SANDIA REPORT

SAND2015-XXXX
Unlimited Release
Printed January 19, 2015

xSDK User Manual

Jeff Johnson, Alicia Klinvex, X. Sherry Li, Barry Smith, Ulrike Meier Yang

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.

Sandia National Laboratories

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

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

Jeff Johnson, Alicia Klinvex, X. Sherry Li, Barry Smith, Ulrike Meier Yang

Abstract

TODO: This needs work.

Some application developers need to be able to use multiple DOE libraries together, such as PETSc, Hypre, SuperLU, Trilinos, and Alquimia. 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 Trilinos 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. 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.

Two of the libraries, PETSc and Trilinos are comprehensive numerical software packages[1] that can solve a large variety of problems with both their own algorithmic implementations and wrappers to other numerical libraries. SuperLU and hypre are specialized libraries that provide sophisticated solvers for a restricted family of problems. SuperLU provides sparse direct linear solvers and ILU preconditioners while hypre provides iterative solvers for linear systems.

The xSDK provides a common system for installing and working with all of these packages so, in some sense, xSDK is meta-package.

Alquimia

Alquimia is an API for providing mature geochemistry and biogeochemistry capabilities to reactive transport codes. It is not a geochemistry solver–it is a library comprising data structures and interfaces that wrap chemistry solvers from well-established codes like PFlotran (`http://www.pflotran.org`) and CrunchFlow (`http://www.csteefel.com/CrunchFlowIntroducti`
allowing developers of newer codes to use these solvers with a single interface. We refer to these chemistry solvers as *chemistry engines*.

Currently, Alquimia supports two chemistry engines: PFlotran and CrunchFlow. There are plans to support more in the future, depending on funding and levels of engagement in the DOE's reactive transport research community.

You can find out more about installing and programming with Alquimia at `https://www.github.com/L`

hypre

The *hypre* software library provides high performance preconditioners and solvers for the solution of large sparse linear systems on massively parallel computers. It was created with the primary goal of providing users with advanced parallel preconditioners. The library

[1]Note that the term "Trilinos package" has a specific meaning that, here we just mean package in the generic sense

features parallel multigrid solvers for both structured and unstructured grid problems. For ease of use, these solvers are accessed from the application code via *hypre*'s conceptual linear system interfaces, which allow a variety of natural problem descriptions and include a structured, a semi-structured interface, and a traditional linear-algebra based interface. The (semi-)structured interfaces are an alternative to the standard matrix-based interface that describes rows, columns, and coefficients of a matrix. Here, instead, matrices are described primarily in terms of stencils and logically structured grids. These interfaces give application users a more natural means for describing their linear systems, and provide access to methods such as structured multigrid solvers, which can take advantage of the additional information beyond just the matrix.

The library is written in C and has a Fortran interface. It supports MPI+OpenMP. More detailed information can be found at [**?**, **?**].

TODO: Fix these citations.

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, preconditioners, nonlinear solvers, and ODE integrators. It also provides a variety of scalable constrained and unconstrained optimization solvers. PETSc utilizes the MPI program model and does not use threads. 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. The library libMesh and the framework MOOSE provide finite element solvers that utilize PETSc.

SuperLU

SuperLU is a general purpose library for the direct solution of large, sparse, nonsymmetric systems of linear equations on high performance machines. The library routines will perform an LU decomposition with partial pivoting and triangular system solves through forward and back substitution. The LU factorization routines can handle non-square matrices but the triangular solves are performed only for square matrices. The matrix columns may be preordered (before factorization) either through library or user supplied routines. This preordering for sparsity is completely separate from the factorization. Working precision iterative refinement subroutines are provided for improved backward stability. Routines are also provided to equilibrate the system, estimate the condition number, calculate the relative backward error, and estimate error bounds for the refined solutions.

There are three separate versions of this code: SuperLU (for sequential machines), SuperLU_MT (for shared memory parallel machines with using OpenMP or Pthread), and

SuperLU_DIST (for distributed memory machines using MPI).

The library is written in C, with Fortran interface. SuperLU_DIST supports MPI+X, where X can be CUDA, OpenMP, or both.

Trilinos

TODO: This needs work. Maybe Jim or Mike should have a look at it.

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.

Chapter 1

xSDK Installation

The easiest way to install the xSDK libaries is the following two step process:

1. curl `https://raw.githubusercontent.com/xsdk-project/installxSDK/master/installxSDK.sh` ¿ installxSDK.sh
2. Run the PETSc configuration script.
3. sh ./installxSDK –prefix=”installation directory” [other configure options]

This script will download and install PETSc, hypre, SuperLU_dist, Trilinos, as well as commonly needed dependent packages.

Many useful options for the script can be found in the README at https://github.com/xsdk-project/installxSDK

Running the script with the option –help will produce a listing of all possible options.

