > m\).

  - **Taylor polynomial mode:**
    \[
    x(t) = \sum_{k=0}^{d} x_k t^k \rightarrow \sum_{k=0}^{d} f_k t^k = f(x(t)) + O(t^{d+1}), \quad f_k = \frac{1}{k!} \frac{d^k}{dt^k} f(x(t))
    \]

Developers: Eric Phipps, Carter Edwards (UQ data types)
Whirlwind Tour of Packages

Discretizations  Methods  Core  Solvers
Teuchos

- Portable utility package of commonly useful tools:
  - ParameterList class: key/value pair database, recursive capabilities.
  - LAPACK, BLAS wrappers (templated on ordinal and scalar type).
  - Dense matrix and vector classes (compatible with BLAS/LAPACK).
  - FLOP counters, timers.
  - Ordinal, Scalar Traits support: Definition of ‘zero’, ‘one’, etc.
  - Reference counted pointers / arrays, and more…

- Takes advantage of advanced features of C++:
  - Templates
  - Standard Template Library (STL)

- Teuchos::ParameterList:
  - Allows easy control of solver parameters.
  - XML format input/output.

Developers: Chris Baker, Roscoe Barlett, Heidi Thornquist, Mike Heroux, Paul Sexton, Kris Kampshoff, Chris Baker, Mark Hoemmen, many others
Trilinos Common Language: Petra

- Petra provides a “common language” for distributed linear algebra objects (operator, matrix, vector)

- Petra1 provides distributed matrix and vector services.
- Exists in basic form as an object model:
  - Describes basic user and support classes in UML, independent of language/implementation.
  - Describes objects and relationships to build and use matrices, vectors and graphs.
  - Has 2 implementations under development.

Petra is Greek for “foundation”.
Perform redistribution of distributed objects:
- Parallel permutations.
- "Ghosting" of values for local computations.
- Collection of partial results from remote processors.

Base Class for All Distributed Objects:
- Performs all communication.
- Requires Check, Pack, Unpack methods from derived class.

Petra Object Model

Graph class for structure-only computations:
- Reusable matrix structure.
- Pattern-based preconditioners.
- Pattern-based load balancing tools.
- Redistribution of matrices, vectors, etc...

Basic sparse matrix class:
- Flexible construction process.
- Arbitrary entry placement on parallel machine.

Describes layout of distributed objects:
- Vectors: Number of vector entries on each processor and global ID.
- Matrices/graphs: Rows/Columns managed by a processor.
- Called "Maps" in Epetra.

Dense Distributed Vector and Matrices:
- Simple local data structure.
- BLAS-able, LAPACK-able.
- Ghostable, redistributable.
- RTOp-able.
Petra Implementations

- **Epetra (Essential Petra):**
  - Current production version.
  - Restricted to real, double precision arithmetic.
  - Uses stable core subset of C++ (circa 2000).
  - Interfaces accessible to C and Fortran users.

- **Tpetra (Templated Petra):**
  - Next generation C++ version.
  - Templated scalar and ordinal fields.
  - Uses namespaces, and STL: Improved usability/efficiency.
  - Builds on top of Kokkos manycore node library.
EpetaExt: Extensions to Epeta

- Library of useful classes not needed by everyone
- Most classes are types of “transforms”.
- Examples:
  - Graph/matrix view extraction.
  - Epeta/Zoltan interface.
  - Explicit sparse transpose.
  - Singleton removal filter, static condensation filter.
  - Overlapped graph constructor, graph colorings.
  - Permutations.
  - Sparse matrix-matrix multiply.
  - MATLAB, MatrixMarket I/O functions.
- Most classes are small, useful, but non-trivial to write.
- Migrating to 64-bit ints

Developer: Robert Hoekstra, Alan Williams, Mike Heroux, Chetan Jhurani
Kokkos: Node-level Data Classes

- Manycore/Accelerator data structures & kernels
- Epetra is MPI-only, Tpetra is MPI+[X+Y].
- Kokkos Arrays: Details tomorrow.
  - Simple multi-dimensional arrays.
  - User specifies dimensions and size. Library handles all else.
  - Very good general performance.
- Pretty-good-kernel (PGK) library:
  - Node-level threaded (X) and vector (Y) sparse and dense kernels.
  - Plug replaceable with vendor-optimized libraries.
- Implement Petra Object Model at Node level:
  - Comm, Map/Perm, Vector/Multivector, RowMatrix, Operator.

Developer: Mike Heroux, Carter Edwards, Christian Trott, Siva Rajamanickam, etc.
Zoltan

- Data Services for Dynamic Applications
  - Dynamic load balancing
  - Graph coloring
  - Data migration
  - Matrix ordering

- Partitioners:

  Geometric (coordinate-based) methods:
  - Recursive Coordinate Bisection (Berger, Bokhari)
  - Recursive Inertial Bisection (Taylor, Nour-Omid)
  - Space Filling Curves (Peano, Hilbert)
  - Refinement-tree Partitioning (Mitchell)

  Hypergraph and graph (connectivity-based) methods:
  - Hypergraph Repartitioning PaToH (Catalyurek)
  - Zoltan Hypergraph Partitioning
  - ParMETIS (U. Minnesota)
  - Jostle (U. Greenwich)

Developers: Karen Devine, Eric Boman, Siva R., LeAnn Riesen
Thyra

- High-performance, abstract interfaces for linear algebra
- Offers flexibility through abstractions to algorithm developers
- Linear solvers (Direct, Iterative, Preconditioners)
  - Abstraction of basic vector/matrix operations (dot, axpy, mv).
  - Can use any concrete linear algebra library (Epetra, PETSc, BLAS).
- Nonlinear solvers (Newton, etc.)
  - Abstraction of linear solve (solve $Ax=b$).
  - Can use any concrete linear solver library:
    - AztecOO, Belos, ML, PETSc, LAPACK
- Transient/DAE solvers (implicit)
  - Abstraction of nonlinear solve.
  - … and so on.

