

SANDIA REPORT

SAND2015-XXXX

Unlimited Release

Printed November 9, 2015

xSDK User Manual

Alicia Marie Klinvex

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

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.

Approved for public release; further dissemination unlimited.



Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: <http://www.osti.gov/bridge>

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: <http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online>



xSDK 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 the xSDK.

Acknowledgement

The authors would like to acknowledge the support of [funding sources] that funded development of the xSDK.

Chapter 1

xSDK Installation

The easiest 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 specify those options in the Trilinos configuration script; an example script is presented as Program 1.1.

Program 1.1. Sample configuration script

```
1  #!/bin/bash
2  #
3
4  TRILINOS_HOME=~/.TrilinosDir/github/Trilinos
5
6  PETSC_INCLUDE_DIR=~/.PETScDir/petsc-install/include
7  PETSC_LIB_DIR=~/.PETScDir/petsc-install/lib
8
9  HYPRE_INCLUDE_DIR=~/.PETScDir/petsc-install/include
10 HYPRE_LIB_DIR=~/.PETScDir/petsc-install/lib
11
12 METIS_INCLUDE_DIR=~/.parmetis-4.0.2/metis/include
13 METIS_LIB_DIR=~/.parmetis-4.0.2/build/Linux-x86_64/libmetis
14
15 PARMETIS_INCLUDE_DIR=~/.parmetis-4.0.2/include
16 PARMETIS_LIB_DIR=~/.parmetis-4.0.2/build/Linux-x86_64/libparmetis
17
18 SUPERLU_INCLUDE_DIR=~/.SuperLU/superlu_dist/include
19 SUPERLU_LIB_DIR=~/.SuperLU/superlu_dist/lib
20
21 rm -rf CMakeFiles CMakeCache.txt
22
23 cmake \
24   -D CMAKE_BUILD_TYPE=DEBUG \
25   -D CMAKE_INSTALL_PREFIX:PATH="/home/amklinv/TrilinosDir/trilinos-install" \
26   -D TPL_ENABLE_MPI:BOOL=ON \
27   -D Trilinos_ENABLE_EXPLICIT_INSTANTIATION:BOOL=ON \
28 \
29   -D BLAS_LIBRARY_NAMES:STRING="libf77blas.so.3" \
30   -D BLAS_LIBRARY_DIRS:PATH="/usr/lib64/atlas" \
31   -D LAPACK_LIBRARY_NAMES:STRING="liblapack.so.3" \
32   -D LAPACK_LIBRARY_DIRS:PATH="/usr/lib64/atlas" \
33 \
34   -D TPL_ENABLE_PETSC:BOOL=ON \
35   -D PETSC_LIBRARY_DIRS:FILEPATH="${PETSC_LIB_DIR}" \
36   -D PETSC_INCLUDE_DIRS:FILEPATH="${PETSC_INCLUDE_DIR}" \
37 \
38   -D TPL_ENABLE_HYPRE:BOOL=ON \
39   -D HYPRE_LIBRARY_DIRS:FILEPATH="${HYPRE_LIB_DIR}" \
40   -D HYPRE_INCLUDE_DIRS:FILEPATH="${HYPRE_INCLUDE_DIR}" \
41 \
42   -D TPL_ENABLE_ParMETIS:BOOL=ON \
43   -D ParMETIS_LIBRARY_DIRS:FILEPATH="${PARMETIS_LIB_DIR};${METIS_LIB_DIR}" \
44   -D TPL_ParMETIS_INCLUDE_DIRS:FILEPATH="${PARMETIS_INCLUDE_DIR};${METIS_INCLUDE_DIR}" \
45 \
46   -D TPL_ENABLE_SuperLUDist:BOOL=ON \
47   -D SuperLUDist_LIBRARY_DIRS:FILEPATH="${SUPERLU_LIB_DIR}" \
48   -D SuperLUDist_INCLUDE_DIRS:FILEPATH="${SUPERLU_INCLUDE_DIR}" \
49 \
50   -D Trilinos_ENABLE_Amesos2:BOOL=ON \
51   -D Amesos2_ENABLE_TESTS:BOOL=ON \
52   -D Amesos2_ENABLE_EXAMPLES:BOOL=ON \
53 \
54   -D Trilinos_EXTRA_REPOSITORIES:STRING=preCopyrightTrilinos \
```

```

55 -D Trilinos_ENABLE_xSDKTrilinos:BOOL=ON \
56 -D xSDKTrilinos_USING_PETSC:BOOL=ON \
57 -D xSDKTrilinos_USING_HYPRE:BOOL=ON \
58 -D xSDKTrilinos_ENABLE_TESTS:BOOL=ON \
59 -D xSDKTrilinos_ENABLE_EXAMPLES:BOOL=ON \
60 \
61 ${TRILINOS_HOME}

