1

# An Overview of Trilinos

**Heidi K. Thornquist**
**Sandia National Laboratories**

ACTS Workshop
August 21-24, 2007

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.

Sandia National Laboratories

2

# Trilinos Development Team

Ross Bartlett
Lead Developer of Thyra and Stratimikos
Developer of Rythmos

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

Jason Cross
Developer of Jpetra

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 Kotulski
Lead Developer of Pliris

Rich Lehoucq
Developer of Anasazi and Belos

Kevin Long
Lead Developer of Thyra,
Developer of Belos and Teuchos

Roger Pawlowski
Lead Developer of NOX

Michael Phenow
Trilinos Webmaster
Lead Developer of New_Package

Eric Phipps
Developer of LOCA, NOX, and Sacado

Denis Ridzal
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 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

Sandia National Laboratories

# Outline of Talk

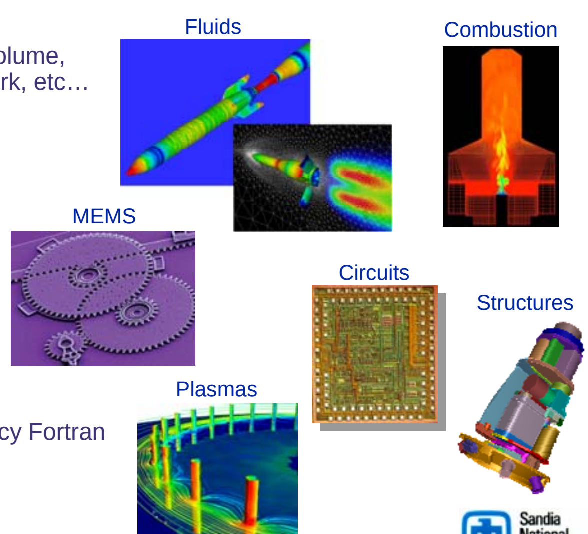
- Background / Motivation / Evolution.
- Trilinos Package Concepts.
- Whirlwind Tour of Trilinos Packages.
- Next Generation Iterative Solvers: Belos & Anasazi.
- Solver Collaborations: ANAs, LALs and APPs.
- Concluding remarks.

Sandia National Laboratories



# 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




Sandia National Laboratories

# Motivation For Trilinos

- Sandia does LOTS of solver work.
- 7 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.

Sandia National Laboratories



# Evolving Trilinos Solution

- Trilinos[1] is an evolving framework to address these challenges:
  - Fundamental atomic unit is a *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 (new_package package):
    - Source code management (cvs, bonsai).
    - Build tools (autotools).
    - Automated regression testing (queue directories within repository).
    - 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 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"

Sandia National Laboratories

7

# Evolving Trilinos Solution

- Beyond a “solvers” framework
- Natural expansion of capabilities to satisfy application and research needs

**physics**

**$L(u)=f$**
*Math. model*

**$L_h(u_h)=f_h$**
*Numerical model*

**$u_h=L_h^{-1}\bullet f_h$**
*Algorithms*

**computation**

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

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

*Trilinos*

**discretizations**
Time domain
Space domain

**methods**
Automatic diff.
Domain dec.
Mortar methods

**solvers**
Linear
Nonlinear
Eigenvalues
Optimization

**core**
Petra
Utilities
Interfaces
Load Balancing

- Discretization methods, AD, Mortar methods, …

Sandia National Laboratories

8

# Trilinos Package Summary

| | Objective | Package(s) |
|---|---|---|
| Discretizations | Spatial Discretizations (FEM,FV,FD) | Intrepid |
| | Time Integration | Rythmos |
| Methods | Automatic Differentiation | Sacado |
| | Mortar Methods | Moertel |
| Core | Linear algebra objects | Epetra, Jpetra, Tpetra |
| | Abstract interfaces | Thyra, Stratimikos, RTOp |
| | Load Balancing | Zoltan, Isorropia |
| | “Skins” | PyTrilinos, WebTrilinos, Star-P, *ForTrilinos* |
| | C++ utilities, (some) I/O | Teuchos, EpetraExt, Kokkos, Triutils |
| Solvers | Iterative (Krylov) linear solvers | AztecOO, Belos, Komplex |
| | Direct sparse linear solvers | Amesos |
| | Direct dense linear solvers | Epetra, Teuchos, Pliris |
| | Iterative eigenvalue solvers | Anasazi |
| | ILU-type preconditioners | AztecOO, IFPACK |
| | Multilevel preconditioners | ML, CLAPS |
| | Block preconditioners | Meros |
| | Nonlinear system solvers | NOX, LOCA |
| | Optimization (SAND) | MOOCHO, Aristos |

Sandia National Laboratories

Trilinos

# Package Concepts

Sandia National Laboratories



# Interoperability vs. Dependence

**(“Can Use”) (“Depends On”)**

- Although most Trilinos packages have no explicit dependence, often packages must interact with *some* other packages:
  - NOX needs operator, vector and solver objects.
  - AztecOO needs preconditioner, matrix, operator and vector objects.
  - Interoperability is enabled at configure time. For example, NOX:
    - `--enable-nox-lapack` compile NOX lapack interface libraries
    - `--enable-nox-epetra` compile NOX epetra interface libraries
    - `--enable-nox-petsc` compile NOX petsc interface libraries
- Trilinos `configure` script is vehicle for:
  - Establishing interoperability of Trilinos components…
  - Without compromising individual package autonomy.
- Trilinos offers seven basic interoperability mechanisms.

