

Scalable preconditioned eigenvalue solver in Hypre

Andrew V. Knyazev (the speaker) and Merico Argentati

*Department of Mathematics and
Center for Computational Mathematics
University of Colorado at Denver*

Preconditioning, Napa, October 28, 2003

Supported by the Lawrence Livermore National Laboratory (LLNL)

Center for Computational Mathematics, University of Colorado at Denver

Abstract

$$B^{-1}(A - \text{?})u = 0$$

We develop a scalable preconditioned eigenvalue solver for partial eigenvalue problems for large symmetric matrices on massively parallel computers, using the multigrid technology and incomplete factorization preconditioning of the LLNL HYPRE software. The solver implements the locally optimal block preconditioned conjugate gradient (LOBPCG). The LOBPCG solver computes one or more of the smallest eigenvalues and the corresponding eigenvectors of a symmetric matrix using preconditioning directly, without using the shift-and-invert scheme and inner-outer iterations.

Acknowledgement

Supported by Lawrence Livermore National Laboratory, Center for Applied Scientific Computing (LLNL-CASC).

We would like to thank Rob Falgout, Charles Tong, Panayot Vassilevski and other members of the Hypre Scalable Linear Solvers project team for their help.

Implementation of LOBPCG Eigensolver Using Hypre

1. Background concerning algorithm
2. Hypre software library
3. LOBPCG Hypre implementation strategy
4. Compiling and Testing
5. Initial Performance Results
6. Conclusions

Background

- LOBPCG - Locally Optimal Block Preconditioned Conjugate Gradient Method
- Authors of code for Hypre – Andrew Knyazev & Merico Argentati both from the University of Colorado at Denver
- Algorithm is described in: A. V. Knyazev, [Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method](#). SIAM Journal on Scientific Computing 23 (2001), no. 2, pp. 517-541.

LOBPCG Algorithm

- LOBPCG solver finds the extreme eigenvalues of a symmetric (positive definite) matrix using preconditioning directly
- For computing only the smallest eigenpair, the algorithm LOPCG (Block size = 1) implements a local optimization of a 3-term recurrence
$$x^{n+1} = \operatorname{argmax}(x, Ax)/(x, x), x \in \operatorname{span}\{x^n, x^{n-1}, B^{-1}(Ax^n - \lambda(x^n)x^n)\}$$
- For finding m smallest eigenpairs of A , the Rayleigh-Ritz method on a $3m$ -dimensional trial subspace, is used during each iteration for the local optimization. Cluster robust, does not miss multiple eigenvalues!
- The algorithm is **matrix free** since the multiplication of a vector by the matrix A and an application of the preconditioner B^{-1} to a vector are needed only as functions. Hessian-free nonquadratic CG.

Previously Developed LOBPCG Software

- MATLAB
- C-language using LAPACK
- Initial PETSc version
- LOBPCG implementations by other groups: MATLAB, C, Fortran

Hypre (High Performance Preconditioners)

- Hypre is a software library for solving large, sparse linear systems on massively parallel computers
- The primary goal is to provide users with advanced parallel preconditioners
- The Hypre libraries are designed to provide robustness, ease of use, flexibility and interoperability

LOBPCG Hypre Implementation

- C-language
- Hypre libraries and LAPACK/BLAS libraries
- Implementation is based on Hypre Linear Algebraic IJ Interface for applications with sparse linear systems
- User interface to solver is “Hypre style”
- User provided functions for MATVEC multiply and preconditioner/solve
- LOBPCG Hypre implementation utilizes Hypre parallel vector manipulation routines and MGS for orthonormalization

LOBPCG Hypre Implementation (cont.)

- Our test driver `IJ_eigen_solver.c` (`ij_es.c`), is a modified version of `IJ_linear_solvers.c` (`ij_ls.c`), which retains much of its functionality/capability
- Test driver allows for input of Matrix Market files and internally generated matrices (Laplacians of several different types and sizes)
- Shell script `IJ_eigen_solver.sh` (`ij_es.c`) provides basic suite of tests
- LOBPCG Hypre is developed for Hypre 1.6.0. The most recent Hypre revision 1.8.1b is supported in LOBPCG 042403 with a patch
- LOBPCG Hypre based code has been included in the current **1.8.1b Release** of the **Hypre**

LOBPCG Software Implemented Using Hypr

IJ_eigen_solver.c (ij_es.c)

lobpcg.c

lobpcg_matrix.c

lobpcg_utilities.c

lobpcg.h

Hypr Include Files

Compiling and Testing

- Make files for various hardware configurations and compilers are provided
- A test program `IJ_eigen_solver.sh` (`ij_es.sh`), similar to `IJ_linear_solvers.sh` (`ij_ls.sh`), has been developed
- LOBPCG code has been compiled and tested on the CU Denver cluster using `scali` and `mpich` libraries and using `gcc`, `pgcc` and `mpicc` compilers

Compiling and Testing (cont.)

