

# Hard and soft locking in iterative methods for symmetric eigenvalue problems

**Andrew V. Knyazev**

*Department of Mathematics*

*Center for Computational Biology*

*Center for Computational Mathematics*

*University of Colorado at Denver*

Copper Mountain Conference on Iterative methods

April 1, 2004 (not a joke)

*This material is based upon work supported by the NSF*

Center for Computational Mathematics, University of Colorado at Denver

## Abstract

Let us consider a problem of computing a number of eigenvectors of a symmetric matrix by simultaneous iterations. When computing several eigenvectors simultaneously it is often observed that some eigenvectors converge faster than the others. To avoid unnecessary computational work, it is common to “lock” the eigenvectors that have already converged within a required tolerance while continue iterating other eigenvectors. A typical locking technique is a deflation by restriction, where the locked eigenvectors are no longer changed, at the same time as the eigenvectors that are still iterating are kept orthogonal to the locked ones. We call this technique “hard locking.”

A different technique, called “soft locking” is for simultaneous iterations combined with the Rayleigh–Ritz method. The core idea of the hard and soft locking is the same: the locked vectors do not participate in the main iterative step. The soft locked vectors, however, fully participate in the Rayleigh–Ritz method and thus are allowed to be changed by the Rayleigh–Ritz method. The orthogonality of the soft locked vectors to iterative vectors follows from the well-known fact that the Ritz vectors can be chosen to be orthogonal automatically.

The soft locking is computationally more expensive compared to the hard locking. The advantage of the soft locking is that it provides more accurate results. First and foremost, adding more vectors to the hard locking procedure decreases the accuracy of the orthogonal complement to their span, which may prevent the iterative vectors from reaching the required tolerance. This effect seems less problematic for soft locking.

Second, the soft locked vectors tend to become more accurate from participating in the Rayleigh–Ritz method even though they are removed from the main iterative step.

In the present talk, we give a detailed description of soft locking and present preliminary theoretical and numerical results demonstrating the effectiveness of the soft locking compared to the traditional hard locking and discuss peculiarities of possible implementations of the hard and soft locking in preconditioned block eigenvalue solvers such as the Locally Optimal Block Preconditioned Conjugate Gradient Method.

## OUTLINE:

1. Preconditioned eigenvalue solvers
2. What is **LOBPCG**?
3. Block iterations in **LOBPCG**
4. **Classical hard** locking in **LOBPCG**
5. **Symmetric hard** locking in **LOBPCG**
6. **Soft** locking in **LOBPCG**
7. **LOBPCG** Software with soft and hard locking
8. Conclusions

## Preconditioned eigenvalue solvers

We consider a preconditioned symmetric definite eigenproblem

$$T(Ax - \lambda Bx) = 0, \quad A = A^*, \quad B = B^* > 0, \quad T = T^* > 0$$

and apply an eigensolver. With an appropriate preconditioning  $T$ , such a method can be very efficient as has been shown numerically.

We concentrate in this talk on one of the most promising preconditioned eigensolvers, namely, on the Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG) method. In LOBPCG for computing a single eigenpair of the pencil  $A - \lambda B$ , the new iterate is determined by the Rayleigh–Ritz method on a three-dimensional subspace, which includes the previous iterate in addition to the current iterate and the preconditioned residual  $T(Ax - \lambda Bx)$ .

## What is LOBPCG?

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

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

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

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*, namely, select  $\tau^{(i)}$  and  $\gamma^{(i)}$  that minimize the Rayleigh quotient  $\lambda(x^{(i+1)})$  by using the *Rayleigh–Ritz method* for the pencil  $A - \lambda B$ .

Three-term recurrence + Rayleigh–Ritz method =  
Locally Optimal Conjugate Gradient Method

## Block iterations in LOBPCG

A well known idea of using *Simultaneous, or Block Iterations* provides an important improvement over single-vector methods, and permits us to compute an  $m > 1$  dimensional invariant subspace, rather than one eigenvector at a time. It can also serve as an acceleration technique over a single-vector methods on parallel computers, as convergence for extreme eigenvalues usually increases with the size of the block, and every step can be naturally implemented on wide varieties of *multiprocessor computers* as well as to take advantage of high level *BLAS* libraries.

