Belos:
Next-Generation Iterative Solvers

2009 Trilinos User Group Meeting
November 4, 2009

Chris Baker
David Day
Mike Heroux
Mike Parks
Heidi Thornquist (Lead)
Outline

- What is Belos?
  - Solvers
  - Belos: What’s in a name?
  - Solver framework structure
  - Simple example
- Spotlight on “Recycling” Solvers
  - Why recycle?
  - Examples
  - Structure of recycling solver
- Summary
What is Belos?

- Solve $Ax=b$ where $A$ large, sparse. Matrix-free.
- Next-generation linear solver library (templated C++)
- Provide generic solver framework solution of large-scale linear systems
- Belos provides solvers for:
  - Single RHS: $Ax = b$
  - Multiple RHS (available simultaneously): $AX = B$
  - Multiple RHS (available sequentially): $Ax_i = b_i, \ i=1,\ldots,k$
  - Sequential Linear systems: $A_i x_i = b_i, \ i=1,\ldots,k$
- Leverage research advances of solver community:
  - Block methods: block GMRES [Vital], block CG/BICG [O’Leary]
  - “Seed” solvers: hybrid GMRES [Nachtigal, et al.]
  - “Recycling” solvers for sequences of linear systems [Parks, et al.]
  - Restarting, orthogonalization techniques
- Belos solver components are: interoperable, extensible, reusable
- **Block linear solvers** $\rightarrow$ **Better multicore performance**
- Multiprecision capability (via Tpetra)
Solvers

- **Hermitian Systems** \((A = A^H)\)
  - Block CG
  - Pseudo-Block CG (Perform single-vector algorithm simultaneously)
  - RCG (Recycling Conjugate Gradients)
  - PCPG (Projected CG)

- **Non-Hermitian System** \((A \neq A^H)\)
  - Block GMRES
  - Pseudo-Block GMRES (Perform single-vector algorithm simultaneously)
  - Block FGMRES (Variable preconditioner)
  - Hybrid GMRES
  - TFQMR
  - GCRODR (Recycling GMRES)
Belos: What’s in a name?

Let’s just stick to the linear algebra…
$x^{(0)}$ is an initial guess

for $j = 1, 2, ...$
    Solve $r$ from $Mr = b - Ax^{(0)}$
    $v^{(1)} = r/\|r\|_2$
    $s := \|r\|_2 e_1$
    for $i = 1, 2, ..., m$
        Solve $w$ from $Mw = Av^{(i)}$
        for $k = 1, ..., i$
            $h_{k,i} = (w, v^{(k)})$
            $w = w - h_{k,i} v^{(k)}$
        end
        $h_{i+1,i} = \|w\|_2$
        $v^{(i+1)} = w/h_{i+1,i}$
        apply $J_1, ..., J_{i-1}$ on $(h_{1,i}, ..., h_{i+1,i})$
        construct $J_i$, acting on $i$th and $(i + 1)$st component of $h_{*,i}$, such that $(i + 1)$st component of $J_i h_{*,i}$ is 0
        $s := J_i s$
        if $s(i+1)$ is small enough then (UPDATE($\tilde{x}, i$) and quit)
    end
end
UPDATE($\tilde{x}, m$)
end

GMRES
Belos Structure

SolverManager Class

\[ x^{(0)} \text{ is an initial guess} \]
\[
\text{for } j = 1, 2, \ldots \]
\[
\text{solve } r \text{ from } Mr = b - Ax^{(0)} \]
\[
v^{(1)} = r / \|r\|_2 \]
\[
s := \|r\|_2 e_1 \]
\[
\text{for } i = 1, 2, \ldots, m \]
\[
\text{solve } w \text{ from } Mw = Av^{(i)} \]
\[
\text{for } k = 1, \ldots, i \]
\[
h_{k,i} = (w, v^{(k)}) \]
\[
w = w - h_{k,i} v^{(k)} \]
\[
\text{end} \]
\[
h_{i+1,i} = \|w\|_2 \]
\[
v^{(i+1)} = w / h_{i+1,i} \]
\[
\text{apply } J_1, \ldots, J_{i-1} \text{ on } (h_{1,i}, \ldots, h_{i+1,i}) \]
\[
\text{construct } J_i, \text{ acting on } i\text{th and } (i+1)\text{st component} \]
\[
of h_{i,i}, \text{ such that } (i+1)\text{st component of } J_i h_{i,i} \text{ is 0} \]
\[
s := J_i s \]
\[
\text{if } s(i+1) \text{ is small enough then } (\text{UPDATE}(\tilde{x}, i) \text{ and quit}) \]
\[
\text{end} \]
\[
\text{UPDATE}(\tilde{x}, m) \]
\[
\text{end} \]
$x^{(0)}$ is an initial guess
for $j = 1, 2, \ldots$,

