SANDIA REPORT

SAND2015-XXXX
Unlimited Release
Printed January 19, 2015

xSDK User Manual

Alicia Marie Klinvex

Sandia National Laboratories

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

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

SAND2015-XXXX
Unlimited Release
Printed January 19, 2015

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.

Introduction

The following are the libraries included in the xSDK. TODO: Maybe each team should write a paragraph or so for this.

Trilinos

The Trilinos Project is an effort to develop algorithms and enabling technologies within an object-oriented software framework for the solution of large-scale, complex multi-physics engineering and scientific problems. Trilinos is organized into 66 different packages, each with a specific focus. These packages include linear and nonlinear solvers, preconditioners (including algebraic multigrid), graph partitioners, eigensolvers, and optimization algorithms, among other things. Users are only required to install the subset of packages related to the problems they are trying to solve.

Trilinos supports MPI+X, where X can be CUDA, OpenMP, etc.

PETSc

PETSc is a suite of data structures and routines for the scalable solution of scientific applications modeled by partial differential equations. It includes linear solvers, nonlinear solvers, and preconditioners. PETSc does not currently support threads, and it does not support solving linear systems with multiple right-hand-sides. While it does not include eigensolvers, there is an eigensolver package called SLEPc built on top of PETSc with a very similar interface.

hypre

Hypre is a library for solving large, sparse linear systems of equations on massively parallel computers. While it contains linear solvers and an eigensolver called LOBPCG, it is probably most well-known for its preconditioners.

Certain subsets of hypre are multithreaded.

SuperLU

SuperLU is a general purpose library for the direct solution of large, sparse, nonsymmetric systems of linear equations on high performance machines. There are three separate versions of this code: SuperLU (for sequential machines), SuperLU_MT (for shared memory parallel machines), and SuperLU_DIST (for distributed memory).

Chapter 1

xSDK Installation

TODO: Ignore this. Barry is going to take care of the circular dependency issue; then we can revisit the installation procedure.

Using PETSc configuration script

The easiest way to install the various xSDK libaries (Trilinos, PETSc, Hypre, and SuperLU) is the following three step process:

1. Download and unpack the PETSc tarball from `http://www.mcs.anl.gov/petsc/`, or pull from the repo via `git clone https://bitbucket.org/petsc/petsc`.
2. Run the PETSc configuration script.
3. Type `make`, then `make install`. Don't worry; the PETSc configuration script will remind you to do this.

The PETSc configuration script is very sophisticated, as you can tell by typing `./configure --help`. It will produce a list for you of all the different configuration options (organized into different categories). The most interesting of these configuation options is `--download-<PACKAGE>`, where `PACKAGE` can be hypre, superlu, trilinos, etc. If you install PETSc via the following line: `./configure --prefix=<HOME>/petsc-install --download-hypre --download-superlu` PETSc will download and install hypre and SuperLU for you, and PETSc itself will be installed to the directory `<HOME>/petsc-install`. If PETSc encounters any problems, such as being unable to find your MPI installation, it will output a helpful error message explaining what the problem is and how you can fix it.

