

Principal Angles Between Subspaces as Related  
to Rayleigh Quotient and Raleigh Ritz  
Inequalities with Applications to Eigenvalue  
Accuracy and an Eigenvalue Solver

**Merico E. Argentati**

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

Center for Computational Mathematics, University of Colorado at Denver

## Outline

- I. Principal Angles Between Subspaces
- II. Rayleigh Quotient Perturbations and Eigenvalue Accuracy
- III. Implementation of a Preconditioned Eigensolver Using Hypre

## Part I: Principal Angles Between Subspaces

- Angles Between Subspaces in Applications
- Theory and Algorithms in the Euclidean Norm
- Generalization to an  $A$ -Based Scalar Product
- Perturbation Theorems
- Numerical Examples
- Conclusions

## Angles Between Subspaces in Applications

- Are important in many application including statistics and information retrieval. In statistics, the angles are closely related to measures of dependency and covariance of random variables, and can be used to describe canonical correlations of a matrix pair.
- Are important in characterizing perturbations of subspaces related to Rayleigh Ritz approximations and subspace iteration.
- Are needed in analyzing perturbations related to eigensolvers, where angles provide information about solution quality and need to be computed with high accuracy.
- There are other interesting applications involving web search engines and genome research.

## Algorithms in the Euclidean Norm

Let  $u$  and  $v$  be two vectors in  $n$ -Dimensional Euclidean Space. We can compute the angle between these vectors as

$$\cos(\theta) = \frac{u^T v}{\|u\| \|v\|}$$

where  $\|\cdot\|$  is the standard Euclidean norm.

Let us consider two real-valued matrices  $F$  and  $G$ , each with  $n$  rows, and their corresponding column-spaces  $\mathcal{F}$  and  $\mathcal{G}$ , which are subspaces in  $R^n$ , assuming that

$$p = \dim\mathcal{F} \geq \dim\mathcal{G} = q \geq 1.$$

Then the *principal angles*

$$\theta_1, \dots, \theta_q \in [0, \pi/2]$$

between  $\mathcal{F}$  and  $\mathcal{G}$  may be defined recursively for  $k = 1, \dots, q$  by

$$\cos(\theta_k) = \max_{u \in \mathcal{F}} \max_{v \in \mathcal{G}} u^T v = u_k^T v_k$$

subject to

$$\|u\| = \|v\| = 1, \quad u^T u_i = 0, \quad v^T v_i = 0, \quad i = 1, \dots, k-1.$$

The vectors  $u_1, \dots, u_q$  and  $v_1, \dots, v_q$  are called principal vectors.

## Other Representations for the Largest and Smallest Principal Angles

- There is an equivalent, slightly more intuitive characterization, of the largest principal angle for  $p = q$ , which is given by

$$\theta_{\max} = \max_{u \in \mathcal{F}} \min_{v \in \mathcal{G}} \angle \{u, v\} \quad u \neq 0, v \neq 0.$$

- Let  $P_{\mathcal{F}}$  and  $P_{\mathcal{G}}$  be orthogonal projectors onto the subspaces  $\mathcal{G}$  and  $\mathcal{F}$ , respectively. Then if  $p = q$ , we have

$$\text{gap}(\mathcal{F}, \mathcal{G}) = \sin(\theta_{\max}) = \|P_{\mathcal{F}} - P_{\mathcal{G}}\|,$$

and

$$\cos(\theta_{\min}) = \|P_{\mathcal{F}}P_{\mathcal{G}}\|.$$

## An SVD Cosine–Based Algorithm

Let columns of matrices  $Q_F \in R^{n \times p}$  and  $Q_G \in R^{n \times q}$  form orthonormal bases for the subspaces  $\mathcal{F}$  and  $\mathcal{G}$  respectively. The reduced SVD of  $Q_F^T Q_G$  is

$$Y^T Q_F^T Q_G Z = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_q),$$

where  $Y \in R^{p \times q}$ ,  $Z \in R^{q \times q}$  both have orthonormal columns. Then the principal angles can be computed as

$$\theta_k = \arccos(\sigma_k), \quad k = 1, \dots, q, \quad 0 \leq \theta_1 \leq \dots \leq \theta_q \leq \frac{\pi}{2},$$

while principal vectors are given by

$$u_k = Q_F y_k, \quad v_k = Q_G z_k, \quad k = 1, \dots, q.$$

## Inaccuracy in the Cosine-based Algorithm

Let  $d$  be a constant and

$$\mathcal{F} = \text{Span} \left\{ (1 \ 0)^T \right\}, \quad \mathcal{G} = \text{Span} \left\{ (1 \ d)^T \right\}.$$

Then the angle between the one-dimensional subspaces  $\mathcal{F}$  and  $\mathcal{G}$  can be computed as

$$\theta = \arcsin \left( \frac{d}{\sqrt{1 + d^2}} \right). \quad (1)$$

Formula (1) is accurate for small angles, so we use it as an “exact” answer in the second column of the table.

It is apparent that SUBSPACE.m returns inaccurate results for  $d \leq 10^{-8}$ , which is approximately  $\sqrt{EPS}$  for double precision.

d	Formula (1)	SUBSPACE.m
1.0	7.853981633974483e-001	7.853981633974483e-001
1.0e-04	9.999999966666666e-005	9.999999986273192e-005
1.0e-06	9.999999999996666e-007	1.000044449242271e-006
1.0e-08	1.000000000000000e-008	-6.125742274543099e-017
1.0e-10	1.000000000000000e-010	-6.125742274543099e-017
1.0e-16	9.999999999999998e-017	-6.125742274543099e-017

## What is the problem?

In this simple one-dimensional example the algorithm of SUBSPACE.m is reduced to computing

$$\cos(\theta) = \frac{(u, v)}{\|u\| \|v\|} = \frac{1}{\sqrt{1 + d^2}}.$$

The cosine, that is a canonical correlation, is computed accurately and simply equals to one for all positive  $d \leq 10^{-8}$  in double precision.

## The fix: a Sine-Based Algorithm (Björck/Golub, 1973)

**Theorem 1** *Singular values  $\mu_1 \leq \mu_2 \leq \dots \leq \mu_q$  of matrix  $Q_G - Q_F(Q_F^T Q_G)$  are given by  $\mu_k = \sqrt{1 - \sigma_k^2}$ ,  $k = 1, \dots, q$ . Moreover, the principal angles satisfy the equalities  $\theta_k = \arcsin(\mu_k)$ .*

*The right principal vectors can be computed as*

$$v_k = Q_G z_k, \quad k = 1, \dots, q,$$

*where  $z_k$  are corresponding orthonormal right singular vectors of matrix  $Q_G - Q_F(Q_F^T Q_G)$ . The left principal vectors are then computed by*

$$u_k = Q_F(Q_F^T v_k) / \sigma_k, \quad k = 1, \dots, q.$$

## Some Issues for Implementation

- In our applications  $n \gg p$ , i.e. the matrices are tall, but not very wide.
- Our goal is not to store in memory, any matrix larger than  $n \times p$ .
- In our code (alg 3.2), we first use the cosine based approach for large angles, and then recompute only the small angles, using the sine based approach.

The well known approach of computing a CS decomposition, that requires computation of  $n \times n$  matrices, is too expensive when  $n \gg p$ .

## Generalization to an $A$ -Based Scalar Product

Let  $A \in R^{n \times n}$  be a fixed symmetric positive definite (SPD) matrix. Let  $(x, y)_A = (x, Ay) = y^T Ax$  be an  $A$ -based scalar product,  $x, y \in R^n$ . Let  $\|x\|_A = \sqrt{(x, x)_A}$  be the corresponding vector norm and let  $\|B\|_A$  be the corresponding induced matrix norm of a matrix  $B \in R^{n \times n}$ .

In many problems the matrix  $A$  ( $n \times n$ ) may only be available as a function. Our implementation is “matrix free”.

We do not assume that a factorization of  $A$  (such as  $K^T K$ ) is available.

