

Preconditioned eigenvalue solver in Hypre and PETSc

**Ilya Lashuk (speaker), Merico Argentati,
Evgenii Ovtchinnikov, Andrew Knyazev**


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

Twelfth Copper Mountain Conference on Multigrid Methods
April 3 - April 8, 2005

*Supported by the Lawrence Livermore National Laboratory and
the National Science Foundation*

Center for Computational Mathematics, University of Colorado at Denver

Abstract


$$T(A - \Omega B)x = 0$$

We develop a scalable preconditioned eigenvalue solver for partial eigenvalue problems for large symmetric matrices on massively parallel computers, using the multigrid technology and domain decomposition preconditioning of the HYPRE and PETSc 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 generalized definite eigenproblem using preconditioning directly, without using the shift-and-invert scheme and inner-outer iterations.

Center for Computational Mathematics, University of Colorado at Denver

Acknowledgements

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

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.

We are indebted to Jose E. Roman, a member of SLEPc team, who has recently written PETSC interface to our Hypre LOBPCG solver (this interface will appear in the next public release of SLEPc) and thus motivated us to write the PETSc version of LOBPCG.

Implementation of LOBPCG Eigensolver

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

Background

- LOBPCG - Locally Optimal Block Preconditioned Conjugate Gradient Method
- Authors of code for Hypre and PETSc – Merico Argentati, Andrew Knyazev, Ilya Lashuk, Evgenii Ovtchinnikov.
- 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 smallest eigenpairs of a symmetric generalized definite eigenvalue problem using preconditioning directly.
- For computing only the smallest eigenpair, the algorithm LOPCG (**Block size = 1**) implements a local optimization of a 3-term recurrence.
- For finding m smallest eigenpairs 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 matrices A , B and an application of the preconditioner T to a vector are needed only as functions.

What is LOBPCG?

The method combines *robustness and simplicity* of the steepest descent method with a *three-term recurrence* formula:

$$x^{(i+1)} = w^{(i)} + \tau^{(i)} x^{(i)} + \gamma^{(i)} x^{(i-1)},$$

$$w^{(i)} = T(Ax^{(i)} - \lambda^{(i)} Bx^{(i)}), \quad \lambda^{(i)} = \lambda(x^{(i)}) = (x^{(i)}, Ax^{(i)}) / (Bx^{(i)}, x^{(i)})$$

with properly chosen scalar iteration parameters $\tau^{(i)}$ and $\gamma^{(i)}$. The easiest and most efficient choice of parameters is based on an idea of *local optimality* Knyazev 1986, namely, select $\tau^{(i)}$ and $\gamma^{(i)}$ that minimize the Rayleigh quotient $\lambda(x^{(i+1)})$ by using the *Rayleigh–Ritz method*.

$$\boxed{\text{Three-term recurrence}} + \boxed{\text{Rayleigh–Ritz method}} = \boxed{\text{Locally Optimal Conjugate Gradient Method}}$$

Currently available LOBPCG software from our group:

- MATLAB version (stable, publicly available)
- Hypre version (beta, publicly available)
- PETSc version (alpha)

LOBPCG implementations by other groups:

- MATLAB (by Peter. Arbenz)
- C++ (by R. Lehoucq and U. Hetmaniuk, part of Anasazi Trilinos)
- Fortran 77 (by Randolph Bank, part of PLTMG)
- Fortran 90 (by G. Zèrah, part of Abinit, complex Hermitian matrices)
- PETSc interface to Hypre LOBPCG (by Jose Roman, part of SLEPc)

Portable, Extensible Toolkit for Scientific Computation (PETSc) and High Performance Preconditioners (Hypre)

- Software libraries for solving large systems on massively parallel computers
- The libraries are designed to provide robustness, ease of use, flexibility and interoperability.
- The primary goal of Hypre is to provide users with advanced high-quality parallel preconditioners for linear systems.
- The primary goal of PETSc is to facilitate the integration of independently developed application modules with strict attention to component interoperability.

LOBPCG Hypre/PETSc Implementation

- Abstract implementation in C-language
- Hypre/PETSc and LAPACK libraries
- User-provided functions for matrix-vector multiply and preconditioner
- LOBPCG implementation utilizes Hypre/PETSc parallel vector manipulation routines.

