SAND2011-8264P

A Tutorial on Anasazi and Belos

2011 Trilinos User Group Meeting
November 1st, 2011

Chris Baker
David Day
Mike Heroux
Mark Hoemmen
Rich Lehoucq
Mike Parks
Heidi Thornquist (Lead)

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Sandia National Laboratories

Outline

- Belos and Anasazi Framework
  - Background / Motivation
  - Framework overview
  - Available solver components
- Using Anasazi and Belos
  - Simple example
  - Through Stratimikos (Belos)
  - Through LOCA (Anasazi)
- Summary

Background / Motivation

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

AztecOO

- 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
- Interface requires Epetra objects
  - Double-precision arithmetic
- Interface to matrix-vector product is defined by the user through the EpetraOperator

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 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

Anasazi and Belos

- Next generation linear solver (Belos) and eigensolver (Anasazi) libraries, written in templated C++.
  - Iterative methods for solving sparse, matrix-free systems
- Provide a generic interface to a collection of algorithms for solving linear problems and eigenproblems.
- Algorithms developed with generic programming techniques.
  - Algorithmic components:
    - Ease the implementation of complex algorithms
  - Operator/MultiVector interface *(and Teuchos::ScalarTraits)*:
    - Allow the user to leverage their existing software investment
    - Multi-precision solver capability
  - Design offers: Interoperability, extensibility, and reusability
- Includes block linear solvers and eigensolvers.

Sandia National Laboratories

Why are Block Solvers Useful?

- In general, block solvers enable the use of faster computational kernels.
- Block Eigensolvers ($Op(A)X = \Lambda X$):
  - Reliably determine multiple and/or clustered eigenvalues.
  - Example applications:
    - Stability analysis / Modal 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

Belos Solver Categories

- 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 techiques

Belos Solvers

- Hermitian Systems ($A = A^H$)
  - Block CG
  - Pseudo-Block CG (Perform single-vector algorithm simultaneously)
  - RCG (Recycling Conjugate Gradients)
  - PCPG (Projected CG)
  - MINRES New!
- 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)
  - Block GCRODR (Block variant of GCRODR) New!

Sandia National Laboratories

Anasazi Solvers

- Hermitian Eigenproblems
  - Block Davidson
  - Locally-Optimal Block Preconditioned Conjugate Gradient (LOBPCG)
  - Implicit Riemannian Trust-Region (IRTR)
- Non-Hermitian Eigenproblems
  - Block Krylov-Schur (BKS)

Anasazi and Belos

(Algorithmic components)

$x^{(0)}$ 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 ith 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
  UPDATE$(\tilde{x}, m)$
end

GMRES Example

Sandia National Laboratories

Anasazi and Belos

(Algorithmic components)

SolverManager 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 $k = 1, ..., i$
      $h_{k,i} = (w, v^{(k)})$
      $w = w - h_{k,i}v^{(k)}$
    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 ith 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
  UPDATE$(\tilde{x}, m)$
end

GMRES Example

Sandia National Laboratories

Anasazi and Belos

(Algorithmic components)

SolverManager Class

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 $k = 1, \ldots, i$
      $h_{k,i} = (w, v^{(k)})$
      $w = w - h_{k,i} v^{(k)}$
    end
    $h_{i+1,i} = \|w\|_2$
    $v^{(i+1)} = w/h_{i+1,i}$
    apply $J_1, \ldots, J_{i-1}$ on $(h_{1,i}, \ldots, h_{i+1,i})$
    construct J_i, acting on ith and $(i+1)$st component
    of $h_{\cdot,i}$, such that $(i+1)$st component of $J_i h_{\cdot,i}$ 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

GMRES Example

Sandia National Laboratories

Anasazi and Belos

(Algorithmic components)

SolverManager Class

Problem Classes / 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 $k = 1, \ldots, i$
      $h_{k,i} = (w, v^{(k)})$
      $w = w - h_{k,i} v^{(k)}$
    end
    $h_{i+1,i} = \|w\|_2$
    $v^{(i+1)} = w/h_{i+1,i}$
    apply $J_1, \ldots, J_{i-1}$ on $(h_{1,i}, \ldots, h_{i+1,i})$
    construct J_i, acting on ith and $(i+1)$st component
    of $h_{\cdot,i}$, such that $(i+1)$st component of $J_i h_{\cdot,i}$ 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