## Algorithms in the $A$ -Based Scalar Product

Principal angles

$$\theta_1, \dots, \theta_q \in [0, \pi/2]$$

between subspaces  $\mathcal{F}$  and  $\mathcal{G}$  in the  $A$ -based scalar product  $(\cdot, \cdot)_A$  are defined recursively for  $k = 1, \dots, q$  by analogy with the previous definition for  $A = I$  as

$$\cos(\theta_k) = \max_{u \in \mathcal{F}} \max_{v \in \mathcal{G}} (u, v)_A = (u_k, v_k)_A \quad (2)$$

subject to

$$\|u\|_A = \|v\|_A = 1, (u, u_i)_A = 0, (v, v_i)_A = 0, i = 1, \dots, k-1. \quad (3)$$

The vectors  $u_1, \dots, u_q$  and  $v_1, \dots, v_q$  are called principal vectors relative to the  $A$ -based scalar product.

**Theorem 2** *Let columns of  $Q_F \in R^{n \times p}$  and  $Q_G \in R^{n \times q}$  be now  $A$ -orthonormal bases for the subspaces  $F$  and  $G$  respectively. Let  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_q$  be singular values of  $Q_F^T A Q_G$  with corresponding left and right singular vectors  $y_k$  and  $z_k$ ,  $k = 1, \dots, q$ . Then the principal angles relative to the scalar product  $(\cdot, \cdot)_A$  as defined in (2) and (3) are computed as*

$$\theta_k = \arccos(\sigma_k), \quad k = 1, \dots, q, \quad (4)$$

where

$$0 \leq \theta_1 \leq \dots \leq \theta_q \leq \frac{\pi}{2},$$

while the principal vectors are given by

$$u_k = Q_F y_k, \quad v_k = Q_G z_k, \quad k = 1, \dots, q.$$

**Theorem 3** *Let  $S = Q_G - Q_F(Q_F^T A Q_G)$  and  $Q_S$  be an  $A$ -orthonormal basis for the column space of  $S$ . Singular values  $\mu_1 \leq \mu_2 \leq \dots \leq \mu_q$  of matrix  $Q_S^T A S$  are given by  $\mu_k = \sqrt{1 - \sigma_k^2}$ . Moreover, the principal angles satisfy the equalities  $\theta_k = \arcsin(\mu_k)$ . The right principal vectors can be computed as*

$$v_k = Q_G z_k, \quad k = 1, \dots, q,$$

*where  $z_k$  are corresponding orthonormal right singular vectors of matrix  $Q_S^T A S$ . The left principal vectors are then computed by*

$$u_k = Q_F(Q_F^T A v_k) / \sigma_k, \quad k = 1, \dots, q.$$

In our code (alg 6.2), we first use the cosine based approach for large angles, and then recompute only the small angles, using the sine based approach.

## Perturbation Analysis

**Theorem 4** *Let  $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_q$  and  $\tilde{\sigma}_1 \geq \tilde{\sigma}_2 \geq \cdots \geq \tilde{\sigma}_q$  be cosine of principal angles between subspaces  $\mathcal{F}$ ,  $\mathcal{G}$ , and  $\tilde{\mathcal{F}}$ ,  $\tilde{\mathcal{G}}$ , respectively, computed in the  $A$ -based scalar product. Then*

$$|\sigma_k - \tilde{\sigma}_k| \leq c_1 \text{gap}_A(\mathcal{F}, \tilde{\mathcal{F}}) + c_2 \text{gap}_A(\mathcal{G}, \tilde{\mathcal{G}}), \quad k = 1, \dots, q,$$

where

$$c_1 = \max\{\cos(\theta_{\min}\{(\mathcal{G} + \tilde{\mathcal{G}}) \ominus \mathcal{G}, \mathcal{F}\}); \cos(\theta_{\min}\{(\mathcal{G} + \tilde{\mathcal{G}}) \ominus \tilde{\mathcal{G}}, \mathcal{F}\})\},$$

$$c_2 = \max\{\cos(\theta_{\min}\{(\mathcal{F} + \tilde{\mathcal{F}}) \ominus \mathcal{F}, \tilde{\mathcal{G}}\}); \cos(\theta_{\min}\{(\mathcal{F} + \tilde{\mathcal{F}}) \ominus \tilde{\mathcal{F}}, \tilde{\mathcal{G}}\})\},$$

where  $\theta_{\min}$  is the smallest angle between corresponding subspaces in the  $A$ -based scalar product.

## Perturbation Analysis (cont.)

**Theorem 5** *Let  $\mu_1 \leq \mu_2 \leq \dots \leq \mu_q$  and  $\tilde{\mu}_1 \leq \tilde{\mu}_2 \leq \dots \leq \tilde{\mu}_q$  be sine of principal angles between subspaces  $\mathcal{F}$ ,  $\mathcal{G}$ , and  $\tilde{\mathcal{F}}$ ,  $\tilde{\mathcal{G}}$ , respectively, computed in the  $A$ -based scalar product. Then*

$$|\mu_k - \tilde{\mu}_k| \leq c_3 \text{gap}_A(\mathcal{F}, \tilde{\mathcal{F}}) + c_4 \text{gap}_A(\mathcal{G}, \tilde{\mathcal{G}}), \quad k = 1, \dots, q,$$

where

$$c_3 = \max\{\sin(\theta_{\max}\{(\mathcal{G} + \tilde{\mathcal{G}}) \ominus \mathcal{G}, \mathcal{F}\}); \sin(\theta_{\max}\{(\mathcal{G} + \tilde{\mathcal{G}}) \ominus \tilde{\mathcal{G}}, \mathcal{F}\})\},$$

$$c_4 = \max\{\sin(\theta_{\max}\{(\mathcal{F} + \tilde{\mathcal{F}}) \ominus \mathcal{F}, \tilde{\mathcal{G}}\}); \sin(\theta_{\max}\{(\mathcal{F} + \tilde{\mathcal{F}}) \ominus \tilde{\mathcal{F}}, \tilde{\mathcal{G}}\})\},$$

where  $\theta_{\max}$  is the largest angle between corresponding subspaces in the  $A$ -based scalar product.

## Numerical Results

The following numerical tests were performed:

- Error Growth with Problem Size - Moderate and Small Angles
- Errors in Individual Angles
- Errors for an Ill-Conditioned Scalar Product
- Sparse matrices

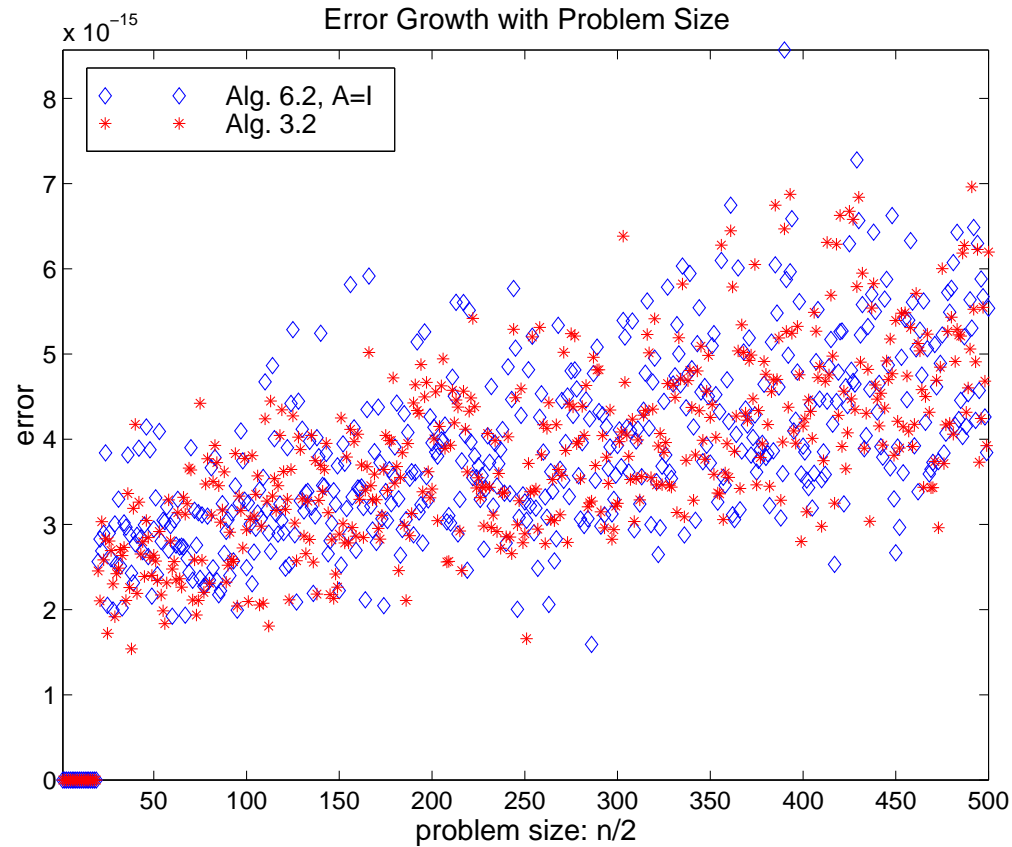


FIGURE 1: Errors in principal angles as functions of  $n/2$ ;  $p = 20$ .