If PETSc finds itself unable to download any packages you request (because you are behind a firewall, for instance), it will output the following message explaining how to fix the problem:

```
===============================================================================
  Trying to download http://ftp.mcs.anl.gov/pub/petsc/externalpackages/hypre-2.10.0b-p1.tar.gz for HYPRE
```

```
===============================================================================
===============================================================================
  Trying to download ftp://ftp.mcs.anl.gov/pub/petsc/externalpackages/hypre-2.10.0b-p1.tar.gz for HYPRE
===============================================================================
-------------------------------------------------------------------------------
Unable to download package hypre from: http://ftp.mcs.anl.gov/pub/petsc/externalpackages/hypre-2.10.0b-p1.tar.gz
* If URL specified manually - perhaps there is a typo?
* If your network is disconnected - please reconnect and rerun ./configure
* Or perhaps you have a firewall blocking the download
* Alternatively, you can download the above URL manually, to /yourselectedlocation/hypre-2.10.0b-p1.tar.gz
  and use the configure option:
  --download-hypre=/yourselectedlocation/hypre-2.10.0b-p1.tar.gz
Unable to download package hypre from: ftp://ftp.mcs.anl.gov/pub/petsc/externalpackages/hypre-2.10.0b-p1.tar.gz
* If URL specified manually - perhaps there is a typo?
* If your network is disconnected - please reconnect and rerun ./configure
* Or perhaps you have a firewall blocking the download
* Alternatively, you can download the above URL manually, to /yourselectedlocation/hypre-2.10.0b-p1.tar.gz
  and use the configure option:
  --download-hypre=/yourselectedlocation/hypre-2.10.0b-p1.tar.gz
*******************************************************************************
```

Without PETSc configuration script

If you would like to manually install Trilinos and enable the PETSc, hypre, and SuperLU interfaces, you may specify those options in the Trilinos configuration script (which I will call do-configure). Note that Trilinos requires all third party libraries to be installed before the configuration process. An example configuration script is below:

Program 1.1. do-configure

```
#!/bin/bash
#

TRILINOS_HOME=/home/amklinv/TrilinosDir/Trilinos
PETSC_DIR=/home/amklinv/PETSc/petsc-3.6.0
PETSC_ARCH=arch-linux2-cxx-debug
PETSC_LIB="-Wl,-rpath,/home/amklinv/PETSc/petsc-3.6.0/arch-linux2-cxx-debug/lib -L/home/amklinv/PETSc/petsc-3.6.0/arch-
     linux2-cxx-debug/lib -lpetsc -Wl,-rpath,/home/amklinv/PETSc/petsc_install/lib -L/home/amklinv/PETSc/petsc_install/lib
      -lsuperlu_4.3 -lsuperlu_dist_4.0 -lHYPRE -Wl,-rpath,/home/amklinv/lapack-3.5.0 -L/home/amklinv/lapack-3.5.0 -llapack
      -lrefblas -lparmetis -lmetis -L/projects/install/rhel6-x86_64/sems/compiler/gcc/4.7.2/openmpi/1.6.5/lib -L/projects/
     install/rhel6-x86_64/sems/compiler/gcc/4.7.2/base/lib/gcc/x86_64-unknown-linux-gnu/4.7.2 -L/projects/install/rhel6-
     x86_64/sems/compiler/gcc/4.7.2/base/lib64 -L/projects/install/rhel6-x86_64/sems/compiler/gcc/4.7.2/base/lib -lmpi_f90
      -lmpi_f77 -lgfortran -lm -lgfortran -lm -lgfortran -lm -lgfortran -lm -lquadmath -lm -lmpi_cxx -lstdc++ -L/projects/
     install/rhel6-x86_64/sems/compiler/gcc/4.7.2/openmpi/1.6.5/lib -L/projects/install/rhel6-x86_64/sems/compiler/gcc
     /4.7.2/base/lib/gcc/x86_64-unknown-linux-gnu/4.7.2 -L/projects/install/rhel6-x86_64/sems/compiler/gcc/4.7.2/base/
     lib64 -L/projects/install/rhel6-x86_64/sems/compiler/gcc/4.7.2/base/lib -ldl -lmpi -lrt -lnsl -lutil -lgcc_s -
     lpthread -ldl"

PETSC_INCLUDE_PATH="${PETSC_DIR}/${PETSC_ARCH}/include;${PETSC_DIR}/include;${PETSC_DIR}"

HYPRE_LIBRARY_DIRS="/home/amklinv/PETSc/petsc-3.6.0/arch-linux2-cxx-debug/externalpackages/hypre-2.10.0b-p1/src/hypre/lib"
HYPRE_INCLUDE_DIRS="/home/amklinv/PETSc/petsc-3.6.0/arch-linux2-cxx-debug/externalpackages/hypre-2.10.0b-p1/src/hypre/
     include"

rm -rf CMakeFiles CMakeCache.txt

cmake \
  -D CMAKE_BUILD_TYPE=DEBUG \
  -D Trilinos_ENABLE_STRONG_C_COMPILE_WARNINGS:BOOL=OFF \
  -D CMAKE_INSTALL_PREFIX:PATH="/home/amklinv/TrilinosDir\trilinos-install" \
  -D TPL_ENABLE_MPI:BOOL=ON \
  -D Trilinos_ENABLE_EXPLICIT_INSTANTIATION:BOOL=ON \
  -D Trilinos_ENABLE_ALL_OPTIONAL_PACKAGES:BOOL=OFF \
