The Application Strategy of Iterative Solution Methodology to Matrix Equations in Hydraulic Solver Package, SPACE

Young W. Na*, Chan.E. Park, Sang Y. Lee
KOPEC, Safety Analyses Dept, 150 Deokjin-dong, Yuseong-gu, Daejeon 305-353, Korea
**Corresponding author: ywna@kopec.co.kr*

1. Introduction

As a part of the Ministry of Knowledge Economy (MKE) project, “Development of safety analysis codes for nuclear power plants”, KOPEC has been developing the hydraulic solver code package applicable to the safety analyses of nuclear power plants (NPP’s).

The matrices of the hydraulic solver are usually sparse and may be asymmetric.

In the earlier stage of this project, typical direct matrix solver packages MA48 [1, 2] and MA28 [3] had been tested as matrix solver for the hydraulic solver code, SPACE. The selection was based on the reasonably reliable performance experience from their former version MA18 [4] in RELAP computer code.

In the later stage of this project, the iterative methodologies have been being tested in the SPACE code. Among a few candidate iterative solution methodologies tested so far, the biconjugate gradient stabilization methodology (BICGSTAB) has shown the best performance in the applicability test and in the application to the SPACE code. Regardless of all the merits of using the direct solver packages, there are some other aspects of tackling the iterative solution methodologies. The algorithm is much simpler and easier to handle. The potential problems related to the robustness of the iterative solution methodologies have been resolved by applying pre-conditioning methods adjusted and modified as appropriate to the application in the SPACE code.

The application strategy of conjugate gradient method was introduced in detail by Schewchuk, Golub and Saad in the middle of 1990’s [5, 6 & 7].

The application of his methodology to nuclear engineering in Korea started about the same time and is still going on and there are quite a few examples of application to neutronics [8, 9 & 10]. Besides, Yang introduced a conjugate gradient method programmed in C++ language [11].

The purpose of this study is to assess the performance and behavior of the iterative solution methodology compared to those of the direct solution methodology still being preferred due to its robustness and reliability.

The main object of this work is not to investigate the whole transient behavior of the models at hand but to focus on the behavior of numerical solutions part of the sparse asymmetric matrix equations in the transient of hydraulic system.

It is outside of the scope of this work to improve the diagonal dominance or to pre-condition the matrix in the process of differencing and linearizing the governing equation, even though it is better to do it that way before applying the solver if there is any efficient way available.

2. Methods and Results

In this section, the iterative solution methodology and the application strategies are described.

2.1 Handling the Asymmetry of Matrices

In solving a set of matrix equation $\mathbf{Ax} = \mathbf{b}$ by applying the iterative solution method, as the residual correction process continues, the norm of error vector may decrease if the matrix is well-conditioned. Since the error vector cannot be obtained, the residual vector can be used instead to check the convergence. The residual vector at k-th iteration is described as:

$$\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\,\mathbf{x}^{(k)} \qquad (1)$$

where

$\mathbf{A}$ = the original matrix
$\mathbf{x}^{(k)}$ = the solution vector obtained at k-th iteration
$\mathbf{b}$ = the right-hand-side (RHS) / source vector
$\mathbf{r}^{(k)}$ = the solution vector obtained at k-th iteration.

In the process of conjugate gradient method with a symmetric matrix system, the residual vector, one direction vector, one scaling factor and one adjustment factor were enough for residual correction. However, in the BICGSTAB with an asymmetric matrix system, other auxiliary vectors and adjustment factors are also involved to handle the asymmetry, which makes the process more complicated.

With all the mathematical manipulations, Yousef Saad summarized the BICGSTAB algorithm as follows [7]:

```
1.  Compute $\mathbf{r}_0 := \mathbf{b} - \mathbf{Ax}_0$ ; $\mathbf{r}^*_0$ arbitrary.
2.  $\mathbf{p}_0 := \mathbf{r}_0$.
3.  For j = 0, 1, 2, …, until convergence Do:
4.          $\alpha_j := (\mathbf{r}_j, \mathbf{r}^*_0) / (\mathbf{Ap}_j, \mathbf{r}^*_0)$
5.          $\mathbf{s}_j := \mathbf{r}_j - \alpha_j \mathbf{Ap}_j$
6.          $\omega_j := (\mathbf{As}_j, \mathbf{s}_j) / (\mathbf{As}_j, \mathbf{As}_j)$
7.          $\mathbf{x}_{j+1} := \mathbf{x}_j + \alpha_j \mathbf{p}_j + \omega_j \mathbf{s}_j$
8.          $\mathbf{r}_{j+1} := \mathbf{s}_j - \omega_j \mathbf{As}_j$
9.          $\beta_j := [(\mathbf{r}_{j+1}, \mathbf{r}^*_0) / (\mathbf{r}_j, \mathbf{r}^*_0)] x [\alpha_j / \omega_j]$
10.         $\mathbf{p}_{j+1} := \mathbf{r}_{j+1} + \beta_j (\mathbf{p}_j - \omega_j \mathbf{Ap}_j)$
11. EndDo.
```

2.2 Handling the Sparsity of Matrices

Only the non-zero elements are kept to handle the sparsity, applying the quasi-band matrix memory index system, compared to the coordinate memory index system used in MA direct solver packages.

