SAND2006-5781C

An Overview of the Thyra Interoperability Effort for Abstract Numerical Algorithms within Trilinos

Roscoe A. Bartlett

Department of Optimization and Uncertainty Estimation

Sandia National Laboratories

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

Outline

- Overview of Trilinos
- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for interoperability and layering
- Fundamental Thyra ANA operator/vector interfaces
- History behind Thyra
- Use cases and the scope of Thyra
- New to Thyra in Trilinos 7.0 =>September 2006?
- Wrapping it up

Outline

- Overview of Trilinos
- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for interoperability and layering
- Fundamental Thyra ANA operator/vector interfaces
- History behind Thyra
- Use cases and the scope of Thyra
- New to Thyra in Trilinos 7.0 =>September 2006?
- Wrapping it up

Overview of Trilinos

Trilinos website

http://software.sandia.gov/trilinos

Trilinos is being developed to:

- Provide a suite of numerical solvers to support predictive simulation for Sandia's customers
- Provide a decoupled and scalable development environment to allow for algorithmic research and production capabilities
- Provide support for growing SQA requirements
- Strategic Goals?

At its most basic level Trilinos provides:

- A common source code repository and management system (CVS based)
- Configuration and building support (autoconf/automake based)
- A common infrastructure for SQA
  - Bug reporting and tracking (i.e. Bugzilla)
  - Automated regression testing and reporting (test harness, results emails and webpage)
- Developer and user communication (i.e. Mailman email lists)
- Common integrated documentation system (Trilinos website and Doxygen)
- Provides independent development environment in terms of "packges"

Sandia National Laboratories

Trilinos Package Summary

Trilinos 7.0
September 2006

Objective	Package(s)
Linear algebra objects	Epetra, Jpetra, Tpetra
Krylov solvers	AztecOO, Belos, Komplex
ILU-type preconditioners	AztecOO, IFPACK
Multilevel preconditioners	ML, CLAPS
Eigen problems	Anasazi
Block preconditioners	Meros
Direct sparse linear solvers	Amesos
Direct dense solvers	Epetra, Teuchos, Pliris
Abstract interfaces	Thyra
Nonlinear system solvers	NOX, LOCA, CAPO
Time Integrators/DAEs	Rythmos
C++ utilities, (some) I/O	Teuchos, EpetraExt, Kokkos
Trilinos Tutorial	Didasko
"Skins"	PyTrilinos, WebTrilinos, Star-P, Stratimikos
Simulation-Constrained Optimization	MOOCHO
Archetype package	NewPackage
Other new in 7.0 release	Galeri, Isorropia, Moertel, RTOp

Trilinos Strategic Goals

- **Scalable Solvers**: As problem size and processor counts increase, the cost of the solver will remain a nearly fixed percentage of the total solution time.
- **Hardened Solvers**: Never fail unless problem essentially unsolvable, 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 capabilities from basic linear algebra to transient and optimization solvers.
- ***Grand* Universal Interoperability**: All Trilinos packages will be interoperable, so that any combination of solver packages that makes sense algorithmically will be possible within Trilinos.
- **Universal Solver RAS**: Trilinos will be:
  - Reliable: Leading edge hardened, scalable solutions for each of these applications
  - Available: Integrated into every major application at Sandia
  - Serviceable: Easy to maintain and upgrade within the application environment.

Courtesy of Mike Heroux, Trilinos Project Leader

Trilinos Development Team

Ross Bartlett
Lead Developer of Thyra and MOOCHO
Developer of Rythmos

Paul Boggs
Developer of Thyra

Todd Coffey
Lead Developer of Rythmos

Jason Cross
Developer of Jpetra

David Day
Developer of Komplex

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 and NOX

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
Developer of Epetra, EpetraExt, AztecOO, Tpetra

Sandia National Laboratories

Outline

- Overview of Trilinos
- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for interoperability and layering
- Fundamental Thyra ANA operator/vector interfaces
- History behind Thyra
- Use cases and the scope of Thyra
- New to Thyra in Trilinos 7.0
- Wrapping it up

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 $f(x,p) \in \mathbf{R}^{n+m} \rightarrow \mathbf{R}^n$
  - Nonlinear equations: Solve $f(x) = 0$ for $x \in \mathbf{R}^n$ — NOX
  - Stability analysis: For $f(x,p) = 0$ find space $p \in \mathcal{P}$ such that $\frac{\partial f}{\partial x}$ 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 $p \in \mathbf{R}^m$ that minimizes $g(p)$
  - Constrained: Find $x \in \mathbf{R}^n$ and $p \in \mathbf{R}^m$ that:
    minimizes $g(x,p)$
    such that $f(x,p) = 0$ — MOOCHO

Sandia National Laboratories

Introducing Abstract Numerical Algorithms

What is an abstract numerical algorithm (ANA)?

An ANA is a numerical algorithm that can be expressed abstractly solely in terms of vectors, vector spaces, linear operators, and other abstractions built on top of these without general direct data access or any general assumptions about data locality

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 $Ax = b$

Key Point

If implemented well with the right infrastructure, ANAs can be extremely reusable!

Linear Conjugate Gradient Algorithm

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

for $i = 1, 2, \ldots$

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

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

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

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

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

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

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

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

check convergence; continue if necessary

end

Types of operations

- linear operator applications
- vector-vector operations
- Scalar operations
- scalar product <x,y> defined by vector space

