SANDIA REPORT

SAND2016-3396 O
Unlimited Release
Printed April 21, 2016

xSDKTrilinos User Manual

Alicia Marie Klinvex

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

SAND2016-3396 O
Unlimited Release
Printed April 21, 2016

xSDKTrilinos User Manual

Alicia Marie Klinvex

Abstract

Some application developers need to be able to use Trilinos together with other libraries, such as PETSc. This is nontrivial because these libraries all expect the data to be stored in different ways, and the way that you call a PETSc KSP linear solver, for instance, looks fundamentally different from the way you would call a Belos linear solver. The IDEAS software productivity project plans to address this problem with the Extreme-scale Scientific Software Development Kit (xSDK). The xSDK will provide an interoperability layer that enables easy installation and combined usage of the IDEAS libraries, including PETSc, Hypre, and SuperLU. This document describes the various interoperability layers and how to install and use xSDKTrilinos.

Chapter 1

xSDK Installation

The recommended way to install the various xSDK libraries is to use the configuration script provided at `http://xsdk-getting-started-guide.readthedocs.org/`. If you would prefer to manually install Trilinos and enable the PETSc, hypre, and SuperLU interfaces, you may use the following procedure.

1. Download and install the external packages (such as PETSc, hypre, and SuperLU)
2. Go to the packages subdirectory of Trilinos and clone the xSDKTrilinos repository
   ```
   $ cd /path/to/Trilinos/packages
   ```
3. Clone the xSDKTrilinos repository
   ```
   $ git clone https://github.com/trilinos/xSDKTrilinos.git
   ```
4. Go to the directory where you wish to build Trilinos and invoke your configuration script.
   ```
   $ cd  /trilinos-build $ ./do-configure
   $ make -j8
   $ make install
   ```
   An example script is presented as Program 1.1.

   The first thing we do in the example script is define where the Trilinos source code is located. Then, we set the paths to the include directory and library directory for each third party library (TPL) we wish to enable. Here, we have decided to enable PETSc, hypre, and SuperLU_Dist. Since SuperLU_Dist requires METIS and ParMETIS, we must enable them as well. The first couple of arguments to CMake are standard Trilinos arguments documented in the Trilinos Configure, Build, Test, and Install Reference Guide (found at `https://trilinos.org/docs/files/TrilinosBuildReference.html`). TPLs are addressed in in section 5.13 of that manual. For each TPL we wish to enable, we add `-D TPL_ENABLE_[TPL_NAME]`, `-D [TPL_NAME]_LIBRARY_DIRS`, and `-D [TPL_NAME]_INCLUDE_DIRS`. We explicitly enable the packages Amesos2 and xSDKTrilinos because they contain the interfaces to SuperLU_Dist, hypre, and PETSc.
5. You may test your install by invoking ctest in your build directory
   ```
   $ ctest
   ```

Program 1.1. Sample configuration script

```
#!/bin/bash
#

TRILINOS_HOME=/path/to/Trilinos

PETSC_INCLUDE_DIR=/path/to/petsc/include
PETSC_LIB_DIR=/path/to/petsc/lib

HYPRE_INCLUDE_DIR=/path/to/hypre/include
HYPRE_LIB_DIR=/path/to/hypre/lib

METIS_INCLUDE_DIR=/path/to/metis/include
METIS_LIB_DIR=/path/to/metis/lib

PARMETIS_INCLUDE_DIR=/path/to/parmetis/include
PARMETIS_LIB_DIR=/path/to/parmetis/lib

SUPERLU_INCLUDE_DIR=/path/to/superlu/include
SUPERLU_LIB_DIR=/path/to/superlu/lib

rm -rf CMakeFiles CMakeCache.txt

cmake \
  -D CMAKE_BUILD_TYPE=DEBUG \
  -D CMAKE_INSTALL_PREFIX:PATH="~/trilinos-install" \
  -D TPL_ENABLE_MPI:BOOL=ON \
  -D Trilinos_ENABLE_EXPLICIT_INSTANTIATION:BOOL=ON \
  -D Teuchos_ENABLE_COMPLEX:BOOL=OFF \
  -D Trilinos_ENABLE_TESTS:BOOL=ON \
  -D Trilinos_ENABLE_EXAMPLES:BOOL=ON \
  -D Trilinos_ENABLE_ALL_OPTIONAL_PACKAGES=ON \
\
  -D BLAS_LIBRARY_NAMES:STRING="libf77blas.so.3" \
  -D BLAS_LIBRARY_DIRS:PATH="/usr/lib64/atlas" \
  -D LAPACK_LIBRARY_NAMES:STRING="liblapack.so.3" \
  -D LAPACK_LIBRARY_DIRS:PATH="/usr/lib64/atlas" \
\
  -D TPL_ENABLE_PETSC:BOOL=ON \
  -D PETSC_LIBRARY_DIRS:FILEPATH="${PETSC_LIB_DIR}" \
  -D PETSC_INCLUDE_DIRS:FILEPATH="${PETSC_INCLUDE_DIR}" \
\
  -D TPL_ENABLE_HYPRE:BOOL=ON \
  -D HYPRE_LIBRARY_DIRS:FILEPATH="${HYPRE_LIB_DIR}" \
  -D HYPRE_INCLUDE_DIRS:FILEPATH="${HYPRE_INCLUDE_DIR}" \
\
  -D TPL_ENABLE_ParMETIS:BOOL=ON \
  -D ParMETIS_LIBRARY_DIRS:FILEPATH="${PARMETIS_LIB_DIR};${METIS_LIB_DIR}" \
  -D TPL_ParMETIS_INCLUDE_DIRS:FILEPATH="${PARMETIS_INCLUDE_DIR};${METIS_INCLUDE_DIR}" \
\
  -D TPL_ENABLE_SuperLUDist:BOOL=ON \
  -D SuperLUDist_LIBRARY_DIRS:FILEPATH="${SUPERLU_LIB_DIR}" \
  -D SuperLUDist_INCLUDE_DIRS:FILEPATH="${SUPERLU_INCLUDE_DIR}" \
\
  -D Trilinos_ENABLE_Amesos2:BOOL=ON \
  -D Trilinos_ENABLE_xSDKTrilinos:BOOL=ON \
\
  ${TRILINOS_HOME}
```

Chapter 2

xSDKTrilinos Interface Usage

This section describes the individual interfaces and their usage.

Trilinos-PETSc

There is a two-way interface between PETSc and Trilinos which allows users to use PETSc datatypes with Trilinos and vice-versa.