GMRES Example

Sandia National Laboratories

Anasazi and Belos

(Algorithmic components)

SolverManager Class

Problem Classes / Operator Classes

Iteration Class

MultiVector Classes

$x^{(0)}$ 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 $k = 1, \ldots, i$
      $h_{k,i} = (w, v^{(k)})$
      $w = w - h_{k,i} v^{(k)}$
    end
    $h_{i+1,i} = \|w\|_2$
    $v^{(i+1)} = w/h_{i+1,i}$
    apply $J_1, \ldots, J_{i-1}$ on $(h_{1,i}, \ldots, h_{i+1,i})$
    construct J_i, acting on ith and $(i+1)$st component
    of $h_{\cdot,i}$, such that $(i+1)$st component of $J_i h_{\cdot,i}$ 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

GMRES Example

Sandia National Laboratories

Anasazi and Belos

(Algorithmic components)

SolverManager Class

Problem Classes / Operator Classes

Iteration Class

MultiVector Classes

OrthoManager Class

(ICGS, IMGS, DGKS, TSQR)

$x^{(0)}$ 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 $k = 1, \ldots, i$
      $h_{k,i} = (w, v^{(k)})$
      $w = w - h_{k,i} v^{(k)}$
    end
    $h_{i+1,i} = \|w\|_2$
    $v^{(i+1)} = w/h_{i+1,i}$
    apply $J_1, \ldots, J_{i-1}$ on $(h_{1,i}, \ldots, h_{i+1,i})$
    construct J_i, acting on ith and $(i+1)$st component
    of $h_{\cdot,i}$, such that $(i+1)$st component of $J_i h_{\cdot,i}$ 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

GMRES Example

Sandia National Laboratories

Anasazi and Belos

(Algorithmic components)

SolverManager Class

Problem Classes / Operator Classes

Iteration Class

MultiVector Classes

OrthoManager Class

(ICGS, IMGS, DGKS, TSQR)

StatusTest 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 $k = 1, \ldots, i$
      $h_{k,i} = (w, v^{(k)})$
      $w = w - h_{k,i} v^{(k)}$
    end
    $h_{i+1,i} = \|w\|_2$
    $v^{(i+1)} = w/h_{i+1,i}$
    apply $J_1, \ldots, J_{i-1}$ on $(h_{1,i}, \ldots, h_{i+1,i})$
    construct J_i, acting on ith and $(i+1)$st component
    of $h_{\cdot,i}$, such that $(i+1)$st component of $J_i h_{\cdot,i}$ 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

GMRES Example

Sandia National Laboratories

Anasazi and Belos

(Algorithmic components)

SolverManager Class

Problem Classes / Operator Classes

Iteration Class

MultiVector Classes

OrthoManager Class

(ICGS, IMGS, DGKS, TSQR)

StatusTest 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 $k = 1, \ldots, i$
      $h_{k,i} = (w, v^{(k)})$
      $w = w - h_{k,i} v^{(k)}$
    end
    $h_{i+1,i} = \|w\|_2$
    $v^{(i+1)} = w/h_{i+1,i}$
    apply $J_1, \ldots, J_{i-1}$ on $(h_{1,i}, \ldots, h_{i+1,i})$
    construct J_i, acting on ith and $(i+1)$st component
    of $h_{\cdot,i}$, such that $(i+1)$st component of $J_i h_{\cdot,i}$ 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

OutputManager Class

SortManager Class

Sandia National Laboratories

Example (Step #1 – Initialize System)

```
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

Sandia National Laboratories

Example (Step #2 – Solver Params)

```
bool verbose = false, debug = false, proc_verbose = false;
int frequency = -1;          // frequency of status test output.
int blocksize = 1;           // blocksize
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;
belosList.set( "Num Blocks", maxsubspace);            // Maximum number of blocks in Krylov
  factorization
belosList.set( "Block Size", blocksize );             // Blocksize 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;
}
belosList.set( "Verbosity", verbosity );
```

Solver Parameters

ParameterList for SolverManager

Trilinos/packages/belos/epetra/example/BlockGmres/BlockGmresEpetraExFile.cpp

Sandia National Laboratories

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;
}
```

LinearProblem Object

Template Parameters

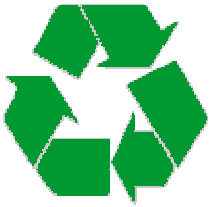
SolverManager Object

```
// 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;
```