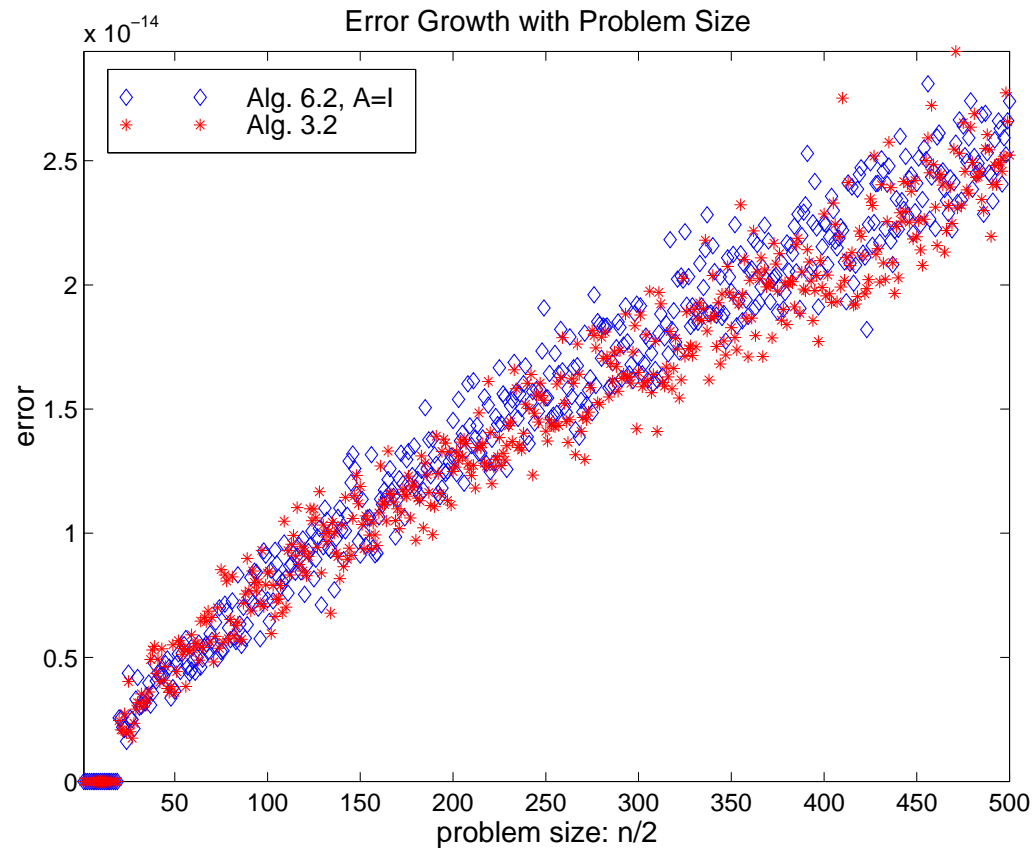


FIGURE 2: Errors in principal angles as functions of  $n/2$ ;  $p = n/2$ .

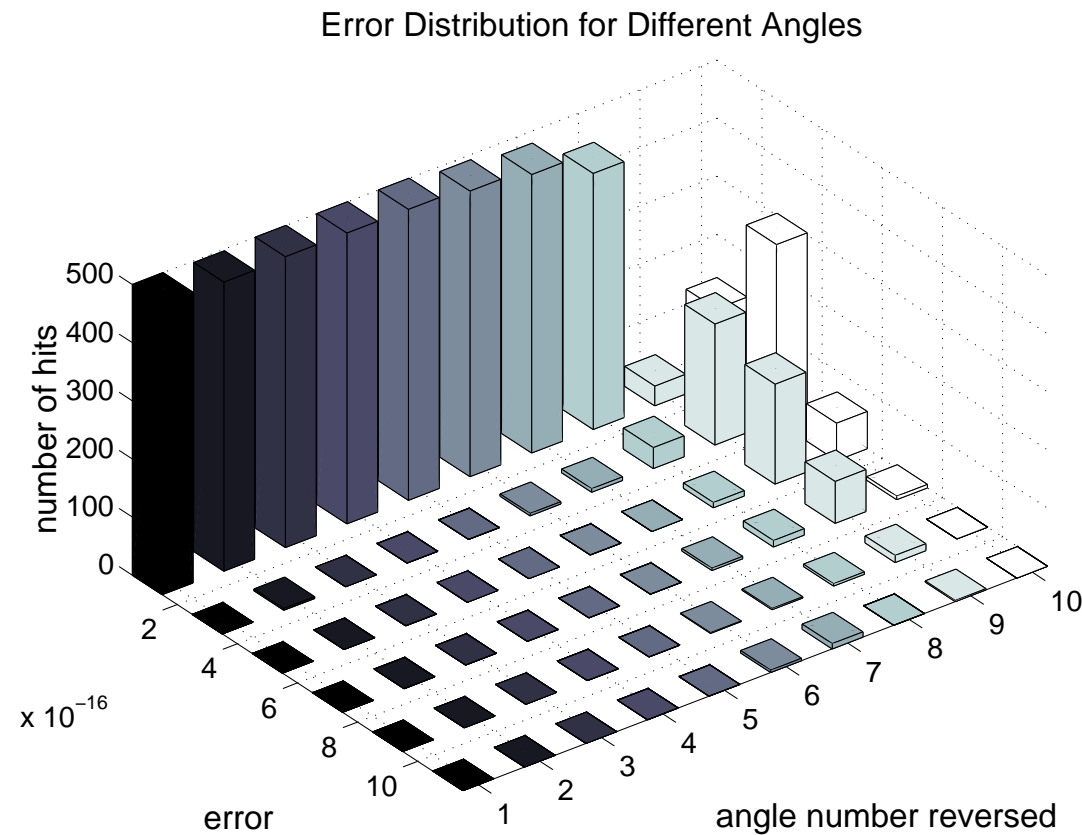


FIGURE 3: Errors for individual angles with  $n = 100, p = 10$ .