Developers: Roscoe Bartlett, Kevin Long
"Skins"

- PyTrilinos provides Python access to Trilinos packages
- Uses SWIG to generate bindings.
- Epetra, AztecOO, IFPACK, ML, NOX, LOCA, Amesos and NewPackage are supported.

Developer: Bill Spotz

- CTrilinos: C wrapper (mostly to support ForTrilinos).
- ForTrilinos: OO Fortran interfaces.

Developers: Nicole Lemaster, Damian Rouson

- WebTrilinos: Web interface to Trilinos
- Generate test problems or read from file.
- Generate C++ or Python code fragments and click-run.
- Hand modify code fragments and re-run.

Developers: Ray Tuminaro, Jonathan Hu, Marzio Sala, Jim Willenbring
Whirlwind Tour of Packages

Discretizations       Methods       Core       Solvers
Interface to direct solvers for distributed sparse linear systems (KLU, UMFPACK, SuperLU, MUMPS, ScaLAPACK)

Challenges:
- No single solver dominates
- Different interfaces and data formats, serial and parallel
- Interface often changes between revisions

Amesos offers:
- A single, clear, consistent interface, to various packages
- Common look-and-feel for all classes
- Separation from specific solver details
- Use serial and distributed solvers; Amesos takes care of data redistribution
- Native solvers: KLU and Paraklete

Developers: Ken Stanley, Marzio Sala, Tim Davis
Amesos2/KLU2

- Second-generation sparse direct solvers package

- Unified interface to multiple solvers, just like Amesos
- KLU2: Default direct solver.

- Amesos2 features:
  - Supports matrices of arbitrary scalar and index types
  - Path to multicore CPU and hybrid CPU/GPU solvers
  - Thread safe: multiple solvers can coexist on the same node
    - Supports new intranode hybrid direct / iterative solver ShyLU
  - Abstraction from specific sparse matrix representation
    - Supports Epetra and Tpetra
    - Extensible to other matrix types

- Still maturing.

Developers: Eric Bavier, Erik Boman, and Siva Rajamanickam
AztecOO

- Krylov subspace solvers: CG, GMRES, Bi-CGSTAB,…
- Incomplete factorization preconditioners

- Aztec is the workhorse solver at Sandia:
  - Extracted from the MPSalsa reacting flow code.
  - Installed in dozens of Sandia apps.
  - 1900+ external licenses.

- AztecOO improves on Aztec by:
  - Using Epetra objects for defining matrix and RHS.
  - Providing more preconditioners/scalings.
  - Using C++ class design to enable more sophisticated use.

- AztecOO interfaces allows:
  - Continued use of Aztec for functionality.
  - Introduction of new solver capabilities outside of Aztec.

Developers: Mike Heroux, Alan Williams, Ray Tuminaro
Belos

- Next-generation linear solver library, written in templated C++.

- Provide a generic framework for developing iterative algorithms for solving large-scale, linear problems.

- Algorithm implementation is accomplished through the use of traits classes and abstract base classes:
  - Operator-vector products: Belos::MultiVecTraits, Belos::OperatorTraits
  - Orthogonalization: Belos::OrthoManager, Belos::MatOrthoManager
  - Status tests: Belos::StatusTest, Belos::StatusTestResNorm
  - Iteration kernels: Belos::Iteration
  - Linear solver managers: Belos::SolverManager

- AztecOO provides solvers for \( Ax = b \), what about solvers for:
  - Simultaneously solved systems w/ multiple-RHS: \( AX = B \)
  - Sequentially solved systems w/ multiple-RHS: \( AX_i = B_i, i=1,\ldots,t \)
  - Sequences of multiple-RHS systems: \( A_iX_i = B_i, i=1,\ldots,t \)

- Many advanced methods for these types of linear systems
  - Block methods: block GMRES [Vital], block CG/BICG [O'Leary]
  - “Seed” solvers: hybrid GMRES [Nachtigal, et al.]
  - Restarting techniques, orthogonalization techniques, …

Developers: Heidi Thornquist, Mike Heroux, Mark Hoemmen, Mike Parks, Rich Lehoucq
IFPACK: Algebraic Preconditioners

- Overlapping Schwarz preconditioners with incomplete factorizations, block relaxations, block direct solves.
- Accept user matrix via abstract matrix interface (Epetra versions).
- Uses Epetra for basic matrix/vector calculations.
- Supports simple perturbation stabilizations and condition estimation.
- Separates graph construction from factorization, improves performance substantially.
- Compatible with AztecOO, ML, Amesos. Can be used by NOX and ML.

Developers: Marzio Sala, Mike Heroux, Siva Rajamanickam, Alan Williams
Ifpack2

- Second-generation IFPACK

- Highly optimized ILUT (60x faster than IFPACK’s!)
- Computed factors fully exploit multicore CPU / GPU
  - Via Tpetra
- Path to hybrid-parallel factorizations
- Arbitrary precision and complex arithmetic support

Developers: Mike Heroux, Siva Rajamanickam, Alan Williams, Michael Wolf
Multi-level Preconditioners

- Smoothed aggregation, multigrid and domain decomposition preconditioning package

- Critical technology for scalable performance of some key apps.

- ML compatible with other Trilinos packages:
  - Accepts user data as Epetra_RowMatrix object (abstract interface). Any implementation of Epetra_RowMatrix works.
  - Implements the Epetra_Operator interface. Allows ML preconditioners to be used with AztecOO, Belos, Anasazi.

- Can also be used completely independent of other Trilinos packages.

- Muelu: Next generation ML (talked about tomorrow).