- The beowulf cluster at CU Denver includes:
 - 36 nodes, 2 processors each
 - each node: 2 PIII 933MHz, 2GB memory
 - linux Redhat 7.2
 - SCI Dolpin interconnect, management interconnect
- Lobpcg has been compiled and tested on a subset of the OCF Production Systems at LLNL, including ASCI blue, M&IC tera, gps and lx

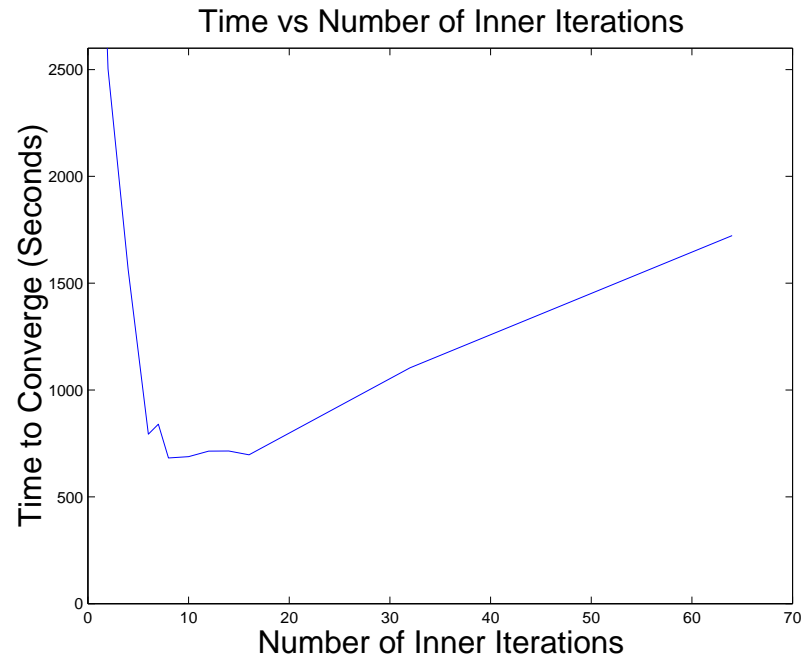
Compiling and Testing (cont.)

- Testing has been done using a variety of internally generated Laplacians and several matrix market files that were used as input
- Quality and accuracy of code, using multiple processors, seems to be quite good based in this limited testing
- Preconditioning is implemented through calls to Hypre preconditioned PCG linear solvers. In our tests, only a few (2-10) inner iterations typically provide the best performance and increasing the number of the inner iterations does not improve the final convergence

Hypre Preconditioners Tested with LOBPCG

- AMG-PCG: algebraic multigrid
- DS-PCG: diagonal scaling
- ParaSails-PCG: approximate inverse of A is generated by attempting to minimize $\|I - AM\|_F$
- Schwarz-PCG: additive Schwarz
- Euclid-PCG: incomplete LU

LOBPCG Performance vs Preconditioner Iterations



7-Point 3-D Laplacian, 8,000,000 x 8,000,000 matrix, 55,760,000 nz,
Preconditioner/Solve: Schwarz-PCG, 10 Processors.

Center for Computational Mathematics, University of Colorado at Denver

Convergence rates for different preconditioners

7-Point 3-D Laplacian with $n = 100$, $nz = 6,940,000$, the blocksize equal to 10 and we vary the type of preconditioner. We run the problem on 10 processors using LLNL ASCI blue using Hypre version Hypre-1.6.0.

We set the maximum number of inner iterations equal to 3 on Figure 1, but DS-PCG (16), ParaSails-PCG (14), Schwarz-PCG (8), Euclid-PCG (6) and AMG-PCG (1) on Figure 2 .