```

Let's examine this script in more detail. The first thing we do is define where the Trilinos source code is located. Then, we set the paths to the include directory and library directory for each package 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>). Third Party Libraries 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.

Chapter 2

xSDK 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, specifically Poisson2D, using Trilinos' Anasazi package.

Program 2.1. PETSc_AnasaziEx.cpp

```
1  #include "petscksp.h"
2  #include "AnasaziBasicEigenproblem.hpp"
3  #include "AnasaziConfigDefs.hpp"
4  #include "AnasaziTpetraAdapter.hpp"
5  #include "AnasaziRTSOLMgr.hpp"
6  #include "Teuchos_ParameterList.hpp"
7  #include "Tpetra_PETScAIJMatrix.hpp"
8
9  int main(int argc, char **args)
10 {
11     using Teuchos::RCP;
12     using Teuchos::rcp;
13     using std::cout;
14     using std::endl;
15
16     typedef Tpetra::PETScAIJMatrix<>          PETScAIJMatrix;
17     typedef PETScAIJMatrix::scalar_type       Scalar;
18     typedef PETScAIJMatrix::local_ordinal_type LO;
19     typedef PETScAIJMatrix::global_ordinal_type GO;
20     typedef PETScAIJMatrix::node_type         Node;
21     typedef Tpetra::Vector<Scalar, LO, GO, Node> Vector;
22     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::Operator¹, 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 Hermitian².

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.

¹RowMatrix 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.

²Some 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