Developers: Ray Tuminaro, Jeremie Gaidamour, Jonathan Hu, Marzio Sala, Chris Siefert
Anasazi

- Next-generation eigensolver library, written in templated C++.
- Provide a generic framework for developing iterative algorithms for solving large-scale eigenproblems.
- Algorithm implementation is accomplished through the use of traits classes and abstract base classes:
  - Operator-vector products: Anasazi::MultiVecTraits, Anasazi::OperatorTraits
  - Orthogonalization: Anasazi::OrthoManager, Anasazi::MatOrthoManager
  - Status tests: Anasazi::StatusTest, Anasazi::StatusTestResNorm
  - Iteration kernels: Anasazi::Eigensolver
  - Eigensolver managers: Anasazi::SolverManager
  - Eigenproblem: Anasazi::Eigenproblem
  - Sort managers: Anasazi::SortManager
- Currently has solver managers for three eigensolvers:
  - Block Krylov-Schur
  - Block Davidson
  - LOBPCG
- Can solve:
  - standard and generalized eigenproblems
  - Hermitian and non-Hermitian eigenproblems
  - real or complex-valued eigenproblems

Developers: Heidi Thornquist, Mike Heroux, Chris Baker, Rich Lehoucq, Ulrich Hetmaniuk
NOX: Nonlinear Solvers

- Suite of nonlinear solution methods

**Broyden’s Method**
\[ M_B = f(x_c) + B_c d \]

**Newton’s Method**
\[ M_N = f(x_c) + J_c d \]

**Tensor Method**
\[ M_T = f(x_c) + J_c d + \frac{1}{2} T_c d d \]

**Globalizations**
- Line Search
  - Interval Halving
  - Quadratic
  - Cubic
- Trust Region
  - Dogleg
  - Inexact Dogleg
  - Moré-Thuente

**Jacobian Estimation**
- Graph Coloring
- Finite Difference
- Jacobian-Free Newton-Krylov

**Implementation**
- Parallel
- OO-C++
- Independent of the linear algebra package!

http://trilinos.sandia.gov/packages/nox

**Developers:** Tammy Kolda, Roger Pawlowski
Library of continuation algorithms

Provides
- Zero order continuation
- First order continuation
- Arc length continuation
- Multi-parameter continuation (via Henderson's MF Library)
- Turning point continuation
- Pitchfork bifurcation continuation
- Hopf bifurcation continuation
- Phase transition continuation
- Eigenvalue approximation (via ARPACK or Anasazi)

Developers: Andy Salinger, Eric Phipps
MOOCHO & Aristos

- **MOOCHO**: Multifunctional Object-Oriented arCHitecture for Optimization
  - Large-scale invasive simultaneous analysis and design (SAND) using reduced space SQP methods.

  **Developer: Roscoe Bartlett**

- **Aristos**: Optimization of large-scale design spaces
  - Invasive optimization approach based on full-space SQP methods

  **ROL**: New package with combined capabilities solves.

  **Developer: Denis Ridzal**
High-level Integration and Coupling of Trilinos Solver Capabilities:

Putting the Pieces Together
Classic vs. New Trilinos Solver Stacks

**Milestone: Successful integration of new TRILINOS solver stack into SIERRA low-Mach aerodynamic capability**

Comparison of current and new Trilinos solver stack

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Current solver stack</th>
<th>New solver stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributed linear algebra</td>
<td>Epetra</td>
<td>Tpetra</td>
</tr>
<tr>
<td>Iterative linear solvers</td>
<td>AztecOO</td>
<td>Belos</td>
</tr>
<tr>
<td>Incomplete factorizations</td>
<td>AztecOO, Ifpack</td>
<td>Ifpack2</td>
</tr>
<tr>
<td>Algebraic multigrid</td>
<td>ML</td>
<td>MueLu</td>
</tr>
<tr>
<td>Partition &amp; load balance</td>
<td>Zoltan</td>
<td>Zoltan2</td>
</tr>
<tr>
<td>Direct solvers interface</td>
<td>Ameses</td>
<td>Ameses2</td>
</tr>
</tbody>
</table>

New Trilinos solver stack (Tpetra, Belos, MueLu, Ifpack2, Zoltan2 and Ameses2) has been integrated into Sierra/Conchas Low Mach and Sierra/STK_nalu code base
Abstract Numerical Algorithms

An **abstract numerical algorithm** (ANA) is a numerical algorithm that can be expressed solely in terms of vectors, vector spaces, and linear operators.

**Example Linear ANA (LANA) : Linear Conjugate Gradients**

Given:
- \( A \in \mathcal{X} \rightarrow \mathcal{X} : \) s.p.d. linear operator
- \( b \in \mathcal{X} : \) right hand side vector

Find vector \( x \in \mathcal{X} \) that solves \( Ax = b \)

**Linear Conjugate Gradient Algorithm**

- ANAs can be very mathematically sophisticated!
- ANAs can be extremely reusable!

\[
\begin{align*}
\text{Compute } r^{(0)} &= b - Ax^{(0)} \text{ for the initial guess } x^{(0)}. \\
\text{for } i = 1, 2, \ldots \\
\rho_{i-1} &= \langle r^{(i-1)}, r^{(i-1)} \rangle \\
\beta_{i-1} &= \frac{\rho_{i-1}/\rho_{i-2}}{\beta_0 = 0} \\
p^{(i)} &= r^{(i-1)} + \beta_{i-1}p^{(i-1)} \quad (p^{(1)} = r^{(1)}) \\
q^{(i)} &= Ap^{(i)} \\
\gamma_i &= \langle p^{(i)}, q^{(i)} \rangle \\
\alpha_i &= \frac{\rho_{i-1}/\gamma_i}{\beta_0 = 0} \\
x^{(i)} &= x^{(i-1)} + \alpha_ip^{(i)} \\
r^{(i)} &= r^{(i-1)} - \alpha_iq^{(i)} \\
\text{check convergence; continue if necessary}
\end{align*}
\]