Solve $r$ from $Mr = b - Ax^{(0)}$
$v^{(1)} = r / \|r\|_2$
s := $\|r\|_2 e_1$
for $i = 1, 2, \ldots, m$

Solve $w$ from $Mw = Av^{(i)}$
for $k = 1, \ldots, i$

$h_{k,i} = (w, v^{(k)})$

\[ w = w - h_{k,i} v^{(k)} \]

$w_{i+1,i} = \|w\|_2$
$v^{(i+1)} = w / w_{i+1,i}$

apply $J_1, \ldots, J_{i-1}$ on $(h_{1,i}, \ldots, h_{i+1,i})$

construct $J_i$, acting on $i$th and $(i+1)$st components of $h_{i,i}$, such that $(i+1)$st component of $J_i h_{i,i}$ is 0

$s := J_i s$

if $s(i+1)$ is small enough then (UPDATE($x$, $i$) and quit)

end

UPDATE($x$, $m$)

end
Belos Structure

SolverManager Class

LinearProblem, Operator Classes

Iteration Class

\(x^{(0)}\) is an initial guess

for \(j = 1, 2, \ldots\)

Solve \(r\) from \(Mr = b - Ax^{(0)}\)

\(v^{(1)} = r/\|r\|_2\)

\(s := \|r\|_2 e_1\)

for \(i = 1, 2, \ldots, m\)

Solve \(w\) from \(Mw = Av^{(i)}\)

for \(k = 1, \ldots, i\)

\(h_{k,i} = (w, v^{(k)})\)

\(w = w - h_{k,i} v^{(k)}\)

\(h_{i+1,i} = \|w\|_2\)

\(v^{(i+1)} = w/h_{i+1,i}\)

apply \(J_1, \ldots, J_{i-1}\) on \((h_{1,i}, \ldots, h_{i+1,i})\)

construct \(J_i\), acting on \(i\)th and \((i+1)\)st component of \(h_{i,i}\), such that \((i+1)\)st component of \(J_i h_{i,i}\) is 0

\(s := J_i s\)

if \(s(i+1)\) is small enough then (UPDATE(\(\tilde{x}\), \(i\)) and quit)

end

UPDATE(\(\tilde{x}\), \(m\))

\n
Sandia National Laboratories
Belos Structure

**SolverManager Class**

**LinearProblem, Operator Classes**

**Iteration Class**

**OrthoManager Class** (ICGS, IMGS, DGKS)

---

$x^{(0)}$ is an initial guess

for $j = 1, 2, \ldots$

Solve $r$ from $Mr = b - Ax^{(0)}$

$v^{(1)} = r / ||r||_2$

$s := ||r||_2 e_1$

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

Solve $w$ from $Mw = Av^{(i)}$

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

$h_{k,i} = (w, v^{(k)})$

$w = w - h_{k,i}v^{(k)}$

$||w||_2$

$v^{(i+1)} = w / h_{i+1,i}$

apply $J_1, \ldots, J_{i-1}$ on $(h_{1,i}, \ldots, h_{i+1,i})$

construct $J_i$, acting on $i$th and $(i + 1)$st component of $h_i$, such that $(i + 1)$st component of $J_i h_{i,i}$ is 0

$s := J_i s$

if $s(i)$ is small enough then (UPDATE($\tilde{x}, i$) and quit)

UPDATE($\tilde{x}, m$)