The LOBPCG Method, is a straightforward generalization of the single-vector version enhanced with the Rayleigh–Ritz procedure on a larger trial subspace.

## Classical hard locking in LOBPCG

LOBPCG with no locking:

$$x^{(i+1)} = w^{(i)} + \tau^{(i)} x^{(i)} + \gamma^{(i)} x^{(i-1)}, \quad w^{(i)} = T(Ax^{(i)} - \lambda^{(i)} Bx^{(i)}),$$

Let  $P$  be a projector on the complement to the span of previously founded eigenvectors. For LOBPCG with locking, we redefine

$$w^{(i)} = PT(Ax^{(i)} - \lambda^{(i)} Bx^{(i)}),$$

so that all iterates are now within the range of  $P$  if the initial guess is chosen within the range. Traditionally,  $P$  is orthogonal in the  $B$ -based scalar product and the complement is orthogonal also in the  $B$ -based scalar product, since this allows the easiest and inexpensive implementation.

## Symmetric hard locking in LOBPCG

Let's consider the projected preconditioned residual operator

$$PT(A - \lambda^{(i)} B),$$

from the definition of the projected preconditioned residual

$$w^{(i)} = PT(Ax^{(i)} - \lambda^{(i)} Bx^{(i)}).$$

Matrices  $A$  and  $B$  are symmetric, and so is  $A - \lambda^{(i)} B$ . Now,  $T$  is symmetric positive definite, thus  $T(A - \lambda^{(i)} B)$  is symmetric in  $T^{-1}$ -based scalar product. However,  $PT(A - \lambda^{(i)} B)$  is not necessarily symmetric in any scalar product. How to make it symmetric and why do we care?

**Theorem 1.** If  $P$  is symmetric in  $T^{-1}$ -based scalar product than  $PT(A - \lambda^{(i)}B)$  is symmetric in the range of  $P$  equipped with the  $T^{-1}$ -based scalar product.

**Theorem 2.** If  $P$  is symmetric in  $T^{-1}$ -based scalar product then the  $P$  projected LOBPCG method is equivalent to the non-projected LOBPCG applied to the pencil  $PTA - \lambda PTB$  within the range of  $P$ , which is symmetric definite in the  $T^{-1}$ -based scalar product.

Thus, we can use backward analysis ideas to investigate the influence of inexactness in the orthoprojector  $P$ . The range of  $P$  is not exactly an invariant subspace, since we lock approximate eigenvectors.

The symmetric locking is somewhat more expensive, compared to the classical one, but it is theoretically justified and may provide better accuracy.

## Soft locking in LOBPCG

The accuracy of the hard locking is expected to deteriorate with the increase of the number of locked vectors, since the inaccuracy in locked vectors results in accumulating inaccuracy of iterative vectors. To resolve this issue, the soft locking allows the locked vectors to be changed by the RR procedure, even though the locked vectors are no longer iterated.

Let  $X_a$  contains active iterative vectors as columns and  $X_{sl}$  is made of the soft locked vectors. Only  $X_a$  participates in iterations, i.e., having  $\Lambda_a^{(i)}$ ,  $X_a^{(i)}$ ,  $X_a^{(i-1)}$  and we compute  $W_a^{(i)} = T(AX_a^{(i)} - BX_a^{(i)}\Lambda_a^{(i)})$ , where  $\Lambda_a^{(i)}$  is a diagonal matrix of approximate eigenvalues corresponding to  $X_a^{(i)}$ . If  $X_{sl}$  is empty, i.e., there is no soft locking, we run the RR on the span of  $X_a^{(i)}$ ,  $X_a^{(i-1)}$  and  $W_a^{(i)}$  and choose  $X_a^{(i+1)}$  as Ritz vectors corresponding to  $\text{rank}(X_a^{(i)})$  smallest Ritz values  $\Lambda_a^{(i+1)}$ .

When  $X_{sl}$  is not empty, we run the RR on the span of  $X_a^{(i)}$ ,  $X_a^{(i-1)}$ ,  $W_a^{(i)}$  and  $X_{sl}^{(i)}$  and choose  $X_a^{(i+1)}$  and  $X_{sl}^{(i)}$  as Ritz vectors corresponding to  $\text{rank}(X_a^{(i)}) + \text{rank}(X_{sl}^{(i)})$  smallest Ritz values  $\Lambda_a^{(i+1)}$  and  $\Lambda_{sl}^{(i+1)}$ .