  -D Trilinos_ENABLE_Amesos:BOOL=ON \
  -D Trilinos_ENABLE_Amesos2:BOOL=ON \
  -D Trilinos_ENABLE_Anasazi:BOOL=ON \
  -D Trilinos_ENABLE_AztecOO:BOOL=ON \
  -D Trilinos_ENABLE_Belos:BOOL=ON \
  -D Trilinos_ENABLE_Epetra:BOOL=ON \
  -D Trilinos_ENABLE_EpetraExt:BOOL=ON \
30   -D Trilinos_ENABLE_Galeri:BOOL=ON \
31   -D Trilinos_ENABLE_Ifpack:BOOL=ON \
32   -D Trilinos_ENABLE_Ifpack2:BOOL=ON \
33   -D Trilinos_ENABLE_Isorropia:BOOL=ON \
34   -D Trilinos_ENABLE_Kokkos:BOOL=ON \
35   -D Trilinos_ENABLE_ML:BOOL=ON \
36   -D Trilinos_ENABLE_Tpetra:BOOL=ON \
37   -D Trilinos_ENABLE_TrilinosCouplings:BOOL=ON \
38   -D Trilinos_ENABLE_Triutils:BOOL=ON \
39   -D Trilinos_ENABLE_NOX:BOOL=ON \
40   -D Trilinos_ENABLE_Zoltan:BOOL=ON \
41   -D Trilinos_ENABLE_Zoltan2:BOOL=ON \
42 \
43   -D EpetraExt_USING_PETSC:BOOL=ON \
44   -D NOX_ENABLE_PETSC:BOOL=ON \
45   -D NOX_ENABLE_ABSTRACT_IMPLEMENTATION_PETSC:BOOL=ON \
46   -D TPL_ENABLE_PETSC:BOOL=ON \
47   -D PETSC_LIBRARY_DIRS:FILEPATH="${PETSC_LIB}" \
48   -D PETSC_INCLUDE_DIRS:FILEPATH="${PETSC_INCLUDE_PATH}" \
49   -D TPL_PETSC_LIBRARIES:STRING="${PETSC_LIB}" \
50   -D TPL_PETSC_INCLUDE_DIRS:STRING="${PETSC_INCLUDE_PATH}" \
51 \
52   -D Amesos2_ENABLE_KLU2::BOOL=ON \
53 \
54   -D EpetraExt_ENABLE_HYPRE:BOOL=ON \
55   -D Ifpack_ENABLE_HYPRE:BOOL=ON \
56   -D TPL_ENABLE_HYPRE:BOOL=ON \
57   -D HYPRE_LIBRARY_DIRS:FILEPATH="${HYPRE_LIBRARY_DIRS}" \
58   -D HYPRE_INCLUDE_DIRS:FILEPATH="${HYPRE_INCLUDE_DIRS}" \
59 \
60   -D Belos_ENABLE_TESTS:BOOL=ON \
61   -D Belos_ENABLE_EXAMPLES:BOOL=ON \
62   -D Ifpack2_ENABLE_TESTS:BOOL=ON \
63   -D Ifpack2_ENABLE_EXAMPLES:BOOL=ON \
64   -D DART_TESTING_TIMEOUT:STRING=300 \
65   -D BLAS_LIBRARY_NAMES:STRING="libf77blas.so.3" \
66   -D BLAS_LIBRARY_DIRS:PATH="/usr/lib64/atlas" \
67   -D LAPACK_LIBRARY_NAMES:STRING="liblapack.so.3" \
68   -D LAPACK_LIBRARY_DIRS:PATH="/usr/lib64/atlas" \
69 \
70   -D Trilinos_EXTRA_REPOSITORIES:STRING=preCopyrightTrilinos \
71   -D Trilinos_ENABLE_xSDKTrilinos:BOOL=ON \
72   -D xSDKTrilinos_USING_PETSC:BOOL=ON \
73   -D xSDKTrilinos_USING_HYPRE:BOOL=ON \
74 \
75   ${TRILINOS_HOME}
```