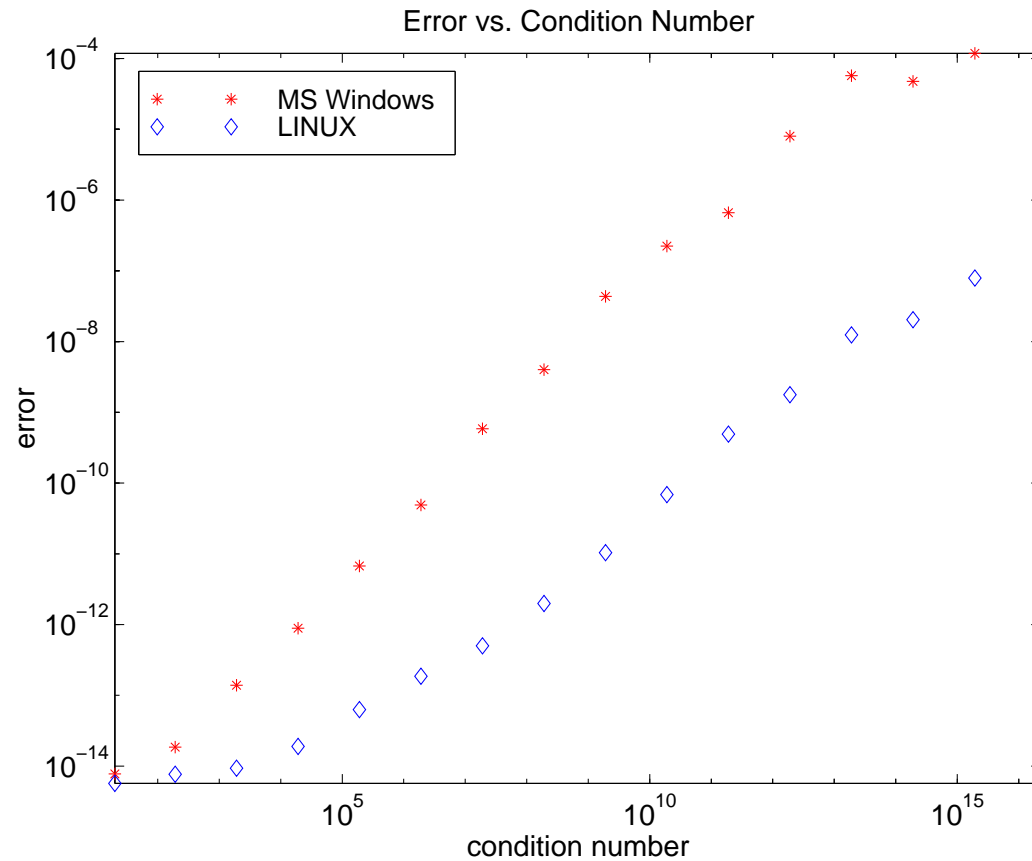


FIGURE 4: **Error increase for MS Windows and LINUX.**

## Software for Computing Principal Angles

Our code SUBSPACEA.m and the function ORTHA.m it uses have been submitted to MathWorks. They are publicly available at <http://www.mathworks.com/support/ftp/linalgv5.shtml/> as well as our fix for SUBSPACE.m.

The SVD-based algorithm for sine has been implemented in MATLAB release 13.

## Part I: Conclusions

- A bug in the cosine-based algorithm for computing principal angles between subspaces, which prevents one from computing small angles accurately in computer arithmetic, is illustrated.
- An algorithm is presented that computes all principal angles accurately in computer arithmetic and is proved to be equivalent to the standard algorithm in exact arithmetic.
- A generalization of the algorithm to an arbitrary scalar product given by a symmetric positive definite matrix is suggested and justified theoretically.
- Perturbation estimates for absolute errors in cosine and sine of principal angles, with improved constants and for an arbitrary scalar product, are derived.

## Part I: Conclusions (cont.)

- A description of the code is given as well as results of numerical tests. The code is robust in practice and provides accurate angles for large-scale and ill-conditioned cases we tested numerically. It is also reasonably efficient for large-scale applications with  $n \gg p$ .
- Our algorithms are “matrix-free”; i.e., they do not require storing in memory any matrices of the size  $n$  and are capable of dealing with  $A$ , which may be specified only as a function that multiplies  $A$  by a given vector.
- MATLAB release 13 has implemented the SVD–based sine version of our algorithm.

## Part II: Rayleigh Quotient Perturbations and Eigenvalue Accuracy

- Rayleigh quotient and Rayleigh Ritz approximations
- 2-D “mini-dimensional” analysis
- 2-D Rayleigh quotient perturbations results
- General Rayleigh quotient perturbations results
- Perturbation of Ritz values
- Conclusions

## Rayleigh Quotient and Residual

Let  $A \in \mathbf{R}^{n \times n}$  be a matrix and  $x$  be a vector in  $\mathbf{R}^n$ . We will assume that the matrix  $A$  is a symmetric and/or positive definite matrix. Then the Rayleigh quotient  $\rho(x; A)$ , is defined for  $x \in \mathbf{R}^n$  with  $x \neq 0$  by

$$\rho(x) = \rho(x; A) = (Ax, x)/(x, x),$$

and the residual is defined by

$$r(x) = r(x; A) = (A - \rho(x; A)I)x,$$

where  $(x, x) = x^T x = \|x\|^2$ .

We are interested in characterizing :  $|\rho(y) - \rho(x)|$ .

## Rayleigh Quotient Perturbations using a Matrix Inner Product

Consider the Frobenius inner product defined on the matrices  $A, B \in \mathbf{R}^{n \times n}$  by

$$(A, B) = \text{trace}(A^T B),$$

where  $\|A\|_F = \sqrt{(A, A)}$  is the Frobenius matrix norm.

For fixed  $x$ ,  $l_x(A) = \rho(x; A)$  is a bounded linear functional on  $\mathbf{R}^{n \times n}$ . According to the Riesz representation theorem, there exists a matrix  $C_x \in \mathbf{R}^{n \times n}$ , such that

$$l_x(A) = \rho(x; A) = (A, C_x).$$

## Rayleigh Quotient Perturbations using a Matrix Inner Product (cont.)

We have

$$\rho(x; A) = (A, C_x) = (A, P_x) = \text{trace}(A^T P_x) = \text{trace}(A P_x),$$

and

$$|\rho(y; A) - \rho(x; A)| = |(A, P_y - P_x)| \leq C \|P_y - P_x\|.$$

Here

$$\|P_y - P_x\| = \sin(\angle\{x, y\}) \quad \text{with} \quad \angle\{x, y\} = \arccos \frac{|(x, y)|}{\|x\| \|y\|}.$$

So

$$|\rho(y; A) - \rho(x; A)| \leq C \sin(\angle\{x, y\}).$$

## 2-D “mini-dimensional” Analysis

Assume that a matrix  $A \in \mathbf{R}^{2 \times 2}$  is symmetric and has eigenvalues  $\lambda_1 \leq \lambda_2$ , with orthonormal eigenvectors, respectively,  $u_1$  and  $u_2$ . Let

$$x = \alpha_1 u_1 + \alpha_2 u_2$$

and

$$y = \beta_1 u_1 + \beta_2 u_2$$

with  $\|x\| = 1$  and  $\|y\| = 1$ .

## 2–D “mini–dimensional” Analysis (cont.)

$$\begin{aligned}
 |\rho(y) - \rho(x)| &= (\lambda_2 - \lambda_1)|\alpha_1\beta_2 + \alpha_2\beta_1| \sin(\angle\{x, y\}) \\
 &\leq (\lambda_2 - \lambda_1)(|\alpha_1||\beta_2| + |\alpha_2||\beta_1|) \sin(\angle\{x, y\})
 \end{aligned}$$

$$\begin{aligned}
 |\rho(y) - \rho(x)| &= (\lambda_2 - \lambda_1)|(\alpha_1\alpha_2 + \beta_1\beta_2)| \tan(\angle\{x, y\}) \\
 &\leq (\lambda_2 - \lambda_1)(|\alpha_1||\alpha_2| + |\beta_1||\beta_2|) \tan(\angle\{x, y\})
 \end{aligned}$$

$$\|r(x; A)\| = (\lambda_2 - \lambda_1)|\alpha_1| |\alpha_2|$$

$$\|r(y; A)\| = (\lambda_2 - \lambda_1)|\beta_1| |\beta_2|$$

## Results for 2-D

1.	$\frac{\rho(x) - \lambda_1}{\lambda_2 - \lambda_1} = \sin^2(\angle\{x, u_1\})$
2.	$\frac{\rho(x) - \lambda_1}{\lambda_2 - \rho(x)} = \tan^2(\angle\{x, u_1\})$
3.	$\begin{aligned}  \rho(x) - \lambda  &\leq \frac{\ r(x)\ }{\ x\ } \tan(\angle\{x, u\}) \\ &= \ (I - P_x)AP_x\  \tan(\angle\{x, u\}) \end{aligned}$
4.	$ \rho(y) - \rho(x)  \leq (\lambda_2 - \lambda_1) \sin(\angle\{x, y\})$
5.	$\frac{ \rho(y) - \rho(x) }{\rho(x)} \leq (\kappa(A) - 1) \sin(\angle\{x, y\}), \quad A \text{ positive def.}$
6.	$\begin{aligned}  \rho(x) - \rho(y)  &\leq \left[ \frac{\ r(x)\ }{\ x\ } + \frac{\ r(y)\ }{\ y\ } \right] \tan(\angle\{x, y\}) \\ &= \left[ \ (I - P_x)AP_x\  + \ (I - P_y)AP_y\  \right] \tan(\angle\{x, y\}) \end{aligned}$

## Ritz Approximations

Given a subspace  $\mathcal{S}$ , we can define an operator  $\tilde{A} = P_{\mathcal{S}}A|_{\mathcal{S}}$  on  $\mathcal{S}$ , where  $P_{\mathcal{S}}$  is the orthogonal projection onto  $\mathcal{S}$  and  $P_{\mathcal{S}}A|_{\mathcal{S}}$  denotes the restriction of  $P_{\mathcal{S}}A$  to  $\mathcal{S}$ . Eigenvalues of  $\tilde{A}$  are called Ritz values,  $\tilde{\lambda}_1 \leq \dots \leq \tilde{\lambda}_m$ , and the corresponding eigenvectors,  $\tilde{u}_1 \leq \dots \leq \tilde{u}_m$ , are called Ritz vectors.

