AIAA JOURNAL
Vol. 37, No. 8, August 1999

Multifrontal Solver Combined with Graph Partitioners

Jeong Ho Kim* and Seung Jo Kim†
Seoul National University, Seoul 151-742, Republic of Korea

For large-scale structural analysis, the performance of a linear equation solver is very important for the overall efficiency of the analysis code. The multifrontal solver is a very efficient direct solver for finite element analysis. By using multiple fronts, it can considerably reduce the computing time spent on solving the system of linear equations arising from finite element analysis. To achieve good performance using the multifrontal solver, a good front partition must be obtained because the performance largely depends on the quality of the front partition, that is, the number of degrees of freedom on the partitioned fronts. In this study, graph-partitioning algorithms that are generally used to decompose a given domain for parallel computation are combined with the multifrontal solver to obtain good front partitions of irregular (unstructured) meshes. The influence of the partitioning quality on the performance of the multifrontal solver is also examined. For regular (structured) meshes, the multifrontal scheme can solve the system of linear equations much more efficiently than the single frontal scheme with the help of a simple front-partitioning algorithm. For large-scale problems with irregular meshes such as the finite element meshes of aerospace structures, the verification was made that the developed multifrontal solver combined with an efficient graph partitioner (Metis) and an appropriate mesh mapping scheme (weighted-edge mapping) shows very good performance.

Introduction

LARGE-SCALE finite element analyses are often required in the structural design and certification of large and complex structures such as aerospace vehicles. In this case most of the computation time is spent on solving the system of algebraic equations associated with the finite element model. To reduce the overall computation time or computational cost, the development of efficient equation solvers is important. Various types of equation solvers have been developed for finite element analysis.[1–4] They can be classified into two categories: direct and iterative solvers. Some types of iterative solvers show better performances than direct solvers for large-scale problems. However, direct solvers have many advantages over iterative solvers, such as numerical robustness. Direct solvers are also preferred for systems of linear equations with multiple right-hand sides. Furthermore, direct solvers can be used much more efficiently for solving local problems arising from parallel implementation of the finite element method using substructuring or the domain decomposition technique.[5]

For the classical direct solvers, there is room for performance improvement: the reduction of fill-ins. The fill-ins are zero entries of the original coefficient matrix, which become nonzero entries after factorization. Matrices arising from finite element analyses are sparse matrices, and consequently the fill-ins inevitably result from the factorization process of direct solvers. Thus, by exploiting the sparsity, the fill-ins or unnecessary operations can be reduced, and the performance of direct solvers can be improved. Band solvers, skyline solvers, and frontal solvers[2] are designed to take advantage of the sparsity of the matrices, and better performance of these solvers can be achieved by the reduced fill-ins. Node or element reordering and an appropriate solution procedure can reduce the fill-ins further.

In this study the multifrontal solver[1,6] that reduces fill-ins by using multiple fronts to solve a system of linear equations arising from finite element analysis has been extended to solve the problems with irregular (unstructured) finite element meshes efficiently by combining the solver with graph partitioners.[5,7–11] The efficiency of the multifrontal solver with respect to the number of fronts is shown, and the influence of front partitioning on the performance of the multifrontal solver is examined.

Presented as Paper 98-1762 at the AIAA/ASME/ASCE/AHS/ASC 39th Structures, Structural Dynamics, and Materials Conference, Long Beach, CA, 20–23 April 1998; received 12 June 1998; revision received 30 December 1998; accepted for publication 9 January 1999.

*Research Assistant, Department of Aerospace Engineering, San 56-1, Shillim-Dong, Kwanak-Ku.

†Professor, Department of Aerospace Engineering, San 56-1, Shillim-Dong, Kwanak-Ku. Member AIAA.

Multifrontal Solver

The concept of the multifrontal method was first introduced by Duff and Reid[6] as a generalization of the frontal method of Irons.[2] The frontal method was originally proposed as a solution procedure for the finite element method and has earned the reputation of being easy and inexpensive to use. The frontal method can be considered as a particular technique for first assembling element stiffness matrices and element nodal force vectors and then solving for the unknown displacements by means of a Gaussian elimination and backsubstitution process in the element-by-element manner.[12] The method is designed to minimize core storage requirements because it does not require the global assembly of the stiffness matrix and force vector. The main idea of the frontal solution is to eliminate the variable while assembling the equations. As soon as the coefficients of an equation are completely assembled from the contributions of all relevant elements, the corresponding variables can be eliminated. Therefore, the complete structural stiffness matrix is never formed as such because after elimination the reduced equation is immediately transferred to backup disk storage. The elimination process of the frontal method is illustrated in Fig. 1.

In their paper Duff and Reid[6] used the concept of generalized elements to deal with general indefinite sparse matrices. They proposed combining a few different fronts or generalized elements to have larger fronts for better choices of pivoting. Various advances in the multifrontal method for the general sparse matrix solution have been achieved by many researchers, and the advances were surveyed by Liu.[12] Geng et al.[1] applied the multifrontal solver to finite element analysis by combining the method with the nested-dissection ordering[13] and showed the efficiency. The elimination process of the multifrontal method applied to the finite element procedure is illustrated in Fig. 2.