Trilinos/packages/belos/epetra/example/BlockGmres/BlockGmresEpetraExFile.cpp

Sandia National Laboratories

Spotlight on Recycling



Sandia National Laboratories

Sequences of Linear Systems

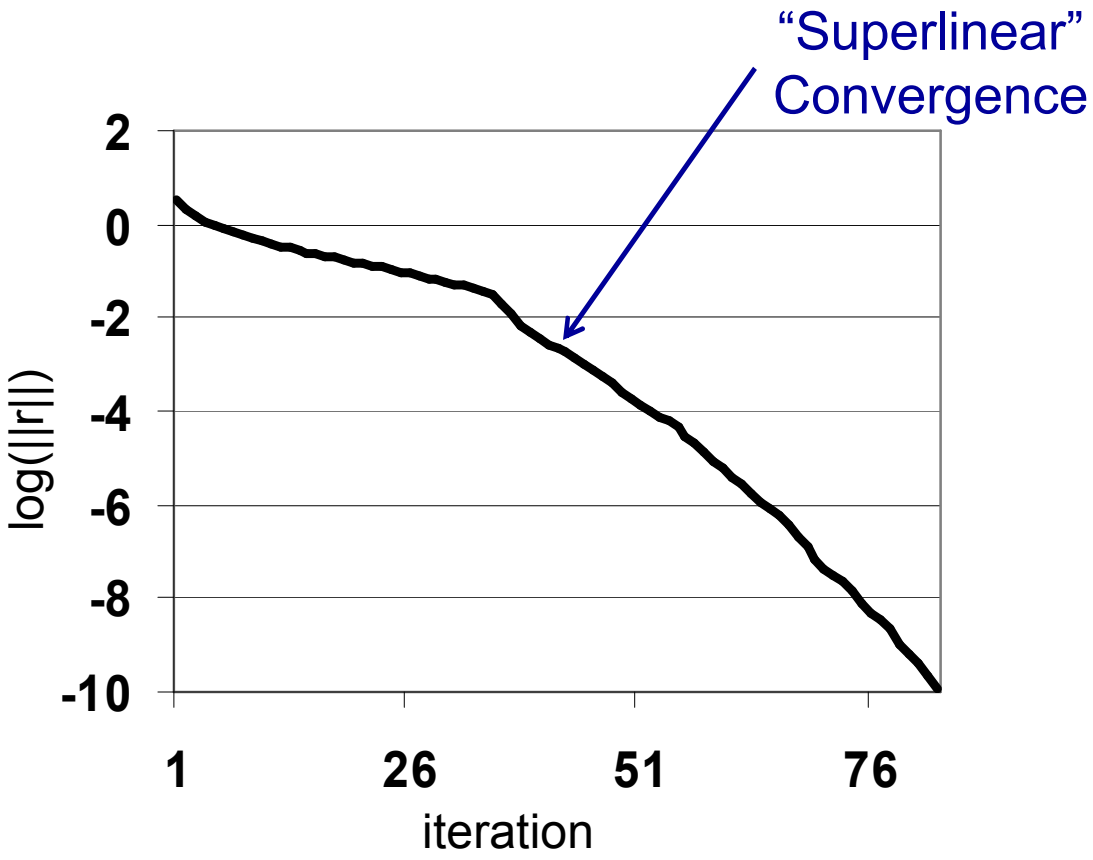
- Consider sequence of linear systems

$$\mathbf{A}^{(i)}\mathbf{x}^{(i)} = \mathbf{b}^{(i)} \qquad i=1,2,3,\ldots$$

- 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)



Sandia National Laboratories

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

Matrix	Off-the-shelf solver	Recycling Solver	Release
General	GMRES	GCRODR	Trilinos 8
SPD	CG	Recycling CG (RCG)	Trilinos 10
Symmetric Indefinite	MINRES	Recycling MINRES (RMINRES)	N/A

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

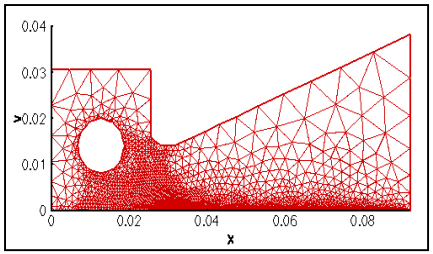
$$\min_{z \in K^m(A, r_0)} \|r_0 - Az\|_2 = \min_{P_m(0)=1} \|p_m(A) r_0\|_2$$
$$\leq \kappa(V) \|r_0\|_2 \min_{P_m(0)=1} \max_{\lambda \in \Lambda(A)} |p_m(\lambda)|$$