**Types of operations**  **Types of objects**

- linear operator applications  
  - Linear Operators
    - \( A \)
  - Vectors
    - \( r, x, p, q \)
  - Scalars
    - \( \rho, \beta, \gamma, \alpha \)
  - Vector spaces?
    - \( \mathcal{X} \)
• Stratimikos created Greek words "stratigiki" (strategy) and "grammikos" (linear)
• Defines class Thyra::DefaultLinearSolverBuilder.
• Provides common access to:
  • Linear Solvers: Amesos, AztecOO, Belos, …
  • Preconditioners: Ifpack, ML, …
• Reads in options through a parameter list (read from XML?)
• Accepts any linear system objects that provide
  • Epetra_Operator / Epetra_RowMatrix view of the matrix
  • SPMD vector views for the RHS and LHS (e.g. Epetra_[Multi]Vector objects)
• Provides uniform access to linear solver options that can be leveraged across multiple applications and algorithms

Key Points
• Stratimikos is an important building block for creating more sophisticated linear solver capabilities!
Stratimikos Parameter List and Sublists

<ParameterList name="Stratimikos">
  <Parameter name="Linear Solver Type" type="string" value="AztecOO"/>
  <ParameterList name="Preconditioner Types">
    <ParameterList name="Ifpack">
      <Parameter name="Prec Type" type="string" value="ILU"/>
      <Parameter name="Overlap" type="int" value="0"/>
    </ParameterList>
    <ParameterList name="ML">
    </ParameterList>
  </ParameterList>
  <ParameterList name="Linear Solver Types">
    <ParameterList name="Amesos">
      <Parameter name="Solver Type" type="string" value="Klu"/>
      <ParameterList name="Amesos Settings">
        <Parameter name="MatrixProperty" type="string" value="general"/>
        ...
      </ParameterList>
      <ParameterList name="Mumps">
      </ParameterList>
      <ParameterList name="Superludist">
      </ParameterList>
    </ParameterList>
    <ParameterList name="AztecOO">
      <ParameterList name="Forward Solve">
        <Parameter name="Max Iterations" type="int" value="400"/>
        <Parameter name="Tolerance" type="double" value="1e-06"/>
        <ParameterList name="AztecOO Settings">
          <Parameter name="Aztec Solver" type="string" value="GMRES"/>
          ...
        </ParameterList>
      </ParameterList>
      <ParameterList name="Belos">
      </ParameterList>
      ...
    </ParameterList>
  </ParameterList>
  <ParameterList name="Preconditioner Types">
    <ParameterList name="Ifpack">
      <Parameter name="Prec Type" type="string" value="ILU"/>
      <Parameter name="Overlap" type="int" value="0"/>
    </ParameterList>
    <ParameterList name="ML">
    </ParameterList>
  </ParameterList>
</ParameterList>

Every parameter and sublist is handled by Thyra code and is fully validated!
 Libraries used by Trilinos

Optional:

BLAS, LAPACK, MPI, MKL, yaml-cpp, Peano, CUDA, CUSPARSE, Thrust, Cusp, TBB, Pthread, HWLOC, BinUtils, ARPREC, QD, MPI, Boost, Scotch, OVIS, gpcd, METIS, ParMETIS, LibTopoMap, PaToH, CppUnit, ADOLC, ADIC, TVMET, MF, ExodusII, Nemesis, XDMF, Netcdf, y12m, SuperLUDist, SuperLUMT, SuperLU, Zlib, UMFPACK, MA28, AMD, Csparse, HYPRE, PETSC, BLACS, SCALAPACK, MUMPS, PARDISO_MKL, PARDISO, Oski, TAUCS, ForUQTK, Dakota, HIPS, HDF5, MATLAB, CASK, SPARSKIT, Qt, gtest, OpenNURBS, Portals, CrayPortals, Gemini, InfiniBand, BGPDCMF, Pablo, HPCToolkit, Pnetcdf, Clp, GLPK, qpOASES, Matio, PAPI, Eigen
Trilinos Integration into an Application

Where to start?

http://trilinos.sandia.gov
“How do I…?”

- Build Trilinos?

- Build my application with Trilinos?

- Learn about common Trilinos programming idioms?

- Download / find an installation of Trilinos?

- Find documentation and help?
Compiling Trilinos

Fetch from trilinos.sandia.gov

$ ls
trilinos-11.2.3-Source
$ mkdir build
$ cd build
$ cmake ../trilinos-11.2.3-Source/
[…]
($ ctest)
$ make install
If you are using Makefiles:
Makefile.export system

If you are using CMake:
- CMake FIND_PACKAGE
Using CMake to build with Trilinos

- CMake: Cross-platform build system
  - Similar function as the GNU Autotools
- Trilinos uses CMake to build
- You don’t have to use CMake to build with Trilinos
- But if you do:
  - FIND_PACKAGE(Trilinos …)
  - Example CMake script in hands-on demo
- I find this much easier than hand-writing Makefiles
Why build systems are important

- Trilinos has LOTS of packages
- As package dependencies (especially optional ones) are introduced, more maintenance is required by the top-level packages:

![Diagram showing dependencies between packages]

NOX either must:
- Account for the new libraries in its configure script (unscalable)
- Depend on direct dependent packages to supply them through export Makefiles
CMAKE_MINIMUM_REQUIRED(VERSION 2.8.8)
PROJECT(MyProject CXX)

FIND_PACKAGE(Trilinos REQUIRED)
INCLUDE_DIRECTORIES(SYSTEM ${Trilinos_INCLUDE_DIRS})

SET(MY_EXECUTABLE "myexec")

ADD_EXECUTABLE(${MY_EXECUTABLE} "main.cpp")
TARGET_LINK_LIBRARIES(${MY_EXECUTABLE} ${Trilinos_LIBRARIES})
TriBITS: Trilinos/Tribal Build, Integrate, Test System

- Based on Kitware open-source toolset CMake, CTest, and Cdash developed during the adoption by Trilinos but later extended for use in CASL-related VERA, SCALE and other projects.