The difference between the elimination process of the single-frontal method and that of the multifrontal method can be understood by comparing Fig. 1 with Fig. 2. By the single-frontal method, one front is used and spreads out all over the whole domain as fully assembled degrees of freedom (DOFs) are eliminated from the front. By the multifrontal method, a given domain is divided into two subdomains recursively, and each subdomain has its own fronts. Then, after the recursive bisection of the given domain is

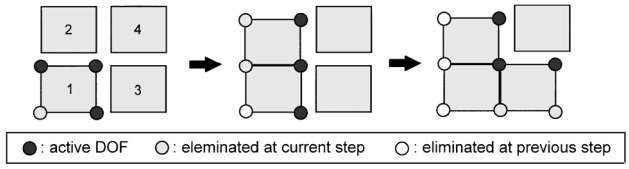


Fig. 1 Elimination process of the frontal method.

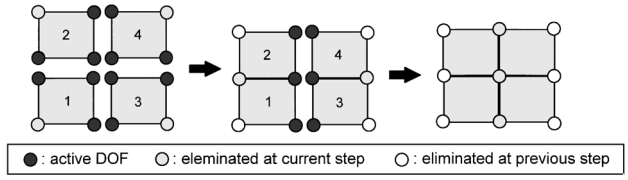


Fig. 2 Elimination process of the multifrontal method.

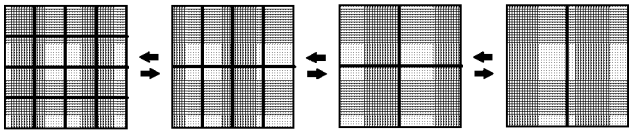


Fig. 3 Front partitioning and merging.

completed, the internal DOFs of each subdomain are eliminated first by the frontal method. The remaining interface DOFs of each subdomain after eliminating internal DOFs become new fronts, and they are merged with each other recursively in the reverse order by which the given domain is divided as shown in Fig. 3. At each merging stage, fully assembled DOFs are eliminated immediately.

By this procedure the interface DOFs of each subdomain are eliminated in the same order as in the nested dissection method. The multifrontal solver implemented by Geng et al.[1] shows very good performance for regular (structured) meshes. Geng et al.[1] estimated the operational counts for factorization of the stiffness matrix constructed by the finite element procedure for a two-dimensional regular mesh of size N ($\sqrt{N} \times \sqrt{N}$ elements) as follows:

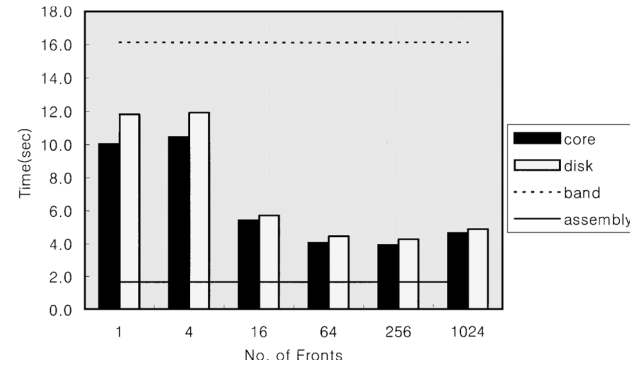
$$(56/3P)N^2$$

where P is the number of fronts and N, P, and N/P are assumed to be large enough. According to the preceding estimation, better performance can be obtained as the number of fronts is increased. However, because this estimation approximately holds only for large N/P, the performance of the multifrontal solver is not improved any more for the value of P larger than a certain number.

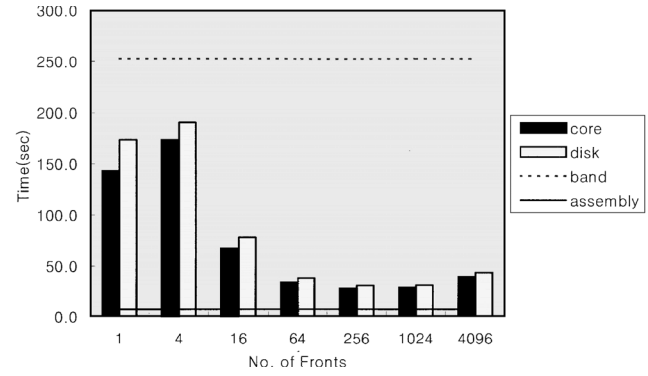
In this study, first, the performance of the multifrontal solver with respect to the number of fronts is tested for simple elasticity problems with regular meshes. They are two-dimensional problems with 64×64 and 128×128 four-noded plane stress elements and a three-dimensional problem with $16 \times 16 \times 16$ eight-noded solid elements. The computing time (CPU time: time used by CPU) and the required memory size are examined. A system that has Pentium Pro 200-MHz CPU and 128 MB of main memory is used. The system runs the Linux operating system and GNU C/C++ compiler.