Using PETSc Mat and Vec with Trilinos solvers

Trilinos has two new interfaces to support using PETSc Mat anywhere a Tpetra::RowMatrix or Tpetra::CrsMatrix can be used. For packages requiring a Tpetra::RowMatrix or Tpetra::Operator, such as Anasazi and Belos, you may wrap a PETSc Mat in a Tpetra::PETScAIJMatrix; otherwise, you can copy it to a Tpetra::CrsMatrix. We will demonstrate each of those functions in the examples below.

Our first example (Program 2.1) shows how to compute the smallest eigenpairs of a PETSc Mat, specificially Poisson2D, using Trilinos' Anasazi package.

Program 2.1. PETSc_AnasaziEx.cpp

```
#include "petscksp.h"
#include "AnasaziBasicEigenproblem.hpp"
#include "AnasaziConfigDefs.hpp"
#include "AnasaziTpetraAdapter.hpp"
#include "AnasaziRTRSolMgr.hpp"
#include "Teuchos_ParameterList.hpp"
#include "Tpetra_PETScAIJMatrix.hpp"

int main(int argc,char **args)
{
  using Teuchos::RCP;
  using Teuchos::rcp;
  using std::cout;
  using std::endl;

  typedef Tpetra::PETScAIJMatrix<>          PETScAIJMatrix;
  typedef PETScAIJMatrix::scalar_type               Scalar;
  typedef PETScAIJMatrix::local_ordinal_type            LO;
  typedef PETScAIJMatrix::global_ordinal_type           GO;
  typedef PETScAIJMatrix::node_type                   Node;
  typedef Tpetra::Vector<Scalar,LO,GO,Node>         Vector;
  typedef Tpetra::Map<LO,GO,Node>                      Map;
23    typedef Tpetra::Operator<Scalar,LO,GO,Node>            OP;
24    typedef Tpetra::MultiVector<Scalar,LO,GO,Node>         MV;
25    typedef Anasazi::RTRSolMgr<Scalar,MV,OP>           SolMgr;
26    typedef Anasazi::BasicEigenproblem<Scalar,MV,OP>  Problem;
27    typedef Anasazi::OperatorTraits<Scalar,MV,OP>         OPT;
28    typedef Anasazi::MultiVecTraits<Scalar,MV>            MVT;
29
30    Mat            A;
31    PetscInt       m = 50,n = 50;
32    PetscInt       nev = 4;
33    PetscErrorCode ierr;
34    MPI_Comm       comm;
35    PetscInt       Istart, Iend, Ii, i, j, J, rank;
36    PetscScalar    v, tol=1e-6;
37
38    // Initialize PETSc
39    PetscInitialize(&argc,&args,NULL,NULL);
40
41    // Create the matrix
42    ierr = MatCreate(PETSC_COMM_WORLD,&A);CHKERRQ(ierr);
43    ierr = MatSetSizes(A,PETSC_DECIDE,PETSC_DECIDE,m*n,m*n);CHKERRQ(ierr);
44    ierr = MatSetType(A, MATAIJ);CHKERRQ(ierr);
45    ierr = MatMPIAIJSetPreallocation(A,5,PETSC_NULL,5,PETSC_NULL);CHKERRQ(ierr);
46    ierr = MatSetUp(A);CHKERRQ(ierr);
47    ierr = PetscObjectGetComm( (PetscObject)A, &comm);CHKERRQ(ierr);
48    ierr = MPI_Comm_rank(comm,&rank);CHKERRQ(ierr);
49
50    ierr = MatGetOwnershipRange(A,&Istart,&Iend);CHKERRQ(ierr);
51
52    for (Ii=Istart; Ii<Iend; Ii++) {
53      v = -1.0; i = Ii/n; j = Ii - i*n;
54      if (i>0)   {J = Ii - n; ierr = MatSetValues(A,1,&Ii,1,&J,&v,INSERT_VALUES);CHKERRQ(ierr);}
55      if (i<m-1) {J = Ii + n; ierr = MatSetValues(A,1,&Ii,1,&J,&v,INSERT_VALUES);CHKERRQ(ierr);}
56      if (j>0)   {J = Ii - 1; ierr = MatSetValues(A,1,&Ii,1,&J,&v,INSERT_VALUES);CHKERRQ(ierr);}
57      if (j<n-1) {J = Ii + 1; ierr = MatSetValues(A,1,&Ii,1,&J,&v,INSERT_VALUES);CHKERRQ(ierr);}
58      v = 4.0; ierr = MatSetValues(A,1,&Ii,1,&Ii,&v,INSERT_VALUES);CHKERRQ(ierr);
59    }
60
61    ierr = MatAssemblyBegin(A,MAT_FINAL_ASSEMBLY);CHKERRQ(ierr);
62    ierr = MatAssemblyEnd(A,MAT_FINAL_ASSEMBLY);CHKERRQ(ierr);
63
64    // Wrap the PETSc matrix as a PETScAIJMatrix.
65    RCP<PETScAIJMatrix> tpetraA = rcp(new PETScAIJMatrix(A));
66
67    // Create an initial guess
68    RCP<MV> initGuess = rcp(new MV(tpetraA->getDomainMap(),4,false));
69    initGuess->randomize();
70
71    // Create an eigenproblem
72    RCP<Problem> problem = rcp(new Problem(tpetraA,initGuess));
73    problem->setNEV(nev);
74    problem->setHermitian(true);
75    problem->setProblem();
76
77    // Create the parameter list
78    Teuchos::ParameterList pl;
79    pl.set("Verbosity", Anasazi::IterationDetails + Anasazi::FinalSummary);
80    pl.set("Convergence Tolerance", tol);
81
82    // Create an Anasazi eigensolver
83    RCP<SolMgr> solver = rcp(new SolMgr(problem, pl));
84
85    // Solve the problem to the specified tolerances
86    Anasazi::ReturnType returnCode = solver->solve();
87    if (returnCode != Anasazi::Converged && rank == 0) {
88      cout << "Anasazi::EigensolverMgr::solve() returned unconverged." << endl;
89      return EXIT_FAILURE;
90    }
91    else if (rank == 0)
92      cout << "Anasazi::EigensolverMgr::solve() returned converged." << endl;
93
94    // Get the eigenvalues and eigenvectors from the eigenproblem
95    Anasazi::Eigensolution<Scalar,MV> sol = problem->getSolution();
96    std::vector<Anasazi::Value<Scalar> > evals = sol.Evals;
97    RCP<MV> evecs = sol.Evecs;
98    int numev = sol.numVecs;
99
100   // Terminate PETSc
101   ierr = PetscFinalize();CHKERRQ(ierr);
102   return EXIT_SUCCESS;
103 }
```

Lines 1–7 Include statements

Lines 11–28 Typedefs and using statements to make the code more readable

Lines 30–36 PETSc variables

Line 39 Initialize PETSc.