Advantages of native Hypre/PETSc implementations of LOBPCG:

- A native Hypre LOBPCG version efficiently takes advantage of powerful Hypre algebraic and geometric multigrid preconditioners
- A native PETSc LOBPCG version gives the PETSc users community an easy access to a customizable code of the high quality modern preconditioned eigensolver

LOBPCG Software Implemented Using Hypre/PETSc

PETSc driver for LOBPCG

Hypre driver for LOBPCG

Interface PETSc-LOBPCG

Interface Hypre-LOBPCG

PETSc libraries

Abstract LOBPCG in C

Hypre libraries

Domain Decomposition and Multilevel Preconditioners Tested with LOBPCG

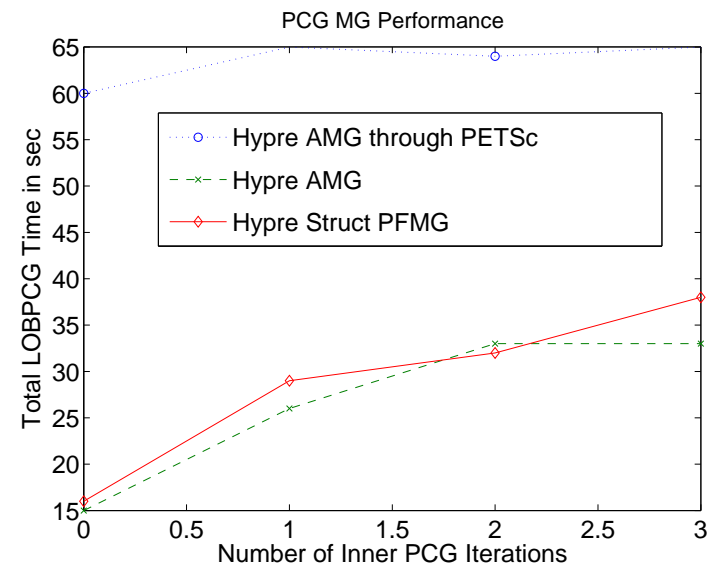
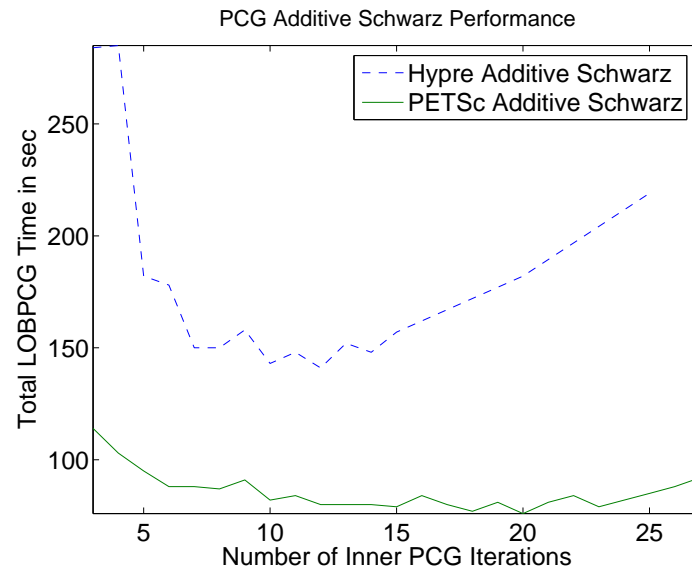
Hypre Implementation:

- PFMG-PCG: geometric multigrid called directly or through PCG
- AMG-PCG: algebraic multigrid called directly or through PCG
- Schwarz-PCG: additive Schwarz called directly or through PCG

PETSc Implementation:

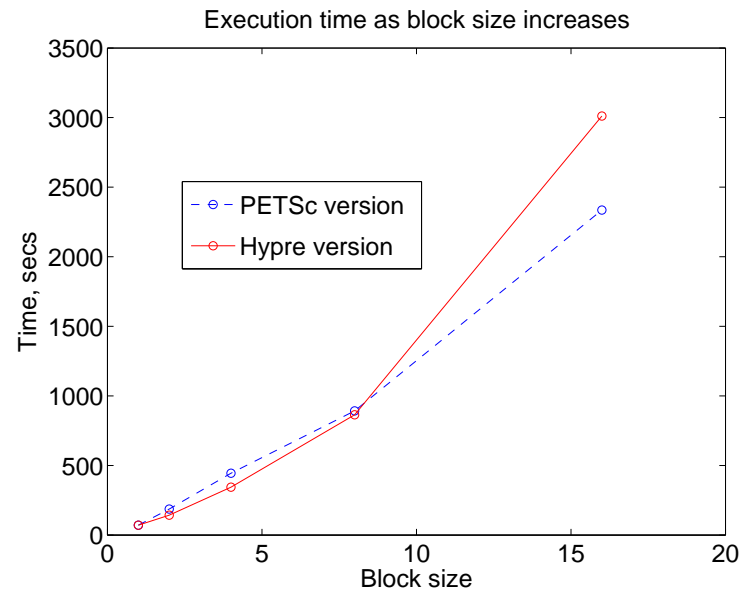
- Additive Schwarz called directly or through PCG
- Algebraic preconditioners from the Hypre package