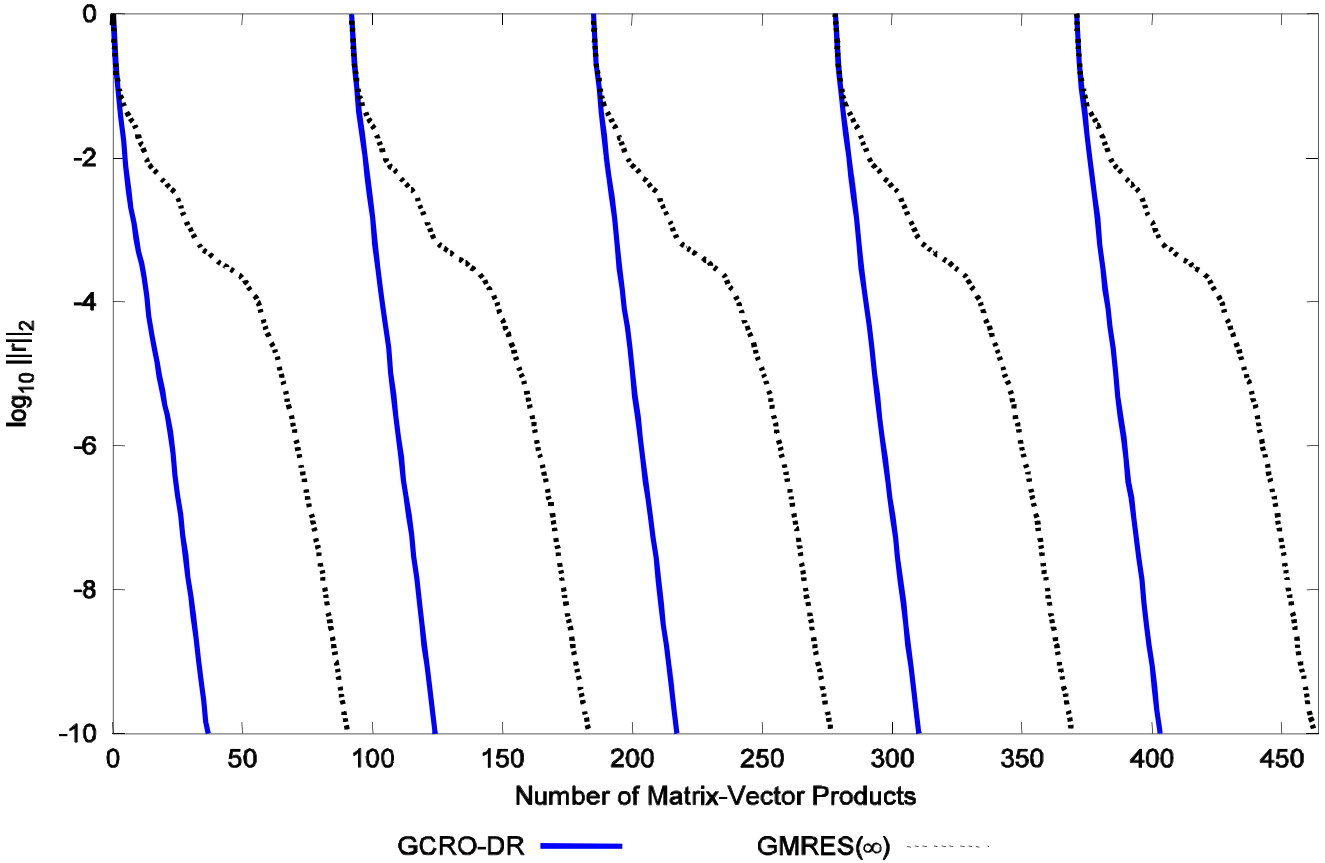
- If $\kappa(V)$ is not large (normality assumption) we can improve bound by removing select eigenvalues

Sandia National Laboratories

Typical Convergence with Recycling

- IC(0) preconditioner
- GMRES – full recurrence
- All Others – Max subspace size 40

