산업기술연구(강원대학교 선업기술연구소 논문집) 제17집, 1997
Research Reprot. The Institute of Industrial Technology, Kangwon Nat'l Univ., Korea, Vol.17, 1997

전자장문제를 위한 Davidson 방번의 병렬화

김 형 중* · 주 욱**

A Parallel Algorithm of Davidson Method for Solving and Electomagnetic Problem

Kim, Hyong Joong* · Zhu Yu**

ABSTRACT:

The analysis of eigenvalue and eigenvector is a crucial procedure for many electromagnetic computation problems. Although it is always the case in practice that only selected eigenpairs are needed, computation of eigenpair still seems to be a time-consuming task. In order to compute the eigenpair more quickly, there are two resorts: one is to select a good algorithm with care and another is to use parallelization technique to improve the speed of the computing. In this paper, one of the best eigensolver, the Davidson method, is parallelized on a cluster of workstations. We apply this scheme to a ridged waveguide design problem and obtain promising linear speedup and scalability.

1. INTRODUCTION

(1) *Waveguide Design Problem*: In electromagnetics, the analysis of eigenvalues and eigenvectors in both closed and open structures enables us to understand the performances of the structures. This kind of problem is common in designing the ridged waveguide. Ridged waveguides have many applications in microwave and antenna systems because of their unique characteristics of low cut-off frequency, wide bandwidth and low impedance compatible with coaxial cables [12]. The electric fields in a cylinderical waveguide satisfy the Helmholtz equation:[1)]

$$\nabla \times \frac{1}{\mu_r} \nabla \times \vec{E} - k_0^2 \varepsilon \vec{E} = 0 \quad (1)$$

$$\hat{n} \times \vec{E} = 0 \quad \text{on electric conductor} \quad (2)$$

$$\hat{n} \times \nabla \times \vec{E} = 0 \quad \text{on magnetic conductor} \quad (3)$$

The electric fields in the waveguide, then, can be expressed as:

$$\vec{E}(x, y, z) = [\hat{x} E_x(x, y) + \hat{y} E_y(x, y)] \, e^{j(\omega t - k_z z)} \quad (4)$$

Applying Galerkins method to (1) with (4) and vector identities, the final integral equation is:

$$\int (\nabla \times \vec{E}) \cdot (\nabla \times \vec{W}_i) \, ds = k_t^2 \int \vec{E} \cdot \vec{W}_i ds \quad (5)$$

where

$$k_t^2 = k_0^2 \varepsilon_r \mu_r - k_z^2 \quad (6)$$

and $\vec{W}$ denotes the shape function of edge based triangular element [4]. With this integral equation, the final eigenvalue problem to be solved for the

* 강원대학교 제어계측공학과 부교수
**강원대학교 제어계측공학과 석사과정
이 논문은 석제복합신소재 제품연구센터의연구비 지원에 의해 이루어졌음

analysis is obtained in the matrix form:nd

$$KE = \lambda ME \quad (7)$$

It is a generalized eigenvalue problem. However, (7) can be transformed into

$$A\,x = \lambda x \quad (8)$$

where A is a dense matrix and λ is an eigenvalue of an n×n matrix A, and x is its corresponding eigenvector. Then the problem is to solve the standard eigenvalue problem of (8). In addition, A is real and symmetric in this paper.

(2) Eigensystem Solver: There are several methods dealing with the eigenpair computation problems: Power method [7], Householders and Jacobis diagonalization method [6, 7], Lanczos method [6, 8], Arnoldis method [9, 10], and the Davidson methods [11, 10, 1, 2]. The latter method has been reported to be quite successful [1, 2]. Davidson method can be regarded as a preconditioned version of the Lanczos method. Although when used with a poor preconditioner it will take long time, the Davidson method may overcome the Lanczos method tremendously [1]. Moreover, it is suitable to be parallelized [2]. So it seems to be a promising method for eigenvalue problem in many applications including electromagnetic fields computation. The Davidsons algorithm that computes the largest or the smallest eigenvalue of the matrix A can be prescribed as follows [1, 2]:

Choose an initial unit vector v_1 : V_1 : = [v_1];
for k = 1, ... do
Compute the iteration matrix : $H_k = V_k^t A V_k$
Compute the largest (smallest) eigenpair (μ_k, y_k) of H_k ;
Compute the corresponding Ritz vector $x_k = V_k y_k$;
Compute the residual $r_k = (\mu_k I - A) x_k$;
If convergence then exit;
Compute the new direction to be incorporated $t_{k+1} = (\mu_k I - D)^{-1} r_k$
Orthogonalize the system [V_k: t_{k+1}] into V_{k+1};
end for

Fig. 1 : Davidsons Algorithm

(3) Hardware Configuration: The network-based computing with cluster of workstations recently has become a successful technique because it is inexpensive and scalable. In our experiment, three HP workstations, the C160s, connected by 10Mbps standard Ethernet, are employed to parallelize the Davidson method. However, our parallelization scheme can be applied to any bus-based multidrop configuration.