Let's examine this script in more detail. Since we are interested in the PETSc interface, we must first define `PETSC_DIR` and `PETSC_ARCH`. `PETSC_LIB` looks intimidating at first, but you can generate it automatically by typing **`make getlinklibs`** in `PETSC_DIR`. We then define the PETSc and hypre include paths. Everything else looks pretty normal until we reach lines 43–50, which enable the Epetra-based PETSc interface. Lines 54–58 enable the Epetra-based hypre interface. Then lines 70–73 enable the Tpetra-based interfaces living in the pre-Copyright package xSDKTrilinos.

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

TODO: Introduction and examples. We have two new interfaces in Trilinos 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 our 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 **??**) 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;
37
38    PetscInitialize(&argc,&args,NULL,NULL);
39    ierr = PetscOptionsGetInt(PETSC_NULL,"-mx",&m,PETSC_NULL);CHKERRQ(ierr);
40    ierr = PetscOptionsGetInt(PETSC_NULL,"-my",&n,PETSC_NULL);CHKERRQ(ierr);
41    ierr = PetscOptionsGetInt(PETSC_NULL,"-nev",&nev,PETSC_NULL);CHKERRQ(ierr);
42
43    ierr = MatCreate(PETSC_COMM_WORLD,&A);CHKERRQ(ierr);
44    ierr = MatSetSizes(A,PETSC_DECIDE,PETSC_DECIDE,m*n,m*n);CHKERRQ(ierr);
45    ierr = MatSetType(A, MATAIJ);CHKERRQ(ierr);
46    ierr = MatMPIAIJSetPreallocation(A,5,PETSC_NULL,5,PETSC_NULL);CHKERRQ(ierr);
47    ierr = MatSetUp(A);CHKERRQ(ierr);
48    ierr = PetscObjectGetComm( (PetscObject)A, &comm);CHKERRQ(ierr);
49    ierr = MPI_Comm_rank(comm,&rank);CHKERRQ(ierr);
50
51    ierr = MatGetOwnershipRange(A,&Istart,&Iend);CHKERRQ(ierr);
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    RCP<PETScAIJMatrix> tpetraA = rcp(new PETScAIJMatrix(A));
65
66    RCP<MV> initGuess = rcp(new MV(tpetraA->getDomainMap(),4,false));
67    initGuess->randomize();
68
69    RCP<Problem> problem = rcp(new Problem(tpetraA,initGuess));
70    problem->setNEV(nev);
71    problem->setHermitian(true);
72    problem->setProblem();
73
74    Teuchos::ParameterList pl;
75    pl.set("Verbosity", Anasazi::IterationDetails + Anasazi::FinalSummary);
76    pl.set("Convergence Tolerance", 1e-6);
77    RCP<SolMgr> solver = rcp(new SolMgr(problem, pl));
78
79    Anasazi::ReturnType returnCode = solver->solve();
80    Anasazi::Eigensolution<Scalar,MV> sol = problem->getSolution();
81    std::vector<Anasazi::Value<Scalar> > evals = sol.Evals;
82    RCP<MV> evecs = sol.Evecs;
83
84    ierr = PetscFinalize();CHKERRQ(ierr);
85    return 0;
86  }
```

Lines 1–7 Include statements

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

Lines 30–36 PETSc datatypes

Lines 38–41 Get the command line arguments using PETSc. This example has three of them: the number of mesh points in the x direction, number of mesh points in the y direction, and the number of desired eigenpairs.

Lines 43–62 Create the PETSc Mat and set its values.

Line 64 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.

Line 66 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().

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 our new Belos 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.
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 2.2) illustrates this process in greater detail.

[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.

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

Program 2.2. Tpetra_KSPEx.cpp

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

int main(int argc, char *argv[]) {
  typedef double                              ST;
  typedef Teuchos::ScalarTraits<ST>          SCT;
  typedef SCT::magnitudeType                  MT;
  typedef Tpetra::MultiVector<>               MV;
  typedef Tpetra::Operator<>                  OP;
  typedef Belos::MultiVecTraits<ST,MV>       MVT;
  typedef Belos::OperatorTraits<ST,MV,OP>    OPT;
  typedef Tpetra::CrsMatrix<>                CrsMatrix;
  typedef Ifpack2::Preconditioner<>          Prec;

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

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

  double tol = 1e-6;
  std::string filename("/home/amklinv/matrices/cage4.mtx");
  std::string ksptype("gmres");
  Teuchos::CommandLineProcessor cmdp(false,false);
  cmdp.setOption("filename",&filename,"Filename for test matrix.");
  cmdp.setOption("tol",&tol,"Relative residual tolerance.");
  cmdp.setOption("ksptype",&ksptype,"Type of linear solver to be used.");
  if (cmdp.parse(argc,argv) != Teuchos::CommandLineProcessor::PARSE_SUCCESSFUL) {
    return -1;
  }

  RCP<CrsMatrix> A = Tpetra::MatrixMarket::Reader<CrsMatrix>::readSparseFile(filename,comm);
  RCP<MV> B = rcp(new MV(A->getRowMap(),1,false));
  RCP<MV> X = rcp(new MV(A->getRowMap(),1,false));
  MVT::MvInit(*X);
  MVT::MvInit(*B,1);

  Ifpack2::Factory factory;
49    RCP<Prec> M = factory.create("RELAXATION", A.getConst());
50    ParameterList ifpackParams;
51    ifpackParams.set("relaxation: type","Jacobi");
52    M->setParameters(ifpackParams);
53    M->initialize();
54    M->compute();
55
56    ParameterList belosList;
57    belosList.set( "Maximum Iterations", 100 );
58    belosList.set( "Convergence Tolerance", tol );
59    belosList.set( "Solver", ksptype );
60
61    RCP<Belos::LinearProblem<double,MV,OP> > problem
62      = rcp( new Belos::LinearProblem<double,MV,OP>( A, X, B ) );
63    problem->setLeftPrec( M );
64    problem->setProblem();
65
66    RCP< Belos::PETScSolMgr<double,MV,OP> > solver
67      = rcp( new Belos::PETScSolMgr<double,MV,OP>(problem, rcp(&belosList,false)) );
68    solver->solve();
69  }
```

Lines 1–9 Include statements

Lines 12–24 Typedefs and using statements to make the code more readable

Lines 26–29 Set up MPI

Lines 31–40 Parse command line arguments. This program allows the user to specify the filename for the matrix, the tolerance for the linear solve, and which linear solver is 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`.

Lines 42–46 Set up the linear system by reading the matrix from a file, setting the initial guess for the solution as $\vec{0}$ and setting the right hand side as $\vec{1}$.

Lines 48–54 Set up the Ifpack2 Jacobi preconditioner.

Lines 56–59 Set the maximum number of iterations, convergence tolerance, and which PETSc KSP solver is being used.

