

A Comparison of Parallel Preconditioners for the Sparse  
Generalized  
Eigenvalue Problems by Rayleigh-quotient Minimization

Sangback Ma

Department of Computer Science  
Hanyang University, Ansan, Korea

Ho-Jong Jang

Department of Mathematics  
Hanyang University, Seoul, Korea

## Introduction

---

Sparse Generalized Eigenvalue Problem  $Ax = \lambda Bx$ .

- Arises in many applications, such as structural mechanics, chemistry, and magnetohydrodynamics.
- The interior eigenvalues are often important.
- The smallest eigenvalue minimizes the Rayleigh-quotient.
- A and B are large and sparse, A is symmetric and B is SPD.
- In this case we could use CG-type method to find the generalized eigenvalues.
- Preconditioners could accelerate the convergence.
- For parallel processing we need good parallel preconditioners.

## Minimization of Rayleigh Quotients

---

Consider the generalized eigenvalue problem

$$Ax = \lambda Bx, \tag{1}$$

where  $A$  and  $B$  are large sparse symmetric matrix of dimension  $n$  and  $B$  is further SPD. Let

$$0 < \lambda_1 < \lambda_2 \leq \lambda_3 \leq \dots \leq \lambda_n$$

be the eigenvalues of (1), and let  $z_1, z_2, \dots, z_n$  be the corresponding eigenvectors.

We recall that the eigenvectors of (1) are the stationary points of the Rayleigh quotient

$$R(x) = \frac{x^T Ax}{x^T Bx}, \tag{2}$$

and the gradient of  $R(x)$  is given by

$$g(x) = \frac{2}{x^T x} [Ax - R(x) x].$$

For an iterate  $x^{(k)}$ , the gradient of  $R(x^{(k)})$ ,

$$\nabla R(x^{(k)}) = g^{(k)} = \frac{2}{x^{(k)T} x^{(k)}} [Ax^{(k)} - R(x^{(k)})x^{(k)}],$$

is used to fix the direction of descent  $p^{(k+1)}$  in which  $R(x)$  is minimized. These directions of descent are defined by

$$p^{(1)} = -g^{(0)}, \quad p^{(k+1)} = -g^{(k)} + \beta^{(k)}p^{(k)}, \quad k = 1, 2, \dots,$$

where  $\beta^{(k)} = \frac{g^{(k)T} g^{(k)}}{g^{(k-1)T} g^{(k-1)}} [1,10]$ . The subsequent iterate  $x^{(k+1)}$  along  $p^{(k+1)}$  through  $x^{(k)}$  is written as

$$x^{(k+1)} = x^{(k)} + \alpha^{(k+1)}p^{(k+1)}, \quad k = 0, 1, \dots,$$

where  $\alpha^{(k+1)}$  is obtained by minimizing  $R(x^{(k+1)})$ ,

$$R(x^{(k+1)}) = \frac{x^{(k)T} Ax^{(k)} + 2\alpha^{(k+1)}p^{(k+1)T} Ax^{(k)} + \alpha^{(k+1)2}p^{(k+1)T} Ap^{(k+1)}}{x^{(k)T} x^{(k)} + 2\alpha^{(k+1)}p^{(k+1)T} x^{(k)} + \alpha^{(k+1)2}p^{(k+1)T} p^{(k+1)}}.$$

The performance of the CG method for computing the eigenpairs of (1) can be improved by using a preconditioner[2 13]. The idea behind the PCG is to apply

the “regular” CG scheme to the transformed system

$$\tilde{A}\tilde{x} = \lambda\tilde{x},$$

where  $\tilde{A} = C^{-1}AC^{-1}$ ,  $\tilde{x} = Cx$ , and  $C$  is nonsingular symmetric matrix. By substituting  $x = C^{-1}\tilde{x}$  into (2), we obtain

$$R(\tilde{x}) = \frac{\tilde{x}^T C^{-1} A C^{-1} \tilde{x}}{\tilde{x}^T C^{-1} C^{-1} \tilde{x}} = \frac{\tilde{x}^T \tilde{A} \tilde{x}}{\tilde{x}^T \tilde{x}}, \quad (3)$$

where the matrix  $\tilde{A}$  is symmetric positive definite. The transformation (3) leaves the stationary values of (2) unchanged, which are eigenvalues of (1), while the corresponding stationary points are obtained from  $\tilde{x}_j = Cz_j$ ,  $j = 1, 2, \dots, n$ .

The matrix  $M = C^2$  is called the preconditioner. There are a number of choices of  $M$  ranging from simple to complicated forms. In this paper, Multi-Color Block SSOR preconditioner is used with parallel computation aspect. The PCG algorithm for solving the smallest eigenpair with implicit preconditioning is summarized as follows.