Types of objects

Linear Operators

- A

Vectors

- r, x, p, q

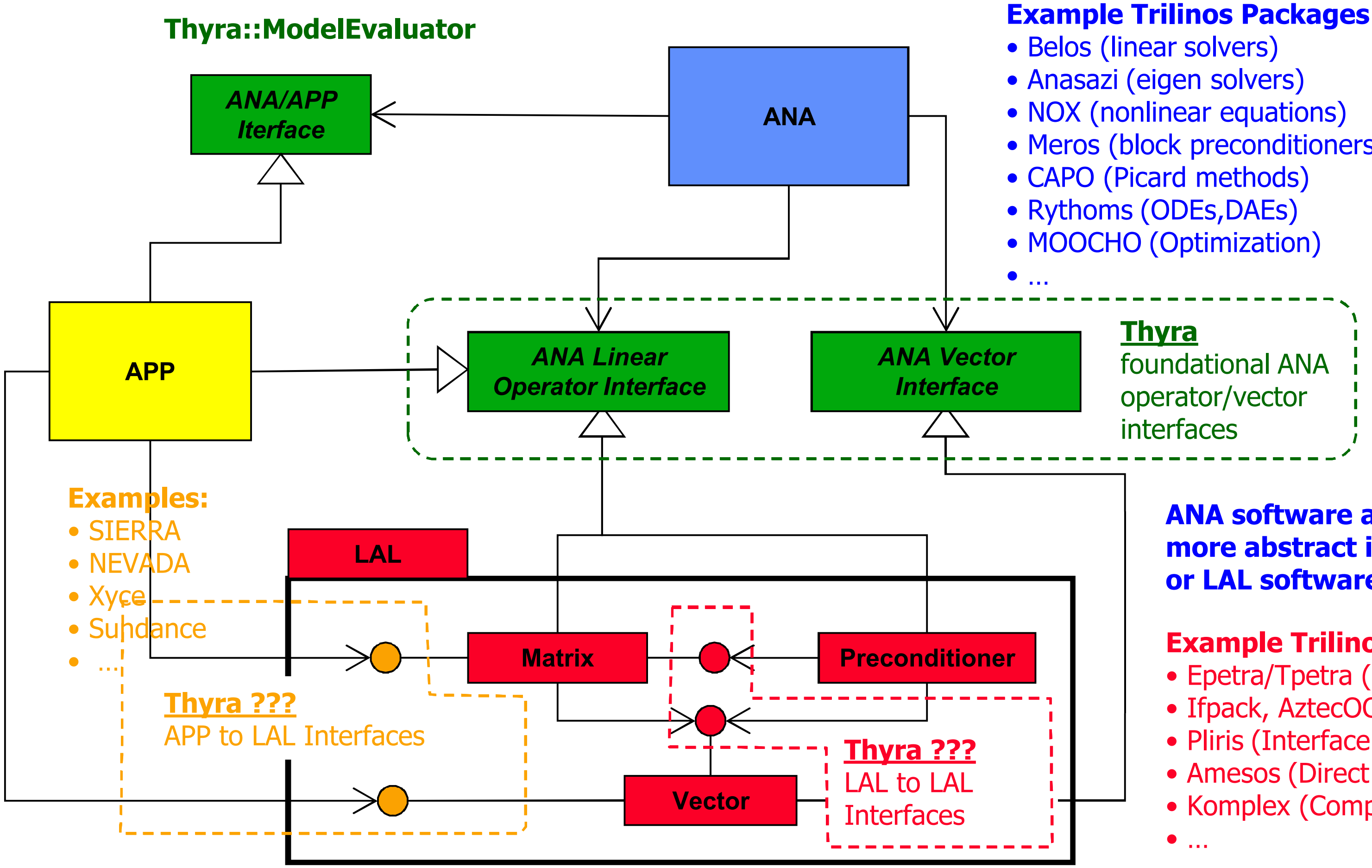
Scalars

- ρ, β, γ, α

Vector spaces?

- $\mathcal{X}$

Software Componentization and Trilinos Interfaces



Example Trilinos Packages:
- Belos (linear solvers)
- Anasazi (eigen solvers)
- NOX (nonlinear equations)
- Meros (block preconditioners)
- CAPO (Picard methods)
- Rythoms (ODEs,DAEs)
- MOOCHO (Optimization)
- …

ANA software allows for a much more abstract interface than APP or LAL software!

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

Three Different Types of Software Components

1) ANA : Abstract Numerical Algorithm (e.g. linear solvers, eigen solvers, 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

Outline

- Overview of Trilinos
- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for interoperability and layering
- Fundamental Thyra ANA operator/vector interfaces
- History behind Thyra
- Use cases and the scope of Thyra
- New to Thyra in Trilinos 7.0
- Wrapping it up

Trilinos Strategic Goals

- **Scalable Solvers**: As problem size and processor counts increase, the cost of the solver will remain a nearly fixed percentage of the total solution time.
- **Hardened Solvers**: Never fail unless problem essentially unsolvable, 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 capabilities from basic linear algebra to transient and optimization solvers.
- ***Grand* Universal Interoperability**: All Trilinos packages will be interoperable, so that any combination of solver packages that makes sense algorithmically will be possible within Trilinos.
- **Universal Solver RAS**: Trilinos will be:
  - Reliable: Leading edge hardened, scalable solutions for each of these applications
  - Available: Integrated into every major application at Sandia
  - Serviceable: Easy to maintain and upgrade within the application environment.

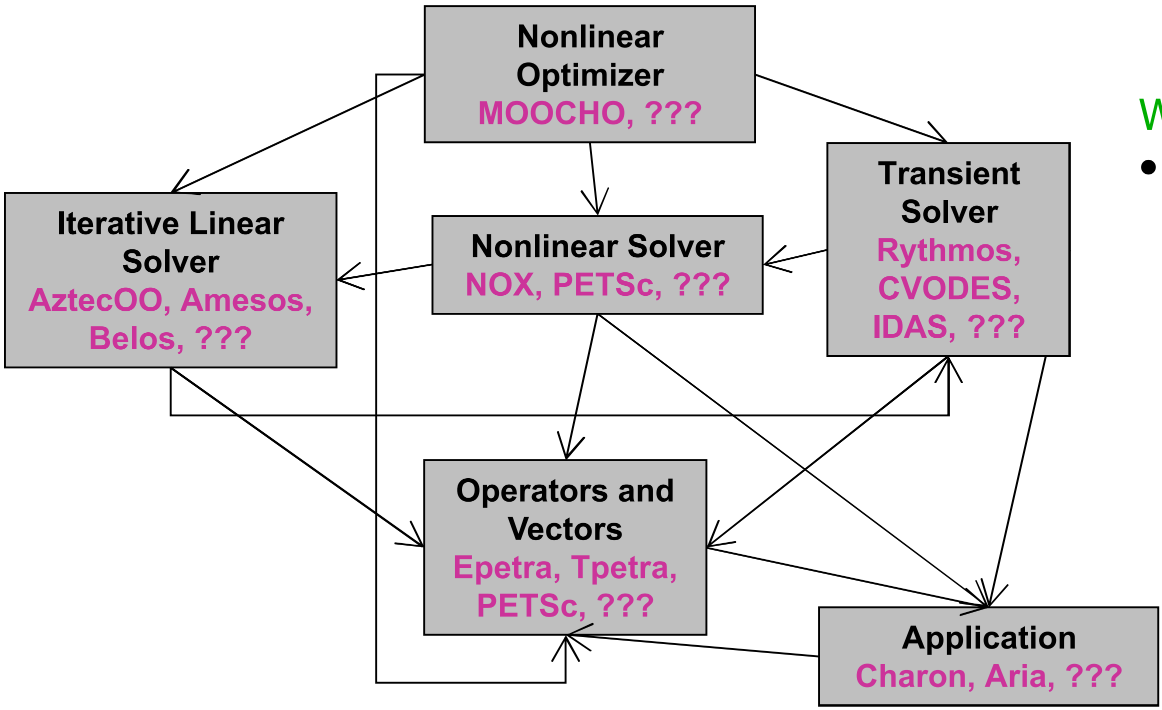
Thyra is being developed to address this issue

Courtesy of Mike Heroux, Trilinos Project Leader

Sandia National Laboratories

Interoperability is Especially Important to Optimization

Numerous interactions exist between layered abstract numerical algorithms (ANAs) in a transient optimization problem



What is needed to solve problem?

- Standard interfaces to break O(N^2) 1-to-1 couplings
  - Operators/vectors
  - Linear Solvers
  - Nonlinear solvers
  - Transient solvers
  - etc.

Thyra is being developed to address interoperability of ANAs

Key Points

- Higher level algorithms, like optimization, require a lot of interoperability
- Interoperability and layering must be “easy” or these configurations will not be achieved in practice

Sandia National Laboratories

Examples of ANA Interoperability and Layering : Rythmos

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

for $x(t) \in \mathbf{R}^n, t \in [t_0, t_f]$

Time Stepper

Advance $x(t_k)$ to $x(t_{k+1})$

where $t_{k+1} = t_k + \Delta t_k$

Implicit Backward Euler method

Solve $f\left(\frac{x_{k+1} - x_k}{\Delta t_k}, x_{k+1}, t_{k+1}\right) = 0$ for x_{k+1}

Nonlinear equations

Solve $r(z) = 0$ for $z \in \mathbf{R}^n$

Newton's method (e.g. NOX)

Choose initial guess z_0, tolerance η

for $k = 0, 1, \ldots$

If "converged" **Stop!**

Solve $\frac{\partial r(z_k)}{\partial z} \delta z_k = -r(z_k)$ for δz

Choose α using a line search method

$z_{k+1} = z_k + \alpha \delta z_k$

end for

Linear equations

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

Preconditioned GMRES

Iterate to "convergence"

$PAx = Pb$

Operator and Preconditioner applications

Apply $y = Ax$

Apply $y = Px$

Matrix-free or Matrix?