- Built-in CMake-based support for partitioning a project into ‘Packages’ with carefully regulated dependencies with numerous features including:
  - Automatic enabling of upstream and downstream packages (critical for large projects like Trilinos, SCALE, and CASL)
  - Integrated MPI and CUDA support
  - Integrated TPL support (coordinate common TPLs across unrelated packages, common behavior for user configuration, etc.)
  - Removal of a lot of boiler-plate CMake code for creating libraries, executables, copying files, etc. …

- Powerful TRIBITS_ADD_[ADVANCED]_TEST(…) CMake wrapper functions to create advanced test

- Integrated support for add-on ‘Repositories’ with add-on ‘Packages’

- TribitsCTestDriverCore.cmake testing driver:
  - Partitioned package-by-package output to CDash and reporting on a package-by-package basis
  - Failed packages don’t propagate errors to downstream packages
  - Integrated coverage and memory testing (showing up on CDash)
  - Nightly and continuous integration (CI) test driver.

- Pre-push synchronous CI testing with the Python checkin-test.py script

- Automated almost continuous integration support using checkin-test.py script

- Plus: TribitsDashboardDriver system, download-cmake.py and numerous other tools
TriBITS: Meta Project, Repository, Packages

Current state of TriBITS
- Flexible aggregation of Packages from different Repositories into meta Projects
- TriBITS directory can be snapshoted out of Trilinos into stand-alone projects (independent of Trilinos)
- Being used by CASL VERA software, and several other CASL-related software packages
- egdist: Managing multiple repositories

Future changes/additions to TriBITS
- Combining concepts of TPLs and Packages to allow flexible configuration and building
- TribitsExampleProject
- Trilinos-independent TriBITS documentation
- Provide open access to TribitsExampleProject and therefore TriBITS
Continuous Integration with CDash

![CDash Dashboard](image-url)
Trilinos Availability / Information

- Trilinos and related packages are available via LGPL or BSD.
- Current release (11.2) is “click release”. Unlimited availability.

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

- More information:

- Annual Forums:
  - Annual Trilinos User Group Meeting in November @ SNL
    - talks and video available for download
  - Spring Developer Meeting, May @ SNL
  - European meeting.
  - SC’XY Linear Algebra Tutorial (with Dongarra, Demmel, Kurzczak).
More detailed package info
Epetra Package

Linear Algebra Package

http://trilinos.sandia.gov/packages/epetra/
Typical Flow of Epetra Object Construction

- Construct Comm
  - Any number of Comm objects can exist.
  - Comms can be nested (e.g., serial within MPI).

- Construct Map
  - Maps describe parallel layout.
  - Maps typically associated with more than one comp object.
  - Two maps (source and target) define an export/import object.

- Construct $x$
- Construct $b$
- Construct $A$
  - Computational objects.
  - Compatibility assured via common map.
int main(int argc, char *argv[]) {
    // Initialize MPI, MpiComm
    MPI_Init(&argc,&argv);
    Epetra_MpiComm Comm( MPI_COMM_WORLD );

    // ***** Create x and b vectors *****
    Epetra_Vector x(Map);
    Epetra_Vector b(Map);
    // Fill RHS with random #s
    b.Random();

    // ***** Create an Epetra_Matrix tridiag(-1,2,-1) *****
    Epetra_CrsMatrix A(Copy, Map, 3);
    double negOne = -1.0; double posTwo = 2.0;
    for (int i=0; i<NumMyElements; i++) {
        int GlobalRow = A.GRID(i);
        int RowLess1 = GlobalRow - 1;
        int RowPlus1 = GlobalRow + 1;
        if (RowLess1!=-1)
            A.InsertGlobalValues(GlobalRow, 1, &negOne, &RowLess1);
        if (RowPlus1!=NumGlobalElements)
            A.InsertGlobalValues(GlobalRow, 1, &negOne, &RowPlus1);
        A.InsertGlobalValues(GlobalRow, 1, &posTwo, &GlobalRow);
    }
    A.FillComplete(); // Transform from GIDs to LIDs

    // *** Map puts same number of equations on each pe ***
    int NumMyElements = 1000;
    Epetra_Map Map(-1, NumMyElements, 0, Comm);
    int NumGlobalElements = Map.NumGlobalElements();

    // *** Create/define AztecOO instance, solve **
    AztecOO solver(problem);
    solver.SetAztecOption(AZ_precond, AZ_Jacobi);
    solver.Iterate(1000, 1.0E-8);

    // ***** Create Linear Problem *****
    Epetra_LinearProblem problem(&A, &x, &b);

    // ***** Report results, finish ********************
    cout << "Solver performed " << solver.NumIters() << " iterations."
         << " Norm of true residual = " 
         << solver.TrueResidual() 
         << endl;
    MPI_Finalize();
    return 0;
}
Details about Epetra Maps

- Note: Focus on Maps (not BlockMaps).
- Getting beyond standard use case…

- Note: All of the concepts presented here for Epetra carry over to Tpetra!
1-to-1 Maps

- **1-to-1 map (defn):** A map is 1-to-1 if each GID appears only once in the map (and is therefore associated with only a single processor).
- Certain operations in parallel data repartitioning require 1-to-1 maps. Specifically:
  - The source map of an import must be 1-to-1.
  - The target map of an export must be 1-to-1.
  - The domain map of a 2D object must be 1-to-1.
  - The range map of a 2D object must be 1-to-1.
2D Objects: Four Maps

- **Epetra 2D objects:**
  - CrsMatrix, FECrsMatrix
  - CrsGraph
  - VbrMatrix, FEVbrMatrix

- **Have four maps:**
  - **RowMap**: On each processor, the GIDs of the **rows** that processor will “manage”.
  - **ColMap**: On each processor, the GIDs of the **columns** that processor will “manage”.
  - **DomainMap**: The layout of domain objects (the \( x \) vector/multivector in \( y=Ax \)).
  - **RangeMap**: The layout of range objects (the \( y \) vector/multivector in \( y=Ax \)).