Sandia National Laboratories

11

# Trilinos Interoperability Mechanisms

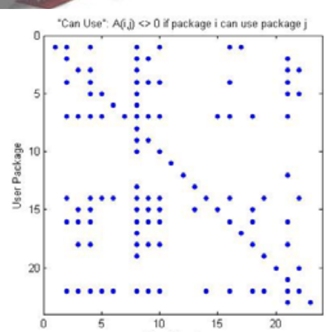
## (Acquired as Package Matures)

| | | |
|---|---|---|
| *Package* builds under Trilinos configure scripts. | ⇒ | *Package* can be built as part of a suite of packages; cross-package interfaces enable/disable automatically |
| *Package* accepts user data as Epetra or Thyra objects | ⇒ | Applications using Epetra/Thyra can use *package* |
| *Package* accepts parameters from Teuchos ParameterLists | ⇒ | Applications using Teuchos ParameterLists can drive *package* |
| *Package* can be used via Thyra abstract solver classes | ⇒ | Applications or other packages using Thyra can use *package* |
| *Package* can use Epetra for private data. | ⇒ | *Package* can then use other packages that understand Epetra |
| *Package* accesses solver services via Thyra interfaces | ⇒ | *Package* can then use other packages that implement Thyra interfaces |
| *Package* available via PyTrilinos | ⇒ | *Package* can be used with other Trilinos packages via Python. |

Sandia National Laboratories

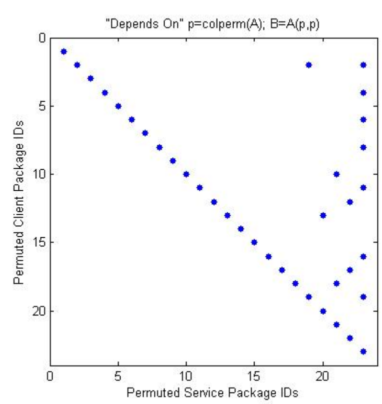
12

# "Can Use" vs. "Depends On"




"Can Use"

- Interoperable without dependence.
- Dense is Good.
- Encouraged.




"Depends On"

- OK, if essential.
- Epetra: 9 clients.
- Thyra, Teuchos, NOX: 2 clients.
- Discouraged.

National Laboratories

# What Trilinos is not …

- Trilinos is not a single monolithic piece of software. Each package:
  - Can be built independent of Trilinos.
  - Has its own self-contained CVS structure.
  - Has its own Bugzilla product and mail lists.
  - Development team is free to make its own decisions about algorithms, coding style, release contents, testing process, etc.
- Trilinos top layer is not a large amount of source code:
  - Trilinos repository (6.0 branch) contains: 660,378 source lines of code (SLOC).
  - Sum of the packages SLOC counts: 648,993.
  - Trilinos top layer SLOC count: 11,385 (1.7%).
- Trilinos is not “indivisible”:
  - You don’t need all of Trilinos to get things done.
  - Any collection of packages can be combined and distributed.
  - Current public release contains only 26 of the 30+ Trilinos packages.

Sandia National Laboratories



Trilinos

# Whirlwind Tour of Packages

Discretizations Methods Core Solvers

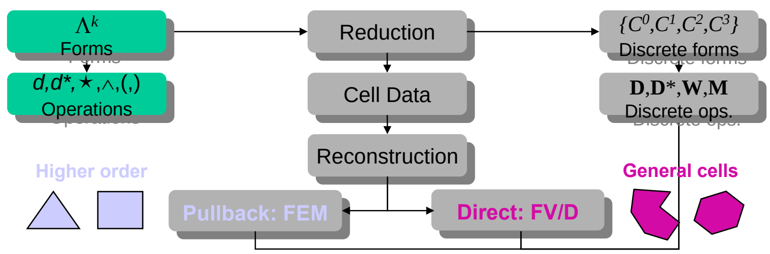
Sandia National Laboratories

# Intrepid

***Interoperable Tools for Rapid Development of Compatible Discretizations***

Intrepid offers an **innovative software design** for compatible discretizations:

- allows access to FEM, FV and FD methods using a common API
- supports hybrid discretizations (FEM, FV and FD) on unstructured grids
- supports a variety of cell shapes:
  - standard shapes (e.g. tets, hexes): high-order finite element methods
  - arbitrary (polyhedral) shapes: low-order mimetic reconstructions
- enables optimization, error estimation, V&V, and UQ using fast invasive techniques (direct support for cell-based derivative computations or via automatic differentiation)




**Developers: Pavel Bochev, Denis Ridzal, David Day**



# 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: Todd Coffey, Roscoe Bartlett**

Sandia National Laboratories

Trilinos

# Whirlwind Tour of Packages

**Discretizations** **Methods** **Core** **Solvers**

Sandia National Laboratories



# Moertel: Mortar Methods

- Capabilities for nonconforming mesh tying and contact formulations in 2 and 3 dimensions using Mortar methods.
- Mortar methods are types of Lagrange Multiplier constraints that can be used in contact formulations and in non-conforming or conforming mesh tying as well as in domain decomposition techniques.
- Used in a large class of nonconforming situations such as the surface coupling of different physical models, discretization schemes or non-matching triangulations along interior interfaces of a domain.

**Developer: Michael Gee**

Sandia National Laboratories

# 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: $(x,\, V) \longrightarrow \left(f,\, \frac{\partial f}{\partial x} V\right)$
    - Propagate derivatives of intermediate variables w.r.t. independent variables forward
    - Directional derivatives, tangent vectors, square Jacobians, $\partial f / \partial x$ when **m ≥ n**.
  - Reverse Mode: $(x,\, W) \longrightarrow \left(f,\, W^T \frac{\partial f}{\partial x}\right)$
    - Propagate derivatives of dependent variables w.r.t. intermediate variables backwards
    - Gradients, Jacobian-transpose products (adjoints), $\partial f / \partial x$ when **n > m**.
  - Taylor polynomial mode: $x(t) = \sum_{k=0}^{d} x_k t^k \longrightarrow \sum_{k=0}^{d} f_k t^k = f(x(t)) + O(t^{d+1}), \quad f_k = \frac{1}{k!} \frac{d^k}{dt^k} f(x(t))$
  - Basic modes combined for higher derivatives.