LOBPCG Performance vs Preconditioner Iterations



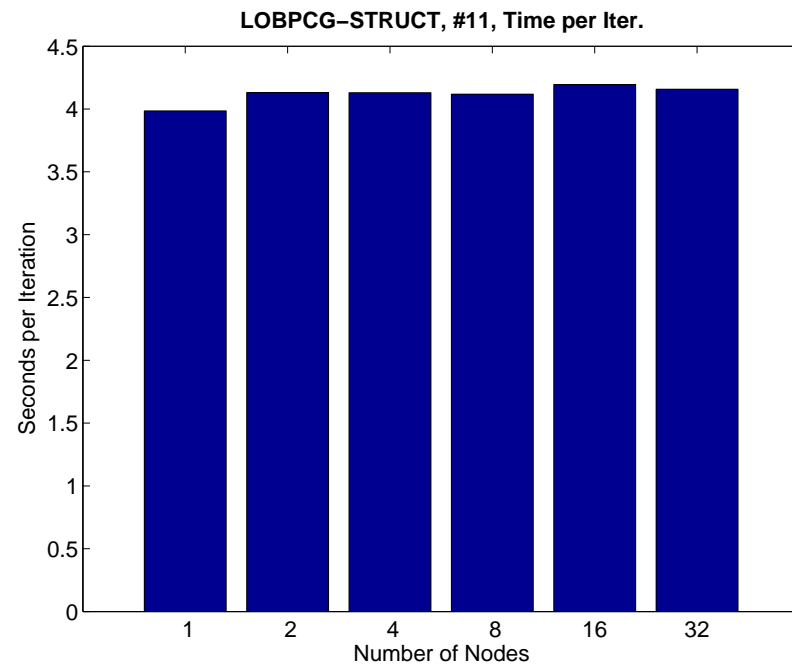
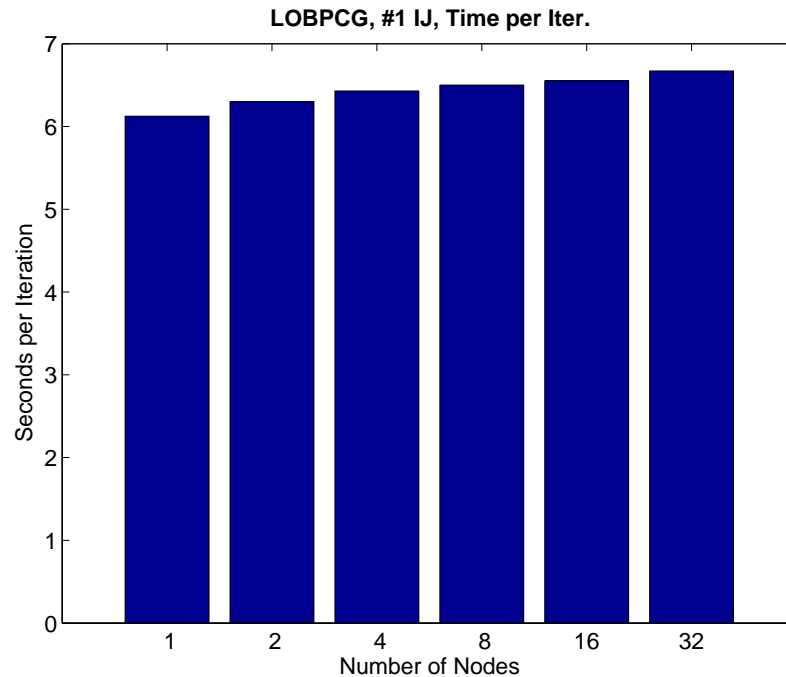
7-Point 3-D Laplacian, 1,000,000 unknowns. 1 MCR node (two 2.4-GHz Pentium 4 Xeon processors and 4 GB of memory).

LOBPCG Performance vs. Block Size



7-Point 3-D Laplacian, 2,000,000 unknowns. Preconditioner: AMG.
System: Sun Fire 880, 6 CPU.