Typically a 1-to-1 map

Typically NOT a 1-to-1 map

Must be 1-to-1 maps!!!
Sample Problem

\[
\begin{bmatrix}
  y \\
  y_1 \\
  y_2 \\
  y_3 \\
\end{bmatrix}
= 
\begin{bmatrix}
  2 & -1 & 0 \\
  -1 & 2 & -1 \\
  0 & -1 & 2 \\
\end{bmatrix}
\begin{bmatrix}
  x \\
  x_1 \\
  x_2 \\
  x_3 \\
\end{bmatrix}
\]
Case 1: Standard Approach

- First 2 rows of $A$, elements of $y$ and elements of $x$, kept on PE 0.
- Last row of $A$, element of $y$ and element of $x$, kept on PE 1.

<table>
<thead>
<tr>
<th>PE 0 Contents</th>
<th>PE 1 Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y = \begin{bmatrix} y_1 \ y_2 \end{bmatrix}$, $A = \begin{bmatrix} 2 &amp; -1 &amp; 0 \ -1 &amp; 2 &amp; -1 \end{bmatrix}$, $x = \begin{bmatrix} x_1 \ x_2 \end{bmatrix}$</td>
<td>$y = \begin{bmatrix} y_3 \end{bmatrix}$, $A = \begin{bmatrix} 0 &amp; -1 &amp; 2 \end{bmatrix}$, $x = \begin{bmatrix} x_3 \end{bmatrix}$</td>
</tr>
</tbody>
</table>

- RowMap = \{0, 1\}
- ColMap = \{0, 1, 2\}
- DomainMap = \{0, 1\}
- RangeMap = \{0, 1\}

- RowMap = \{2\}
- ColMap = \{1, 2\}
- DomainMap = \{2\}
- RangeMap = \{2\}

Notes:
- Rows are wholly owned.
- RowMap=DomainMap=RangeMap (all 1-to-1).
- ColMap is NOT 1-to-1.
- Call to FillComplete: $A$.FillComplete();
### Case 2: Twist 1

- First 2 rows of \( A \), first elem of \( y \) and last 2 elems of \( x \), kept on PE 0.
- Last row of \( A \), last 2 element of \( y \) and first element of \( x \), kept on PE 1.

#### PE 0 Contents

\[
y = [y_1],...A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \end{bmatrix},...x = \begin{bmatrix} x_2 \\ x_3 \end{bmatrix}
\]

- RowMap = \{0, 1\}
- ColMap = \{0, 1, 2\}
- DomainMap = \{1, 2\}
- RangeMap = \{0\}

#### PE 1 Contents

\[
y = \begin{bmatrix} y_2 \\ y_3 \end{bmatrix},...A = \begin{bmatrix} 0 & -1 & 2 \end{bmatrix},...x = \begin{bmatrix} x_1 \end{bmatrix}
\]

- RowMap = \{2\}
- ColMap = \{1, 2\}
- DomainMap = \{0\}
- RangeMap = \{1, 2\}

#### Notes:

- Rows are wholly owned.
- RowMap is NOT = DomainMap
- RowMap is NOT = RangeMap (all 1-to-1).
- ColMap is NOT 1-to-1.
- Call to FillComplete:

\[
\text{A.FillComplete(DomainMap, RangeMap)}
\]
Case 2: Twist 2

- First row of $A$, part of second row of $A$, first element of $y$ and last 2 elements of $x$, kept on PE 0.
- Last row, part of second row of $A$, last 2 element of $y$ and first element of $x$, kept on PE 1.

<table>
<thead>
<tr>
<th>PE 0 Contents</th>
<th>PE 1 Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y = \begin{bmatrix} y_1 \end{bmatrix}$, $A = \begin{bmatrix} 2 &amp; -1 &amp; 0 \ -1 &amp; 1 &amp; 0 \end{bmatrix}$, $x = \begin{bmatrix} x_2 \ x_3 \end{bmatrix}$</td>
<td>$y = \begin{bmatrix} y_2 \ y_3 \end{bmatrix}$, $A = \begin{bmatrix} 0 &amp; 1 &amp; -1 \ 0 &amp; -1 &amp; 2 \end{bmatrix}$, $x = \begin{bmatrix} x_1 \end{bmatrix}$</td>
</tr>
</tbody>
</table>

- RowMap = $\{0, 1\}$
- ColMap = $\{0, 1\}$
- DomainMap = $\{1, 2\}$
- RangeMap = $\{0\}$
- RowMap = $\{1, 2\}$
- ColMap = $\{1, 2\}$
- DomainMap = $\{0\}$
- RangeMap = $\{1, 2\}$

Notes:
- Rows are NOT wholly owned.
- RowMap is NOT $\neq$ DomainMap
  is NOT $\neq$ RangeMap (all 1-to-1).
- RowMap and ColMap are NOT 1-to-1.
- Call to FillComplete:
  $\text{A.FillComplete(DomainMap, RangeMap);}$
What does FillComplete Do?