## Preconditioned CG for eigenvalue problem

### ALGORITHM 0.1 PCG for Computing the Smallest Eigenpair of (1)

1. *Compute the preconditioner  $M$ .*
2. *Choose an initial guess  $x^{(0)} \neq 0$ .*
3. *Construct the initial gradient direction  $g^{(0)}$ .*

$$\text{Set } p^{(1)} = -g^{(0)} \text{ and } Mh^{(0)} = g^{(0)}.$$

4. *Iterate for  $k = 0$  to  $NMAX$ (maximum number of iterations).*
5. *If  $k = 0$  then set  $\beta^{(k)} = 0$ , otherwise compute*

$$Mh^{(k)} = g^{(k)} \text{ and } \beta^{(k)} = \frac{g^{(k)T} h^{(k)}}{g^{(k-1)T} h^{(k-1)}}.$$

6. *Compute*

$$p^{(k+1)} = -h^{(k)} + \beta^{(k)}p^{(k)}. \quad (4)$$

7. *Compute*  $\alpha^{(k+1)}$  *by minimizing*  $R(x^{(k+1)})$ .

8. *Compute*

$$x^{(k+1)} = x^{(k)} + \alpha^{(k+1)}p^{(k+1)}. \quad (5)$$

9. *Test on convergence.*

- The theoretical proof of convergence is available for a variant of the above version called *locally optimal block* preconditioned CG. LOBPCG is described in Andrew 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.

## **Preconditioners**

---

- As in the case of linear system solvers, there is no widely accepted parallel preconditioner.
- ILU type is powerful but inherently serial.
- Multi-coloring is simple way to achieve a parallelism of order  $n$  but convergence rate is deteriorated.

## **Multi-color reordering**

---

- Given a mesh, multi-coloring consists of assigning a color to each point so that the couplings between two points of the same color are eliminated in the discretization matrix.

- However, it has been known that the convergence rate for the reordered systems often deteriorates. The table 1 contains the rates of convergence for SSOR with optimal  $\omega$ , and ILU(0) preconditioned CG methods with natural and red/black ordering for the 5-point Laplacian matrix.[7]

Table 1: Rate of convergence when reordering is used.  $h$  is the meshsize.

	SOR	SSOR	ILU-CG
Natural Ordering	$O(h)$	$O(h)$	$O(\sqrt{h})$
Red/Black Ordering	$O(h)$	$O(h^2)$	$O(h)$

- For the model problem SSOR and preconditioned-CG with the Red/Black ordering have a worse convergence rate than with the natural ordering, while SOR has the same rate if optimal  $\omega$  is used.

## **Wavefront-ordering(Level scheduling)**

---

Rather than pursuing the parallelisms through reordering, the wavefront technique exploits the structure of the given matrix. If the matrix comes from the discretizations of PDEs such as by FDM or FEM, the value of a certain node is usually dependent on only the values of its neighbors. Hence, once the values of its neighbors are known that node can be updated.

For references, see [7].

# Multi-Color Block SSOR(Symmetric Successive OverRelaxation)

## Method

---

- Multi-Coloring is a way to achieve parallelism of order  $N$ , where  $N$  is the order of the matrix. For example, it is known that for 5-point Laplacian we can order the matrix in 2-colors so that the nodes are not adjacent with the nodes with the same color. This is known as Red/Black ordering. For planar graphs maximum four colors are needed.
- Blocked methods are useful in that they minimize the interprocessor communications, and increases the convergence rate as compared to point methods. SSOR is a symmetric preconditioner that is expected to perform as efficiently as incomplete Cholesky factorization combined with blocking. Instead we need to invert the diagonal block.

- SSOR needs a  $\omega$  parameter for overrelaxation. However, it is known that the convergence rate is not so sensitive to the  $\omega$  parameter.

Let the domain be divided into  $L$  blocks. Suppose that we apply a multi-coloring technique, such as a greedy algorithm described in [11], to these blocks so that a block of one color has no coupling with a block of the same color. Let  $D_j$  be the coupling within the block  $j$ , and  $\text{color}(j)$  be the color of the  $j$ -th block. We denote by  $U_{j,k}$ ,  $k = 1, q$ ,  $j < k$  and  $L_{j,k}$ ,  $k < j$  the couplings between the  $j$ -th color block and the  $k$ -th block.

Then, we can describe the Multi-Color Block SSOR as follows.