Using the Courant-Fischer Theorem, Ritz values are characterized by

$$\tilde{\lambda}_j = \min_{\substack{\mathcal{G}^j \subseteq \mathcal{S} \\ \dim \mathcal{G}^j = j}} \max_{g \in \mathcal{G}^j} \rho(g; A), \quad j = 1, \dots, m. \quad (5)$$

Equivalently, let the columns of  $Q \in \mathbf{R}^{n \times m}$  form an orthonormal basis for  $\mathcal{S}$ , then  $\hat{A} = Q^T A Q = Q^T \tilde{A} Q$  is a matrix representation of the operator  $\tilde{A} = P_{\mathcal{S}}A|_{\mathcal{S}}$  in this basis, and  $\tilde{\lambda}_j$ ,  $j = 1, \dots, m$  are the eigenvalues of  $\hat{A}$ .

## Subspaces and Rayleigh–Ritz Approximations

**Lemma 1** *Let  $\mathcal{S} \subseteq \mathbf{R}^n$  be an  $m$ -dimensional subspace with  $0 < m \leq n$ , and  $A \in \mathbf{R}^{n \times n}$  be a symmetric matrix. Let  $\tilde{A} = P_{\mathcal{S}}A|_{\mathcal{S}}$ , where  $P_{\mathcal{S}}$  is an orthogonal projection onto  $\mathcal{S}$ . Let  $\tilde{\lambda}_1 \leq \dots \leq \tilde{\lambda}_m$  be the Ritz values for  $A$  w.r.t to the subspace  $\mathcal{S}$ , and  $\lambda_1 \leq \dots \leq \lambda_n$  be the eigenvalues of  $A$ . Then the following properties hold:*

1. *Considering the Raleigh Quotient,  $\rho(x; \tilde{A}) = \rho(x; A), \forall x \in \mathcal{S}$ .*
2. *The norm of the residual is dominated as follows:  $\|r(x; \tilde{A})\| \leq \|r(x; A)\|, \forall x \in \mathcal{S}$ . If the subspace  $\mathcal{S}$  is invariant w.r.t  $A$ , then we have equality.*
3.  *$A$  has a Ritz-pair  $(\tilde{\lambda}, \tilde{u})$  (i.e. Ritz value and Ritz vector), if and only if  $A\tilde{u} - \tilde{\lambda}\tilde{u} \in \mathcal{S}^{\perp}$ .*
4. *The Ritz values satisfy:  $\lambda_i \leq \tilde{\lambda}_i \leq \lambda_n, i = 1, \dots, m$ .*
5. *Assuming that  $A$  is positive definite,  $\kappa(\tilde{A}) \leq \kappa(A)$ , where  $\kappa(\cdot)$  is the spectral condition number.*

## Results for General Case

1.	$\sin^2(\angle\{x, u_1\}) \leq \frac{\rho(x) - \lambda_1}{\lambda_2 - \lambda_1} \leq \frac{\lambda_n - \lambda_1}{\lambda_2 - \lambda_1} \sin^2(\angle\{x, u_1\})$
2.	$\frac{\rho(x) - \lambda_1}{\lambda_n - \rho(x)} \leq \tan^2(\angle\{x, u_1\})$
3.	$ \rho(x) - \lambda  \leq \frac{\ r(x)\ }{\ x\ } \tan(\angle\{x, u\})$ $= \ (I - P_x)AP_x\  \tan(\angle\{x, u\})$
4.	$\frac{ \rho(x) - \lambda }{ \rho(x) } \leq \tan(\angle\{x, Ax\}) \tan(\angle\{x, u\})$

## Results for General Case (cont.)

5.	$ \rho(y) - \rho(x)  \leq (\lambda_n - \lambda_1) \sin(\angle\{x, y\})$
6.	$\frac{ \rho(y) - \rho(x) }{\rho(x)} \leq (\kappa(A) - 1) \sin(\angle\{x, y\}), \quad A \text{ positive def.}$
7.	$ \rho(x) - \rho(y)  \leq \left[ \frac{\ r(x)\ }{\ x\ } + \frac{\ r(y)\ }{\ y\ } \right] \tan(\angle\{x, y\})$ $= [\ (I - P_x)AP_x\  + \ (I - P_y)AP_y\ ] \tan(\angle\{x, y\})$ $= [ \rho(x)  \tan(\angle\{x, Ax\}) +  \rho(y)  \tan(\angle\{y, Ay\})] \tan(\angle\{x, y\})$

## The Two Main Inequalities Are Sharp

For  $s \in (0, 1]$

$$\sup_{\sin(\angle\{x,y\})=s} \frac{|\rho(y) - \rho(x)|}{\sin(\angle\{x,y\})} = \lambda_n - \lambda_1.$$

For  $t \in (0, \infty)$

$$\sup_{\tan(\angle\{x,y\})=t} \frac{|\rho(y) - \rho(x)|}{[\|(I - P_x)AP_x\| + \|(I - P_y)AP_y\|] \tan(\angle\{x,y\})} = 1.$$

## Bounds for Rayleigh–Ritz Estimates

**Theorem 6** *Let  $A \in \mathbf{R}^{n \times n}$  be a symmetric matrix and let the eigenvalues of  $A$  be  $\lambda_1 \leq \dots \leq \lambda_n$ , Let  $\mathcal{X}$  and  $\mathcal{Y}$  both be  $m$ -dimensional subspaces of  $\mathbf{R}^n$ , and suppose  $\alpha_1 \leq \dots \leq \alpha_m$  are the Ritz values for  $A$  w.r.t  $\mathcal{X}$ , and  $\beta_1 \leq \dots \leq \beta_m$  are the Ritz values for  $A$  w.r.t  $\mathcal{Y}$ . Then*

$$|\alpha_j - \beta_j| \leq (\lambda_n - \lambda_1) \sin(\angle\{\mathcal{X}, \mathcal{Y}\}) \quad j = 1, \dots, m,$$

*and if in addition,  $A$  is positive definite, then the relative error is bounded by*

$$\frac{|\alpha_j - \beta_j|}{\alpha_j} \leq (\kappa(A) - 1) \sin(\angle\{\mathcal{X}, \mathcal{Y}\}) \quad j = 1, \dots, m,$$

*where  $\kappa(A)$  is the spectral condition number of  $A$ .*

## Part II: Conclusions

- There are two reasons for studying this problem: first, the results can be used in the design and analysis of algorithms, and second, a knowledge of the sensitivity of an eigenpair is of both theoretical and of practical interest.
- Initially restricting the domain to a 2-D subspace (i.e. performing “mini-dimensional” analysis) is a powerful technique that can yield interesting and significant results in the general case.
- This extension from 2-D to the general case requires a careful analysis of the properties of the Rayleigh quotient, the residual and the spectral condition number relative to the operators  $A$  and  $P_S A|_S$ .

## Part II: Conclusions (cont.)

- Several completely new results are presented. One of the interesting findings characterizes the perturbation of Ritz values for a symmetric positive definite matrix and two different subspaces, in terms of the spectral condition number and the largest principal angle between the subspaces.
- We also provide several alternative proofs, one of which uses a somewhat unique approach of expressing the Rayleigh quotient as a Frobenius inner product of matrices. We also provide an alternative, very simple, proof of  $|\rho(y) - \rho(x)| \leq (\lambda_n - \lambda_1) \sin(\angle\{x, y\})$ .