**Developers: Eric Phipps, David Gay**

Sandia National Laboratories



Trilinos

# Whirlwind Tour of Packages

**Discretizations** **Methods** **Core** **Solvers**

Sandia National Laboratories

21

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

**Developers: Roscoe Barlett, Kevin Long, Heidi Thorquist, Mike Heroux, Paul Sexton, Kris Kampshoff, Chris Baker**

Sandia National Laboratories

22

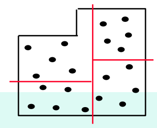
# Kokkos

- Collection of several sparse/dense kernels that affect the performance of preconditioned Krylov methods
- Goal:
  - Isolate key non-BLAS kernels for the purposes of optimization.
- Kernels:
  - Dense vector/multivector updates and collective ops (not in BLAS/Teuchos).
  - Sparse MV, MM, SV, SM.
- Serial-only for now.
- Reference implementation provided (templated).
- Mechanism for improving performance:
  - Default is aggressive compilation of reference source.
  - BeBOP: Jim Demmel, Kathy Yelick, Rich Vuduc, UC Berkeley.
  - Vector version: Cray.

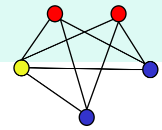
**Developer: Mike Heroux**

Sandia National Laboratories

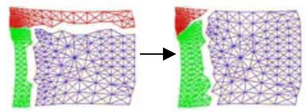
# Zoltan

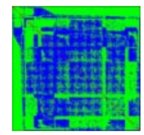


- Data Services for Dynamic Applications

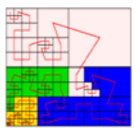


  - Dynamic load balancing
  - Graph coloring
  - Data migration
  - Matrix ordering



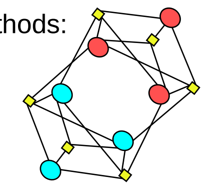


- Partitioners:



  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 Partitioning
  - Hypergraph Repartitioning PaToH (Catalyurek)
  - Zoltan Hypergraph Partitioning
  - ParMETIS (U. Minnesota)
  - Jostle (U. Greenwich)



**Developers: Karen Devine, Eric Boman, Robert Heaphy**

Sandia National Laboratories



# Trilinos Common Language: Petra

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

- Petra[1] 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 3 implementations under development.

**[1]Petra is Greek for "foundation".**

Sandia National Laboratories

## 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.
- Jpetra (Java Petra):
  - Pure Java. Portable to any JVM.
  - Interfaces to Java versions of MPI, LAPACK and BLAS via interfaces.

**Developers: Mike Heroux, Rob Hoekstra, Alan Williams, Paul Sexton**

Sandia National Laboratories



## EpetraExt: Extensions to Epetra

- Library of useful classes not needed by everyone
- Most classes are types of “transforms”.
- Examples:
  - Graph/matrix view extraction.
  - Epetra/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.

**Developer: Robert Hoekstra, Alan Williams, Mike Heroux**

Sandia National Laboratories

# 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, ML, PETSc, LAPACK
- Transient/DAE solvers (implicit)
  - Abstraction of nonlinear solve.
  - … and so on.

**Developers: Roscoe Bartlett, Kevin Long**

Sandia National Laboratories



# PyTrilinos

- PyTrilinos provides Python access to Trilinos packages.
- Uses SWIG to generate bindings.
- Epetra, AztecOO, IFPACK,
  ML, NOX, LOCA, Amesos
  and NewPackage are support.
- Possible to:
  - Define RowMatrix implementation in Python.
  - Use from Trilinos C++ code.
- Performance for large grain is equivalent to C++.
- Several times hit for very fine grain code.

**Developer: Bill Spotz**

Sandia National Laboratories

# WebTrilinos

- 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.
  - Will use during hands-on.

**Developers: Ray Tuminaro, Jonathan Hu, Marzio Sala**

Sandia National Laboratories



Trilinos

# Whirlwind Tour of Packages

**Discretizations** **Methods** **Core** **Solvers**

Sandia National Laboratories

31

## 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**

Sandia National Laboratories

32


## : 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.

**Developers: Ray Tuminaro, Jonathan Hu, Marzio Sala**

Sandia National Laboratories

# Amesos

- 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**

Sandia National Laboratories



# 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**

Sandia National Laboratories

# Belos and Anasazi

- Next generation linear solvers (Belos) and eigensolvers (Anasazi) libraries, written in templated C++.

- Provide a generic interface to a collection of algorithms for solving large-scale linear problems and eigenproblems.
- Algorithm implementation is accomplished through the use of abstract base classes (mini interface) and traits classes. Interfaces are derived from these base classes to:
  - operator-vector products
  - status tests
  - orthogonalization
  - any arbitrary linear algebra library.
- Includes block linear solvers and eigensolvers.

**Developers: Heidi Thornquist, Mike Heroux, Chris Baker, Rich Lehoucq, Ulrich Hetmaniuk**

Sandia National Laboratories



# NOX: Nonlinear Solvers

- Suite of nonlinear solution methods

- NOX uses abstract vector and “group” interfaces:
  - Allows flexible selection and tuning of various strategies:
    - Directions.
    - Line searches.
  - Epetra/AztecOO/ML, LAPACK, PETSc implementations of abstract vector/group interfaces.
- Designed to be easily integrated into existing applications.

**Developers: Tammy Kolda, Roger Pawlowski**

Sandia National Laboratories

# LOCA

- 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**

Sandia National Laboratories



# 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.
  - Efficiently manages inexactness in the inner linear system solves.