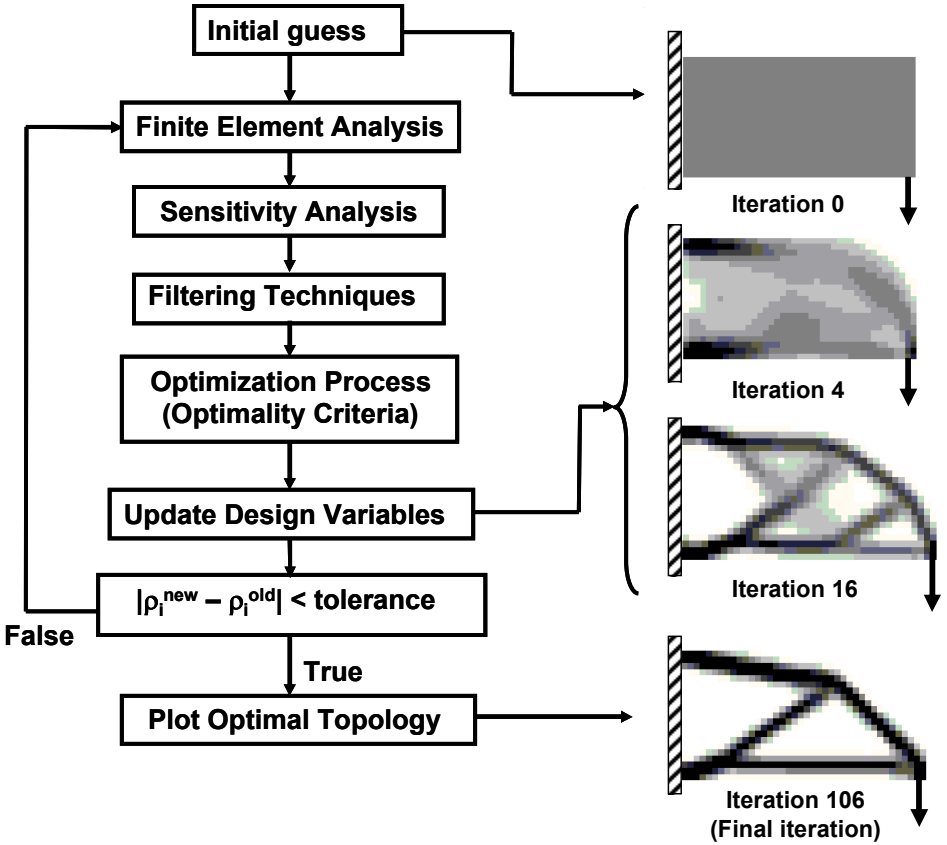




Sandia National Laboratories

Example #1 Topology Optimization*

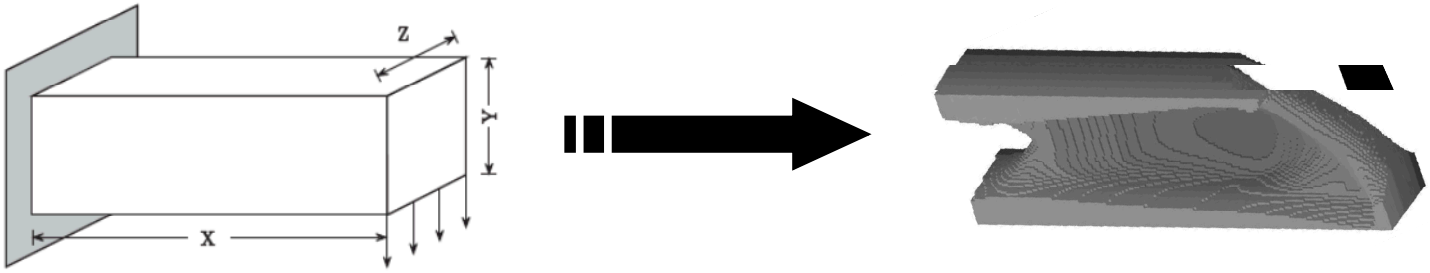
- Optimize material distribution, ρ, in design domain
- Minimize compliance $u^T K(\rho)u$, where $K(\rho)u=f$



*S. Wang, E. de Sturler, and G. H. Paulino, *Large-scale topology optimization using preconditioned Krylov subspace methods with recycling*, International Journal for Numerical Methods in Engineering, Vol. 69, Issue 12, pp. 2441—2468, 2007.