### **ALGORITHM 0.2 Multi-Color Block SSOR**

*Let  $q$  be the total number of colors, and  $\text{color}(i)$ ,  $i = 1, L$ , be the array of the color for each block.*

- 1. Choose  $u_0$ , and  $\omega > 0$ .*
- 2. For  $i > 0$  Until Convergence Do*

3. For  $kolor = 1, q$  Do

4. For  $j = 1, L$  Do

5. if( $color(j) = kolor$ ) then

6.  $(u_{i+1/2})_j = D_j^{-1}(b - \omega * \sum_{k \neq kolor}^{k=q} L_{j,k} u_{i+1/2}).$

7. endif

8. Endfor

9. For  $kolor = 1, q$  Do

10. For  $j = 1, L$  Do

11. if( $color(j) = kolor$ ) then

12.  $(u_{i+1})_j = D_j^{-1}(u_{i+1/2} - \omega * \sum_{k \neq kolor}^{k=q} U_{j,k} u_{i+1}).$

13. *endif*

14. *Endfor*

15. *Endfor*

16. *Endfor*

Note that the innermost loop in line six and seven can be executed in parallel.

## Point-SSOR algorithm

---

**ALGORITHM 0.3** *Point-SSOR* Let  $A = D - E - F$ , where  $D$  is the diagonal part,  $-E$ , is the lowertriangular part and  $-F$  the uppertriangular part.

1. Choose  $x_0$ .

2. For  $i = 0, \dots$  Do

$$(D - \omega E)x_{i+\frac{1}{2}} = ((1 - \omega)D + \omega F)x_i + \omega b \quad (6)$$

$$(D - \omega F)x_{i+1} = ((1 - \omega)D + \omega E)x_{i+\frac{1}{2}} + \omega b$$

Endfor

## Experiments

---

Test problems

- **Problem 1** Poisson Equation on a Square

$$-\Delta u = f \quad (7)$$

$$\Omega = (0, 1) \times (0, 1)$$

$$u = 0 \text{ on } \delta\Omega$$

$$f = x(1-x) + y(1-y)$$

- **Problem 1** Elman's problem [3]

$$-(bu_x)_x - (cu_y)_y + f u = g \quad (8)$$

$$\Omega = (0, 1) \times (0, 1)$$

$$u = 0 \text{ on } \delta\Omega$$

where  $b = \exp(-xy)$ ,  $c = \exp(xy)$ ,  $f = \frac{1}{(1+xy)}$ ,

and  $g$  is such that exact solution  $u = x \exp(xy) \sin(\pi x) \sin(\pi y)$

## Results

---

- Tables 2 - 7 contain the timings for the cases with 4 preconditioners with various  $N$ . All of our test problems assume  $B = I$ .
- We used MPI(Message Passing Machine) library for the interprocessor communications.
- For the multi-coloring we have adopted the greedy heuristic as described in [12].
- We have used the *epsilon* parameter to be  $10^{-6}$  for stopping criterion. 'X' stands for the cases with insufficient memory. As for the  $\omega$  parameter we have set  $\omega$  to be 1.
- For the square domain we used the Block-Row mapping, i.e, that the domain is divided into  $p$  rectangle-shaped blocks, where  $p$  is the number of the available

processors. Further we assume that there is a one-to-one correspondence between the  $p$  blocks and  $p$  processors.

- For the inversion of diagonal blocks in Block SSOR method, we have used the MA48 routine of the Harwell library, which adopts direct methods for sparse matrices with the reordering strategy reducing fill-ins. The cost of the MA48 is roughly proportional to  $L^2$ , where  $L$  is the size of the matrix. Since  $L$  is roughly  $N/p$ , we expect a quadratic decrease with the increasing number of processors.
- The results show that for small number of processors Multi-Color ILU(0) shows the best performance, but for large number of processors the Multi-Color Block shows the best performance. In all cases, Multi-Color ILU(0) outperforms ILU(0) in the wavefront order and Point SSOR preconditioners. The reason that the performance of the Multi-Color Block SSOR improves with increasing number of processors is that the block inversion cost is inversely proportional

to  $p^2$ .

- For  $N = 128^2$  and  $256^2$  the CPUtime often increases as the number of processors increases. This is due to the communication overhead associated with the high number of processors.

Table 2: Problem 1 with FDM,  $N=128^2$ 

	p = 4	p = 8	p = 16	p = 32	p = 64
	CPU time				
ILU(0)/wavefront	1.6	0.8	0.8	1.1	1.6
ILU(0)/Multi-color	0.9	0.7	0.7	1.0	1.5
Point-SSOR	2.1	0.8	0.9	1.3	1.7
MC-Block SSOR	3.7	1.3	0.72	0.54	0.6

Table 3: Problem 1 with FDM,  $N=256^2$ 

	p = 4	p = 8	p = 16	p = 32	p = 64
	CPU time				
ILU(0)/wavefront	5.8	3.2	2.5	2.3	3.1
ILU(0)/Multi-color	4.9	3.1	2.3	2.0	2.8
Point-SSOR	7.5	3.9	3.1	2.8	3.8
MC-Block SSOR	27.2	10.7	5.0	2.2	1.4