```
1  #include <Teuchos_ScalarTraits.hpp>
2  #include <Teuchos_RCP.hpp>
3  #include <Teuchos_GlobalMpiSession.hpp>
4  #include <Teuchos_oblackholestream.hpp>
5  #include <Teuchos_Tuple.hpp>
6  #include <Teuchos_VerboseObject.hpp>
7  #include <Tpetra_DefaultPlatform.hpp>
8  #include <Tpetra_Map.hpp>
9  #include <Tpetra_MultiVector.hpp>
10 #include <Tpetra_CrsMatrix.hpp>
11 #include "Amesos2.hpp"
12 #include "Amesos2_Version.hpp"
13 #include "Amesos2_KLU2.hpp"
14 #include "petscksp.h"
15 #include "Tpetra_PETScAIJMatrix.hpp"
16 #include "Tpetra_Vector.hpp"
17
18 int main(int argc, char **args)
19 {
20     using Teuchos::RCP;
21     using Teuchos::rcp;
22     using Teuchos::ArrayView;
23
24     typedef Tpetra::PETScAIJMatrix<>      PETScAIJMatrix;
25     typedef PETScAIJMatrix::scalar_type   Scalar;
26     typedef PETScAIJMatrix::local_ordinal_type LO;
27     typedef PETScAIJMatrix::global_ordinal_type GO;
28     typedef PETScAIJMatrix::node_type     Node;
29
30     typedef Tpetra::CrsMatrix<Scalar, LO, GO> CrsMatrix;
31     typedef Tpetra::Vector<Scalar, LO, GO>   Vector;
32     typedef Tpetra::Map<LO, GO>              Map;
33     typedef Tpetra::Operator<Scalar, LO, GO> OP;
34     typedef Tpetra::MultiVector<Scalar, LO, GO> MV;
35     typedef Tpetra::Vector<Scalar, LO, GO>   Vector;
36     typedef Amesos2::Solver<CrsMatrix, MV>    Solver;
37
38     Vec      x, b;
39     Mat      A;
40     PetscRandom rctx;
41     PetscInt  i, j, Ii, J, Istart, Iend;
42     PetscInt  m = 4, n = 4;
43     PetscErrorCode ierr;
44     PetscScalar v;
45     PetscInt rank=0;
46     MPI_Comm comm;
47
48     // Start PETSc
49     PetscInitialize(&argc, &args, NULL, NULL);
50
51     // Create the matrix
52     ierr = MatCreate(PETSC_COMM_WORLD, &A); CHKERRQ(ierr);
53     ierr = MatSetSizes(A, PETSC_DECIDE, PETSC_DECIDE, m*n, m*n); CHKERRQ(ierr);
54     ierr = MatSetType(A, MATAIJ); CHKERRQ(ierr);
55     ierr = MatMPIAIJSetPreallocation(A, 5, PETSC_NULL, 5, PETSC_NULL); CHKERRQ(ierr);
56     ierr = MatSetUp(A); CHKERRQ(ierr);
57     PetscObjectGetComm((PetscObject)A, &comm);
58     ierr = MPI_Comm_rank(comm, &rank); CHKERRQ(ierr);
59
60     ierr = MatGetOwnershipRange(A, &Istart, &Iend); CHKERRQ(ierr);
61     for (Ii=Istart; Ii<Iend; Ii++) {
62         v = -1.0; i = Ii/n; j = Ii - i*n;
63         if (i>0) {J = Ii - n; ierr = MatSetValues(A, 1, &Ii, 1, &J, &v, INSERT_VALUES); CHKERRQ(ierr);}
64         if (i<m-1) {J = Ii + n; ierr = MatSetValues(A, 1, &Ii, 1, &J, &v, INSERT_VALUES); CHKERRQ(ierr);}
65         if (j>0) {J = Ii - 1; ierr = MatSetValues(A, 1, &Ii, 1, &J, &v, INSERT_VALUES); CHKERRQ(ierr);}
66         if (j<n-1) {J = Ii + 1; ierr = MatSetValues(A, 1, &Ii, 1, &J, &v, INSERT_VALUES); CHKERRQ(ierr);}
67         v = 4.0; ierr = MatSetValues(A, 1, &Ii, 1, &Ii, &v, INSERT_VALUES); CHKERRQ(ierr);
68     }
69
70     ierr = MatAssemblyBegin(A, MAT_FINAL_ASSEMBLY); CHKERRQ(ierr);
71     ierr = MatAssemblyEnd(A, MAT_FINAL_ASSEMBLY); CHKERRQ(ierr);
72
73     // Create a random solution vector and corresponding right-hand-side
74     ierr = VecCreate(PETSC_COMM_WORLD, &x); CHKERRQ(ierr);
75     ierr = VecSetSizes(x, PETSC_DECIDE, m*n); CHKERRQ(ierr);
76     ierr = VecSetFromOptions(x); CHKERRQ(ierr);
77     ierr = PetscRandomCreate(PETSC_COMM_WORLD, &rctx); CHKERRQ(ierr);
78     ierr = PetscRandomSetFromOptions(rctx); CHKERRQ(ierr);
79     ierr = VecSetRandom(x, rctx); CHKERRQ(ierr);
80     ierr = PetscRandomDestroy(&rctx); CHKERRQ(ierr);
81     ierr = VecDuplicate(x, &b); CHKERRQ(ierr);
82     ierr = MatMult(A, x, b); CHKERRQ(ierr);
83
84     // Copy the matrix from a PETSc data structure to a Tpetra CrsMatrix
85     RCP<CrsMatrix> tpetraA = xSDKTrilinos::deepCopyPETScAIJMatrixToTpetraCrsMatrix<Scalar, LO, GO, Node>(A);
86
87     // Copy the PETSc vectors to Tpetra vectors
88     RCP<Vector> tpetraX = xSDKTrilinos::deepCopyPETScVecToTpetraVector<Scalar, LO, GO, Node>(x);
89     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::Multi-Vector, with a PETSc KSP linear solver, you may use the new Belos³ interface: PETSc-SolMgr. 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.