Sandia National Laboratories

Example #1: Topology Optimization



Size	Num. DOFs	Direct Solve Time	Recycling Solve Time
Small	**9,360**	**0.96**	**1.68**
Medium	**107,184**	**179.30**	**50.41**
Large	**1,010,160**	**26154.00**	**1196.30**

Recycling Solve = RMINRES + IC(0) PC

Direct Solve = multifrontal, supernodal Cholesky factorization from TAUCS

[*]S. Wang, E. de Sturler, and G. H. Paulino, *Large-scale topology optimization using preconditioned Krylov subspace methods with recycling*, International Journal for Numerical Methods in Engineering, Vol. 69, Issue 12, pp. 2441—2468, 2007.

Sandia National Laboratories

Example #2 Stochastic PDEs*

- Stochastic elliptic equation

$$-\nabla \cdot \big(\mathbf{a}(\mathbf{x},\omega)\big)\nabla \mathbf{u}(\mathbf{x},\omega) = \mathbf{f}(\mathbf{x}) \qquad \mathbf{x} \in \mathbf{D}, \omega \in \Omega$$
$$\mathbf{u}(\mathbf{x},\omega) = \mathbf{0} \qquad \mathbf{x} \in \partial\mathbf{D}, \omega \in \Omega$$

- KL expansion + double orthogonal basis + discretization
  - Separate deterministic and stochastic components
  - Yield sequence of uncoupled equations

$$\mathbf{A}^{(i)}\mathbf{x}^{(i)} = \mathbf{b}^{(i)} \qquad \mathbf{i=1,2,3,\ldots}$$

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

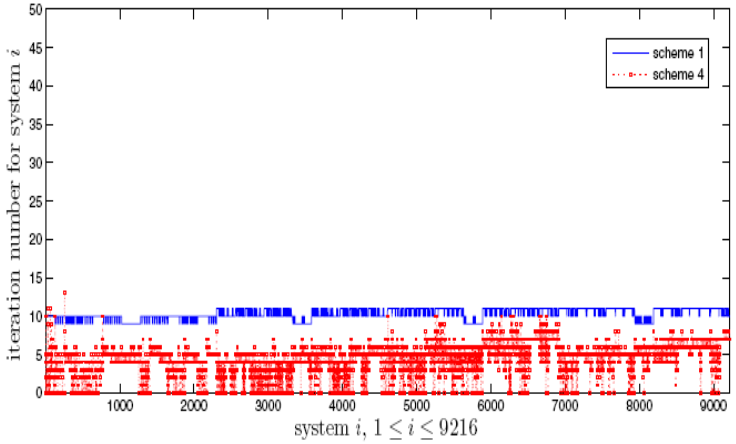
*C. Jin, X-C. Cai, and C. Li, *Parallel Domain Decomposition Methods for Stochastic Elliptic Equations,* SIAM Journal on Scientific Computing, Vol. 29, Issue 5, pp. 2069—2114, 2007.