- A bunch of stuff.
- One task is to create (if needed) import/export objects to support distributed matrix-vector multiplication:
  - If ColMap ≠ DomainMap, create Import object.
  - If RowMap ≠ RangeMap, create Export object.
- A few rules:
  - Rectangular matrices will always require: A.FillComplete(DomainMap,RangeMap);
    - DomainMap and RangeMap must be 1-to-1.
Tpetra

- The new kid on the linear algebra block.
- Originally created to overcome the shortcomings of Epetra, e.g., int as index type.
- More flexible handling of memory (through Kokkos)
- “There is no such thing as too many templates.”
- Virtually up to speed with Epetra performance now.
Typical Flow of Tpetra Object Construction

- **Construct Platform**
  - Options are:
    - Kokkos node. Options are:
      - Generalization of Epetra “Comm”
      - Composed with Kokkos::Node
      - Pthreads, OpenMP, Thrust (CUDA), TBB, Serial

- **Construct Map**

- **Construct Node**

- **Construct x**

- **Construct b**

- **Construct A**
Data Classes Stacks

- Epetra
- Simple Array Types
- Xpetra
- Tpetra
- Manycore BLAS
- Kokkos POM Layer
- Node sparse structures
- Kokkos Array
- User Array Types

Classic Stack

New Stack
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
Linear System Solves
AztecOO

- Krylov subspace solvers: CG, GMRES, BiCGSTAB,…
- Incomplete factorization preconditioners

- Aztec is Sandia’s workhorse solver:
  - Extracted from the MPSalsa reacting flow code
  - Installed in dozens of Sandia apps
  - 1900+ external licenses

- AztecOO improves on Aztec by:
  - Using Epetra objects for defining matrix and vectors
  - Providing more preconditioners/scalings
  - Using C++ class design to enable more sophisticated use

- AztecOO interface allows:
  - Continued use of Aztec for functionality
  - Introduction of new solver capabilities outside of Aztec

Developers: Mike Heroux, Alan Williams, Ray Tuminaro
AztecOO Extensibility

AztecOO is designed to accept externally defined:
  - **Operators** (both $A$ and $M$):
    - The linear operator $A$ is accessed as an Epetra_Operator.
    - Users can register a preconstructed preconditioner as an Epetra_Operator.
  - **RowMatrix**:
    - If $A$ is registered as a RowMatrix, Aztec’s preconditioners are accessible.
    - Alternatively $M$ can be registered separately as an Epetra_RowMatrix, and Aztec’s preconditioners are accessible.
  - **StatusTests**:
    - Aztec’s standard stopping criteria are accessible.
    - Can override these mechanisms by registering a StatusTest Object.
AztecOO understands Epetra_Operator

- AztecOO is designed to accept externally defined:
  - Operators (both $A$ and $M$).
  - RowMatrix (Facilitates use of AztecOO preconditioners with external $A$).
  - StatusTests (externally-defined stopping criteria).
Belos and Anasazi

- Next generation linear solver / eigensolver library, written in templated C++.
- Provide a generic interface to a collection of algorithms for solving large-scale linear problems / eigenproblems.
- Algorithm implementation is accomplished through the use of traits classes and abstract base classes:
  - e.g.: MultiVecTraits, OperatorTraits
  - e.g.: SolverManager, Eigensolver / Iteration, Eigenproblem/LinearProblem, StatusTest, OrthoManager, OutputManager
- Includes block linear solvers / eigensolvers:
  - Higher operator performance.
  - More reliable.
- Solves:
  - $AX = X\Lambda$ or $AX = B\Lambda$ (Anasazi)
  - $AX = B$ (Belos)
Why are Block Solvers Useful?

- **Block Solvers (in general):**
  - Achieve better performance for operator-vector products.

- **Block Eigensolvers (** $\text{Op}(A)X = LX$ **):**
  - Reliably determine multiple and/or clustered eigenvalues.
  - Example applications: Modal analysis, stability analysis, bifurcation analysis (LOCA)

- **Block Linear Solvers (** $\text{Op}(A)X = B$ **):**
  - Useful for when multiple solutions are required for the same system of equations.
  - Example applications:
    - Perturbation analysis
    - Optimization problems
    - Single right-hand sides where $A$ has a handful of small eigenvalues
    - Inner-iteration of block eigensolvers
Belos and Anasazi are solver libraries that:

1. Provide an abstract interface to an operator-vector products, scaling, and preconditioning.
2. Allow the user to enlist any linear algebra package for the elementary vector space operations essential to the algorithm. (Epetra, PETSc, etc.)
3. Allow the user to define convergence of any algorithm (a.k.a. status testing).
4. Allow the user to determine the verbosity level, formatting, and processor for the output.
5. Allow these decisions to be made at runtime.
6. Allow for easier creation of new solvers through “managers” using “iterations” as the basic kernels.
Nonlinear System Solves
NOX and LOCA are a combined package for solving and analyzing sets of nonlinear equations.

- NOX: Globalized Newton-based solvers.
- LOCA: Continuation, Stability, and Bifurcation Analysis.

We define the nonlinear problem:

\[
\text{given } F : \mathbb{R}^n \rightarrow \mathbb{R}^n, \\
\text{find } x_\star \in \mathbb{R}^n \text{ such that } F(x_\star) = 0 \in \mathbb{R}^n
\]

\(F\) is the residual or function evaluation

\(x\) is the solution vector

\(J \in \mathbb{R}^{n \times n}\) is the Jacobian Matrix defined by:

\[J_{ij} = \frac{\partial F_i}{\partial x_j}\]
Nonlinear Solver Algorithms

Broyden’s Method
\[ M_B = f(x_c) + B_c d \]

Newton’s Method
\[ M_N = f(x_c) + J_c d \]

Tensor Method
\[ M_T = f(x_c) + J_c d + \frac{1}{2} T_c d d \]

Globalizations

Line Search
- Interval Halving
- Quadratic
- Cubic
- More’-Thuente
- Curvilinear (Tensor)

Trust Region
- Dogleg
- Inexact Dogleg

Homotopy
- Artificial Parameter Continuation
- Natural Parameter Continuation

Iterative Linear Solvers: Adaptive Forcing Terms
- Jacobian-Free Newton-Krylov
- Jacobian Estimation: Colored Finite Difference
Example: Newton’s Method for $F(x) = 0$

- Choose an initial guess $x_0$
- For $k = 0,1,2,...$
  - Compute $F_k = F(x_k)$
  - Compute $J_k$ where $(J_k)_{ij} = \frac{\partial F_i(x_k)}{\partial x_j}$
  - Let $d_k = -J_k^{-1}F_k$
  - (Optional) Let $\lambda_k$ be a calculated step length
  - Set $x_{k+1} = x_k + \lambda_k d_k$
  - Test for Convergence or Failure
Stopping Criteria
(StatusTests)

Highly Flexible Design: Users build a convergence test hierarchy and registers it with the solver (via solver constructor or reset method).