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 CONFIGURE with GIVEN OPTIONS    (see
-------------------------------------------------------------------------------
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
*******************************************************************************
```

Should any other problem occur during the installation process the file xsdk/petsc/configure.log contains all information about the configuation and build process. You should email this file to petsc-maint@mcs.anl.gov for help.

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, 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 variables

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.

Lines 66–67 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 69–72 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 74–77 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 79–82 Solve the eigenvalue problem. After it is solved, we may grab the eigenvalues and eigenvectors via getSolution().

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 cube, and b is a random vector.

Program 2.2. PETSc_Amesos2Ex.cpp

```
#include <Teuchos_ScalarTraits.hpp>
#include <Teuchos_RCP.hpp>
#include <Tpetra_Map.hpp>
#include <Tpetra_MultiVector.hpp>
#include <Tpetra_CrsMatrix.hpp>
#include <Tpetra_Vector.hpp>
#include "Amesos2.hpp"
#include "Amesos2_Version.hpp"
#include "Amesos2_KLU2.hpp"

#include "petscksp.h"
#include "Tpetra_PETScAIJMatrix.hpp"

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

[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.

```
17    using Teuchos::rcp;
18    using Teuchos::ArrayView;
19
20    typedef Tpetra::PETScAIJMatrix<>             PETScAIJMatrix;
21    typedef PETScAIJMatrix::scalar_type          Scalar;
22    typedef PETScAIJMatrix::local_ordinal_type   LO;
23    typedef PETScAIJMatrix::global_ordinal_type  GO;
24    typedef PETScAIJMatrix::node_type            Node;
25    typedef Tpetra::CrsMatrix<Scalar,LO,GO>      CrsMatrix;
26    typedef Tpetra::Vector<Scalar,LO,GO>         Vector;
27    typedef Tpetra::Map<LO,GO>                   Map;
28    typedef Tpetra::Operator<Scalar,LO,GO>       OP;
29    typedef Tpetra::MultiVector<Scalar,LO,GO>    MV;
30    typedef Amesos2::Solver<CrsMatrix,MV>        Solver;
31
32    Vec             x,b;
33    Mat             A;
34    PetscRandom     rctx;
35    PetscInt        i,j,Ii,J,Istart,Iend;
36    PetscInt        m = 50,n = 50;
37    PetscErrorCode  ierr;
38    PetscScalar     v;
39    PetscInt rank=0;
40    MPI_Comm comm;
41
42    PetscInitialize(&argc,&args,NULL,NULL);
43    ierr = PetscOptionsGetInt(PETSC_NULL,"-mx",&m,PETSC_NULL);CHKERRQ(ierr);
44    ierr = PetscOptionsGetInt(PETSC_NULL,"-my",&n,PETSC_NULL);CHKERRQ(ierr);
45
46    ierr = MatCreate(PETSC_COMM_WORLD,&A);CHKERRQ(ierr);
47    ierr = MatSetSizes(A,PETSC_DECIDE,PETSC_DECIDE,m*n,m*n);CHKERRQ(ierr);
48    ierr = MatSetType(A, MATAIJ);CHKERRQ(ierr);
49    ierr = MatMPIAIJSetPreallocation(A,5,PETSC_NULL,5,PETSC_NULL);CHKERRQ(ierr);
50    ierr = MatSetUp(A);CHKERRQ(ierr);
51    PetscObjectGetComm( (PetscObject)A, &comm);
52    ierr = MPI_Comm_rank(comm,&rank);CHKERRQ(ierr);
53
54    ierr = MatGetOwnershipRange(A,&Istart,&Iend);CHKERRQ(ierr);
55    for (Ii=Istart; Ii<Iend; Ii++) {
56      v = -1.0; i = Ii/n; j = Ii - i*n;
57      if (i>0)   {J = Ii - n; ierr = MatSetValues(A,1,&Ii,1,&J,&v,INSERT_VALUES);CHKERRQ(ierr);}
58      if (i<m-1) {J = Ii + n; ierr = MatSetValues(A,1,&Ii,1,&J,&v,INSERT_VALUES);CHKERRQ(ierr);}
59      if (j>0)   {J = Ii - 1; ierr = MatSetValues(A,1,&Ii,1,&J,&v,INSERT_VALUES);CHKERRQ(ierr);}
60      if (j<n-1) {J = Ii + 1; ierr = MatSetValues(A,1,&Ii,1,&J,&v,INSERT_VALUES);CHKERRQ(ierr);}
61      v = 4.0; ierr = MatSetValues(A,1,&Ii,1,&Ii,&v,INSERT_VALUES);CHKERRQ(ierr);
62    }
63
64    ierr = MatAssemblyBegin(A,MAT_FINAL_ASSEMBLY);CHKERRQ(ierr);
65    ierr = MatAssemblyEnd(A,MAT_FINAL_ASSEMBLY);CHKERRQ(ierr);
66
67    ierr = VecCreate(PETSC_COMM_WORLD,&x);CHKERRQ(ierr);
68    ierr = VecSetSizes(x,PETSC_DECIDE,m*n);CHKERRQ(ierr);
69    ierr = VecDuplicate(x,&b);CHKERRQ(ierr);
70    ierr = PetscRandomCreate(PETSC_COMM_WORLD,&rctx);CHKERRQ(ierr);
71    ierr = PetscRandomSetFromOptions(rctx);CHKERRQ(ierr);
72    ierr = VecSetRandom(b,rctx);CHKERRQ(ierr);
73    ierr = PetscRandomDestroy(&rctx);CHKERRQ(ierr);
74
75    RCP<CrsMatrix> tpetraA = xSDKTrilinos::deepCopyPETScAIJMatrixToTpetraCrsMatrix<Scalar,LO,GO,Node>(A);
76    RCP<Vector> tpetraX = xSDKTrilinos::deepCopyPETScVecToTpetraVector<Scalar,LO,GO,Node>(x);
77    RCP<Vector> tpetraB = xSDKTrilinos::deepCopyPETScVecToTpetraVector<Scalar,LO,GO,Node>(b);
78
79    RCP<Solver> solver = Amesos2::create<CrsMatrix,MV>("KLU2", tpetraA, tpetraX, tpetraB);
80    solver->symbolicFactorization();
81    solver->numericFactorization();
82    solver->solve();
83
84    ierr = PetscFinalize(); CHKERRQ(ierr);
85    return 0;
86  }
```