The dynamic memory allocation method is also used to handle the variable dimension involved with this sparse matrix memory index system.

2.3 Pre-conditioning Strategy

The robustness of the convergence and the behavior in terms of residual reduction can be improved by pre-conditioning.

For a given convergence criteria, the number of iterative process can be reduced by improving the conditions of the matrix system by applying pre-conditioning methods. The simple Jacobi pre-conditioning or the ILU pre-conditioning can be applied.

The later one reduces the number of iterations dramatically, maybe down to less than few times, in most typical cases and yet it is more complicated and involves extra time-consuming process, such that in some cases it takes more overall calculation time than the case without the pre-conditioning, which must be overcome by applying appropriate strategy.

2.4 Time-Saving Strategy of Applying the Iterative Solution Method

As the time-saving process was the most important strategy in the direct solver packages, somewhat different type of time-saving process is equally important in the iterative solution process also.

The dynamic memory allocation and the memory index mapping are set outside of transient iterations.

Defining the coefficient matrix and pre-conditioning is performed at the beginning of each transient time step, but outside of the iterative processes for the solution of matrix equation.

2.5 Results

The method has been tested with typical 10x10, 32x32 asymmetric matrix equations and also with the 45 cells Marviken test case, in which fluid is discharged through a nozzle at the bottom of a full-size vertical reactor. The time frame is 60 seconds. When this solution methodology was tested in steady-state stabilization process, the iterations required to satisfy the convergence criteria of $1x10^{-14}$, which is the reasonably achievable calculation accuracy with the double precision version MA package, was a little more than 40 times during the calculation of earlier settling-down process. This number is dramatically reduced down to 10~15 times, mostly 5 or 10 in the subsequent transient analyses without pre-conditioning. The result shows a good acceptable behavior up to this point.

Since the code is to be used with one short time steady-state initialization and many subsequent longer time-frame restart transient case analyses for one system model in general, this change in convergence behavior can be judged to be optimistic.

3. Conclusions

The test results have shown that the iterative solution methodology selected and applied to the hydraulic solver package being developed works well with typical test cases.

Acknowledgment

This study was performed under the project, "Development of safety analysis codes for nuclear power plants" sponsored by the Ministry of Knowledge Economy.

REFERENCES

[1] Aspentech, HSL 2007 MA48 Version 2.1.0 Package Specification, AERE, Harwell, Oxfordshire, February 27, 2008.

[2] I.S. Duff & J.K. Reid, MA48, a Fortran Code for Direct Solution of Sparse Unsymmetric Linear Systems of Equations, RAL-93-072, Central Computing Department, Rutherford Appleton Laboratory, Oxon OX11 0QX, October 1993.

[3] I.S. Duff, MA28 - A set of Fortran Subroutines for Sparse Unsymmetric Linear Equations, AERE - R 8730, Computer Science and Systems Division, AERE Harwell, Oxfordshire, November 1980.

[4] A.R. Curtis & J.K. Reid, MA18 - Fortran Subroutines for the Solution of Sparse Sets of Linear Equations, AERE - R 6844, Theoretical Physics Division, AERE Harwell, Berkshire 1971.

[5] Schewchuk J. R., An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, Edition 1&1/4, School of Computer Science, Carnegie Mellon University, PA 15213, 1994.

[6] Golub G. H. , et. Al., Matrix Computations, 3rd Ed., The Johns Hopkins University Press, Baltimore & London, 1996.

[7] Saad Y., Iterative Methods for Sparse Linear Systems, PWS Publishing Co., Boston MA, U.S.A. , 1996.

[8] H. G. Joo and T. J. Downar, "Incomplete Domain Decomposition Preconditioning for Coarse Mesh Neutron Diffusion Problems," Proc. Int. Conf. Mathematics and Computational Reactor Physics and Environmental Analysis, Vol. 2, p. 1584, April 30–May 4, 1995.

[9] Ku Young Chung and Chang Hyo Kim, "Temporal Adaptive Three-Grid Correction Method for Transient Nonlinear Nodal Calculations," Nuclear Science and Engineering, 151, 212-223, 2005.

[10] T.Y.Han, H.G.Joo, H.C.Lee, C.H.Kim, "Multi-group unified nodal method with two-group coarse-mesh finite difference formulation", Annals of Nuclear Energy, Vol. 35, pp. 1975-1985, 2008.

[11] Yang D., C++ and Object-Oriented Numeric Computing for Scientists and Engineers, Springer-Verlag New York, Inc., 2001.