(4) PVM: The program is developed upon PVM, Parallel Virtual Machine [5]. The PVM is a middleware that permits a network of heterogeneous Unix computers to be used as a single large parallel computer. Under the PVM, a user-defined collection of computers appears as one large distributed-memory computer [5]. By sending and receiving messages, multiple tasks of an application can cooperate to solve a problem in parallel. The PVM is freely available, well designed, and not restricted to any specified type of machine so it is now widely used and becomes almost a standard for message-passing system developing tool like MPI (Message Passing Interface).

2. Parallelization Strategy

Two typical programming models in distributed memory system are SPMD (Single Program, Multiple Data) and Master/Slave [5]. Even though the Davidson method is an eigensystem solver, it computes eigenpairs indirectly from a very small $k \times k$ matrix H_k , the interaction matrix, rather than directly from the large-scale matrix A. The size of the interaction matrix H_k increases as the iteration number k increases. Thus, performance of the Davidson method may depend on the number of iterations. However, k is so smaller than n that the complexity of computing eigenvalue of H_k is trivial as long as A is dense.

In general, the eigensolver itself is difficult to parallelize. We apply the Householder transform to get a tridiagonal matrix, and use bisection algorithm to find a root of the characteristic polynomial. This procedure is still far from parallelization. In addition, the matrix H_k is not worth to be parallelized as long as k is small as mentioned before. Thus, it is better to employ the Master/Slave model in which the master solves this small eigenvalue problem while the salves take over the tasks of computing the most time consuming step: the matrix-vector multiplication and orthogonalization [2]. Our main idea is that every slave holds a part of the matrix A to execute the matrix-vector multiplication and the orthogonalization in parallel, which is the bottleneck of the Davidsons algorithm. Our algorithm is shown as in Figure 2 and 3, where the m stands for the number of the slaves, p stands for the n/m (for brevity, we assume is n/m an integer). In addition, the representation syntax in the Matlab [3] is used for easier understanding.

1. Initialize the working space, start the slaves;
for k = 1, ... **do** $H_k(k,:)=0$;
2. **for** i = 1 : m **do**
Receive the interaction matrices $H_{ki}(k,:)$ from slaves and add them together :
$H_k(k,:)= H_k(k,:)+ H_{ki}(k,:)$
$H_k(k,:)= H_k^T(k,:)$
3. Compute the largest eigenpair (μ_k, y) of H_k
4. If convergence then exit ;
5. Broadcast the to the slaves (μ_k, y) ;
end for k;

Fig. 2: Algorithm of the Master

1. Read submatrices
$A_q = A(qp:(q+1)p-1,:)$from the disk, get part of the diagnal of A :
2. $d_q = D(qp:(q+1)p-1)$ where D stands for the diagnal of A;
While not finished **do**
3. Compute, $b_k = A_q V_k(:,k)$, and $B_k = [B_{k-1}: b_k]_{p\times k}$;
4. Compute
$H_{kq}(k,:) = V_k^t(q:q+p+1,:)\, b_k$ send it to master;
5. Receive the largest eigenpair (μ_k, y_k) from master ;
6. Compute part of the new direction to be incoporated
$t_{kq} =$
$(B_k - \mu V_k(q:q+p-1,:))\, y_k ./ (d_q - \mu)$
7. **For** i = 1 : k
compute the coefficients to be used in the orthogonalization
$r_q(i) = \langle t_{kq}, V_k(q:q+p-1,i)\rangle$
8. Broadcast the $r(i)$ to the other slaves ;
9. **For** i = 1 : m - 1 **do** receive r_i from other slaves and $r_q = r_q + r_i$;
10. **for** i = 1 : k **do**
$t_{kq} = t_{kq} - r_i(i)\, V_k(q:q+p-1,i)$;
11. Broadcast the t_{kq} to other slaves ;
12. **for** i = 1 : m-1 **do** receive t_{ki} except t_{kq} from other slaves and construct the whole new direction vector t_{k+1} ;
13. Normalize the $t_{k+1} = t_{k+1}/\| t_{k+1}\|$ and $V_{k+1} = [V_k : t_{k+1}]$;
end while

Fig. 3: Algorithm of the qth Slave

3. Experimental Results

A. Dense Matrix

In our experiments, we first apply our algorithm

to a real world eigenvalue problem derived from a ridged waveguides design (we denote it stiff below) [12]. The resultant matrix is a 677×677 dense (see Table I). Total number of iterations is just 39. Thus, the number of operations for computing eigenvalue of H_k is far less than that of matrix-vector multiplication, which justifies the Master/Slave model

Table I: Description of the Matrix from the Ridged Waveguide Design

Name	Matrix Resource	Dimension	Density	# of Iteration	Maximum Eigenvalue
stiff	See [12]	677	full	39	7.641128e+07

Table II: Timing Data of the Matrix Stiff (Timing Unit: Seconds)

# of Workstations	Timing	Speedup
1	18.689	1.0000
2	9.4964	1.9680
3	6.4958	2.8771

From Table II, it is clear that the speedup is almost linear and scalable when A is dense.