Lines 42–62 Create the PETSc Mat and set its values.

Line 65 Wrap the PETSc Mat in a Tpetra::PETScAIJMatrix. Since Anasazi only requires a Tpetra::Operator[1], we do not have to deep copy the data to a Tpetra::CrsMatrix.

Lines 68–69 Create a random initial guess for the eigensolver. Note that we can treat tpetraA just like any other Tpetra::RowMatrix and obtain its domain map via getDomainMap().

Lines 72–75 Create the eigenproblem for Anasazi. We provide the operator A as well as our initial guess for the set of desired eigenvectors to the constructor. We then request a certain number of eigenvectors and inform the eigensolver that our problem is Hermitian[2].

Lines 78–80 Create the parameter list for the Riemannian Trust Region eigensolver. We have elected to have the solver print out the list of approximate eigenvalues at each iteration, along with their associated residuals. We also set our convergence tolerance here.

Lines 83–98 Solve the eigenvalue problem. After it is solved, we may grab the eigenvalues and eigenvectors via getSolution().

Line 101 Terminate PETSc.

The second example (Program 2.2) demonstrates how to use PETSc datatypes with Trilinos packages that require a Tpetra::CrsMatrix. One such package is Amesos2, which contains a variety of direct solvers (and interfaces to external direct solvers). In this example, we will solve a linear system $Ax = b$ where A is the 2D discretization of the Poisson operator on a unit square, and b is a random vector.

[1]RowMatrix is a specific type of Operator, and CrsMatrix is a specific type of RowMatrix. Therefore, you can use a RowMatrix anywhere an Operator is accepted, but you can't necessarily use a RowMatrix anywhere a CrsMatrix is expected.

[2]Some eigensolvers are optimized for use on Hermitian eigenproblems. Others do not work on non-Hermitian problems at all, so it is important to specify this.

Program 2.2. PETSc_Amesos2Ex.cpp

```
#include <Teuchos_ScalarTraits.hpp>
#include <Teuchos_RCP.hpp>
#include <Teuchos_GlobalMPISession.hpp>
#include <Teuchos_oblackholestream.hpp>
#include <Teuchos_Tuple.hpp>
#include <Teuchos_VerboseObject.hpp>
#include <Tpetra_DefaultPlatform.hpp>
#include <Tpetra_Map.hpp>
#include <Tpetra_MultiVector.hpp>
#include <Tpetra_CrsMatrix.hpp>
#include "Amesos2.hpp"
#include "Amesos2_Version.hpp"
#include "Amesos2_KLU2.hpp"
#include "petscksp.h"
#include "Tpetra_PETScAIJMatrix.hpp"
#include "Tpetra_Vector.hpp"

int main(int argc,char **args)
{
  using Teuchos::RCP;
  using Teuchos::rcp;
  using Teuchos::ArrayView;

  typedef Tpetra::PETScAIJMatrix<>    PETScAIJMatrix;
  typedef PETScAIJMatrix::scalar_type         Scalar;
  typedef PETScAIJMatrix::local_ordinal_type      LO;
  typedef PETScAIJMatrix::global_ordinal_type     GO;
  typedef PETScAIJMatrix::node_type             Node;

  typedef Tpetra::CrsMatrix<Scalar,LO,GO>  CrsMatrix;
  typedef Tpetra::Vector<Scalar,LO,GO>        Vector;
  typedef Tpetra::Map<LO,GO>                     Map;
  typedef Tpetra::Operator<Scalar,LO,GO>          OP;
  typedef Tpetra::MultiVector<Scalar,LO,GO>       MV;
  typedef Tpetra::Vector<Scalar,LO,GO>        Vector;
  typedef Amesos2::Solver<CrsMatrix,MV>       Solver;

  Vec            x,b;
  Mat            A;
  PetscRandom    rctx;
  PetscInt       i,j,Ii,J,Istart,Iend;
  PetscInt       m = 4,n = 4;
  PetscErrorCode ierr;
  PetscScalar    v;
  PetscInt rank=0;
  MPI_Comm comm;

  // Start PETSc
  PetscInitialize(&argc,&args,NULL,NULL);

  // Create the matrix
  ierr = MatCreate(PETSC_COMM_WORLD,&A);CHKERRQ(ierr);
  ierr = MatSetSizes(A,PETSC_DECIDE,PETSC_DECIDE,m*n,m*n);CHKERRQ(ierr);
  ierr = MatSetType(A, MATAIJ);CHKERRQ(ierr);
  ierr = MatMPIAIJSetPreallocation(A,5,PETSC_NULL,5,PETSC_NULL);CHKERRQ(ierr);
  ierr = MatSetUp(A);CHKERRQ(ierr);
  PetscObjectGetComm( (PetscObject)A, &comm);
  ierr = MPI_Comm_rank(comm,&rank);CHKERRQ(ierr);

  ierr = MatGetOwnershipRange(A,&Istart,&Iend);CHKERRQ(ierr);
  for (Ii=Istart; Ii<Iend; Ii++) {
    v = -1.0; i = Ii/n; j = Ii - i*n;
    if (i>0)   {J = Ii - n; ierr = MatSetValues(A,1,&Ii,1,&J,&v,INSERT_VALUES);CHKERRQ(ierr);}
    if (i<m-1) {J = Ii + n; ierr = MatSetValues(A,1,&Ii,1,&J,&v,INSERT_VALUES);CHKERRQ(ierr);}
    if (j>0)   {J = Ii - 1; ierr = MatSetValues(A,1,&Ii,1,&J,&v,INSERT_VALUES);CHKERRQ(ierr);}
    if (j<n-1) {J = Ii + 1; ierr = MatSetValues(A,1,&Ii,1,&J,&v,INSERT_VALUES);CHKERRQ(ierr);}
    v = 4.0; ierr = MatSetValues(A,1,&Ii,1,&Ii,&v,INSERT_VALUES);CHKERRQ(ierr);
  }

  ierr = MatAssemblyBegin(A,MAT_FINAL_ASSEMBLY);CHKERRQ(ierr);
  ierr = MatAssemblyEnd(A,MAT_FINAL_ASSEMBLY);CHKERRQ(ierr);

  // Create a random solution vector and corresponding right-hand-side
  ierr = VecCreate(PETSC_COMM_WORLD,&x);CHKERRQ(ierr);
  ierr = VecSetSizes(x,PETSC_DECIDE,m*n);CHKERRQ(ierr);
  ierr = VecSetFromOptions(x);CHKERRQ(ierr);
  ierr = PetscRandomCreate(PETSC_COMM_WORLD,&rctx);CHKERRQ(ierr);
  ierr = PetscRandomSetFromOptions(rctx);CHKERRQ(ierr);
  ierr = VecSetRandom(x,rctx);CHKERRQ(ierr);
  ierr = PetscRandomDestroy(&rctx);CHKERRQ(ierr);
  ierr = VecDuplicate(x,&b);CHKERRQ(ierr);
  ierr = MatMult(A,x,b);CHKERRQ(ierr);

  // Copy the matrix from a PETSc data structure to a Tpetra CrsMatrix
  RCP<CrsMatrix> tpetraA = xSDKTrilinos::deepCopyPETScAIJMatrixToTpetraCrsMatrix<Scalar,LO,GO,Node>(A);

  // Copy the PETSc vectors to Tpetra vectors
  RCP<Vector> tpetraX = xSDKTrilinos::deepCopyPETScVecToTpetraVector<Scalar,LO,GO,Node>(x);
  RCP<Vector> tpetraB = xSDKTrilinos::deepCopyPETScVecToTpetraVector<Scalar,LO,GO,Node>(b);
```

```
 90
 91    // Initialize the solution to 0
 92    tpetraX->putScalar(0);
 93
 94    // Create an Amesos2 linear solver
 95    RCP<Solver> solver = Amesos2::create<CrsMatrix,MV>("KLU2", tpetraA, tpetraX, tpetraB);
 96    solver->symbolicFactorization();
 97    solver->numericFactorization();
 98
 99    // Perform a linear solve with Amesos2
100    solver->solve();
101
102    // Terminate PETSc
103    ierr = PetscFinalize(); CHKERRQ(ierr);
104    return EXIT_SUCCESS;
105  }
```