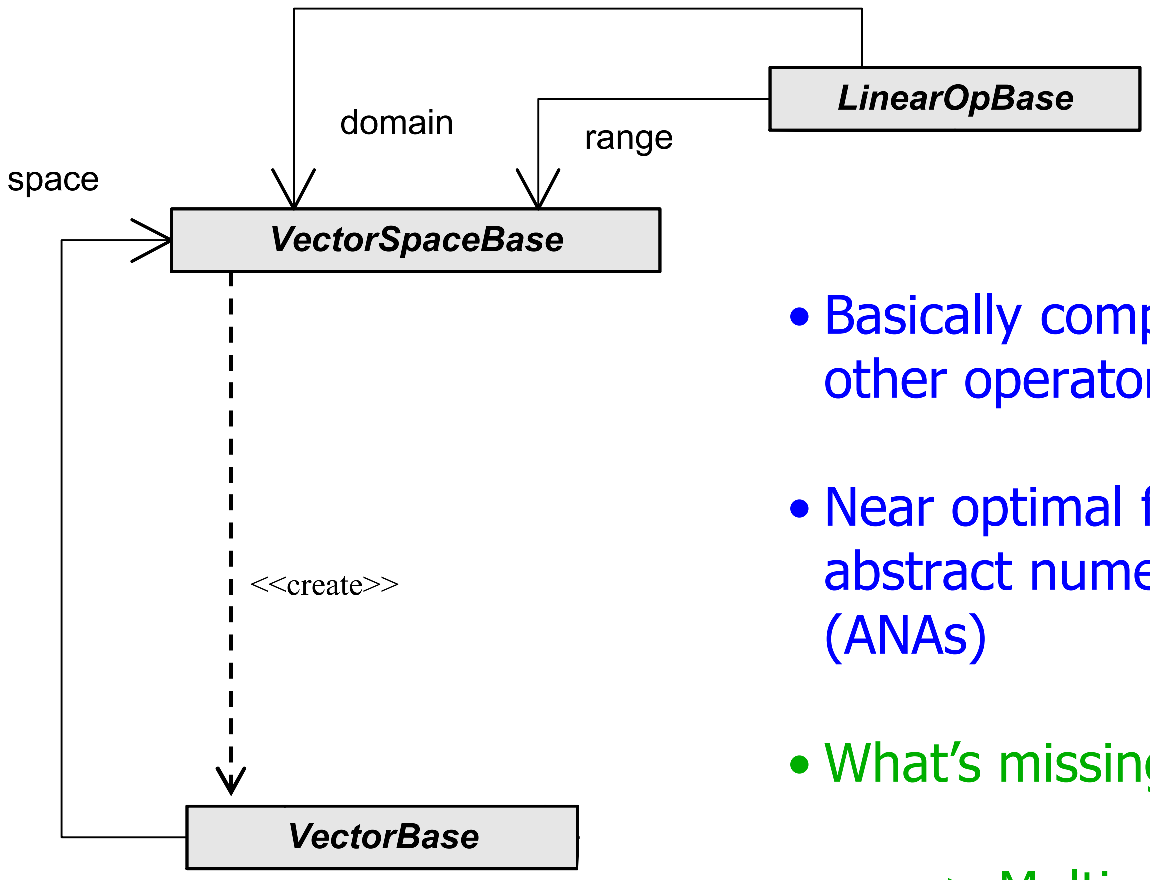
Preconditioners can be defined in many different ways

Sandia National Laboratories

Outline

- Overview of Trilinos
- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for interoperability and layering
- Fundamental Thyra ANA operator/vector interfaces
- History behind Thyra
- Use cases and the scope of Thyra
- New to Thyra in Trilinos 7.0
- Wrapping it up

Fundamental Thyra ANA Operator/Vector Interfaces



- Basically compatible with many other operator/vector interfaces
- Near optimal for many but not all abstract numerical algorithms (ANAs)
- What's missing?

=> Multi-vectors!

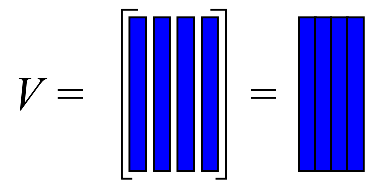
Introducing Multi-Vectors

What is a multi-vector?

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

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

Example: m = 4 columns



What ANAs can exploit multi-vectors?

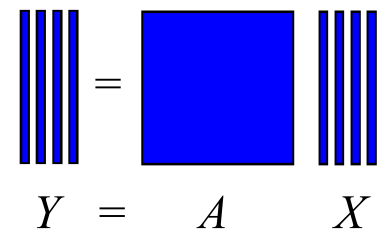
- Block linear solvers (e.g. block GMRES)
- Block eigen solvers (i.e. block Arnoldi)
- Compact limited memory quasi-Newton
- Tensor methods for nonlinear equations

Why are multi-vectors important?

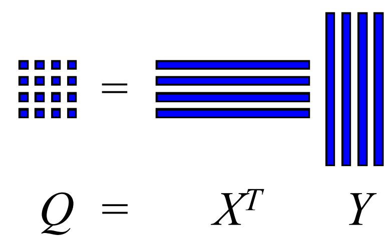
- Cache performance
- Reduce global communication

Examples of multi-vector operations

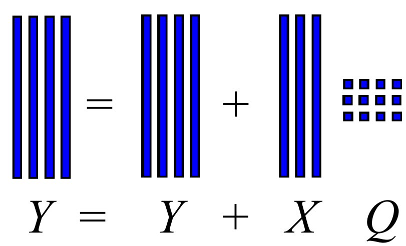
- Operator applications (i.e. mat-vecs)



- Block dot products (m^2)