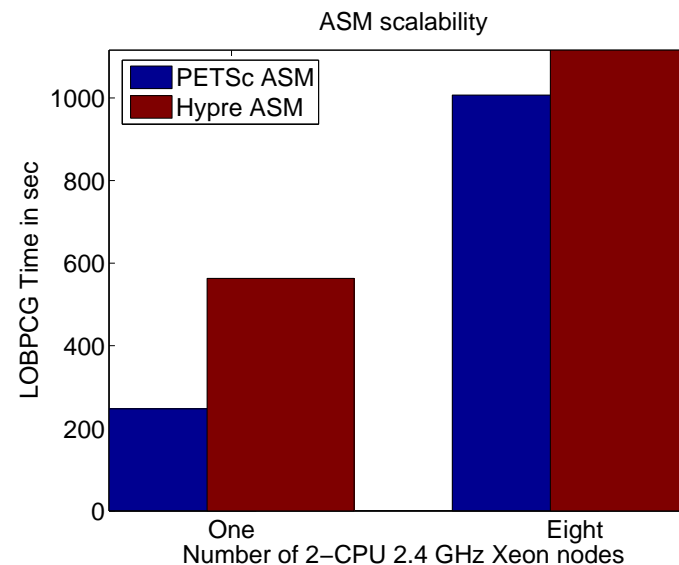
LOBPCG Scalability



Hypre, 7-Point Laplacian, 1,000,000 unknowns per node. Preconditioner: AMG. System: Beowulf (36 dual P3 1GHz 2GB nodes)

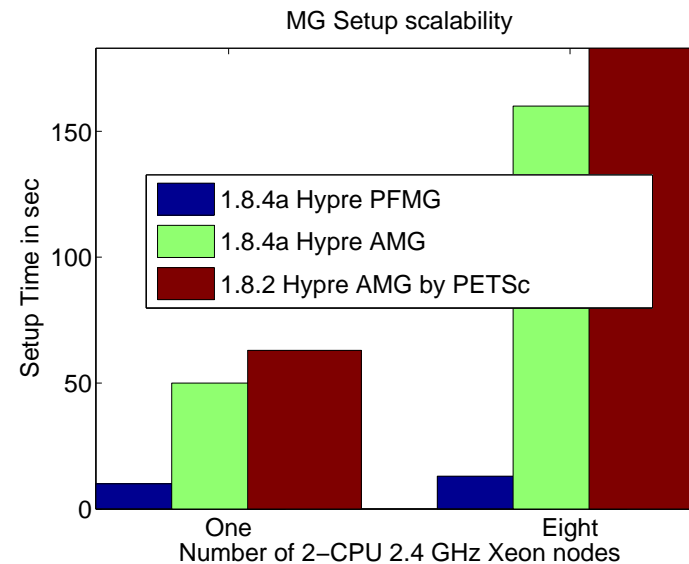
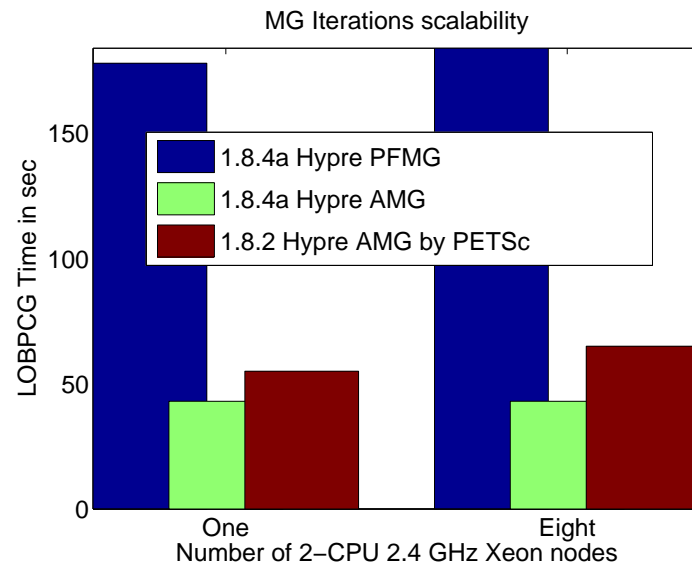
Center for Computational Mathematics, University of Colorado at Denver

LOBPCG Scalability



7-Point Laplacian, 2,000,000 unknowns per node. Preconditioner: ASM.
System: LLNL MCR, cluster of dual Pentium 4 Xeon (2.4-GHz, 4 GB) nodes.

LOBPCG Scalability



7-Point Laplacian, 2,000,000 unknowns per node. Preconditioners: AMG, PFMG. System: LLNL MCR, cluster of dual Pentium 4 Xeon (2.4-GHz, 4 GB) nodes.