Lines 1–65 These lines are very similar to the previous example. We set up convenient typedefs, read in the command line arguments with PETSc (in this case, the x and y dimensions of the grid), and set up the Poisson2D matrix.

Lines 67–73 Create the initial guess for x and the random RHS b.

Lines 75–77 Copy the PETSc Mat and Vecs to Tpetra::CrsMatrix and Tpetra::Vector, so that we can use them with the Amesos2 linear solvers.

Lines 79–82 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.
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

[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

```
#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;
  RCP<Prec> M = factory.create("RELAXATION", A.getConst());
  ParameterList ifpackParams;
  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. 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

```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;
  }
```

```
 41
 42   // Create the 2D Laplace operator
 43   int n = nx*nx;
 44   RCP<Map> map = rcp(new Map(n,0,comm));
 45   RCP<CrsMatrix> A = rcp(new CrsMatrix(map,5));
 46   for(LO i = 0; i<nx; i++) {
 47     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. 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 "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());
```

```
115   belosList->set("Convergence Tolerance", tol);
116   if(verbose)
117     belosList->set("Verbosity", Belos::Errors + Belos::Warnings + Belos::TimingDetails + Belos::StatusTestDetails);
118   else
119     belosList->set("Verbosity", Belos::Errors + Belos::Warnings);
120   RCP<Solver> newSolver = rcp(new Solver(problem,belosList));
121 
122   // Perform solve
123   newSolver->solve();
124 
125   return 0;
126 }
```

Lines 1–86 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 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

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

```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.

PETSc to Other Packages

PETSc uses run-time binding of data structures and solver algorithms which means that in most causes you do not need to change your simulation code to switch between solvers in hypre, SuperLU, and Trilinos. You use the PETSc options database (for example command line arguments) to select your solvers; it is also possible to hardware particular solvers into the source code but we do not recommend this approach. The solvers in the other xSDK packages are interfaced to PETSc in the PETSc abstract PC preconditioners interface[5]. To select a particular preconditioner one uses

```
-pc_type typename [-pc_* other potential options related to the solver]
```

PETSc utilizing SuperLU

PETSc can use both SuperLU and SuperLU_Dist solvers. Since the SuperLU_Dist solvers are parallel and in general are faster and require less memory than SuperLU we will only discuss the details of utilizing SuperLU_Dist. It is utilized with

```
-pc_type lu -pc_factor_mat_solver_package superlu_dist [-mat_superlu_dist_*
SuperLU_Dist options]
```

All the options will be printed if you run with `-pc_type lu -pc_factor_mat_solver_package superlu_dist -help | grep superlu_dist`. We also display many of the options in Table 2.2.

[5]This same interface is used for direct solvers

Option	Arguments	Default	Description
-mat_superlu_dist_statprint	Yes/No	No	Prints statistics on factorization
TODO: Someone needs to with -help and add the rest.			

Table 2.2. PETSc SuperLU_Dist Options

Option	Arguments	Default	Description
-pc_hypre_boomeramg_max_levels	1	Problem dependent	Sets maximum number of levels
TODO: Someone needs to with -help and add the rest.			

Table 2.3. PETSc hypre BoomerAMG Options

PETSc utilizing hypre

PETSc utilizes several of the solvers/preconditioners in hypre via the command

```
-pc_type hypre -pc_hypre_type [boomeramg (default),pilut,parasails,ams,ads]
```

Since BoomerAMG is the crown jewel of hypre we recommend its use whenever possible and do not discuss here the use of the other solvers. Running with `-pc_type hypre -help | grep hypre` for example will show the options for the `pilut` preconditioner. Many of the options are described in Table 2.3.

PETSc utilizing Trilinos

PETSc can utilize the ML solver in Trilinos.

```
-pc_type ml [-pc_ml_* options for ML]
```

Running with `-pc_type ml -help | grep pc_ml` will show the available options, many of which are detailed in Table 2.4.

TODO: Is this the only interface PETSc has to Trilinos?

Option	Arguments	Default	Description
-pc_ml_maxNlevels	1	10	Sets maximum number of levels
TODO: Someone needs to with -help and add the rest.			

Table 2.4. PETSc ML Options

v1.40

Sandia National Laboratories