Scalable preconditioned eigensolver in Hypre 18

Andrew Knyazev and Merico Argentati

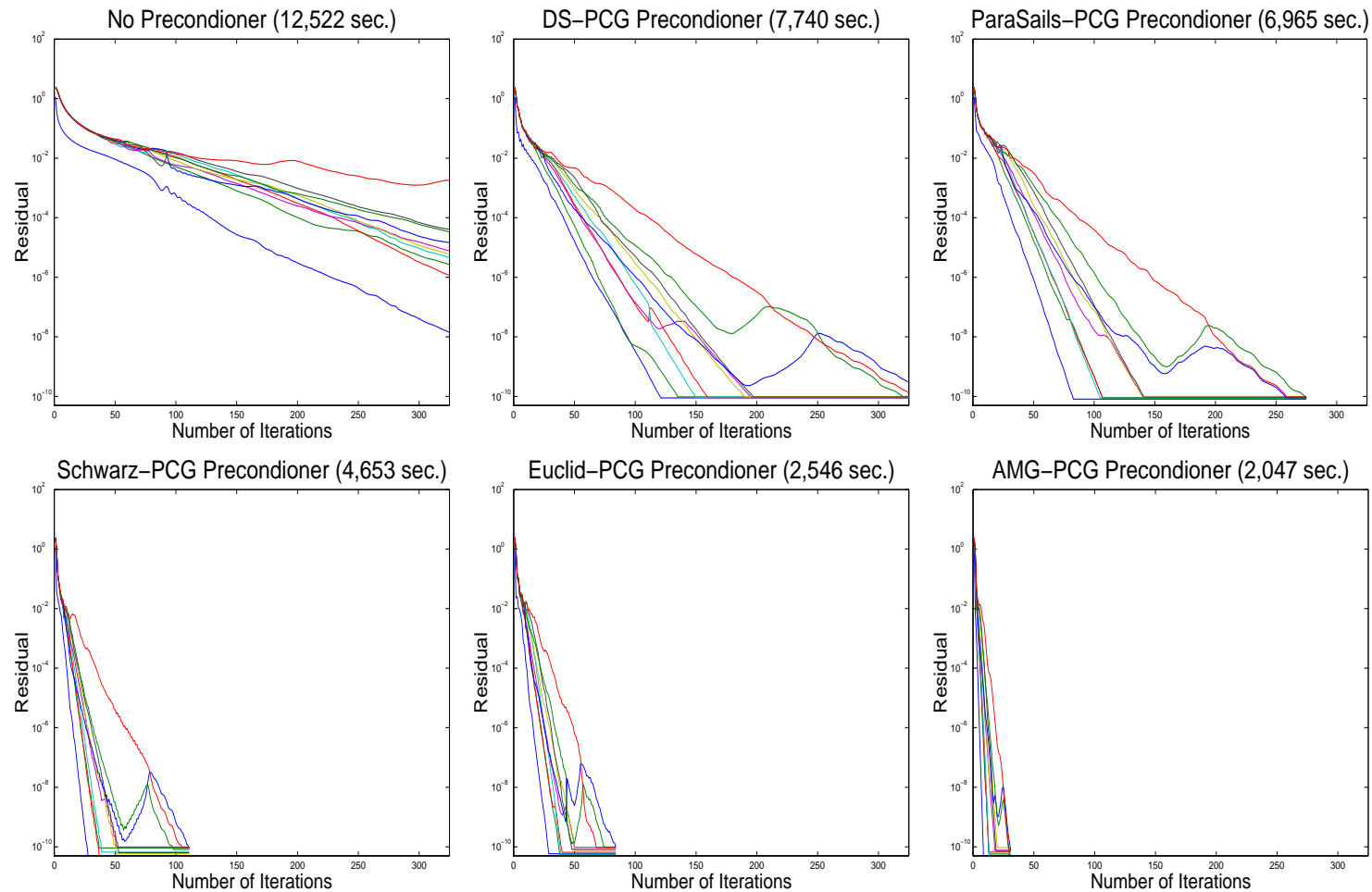


FIGURE 1: *Convergence for different preconditioners with 3 inner iterations*

Scalable preconditioned eigensolver in Hypre 19

Andrew Knyazev and Merico Argentati

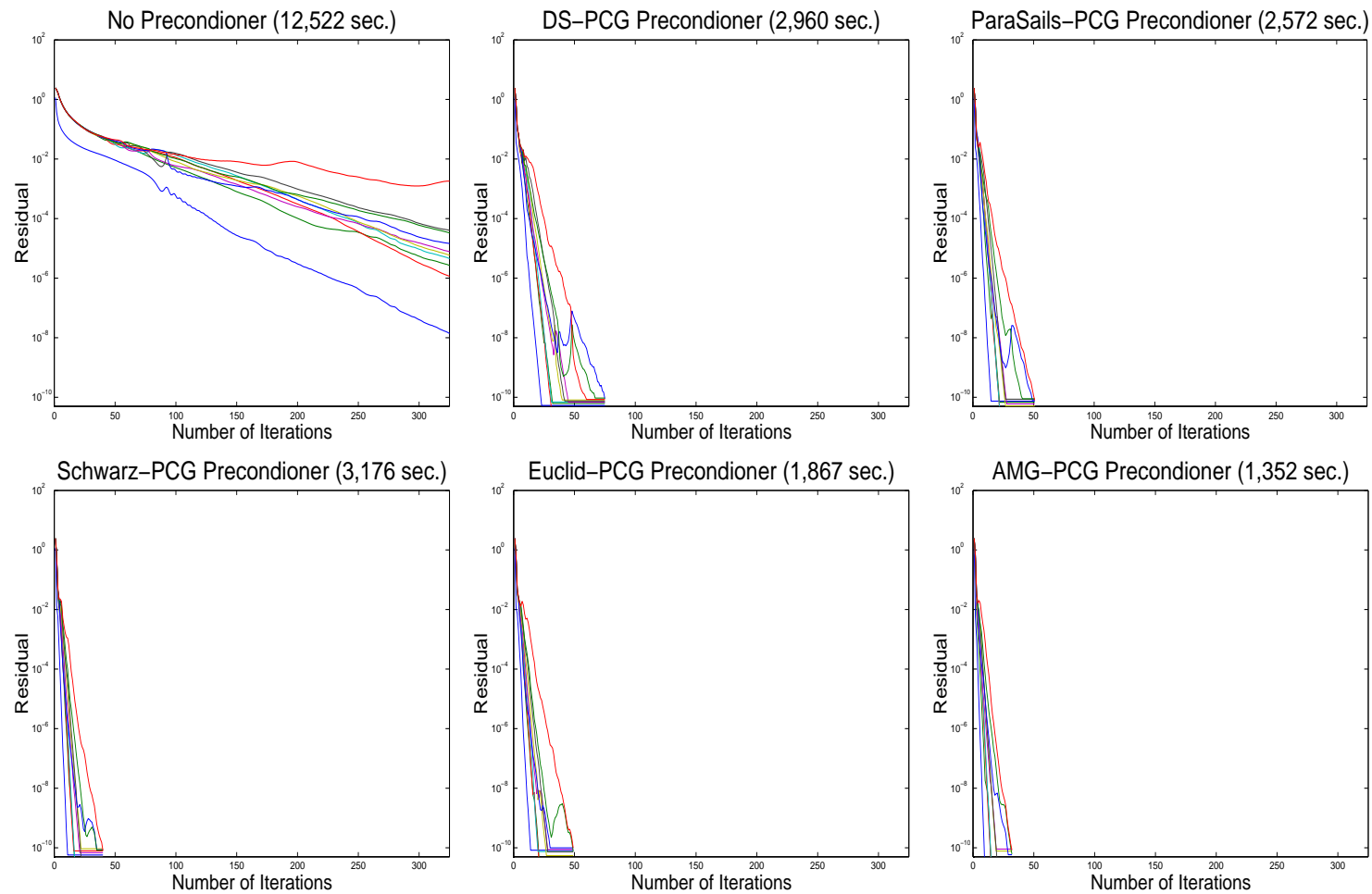


FIGURE 2: *Convergence for different preconditioners with best inner iterations*

Scalability on IBM ASCI blue at LLNL

Nproc	n	Prec. setup (Seconds)	Apply Prec. (Seconds)	Lin. Alg. (Seconds)	Itr	Apply/Itr (Seconds)	Alg/Itr (Seconds)
4	100	342	242.8	14.5	9	26.9	1.61
8	126	352	287.2	17.6	10	28.7	1.76
16	159	349	401.9	28.6	13	30.9	2.20
32	200	456	517.8	49.4	16	32.6	3.09
64	252	377	673.4	84.5	21	32.1	4.03

TABLE 1: Scalability Data for 3–D Laplacian

7–Point 3–D Laplacian, Block size: 1, LOBPCG tolerance: 10^{-6} , Inner (pcg) iterations: 10, Preconditioner/Solve: Schwarz-PCG.