## Part III: Implementation of a Preconditioned Eigensolver Using Hypre

- Background concerning algorithm
- Hypre software library
- LOBPCG Hypre implementation strategy
- User interface (API)
- Compiling and Testing
- Initial Performance Results
- 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 smallest eigenvalues of a symmetric positive definite matrix
- The algorithm is matrix free since the multiplication of a vector by the matrix  $A$  and an application of the preconditioner  $T$  to a vector are needed only as functions
- For computing only the smallest eigenpair, the algorithm implements a local optimization of a 3-term recurrence

$$x^{n+1} \in \text{span}\{x^n, x^{n-1}, T(Ax^n - \lambda(x^n)x^n)\}$$

- If 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

## Previously Developed LOBPCG Software

- MATLAB
- C-language based on LAPACK/BLAS libraries
- Initial PETSc version

## 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

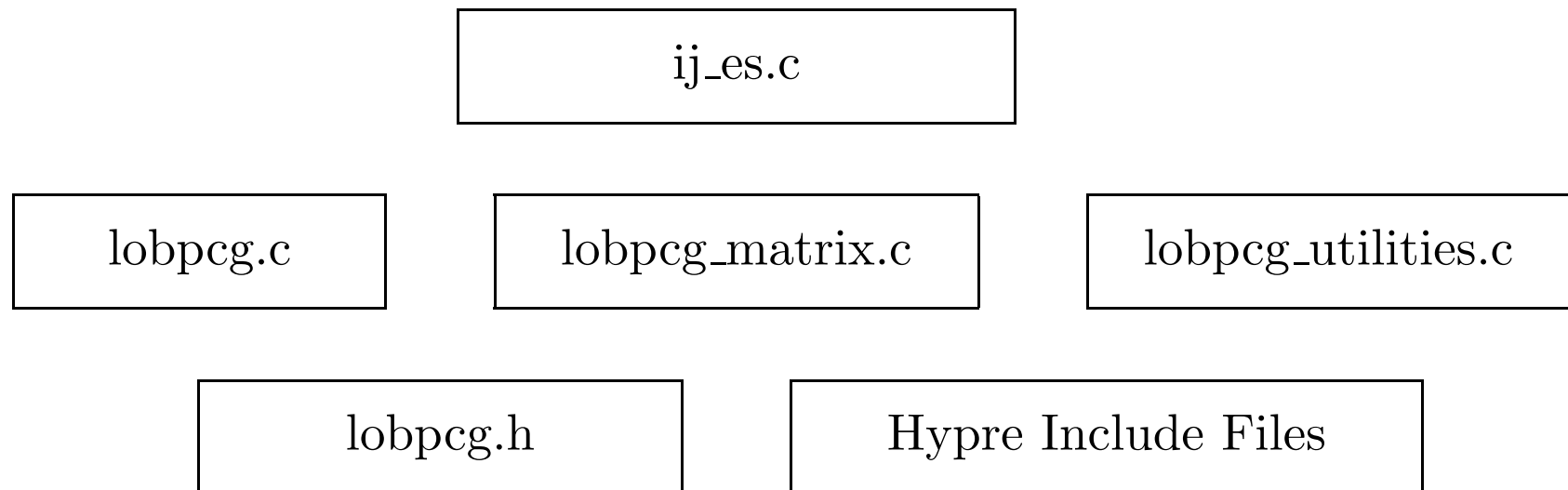
## LOBPCG Hypre Implementation

- C-language
- Hypre libraries (hypre-1.6.0) and LAPACK/BLAS libraries
- Implementation is based on Hypre Linear Algebraic (IJ) Interface for applications with sparse linear systems
- User interface to solver is a “Hypre style” API
- 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.)

- Test driver `ij_es.c`, is modified version of `IJ_linear_solvers.c` (from Hypre Release 1.6.0) 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_es.sh` provides basic suite of tests
- LOBPCG Hypre based code will be included in the next **Beta Release** of the **Hypre Project**

## LOBPCG Software Implemented Using Hypre



## LOBPCG User Interface (API)

### I. Setup Functions

```
HYPRE_LobpcgCreate(HYPRE_LobpcgData *lobpcg);  
HYPRE_LobpcgSetup(HYPRE_LobpcgData lobpcg);  
HYPRE_LobpcgSetVerbose(HYPRE_LobpcgData lobpcg);  
HYPRE_LobpcgSetMaxIterations(HYPRE_LobpcgData lobpcg,int max_iter);  
HYPRE_LobpcgSetTolerance(HYPRE_LobpcgData lobpcg,double tol);  
HYPRE_LobpcgSetBlocksize(HYPRE_LobpcgData lobpcg,int bsize);  
HYPRE_LobpcgSetSolverFunction(HYPRE_LobpcgData lobpcg,  
    int (*FunctSolver)(HYPRE_ParVector x,HYPRE_ParVector y));  
HYPRE_LobpcgDestroy(HYPRE_LobpcgData lobpcg);
```

### II. Lobpcg Solver

```
HYPRE_LobpcgSolve(HYPRE_LobpcgData lobpcgdata,  
    int (*FunctA)(HYPRE_ParVector x,HYPRE_ParVector y),  
    HYPRE_ParVector *v,double **eigval);
```

### III. Output Functions

```
HYPRE_LobpcgGetEigval(HYPRE_LobpcgData lobpcg,double **eigval);  
HYPRE_LobpcgGetResvec(HYPRE_LobpcgData lobpcg,double ***resvec);  
HYPRE_LobpcgGetEigvalHistory(HYPRE_LobpcgData lobpcg,double ***eigvalhistory);
```

## Compiling and Testing

- Make files for various hardware configurations and compilers are provided
- A test program `ij_es.sh`, similar to `IJ_linear_solvers.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

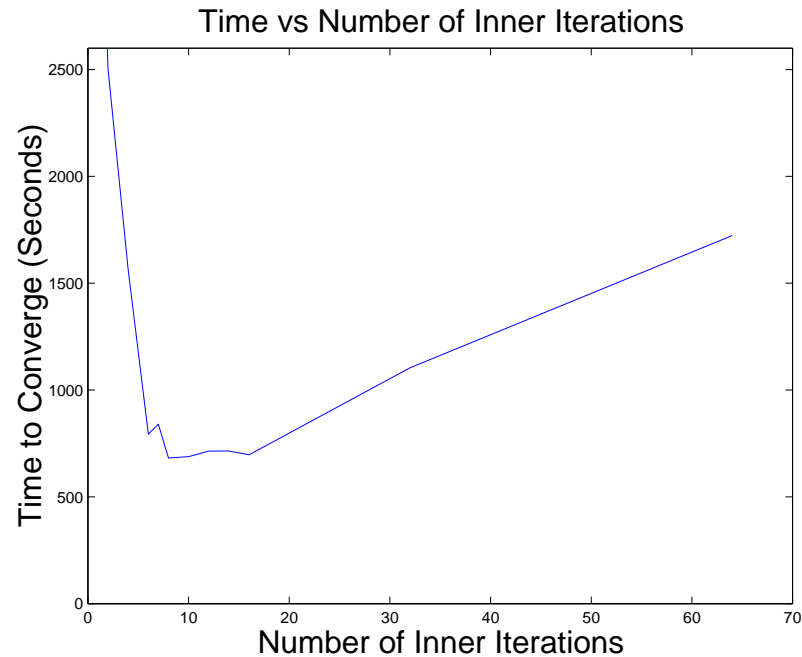
## Compiling and Testing (cont.)

- 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.