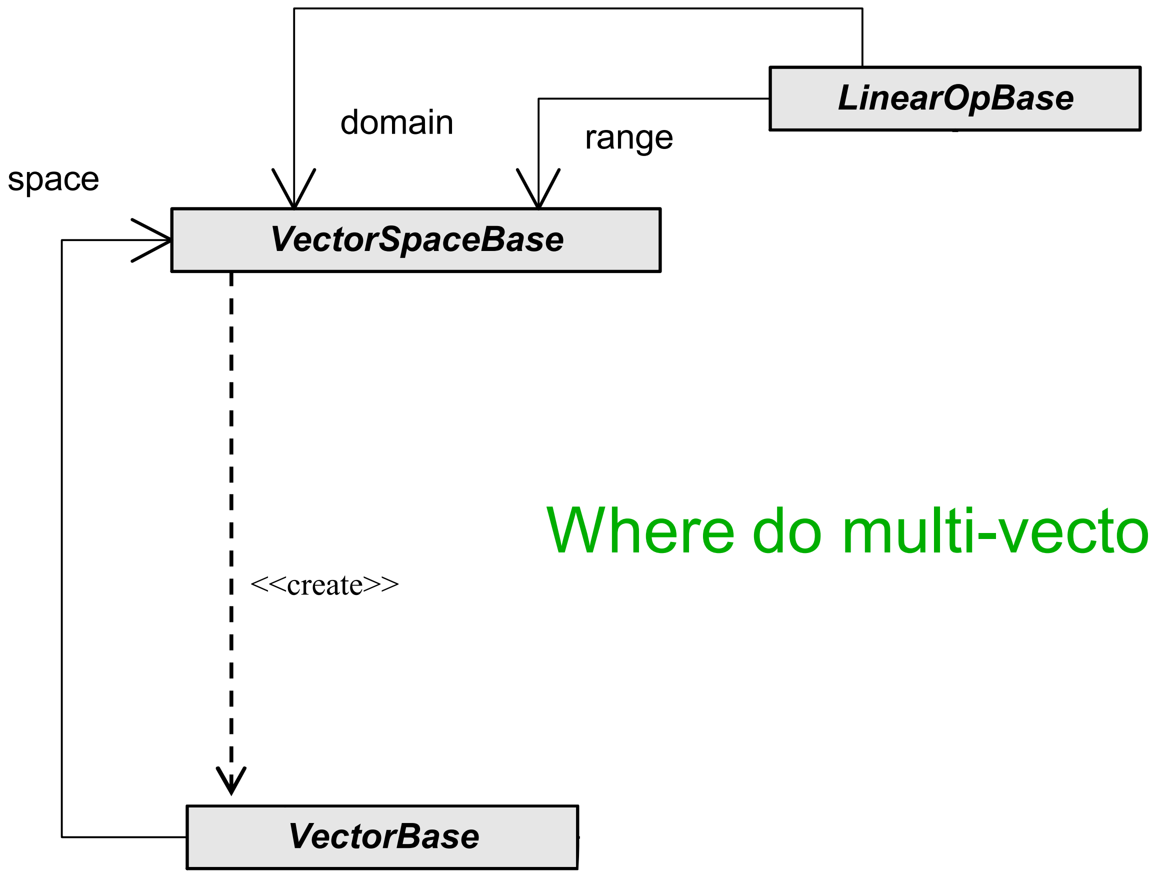


- Block update



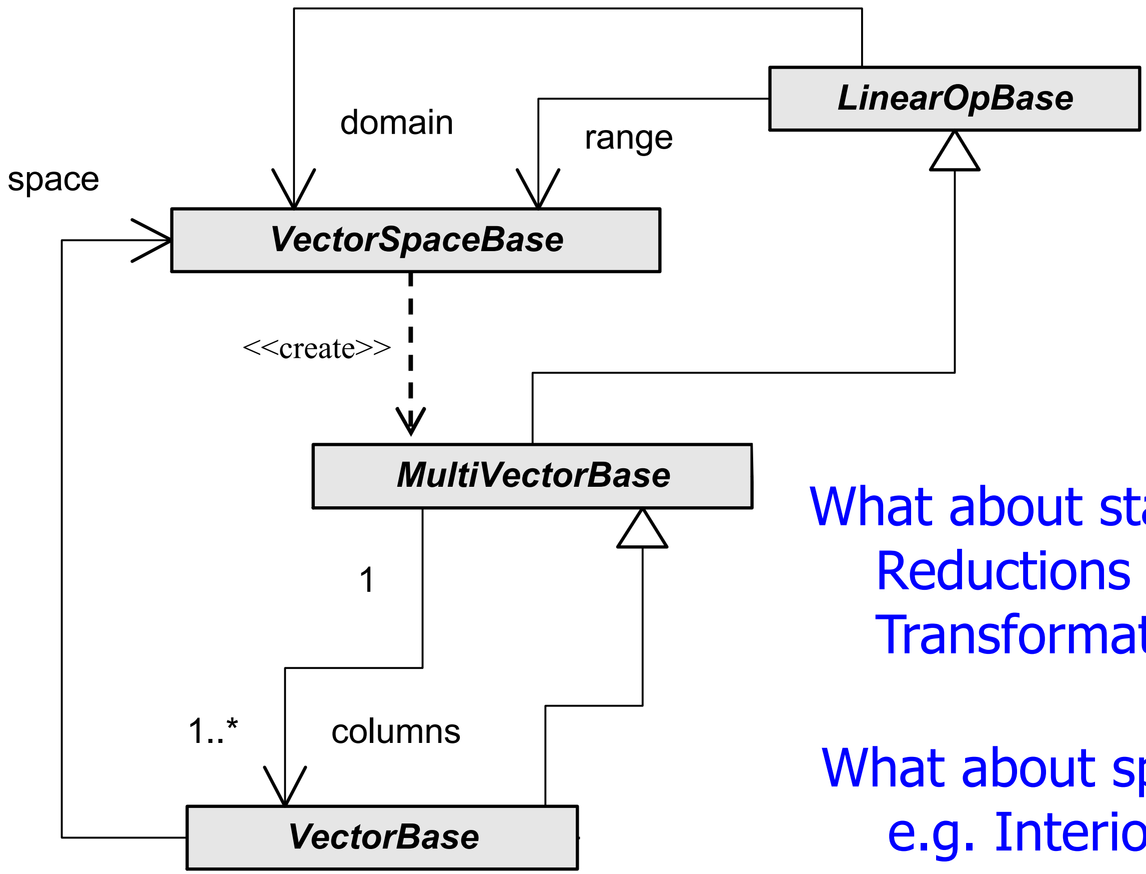
Sandia National Laboratories

Fundamental Thyra ANA Operator/Vector Interfaces



Where do multi-vectors fit in?

Fundamental Thyra ANA Operator/Vector Interfaces



What about standard vector ops?
Reductions (norms, dot etc.)?
Transformations (axpy, scaling etc.)?

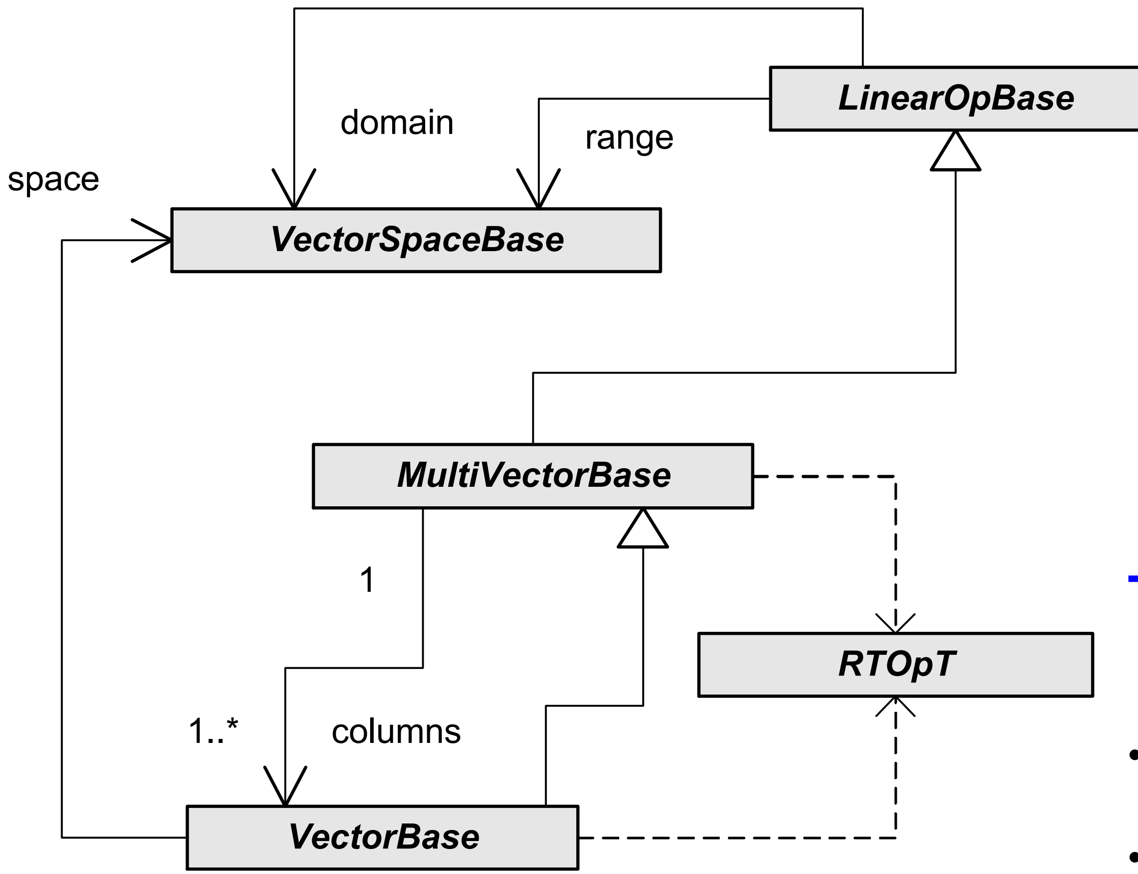
What about specialized vector ops?
e.g. Interior point methods for opt

Key Point

It is easy to come up with a list of 100 or more vector/array operations from a simple literature search into active-set, interior-point, and other algorithms!