---
Belos Structure

SolverManager Class
LinearProblem, Operator Classes
Iteration Class
OrthoManager Class (ICGS, IMGS, DGKs)

$x^{(0)}$ is an initial guess
for $j = 1, 2, ...$

Solve $r$ from $Mr = b - Ax^{(0)}$
$v^{(1)} = r / ||r||_2$
$s := ||r||_2 e_1$

for $i = 1, 2, ..., m$

Solve $w$ from $Mw = Av^{(i)}$

for $k = 1, ..., i$

$h_{k,i} = (w, v^{(k)})$

$w = w - h_{k,i} v^{(k)}$

$\|w\|_2$
$v^{(i+1)} = w / h_{i+1,i}$

apply $J_1, ..., J_{i-1}$ on $(h_{1,i}, ..., h_{i+1,i})$

construct $J_i$, acting on $i$th and $(i+1)$st component of $h_{i,i}$, such that $(i+1)$st component of $J_i h_{i,i}$ is 0

$s := J_i s$

if $s(i+1)$ is small enough then (UPDATE($\hat{x}, i$) and quit)

UPDATE($\hat{x}, m$)

end
Belos Structure

SolverManager Class

LinearProblem, Operator Classes

Iteration Class

OrthoManager Class (ICGS, IMGS, DGKS)

StatusTest Class

OutputManager Class

$x^{(0)}$ is an initial guess
for $j = 1, 2, ...$

Solve $r$ from $Mr = b - Ax^{(0)}$

$v^{(1)} = r/\|r\|_2$

$s := \|r\|_2 e_1$

for $i = 1, 2, ..., m$

Solve $w$ from $Mw = Av^{(i)}$

for $k = 1, ..., i$

$h_{k,i} = (w, v^{(k)})$

$w = w - h_{k,i}v^{(k)}$

$h_{i+1,i} = \|w\|_2$

$v^{(i+1)} = w/h_{i+1,i}$

apply $J_1, ..., J_{i-1}$ on $(h_{1,i}, ..., h_{i+1,i})$

construct $J_i$, acting on $i$th and $(i + 1)$st component of $h_{i,i}$, such that $(i + 1)$st component of $J_i h_{i,i}$ is 0

$s := J_i s$

if $s(i+1)$ is small enough then ($UPDATE(\tilde{x}, i)$ and quit)

end

UPDATE($\tilde{x}, m$)

end
Example (Step #1 – Initialize System)

```c
int main(int argc, char *argv[]) {
    MPI_Init(&argc,&argv);
    Epetra_MpiComm Comm(MPI_COMM_WORLD);
    int MyPID = Comm.MyPID();

    typedef double ST;
    typedef Teuchos::ScalarTraits<ST> SCT;
    typedef SCT::magnitudeType MT;
    typedef Epetra_MultiVector MV;
    typedef Epetra_Operator OP;
    typedef Belos::MultiVecTraits<ST,MV> MVT;
    typedef Belos::OperatorTraits<ST,MV,OP> OPT;

    using Teuchos::ParameterList;
    using Teuchos::RCP;
    using Teuchos::rcp;

    // Get the problem
    std::string filename("orsirr1.hb");
    RCP<Epetra_Map> Map;
    RCP<Epetra_CrsMatrix> A;
    RCP<Epetra_MultiVector> B, X;
    RCP<Epetra_Vector> vecB, vecX;
    EpetraExt::readEpetraLinearSystem(filename, Comm, &A, &Map, &vecX, &vecB);
    X = Teuchos::rcp_implicit_cast<Epetra_MultiVector>(vecX);
    B = Teuchos::rcp_implicit_cast<Epetra_MultiVector>(vecB);
}
```