The speedup of our scheme is satisfactory which is 1.968 for 2 machines and 2.877 for 3 machines. At the same time its convergence profile is shown in Figure 1. The convergence speed is very fast which supports the claim that Davidson method provides a second order convergence near the solution [2]. In addition, we compare the Davidson method with the Power method. Figure 1 shows the convergence profile of the Power method. Its convergence speed is very slow. Of course, its parallelization is straightforward. The power method relies mainly on the matrix-vector multiplications which is easy to parallelize. It is obvious the Davidson method beats the Power method in terms of the convergence rate as well as the computing time.

B. Sparse Matrix

Then, can we get the same promising results with the sparse matrices? To answer this question, we select the artificial sparse test matrix set as in [1] given below:

$$a_{ij} = \begin{cases} \text{if } i=j \text{ normally distribued}: N(0, 5^2) \\ \text{if } i \neq j \begin{cases} \text{with probability } \alpha: N(0,1), \\ \text{with probability } (1-\alpha): 0. \end{cases} \end{cases} \quad (9)$$

Computation stops when

$$| \mu_k - \mu_{k-1} | < 10^{-11},$$

where the parameter μ_k is the convergent eigenvalue at the kth iteration. The parameter α is approximately the density of the matrix which means the ratio α =(number of non-zero elements) / (n×n).

Table III: Description of the Testing Matrices

Name	Matrix Resource	Dimension	Density	# of Iteration	Maximum Eigenvalue
test1	Eq. (3)	8,000	0.0002	27	19.64068
test2	Eq. (3)	8,000	0.0010	66	23.15136
test3	Eq. (3)	8,000	0.0200	77	28.64236
test4	Eq. (3)	12,000	0.0002	31	19.82698
test5	Eq. (3)	20,000	0.0002	23	23.91715

Table IV: Timing Data of the Testing Matrices Above (Unit: Seconds)

# of Workstations	test1	test2	test3	test4	test5
1	41.394	117.57	227.41	26.330	40.997
2	26.850	61.400	114.92	14.754	23.409
3	20.718	68.339	106.03	13.763	31.800

Table V: Speed-up Performance of Our Parallelization Scheme

# of Workstations	test1	test2	test3	test4	test5
1	1.0000	1.0000	1.0000	1.0000	1.0000
2	1.5416	1.9149	1.9788	1.7843	1.7513
3	1.9980	1.7205	2.1447	1.9128	1.2903

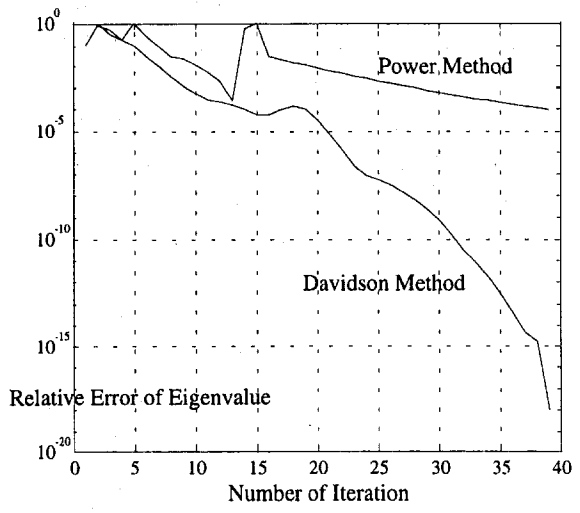

Fig. 4: The iteration profile of the Davidson method and the Power Method

From the experiment results in Table IV and V, we find that the speedup performance is not as good as on the dense matrix when A is sparse. The speed-up is in between 1.5 and 1.98 using two computers. The speed-up may not be satisfactory for some sparse problems surprisingly when three computers are engaged. From the speed-up results of test1, test2 and test3 in Table III we find that the speed-up increases radically as the number of non-zero elements increases. Results of test1 and test4 show that when the dimension of the problem increases, speed-up increases obviously. However, test5 shows that speed-up with three computers can be lower than with two machines. We can account for it below.