**Developer: Denis Ridzal**

Sandia National Laboratories

## Full “Vertical” Solver Coverage

| | | Trilinos Packages |
|---|---|---|
| Optimization Problems:<br>· Unconstrained:<br>· Constrained: | Find $u \in \Re^n$ that minimizes $f(u)$<br>Find $y \in \Re^m$ and $u \in \Re^n$ that minimizes $f(y,u)$ s.t. $c(y,u)=0$ | MOOCHO,<br>Aristos |
| · Transient Problems:<br>· DAEs/ODEs: | Solve $f(\dot{x}(t), x(t), t) = 0$<br>$t \in [0,T]$, $x(0) = x_0$, $\dot{x}(0) = x'_0$<br>for $x(t) \in \Re^n$, $t \in [0,T]$ | Rythmos |
| · Nonlinear Problems:<br>· Nonlinear equations:<br>· Stability analysis: | Given nonlinear op $c(x,u) \in \Re^{n+m} \to \Re^n$<br>Solve $c(x)=0$ for $x \in \Re^n$<br>For $c(x,u)=0$ find space $u \in U \ni \frac{\partial c}{\partial x}$ singular | NOX<br>LOCA |
| · Implicit Linear Problems:<br>· Linear equations:<br>· Eigen problems: | Given linear ops (matrices) $A, B \in \Re^{n \times n}$<br>Solve $Ax = b$ for $x \in \Re^n$<br>Solve $Av = \lambda Bv$ for (all) $v \in \Re^n$, $\lambda \in \Re$ | AztecOO, Belos, Ifpack,ML,etc.<br>Anasazi |
| · Explicit Linear Problems:<br>· Matrix/graph equations:<br>· Vector problems: | Compute $y = Ax$; $A = A(G)$; $A \in \Re^{m \times n}, G \in \square^{m \times n}$<br>Compute $y = \alpha x + \beta w$; $\alpha = < x, y >$; $x, y \in \Re^n$ | Epetra, Tpetra |

Sandia National Laboratories



# Next Generation Iterative Solvers: Belos & Anasazi

Sandia National Laboratories

## Background

- Several iterative linear solver / eigensolver libraries exist:
  - PETSc, SLAP, LINSOL, Aztec(OO), …
  - SLEPc, PRIMME, LOBPCG (hypre / BLOPEX), ARPACK, …
- None are (completely) written in C++
- None of the linear solver libraries can efficiently deal with multiple right-hand sides
- Stopping criteria are predetermined for most libraries
- The underlying linear algebra is (usually) static





## AztecOO Linear Solver Library

- A C++ wrapper around Aztec library written in C.
- Algorithms: GMRES, CG, CGS, BiCGSTAB, TFQMR.
- Offers status testing capabilities.
- Output verbosity level can be determined by user.
- Uses Epetra to perform underlying vector space operations.
- Interface to matrix-vector product is defined by the user through the Epetra_Operator.

# ARnoldi PACKage (ARPACK)

- Written in Fortran 77.
- Algorithms: Implicitly Restarted Arnoldi/Lanczos
- Static convergence tests.
- Output formatting, verbosity level is determined by user.
- Uses LAPACK/BLAS to perform underlying vector space operations.
- Offers abstract interface to matrix-vector products through reverse communication.

Sandia National Laboratories



# Scalable Library for Eigenvalue Problem Computations ( SLEPc )

- Written in C (Hernández, Román, and Vidal, 2003).
- Provides some basic eigensolvers as well as wrappers around:
  - ARPACK (Lehoucq, Maschhoff, Sorensen, and Yang, 1998)
  - BLZPACK (Marques, 1995)
  - PLANSO (Wu and Simon 1997)
  - TRLAN (Wu and Simon, 2001)
- Native Algorithms: Power/Subspace Iteration, RQI, Arnoldi
- Wrapped Algorithms: IRAM/IRLM (ARPACK), Block Lanczos (BLZPACK), and Lanczos (PLANSO / TRLAN)
- Static/limited convergence tests.
- Uses PETSc to perform underlying vector space operations, matrix-vector products, and linear solves.
- Allows the creation / registration of new matrix-vector products.

Sandia National Laboratories

# Linear / Eigensolver Software Design

Why not create a solver library that:

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

Sandia National Laboratories



# Generic Iterative Method

Linear/Eigen Problem

Status Test

```
AlgorithmA( … )
   while ( myStatusTest.CheckStatus(…)
               == Unconverged )
      …
      …
      % Compute operator-vector product
      myProblem.ApplyOp(NOTRANS,v,w);
      α = w.dot(v);
      …
   end
   myOM.print(something);
   return (solution);
end
```

Generic Operator Interface

Generic Linear Algebra Interface

Output Manager

Sandia National Laboratories

# Benefits of Generic Programming

1) Generic programming techniques ease the implementation of complex algorithms.
2) Developing algorithms with generic programming techniques is easier on the developer, while still allowing them to build a flexible and powerful software package.
3) Generic programming techniques also allow the user to leverage their existing software investment.

**Caveat: It's not as easy as taking a piece of code and adding:**

```
template <typename OT, typename ST>
```

**More work has to be done to handle "numeric traits"**

Sandia National Laboratories



# Teuchos Numeric Traits

Generic programming technique that uses templated interfaces to define the standard behavior of datatypes.

- OrdinalTraits
  - zero & one
  - int & long int
- ScalarTraits
  - zero, one, magnitude type, absolute value, conjugate, square root, random number generator, …
  - std::numeric_limits
  - float, double, complex<float>, and complex<double>
- Arbitrary precision arithmetic (ARPREC, GMP)
- Templated BLAS/LAPACK wrappers, serial dense matrix/vector

Sandia National Laboratories

# Teuchos Templated BLAS Example

```
double DNRM2( const int n, const double* x, const int incx) const
 {
   int i, ix = 0;
   double result 0.0;
   if ( n > 0 )
     {
       // Set the initial index.
       if (incx < 0) { ix = (-n+1)*incx; }

       for(i = 0; i < n; ++i)
         {
           result += x[ix] * x[ix];
           ix += incx;
         }
       result = std::sqrt(result);
     }
   return result;
 } /* end NRM2 */
```