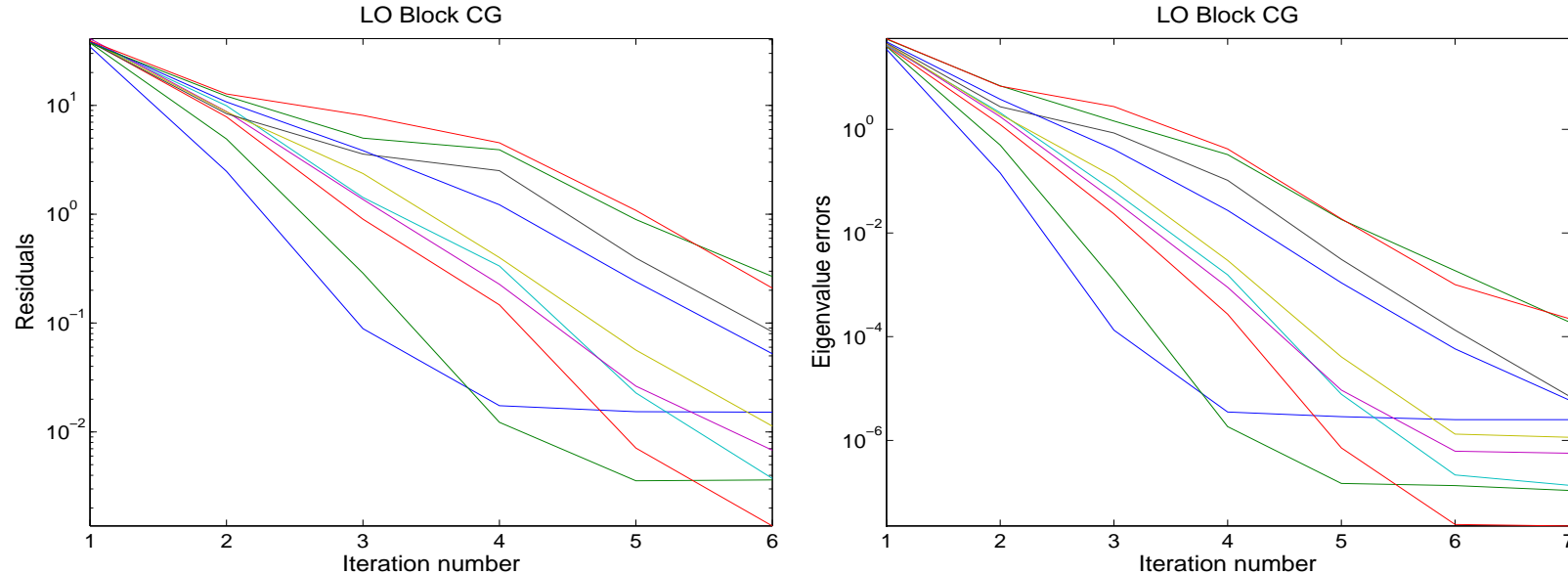


FIGURE 1: Unlocked vectors: 10,10,9,8,4,2

## LOBPCG Software with soft and hard locking

The LOBPCG code with soft locking is publicly available in MATLAB, see <http://math.cudenver.edu/~aknyazev/software/CG/> and in C using MPI and HYPRE libraries for massively parallel computers, see <http://www.llnl.gov/CASC/hypre/>

The LOBPCG revision with both versions of hard locking, in addition to soft locking, is a work in progress.

## Conclusions

1. Locking is a necessary and important part of most eigensolvers.
2. Approximate locking is not well theoretically studied.
3. Approximate classical hard locking is somewhat theoretically supported only asymptotically (D'yakonov, Knyazev, 1992).
4. Approximate symmetric hard locking is well theoretically supported, but more expensive. Work in progress.
5. Soft locking is more expensive than the hard locking, but may be more accurate. Work in progress.

Work in progress jointly with E. Ovchinnikov (next speaker)