Parameters for Templates

Get linear system from disk

Trilinos/packages/belos/epetra/example/BlockGmres/BlockGmresEpetraExFile.cpp
Example (Step #2 – Solver Params)

```c++
bool verbose = false, debug = false, proc_verbose = false;
int frequency = -1; // frequency of status test output.
int blocksize = 1; // block size
int numrhs = 1; // number of right-hand sides to solve for
int maxiters = 100; // maximum number of iterations allowed
int maxsubspace = 50; // maximum number of blocks
int maxrestarts = 15; // number of restarts allowed
MT tol = 1.0e-5; // relative residual tolerance

const int NumGlobalElements = B->GlobalLength();

ParameterList belosList;
benosList.set( "Num Blocks", maxsubspace ); // Maximum number of blocks in Krylov factorization
belosList.set( "Block Size", blocksize ); // Block size to be used by iterative solver
belosList.set( "Maximum Iterations", maxiters ); // Maximum number of iterations allowed
belosList.set( "Maximum Restarts", maxrestarts ); // Maximum number of restarts allowed
belosList.set( "Convergence Tolerance", tol ); // Relative convergence tolerance requested
int verbosity = Belos::Errors + Belos::Warnings;
if (verbose) {
    verbosity += Belos::TimingDetails + Belos::StatusTestDetails;
    if (frequency > 0)
        belosList.set( "Output Frequency", frequency );
}
if (debug) {
    verbosity += Belos::Debug;
}
benosList.set( "Verbosity", verbosity );
```

Trilinos/packages/belos/epetra/example/BlockGmres/BlockGmresEpetraExFile.cpp
Example (Step #3 – Solve)

// Construct linear problem instance.
Belos::LinearProblem<double,MV,OP> problem( A, X, B );
bool set = problem.setProblem();
if (set == false) {
    std::cout << std::endl << "ERROR: Belos::LinearProblem failed to set up correctly!" << std::endl;
    return -1;
}

// Start block GMRES iteration
Belos::OutputManager<double> My_OM();
// Create solver manager.
RCP< Belos::SolverManager<double,MV,OP> > newSolver =
    rcp( new Belos::BlockGmresSolMgr<double,MV,OP>(rcp(&problem,false), rcp(&belosList,false)));
// Solve
Belos::ReturnType ret = newSolver->solve();
if (ret!=Belos::Converged) {
    std::cout << std::endl << "ERROR: Belos did not converge!" << std::endl;
    return -1;
}
std::cout << std::endl << "SUCCESS: Belos converged!" << std::endl;
return 0;
Spotlight on Recycling
Sequences of Linear Systems

- Consider sequence of linear systems

\[ A^{(i)}x^{(i)} = b^{(i)} \quad i=1,2,3,\ldots \]

- Applications:
  - Newton/Broyden method for nonlinear equations
  - Materials science and computational physics
  - Transient circuit simulation
  - Crack propagation
  - Optical tomography
  - Topology optimization
  - Large-scale fracture in disordered materials
  - Electronic structure calculations
  - Stochastic finite element methods

- Iterative (Krylov) methods build search space and select optimal solution from that space

- **Building search space is dominant cost**

- For sequences of systems, get fast convergence rate and good initial guess immediately by **recycling** selected search spaces from previous systems
Why Recycle?

- Typically, dominant subspace exists such that almost any Krylov space (from any starting vector) has large components in that space (why restarting is bad)

"Superlinear" Convergence
Why Recycle?

- Typically, dominant subspace exists such that almost any Krylov space (from any starting vector) has large components in that space (why restarting is bad)
- Optimality derives from orthogonal projection
  - new search directions should be far from this dominant subspace for fast convergence
- If such a dominant subspace persists (approximately) from one system to the next, it can be recycled
  - Typically true when changes to problem are small and/or highly localized

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Off-the-shelf solver</th>
<th>Recycling Solver</th>
<th>Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>GMRES</td>
<td>GCRODR</td>
<td>Trilinos 8</td>
</tr>
<tr>
<td>SPD</td>
<td>CG</td>
<td>Recycling CG (RCG)</td>
<td>Trilinos 10</td>
</tr>
<tr>
<td>Symmetric Indefinite</td>
<td>MINRES</td>
<td>Recycling MINRES (RMINRES)</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Deflation

- Invariant subspace associated with small eigenvalues delays convergence
- Corresponds to smooth modes that change little for small localized changes in the problem