# Teuchos Templated BLAS Example

```
typedef ScalarTraits<ScalarType>::magnitudeType MagnitudeType
template<typename OrdinalType, typename ScalarType>
MagnitudeType  BLAS<OrdinalType, ScalarType>::NRM2( const OrdinalType n,
                                                    const ScalarType* x,
                                                    const OrdinalType incx) const
 {
   OrdinalType izero = OrdinalTraits<OrdinalType>::zero();
   OrdinalType ione = OrdinalTraits<OrdinalType>::one();
   MagnitudeType result = ScalarTraits<MagnitudeType>::zero();
   OrdinalType i, ix = izero;
   if ( n > izero )
     {
       // Set the initial index.
       if (incx < izero) { ix = (-n+ione)*incx; }

       for(i = izero; i < n; i++)
         {
           result += ScalarTraits<ScalarType>::magnitude(ScalarTraits<ScalarType>::conjugate(x[ix]) * x[ix]);
           ix += incx;
         }
       result = ScalarTraits<MagnitudeType>::squareroot(result);
     }
   return result;
 } /* end NRM2 */
```

51

# Belos and Anasazi

- Next generation eigensolvers 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 $AX = BX\Lambda$ (Anasazi)
  - $AX = B$ (Belos)

Sandia National Laboratories

52

# Why are Block Solvers Useful?

- Block Solvers ( in general ):
  - Achieve better performance for operator-vector products.
- Block Eigensolvers ( $Op(A)X = \Lambda X$ ):
  - Reliably determine multiple and/or clustered eigenvalues.
  - Example applications: Modal analysis, stability analysis, bifurcation analysis (LOCA)
- Block Linear Solvers ( $Op(A)X = B$ ):
  - 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

Sandia National Laboratories

53

# Anasazi / Belos Classes

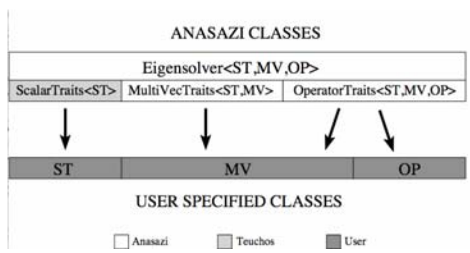
- Eigenproblem / LinearProblem Class
  - Describes the problem and stores the answer
- Eigensolver / Linear Solver Manager (SolverManager) Class
  - Parameter list driven strategy object describing behavior of solver
- Eigensolver / Iteration Class
  - Provide basic iteration interface.
- MultiVecTraits and OperatorTraits
  - Traits classes for interfacing linear algebra
- SortManager Class [Anasazi only]
  - Allows selection of desired eigenvalues
- OrthoManager Class
  - Provide basic interface for orthogonalization
- StatusTest Class
  - Control testing of convergence, etc.
- OutputManager Class
  - Control verbosity and printing in a MP scenario

Sandia National Laboratories

54

# Linear Algebra Interface

- `MultiVecTraits`
  - Abstract interface to define the linear algebra required by most iterative eigensolvers / linear solvers:
    - creational methods
    - dot products, norms
    - update methods
    - initialize / randomize




- `OperatorTraits`
  - Abstract interface to enable the application of an operator to a multivector.

Sandia National Laboratories

# Interfacing Your LAL

- Three approaches for using your own linear algebra:
  - Specialize the traits classes for the multivector and operator:
    - Fill out MultiVecTraits<ST,MyMV>
    - Fill out OperatorTraits<ST,MyMV,MyOP>
  - Subclass abstract base classes, for which traits specializations are already defined:
    - class MyMV : public MultiVec<ST>
    - class MyOP : public Operator<ST>
  - Implement your linear algebra in the Thyra framework:
    - Then use the adapter to Thyra.
- Test your own linear algebra interfaces using:
  - TestMultiVecTraits<ST,MV>(…)
  - TestOperatorTraits<ST,MV,OP>(…)
- Or you can simply use Epetra!

Sandia National Laboratories



# Eigenproblem / LinearProblem Interface

- Provides an interface between the basic iterations and the problem to be solved.
- Anasazi:
  - Abstract base class Anasazi::Eigenproblem
    - Allows spectral transformations to be removed from the algorithm.
    - Differentiates between standard and generalized eigenproblems.
    - Stores initial vector, stiffness/mass matrix, operator, eigensolution.
  - Concrete class Anasazi::BasicEigenproblem
    - Describes standard or general, Hermitian or non-Hermitian eigenproblems.
- Belos:
  - Concrete class Belos::LinearProblem
  - Allows preconditioning to be removed from the algorithm.
  - Initial vector, left/right preconditioner, left/right scaling, operator, solution vectors.

Sandia National Laboratories

57

# StatusTest Interface

- Informs eigensolver / linear solver iteration of change in state, as defined by user.
- Similar to NOX / AztecOO.
- Multiple criterion can be logically connected.
- Abstract base class **`Anasazi::StatusTest / Belos::StatusTest`**
  - `TestStatus checkStatus( Anasazi::Eigensolver<…>* solver );`
    `TestStatus checkStatus( Belos::Iteration<…>* solver );`
  - `TestStatus getStatus() const;`
  - `void clearStatus();`
  - `void reset();`
  - `ostream& print( ostream& os, int indent = 0 ) const;`
- Concrete classes:
  - **`StatusTestMaxIters`**
  - **`StatusTestResNorm`**
  - **`StatusTestOrderedResNorm`** [Anasazi only]
  - **`StatusTestOutput`**
  - **`StatusTestCombo`**

Sandia National Laboratories

58

# Orthogonalization Manager Interface

- Abstract interface to orthogonalization / orthonormalization routines for multivectors.
- Abstract base class **`OrthoManager/MatOrthoManager`**
  - **`Inner product`**
  - **`Multivector norm`**
  - **`Project multivector`**
  - **`Normalize multivector`**
  - **`Project and normalize multivector`**
  - **`Orthogonalization error`**
  - **`Orthonormalization error`**
- Concrete classes: Anasazi
  - **`Anasazi::BasicOrthoManager (DGKS)`**
  - **`Anasazi::SVQBOrthoManager`**
- Concrete classes: Belos
  - **`Belos::DGKSOrthoManager`**
  - **`Belos::ICGSOrthoManager`**
  - **`Belos::IMGSOrthoManager`**

