Teko: A Package for Multiphysics Preconditioners

A Trilinos User Talk

November 4th, 2009

Eric C. Cyr
Sandia National Laboratories
Outline

1. What is Teko?
2. Design Requirements and Overview
3. Using Thyra
4. Using Stratimikos
5. Thoughts on Trilinos
6. Summary
What is Teko?

• Teko means “Fuse (v)” in Greek
• Facilitates implementing “block” preconditioners
• Target applications are multiphysics systems
• Similar capabilities exist in Meros
  – Primarily focuses on Navier-Stokes solvers
  – Some utilized Thyra technology not actively supported
Current Teko Capabilities

• Several preconditioners
  - Block 2x2 LU
  - Block Gauss-Seidel
  - Block Jacobi
  - Additive
  - Multiplicative
  - LSC (Navier-Stokes)
  - SIMPLE (Navier-Stokes)

• Thyra used for operator manipulation
• Able to use Trilinos solver/preconditioner capabilities for sub solves
• Blocking capability turns large system into blocked system
What is a block preconditioner?

• MHD Equations are:

\[
\frac{\partial (\rho u)}{\partial t} + \nabla \cdot (\rho u \otimes u + pI + \Pi) - \frac{1}{\mu_0} \nabla \times B \times B = 0
\]

\[
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) = 0
\]

\[
\frac{\partial B}{\partial t} - \nabla \times (u \times B) + \nabla \times (\frac{\eta}{\mu_0} \nabla \times B) = 0
\]

• Discretizing using an implicit time step gives

\[
\begin{bmatrix}
F & B^T \\
B & C \\
Y & 0
\end{bmatrix}
\begin{bmatrix}
Z \\
0
\end{bmatrix}
\begin{bmatrix}
u \\
p \\
b
\end{bmatrix}
=
\begin{bmatrix}
f \\
g \\
h
\end{bmatrix}
\]

• Block Gauss-Seidel requires (approximate) inverses of

\[
\begin{bmatrix}
F & B^T \\
B & C
\end{bmatrix}
\text{ and } D
\]
What are the design requirements?

1. Preconditioners usable by Aztec and Belos solvers
   - Reuse preconditioners in a recursive manner
2. Easily write a preconditioner
   - Blocking and manipulation capability:
     • Abstraction for reuse (I want to use Epetra, Tpetra or PETSc)
     • Specification of high-level algorithms easily
   - Inversion of blocks:
     • Use any preconditioner/solver in Trilinos
     • Easily specified by user
10,000 Feet: Teko Design

Abstraction layer
Concrete packages

Color Code
Abstraction layer
Concrete packages
Teko Example

• Implement simple preconditioner ($\tilde{A}$) for $A$ using Teko

\[
A = \begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix}, \quad \tilde{A} = \begin{bmatrix} P & 0 \\ A_{10} & H \end{bmatrix}
\]

where

\[
P = A_{00} + \alpha A_{01} \text{ and } H = \text{diag}(A_{11})
\]

• Notice preconditioner requires

$\alpha$ and $P^{-1}$
**Teko Example: Factory Definition**

```cpp
// Declaration of the preconditioner factory
class ExamplePreconditionerFactory :
    public Teko::BlockPreconditionerFactory {
public:
    // Constructor
    ExamplePreconditionerFactory(
        const RCP<const Teko::InverseFactory> & inverse, double alpha);
    : inverse_(inverse), alpha_(alpha) {}

    // Function inherited from Teko::BlockPreconditionerFactory
    Teko::LinearOp
    buildPreconditionerOperator(Teko::BlockedLinearOp & blo,
                                Teko::BlockPreconditionerState & state) const;

protected:
    // class members
    RCP<const Teko::InverseFactory> inverse_;
    double alpha_;
};
```

- **Teko uses Thyra’s preconditioner abstraction: use Aztec and Belos**
- **Teko simplifies implementing preconditioners by introducing an abstraction**
Recall:

\[
A = \begin{bmatrix}
A_{00} & A_{01} \\
A_{10} & A_{11}
\end{bmatrix}
\]

\[
H = \text{diag}(A_{11})
\]

\[
P = A_{00} + \alpha A_{01}
\]

\[
\tilde{A} = \begin{bmatrix}
P & 0 \\
A_{10} & H
\end{bmatrix}
\]
How is Thyra being used?