The results are shown in Figs. 4–7. We have implemented two versions of the multifrontal solver. The core version uses core memory to keep the matrix coefficients related to the eliminated DOFs, and the disk version uses a hard disk to do the same thing. In Figs. 4–7, the versions are distinguished by the labels "core" and "disk." The results are also compared with those of a band solver. As shown in Figs. 4–7, the performance of the multifrontal solver with one front (which is identical to the conventional frontal solver) is superior to that of a band solver and can be further improved by increasing the number of fronts to the appropriate number. For the problems in Fig. 4, the best performance (the shortest computing time) is obtained when the number of fronts is equal to 256. The solution time for the 64×64 finite element mesh with 256 fronts is about 2.6 times faster than that of the conventional frontal solver, as shown in Fig. 4a.

For problems of larger size, the performance of the multifrontal solver is much better than that of the single-frontal method. As



a) 64 × 64 four-noded elements



b) 128 × 128 four-noded elements

Fig. 4 CPU time vs the number of fronts for plane-stress problem.

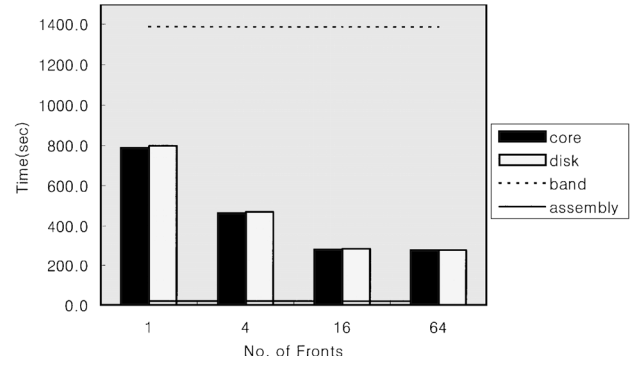


Fig. 5 CPU time vs the number of fronts for 16 × 16 × 16 solid elements.

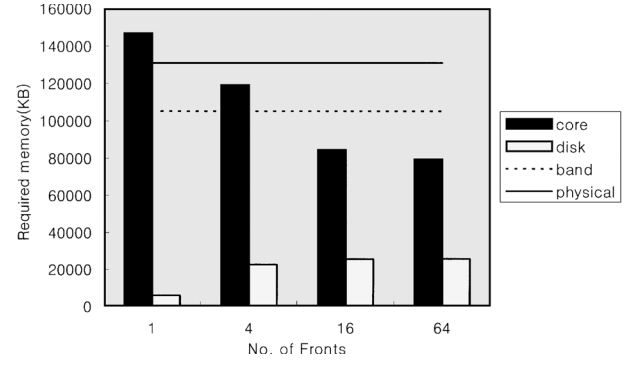


Fig. 6 Memory usage for 16 × 16 × 16 solid elements.

shown in Fig. 4b, the solution time with 256 fronts is about 5.2 times faster than that with single front.

Similar results are obtained for a three-dimensional elasticity problem with a regular mesh, as shown in Fig. 5. The best performance is obtained when 64 fronts are used and is about 2.8 times faster than the performance with a single front.

The total solution time includes the time spent on computing element stiffness matrices, which is constant as the number of fronts

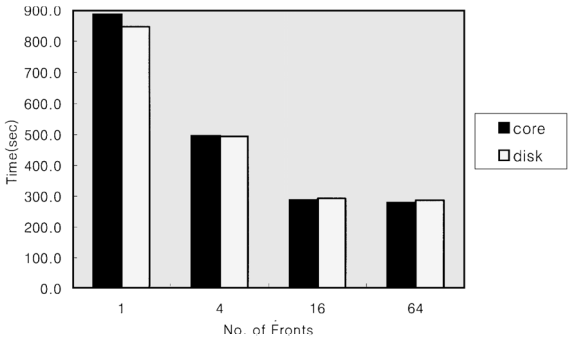


Fig. 7 Elapsed time vs the number of fronts for 16 × 16 × 16 solid elements.

is varied. This is indicated by the straight lines in Figs. 4 and 5. Thus, the pure equation solution time in each case is equivalent to the portion above the line.

Comparing the core version and the disk version in Figs. 4 and 5, the core version shows slightly better performance. However, as shown in Fig. 6, the memory usage of the disk version is much less than that of the core version or the band solver. Thus, the disk version can solve very large-scale problems that the core version and/or the band solver cannot possibly solve. Even if they can be solved by the core version using virtual memory (disk-swap space), the actual elapsed time during the solution procedure of the disk version can be less than that of the core version. The single-front case in Fig. 7 shows this situation. For the single-front case the required memory size of the core version is greater than the physical memory size of the system used. Thus, the core version uses virtual memory. In this case the elapsed time for the disk version to obtain the solution is less than that for the core version.

For regular meshes such as the preceding problems, the optimal front partition is straightforward, that is, the partition obtained by dividing the mesh into two equal-sized parts recursively by the plane normal to the longest direction of the mesh. All of the results in this section were obtained by using this partition. In the next section the application of the multifrontal solver to irregular meshes is discussed.

Combination of the Multifrontal Solver with Graph Partitioners

The preceding section showed that the multifrontal solver is an efficient direct solver for finite element analysis. However, the efficiency of the solver is largely dependent on the quality of the front partition (or the number of DOFs on the partitioned fronts). For the problems with regular meshes considered in the preceding section and those tested by Geng et al.,[1] any specific front-partitioning algorithm was not considered because the optimal front partition could be obtained easily for regular meshes by the simple procedure mentioned in the preceding section. But, in general, most of the real-world (not academic) problems to be solved by the finite element method have irregular meshes, and finding the optimal front partition of these irregular meshes is not so simple. Therefore, appropriate partitioning information must be furnished to the multifrontal solver by some efficient front-partitioning algorithms that can deal with general meshes. In other words, the procedure that divides the global mesh into two parts recursively is required to use the multifrontal solver generally and efficiently. This procedure is discussed in this section.

After the elimination of the internal DOFs in the partitioned fronts, the multifrontal solver merges neighboring fronts recursively in the reverse order by which they are partitioned, and then fully assembled DOFs on merged fronts are eliminated at each merging step. If the size of the merging fronts at one merging step is F, then the operational count to eliminate the DOFs at this step is about $\mathcal{O}(F^3)$. Because this procedure is repeated recursively, the size of the merging fronts at each step greatly affects the performance of the multifrontal solver. Therefore, the size must be kept as small as possible in the front-partitioning step to achieve good performance with the same number of partitioned fronts. Front

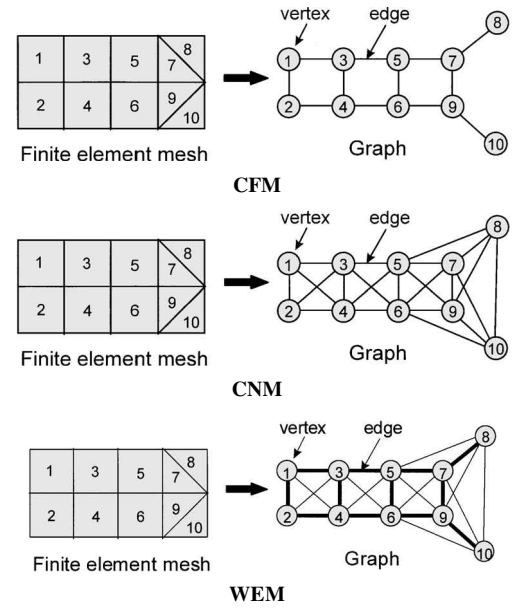


Fig. 8 Finite element mapping schemes.

partitioning divides a given mesh into two equal-sized parts recursively and is partly analogous to the mesh-partitioning procedure that appears in the domain decomposition method,[5] one of the popular schemes to solve the elliptic boundary-value problems in parallel computing systems. Thus, graph or mesh partitioning algorithms[5,7–11] that have been generally used to decompose the problem domain to realize the parallelism could be used for front partitioning.

Many mesh-partitioning algorithms are based on graph-partitioning algorithms, and finite element meshes to be partitioned are converted to graph structures to obtain mesh partitions by using graph-partitioning algorithms. Most graph-partitioning algorithms recursively bisect the original graph to obtain the desired number of partitioned subgraphs. This procedure can be directly applied to front partitioning. To apply this procedure to front partitioning, the finite element mesh must be mapped to a graph structure, and this graph structure will be partitioned by graph-partitioning algorithms. Three ways to map the finite element mesh to a graph structure are considered. They are named *common-face mapping* (CFM), *common-node mapping* (CNM), and *weighted-edge mapping* (WEM), as illustrated by Fig. 8. Each element of the finite element mesh is mapped to a vertex of the graph by all three mapping schemes. The difference among them is the way by which the edges of the graph are constructed. The common-face mapping scheme constructs an edge of weight 1 between two elements that have a common line (for a two-dimensional mesh) or a common face (for a three-dimensional mesh). The common-node mapping scheme constructs an edge of weight 1 between any two elements that have at least one common node. The third scheme, WEM, is proposed in this paper to implement the connectivity of finite elements better for front partitioning. In this scheme the concept of weight is introduced to the edges of a graph, and the weight of the edge between two neighboring elements is set to the number of common nodes. For example, the thick lines in Fig. 8 indicate the edges of weight 2. The CFM and CNM schemes have been used for mesh partitioning.[9,14] To describe the communication pattern better for parallel computation, Venkatakrishnan et al.[14] proposed the use of the CNM scheme that was called a communication graph of the given finite element mesh. The effect of the mapping schemes on front partitioning may be varied according to the graph-partitioning algorithm and the finite elements used and will be discussed in the next section.

After a graph is obtained from a finite element mesh, the graph is partitioned using graph-partitioning algorithms, and then the partitioning information of the graph is used to obtain a front partition for the multifrontal solver. Front partitioning is different from graph or mesh partitioning for parallel computation in that the optimal front

partition can be found if the optimal bisection is obtained at every recursive bisection step, whereas the optimal mesh partition may not be found. That is, front partitioning is the same process as recursive bisection, whereas mesh partitioning only uses the recursive bisection scheme to obtain the desired number of partitions.[10,11]

To implement front partitioners, graph-partitioning algorithms based on the recursive bisection scheme were used. In this research, as a state-of-the-art graph partitioner, Metis[7,8,11] was used. By using the graph-partitioning routine provided by Metis, bisection of the given graph is performed recursively. Metis implements the multilevel graph-partitioning scheme, which is quickly compared with other algorithms and provides high-quality partitions. Metis provides the k-way partitioning scheme, which means that the given graph is partitioned into the desired number of subgraphs at once, in contrast with recursive bisection, and the resulting partition may be better for parallel computation. However, as already mentioned, the k-way scheme cannot be used for front partitioning. Therefore, the graph-partitioning routine provided by Metis was used only to divide a graph into two pieces. As another graph-partitioning scheme, the recursive spectral bisection (RSB)[6,9] scheme that had been commonly used before the introduction of Metis also has been implemented in the multifrontal solver to evaluate the performance with respect to the front-partitioning schemes.

Effect of Front Partition

The multifrontal solver has been extended to be able to deal with irregular finite element meshes efficiently by the combination with graph partitioners and mesh-mapping schemes. The effect of front-partitioning quality on the performance of the developed multifrontal solver is examined in this section.

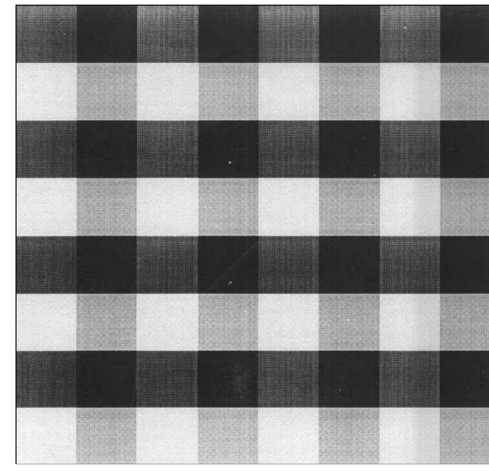
Let us begin with the earlier regular mesh example, which has 128×128 four-noded plane stress elements. Though the optimal front partition of regular meshes such as this example could be easily obtained as already explained, front partitioning of the mesh was carried out using the front-partitioning procedures proposed in this study to compare the performance with the optimal case. Because the three mesh-mapping schemes described in the preceding section make little difference in the performance of the multifrontal solver for this case, only the results using the WEM scheme are presented. With the graph mapped by the WEM scheme, graph partitioning into 64 parts is performed using two different graph-partitioning schemes. The resulting front partitions are shown in Fig. 9, with the front partition obtained by simply dividing the mesh according to the element numbering without considering any particular graph-partitioning schemes, which is labeled "Bad partition." The partition labeled "RSB" has been obtained using the RSB algorithm, and the partition labeled "Metis" has been obtained using the graph-partitioning routine provided by Metis at each recursive bisection step.

The performance results are shown in Fig. 10. All of the results presented in this section are relative computing times, which means the computing time divided by the solution time using a single front. That is,

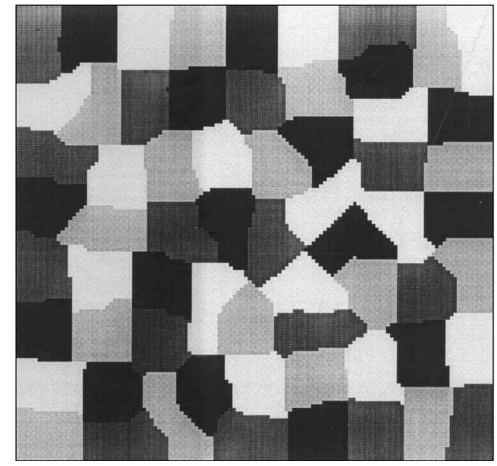
$$t_r = t/t_0$$

where t_r is relative computing time, t is computing time, and t_0 is solution time using a single front.

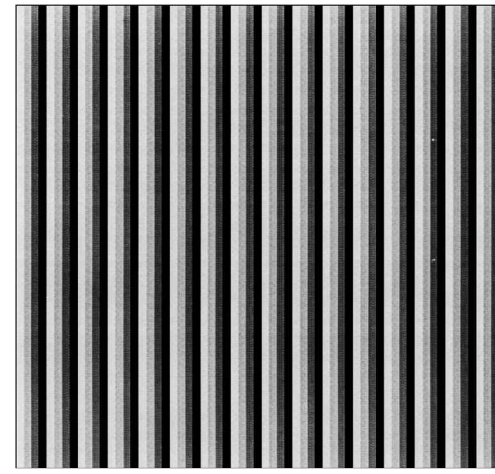
In Fig. 10 the time spent on the equation solution procedure is called "Solve," and the time spent on partitioning the graph is called "Partition." The RSB algorithm gives a partition of better quality than Metis in this case (not always). Thus, the multifrontal solver with the RSB graph partitioner spends less time on the equation solution procedure than that with the Metis graph partitioner, as shown in Fig. 10. However, because the RSB algorithm spends too much time on partitioning the graph, the overall performance (total computing time) of the multifrontal solver combined with that of the RSB algorithm is worse even than the performance of the single-frontal method as well as that of the multifrontal solver combined with Metis. The performance with the front partition labeled "Bad partition" is extremely bad, as shown in Fig. 10, and multiple fronts only make the performance worse in this case. Thus, the quality of a front partition greatly affects the performance of the multifrontal solver, and it is necessary to use appropriate graph partitioners to achieve good performance for general finite element problems.



RSB



Metis



Bad partition

Fig. 9 Configuration of partitioned mesh.

Three-dimensional tests have been performed using $16 \times 16 \times 16$ eight-noded solid elements and $8 \times 8 \times 16$ 18-noded solid elements. An 18-noded element has two nodes in the thickness direction, as shown in Fig. 11. For each type of element, the influence of partitioning schemes and mesh-mapping schemes on the performance is observed. Tables 1 and 2 show the results obtained by using $16 \times 16 \times 16$ eight-noded solid elements and $8 \times 8 \times 16$ 18-noded solid elements, respectively. The values in Tables 1 and 2 are relative computing times just defined. The computing times for these

Table 1 Relative computing times for 16 × 16 × 16 eight-noded solid elements

Mapping scheme	RSB		Metis	
	Partition	Total	Partition	Total
CFM	0.008	0.358	0.004	0.356
CNM	0.013	0.357	0.005	0.456
WEM	0.012	0.357	0.005	0.351

Table 2 Relative computing times for 8 × 8 × 16 18-noded solid elements

Mapping scheme	RSB		Metis	
	Partition	Total	Partition	Total
CFM	0.001	0.637	0.001	0.461
CNM	0.002	0.638	0.001	0.432
WEM	0.002	0.355	0.002	0.352

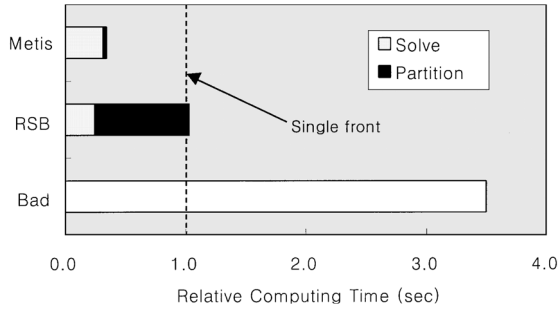


Fig. 10 Performance vs the partitioning schemes.

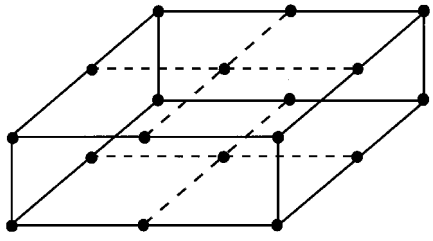


Fig. 11 Eighteen-noded solid element.

problems with a single front are 783.0 and 803.9 s, respectively. Compared with two-dimensional problems, the time portion of front partitioning is very small for a three-dimensional case, and the partitioning time has little effect on the overall performance of the solver.

According to the results using 16 × 16 × 16 eight-noded solid elements shown in Table 1, the RSB algorithm is not sensitive to the mesh-mapping scheme for this case, whereas Metis is. When the CNM scheme is used, the result obtained by Metis shows poor performance compared with those of the other cases. The CNM scheme cannot fully reflect the connectivity information of the given three-dimensional finite element mesh on the graph, and consequently the quality of the front partition obtained by using the graph may not be so good.

For 18-noded elements, which are generally used to model composite laminates for three-dimensional, lamina-wise finite element analysis, the effect of mesh-mapping schemes on the overall performance of the multifrontal solver becomes more significant. As shown in Table 2, the WEM scheme shows excellent performance for both RSB and Metis, and the performance with the WEM scheme is much better than that with the other mapping schemes. The mesh-mapping scheme greatly affects the performance of the multifrontal solver in this case. This result is caused by the geometric characteristics of the 18-noded element. An 18-noded solid element has 9 nodes on the upper and lower face and 6 nodes on four side faces, as shown in Fig. 11. The CFM scheme or the CNM scheme cannot manipulate the directional inhomogeneity, whereas the WEM scheme can deal with it by the weight of the corresponding edges. Thus, the WEM scheme can be used as a reliable mapping scheme for various types of elements.

Performance with Irregular Meshes

In the preceding section, the effect of front partitioning on the performance of the multifrontal solver was examined for problems with regular meshes, and both RSB and Metis with the WEM scheme produce good front partitions for three-dimensional problems.

In this section, by using the multifrontal solver of the disk version, large-scale, three-dimensional finite element analyses are performed for irregular meshes such as the finite element mesh of the thick composite plate with two holes (Fig. 12) and the composite shell with cutouts (Fig. 13).

The total DOFs of the mesh shown in Fig. 12 are 34,155, and there are 9088 eight-noded, three-dimensional composite elements in the mesh. Memory usage for the best performance of the solver is about 35 MB. For the single-frontal method element ordering greatly affects the performance. Therefore, the performance results that are obtained with and without element reordering are compared. The spectral-element reordering algorithm,[15] which is stable and produces high quality of ordering, is used. As a mesh-mapping scheme, the WEM scheme is used. The results are shown in Table 3, where the best performance with 512 fronts is 1.8 times faster than that of the single-frontal method with element reordering. For the multifrontal solver with a large enough number of fronts to achieve the best performance, the front-partitioning procedure itself can take the role of reordering for the single-frontal method, and element reordering in each partitioned front is no longer required. However, element reordering is positively necessary to obtain good and stable

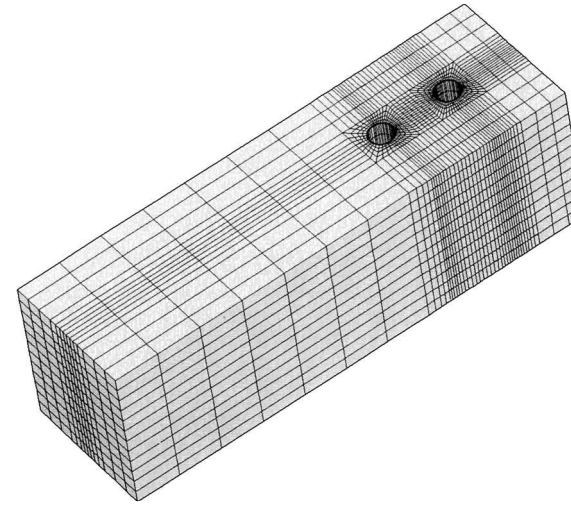


Fig. 12 Finite element mesh of thick composite plate with two holes (thickness exaggerated 10 times).

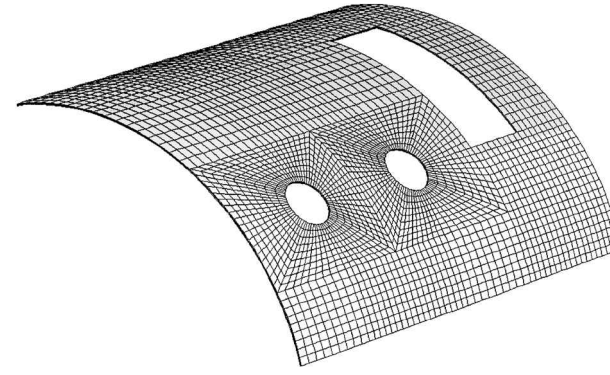


Fig. 13 Finite element mesh of a three-dimensional composite shell with cutouts.

Table 3 Effect of element reordering (CPU time in seconds)

Number of fronts	With element reordering: Partition + reorder	With element reordering: Total	Without element reordering: Partition	Without element reordering: Total
1	17.78	1158	N/A[a]	N/A
256	8.64	702.7	7.0	698.8
512	9.44	656.8	7.62	651.7
1024	10.12	657.7	8.32	656.2

[a]Too much disk space and computing time required.

Table 4 Performance of the multifrontal solver vs the single-front solver for the three-dimensional composite shell structure with four layers

Solution method	Reorder or partition, s	Total solution time, s	Memory usage, MB	Disk usage, MB
Single front	30.1	2822	12	482
RSB–WEM	64.0	913	33	217
RSB–CFM	46.6	1530	42	302
Metis–WEM	10.2	664	31	185
Metis–CFM	8.8	689	32	187

Table 5 Performance of the multifrontal solver vs the single frontal solver for the three-dimensional composite shell structure with eight layers

Solution method	Reorder or partition, s	Total solution time, s	Memory usage, MB	Disk usage, MB
Single front	91.9	16,180	30	1563
RSB–WEM	223.8	4,759	105	644
RSB–CFM	179.7	7,722	130	878
Metis–WEM	23.2	3,063	77	520
Metis–CFM	18.8	3,306	88	540

performance for the case of a single or a small number of fronts. The result for the single-front case without element reordering is not available in Table 3 because too much disk space is required and it takes too much time.

The finite element mesh of the composite shell with cutouts shown in Fig. 13 has 46,680 DOFs and 11,664 elements. The finite element model consists of four layers of composite laminates. The computation performance with 2048 fronts is compared with that of the single-frontal method with element reordering and is shown in Table 4. Four combinations of graph partitioning and mesh-mapping schemes are considered. The multifrontal solver with the graph mapped by the WEM scheme and the partition obtained by Metis (labeled "Metis–WEM") shows the best performance and is more than 4.2 times faster than the single-frontal method with element reordering. In this case the quality of the partition obtained by RSB is worse than that obtained by Metis. Although eight-noded elements are used, the graphs generated by the WEM scheme always produce better partitions and better performance than those generated by the CFM scheme for both RSB and Metis, which is not the case with the earlier regular-mesh example. The partitioning time using the WEM scheme is slightly longer than that using the CFM scheme because the graph mapped by the WEM scheme has more edges than the graph mapped by the CFM scheme. However, it is negligible compared with the performance improvement by the WEM scheme.

The computing results of the finite element model with the same geometry that has eight layers is shown in Table 5. This model has 84,024 DOFs and 23,328 elements. The performance with 4096 fronts is the best and is also compared with that of the single-front method in Table 5. The results are similar to the results in Table 4, and the developed multifrontal solver with the front partition found by the WEM scheme and Metis is about 5.3 times faster than the single-front method in this case.

From the preceding results with irregular meshes, one can see that the performance of the multifrontal solver compared with that of the single-front method is improved as the size of the problem increases and the effect of mesh-mapping schemes on the performance of the solver is very significant not only for 18-noded elements but also for 8-noded elements. The combination of Metis and the WEM scheme always gives the best performance of the multifrontal solver. Furthermore, the size of the required memory for this combination is the smallest, as shown in Table 5.