Sandia National Laboratories

# Sort Manager Interface

**[Anasazi only]**

- Abstract interface for managing the sorting of the eigenvalues computed by the eigensolver.
- Important tool to complement spectral transformations.
- Only two methods:
  - `ReturnType sort(Eigensolver<ST,MV,OP>* solver, int n, ST *evals, std::vector<int> *perm=0);`
  - `ReturnType sort(Eigensolver<ST,MV,OP>* solver, int n, ST *r_evals, ST *i_evals, std::vector<int> *perm=0);`
- Concrete class `Anasazi::BasicSort`
  - Provides basic sorting methods:
    - largest/smallest magnitude
    - largest/smallest real part
    - largest/smallest imaginary part



# Output Manager Interface

- Allows user to control which processor will handle output from the solver and the verbosity.
- Default is lowest verbosity, outputting on proc 0.
- Methods:
  - Get/Set Methods:
    - `void setVerbosity( int vbLevel );`
    - `int getVerbosity( );`
    - `ostream& stream( MsgType type );`
  - Query Methods:
    - `bool isVerbosity( MsgType type );`
  - Print Methods:
    - `void print( MsgType type, const string output );`
- For Anasazi, this is an abstract base class
  - Concrete Class `Anasazi::BasicOutputManager`
    - Default is lowest verbosity (errors), output on one processor.

# Eigensolver / Iteration Interface

- Provides an abstract interface to basic iterations.
- Abstract base class `Anasazi::Eigensolver / Belos::Iteration`
  - get / reset number of iterations
  - native residuals
  - current / maximum subspace size
  - set / get auxiliary vectors [Anasazi only]
  - problem
  - initialize / iterate
- Anasazi concrete iterations:
  - `Anasazi::BlockDavidson`
  - `Anasazi::LOBPCG`
  - `Anasazi::BlockKrylovSchur`
- Belos concrete iterations:
  - `Belos::CGIter, Belos::BlockCGIter`
  - `Belos::BlockGmresIter, Belos::PseudoBlockGmresIter`
  - `Belos::GCRODRIter`
- Requires these classes:
  - `Eigenproblem / LinearProblem`
  - `OutputManager`
  - `OrthoManager`
  - `StatusTest`
  - `SortManager` [Anasazi only]

Sandia National Laboratories



# Eigensolver / LinearSolver Manager Interface

- Provides an abstract interface to solver managers (strategies)
- Abstract base class `SolverManager`
  - Access to the eigenproblem / linear problem
  - Solve the eigenproblem / linear problem
- Solvers are parameter list driven, take two arguments:
  - `Anasazi::Eigenproblem / Belos::LinearProblem`
  - `Teuchos::ParameterList`
- Anasazi concrete solver managers:
  - `Anasazi::BlockDavidsonSolMgr`
  - `Anasazi::LOBPCGSolMgr`
  - `Anasazi::BlockKrylovSchurSolMgr`
- Belos concrete solver managers:
  - `Belos::BlockCGSolMgr`
  - `Belos::BlockGmresSolMgr`
  - `Belos::PseudoBlockGmresSolMgr`
  - `Belos::GCRODRSolMgr`

Sandia National Laboratories

# Anasazi / Belos Status

- Anasazi (Trilinos Release 7.0):
  - Solvers: Block Krylov-Schur, Block Davidson, LOBPCG
  - Can solve standard and generalized eigenproblems
  - Can solve Hermitian and non-Hermitian eigenproblems
  - Can target largest or smallest eigenvalues
  - Block size is independent of number of requested eigenvalues
- Belos (Trilinos Release 8.0):
  - Solvers: BlockCG, BlockGMRES, Pseudo-block GMRES, Recycled Krylov (GCRO-DR)
  - Belos::EpetraOperator and Thyra::LOWS interface allows for integration into other codes
  - Block size is independent of number of right-hand sides
- Linear algebra adapters for Epetra, NOX/LOCA, and Thyra
- Epetra interface accepts Epetra_Operators, so can be used with ML, AztecOO, Ifpack, Belos, etc…
- Configurable via Teuchos::ParameterList

Sandia National Laboratories



# Trilinos Availability / Information

- Trilinos and related packages are available via LGPL.
- Current release (7.0) is "click release". Unlimited availability.
  - 1800+ Downloads since 3/05 (not including internal Sandia users).
  - 750 registered users:
    - 57% university, 11% industry, 20% gov't.
    - 35% European, 35% US, 10% Asian.
- Trilinos Release 8: September 2007.
- Trilinos Awards:
  - 2004 R&D 100 Award.
  - SC2004 HPC Software Challenge Award.
  - Sandia Team Employee Recognition Award.
  - Lockheed-Martin Nova Award Nominee.
- More information and downloads:
  - http://trilinos.sandia.gov
- 5th Annual Trilinos User Group Meeting: November 6-8, 2007 at SNL.

Sandia National Laboratories

# Solver Collaborations: ANAs, LALs and APPs



## Categories of Abstract Problems and Abstract Algorithms

Trilinos Packages

- Linear Problems: Given linear operator (matrix) $A \in \mathbf{R}^{n \times n}$
  - Linear equations: Solve $Ax = b$ for $x \in \mathbf{R}^n$ — **Belos**
  - Eigen problems: Solve $Av = \lambda v$ for (all) $v \in \mathbf{R}^n$ and $\lambda \in \mathbf{R}$ — **Anasazi**
- Nonlinear Problems: Given nonlinear operator $c(x, u) \in \mathbf{R}^{n+m} \rightarrow \mathbf{R}^n$
  - Nonlinear equations: Solve $c(x) = 0$ for $x \in \mathbf{R}^n$ — **NOX**
  - Stability analysis: For $c(x, u) = 0$ find space $u \in \mathcal{U}$ such that $\frac{\partial c}{\partial x}$ is singular — **LOCA**
- 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 $x(t) \in \mathbf{R}^n, t \in [0, T]$ — **Rythmos**
- Optimization Problems:
  - Unconstrained: Find $u \in \mathbf{R}^n$ that minimizes $f(u)$ — **MOOCHO**
  - Constrained: Find $y \in \mathbf{R}^m$ and $u \in \mathbf{R}^n$ that:
    minimizes $f(y, u)$
    such that $c(y, u) = 0$

# Introducing 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

Compute $r^{(0)} = b - \boxed{Ax^{(0)}}$ for the initial guess $x^{(0)}$.