³Belos 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

```

1  #include "BelosConfigDefs.hpp"
2  #include "BelosLinearProblem.hpp"
3  #include "BelosTpetraAdapter.hpp"
4  #include "BelosPETScSolMgr.hpp"
5  #include "Ifpack2_Factory.hpp"
6  #include "Teuchos_CommandLineProcessor.hpp"
7  #include "Teuchos_ParameterList.hpp"
8  #include "Teuchos_StandardCatchMacros.hpp"
9  #include "Tpetra_CrsMatrix.hpp"
10 #include "Tpetra_DefaultPlatform.hpp"
11 #include "Tpetra_MultiVector.hpp"
12 #include "MatrixMarket_Tpetra.hpp"
13
14 int main(int argc, char *argv[]) {
15     typedef Tpetra::MultiVector<>      MV;
16     typedef Tpetra::Operator<>          OP;
17     typedef Tpetra::CrsMatrix<>         CrsMatrix;
18     typedef Ifpack2::Preconditioner<>   Prec;
19

```

```

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

```
1  #include "Tpetra_Map.hpp"
2  #include "Tpetra_CrsMatrix.hpp"
3  #include "Tpetra_DefaultPlatform.hpp"
4  #include "Ifpack2_Preconditioner.hpp"
5  #include "Ifpack2_Hypre.hpp"
6  #include "Teuchos_CommandLineProcessor.hpp"
7  #include "Teuchos_ParameterList.hpp"
8  #include "Teuchos_StandardCatchMacros.hpp"
9
10 int main(int argc, char *argv[]) {
11     using Teuchos::Array;
12     using Teuchos::RCP;
13     using Teuchos::rcp;
14     using Teuchos::ParameterList;
15     using Ifpack2::FunctionParameter;
16     using Ifpack2::Hypre::Prec;
17     using Ifpack2::Hypre::Solver;
18
19     typedef Tpetra::CrsMatrix<>::scalar_type Scalar;
20     typedef Tpetra::CrsMatrix<>::local_ordinal_type LO;
21     typedef Tpetra::CrsMatrix<>::global_ordinal_type GO;
22     typedef Tpetra::CrsMatrix<>::node_type Node;
23     typedef Tpetra::DefaultPlatform::DefaultPlatformType Platform;
24     typedef Tpetra::CrsMatrix<Scalar> CrsMatrix;
25     typedef Tpetra::MultiVector<Scalar> MV;
26     typedef Tpetra::Operator<Scalar> OP;
27     typedef Ifpack2::Preconditioner<Scalar> Preconditioner;
28     typedef Tpetra::Map<> Map;
29
30     // Initialize the MPI session
31     Teuchos::oblackholestream blackhole;
32     Teuchos::GlobalMPISession mpiSession(&argc,&argv,&blackhole);
33
34     // Get the default communicator
35     Platform &platform = Tpetra::DefaultPlatform::getDefaultPlatform();
36     RCP<const Teuchos::Comm<int>> > comm = platform.getComm();
37
38     // Get parameters from command-line processor
39     int nx = 10;
40     Scalar tol = 1e-6;
41     bool verbose = false;
42     Teuchos::CommandLineProcessor cmdp(false,true);
43     cmdp.setOption("nx",&nx, "Number of mesh points in x direction.");
44     cmdp.setOption("tolerance",&tol, "Relative residual used for solver.");
45     cmdp.setOption("verbose","quiet",&verbose, "Whether to print a lot of info or a little bit.");
46     if(cmdp.parse(argc,argv) != Teuchos::CommandLineProcessor::PARSE_SUCCESSFUL) {
47         return -1;
48     }
49
50     // Create the row map
51     int n = nx*nx;
52     RCP<Map> map = rcp(new Map(n,0,comm));
53
54     // Create the 2D Laplace operator
55     RCP<CrsMatrix> A = rcp(new CrsMatrix(map,5));
56     for(LO i = 0; i<nx; i++) {
57         for(LO j = 0; j<nx; j++) {
58             GO row = i*nx+j;
59             if(!map->isNodeGlobalElement(row))
60                 continue;
61
62             Array<LO> indices;
63             Array<Scalar> values;
64
65             if(i > 0) {