Sandia National Laboratories

Fundamental Thyra ANA Operator/Vector Interfaces



A Few Quick Facts about Thyra Interfaces

- All interfaces are expressed as abstract C++ base classes (i.e. object-oriented)
- All interfaces are templated on a `Scalar` data (i.e. generic)

The Key to success!

Reduction/Transformation Operators

- Supports all needed element-wise vector operations
- Data/parallel independence
- Optimal performance

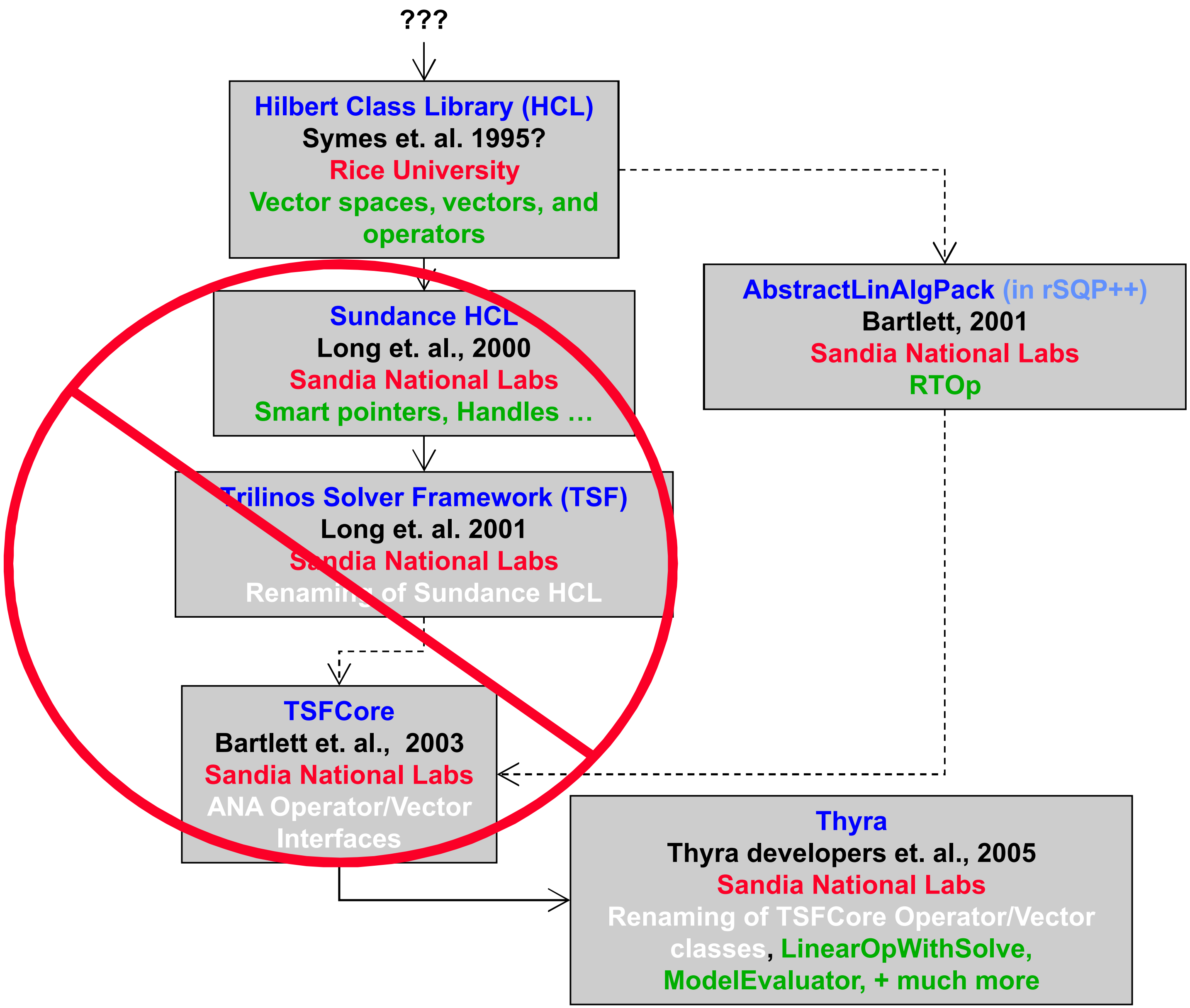
R. A. Bartlett, B. G. van Bloemen Waanders and M. A. Heroux. *Vector Reduction/Transformation Operators*, ACM TOMS, March 2004

Sandia National Laboratories

Outline

- Overview of Trilinos
- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for interoperability and layering
- Fundamental Thyra ANA operator/vector interfaces
- History behind Thyra
- Use cases and the scope of Thyra
- New to Thyra in Trilinos 7.0
- Wrapping it up

History Behind Thyra

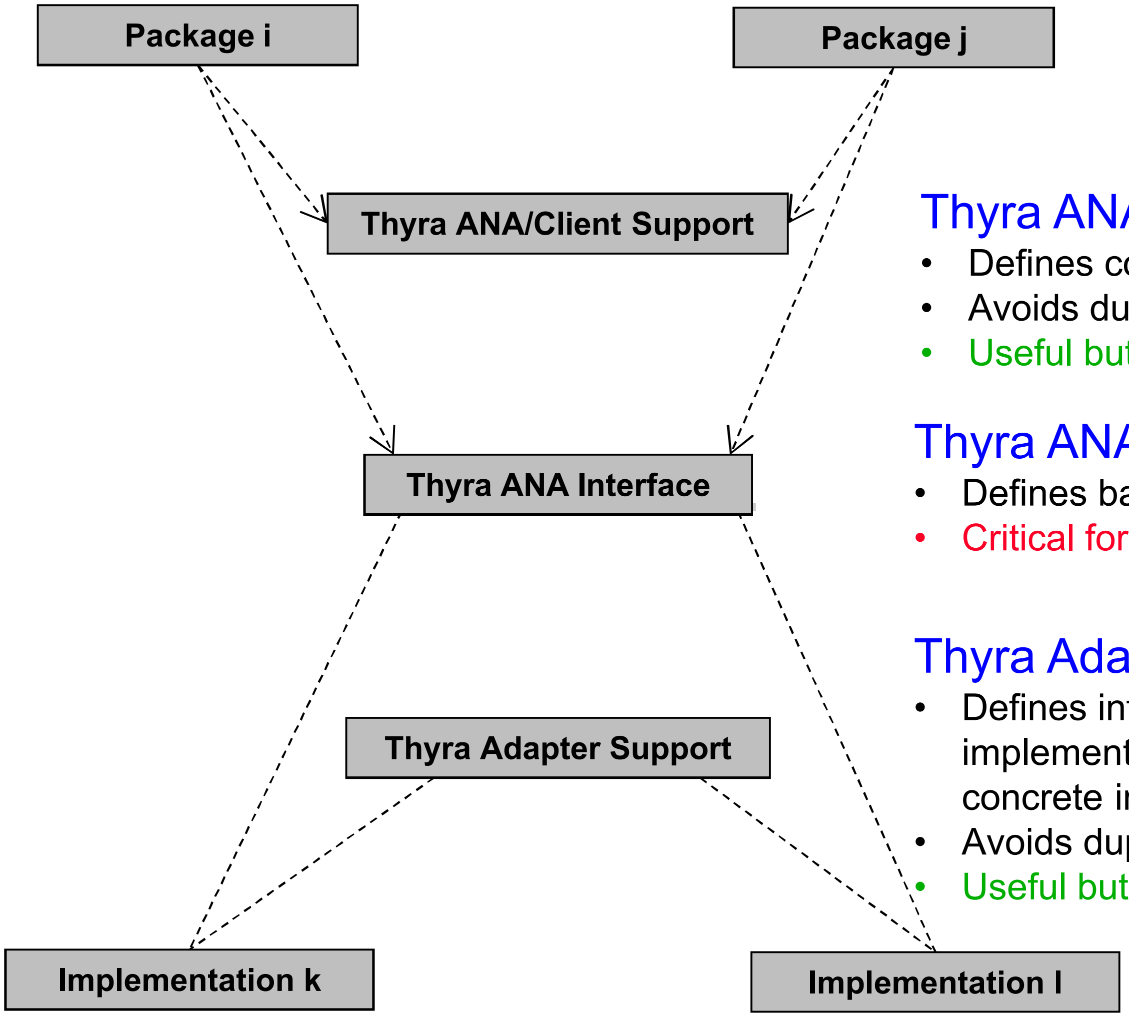


Sandia National Laboratories

Outline

- Overview of Trilinos
- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for interoperability and layering
- Fundamental Thyra ANA operator/vector interfaces
- History behind Thyra
- Use cases and the scope of Thyra
- New to Thyra in Trilinos 7.0
- Wrapping it up

Use Cases and the Scope of Thyra



Thyra ANA/Client Support Software

- Defines conveniences to aid in writing ANAs
- Avoids duplication of effort
- Useful but optional!