**for** $i = 1, 2, \ldots$

$\rho_{i-1} = \langle r^{(i-1)}, r^{(i-1)} \rangle$

$\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$ $(\beta_0 = 0)$

$p^{(i)} = r^{(i-1)} + \beta_{i-1} p^{(i-1)}$ $(p^{(1)} = r^{(1)})$

$q^{(i)} = \boxed{Ap^{(i)}}$

$\gamma_i = \langle p^{(i)}, q^{(i)} \rangle$

$\alpha_i = \rho_{i-1} / \gamma_i$

$x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$

$r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$

check convergence; continue if necessary

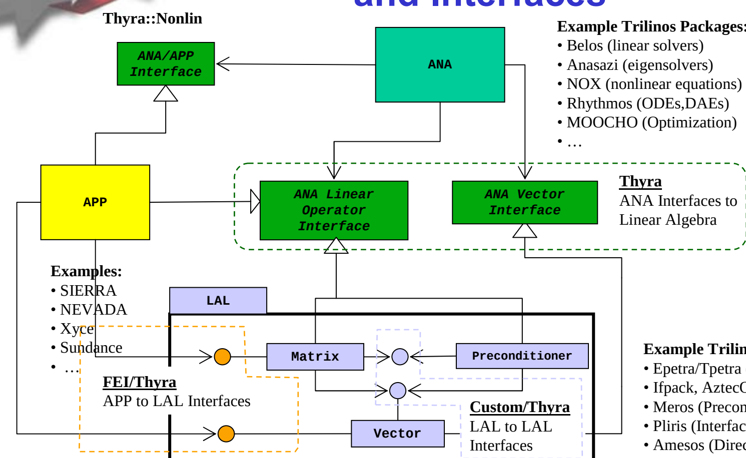
**end**

| Types of operations | Types of objects |
|---|---|
| linear operator applications | Linear Operators • $A$ |
| vector-vector operations | Vectors • $r,\ x,\ p,\ q$ |
| Scalar operations | Scalars • $\rho,\ \beta,\ \gamma,\ \alpha$ |
| scalar product <x,y> defined by vector space | Vector spaces? • $\mathcal{X}$ |

Sandia National Laboratories



# Solver Software Components and Interfaces




**Example Trilinos Packages:**
- Belos (linear solvers)
- Anasazi (eigensolvers)
- NOX (nonlinear equations)
- Rhythmos (ODEs,DAEs)
- MOOCHO (Optimization)
- …

**Example Trilinos Packages:**
- Epetra/Tpetra (Mat,Vec)
- Ifpack, AztecOO, ML (Preconditioners)
- Meros (Preconditioners)
- Pliris (Interface to direct solvers)
- Amesos (Direct solvers)
- Komplex (Complex/Real forms)
- …

**Types of Software Components**

**1) ANA : Abstract Numerical Algorithm (e.g. linear solvers, eigensolvers, nonlinear solvers, stability analysis, uncertainty quantification, transient solvers, optimization etc.)**

**2) LAL : Linear Algebra Library (e.g. vectors, sparse matrices, sparse factorizations, preconditioners)**

**3) APP : Application (the model: physics, discretization method etc.)**

Sandia National Laboratories

# Examples of Nonlinear Abstract Numerical Algorithms

| Class of Numerical Problem | Example ANA |
|---|---|
| Linear equations<br>Solve $Ax = b$ for $x \in \mathbf{R}^n$ | GMRES (e.g. Belos) |
| Nonlinear equations<br>Solve $c(x) = 0$ for $x \in \mathbf{R}^n$ | Newton's method (e.g. NOX, MOOCHO) |
| Initial Value DAE/ODE | Backward Euler method (e.g. Rythmos?) |

Newton's method (e.g. NOX, MOOCHO)

Choose initial guess $x_0$, tolerance $\eta$
**for** $k = 0, 1, \ldots$
  If $||c(x_k)|| \le \eta$ **Stop!**
  $\boxed{\text{Solve } \frac{\partial c(x_k)}{\partial x}\delta x_k = -c(x_k) \text{ for } \delta x}$ **Requires**
  Choose $\alpha$ using a line search method
  $x_{k+1} = x_k + \alpha \delta x_k$
**end for**

Initial Value DAE/ODE

Solve for $x(t) \in \mathbf{R}^n$, $t \in [0, T]$ such that:

- $f(\frac{\partial x}{\partial t}, x(t), t) = 0, t \in [0, T]$
- $x(0) = x_0$
- $\frac{\partial x(0)}{\partial t} = x'_0$

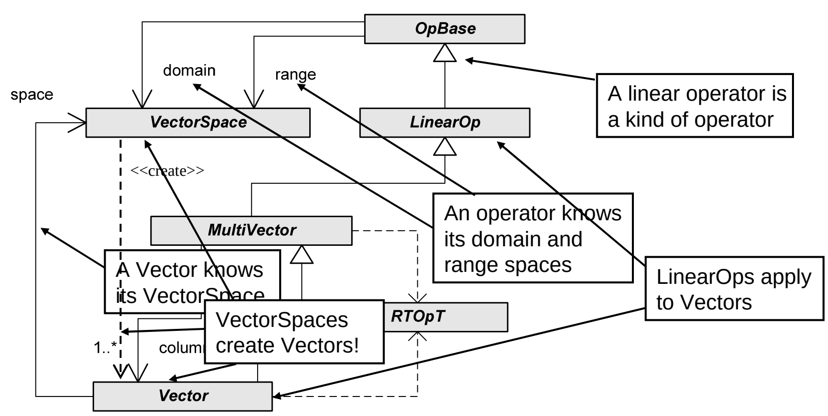
Backward Euler method (e.g. Rythmos?)