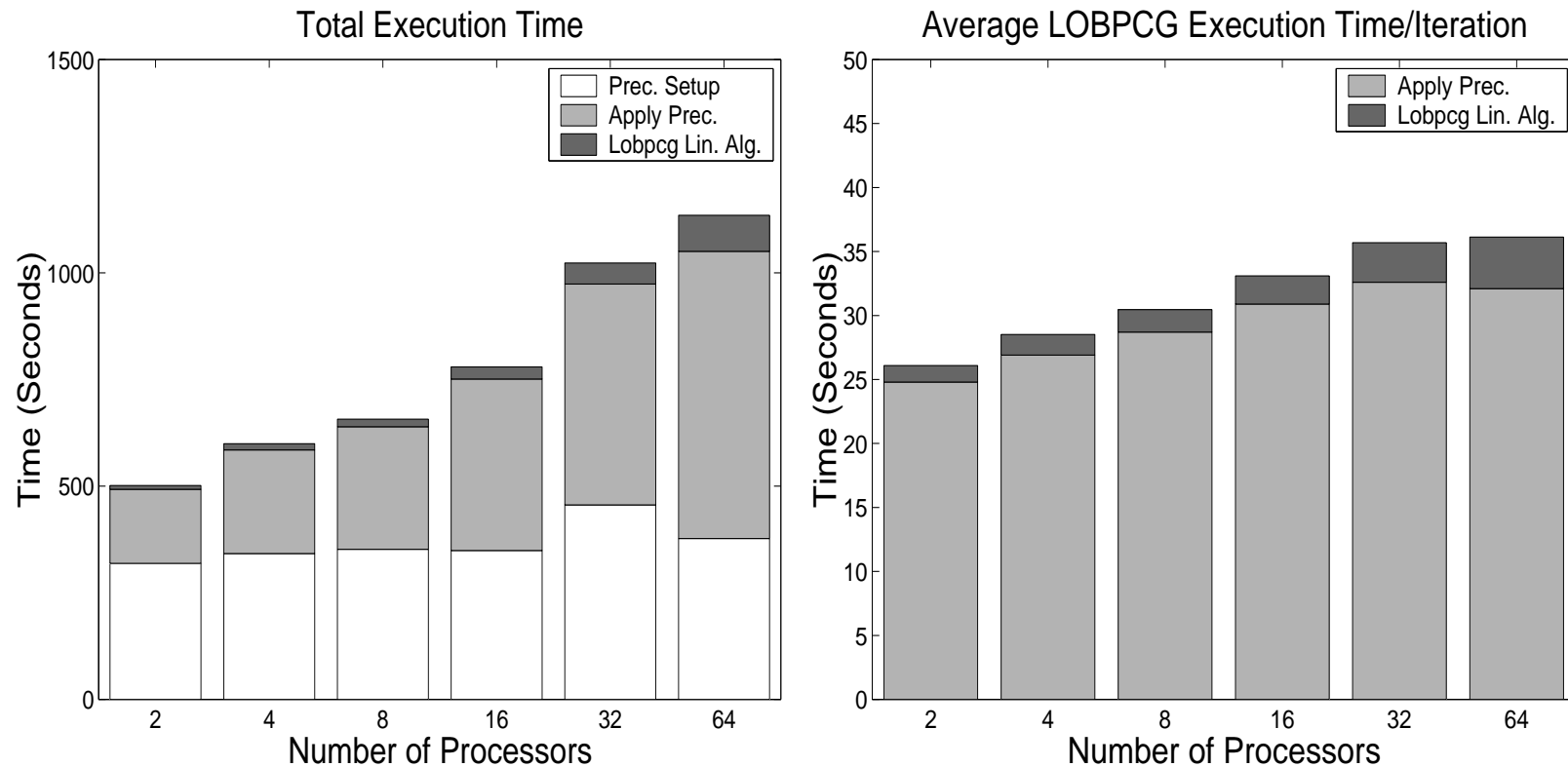


FIGURE 3: *Execution Time as Problem Size Increases for 3-D Laplacian*

Conclusions

- Implementation illustrates that the LOBPCG matrix free algorithm can be implemented using parallel libraries
- User interface routines
 - provide ease of use for a variety of users
 - have been developed using the Hypr standard interface
 - provide flexible capability for the user to provide MATVEC multiply and preconditioned solver functions, which leverage the power of Hypr preconditioners
- Initial scalability looks promising, but more testing is needed by other users on larger problems