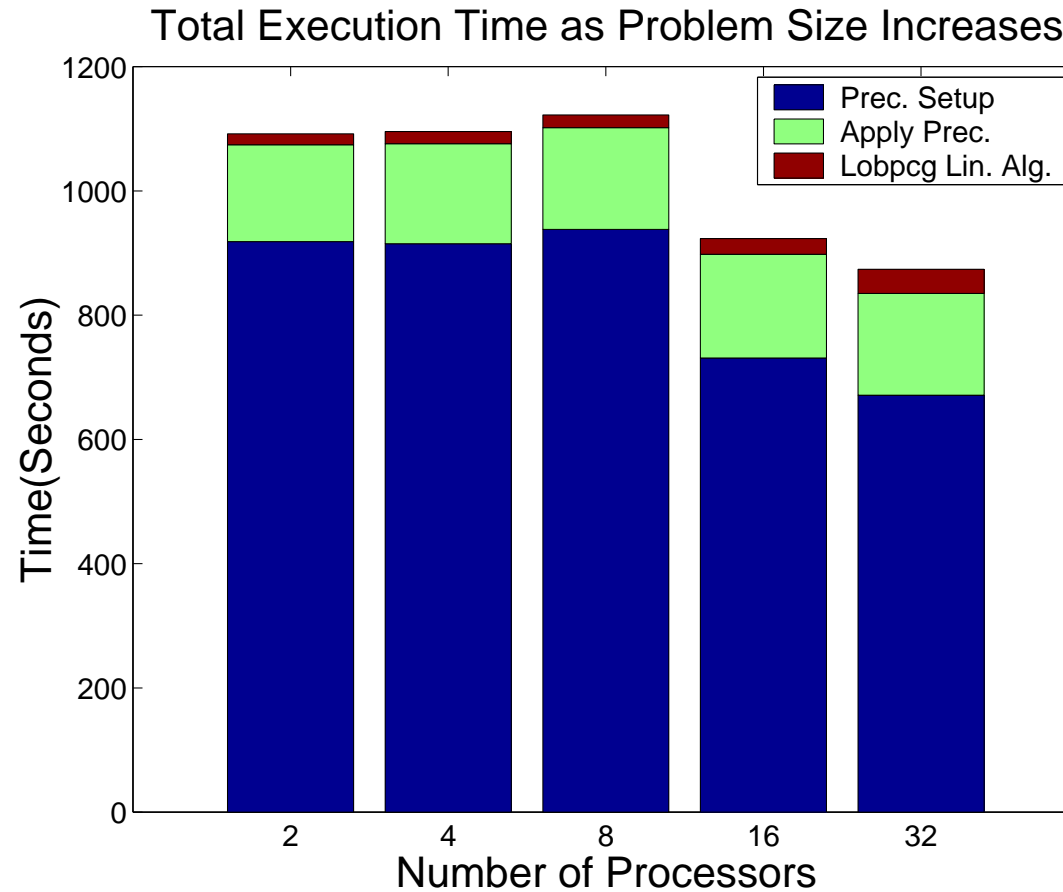
Center for Computational Mathematics, University of Colorado at Denver

## Timing Results – Scalability I

Nproc	Nz	Prec. setup	Apply Prec.	LOBPCG Lin. Alg.
2	6,940,000	918.5	155.7	17.7
4	13,907,376	914.9	161.1	19.4
8	27,986,067	937.9	163.9	20.3
16	55,760,000	730.9	166.9	25.3
32	112,000,000	671.0	163.8	38.9

7-Point 3-D Laplacian, Block size: 1, Lobpcg iterations: 10, Inner (pcg) iterations: 3, Preconditioner/Solve: Schwarz-PCG.