Given $x_0$
Choose $N$, $\delta t = T/N$
**for** $k = 1, 2, \ldots, N$
  $\boxed{\text{Solve } f\left(\frac{x_{k+1}-x_k}{\delta t}, x_{k+1}, t_k\right) = 0 \text{ for } x_{k+1}}$ **Requires**
**end for**

Sandia National Laboratories



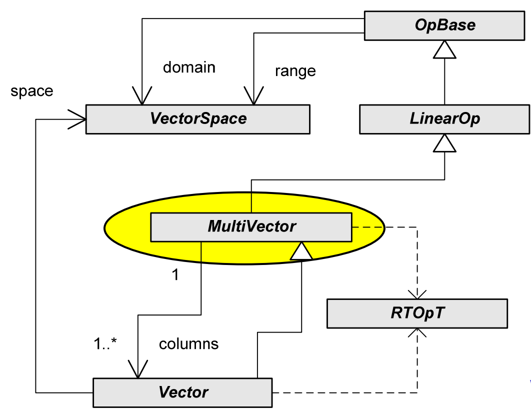
# Basic Thyra Vector and Operator Interfaces




Sandia National Laboratories

71

# Basic Thyra Vector and Operator Interfaces




## What is a multi-vector?

- An *m* multi-vector *V* is a tall thin dense matrix composed of *m* column vectors $v_j$

$$V = \begin{bmatrix} v_1 & v_2 & \dots & v_m \end{bmatrix} \in \mathcal{S} \times \mathbf{R}^m$$

## What ANAs can exploit multi-vectors?

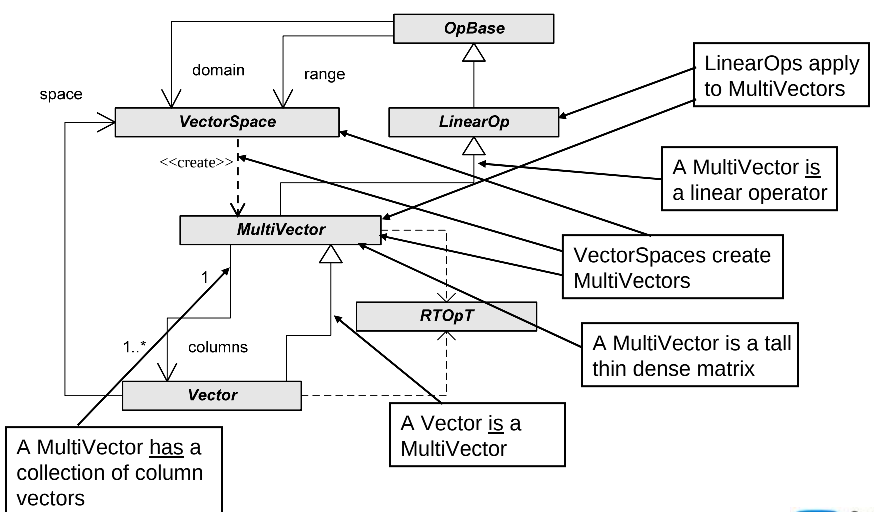
- Block linear solvers
- Block eigensolvers
- Compact limited memory quasi-Newton
- Tensor methods for nonlinear equations

## Why are multi-vectors important?

- Cache performance
- Reduce global communication

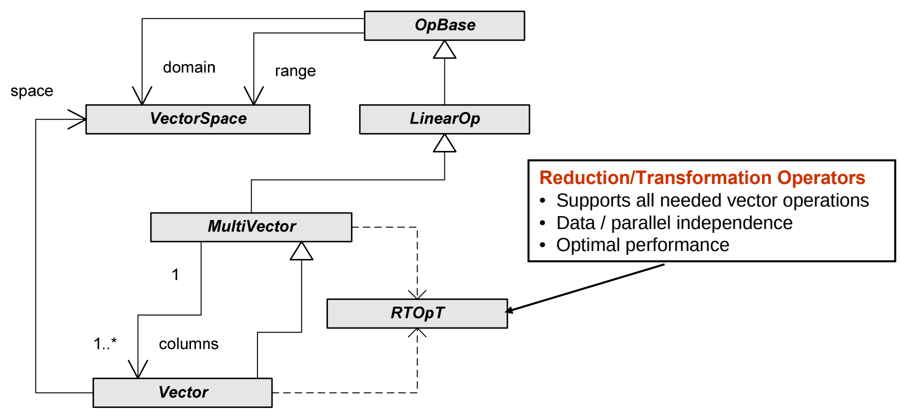
72

# Basic Thyra Vector and Operator Interfaces



73

# Basic Thyra Vector and Operator Interfaces




R. A. Bartlett, B. G. van Bloemen Waanders and M. A. Heroux. Vector Reduction/Transformation Operators, ACM TOMS, March 2004

74

# Trilinos Availability / Information

- Trilinos and related packages are available via LGPL.
- Current release (7.0) is “click release”. Unlimited availability.
  - 1800+ Downloads since 3/05 (not including internal Sandia users).
  - 750 registered users:
    - 57% university, 11% industry, 20% gov’t.
    - 35% European, 35% US, 10% Asian.
- Trilinos Release 8: September 2006.
- Trilinos Awards:
  - 2004 R&D 100 Award.
  - SC2004 HPC Software Challenge Award.
  - Sandia Team Employee Recognition Award.
  - Lockheed-Martin Nova Award Nominee.
- More information:
  - http://trilinos.sandia.gov
- 5th Annual Trilinos User Group Meeting: November 6-8, 2007 at SNL.