- Remove them to improve convergence!
  - Recycle space = approximate eigenspace

\[
\begin{align*}
\min_{z \in \mathcal{K}^m(A, r_0)} \| r_0 - Az \|_2 &= \min_{p_m(0)=1} \| p_m(A) r_0 \|_2 \\
&\leq \kappa(V) \| r_0 \|_2 \min_{p_m(0)=1} \max_{\lambda \in \Lambda(A)} | p_m(\lambda) |
\end{align*}
\]

- If \( \kappa(V) \) is not large (normality assumption) we can improve bound by removing select eigenvalues
Typical Convergence with Recycling

- IC(0) preconditioner
- GMRES – full recurrence
- All Others – Max subspace size 40
Example #1 Topology Optimization

- Optimize material distribution, $\rho$, in design domain
- Minimize compliance $u^T K(\rho) u$, where $K(\rho) u = f$

Example #1: Topology Optimization

<table>
<thead>
<tr>
<th>Size</th>
<th>Num. DOFs</th>
<th>Direct Solve Time</th>
<th>Recycling Solve Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>9,360</td>
<td>0.96</td>
<td>1.68</td>
</tr>
<tr>
<td>Medium</td>
<td>107,184</td>
<td>179.30</td>
<td>50.41</td>
</tr>
<tr>
<td>Large</td>
<td>1,010,160</td>
<td>26154.00</td>
<td>1196.30</td>
</tr>
</tbody>
</table>

Recycling Solve = RMINRES + IC(0) PC

Direct Solve = multifrontal, supernodal Cholesky factorization from TAUCS

Example #2 Stochastic PDEs*

- Stochastic elliptic equation

\[-\nabla \cdot (a(x, \omega)) \nabla u(x, \omega) = f(x) \quad x \in D, \omega \in \Omega\]
\[u(x, \omega) = 0 \quad x \in \partial D, \omega \in \Omega\]

- KL expansion + double orthogonal basis + discretization
  - Separate deterministic and stochastic components
  - Yield sequence of uncoupled equations
    \[A^{(i)}x^{(i)} = b^{(i)} \quad i=1,2,3,\ldots\]

- Preprocess for recycling Krylov solver
  - Use reordering scheme to minimize change in spectra of linear system

Example #2 Stochastic PDEs*

- Scheme #1: No Krylov recycling
- Scheme #4: Recycle Krylov spaces using reordering
- Many systems require zero iterations!

---

Table 4.1

Running time for different schemes and preconditioning (seconds).

<table>
<thead>
<tr>
<th>Preconditioner</th>
<th>Scheme 1</th>
<th>Scheme 2</th>
<th>Scheme 3</th>
<th>Scheme 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-level ASM</td>
<td>12030</td>
<td>7693</td>
<td>4205</td>
<td>3882</td>
</tr>
<tr>
<td>Two-level ASM</td>
<td>20980</td>
<td>14130</td>
<td>10740</td>
<td>8476</td>
</tr>
</tbody>
</table>

Structure of Recycling Solver

Solve System \(i-1\)

Recycle Space

Recycle Space

Converged?

\(AU = C\)

\((I-CC^T)AV = VH\)

Krylov Space

Create new recycle space

Recycle Space

Solve System \(i+1\)
Structure of Recycling Solver

- Solve System $i-1$
- Recycle Space
- Cycle
  - $AU=C$
  - $(I-CC^T)AV=VH$
- Recycle Space
- Krylov Space
- Converged?
  - Y
  - Recycle Space
  - Solve System $i+1$
  - “Dominant” subspace selection
- N
  - Create new recycle space
- Choice of Method: GMRES, MINRES, CG, etc.
Summary

- Belos is a **next-generation** linear solver library

- Belos lets you solve:
  - Single RHS: $Ax = b$
  - Multiple RHS (available simultaneously): $AX = B$
  - Multiple RHS (available sequentially): $Ax_i = b_i, \ i=1,\ldots,k$
  - Sequential Linear systems: $A_ix_i = b_i, \ i=1,\ldots,k$

- Belos contains these solvers:
  - Block CG, Pseudo-Block CG, RCG, PCPG, Block GMRES, Pseudo-Block GMRES, Block FGMRES, Hybrid GMRES, TFQMR, GCRODR

- Check out the Trilinos Tutorial:

- See Belos website for more: