Transactions of the Korean Nuclear Society Autumn Meeting
Gyeongju, Korea, November 2-3, 2006

Application of an Iterative Solution Scheme to MARS

K. D. Kim, H. S. Lim and B. D. Chung
Thermal-Hydraulic Safety Research Division, Korea Atomic Energy Research Institute, 150 Dukjin-dong, Yuseong, Daejeon, 305-353, kdkim@kaeri.re.kr

1. Introduction

The recent version of MARS code (MARS 3.1) [1] has been enhanced over the previous version of MARS for several important models for the last couple of years. The most prominent enhancement that distinguishes MARS versions 3.X from the previous MARS versions 2.X is the fully multi-dimensional thermal-hydraulic modeling capability in the recent MARS code. This makes it possible to examine detailed behaviors where multi-dimensional effects may occur. This enhancement, however, makes the problem size too large to be able to efficiently handle it by using the default direct sparse matrix solver. To effectively solve the wide-banded system of linear equations for a multi-dimensional component, the latest version of RELAP5 replaced its default solver with the Border Profiled Lower Upper (BPLU) matrix solver which is designed to solve "nearly-banded" coefficient matrices directly [2]. However, direct matrix solvers usually cause the problems as the matrix size becomes larger because operation count for this method is proportional to third power of the matrix size and the truncation error during the forward and backward sweeps increases. The new iterative sparse matrix solver has been implemented to the latest version of MARS to efficiently solve sparse linear systems. The accompanying advantage of using an iterative solver is that this solver is easy to take advantage of a parallel machine. This paper will briefly describe the iterative solver and the evaluation results.

2. Iterative Sparse Matrix Solver

Preconditioned bi-conjugate gradient (PBCG) method [3] is implemented in MARS to efficiently solve sparse linear systems of the form $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$. Many variations of conjugate gradient methods exist and the bi-conjugate gradient method used in this application is specially designed to solve linear, but not necessarily symmetric equations. The attractiveness of these methods is that they reference $\mathbf{A}$ only through its multiplication of a vector, or the multiplication of its transpose and vector which can be very efficient for a properly stored sparse matrix. The conjugate gradient method is based on the idea of minimizing a function, whose gradient can be reduced to an original linear equation, instead of solving the original linear equation. Precondition is applied to the bi-conjugate gradient method to increase the rate of a convergence. Since the structures of matrix $\mathbf{A}$ depend on problems and they are different for every problem, it is very hard to determine a preconditioner which works for every problem. For this reason matrix $\mathbf{A}$ has been used as a preconditioner in this application. The matrix structure of MARS used for most problems is mainly tri- (for 1D problem) and/or a band-diagonal (for multi-D problem) matrix with a few none zero elements. It is surely inefficient to allocate storage for all the elements including zero elements in a memory management aspect and prohibitive in machine time to loop over the entire matrix in search of nonzero elements. Not to waste operations on zero entries, an indexed storage scheme which stores only none zero matrix elements along with the auxiliary information for the locations of the none zero elements is used.

In the MARS 3.X code, this iterative sparse matrix solver has been implemented as an option and tested with various types of MARS inputs. The option 99 in card 1 has to choose to select a PBCG method instead of the previously used direct sparse matrix solver for the solution of hydraulic equations. Instead of using the same solution scheme for the entire problem, a user can select the solution scheme for each loop in a problem by changing the value of the 5^{th} word in cards 121-129 i.e., 0 and 1 are for a direct sparse matrix solver which is a default solver; 2 and 3 are for the PBCG method. This input feature is very useful because a MARS input can include many loops in one problem and some loops may be too small to use the PBCG method.

PBCG is inherently parallelizable since its operations are independent of each other. Therefore, MARS is ready to take advantage of a shared-memory and a distributed-memory parallel architecture to run fast.

3. Performance Evaluation

Speedups are achieved for MARS running with the PBCG method over a previously used direct sparse matrix solver which is a default solver. For almost all one-dimensional problems, there is no speed-up or as fast; however, for problems with wider bandwidths, especially those with three-dimensional regions, for which it was intended, significant speed-ups may be achieved. One of the evaluation problems which uses the multi-d component illustrates a significant reduction in run time that can be achieved. The problem selected for an illustration is a simple 2-d vertical plate subdivided into an equal number of volumes in x- and y- directions. The test system was initially filled with single phase water and the transient starts with an air injection at the lower part of the left side at time zero second and a time dependent volume was attached to the top-right side as shown in Figure 1. Five cases were

examined with this model by changing the size of the test problem. Table 1 compares the total CPU times, the CPU time for solving linear equations and the average iteration number per each time step for the PBCG solver for the five cases for the default sparse matrix solver and the PBCG solver on the PC equipped with Pentium D CPU 3.4GHz and 1 GB ram. For PBCG, the convergence criterion for relative error was used as 10^{-10}. Results show that the speedup becomes larger with the size of the problem and their calculation results are almost identical with little differences in the least significant numbers. The minimum size to achieve a speedup is about 250 volumes.