- **Norm F**: \{Inf, One, Two\} \{absolute, relative\} \quad \|F\| \leq \text{tol}
- **Norm Update \(\Delta X\)**: \{Inf, One, Two\} \quad \|x_k - x_{k-1}\| \leq \text{tol}
- **Norm Weighted Root Mean Square (WRMS)**:

\[
C \left( \frac{1}{N} \sum_{i=1}^{N} \left( \frac{x_i^k - x_i^{k-1}}{RTOL |x_i^{k-1}| + ATOL_i} \right)^2 \right) \leq \text{tol}
\]

- **Max Iterations**: Failure test if solver reaches max # iters
- **FiniteValue**: Failure test that checks for NaN and Inf on \(\|F\|\)
- **Stagnation**: Failure test that triggers if the convergence rate fails a tolerance check for n consecutive iterations.

\[
\frac{\|F_k\|}{\|F_{k-1}\|} \geq \text{tol}
\]

- **Combination**: \{AND, OR\}
- **Users Designed**: Derive from NOX::StatusTest::Generic
Building a Status Test

- Converge if both: \( \|F\| \leq 1.0 \times 10^{-6} \) \( \|\delta x\|_{WRMS} \leq 1.0 \)
- Fail if value of \( \|F\| \) becomes Nan or Inf
- Fail if we reach maximum iterations

**FiniteValue**: `finiteValueTest`

**MaxIters**: `maxItersTest`

**Combo (AND)**: `convergedTest`

**Combo (OR)**: `allTests`

```
NOX::StatusTest::NormF normFTest();
NOX::StatusTest::NormWRMS normWRMSTest();
NOX::StatusTest::Combo convergedTest(NOX::StatusTest::Combo::AND);
convergedTest.addStatusTest(normFTest);
convergedTest.addStatusTest(normWRMSTest);
NOX::StatusTest::FiniteValue finiteValueTest;
NOX::StatusTest::MaxIters maxItersTest(200);
NOX::StatusTest::Combo allTests(NOX::StatusTest::Combo::OR);
allTests.addStatusTest(finiteValueTest);
allTests.addStatusTest(maxItersTest);
allTests.addStatusTest(convergedTest);
```
Status Tests (ctd.)

-- Final Status Test Results --
Converged....OR Combination ->
  Converged....AND Combination ->
    Converged....F-Norm = 3.567e-13 < 1.000e-08
      (Length-Scaled Two-Norm, Absolute Tolerance)
    Converged....WRMS-Norm = 1.724e-03 < 1
      (Min Step Size:  1.000e+00 >= 1)
      (Max Lin Solv Tol:  4.951e-14 < 0.5)
  ????????????Finite Number Check (Two-Norm F) = Unknown
  ????????????Number of Iterations = -1 < 200

User Defined are Derived from NOX::StatusTest::Generic
NOX::StatusTest::StatusType checkStatus(const NOX::Solver::Generic &problem)

NOX::StatusTest::StatusType
checkStatusEfficiently(const NOX::Solver::Generic &problem,
   NOX::StatusTest::CheckType checkType)

NOX::StatusTest::StatusType getStatus() const

ostream& print(ostream &stream, int indent=0) const
NOX Interface

NOX solver methods are ANAs, and are implemented in terms of group/vector abstract interfaces:

<table>
<thead>
<tr>
<th>Group</th>
<th>Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>computeF()</td>
<td>innerProduct()</td>
</tr>
<tr>
<td>computeJacobian()</td>
<td>scale()</td>
</tr>
<tr>
<td>applyJacobianInverse()</td>
<td>norm()</td>
</tr>
<tr>
<td></td>
<td>update()</td>
</tr>
</tbody>
</table>

NOX solvers will work with any group/vector that implements these interfaces.

Four concrete implementations are supported:
1. LAPACK
2. EPETRA
3. PETSc
4. Thyra
NOX Framework

Solver Layer
- Solvers
  - Line Search
  - Trust Region
- Directions
  - e.g., Newton
- Line Searches
  - e.g., Polynomial
- Status Tests
  - e.g., Norm F

Abstract Layer
- Abstract Vector & Abstract Group

Linear Algebra Interface
- Implementations
  - EPetra
  - PETSc
  - LAPACK
  - USER DEFINED
- EPetra Dependent Features
  - Jacobian-Free Newton-Krylov
  - Preconditioning
  - Graph Coloring / Finite Diff.

Application Interface Layer
- User Interface
  - Compute F
  - Compute Jacobian
  - Compute Preconditioner
The Epetra “Goodies”

• Matrix-Free Newton-Krylov Operator
  • Derived from Epetra_Operator
  • Can be used to estimate Jacobian action on a vector
  • NOX::Epetra::MatrixFree

\[
J_y = \frac{F(x + y\delta) - F(x)}{\delta}
\]

\[
J_j = \frac{F(x + \delta e_j) - F(x)}{\delta}
\]

• Finite Difference Jacobian
  • Derived from an Epetra_RowMatrix
  • Can be used as a preconditioner matrix
  • NOX::Epetra::FiniteDifference

• Graph Colored Finite Difference Jacobian
  • Derived from NOX::Epetra::FiniteDifference
  • Fast Jacobian fills – need connectivity/coloring graph
  • (NOX::Epetra::FiniteDifferenceColoring)

• Full interface to AztecOO using NOX parameter list
• Preconditioners: internal AztecOO, Ifpack, User defined
• Scaling object