66                 indices.push_back(row - nx);
67                 values.push_back(-1.0);
68             }
69             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> funts[10];
97 funts[0] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetPrintLevel, 1)); // print AMG solution info
98 funts[1] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetCoarsenType, 6)); // Falgout coarsening
99 funts[2] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetRelaxType, 6)); // Sym GS/Jacobi hybrid
100 funts[3] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetNumSweeps, 1)); // Sweeps on each level
101 funts[4] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetTol, 0.0)); // Conv tolerance zero
102 funts[5] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetMaxIter, 1)); // Do only one iteration!
103 funts[6] = rcp(new FunctionParameter(Solver, &HYPRE_PCGSetMaxIter, 1000)); // Maximum iterations
104 funts[7] = rcp(new FunctionParameter(Solver, &HYPRE_PCGSetTol, tol)); // Convergence tolerance
105 funts[8] = rcp(new FunctionParameter(Solver, &HYPRE_PCGSetTwoNorm, 1)); // Use the two-norm as the stopping
106     criteria
107 funts[9] = rcp(new FunctionParameter(Solver, &HYPRE_PCGSetPrintLevel, 2)); // Print solve info
108
109 // Create the hypre solver
110 RCP<Preconditioner> prec = rcp(new Ifpack2::Ifpack2_Hypre<Scalar,L0,G0,Node>(A));
111 ParameterList hypreList;
112 hypreList.set("SolveOrPrecondition", Solver);
113 hypreList.set("Solver", Ifpack2::Hypre::PCG);
114 hypreList.set("Preconditioner", Ifpack2::Hypre::BoomerAMG);
115 hypreList.set("SetPreconditioner", true);
116 hypreList.set("NumFunctions", 10);
117 hypreList.set<RCP<FunctionParameter>>("Functions", funts);
118 prec->setParameters(hypreList);
119 prec->compute();
120
121 // Perform solve
122 prec->apply(*B,*X);
123 }

```

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 multi-grid 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

```
1  #include "BelosConfigDefs.hpp"
2  #include "BelosLinearProblem.hpp"
3  #include "BelosTpetraAdapter.hpp"
4  #include "BelosPseudoBlockCGSolMgr.hpp"
5  #include "MatrixMarket_Tpetra.hpp"
6  #include "Tpetra_Map.hpp"
7  #include "Tpetra_CrsMatrix.hpp"
8  #include "Tpetra_DefaultPlatform.hpp"
9  #include "Ifpack2_Preconditioner.hpp"
10 #include "Ifpack2_Hypre.hpp"
11 #include "Teuchos_CommandLineProcessor.hpp"
12 #include "Teuchos_ParameterList.hpp"
13 #include "Teuchos_StandardCatchMacros.hpp"
14
15 int main(int argc, char *argv[]) {
16     using Teuchos::Array;
17     using Teuchos::RCP;
18     using Teuchos::rcp;
19     using Teuchos::ParameterList;
20     using Ifpack2::FunctionParameter;
21     using Ifpack2::Hypre::Prec;
22
23     // Specify types used in this example
24     typedef Tpetra::CrsMatrix<>::scalar_type Scalar;
25     typedef Tpetra::CrsMatrix<>::local_ordinal_type LO;
26     typedef Tpetra::CrsMatrix<>::global_ordinal_type GO;
27     typedef Tpetra::CrsMatrix<>::node_type Node;
28     typedef Tpetra::DefaultPlatform::DefaultPlatformType Platform;
29     typedef Tpetra::CrsMatrix<Scalar> CrsMatrix;
30     typedef Tpetra::MultiVector<Scalar> MV;
31     typedef Tpetra::Operator<Scalar> OP;
32     typedef Ifpack2::Preconditioner<Scalar> Preconditioner;
33     typedef Tpetra::Map<> Map;
34
35     // Initialize the MPI session
36     Teuchos::oblackholestream blackhole;
```