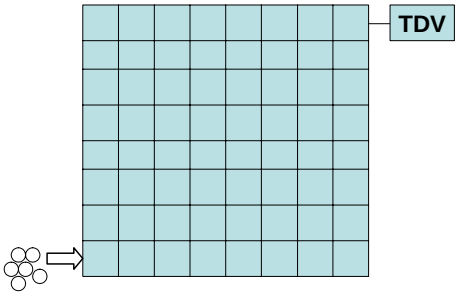

Figure 1. 2-d flow problem

Table 1. Comparison of run times for 2d-flow problems using default and PBCG solver

No. of volumes		270	450	900	1800	2916
Direct scheme	Total (s)	55	124	477	1289	2162
	Matrix (s)	9.4	39	272	788	1564
PBCG	Total (s)	55	106	258	644	734
	Matrix (s)	10.2	21	54	143	142
	avg. iteration	79	97	123	164	189
Speed up	Total	1.0	1.2	1.85	2.0	2.95
	Matrix	0.92	1.85	5.04	5.51	11.01

Many other cases have been examined with various types of problems to evaluate the new solver. Table 2 includes two problems for a 1-d case and 2 problems for a 3-d case. First case is the Marviken test problem [4] which is modeled as a 1-d problem with 45 volumes. For the second case, one of the standard installation problems, "typpwr.i" has been used, which includes 107 volumes for the primary loop and 16 volumes for each secondary side of two SGs. As expected, no-speedup was achieved for both 1-d cases. Third case is examined with the model for the Panda experiment [4], which includes multi-d components with a total of 3415 volumes. The speedup for this case was 1.3 which is much worse results than expected. The reason is that the "dead volumes" used to model the Panda experiment make all the matrix elements close to zero including the diagonal element for the corresponding row and increase the iteration number to converge. To confirm this explanation, the last case which includes a multi-d component with 2415 volumes and no "dead-volume" has been evaluated and a good speedup was obtained.

Table 2. Comparison of run times for various problems using default and PBCG solver

Problems		1-d case		3-d case	
		Marviken 45	Typpwr 107/16/16	Panda 3415	Tank 2425
Direct scheme	Total (s)	28.8	17.3	3811	1731
	Matrix (s)	0.27	0.14	2101	1240
PBCG	Total (s)	30.9	21.2	3034	647
	Matrix (s)	3.17	1.71	1142	128
	avg. iteration	42	114	404	185
Speed up	Total	0.93	0.82	1.3	2.7
	Matrix	0.3	0.1	1.84	9.68

3. Conclusion

The iterative sparse matrix solver, preconditioned bi-conjugate gradient method, is adopted to the latest version of MARS to efficiently solve the large sparse linear systems. As can be seen from the previous section, speed-ups over a previously used direct sparse matrix solver are achieved in MARS equipped with an iterative solver for multi-dimensional problems with size over ~250 which it was intended. For one-dimensional problems, this solver runs as fast or faster than the previously used direct solver. There is some room for further improvement for the treatment of the "dead volumes" in a multi-d component by eliminating these volumes from the equation system to be solved.

PBCG can inherently take advantage of a shared-memory and a distributed-memory parallel architecture to run fast because its operations are independent of each other. The feasibility study for a parallelization of MARS is under way on the cluster PCs by using message passing interface technology [5].

ACKNOWLEDGEMENTS

This work has been performed as a part of the Nuclear R&D Program supported by the Ministry of Science and Technology (MOST) of the Republic of Korea.

REFERENCES

[1] MARS Code Manual: Volume I: Code Structure, System Models, and Solution Methods, KAERI/TR-2812/2004, Korea Atomic Energy Research Institute, 2004.

[2] G. L. Mesina, Border-Profile LU Solver for RELAP5-3D, Proceedings of 1998 RELAP5 International User Seminar, May 17-2l, 1998, College Station, Texas.

[3] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, Numerical Recipes in C – The Art of Scientific Computing, 2nd Edition, Cambridge University Press, New York, 1992.

[4] MARS Code Manual: Volume IV: Developmental Assessment Report, KAERI/TR-3042/2005, pp.97-102 and pp.142:146, Korea Atomic Energy Research Institute, 2005.

[5] Massage Passing Interface Forum, MPI-2: Extension to the message-passing interface, July 1997, http://www.mpi-forum.org/docs/mpi2-report.html