Conclusions

In this paper, a new multifrontal solver has been developed by combining the multifrontal scheme with graph partitioners and mesh-mapping schemes. The developed multifrontal solver inherits the merits of the conventional frontal method, such as reduction of core memory requirement, robustness, efficiency, and fitness to the finite element procedure, and has been shown to have performance superior to that of the frontal method.

By the new solver, finite element meshes are converted or mapped to graph structures, and by using graph partitioners based on the recursive bisection scheme, a front partition for the multifrontal scheme is found. According to the results of the performance test, the front-partition quality greatly affects the performance, and superior performance of the multifrontal solver can be guaranteed by using an appropriate mesh-mapping scheme and an efficient graph partitioner. Mesh-mapping schemes are very significant factors for good performance. The graphs mapped by the WEM scheme always give the best performance. The graph-partitioning routine provided by Metis is considered to be the best graph partitioner for the multifrontal method. If any other more efficient graph-partitioning algorithms are available, they can be easily applied to the multifrontal solver to improve the performance. Because graph partitioning is an important issue for parallel computing, many researchers are interested in developing graph partitioners, and many efficient graph partitioners are being developed. Thus, by using state-of-the-art graph partitioners, the performance of the multifrontal solver could be improved further.

Large-scale structural analysis problems were solved using the developed solver, and the excellent performance of the solver was shown. By using an appropriate number of fronts and an effective front partitioner, unsolvably large-sized finite element problems could be solved within tolerable computing time on a machine with affordable core memory.

Acknowledgments

This research was supported by the Agency for Defence Development of the Republic of Korea, Contract UD98011AD, and by the Institute of Advanced Machinery and Design of Seoul National University.

References

[1]Geng, P., Oden, J. T., and van de Geijn, R. A., "Parallel Multifrontal Algorithm and Its Implementation," *Computer Methods in Applied Mechanics and Engineering*, Vol. 149, No. 1–4, 1997, pp. 289–302.

[2]Irons, B. M., "A Frontal Solution Program for Finite Element Analysis," *International Journal for Numerical Methods in Engineering*, Vol. 2, 1970, pp. 5–32.

[3]Poole, E. L., Knight, N. F., and Davis, D. D., Jr., "High Performance Equation Solvers and Their Impact on Finite Element Analysis," *International Journal for Numerical Methods in Engineering*, Vol. 33, No. 4, 1992, pp. 855–868.

[4]Storaasli, O. O., "Performance of NASA Equation Solvers on Computational Mechanics Applications," NASA TR 96-1505, 1996.

[5]Farhat, C., Lanter, S., and Simon, H. D., "TOP/DOMDEC—A Software Tool for Mesh Partitioning and Parallel Processing," *Computing Systems in Engineering*, Vol. 6, No. 1, 1995, pp. 13–26.

[6]Duff, I. S., and Reid, J. K., "The Multifrontal Solution of Indefinite Sparse Symmetric Linear Equations," *ACM Trans. Math. Software*, Vol. 9, No. 3, 1973, pp. 302–325.

[7]Karypis, G., and Kumar, V., "A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs," Dept. of Computer Science, TR 95-035, Univ. of Minnesota, 1995.

[8]Karypis, G., and Kumar, V., "Metis: Unstructured Graph Partitioning and Sparse Matrix Ordering System," Dept. of Computer Science, TR, Univ. of Minnesota, 1995.

[9]Simon, H. D., "Partitioning of Unstructured Problems for Parallel Processing," *Computing Systems in Engineering*, Vol. 2, No. 2, 1991, pp. 135–148.

[10]Simon, H. D., and Teng, S. H., "How Good is Recursive Bisection," NASA TR-RNR-93-012, 1993.

[11]Karypis, G., and Kumar, V., "Multilevel K-Way Partitioning Scheme for Irregular Graphs," Dept. of Computer Science, TR 95-064, Univ. of Minnesota, 1995.

[12]Liu, W. H., and Sherman, A. H., "Comparative Analysis of the Cuthill–Mckee and the Reverse Cuthill–Mckee Ordering Algorithms for Sparse Matrices," *SIAM Journal on Numerical Analysis*, Vol. 13, No. 2, 1976, pp. 198–213.

[13]George, A., "Nested Dissection of a Regular Finite Element Mesh," *SIAM Journal on Numerical Analysis*, Vol. 10, No. 2, 1972, pp. 345–363.

[14]Venkatakrishnan, V., Simon, H. D., and Barth, T. J., "A MIMD Implementation of a Parallel Euler Solver for Unstructured Grids," NASA Rept. RNR-91-024, 1991.

[15]Paulino, G. H., Menezes, I. F. M., Gattass, M., and Mukherjee, S., "Node and Element Resequencing Using the Laplacian of a Finite Element Graph: Part I—General Concepts and Algorithm," *International Journal for Numerical Methods in Engineering*, Vol. 37, No. 9, 1994, pp. 1511–1530.

S. Saigal
Associate Editor