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

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

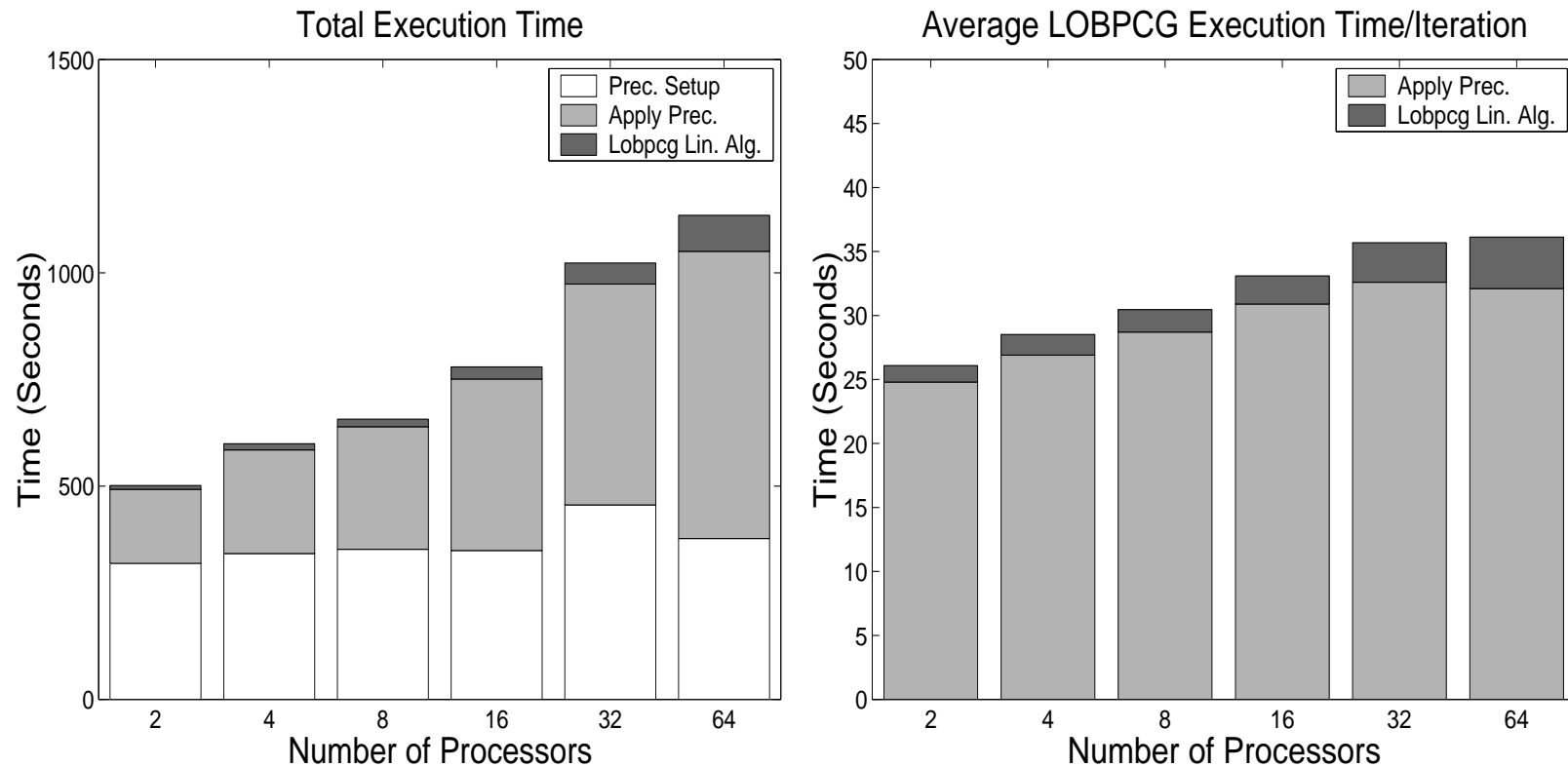


FIGURE 1: *Execution Time as Problem Size Increases for 3-D Laplacian*

Conclusions

- LOBPCG matrix free algorithm can be implemented using parallel Hypre and PETSc libraries
- The abstract implementation of LOBPCG in C allows easy deployment with different software packages
- User interface routines
 - are easy to use
 - are based on Hypre/PETSc standard interfaces
 - give user an opportunity to provide matrix-vector multiply and preconditioned solver functions
- Initial scalability looks promising, but more testing is needed by other users on larger problems