## Timing Results – Scalability I (cont.)

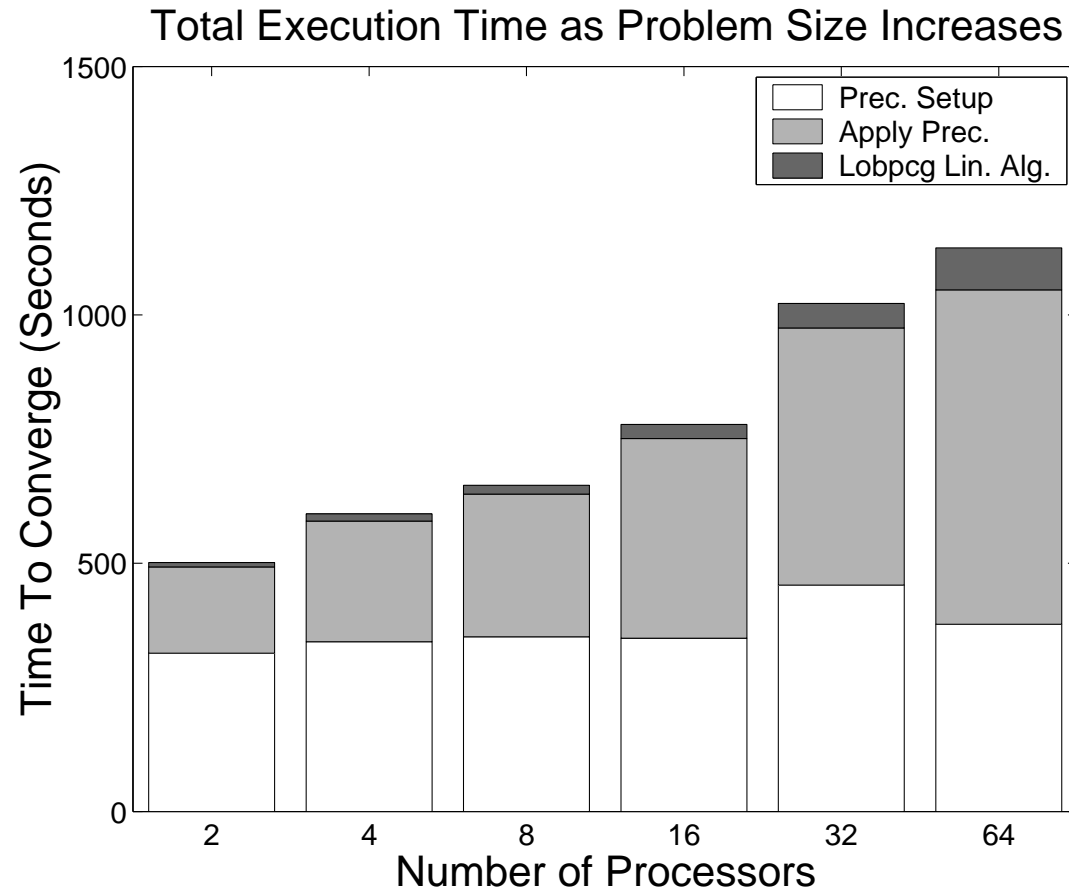


## Timing Results – Scalability II

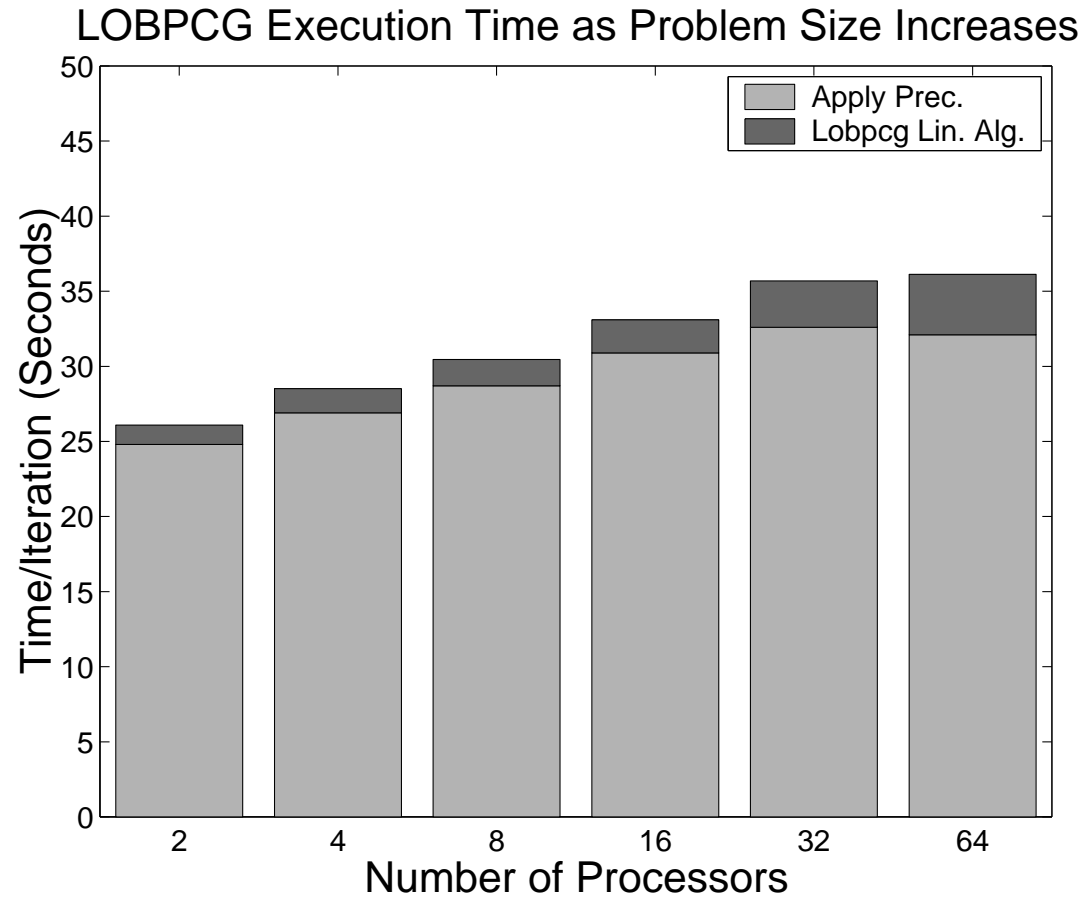
Nproc	Nz	Prec. setup	Apply Prec.	LOBPCG Lin. Alg.	Itr	Apply/Itr	Alg/Itr
2	3.4M	319	173.4	9.10	7	24.8	1.30
4	6.9M	342	242.8	14.5	9	26.9	1.61
8	13.9M	352	287.2	17.6	10	28.7	1.76
16	27.9M	349	401.9	28.6	13	30.9	2.20
32	55.8M	456	517.8	49.4	16	32.6	3.09
64	111.6M	377	673.4	84.5	21	32.1	4.03

7-Point 3-D Laplacian, Block size: 1, Inner (pcg) iterations: 10,  
Preconditioner/Solve: Schwarz-PCG.

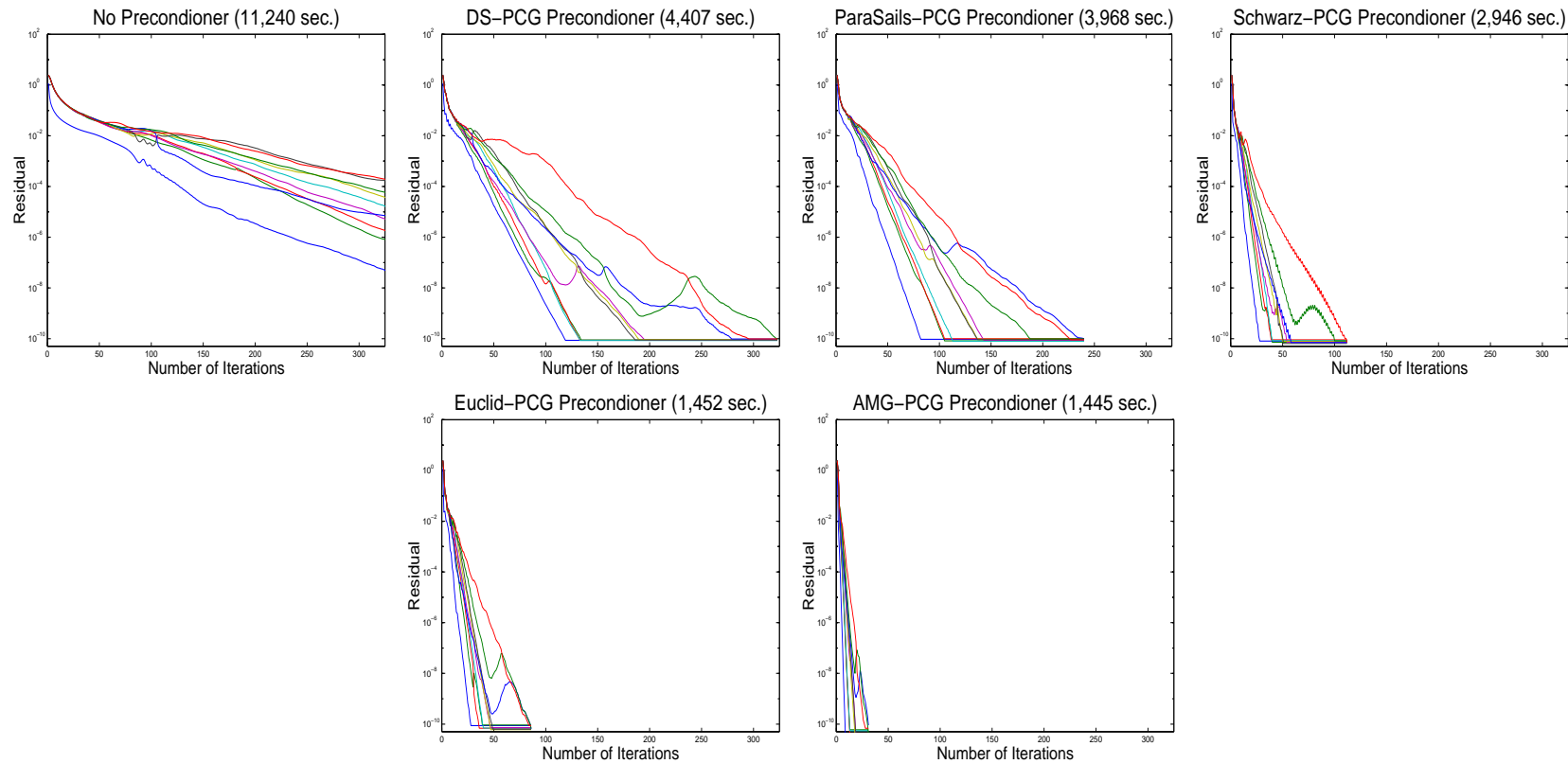
## Scalability II



## Scalability II (cont)



# Convergence Rates for Different Preconditioners



## Part III: Conclusions

- This implementation illustrates that this “matrix-free” algorithm can be successfully implemented using parallel libraries that are designed to run on a great variety of multiprocessor platforms.
- We gain a significant leverage and flexibility in solving large sparse eigenvalue problems because the user can provide their own matrix–vector multiply and preconditioned solver functions and/or use the standard Hypr preconditioned solver functions.
- The user interface routines
  - Provide ease of use for a variety of users.
  - Have been developed with the goal of using the Hypr standard user interface.

## Part III: Conclusions (cont.)

- Initial scalability measurements look promising, but more testing is needed by other users. However, scalability is mainly dependent on the scalability within Hypr since most of the computations are involved with building and applying the preconditioner. In practical problems 90%–99% of the computational effort is required for building the preconditioner and in the applying the preconditioner during execution of the algorithm.
- Enhancements are possible to improve robustness, efficiency and readability/clarity of code.
- The LOBPCG Hypr software has been integrated into the Hypr software at LLNL and has been included in the recently released Hypr Beta Release – Hypr–1.8.0b, and so is now available for users to test.

## Future Work Directions

- Principal Angles Between Subspaces
  - Sparse implementation of algorithm
  - Applications such as information retrieval/web search engines
- Rayleigh Quotient Perturbations and Eigenvalue Accuracy
  - Considering Ritz values for  $A$  w.r.t  $\mathcal{X}$  and  $\mathcal{Y}$ , generalize

$$|\alpha_j - \beta_j| \leq (\lambda_n - \lambda_1) \sin(\angle\{\mathcal{X}, \mathcal{Y}\}).$$

- Is following true?

$$|\alpha_j - \beta_j| \leq [\|(I - P_{\mathcal{X}})AP_{\mathcal{X}}\| + \|(I - P_{\mathcal{Y}})AP_{\mathcal{Y}}\|] \tan(\angle\{\mathcal{X}, \mathcal{Y}\}).$$

- Implementation of a Preconditioned Eigensolver Using Hypre
  - Enhancements based on feedback from users
  - Implement generalized eigenvalue problem  $Ax = \lambda Bx$