Table 4: Problem 1 with FDM,  $N=512^2$ 

	p = 4	p = 8	p = 16	p = 32	p = 64
	CPU time				
ILU(0)/wavefront	X	X	12.5	8.6	8.2
ILU(0)/Multi-color	X	X	11.4	7.4	7.0
Point-SSOR	X	X	13.8	10.2	9.9
MC-Block SSOR	X	X	34.7	14.6	7.4

Table 5: Problem 2 with FDM,  $N=128^2$ 

	p = 4	p = 8	p = 16	p = 32	p = 64
	CPU time				
ILU(0)/wavefront	2.1	1.0	1.2	1.5	2.4
ILU(0)/Multi-color	0.9	1.0	1.2	1.4	2.2
Point-SSOR	2.8	1.2	1.4	1.7	2.7
MC-Block SSOR	4.3	1.6	1.0	0.7	0.8

Table 6: Problem 2 with FDM,  $N=256^2$ 

	p = 4	p = 8	p = 16	p = 32	p = 64
	CPU time				
ILU(0)/wavefront	9.3	5.1	4.1	3.8	5.1
ILU(0)/Multi-color	8.0	4.9	3.8	3.3	4.7
Point-SSOR	11.4	6.0	4.7	4.2	6.9
MC-Block SSOR	29.5	12.6	6.8	3.0	2.1

Table 7: Problem 2 with FDM,  $N=512^2$ 

	p = 4	p = 8	p = 16	p = 32	p = 64
	CPU time				
ILU(0)/wavefront	X	X	18.1	12.9	12.2
ILU(0)/Multi-color	X	X	16.9	11.1	10.7
Point-SSOR	X	X	20.0	14.6	14.4
MC-Block SSOR	X	X	47.7	19.7	10.6

## Conclusions

---

- For the problems tested Multi-Color BSSOR shows the best performance than the other preconditioners with high number of processors, while for small number of processors Multi-Color ILU(0) shows the best performance,
- Due to the nature of MA48 library, we expect MC-BSSOR to be *scalable* with the increasing number of processors.

## References

- [1] W. W. BRADBURY AND R. FLETCHER, *New iterative methods for the solution of the eigenproblem*, Numer. Math., 9(1966), pp. 259–267.
- [2] Y. CHO AND Y. K. YONG, *A multi-mesh, preconditioned conjugate gradient solver for eigenvalue problems in finite element models*, Comput. Struct., 58(1996), pp. 575–583.
- [3] H. ELMAN, *Iterative methods for large, sparse, nonsymmetric systems of linear equations*, Ph. D Thesis, Yale University, 1982
- [4] Y. T. FENG AND D. R. J. OWEN, *Conjugate gradient methods for solving the smallest eigenpair of large symmetric eigenvalue problems*, Internat. J. Numer. Methods Engrg., 39(1996), pp. 2209–2229.
- [5] G. GAMBOLATI, G. PINI, AND M. PUTTI, *Nested iterations for symmetric eigenproblems*, SIAM J. Sci. Comput., 16(1995), pp. 173.191.
- [6] G. GAMBOLATI, F. SARTORETTO, AND P. FLORIAN, *An orthogonal accelerated*

*deflation technique for large symmetric eigenproblems*, Comput. Methods Appl. Mech. Engrg., 94(1992), pp. 13–23.

- [7] C. -C. JAY KUO AND TONY CHAN, *Tow-color Fourier analysis of iterative algorithms for elliptic problems with red/black ordering*, SIAM J. Sci Stat, 11(1990), pp. 767-793.
- [8] SANGBACK MA, *Comparisons of the parallel preconditioners on the CRAY-T3E for large nonsymmetric linear systems*, International Journal of High Speed Computing, 10(1999), pp. 285-300.
- [9] J. A. MEIJERINK AND H. A. VAN DER VORST, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Mathematica Computation, Vol. 31, 1977, pp. 148-162
- [10] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.
- [11] A. RUHE, *Computation of eigenvalues and eigenvectors*, in Sparse Matrix Techniques, V. A. Baker, ed., Springer-Verlag, Berlin(1977), pp. 130.184.

- [12] Y. SAAD, *Highly parallel preconditioner for general sparse matrices*, in Recent Advances in Iterative Methods, IMA Volumes in Mathematics and its Applications, Vol. 60, G. Golub, M. Luskin and A. Greenbaum, eds, Springer-Verlag, Berlin, 1994, pp. 165-199.
- [13] Y. SAAD, *Krylov subspace methods on supercomputers*, SIAMJ. Sci Stat, 10(1989), pp. 1200-1232,
- [14] F. SARTORETTO, G. PINI AND G. GAMBOLATI, *Accelerated simultaneous iterations for large finite element eigenproblems*, J. Comput. Phys., 81(1989), pp. 53-69.
- [15] H. R. SCHWARZ, *Eigenvalue problems and preconditioning*, ISNM, 96(1991), pp. 191-208.