Frst we examine the multiplication flops needed in the computing steps as given in Table VI. Now we are able to estimate roughly the theoretical speedup of our parallelization scheme. The diagonalization may be estimated approximately $2\,k^3\,l^3$ flops and the computation of the H_k is estimated at $(kn+\alpha\,n^2)/m$. Note that due to the difficulty in estimating the communication time during the computation the result can only demonstrate the relationship between the speedup and the density of the matrix of interest rather than the genuine data. Figure 5 shows the relationship between speed-up and number of iterations of matrices with different α .

It is obvious that the speedup becomes bad when the density of the matrix decreases. This helps explain the experimental results of the unsatisfactory speedup stated above.

Table VI: Multiplications needed in the Davidson Method of the kth Iteration on the Single Machine and on m Machines:

Computation Step	Mutiplications on One Machine	Mutiplications on m Machines
Computation of H_k	$kn + O(\alpha n^2)$	$(kn + O(\alpha n^2))/n$
Diagonalization of H_k	$O(k^3 l^3)$	$O(k^3 l^3)$
Computation of the Ritz vectors	kn	kn / m
Computation of the residuals	kn	kn / m
Orthogonalization process	$2kn$	$2kn / m$

Note: Here denotes the density if the matrix of interest

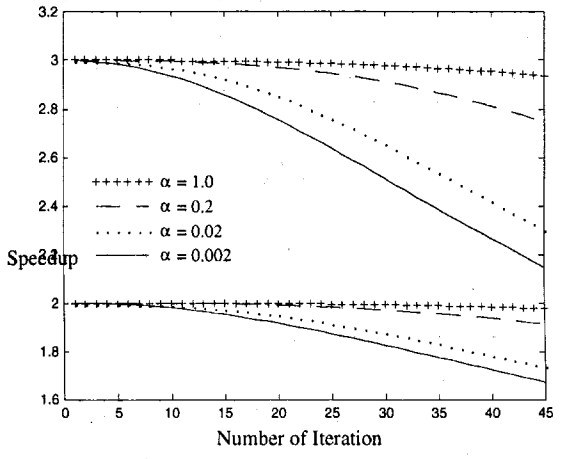

Fig. 5: The estimation of speedup vs. # of iterations of a 2000×2000 matrices with various densities α

4. Conclusion and Future Works

We present a parallel algorithm of the Davidson method on a cluster of workstations. We apply the algorithm solving a ridged waveguide design problem and get a linear speedup and scalability. It is totally due to the denseness of the resultant matrix. However, speedup is not satisfactory when a matrix is sparse.

As a result it is recommended that when using

our algorithm in the same computing environment described above the result may not be satisfactory with high sparsity. However, our method works successfully on a dense matrix and the speed-up as well as the convergence rate are attractive. Obviously, the boundary element methods produce dense matrices. Therefore, our method will have a wide use in the application area.

In addition we want to mention that the configuration stated above is not ideal for parallel computing since the network is too slow compared with the high-end computers. The future works will include the parallelization of the generalized eigenvalue algorithms and the Davidson algorithm for non-symmetric matrices.

Acknowledgment

The authors would like to thank Mr. Seung Woo Lee who provided the real-world matrix from the ridged waveguide design problem. We appreciate anonymous reviewers whose comments and suggestions are constructive and helpful.

References

1. M. Crouzeix, B. Philippe, and M. Sadkane, "The Davidson Method", *SIAM Journal of Scientific Computing*, vol. 15, pp. 62-76, January 1994.
2. A. Stathopoulos, and C.F. Fischer A "Davidson program for finding a few selected extreme eigenpairs", *Computer Physics Communication,* vol. 77, pp 269-290, 1993.
3. *Matlab Users Guide*, The Mathworks Inc., August 1992
4. J. Jin, *The Finite Element Method in Electromagnetics*, John Wiley & Sons, 1993.
5. Al Geist *et al., PVM 3 Users Guide and Reference Manual,* Oak Ridge National Laboratory, Tennessee, 1994.
6. B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, 1980.
7. G. H. Golub, and C. F. Van Loan, *Matrix Computations*, 2nd ed., The John Hopkins University Press, 1989.
8. C. Lanczos, "An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators", *Journal of Research of the National Bureau of Standards*, vol. 45, pp. 255-282, 1950.
9. W. E. Arnoldi, "The Principle of Minimized Iterations in the Solution of the Matrix Eigenvalue Problem", *Quarterly of Applied Mathematics*, vol. 9, pp. 17-29, 1951.
10. Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, Manchester, UK, 1992.
11. E. R. Davidson, "The Iterative Calculation of a Few of the Lowest Eigenvalues and Corresponding Eigenvectors of Large Real Symmetric Matrices", *Journal of Computational Physics.*, vol. 17, pp. 87-94, 1975.
12. H.-b. Lee *et al.*, "An Optimum Design Method for Eigenvalue Problems", *IEEE Transactions on Magnetics*, vol. 32, pp. 1246-1249, May 1996.