SAND2007-7130P

Amesos
Sparse Direct Solver Package

Tim Davis, Mike Heroux, Rob Hoekstra, Marzio Sala, Ken Stanley, Heidi Thornquist, Jim Willenbring

Trilinos Users Group
November 6th, 2007

LOCKHEED MARTIN

Sandia National Laboratories

What's new in Amesos

- KLU / BTF version update
- Paraklete improvements
- Interface – Timing details

Currently Available Solver Interfaces

- **KLU:** Native. Serial unsymmetric [Davis]
- **SuperLU:** Serial unsymmetric [Li et al.]; v3.0
- **UMFPACK:** Serial unsymmetric [Davis]; v4.4
- **LAPACK:** Serial dense unsymmetric [Dongarra et al.]
- **Paraklete:** Native. Parallel unsymmetric [Davis]
- **SuperLUdist:** Parallel unsymmetric [Li et al.]; v2.0
- **MUMPS:** Parallel unsymmetric [Amestoy et al.]; v4.6.2
- **ScaLAPACK:** Parallel dense unsymmetric [Dongarra et al.]
- **DSCPACK:** Parallel Symmetric [Ragavan]; v1.0
- **TAUCS:** Parallel Symmetric [Toledo et al.]; v2.2

Current Amesos Factory Interface

```
// Create an Epetra_LinearProblem
Epetra_LinearProblem Problem(Matrix, LHS, RHS);

// Create a solver object.
Teuchos::RCP<Amesos_BaseSolver> Solver;

// Create the solver factory.
Amesos Factory;

// Specifiy the solver. SolverType can be one
// of the following values:
// - "Lapack"
// - "Klu"
// - "Umfpack"
// - "Superlu"
// - "Scalapack"
// - "Superludist"
// - "Mumps"
// - "Taucs"
// - "Dscpack"
std::string SolverType

// Create the solver using the factory.
Solver = Factory.Create(SolverType, Problem);
```

```
// Set solver parameters
Teuchos::ParameterList List;
List.set("ParameterName", ParameterValue);
Solver->SetParameters(List);

// Perform symbolic factorization
// (only need Matrix graph, not values)
Solver->SymbolicFactorization();

// Perform numeric factorization
// (Matrix values can change here)
Solver->NumericFactorization();

// Perform solve
// (LHS and RHS of Problem can change here)
Solver->Solve();
```

KLU / BTF Version Update

- KLU / BTF 1.0 released Spring 2007
  - first official release
- Integrated into Amesos for Trilinos 8.0

```
// Create an Epetra_LinearProblem
Epetra_LinearProblem Problem(Matrix, LHS, RHS);

// Create a solver object.
Teuchos::RCP<Amesos_BaseSolver> Solver;

// Create the solver factory.
Amesos Factory;

// Create the solver using the factory.
Solver = Factory.Create("Klu", Problem);

// Perform symbolic factorization
Solver->SymbolicFactorization();

// Perform numeric factorization
Solver->NumericFactorization();

// Perform solve
Solver->Solve();
```

Paraklete Improvements

- Native distributed memory sparse solver
  - Tim Davis
- Memory leak fixed
  - Enabled the solution of sequences of linear systems
- Block triangular form (BTF) permutation
  - Serial symbolic analysis
  - Integration into Amesos soon …

Interface – Timing Details

- **Amesos_Time class**
  - rewritten for more efficiency
  - allows same timing output
    - `ParamList.set("Print Timing", true);`

```
-------------------------------------------------------------------------------
Amesos_Klu : Time to convert matrix to Klu format = 2.3e-05 (s)
Amesos_Klu : Time to redistribute matrix = 1.6e-05 (s)
Amesos_Klu : Time to redistribute vectors = 4e-06 (s)
Amesos_Klu : Number of symbolic factorizations = 1
Amesos_Klu : Time for sym fact = 0.00015 (s), avg = 0.00015 (s)
Amesos_Klu : Number of numeric factorizations = 1
Amesos_Klu : Time for num fact = 8.8e-05 (s), avg = 8.8e-05 (s)
Amesos_Klu : Number of solve phases = 1
Amesos_Klu : Time for solve = 1.7e-05 (s), avg = 1.7e-05 (s)
Amesos_Klu : Total time spent in Amesos = 0.000255 (s)
Amesos_Klu : Total time spent in the Amesos interface = 7.8e-05 (s)
Amesos_Klu : (the above time does not include KLU time)
Amesos_Klu : Amesos interface time / total time = 0.305882
-------------------------------------------------------------------------------
```

  - enables timings to be loaded into a Teuchos::ParameterList

Interface - Timing Details

```
Teuchos::ParameterList TimingsList;
Solver->GetTiming( TimingsList );

// you can find out how much time was spent in ...
double sfact_time, nfact_time, solve_time;
double mtx_conv_time, mtx_redist_time, vec_redist_time;

// 1) The symbolic factorization
//    (parameter doesn't always exist)
sfact_time = TimingsList.get( "Total symbolic factorization time", 0.0 );

// 2) The numeric factorization
//    (always exists if NumericFactorization() is called)
nfact_time = Teuchos::getParameter<double>( TimingsList, "Total numeric factorization time" );

// 3) Solving the linear system
//    (always exists if Solve() is called)
solve_time = Teuchos::getParameter<double>( TimingsList, "Total solve time" );

// 4) Converting the matrix to the accepted format for the solver
//    (always exists if SymbolicFactorization() is called)
mtx_conv_time = Teuchos::getParameter<double>( TimingsList, "Total solve time" );

// 5) Redistributing the matrix for each solve to the accepted format for the solver
mtx_redist_time = TimingsList.get( "Total matrix redistribution time", 0.0 );

// 6) Redistributing the vector for each solve to the accepted format for the solver
vec_redist_time = TimingsList.get( "Total vector redistribution time", 0.0 );
```