```

37 Teuchos::GlobalMPISession mpiSession(&argc,&argv,&blackhole);
38
39 // Get the default communicator and node
40 Platform &platform = Tpetra::DefaultPlatform::getDefaultPlatform();
41 RCP<const Teuchos::Comm<int> > comm = platform.getComm();
42 RCP<Node> node = platform.getNode();
43
44 // Get parameters from command-line processor
45 int nx = 10;
46 Scalar tol = 1e-6;
47 bool verbose = false;
48 Teuchos::CommandLineProcessor cmdp(false,true);
49 cmdp.setOption("nx",&nx, "Number of mesh points in x direction.");
50 cmdp.setOption("tolerance",&tol, "Relative residual used for solver.");
51 cmdp.setOption("verbose","quiet",&verbose, "Whether to print a lot of info or a little bit.");
52 if(cmdp.parse(argc,argv) != Teuchos::CommandLineProcessor::PARSE_SUCCESSFUL) {
53     return -1;
54 }
55
56 // Create the row map
57 int n = nx*nx;
58 RCP<Map> map = rcp(new Map(n,0,comm));
59
60 // Create the 2D Laplace operator
61 RCP<CrsMatrix> A = rcp(new CrsMatrix(map,5));
62 for(L0 i = 0; i<nx; i++) {
63     for(L0 j = 0; j<nx; j++) {
64         GO row = i*nx+j;
65         if(!map->isNodeGlobalElement(row))
66             continue;
67
68         Array<L0> indices;
69         Array<Scalar> values;
70
71         if(i > 0) {
72             indices.push_back(row - nx);
73             values.push_back(-1.0);
74         }
75         if(i < nx-1) {
76             indices.push_back(row + nx);
77             values.push_back(-1.0);
78         }
79         indices.push_back(row);
80         values.push_back(4.0);
81         if(j > 0) {
82             indices.push_back(row-1);
83             values.push_back(-1.0);
84         }
85         if(j < nx-1) {
86             indices.push_back(row+1);
87             values.push_back(-1.0);
88         }
89         A->insertGlobalValues(row,indices,values);
90     }
91 }
92 A->fillComplete();
93
94 // Create the initial guess and right hand side
95 RCP<MV> trueX = rcp(new MV(A->getRowMap(),1,false));
96 RCP<MV> X = rcp(new MV(A->getRowMap(),1));
97 RCP<MV> B = rcp(new MV(A->getRowMap(),1,false));
98 trueX->randomize();
99 A->apply(*trueX,*B);
100
101 // Create the parameters for hypre
102 RCP<FunctionParameter> functs[6];
103 functs[0] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetPrintLevel, 1)); // print AMG solution info
104 functs[1] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetCoarsenType, 6)); // Falgout coarsening
105 functs[2] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetRelaxType, 6)); // Sym GS/Jacobi hybrid
106 functs[3] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetNumSweeps, 1)); // Sweeps on each level
107 functs[4] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetTol, 0.0)); // Conv tolerance zero
108 functs[5] = rcp(new FunctionParameter(Prec, &HYPRE_BoomerAMGSetMaxIter, 1)); // Do only one iteration!
109
110 // Create the preconditioner
111 RCP<Preconditioner> prec = rcp(new Ifpack2::Ifpack2_Hypre<Scalar,L0,GO,Node>(A));
112 ParameterList hypreList;
113 hypreList.set("SolveOrPrecondition", Prec);
114 hypreList.set("Preconditioner", Ifpack2::Hypre::BoomerAMG);
115 hypreList.set("NumFunctions", 6);
116 hypreList.set(RCP<FunctionParameter>*("Functions", functs);
117 prec->setParameters(hypreList);
118 prec->compute();
119
120 // Create the linear problem
121 RCP< Belos::LinearProblem<Scalar,MV,OP> > problem = rcp(new Belos::LinearProblem<Scalar,MV,OP>(A,X,B));
122 problem->setHermitian();
123 problem->setLeftPrec(prec);
124 problem->setProblem();
125
126 // Create the parameter list
127 RCP<ParameterList> belosList = rcp(new ParameterList());
128 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 hypr 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 hypr 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 hypr’s BoomerAMG preconditioner.