Thyra ANA Interoperability Interfaces

- Defines basic functionality needed for ANAs
- Critical for scalable interoperability!

Thyra Adapter Support Software

- Defines infrastructure support and concrete implementations to make it easy to provide concrete implementations for Thyra ANA interfaces
- Avoids duplication of effort
- Useful but optional!

Outline

- Overview of Trilinos
- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for interoperability and layering
- Fundamental ANA operator/vector interfaces
- History behind Thyra
- Use cases and the scope of Thyra
- New to Thyra in Trilinos 7.0 => September 2006?
- Wrapping it up

Outline

- Overview of Trilinos
- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for interoperability and layering
- Fundamental ANA operator/vector interfaces
- History behind Thyra
- Use cases and the scope of Thyra
- New to Thyra in Trilinos 7.0
  - Implicit linear operators and handle classes with operator overloading
  - Linear solves and preconditioners
  - Nonlinear model evaluator
- Wrapping it up

Thyra Implicit ANA Operator/Vector Subclasses (Client Support)

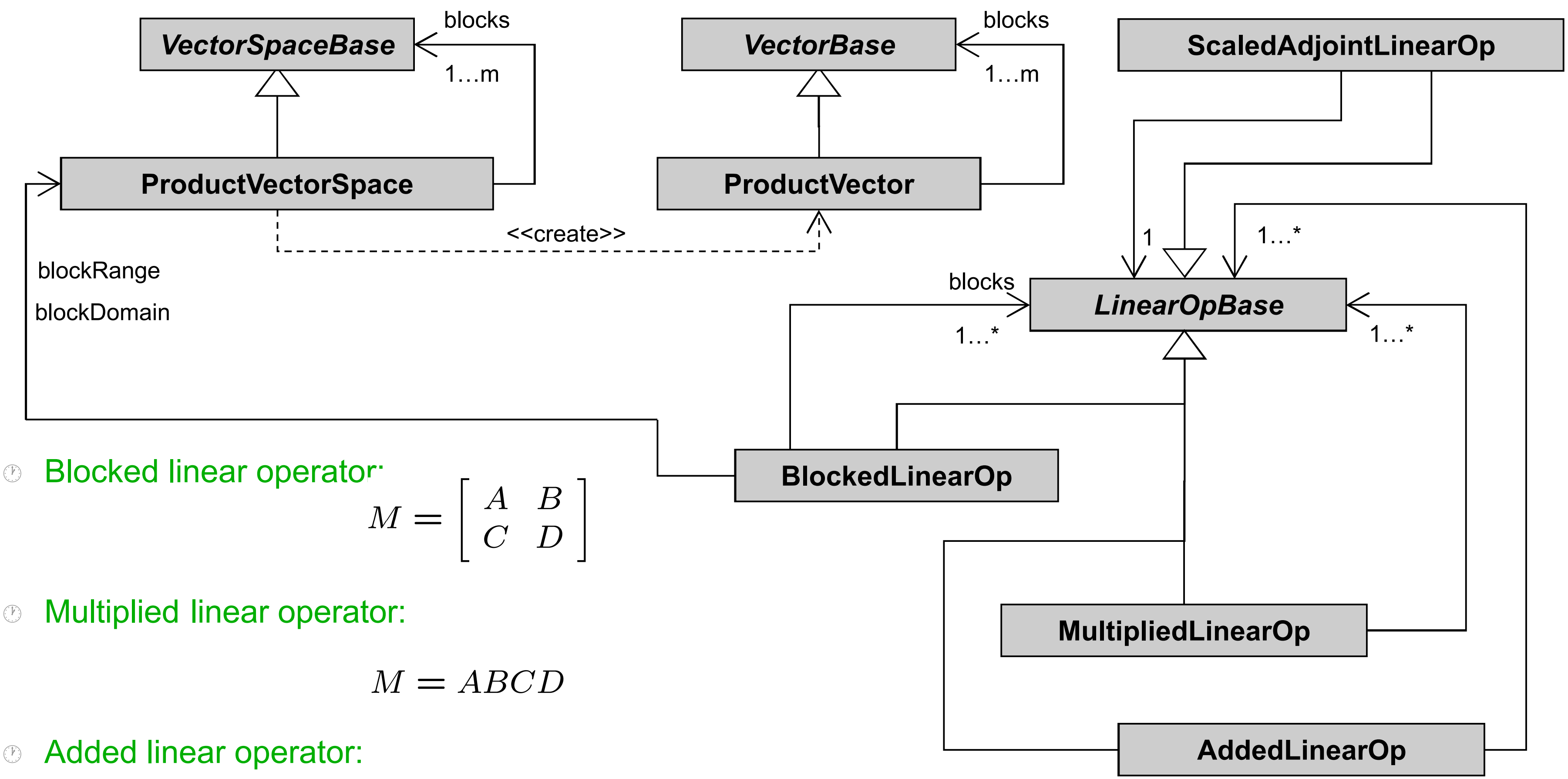
"Composite" subclasses allow a collection of objects to be manipulated as one object

- Product vector spaces and product vectors:
  - Product vector spaces: $\mathcal{X} = \mathcal{V}_1 \times \mathcal{V}_2 \times \ldots \times \mathcal{V}_m$
  - Product vectors: $x^T = \begin{bmatrix} v_1^T & v_2^T & \ldots & v_m^T \end{bmatrix}$

"Decorator" subclasses wrap an object and changes its behavior

- Scaled/Adjoint(transposed) linear operator:

$$M = \alpha A^H$$



- Blocked linear operator:

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

- Multiplied linear operator:

$$M = ABCD$$

- Added linear operator:

$$M = A + B + C + D$$

Sandia National Laboratories

Handle Classes for Matlab-like Linear Algebra (Client Support)

```
template<class Scalar>
bool silliestCgSolve(
  const LinearOperator<Scalar>                              &A
  ,const Vector<Scalar>                                     &b
  ,const int                                                maxNumIters
  ,const typename Teuchos::ScalarTraits<Scalar>::magnitudeType tolerance
  ,Vector<Scalar>                                           x
  )
{
  // Create some typedefs
  …
  // Initialization of the algorithm
  const VectorSpace<Scalar>  space   = A.domain();
  Vector<Scalar>             r       = b - A*x;
  ScalarMag                  r0_nrm = norm(r);
  if(r0_nrm==zero) return true;
  Vector<Scalar>      p(space), q(space);
  Scalar              rho_old = -one;
  // Perform the iterations
  for( int iter = 0; iter <= maxNumIters; ++iter ) {
    // Check convergence and output iteration
    const ScalarMag  r_nrm = norm(r);
    const bool isConverged = (r_nrm/r0_nrm)<=tolerance;
    if( r_nrm/r0_nrm < tolerance ) return true; // Success!
    // Compute the iteration
    const Scalar      rho = inner(r,r);
    if(iter==0)       copyInto(r,p);
    else              p = Scalar(rho/rho_old)*p + r;
    q = A*p;
    const Scalar alpha = rho/inner(p,q);
    x += Scalar(+alpha)*p;
    r += Scalar(-alpha)*q;
    rho_old = rho;
  }
  return false; // Failure
}
```