Lines 1–71 These lines are very similar to the previous example, where we set up convenient typedefs and create the PETSc Poisson2D matrix.

Lines 74–82 Create a random solution vector x and its corresponding RHS b.

Lines 85–89 Deep copy the PETSc Mat and Vecs to Tpetra::CrsMatrix and Tpetra::Vector, so that we can use them with the Amesos2 linear solvers, which require a Tpetra::CrsMatrix.

Lines 95–100 Create an Amesos2 linear solver, specifically the native solver KLU2. Perform the symbolic and numeric factorizations, then solve the linear system.

Is the data copied or wrapped?

If you are using a part of Trilinos that requires Operator or RowMatrix, the data is wrapped. If you need a CrsMatrix specifically, the data is deep-copied.

Using Trilinos datatypes with PETSc KSP solvers

If you would like to use Trilinos datatypes, such as Tpetra::Operator and Tpetra::MultiVector, with a PETSc KSP linear solver, you may use the new Belos[3] interface: PETScSolMgr. This interface is very similar to that of the other native Belos linear solvers, which makes solving linear systems such as $AX = B$ a simple process.

1. (Optional) Create a Tpetra::Operator for the preconditioner $M \approx A$. You may use the preexisting preconditioners of Ifpack2 and MueLu, or you may create your own custom preconditioner. Alternatively, you may choose not to use a preconditioner at all.

[3]Belos is the iterative solver package of Trilinos.

Parameter	Description	Default Value
Maximum Iterations	integer defining the maximum number of iterations to be performed.	1000
Solver	string defining the linear solver to be used. A list of all valid linear solver options can be found at `http://www.mcs.anl.gov/petsc/petsc-current/docs/manualpages/KSP/KSPType.html`	KSPGMRES
Verbosity	Belos::MsgType defining the amount of output the program should produce. Options include Belos::Errors, Belos::Warnings, Belos::IterationDetails, Belos::TimingDetails, and Belos::StatusTestDetails	Belos::Errors
Convergence Tolerance	double defining the tolerance of the linear solver	10^{-8}

Table 2.1. Belos::PETScSolMgr parameters

2. Create a Belos::LinearProblem containing the operator A, the initial guess X, the right-hand side B, and the preconditioner M (if you have one).

3. Create a Teuchos::ParameterList containing the parameters you wish to set. These parameters are summarized in Table 2.1.

4. Create a Belos::PETScSolMgr with the LinearProblem and ParameterList from the previous steps.

5. Call solve()

The following example (Program 2.3) illustrates this process in greater detail. Note that this example does not contain a single of PETSc code.