Sandia National Laboratories

Example #2 Stochastic PDEs*

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

One-Level ASM



Two-Level ASM

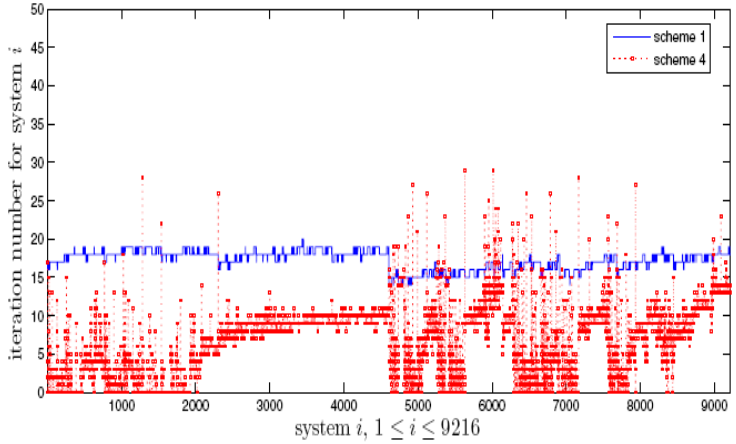


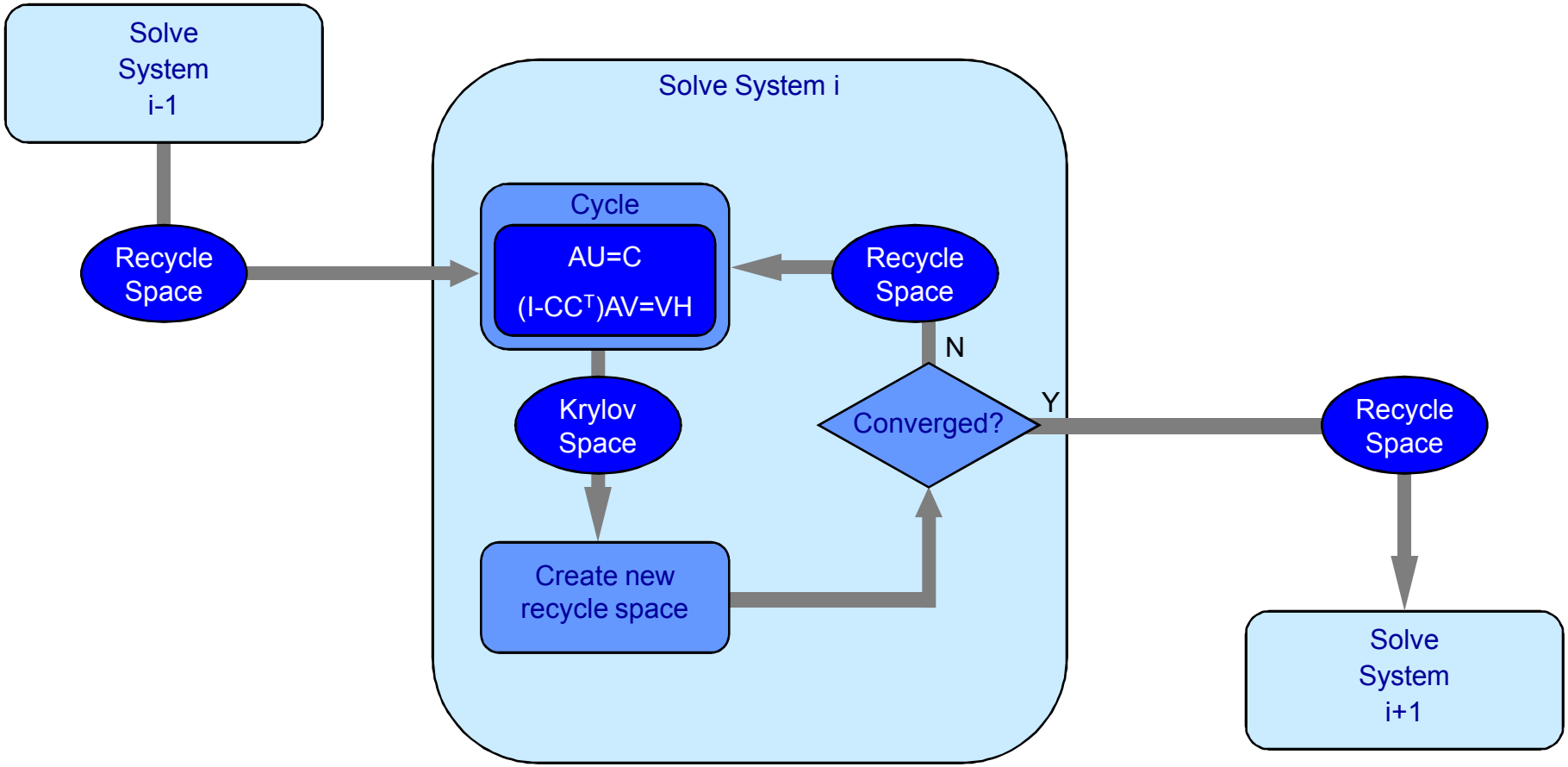
TABLE 4.1
Running time for different schemes and preconditioning (seconds).

Preconditioner	Scheme			
	1	2	3	4
One-level ASM	12030	7693	4205	3882
Two-level ASM	20980	14130	10740	8476

*C. Jin, X-C. Cai, and C. Li, *Parallel Domain Decomposition Methods for Stochastic Elliptic Equations,* SIAM Journal on Scientific Computing, Vol. 29, Issue 5, pp. 2069—2114, 2007.