Lines 61–64 Set up the linear problem for the Belos solver.

Lines 66-68 Solve the linear system.

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.

Trilinos-hypre

TODO: Introduction.

In this example (Program 2.3), we will examine how to use hypre solvers and preconditioners with Tpetra objects.

Program 2.3. Hypre_SolveEx.cpp

```cpp
#include "Ifpack2_Preconditioner.hpp"
#include "Ifpack2_Hypre.hpp"
#include "Teuchos_CommandLineProcessor.hpp"
#include "Tpetra_CrsMatrix.hpp"
#include "Tpetra_DefaultPlatform.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 Ifpack2::Preconditioner<Scalar> Preconditioner;
  typedef Tpetra::Map<> Map;

  // Initialize MPI
  Teuchos::oblackholestream blackhole;
  Teuchos::GlobalMPISession mpiSession(&argc,&argv,&blackhole);
  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;
  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.");
  if(cmdp.parse(argc,argv) != Teuchos::CommandLineProcessor::PARSE_SUCCESSFUL) {
    return -1;
  }

  // Create the 2D Laplace operator
  int n = nx*nx;
  RCP<Map> map = rcp(new Map(n,0,comm));
  RCP<CrsMatrix> A = rcp(new CrsMatrix(map,5));
  for(LO i = 0; i<nx; i++) {
    for(LO j = 0; j<nx; j++) {
```

```
 48        GO row = i*nx+j;
 49        if(!map->isNodeGlobalElement(row))
 50          continue;
 51
 52        Array<LO> indices;
 53        Array<Scalar> values;
 54
 55        if(i > 0) {
 56          indices.push_back(row - nx);
 57          values.push_back(-1.0);
 58        }
 59        if(i < nx-1) {
 60          indices.push_back(row + nx);
 61          values.push_back(-1.0);
 62        }
 63        indices.push_back(row);
 64        values.push_back(4.0);
 65        if(j > 0) {
 66          indices.push_back(row-1);
 67          values.push_back(-1.0);
 68        }
 69        if(j < nx-1) {
 70          indices.push_back(row+1);
 71          values.push_back(-1.0);
 72        }
 73        A->insertGlobalValues(row,indices,values);
 74      }
 75    }
 76    A->fillComplete();
 77
 78    // Create the vectors
 79    RCP<MV> X = rcp(new MV(A->getRowMap(),1));
 80    RCP<MV> B = rcp(new MV(A->getRowMap(),1,false));
 81    B->randomize();
 82
 83    // Create the parameters for hypre
 84    RCP<FunctionParameter> functs[10];
 85    functs[0] = rcp(new FunctionParameter(Prec,   &HYPRE_BoomerAMGSetPrintLevel, 1));  // print AMG solution info
 86    functs[1] = rcp(new FunctionParameter(Prec,   &HYPRE_BoomerAMGSetCoarsenType, 6)); // Falgout coarsening
 87    functs[2] = rcp(new FunctionParameter(Prec,   &HYPRE_BoomerAMGSetRelaxType, 6));   // Sym GS/Jacobi hybrid
 88    functs[3] = rcp(new FunctionParameter(Prec,   &HYPRE_BoomerAMGSetNumSweeps, 1));   // Sweeps on each level
 89    functs[4] = rcp(new FunctionParameter(Prec,   &HYPRE_BoomerAMGSetTol, 0.0));       // Conv tolerance zero
 90    functs[5] = rcp(new FunctionParameter(Prec,   &HYPRE_BoomerAMGSetMaxIter, 1));     // Do only one iteration!
 91    functs[6] = rcp(new FunctionParameter(Solver, &HYPRE_PCGSetMaxIter, 1000));        // Maximum iterations
 92    functs[7] = rcp(new FunctionParameter(Solver, &HYPRE_PCGSetTol, tol));             // Convergence tolerance
 93    functs[8] = rcp(new FunctionParameter(Solver, &HYPRE_PCGSetTwoNorm, 1));           // Use the two-norm as the stopping
          criteria
 94    functs[9] = rcp(new FunctionParameter(Solver, &HYPRE_PCGSetPrintLevel, 2));        // Print solve info
 95
 96    // Create the hypre solver and preconditioner
 97    RCP<Preconditioner> prec = rcp(new Ifpack2::Ifpack2_Hypre<Scalar,LO,GO,Node>(A));
 98    ParameterList hypreList;
 99    hypreList.set("SolveOrPrecondition", Solver);
100    hypreList.set("Solver", Ifpack2::Hypre::PCG);
101    hypreList.set("Preconditioner", Ifpack2::Hypre::BoomerAMG);
102    hypreList.set("SetPreconditioner", true);
103    hypreList.set("NumFunctions", 10);
104    hypreList.set<RCP<FunctionParameter>*>("Functions", functs);
105    prec->setParameters(hypreList);
106    prec->compute();
107
108    // Perform solve
109    prec->apply(*B,*X);
110
111    return 0;
112  }
```