Program 2.3. Tpetra_KSPEx.cpp

```
#include "BelosConfigDefs.hpp"
#include "BelosLinearProblem.hpp"
#include "BelosTpetraAdapter.hpp"
#include "BelosPETScSolMgr.hpp"
#include "Ifpack2_Factory.hpp"
#include "Teuchos_CommandLineProcessor.hpp"
#include "Teuchos_ParameterList.hpp"
#include "Teuchos_StandardCatchMacros.hpp"
#include "Tpetra_CrsMatrix.hpp"
#include "Tpetra_DefaultPlatform.hpp"
#include "Tpetra_MultiVector.hpp"
#include "MatrixMarket_Tpetra.hpp"

int main(int argc, char *argv[]) {
  typedef Tpetra::MultiVector<>                    MV;
  typedef Tpetra::Operator<>                       OP;
  typedef Tpetra::CrsMatrix<>               CrsMatrix;
  typedef Ifpack2::Preconditioner<>              Prec;

20    using Teuchos::ParameterList;
21    using Teuchos::RCP;
22    using Teuchos::rcp;
23
24    // Initialize MPI
25    Teuchos::oblackholestream blackhole;
26    Teuchos::GlobalMPISession mpiSession (&argc, &argv, &blackhole);
27
28    // Get the default communicator
29    RCP<const Teuchos::Comm<int> > comm = Tpetra::DefaultPlatform::getDefaultPlatform().getComm();
30
31    // Read the command line arguments
32    int numrhs = 2;
33    int maxiters = 100;
34    std::string filename("cage4.mtx");
35    double tol = 1.0e-5;
36    Teuchos::CommandLineProcessor cmdp(false,false);
37    cmdp.setOption("filename",&filename,"Filename for test matrix.");
38    cmdp.setOption("tol",&tol,"Relative residual tolerance used by GMRES solver.");
39    cmdp.setOption("num-rhs",&numrhs,"Number of right-hand sides to be solved for.");
40    cmdp.setOption("max-iters",&maxiters,"Maximum number of iterations per linear system.");
41    if (cmdp.parse(argc,argv) != Teuchos::CommandLineProcessor::PARSE_SUCCESSFUL) {
42      return -1;
43    }
44
45    // Get the matrix from a file
46    RCP<CrsMatrix> A = Tpetra::MatrixMarket::Reader<CrsMatrix>::readSparseFile(filename,comm);
47
48    // Create a random RHS and set the initial guess to 0
49    RCP<MV> B = rcp(new MV(A->getRowMap(),numrhs,false));
50    RCP<MV> X = rcp(new MV(A->getRowMap(),numrhs,false));
51    RCP<MV> trueX = rcp(new MV(A->getRowMap(),numrhs,false));
52    trueX->randomize();
53    A->apply(*trueX,*B);
54    X->putScalar(0);
55
56    // Construct preconditioner
57    Ifpack2::Factory factory;
58    RCP<Prec> M = factory.create("RELAXATION", A.getConst());
59    ParameterList ifpackParams;
60    ifpackParams.set("relaxation: type","Jacobi");
61    M->setParameters(ifpackParams);
62    M->initialize();
63    M->compute();
64
65    // Create parameter list for the Belos solver manager
66    ParameterList belosList;
67    belosList.set( "Maximum Iterations", maxiters );       // Maximum number of iterations allowed
68    belosList.set( "Convergence Tolerance", tol );         // Relative convergence tolerance requested
69    belosList.set( "Verbosity", Belos::IterationDetails ); // Print convergence information
70    belosList.set( "Solver", "bcgs" );                     // Use BiCGStab as the linear solver
71
72    // Construct a preconditioned linear problem
73    RCP<Belos::LinearProblem<double,MV,OP> > problem
74      = rcp( new Belos::LinearProblem<double,MV,OP>( A, X, B ) );
75    problem->setLeftPrec( M );
76    problem->setProblem();
77
78    // Create an iterative solver manager
79    RCP< Belos::PETScSolMgr<double,MV,OP> > solver
80      = rcp( new Belos::PETScSolMgr<double,MV,OP>(problem, rcp(&belosList,false)) );
81
82    // Perform solve
83    solver->solve();
84  }
```

Lines 1–12 Include statements

Lines 15–22 Typedefs and using statements to make the code more readable

Lines 25–29 Set up MPI and get the default communicator.

Lines 32–43 Parse command line arguments. This program allows the user to specify the filename for the matrix, the tolerance for the linear solve, the maximum number of iterations, and the number of right hand sides for the linear system.

Lines 46–54 Set up the linear system by reading the matrix from a file, creating a random solution and setting the right hand side accordingly. The initial guess for the solution is set to $\vec{0}$.

Lines 57–63 Set up the Ifpack2 Jacobi preconditioner.

Lines 66–70 Set the Belos parameters via a Teuchos::ParameterList. We set the maximum number of iterations, convergence tolerance, and which PETSc KSP solver is being used. Here we chose BiCGStab, but a list of all valid linear solver options can be found at `http://www.mcs.anl.gov/petsc/petsc-current/docs/manualpages/KSP/KSPType.html`. We also set the verbosity to IterationDetails, meaning PETSc will print the residual norm at each iteration.

Lines 73–76 Set up the linear problem for the Belos solver.

Lines 79–80 Create the linear solver. Note that even though PETSc is being used to solve the linear system, you construct and use a Belos Solver Manager as you would for any of the native Krylov solvers.

Line 83 Solve the linear system. The solution will overwrite the initial guess vector X.

Can I use this to solve linear systems with multiple right-hand sides?

Yes. Unfortunately, PETSc has no support for multivectors at this time, so each of the right hand sides will be processed independently. If you want block or pseudo-block linear solvers, those are available within Trilinos.

Is the data copied or wrapped?

The raw Tpetra matrix (or operator) data is wrapped rather than deep copied. The same applies to the preconditioner, if you are using a preconditioner.

Trilinos-hypre

Trilinos also has a new Tpetra-based interface to hypre. This interface lives in the Ifpack2 library with the native Trilinos preconditioners. The following examples will demonstrate how to take advantage of this exciting new functionality.

In our first example (Program 2.4), we will examine how to use hypre solvers and preconditioners with Tpetra objects.