Key Points

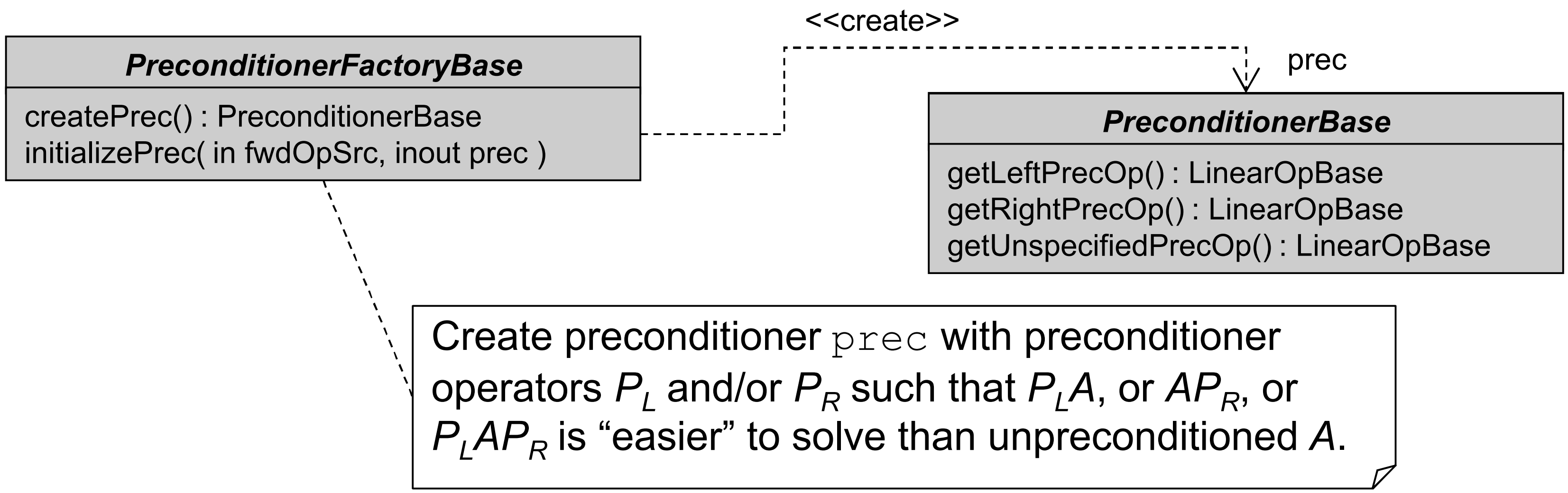
- Handle classes hide memory management
- Matlab-like notation for linear algebra!
  - Template meta-program methods used to reduce operator-overloading overhead and avoid creation of temps.
- Works with any linear operator and vector implementation (e.g. Epetra, PETSc, etc.)
- Works in any computing configuration, i.e. serial, SPMD, client/server etc.!
- Works with any `Scalar` type (i.e. `float`, `double`, `complex<double>`, extended precision, etc.) that has a traits class
- Still some more work to be done
  - Better elimination of some temporaries
  - Support for multi-vectors

Outline

- Overview of Trilinos
- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for interoperability and layering
- Fundamental ANA operator/vector interfaces
- History behind Thyra
- Use cases and the scope of Thyra
- New to Thyra in Trilinos 7.0
  - Implicit linear operators and handle classes with operator overloading
  - Linear solves and preconditioners
  - Nonlinear model evaluator
- Wrapping it up

Preconditioners and Preconditioner Factories

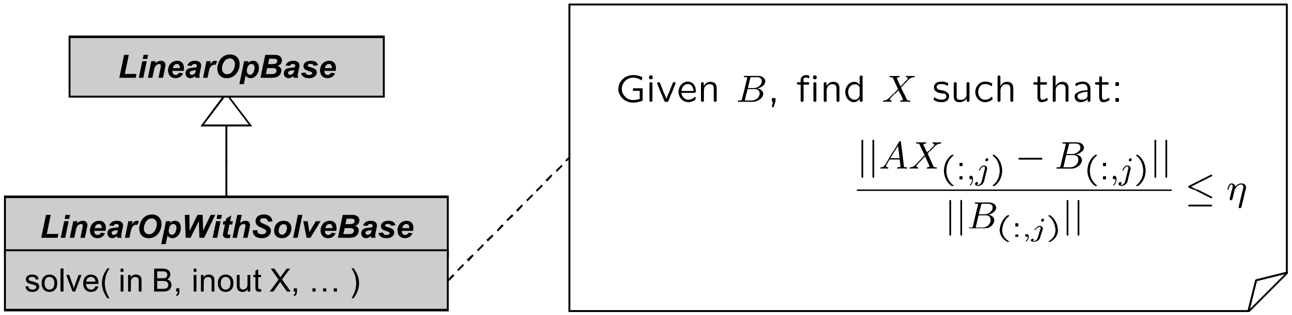
PreconditionerFactoryBase : Creates and initializes **PrecondtionerBase** objects



- Allows unlimited creation/reuse of preconditioner objects
- Supports reuse of factorization structures
- Adapters currently available for Ifpack and ML
- New Stratimikos package provides a singe parameter-driver wrapper for all of these

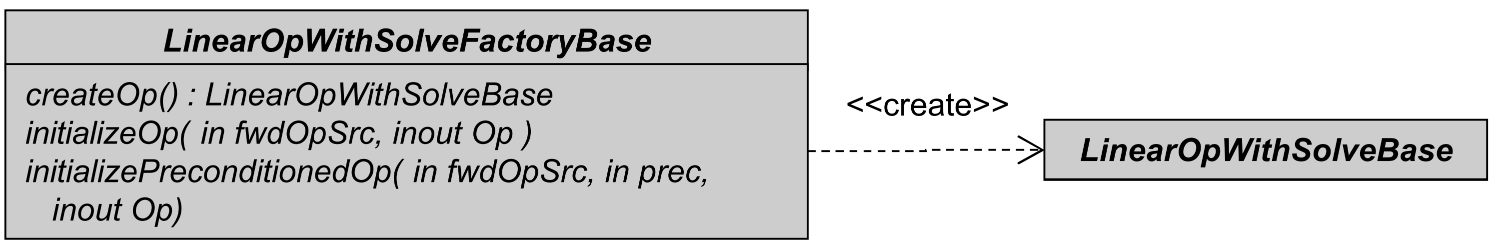
Linear Operator With Solve and Factories

LinearOpWithSolveBase : Combines a linear operator and a linear solver



- Appropriate for both direct and iterative solvers
- Supports multiple simultaneous solutions as multi-vectors
- Allows targeting of different solution criteria to different RHSs
- Supports a “default” solve

LinearOpWithSolveFactoryBase : Uses LinearOpBase objects in initialize LOWSB objects



- Allows unlimited creation/reuse of LinearOpWithSolveBase objects
- Supports reuse of factorizations/preconditioners
- Supports client-created external preconditioners (which are ignored by direct solvers)
- Appropriate for both direct and iterative solvers
- Concrete adaptors for Amesos, AztecOO, and Belos (not released) are available
- New Stratimikos package provides a single parameter-driven wrapper to all of these!