Lines 1–5 Include statements

Lines 8–24 Typedefs and using statements to make the code more readable

Lines 26–30 Set up MPI

Lines 32–40 Parse command line arguments. This program allows the user to specify how large the problem should be and how accurately the linear system should be solved.

Lines 42–76 Set up the 2D Laplace operator.

Lines 78–81 Create a random right-hand-side and initialize the solution vector to 0.

Lines 83–94 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 96–106 Create the hypre solver. Line 99 specifies that we will be using a hypre linear solver, and line 100 says it will be the conjugate gradient method. Line 101 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 103 and 104 specify the hypre parameters such as print level, tolerance, and maximum number of iterations.

Lines 108–109 Solve the linear system. apply actually calls the hypre linear solve routine PCG with a BoomerAMG preconditioner, as we specified above.

In the next example (Program 2.4), we will examine how to use hypre preconditioners with Belos solvers.

Program 2.4. Hypre_BelosEx.cpp

```
#include "BelosTpetraAdapter.hpp"
#include "BelosPseudoBlockCGSolMgr.hpp"
#include "Ifpack2_Preconditioner.hpp"
#include "Ifpack2_Hypre.hpp"
#include "Teuchos_CommandLineProcessor.hpp"
#include "Tpetra_CrsMatrix.hpp"
#include "Tpetra_DefaultPlatform.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;

  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;
  typedef Belos::PseudoBlockCGSolMgr<Scalar,MV,OP> Solver;

  // Initialize MPI
  Teuchos::oblackholestream blackhole;
  Teuchos::GlobalMPISession mpiSession(&argc,&argv,&blackhole);
  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 2D Laplace operator
  int n = nx*nx;
  RCP<Map> map = rcp(new Map(n,0,comm));
  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 vectors
  RCP<MV> X = rcp(new MV(A->getRowMap(),1));
  RCP<MV> B = rcp(new MV(A->getRowMap(),1,false));
  B->randomize();

  // 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 hypre 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 Belos linear solver
  RCP<ParameterList> belosList = rcp(new ParameterList());
  belosList->set("Convergence Tolerance", tol);
  if(verbose)
    belosList->set("Verbosity", Belos::Errors + Belos::Warnings + Belos::TimingDetails + Belos::StatusTestDetails);
  else
    belosList->set("Verbosity", Belos::Errors + Belos::Warnings);
  RCP<Solver> newSolver = rcp(new Solver(problem,belosList));

  // Perform solve
```

```
123     newSolver->solve();
124
125     return 0;
126   }
```

Lines 1–86 These lines are not substantially different from the previous example. Again, we set up our operator, solution vector, and right hand side.

Lines 88–95 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 97–105 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 107–111 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 113–120 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).

Lines 122–123 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

TODO.

v1.40

Sandia National Laboratories