Program 2.4. Hypre_SolveEx.cpp

```
#include "Tpetra_Map.hpp"
#include "Tpetra_CrsMatrix.hpp"
#include "Tpetra_DefaultPlatform.hpp"
#include "Ifpack2_Preconditioner.hpp"
#include "Ifpack2_Hypre.hpp"
#include "Teuchos_CommandLineProcessor.hpp"
#include "Teuchos_ParameterList.hpp"
#include "Teuchos_StandardCatchMacros.hpp"

int main(int argc, char *argv[]) {
  using Teuchos::Array;
  using Teuchos::RCP;
  using Teuchos::rcp;
  using Teuchos::ParameterList;
  using Ifpack2::FunctionParameter;
  using Ifpack2::Hypre::Prec;
  using Ifpack2::Hypre::Solver;

  typedef Tpetra::CrsMatrix<>::scalar_type Scalar;
  typedef Tpetra::CrsMatrix<>::local_ordinal_type LO;
  typedef Tpetra::CrsMatrix<>::global_ordinal_type GO;
  typedef Tpetra::CrsMatrix<>::node_type Node;
  typedef Tpetra::DefaultPlatform::DefaultPlatformType Platform;
  typedef Tpetra::CrsMatrix<Scalar> CrsMatrix;
  typedef Tpetra::MultiVector<Scalar> MV;
  typedef Tpetra::Operator<Scalar> OP;
  typedef Ifpack2::Preconditioner<Scalar> Preconditioner;
  typedef Tpetra::Map<> Map;

  // Initialize the MPI session
  Teuchos::oblackholestream blackhole;
  Teuchos::GlobalMPISession mpiSession(&argc,&argv,&blackhole);

  // Get the default communicator
  Platform &platform = Tpetra::DefaultPlatform::getDefaultPlatform();
  RCP<const Teuchos::Comm<int> > comm = platform.getComm();

  // Get parameters from command-line processor
  int nx = 10;
  Scalar tol = 1e-6;
  bool verbose = false;
  Teuchos::CommandLineProcessor cmdp(false,true);
  cmdp.setOption("nx",&nx, "Number of mesh points in x direction.");
  cmdp.setOption("tolerance",&tol, "Relative residual used for solver.");
  cmdp.setOption("verbose","quiet",&verbose, "Whether to print a lot of info or a little bit.");
  if(cmdp.parse(argc,argv) != Teuchos::CommandLineProcessor::PARSE_SUCCESSFUL) {
    return -1;
  }

  // Create the row map
  int n = nx*nx;
  RCP<Map> map = rcp(new Map(n,0,comm));

  // Create the 2D Laplace operator
  RCP<CrsMatrix> A = rcp(new CrsMatrix(map,5));
  for(LO i = 0; i<nx; i++) {
    for(LO j = 0; j<nx; j++) {
      GO row = i*nx+j;
      if(!map->isNodeGlobalElement(row))
        continue;

      Array<LO> indices;
      Array<Scalar> values;

      if(i > 0) {
        indices.push_back(row - nx);
        values.push_back(-1.0);
      }
      if(i < nx-1) {
```

```
 70         indices.push_back(row + nx);
 71         values.push_back(-1.0);
 72       }
 73       indices.push_back(row);
 74       values.push_back(4.0);
 75       if(j > 0) {
 76         indices.push_back(row-1);
 77         values.push_back(-1.0);
 78       }
 79       if(j < nx-1) {
 80         indices.push_back(row+1);
 81         values.push_back(-1.0);
 82       }
 83       A->insertGlobalValues(row,indices,values);
 84     }
 85   }
 86   A->fillComplete();
 87 
 88   // Create the initial guess and right hand side
 89   RCP<MV> trueX = rcp(new MV(A->getRowMap(),1,false));
 90   RCP<MV> X = rcp(new MV(A->getRowMap(),1));
 91   RCP<MV> B = rcp(new MV(A->getRowMap(),1,false));
 92   trueX->randomize();
 93   A->apply(*trueX,*B);
 94 
 95   // Create the parameters for hypre
 96   RCP<FunctionParameter> functs[10];
 97   functs[0] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetPrintLevel, 1));  // print AMG solution info
 98   functs[1] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetCoarsenType, 6)); // Falgout coarsening
 99   functs[2] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetRelaxType, 6));   // Sym GS/Jacobi hybrid
100   functs[3] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetNumSweeps, 1));   // Sweeps on each level
101   functs[4] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetTol, 0.0));       // Conv tolerance zero
102   functs[5] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetMaxIter, 1));     // Do only one iteration!
103   functs[6] = rcp(new FunctionParameter(Solver, &HYPRE_PCGSetMaxIter, 1000));      // Maximum iterations
104   functs[7] = rcp(new FunctionParameter(Solver, &HYPRE_PCGSetTol, tol));           // Convergence tolerance
105   functs[8] = rcp(new FunctionParameter(Solver, &HYPRE_PCGSetTwoNorm, 1));         // Use the two-norm as the stopping
        criteria
106   functs[9] = rcp(new FunctionParameter(Solver, &HYPRE_PCGSetPrintLevel, 2));      // Print solve info
107 
108   // Create the hypre solver
109   RCP<Preconditioner> prec = rcp(new Ifpack2::Ifpack2_Hypre<Scalar,LO,GO,Node>(A));
110   ParameterList hypreList;
111   hypreList.set("SolveOrPrecondition", Solver);
112   hypreList.set("Solver", Ifpack2::Hypre::PCG);
113   hypreList.set("Preconditioner", Ifpack2::Hypre::BoomerAMG);
114   hypreList.set("SetPreconditioner", true);
115   hypreList.set("NumFunctions", 10);
116   hypreList.set<RCP<FunctionParameter>*>("Functions", functs);
117   prec->setParameters(hypreList);
118   prec->compute();
119 
120   // Perform solve
121   prec->apply(*B,*X);
122 }
```

Lines 1–8 Include statements

Lines 11–28 Typedefs and using statements to make the code more readable

Lines 31–36 Set up MPI

Lines 39–48 Parse command line arguments. This program allows the user to specify how large the problem should be, how accurately the linear system should be solved, and how much information should be printed.

Lines 51–86 Set up the 2D Laplace operator.

Lines 89–93 Create a random right-hand-side and initialize the solution vector to 0.

Lines 96–106 Set hypre options (documented in the hypre user and reference manuals found at `http://computation.llnl.gov/project/linear_solvers/software.php`). In this example, we have elected to use the conjugate gradient method with an algebraic multigrid preconditioner. We have specified a particular coarsening and relaxation type. The most important thing to note about BoomerAMG is that if you would like to use it as a preconditioner, you must set its maximum number of iterations to 1; otherwise, hypre will assume you meant to use it as a linear solver. We then set the tolerance and maximum number of iterations for hypre's conjugate gradient solver.

Lines 109–118 Create the hypre solver. Line 111 specifies that we will be using a hypre linear solver, and line 112 says it will be the conjugate gradient method. Line 113 says we would also like to use BoomerAMG. Remember that you must also set "SetPreconditioner" to true, or the preconditioner will not be used. Lines 115 and 116 specify the hypre parameters such as print level, tolerance, and maximum number of iterations.

Line 121 Solve the linear system. The function "apply" actually calls the hypre linear solve routine PCG with a BoomerAMG preconditioner, as we specified above.

In the next example (Program 2.5), we will examine how to use hypre preconditioners with Belos solvers.