- All linear operator objects are based on Thrya
- Abstract Numerical Algorithms (ANA) idea permits expression of operations (+, -, *, etc...)
- Reuse of different linear algebra packages (in principle)
- Linear operator abstraction makes handling composite operators (relatively) transparent

\[
\begin{bmatrix}
F & B^T \\
B & C \\
Y & 0
\end{bmatrix}
\begin{bmatrix}
Z \\
0 \\
D
\end{bmatrix}
\]
How is Thyra being used?

• I’ve focused on small set of Thyra operators to use
  – Thyra::LinearOpBase and Thyra::PhysicallyBlockedLinearOpBase
  – Provide limited set of conversion functions

• In Teko RCPs are typedefed away

  typedef Teuchos::RCP<Thyra::PhysicallyBlockedLinearOpBase> BlockedLinearOp;
  typedef Teuchos::RCP<const Thyra::LinearOpBase> LinearOp;
  typedef Teuchos::RCP<Thyra::LinearOpBase> ModifiableLinearOp;

  – Tried to hide complexity of RCPs
  – I think RCPs are great (users should get used to them)
  – I may revert back
How is Thyra being used?

Some issues with Thyra

• What use cases does Thyra support?
  – Need diagonals
  – Need rows
  – Need explicit matrix products and sums
  – Is it more than a matrix-vector multiply?

• Complexity of interfaces and software architecture
  – Teko provides abstractions and wrapper functions
  – Depth of hierarchy can be overwhelming
What about inverses?

• Teko provides functionality for building inverses
  – Pair an “inverse factory” with linear operator
    
    RCP<const Teko::InverseFactory> inverse = ...  
    Teko::LinearOp P = ...  
    const Teko::LinearOp invP = Teko::buildInverse(*inverse,P);
  
  – Inverse factory abstraction makes building inverses easy

• How are inverse factories created and specified?
  – Inverse library object builds “inverse factories”
  – Library can be specified through an XML file
Building Inverse Libraries

- Inverse Libraries specified using parameter list
- Built on top of Stratimikos

```xml
<Parameter name="Use Preconditioner" type="string" value="GS-Prec"/>
<ParameterList name="Inverse Factory Library">
  <ParameterList name="GS-Prec">
    <Parameter name="Type" type="string" value="Block Gauss-Seidel"/>
    <Parameter name="Inverse Type 1" type="string" value="Diagonal 1"/>
    <Parameter name="Inverse Type 2" type="string" value="Amesos"/>
  </ParameterList>
  <ParameterList name="Diagonal 1">
    <Parameter name="Type" type="string" value="Ifpack"/>
    <ParameterList name="Ifpack Settings">
      <Parameter name="schwarz: reordering type" type="string" value="rcm"/>
      <Parameter name="fact: level-of-fill" type="int" value="2"/>
    </ParameterList>
  </ParameterList>
</ParameterList>
```
Building Inverse Libraries

- Can build inverse library from XML list
- Uses Stratmikos to create solvers/preconditioners
- Inverse factory wraps `Thyra::LOWSFactoryBase` and `Thyra::PreconditionerFactoryBase`
- Inverse library builds sub-inverses as required
About Stratimikos

- Provides uniform interface for building Trilinos solvers/preconditioners
- Builds and applies inverse operator
- Distinguishes between a solver and a preconditioner
  - Inverse library hides this distinction
  - Value of this distinction in my case is debatable
- Building of multiple “inverses” of same type is tricky
  - Teko builds new Stratimikos object for each “inverse”
Blocking an Epetra_CRSMatrix

- Not all multiphysics matrices are (currently) blocked
- Teko provides capability for blocking a “strided” operator

\[
\begin{bmatrix}
A
\end{bmatrix} \rightarrow \begin{bmatrix}
F & B^T \\
B & C
\end{bmatrix}
\]

\[
\begin{bmatrix}
u_1 & v_1 & p_1 \\
u_2 & v_2 & p_2 \\
\ldots & \ldots & \ldots \\
u_8 & v_8 & p_8
\end{bmatrix}
\]
General Trilinos Comments

• I like CMake
  – Built before with autotools
  – Still don’t like Makefile.export, good documentation is preferred
  – Building Teko with CMake reasonably easy

• I dislike
  – “You can browse all of <package name> as a single doxygen collection. Warning: This is not the recommended way to learn about <package name> software.”

• Experience with documentation is mixed
  – Documentation of parameter lists
Summary

1. Teko is package for multiphysics preconditioners
2. Sits on top of Thyra and Stratimikos
   - Thyra for Abstract Numerical Algorithms operations
   - Stratimikos for interfacing to Trilinos solver and preconditioner libraries
3. Teko will be available in “dev” branch soon (hopefully!)