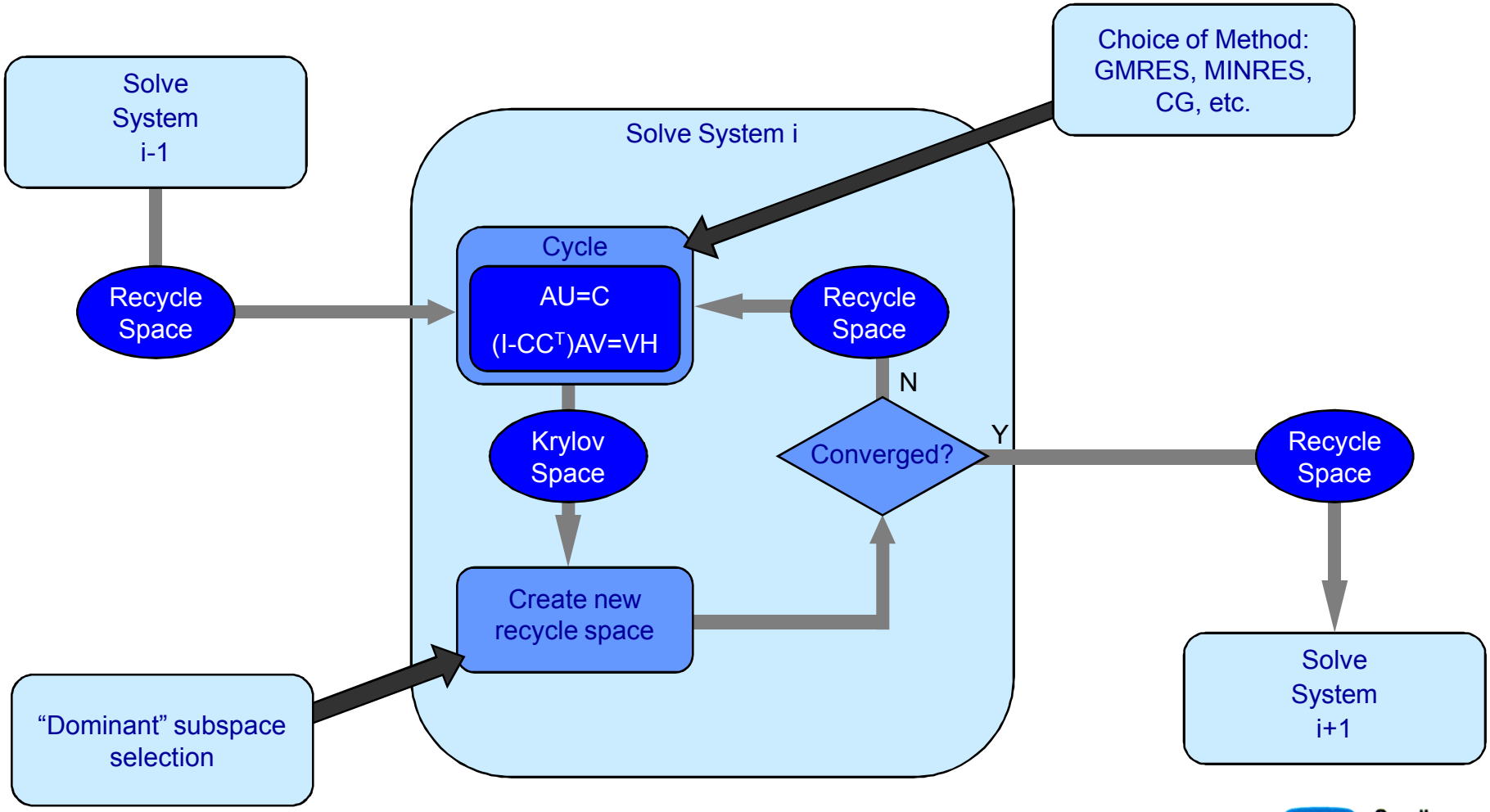
Sandia National Laboratories

Structure of Recycling Solver



Sandia National Laboratories

Structure of Recycling Solver



Sandia National Laboratories

Summary

- Belos and Anasazi are **next-generation** linear and eigensolver libraries
  - Designed for interoperability, extensibility, and reusability
- Belos and Anasazi are readily available:
  - Can be used as standalone linear and eigensolvers
  - Belos available through Stratimikos
  - Anasazi available through LOCA
- Check out the Trilinos Tutorial
  `http://trilinos.sandia.gov/Trilinos10.8Tutorial.pdf`
- See website for more:
  `http://trilinos.sandia.gov/packages/belos`
  `http://trilinos.sandia.gov/packages/anasazi`

Sandia National Laboratories