Program 2.5. Hypre_BelosEx.cpp

```cpp
#include "BelosConfigDefs.hpp"
#include "BelosLinearProblem.hpp"
#include "BelosTpetraAdapter.hpp"
#include "BelosPseudoBlockCGSolMgr.hpp"
#include "MatrixMarket_Tpetra.hpp"
#include "Tpetra_Map.hpp"
#include "Tpetra_CrsMatrix.hpp"
#include "Tpetra_DefaultPlatform.hpp"
#include "Ifpack2_Preconditioner.hpp"
#include "Ifpack2_Hypre.hpp"
#include "Teuchos_CommandLineProcessor.hpp"
#include "Teuchos_ParameterList.hpp"
#include "Teuchos_StandardCatchMacros.hpp"

int main(int argc, char *argv[]) {
  using Teuchos::Array;
  using Teuchos::RCP;
  using Teuchos::rcp;
  using Teuchos::ParameterList;
  using Ifpack2::FunctionParameter;
  using Ifpack2::Hypre::Prec;

  // Specify types used in this example
  typedef Tpetra::CrsMatrix<>::scalar_type Scalar;
  typedef Tpetra::CrsMatrix<>::local_ordinal_type LO;
  typedef Tpetra::CrsMatrix<>::global_ordinal_type GO;
  typedef Tpetra::CrsMatrix<>::node_type Node;
  typedef Tpetra::DefaultPlatform::DefaultPlatformType Platform;
  typedef Tpetra::CrsMatrix<Scalar> CrsMatrix;
  typedef Tpetra::MultiVector<Scalar> MV;
  typedef Tpetra::Operator<Scalar> OP;
  typedef Ifpack2::Preconditioner<Scalar> Preconditioner;
  typedef Tpetra::Map<> Map;

  // Initialize the MPI session
  Teuchos::oblackholestream blackhole;
  Teuchos::GlobalMPISession mpiSession(&argc,&argv,&blackhole);

  // Get the default communicator and node
  Platform &platform = Tpetra::DefaultPlatform::getDefaultPlatform();
  RCP<const Teuchos::Comm<int> > comm = platform.getComm();
  RCP<Node> node = platform.getNode();

  // Get parameters from command-line processor
  int nx = 10;
  Scalar tol = 1e-6;
  bool verbose = false;
  Teuchos::CommandLineProcessor cmdp(false,true);
  cmdp.setOption("nx",&nx, "Number of mesh points in x direction.");
  cmdp.setOption("tolerance",&tol, "Relative residual used for solver.");
  cmdp.setOption("verbose","quiet",&verbose, "Whether to print a lot of info or a little bit.");
  if(cmdp.parse(argc,argv) != Teuchos::CommandLineProcessor::PARSE_SUCCESSFUL) {
    return -1;
  }

  // Create the row map
  int n = nx*nx;
  RCP<Map> map = rcp(new Map(n,0,comm));

  // Create the 2D Laplace operator
  RCP<CrsMatrix> A = rcp(new CrsMatrix(map,5));
  for(LO i = 0; i<nx; i++) {
    for(LO j = 0; j<nx; j++) {
      GO row = i*nx+j;
      if(!map->isNodeGlobalElement(row))
        continue;

      Array<LO> indices;
      Array<Scalar> values;

      if(i > 0) {
        indices.push_back(row - nx);
        values.push_back(-1.0);
      }
      if(i < nx-1) {
        indices.push_back(row + nx);
        values.push_back(-1.0);
      }
      indices.push_back(row);
      values.push_back(4.0);
      if(j > 0) {
        indices.push_back(row-1);
        values.push_back(-1.0);
      }
      if(j < nx-1) {
        indices.push_back(row+1);
        values.push_back(-1.0);
      }
      A->insertGlobalValues(row,indices,values);
    }
  }
  A->fillComplete();

  // Create the initial guess and right hand side
  RCP<MV> trueX = rcp(new MV(A->getRowMap(),1,false));
  RCP<MV> X = rcp(new MV(A->getRowMap(),1));
  RCP<MV> B = rcp(new MV(A->getRowMap(),1,false));
  trueX->randomize();
  A->apply(*trueX,*B);

  // Create the parameters for hypre
  RCP<FunctionParameter> functs[6];
  functs[0] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetPrintLevel, 1)); // print AMG solution info
  functs[1] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetCoarsenType, 6)); // Falgout coarsening
  functs[2] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetRelaxType, 6)); // Sym GS/Jacobi hybrid
  functs[3] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetNumSweeps, 1)); // Sweeps on each level
  functs[4] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetTol, 0.0)); // Conv tolerance zero
  functs[5] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetMaxIter, 1)); // Do only one iteration!

  // Create the preconditioner
  RCP<Preconditioner> prec = rcp(new Ifpack2::Ifpack2_Hypre<Scalar,LO,GO,Node>(A));
  ParameterList hypreList;
  hypreList.set("SolveOrPrecondition", Prec);
  hypreList.set("Preconditioner", Ifpack2::Hypre::BoomerAMG);
  hypreList.set("NumFunctions", 6);
  hypreList.set<RCP<FunctionParameter>*>("Functions", functs);
  prec->setParameters(hypreList);
  prec->compute();

  // Create the linear problem
  RCP< Belos::LinearProblem<Scalar,MV,OP> > problem = rcp(new Belos::LinearProblem<Scalar,MV,OP>(A,X,B));
  problem->setHermitian();
  problem->setLeftPrec(prec);
  problem->setProblem();

  // Create the parameter list
  RCP<ParameterList> belosList = rcp(new ParameterList());
  belosList->set("Convergence Tolerance", tol);
```

```
129   if(verbose)
130     belosList->set("Verbosity", Belos::Errors + Belos::Warnings + Belos::TimingDetails + Belos::StatusTestDetails);
131   else
132     belosList->set("Verbosity", Belos::Errors + Belos::Warnings);
133 
134   // Create the Belos linear solver
135   RCP< Belos::SolverManager<Scalar,MV,OP> > newSolver = rcp(new Belos::PseudoBlockCGSolMgr<Scalar,MV,OP>(problem,belosList
          ));
136 
137   // Perform solve
138   newSolver->solve();
139 }
```

Lines 1–99 These lines are not substantially different from the previous example. We defined convenient typedefs, then set up our operator, solution vector, and right hand side.

Lines 102–108 This time, we have elected not to use a hypre linear solver, so we only set the parameters related to the AMG preconditioner. Again, it is very important to set the maximum number of iterations if you wish to use AMG as a preconditioner.

Lines 111–118 Create the hypre preconditioner. This time, we specify that we would like to precondition rather than solve, since we will be using a Belos linear solver.

Lines 121–124 Create a Belos::LinearProblem that encapsulates the operator, solution vector, right-hand side, and preconditioner. We also specify that our operator is Hermitian so that Belos allows us to use PCG.

Lines 127–135 Create a Belos linear solver. The Belos solvers have many parameters, but we only specify the convergence tolerance and verbosity (what information will be printed).

Line 138 Solve the linear system using a Belos pseudo-block conjugate gradient solver with hypre's BoomerAMG preconditioner.

Is the data wrapped or copied?

The Tpetra matrix data is deep-copied to a hypre matrix.

Trilinos-SuperLU

Trilinos has a Tpetra-based interface to SuperLU-Dist in the sparse factorization package Amesos2. All Amesos2 solvers can be used the following way:

1. Create the solver using the Amesos2 solver factory, which takes as input a string denoting which solver is to be used (KLU2, MUMPS, PARDISO, etc) and the matrix to be factored.