Is the data wrapped or copied?

The Tpetra matrix data is deep-copied to a hypr 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

```

1  #include <Teuchos_ScalarTraits.hpp>
2  #include <Teuchos_RCP.hpp>
3  #include <Teuchos_GlobalMPISession.hpp>
4  #include <Teuchos_Tuple.hpp>
5  #include <Teuchos_VerboseObject.hpp>
6  #include <Teuchos_ParameterList.hpp>
7
8  #include <Tpetra_DefaultPlatform.hpp>
9  #include <Tpetra_Map.hpp>
10 #include <Tpetra_MultiVector.hpp>
11 #include <Tpetra_CrsMatrix.hpp>
12
13 #include "Amesos2.hpp"
14 #include "Amesos2_Version.hpp"
15
16
17 int main(int argc, char *argv[]) {
18     typedef double Scalar;
19     typedef Teuchos::ScalarTraits<Scalar>::magnitudeType Magnitude;
20
21     typedef double Scalar;
22     typedef int LO;
23     typedef int GO;
24
25     typedef Tpetra::CrsMatrix<Scalar,LO,GO> MAT;
26     typedef Tpetra::MultiVector<Scalar,LO,GO> MV;
27
28     using Tpetra::global_size_t;
29     using Teuchos::tuple;
30     using Teuchos::RCP;
31     using Teuchos::rcp;
32
33     Teuchos::GlobalMPISession mpiSession(&argc,&argv);
34     Teuchos::RCP<const Teuchos::Comm<int>> comm = Tpetra::DefaultPlatform::getDefaultPlatform().getComm();
35     size_t myRank = comm->getRank();
36
37     RCP<Teuchos::FancyOStream> fos = Teuchos::fancyOStream(Teuchos::rcpFromRef(std::cout));
38     if( myRank == 0 ) *fos << Amesos2::version() << std::endl << std::endl;
39
40     // create a Map
41     global_size_t nrows = 6;
42     RCP<Tpetra::Map<LO,GO>> map = rcp( new Tpetra::Map<LO,GO>(nrows,0,comm) );
43     RCP<MAT> A = rcp( new MAT(map,3) ); // max of three entries in a row
44

```

⁴Parameters 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<G0>(0,2,4),tuple<Scalar>(7,-3,-1));
60      A->insertGlobalValues(1,tuple<G0>(0,1),tuple<Scalar>(2,8));
61      A->insertGlobalValues(2,tuple<G0>(2),tuple<Scalar>(1));
62      A->insertGlobalValues(3,tuple<G0>(0,3),tuple<Scalar>(-3,5));
63      A->insertGlobalValues(4,tuple<G0>(1,4),tuple<Scalar>(-1,4));
64      A->insertGlobalValues(5,tuple<G0>(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.