Outline

- Overview of Trilinos
- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for interoperability and layering
- Fundamental ANA operator/vector interfaces
- History behind Thyra
- Use cases and the scope of Thyra
- New to Thyra in Trilinos 7.0
  - Implicit linear operators and handle classes with operator overloading
  - Linear solves and preconditioners
  - Nonlinear model evaluator
- Wrapping it up

Overview of Nonlinear Model Evaluator Interface

Approach: Develop a single, scalable interface to address many different types of numerical problems (e.g. nonlinear equations, stability/bifurcation methods, uncertainty quantification, ODEs/DAEs, optimization …) and combinations of problem types.

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

- (Some) Output functions:
  - State function: $(\dot{x}, x, \{p_l\}, t) \Rightarrow f \in \mathcal{F}$
  - Auxiliary response functions: $(\dot{x}, x, \{p_l\}, t) \Rightarrow g_j \in \mathcal{G}_j$, for $j = 1 \ldots N_g$
  - State/state derivative operator: $(\dot{x}, x, \{p_l\}, t) \Rightarrow W = \alpha \frac{\partial f}{\partial \dot{x}} + \beta \frac{\partial f}{\partial x}$

Key Points

- Flexible/extendable specification of model inputs and outputs
- Address a large number steady-state and transient numerical problems and applications
- Designed for augmentation!

Some Examples of Supported Nonlinear Problem Types

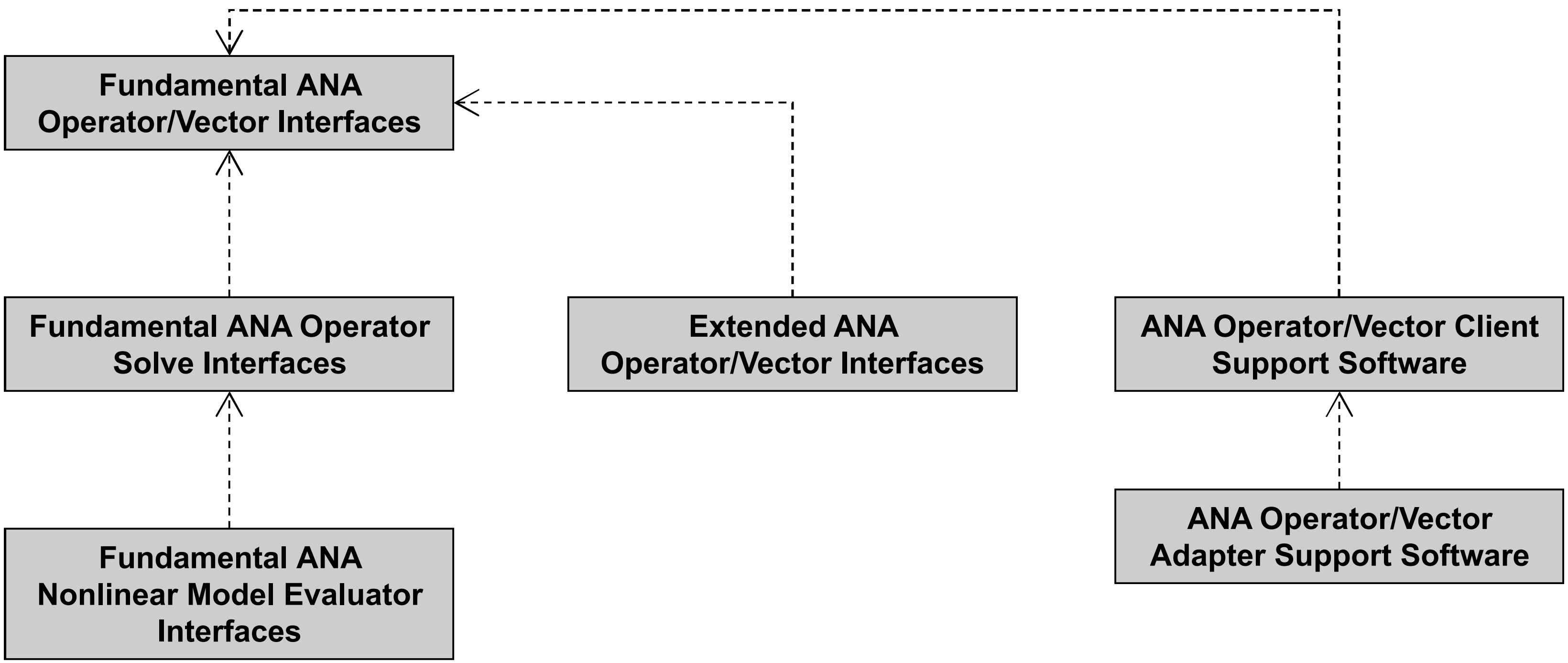
Nonlinear equations:	Solve $f(x) = 0$ for $x \in \mathbf{R}^n$
Stability analysis:	For $f(x,p) = 0$ find space $p \in \mathcal{P}$ such that $\frac{\partial f}{\partial x}$ is singular
Explicit ODEs:	Solve $\dot{x} = f(x,t) = 0, t \in [0,T]$, $x(0) = x_0$, for $x(t) \in \mathbf{R}^n, t \in [0,T]$
DAEs/Implicit ODEs:	Solve $f(\dot{x}(t), x(t), t) = 0, t \in [0,T]$, $x(0) = x_0$, $\dot{x}(0) = x'_0$ for $x(t) \in \mathbf{R}^n, t \in [0,T]$
Explicit ODE Forward Sensitivities:	Find $\frac{\partial x}{\partial p}(t)$ such that: $\dot{x} = f(x,p,t) = 0, t \in [0,T]$, $x(0) = x_0$, for $x(t) \in \mathbf{R}^n, t \in [0,T]$
DAE/Implicit ODE Forward Sensitivities:	Find $\frac{\partial x}{\partial p}(t)$ such that: $f(\dot{x}(t), x(t), p, t) = 0, t \in [0,T]$, $x(0) = x_0$, $\dot{x}(0) = x'_0$, for $x(t) \in \mathbf{R}^n, t \in [0,T]$
Unconstrained Optimization:	Find $p \in \mathbf{R}^m$ that minimizes $g(p)$
Constrained Optimization:	Find $x \in \mathbf{R}^n$ and $p \in \mathbf{R}^m$ that: minimizes $g(x,p)$ such that $f(x,p) = 0$
ODE Constrained Optimization:	Find $x(t) \in \mathbf{R}^n$ in $t \in [0,T]$ and $p \in \mathbf{R}^m$ that: minimizes $\int_0^T g(x(t),p)$ such that $\dot{x} = f(x(t),p,t) = 0$, on $t \in [0,T]$ where $x(0) = x_0$

Sandia National Laboratories

Outline

- Introduction of abstract numerical algorithms (ANAs) and Trilinos software and interfaces
- The need for interoperability and layering
- Fundamental ANA operator/vector interfaces
- History behind Thyra
- Use cases and the scope of Thyra
- New to Thyra in Trilinos 7.0
- Wrapping it up

Dependencies between Thyra Software "Collections"



Key Points

- These interfaces are as minimal as possible and the dependencies between them is carefully regulated!
- The support software is carefully separated from the interoperability interfaces!

Summary

- **Thyra** interfaces provide minimal but efficient connectivity between ANAs and linear algebra implementations and applications
- **Thyra** is the critical standard for interoperability between ANAs in Trilinos
- **Thyra** can be used in Serial/SMP, SPMD, client/server and master/slave
- **Thyra** provides a growing set of optional utilities for ANA development and subclass implementation support
- **Thyra** support for nonlinear ANAs (i.e. the model evaluator) is being developed as well as general support for linear solvers
- **Thyra** interfaces and adapters are provided for preconditioner factories and linear solver factories (Stratimikos)
- **Thyra** adapters are available for Epetra, Amesos, AztecOO, Belos, Anasazi, Rythmos, and MOOCHO with others on the way (e.g. NOX, …)
- **Python/Thyra** wrappers are on the way as well (Bill Spotz)!

Trilinos website

http://software.sandia.gov/trilinos

Sandia National Laboratories