2. Set the parameters for that solver (optional)[4]

3. Perform a symbolic factorization based on the sparsity pattern of the matrix

4. Perform a numeric factorization based on the entries of the matrix. If the matrix's values changed, this factorization must be performed again.

5. Set the initial guess and right-hand side vectors.

6. Solve the linear system. Note that multiple solves can be done without needing to refactor the matrix.

There are numerous examples on the Amesos2 Doxygen page (`https://trilinos.org/docs/dev/packages/amesos2/doc/html/examples.html`) We now present one such example demonstrating how to use this interface to solve a sparse linear system.

Program 2.6. SuperLU_Amesos2Ex.cpp

```
#include <Teuchos_ScalarTraits.hpp>
#include <Teuchos_RCP.hpp>
#include <Teuchos_GlobalMPISession.hpp>
#include <Teuchos_Tuple.hpp>
#include <Teuchos_VerboseObject.hpp>
#include <Teuchos_ParameterList.hpp>

#include <Tpetra_DefaultPlatform.hpp>
#include <Tpetra_Map.hpp>
#include <Tpetra_MultiVector.hpp>
#include <Tpetra_CrsMatrix.hpp>

#include "Amesos2.hpp"
#include "Amesos2_Version.hpp"


int main(int argc, char *argv[]) {
  typedef double Scalar;
  typedef Teuchos::ScalarTraits<Scalar>::magnitudeType Magnitude;

  typedef double Scalar;
  typedef int LO;
  typedef int GO;

  typedef Tpetra::CrsMatrix<Scalar,LO,GO> MAT;
  typedef Tpetra::MultiVector<Scalar,LO,GO> MV;

  using Tpetra::global_size_t;
  using Teuchos::tuple;
  using Teuchos::RCP;
  using Teuchos::rcp;

  Teuchos::GlobalMPISession mpiSession(&argc,&argv);
  Teuchos::RCP<const Teuchos::Comm<int> > comm = Tpetra::DefaultPlatform::getDefaultPlatform().getComm();
  size_t myRank = comm->getRank();

  RCP<Teuchos::FancyOStream> fos = Teuchos::fancyOStream(Teuchos::rcpFromRef(std::cout));
  if( myRank == 0 ) *fos << Amesos2::version() << std::endl << std::endl;

  // create a Map
  global_size_t nrows = 6;
  RCP<Tpetra::Map<LO,GO> > map = rcp( new Tpetra::Map<LO,GO>(nrows,0,comm) );
  RCP<MAT> A = rcp( new MAT(map,3) ); // max of three entries in a row

```

[4]Parameters are documented at `https://trilinos.org/docs/dev/packages/amesos2/doc/html/group__amesos2__solver__parameters.html`

```
 45    /*
 46     * We will solve a system with a known solution, for which we will be using
 47     * the following matrix:
 48     *
 49     * [ [ 7,  0,  -3, 0, -1,  0 ]
 50     *   [ 2,  8,  0,  0,  0,  0 ]
 51     *   [ 0,  0,  1,  0,  0,  0 ]
 52     *   [-3,  0,  0,  5,  0,  0 ]
 53     *   [ 0, -1,  0,  0,  4,  0 ]
 54     *   [ 0,  0,  0, -2,  0,  6 ] ]
 55     *
 56     */
 57    // Construct matrix
 58    if( myRank == 0 ){
 59      A->insertGlobalValues(0,tuple<GO>(0,2,4),tuple<Scalar>(7,-3,-1));
 60      A->insertGlobalValues(1,tuple<GO>(0,1),tuple<Scalar>(2,8));
 61      A->insertGlobalValues(2,tuple<GO>(2),tuple<Scalar>(1));
 62      A->insertGlobalValues(3,tuple<GO>(0,3),tuple<Scalar>(-3,5));
 63      A->insertGlobalValues(4,tuple<GO>(1,4),tuple<Scalar>(-1,4));
 64      A->insertGlobalValues(5,tuple<GO>(3,5),tuple<Scalar>(-2,6));
 65    }
 66    A->fillComplete();
 67
 68    // Create random X
 69    const size_t numVectors = 1;
 70    RCP<MV> X = rcp(new MV(map,numVectors));
 71    X->randomize();
 72
 73    /* Create B
 74     *
 75     * Use RHS:
 76     *
 77     *  [[-7]
 78     *   [18]
 79     *   [ 3]
 80     *   [17]
 81     *   [18]
 82     *   [28]]
 83     */
 84    RCP<MV> B = rcp(new MV(map,numVectors));
 85    int data[6] = {-7,18,3,17,18,28};
 86    for( int i = 0; i < 6; ++i ){
 87      if( B->getMap()->isNodeGlobalElement(i) ){
 88        B->replaceGlobalValue(i,0,data[i]);
 89      }
 90    }
 91
 92    // Check first whether SuperLU is supported
 93    if( Amesos2::query("SuperLU_DIST") ){
 94
 95      // Constructor from Factory
 96      RCP<Amesos2::Solver<MAT,MV> > solver = Amesos2::create<MAT,MV>("SuperLU_DIST", A, X, B);
 97
 98      solver->symbolicFactorization();
 99      solver->numericFactorization();
100      solver->solve();
101
102      /* Print the solution
103       *
104       * Should be:
105       *
106       *  [[1]
107       *   [2]
108       *   [3]
109       *   [4]
110       *   [5]
111       *   [6]]
112       */
113      X->describe(*fos,Teuchos::VERB_EXTREME);
114    } else {
115      *fos << "SuperLU solver not enabled.  Exiting..." << std::endl;
116    }
117  }
```

Lines 1–31 Include statements and typedefs

Lines 33–35 Initialize MPI

Lines 41–66 Create a map describing the parallel distribution of the matrix rows. Then, create the matrix, specifying that each row will have at most three entries. We then set the entries of the matrix by calling insertGlobalValues.

Lines 68–90 Create the initial guess and right-hand side vector. The initial guess X will be overwritten by the solution computed by SuperLU_Dist.

Lines 92–116 Ask Amesos2 whether SuperLU_Dist has been enabled. If so, create a SuperLU_Dist solver using the Amesos2 solver factory. Note that the interface is the same regardless of the solver; if you wished to use Amesos2's native KLU2 solver, you would simply replace "SuperLU_DIST" in lines 93 and 96 with "KLU2". Perform a symbolic factorization, numeric factorization, then a linear solve.

Sandia National Laboratories