PHYSOR 2004 -The Physics of Fuel Cycles and Advanced Nuclear Systems: Global Developments
Chicago, Illinois, April 25-29, 2004, on CD-ROM, American Nuclear Society, Lagrange Park, IL. (2004)

Improving the Computation Efficiency of COBRA-TF for LWR Safety Analysis of Large Problems

Diana Cuervo[*1], Maria N. Avramova[2] and Kostadin N. Ivanov[2]
[1]*Department of Nuclear Engineering, Polytechnic University of Madrid*
Avda. Arco de la Victoria s/n, 28040 Madrid, Spain
[2]*Department of Mechanical and Nuclear Engineering, The Pennsylvania State University*
230 Reber Building, University Park, PA 16802, USA

A matrix solver is implemented in COBRA-TF in order to improve the computation efficiency of both numerical solution methods existing in the code, the Gauss elimination and the Gauss-Seidel iterative technique. Both methods are used to solve the system of pressure linear equations and relay on the solution of large sparse matrices. The introduced solver accelerates the solution of these matrices in cases of large number of cells. The execution time is reduced in half as compared to the execution time without using matrix solver for the cases with large matrices. The achieved improvement and the planned future work in this direction are important for performing efficient LWR safety analyses of large problems.

KEYWORDS: efficiency, subchannel analysis, numerical methods, matrix solver.

1. Introduction

The increased use of detailed descriptions of reactor core for subchannel safety analysis is making necessary to improve code numerical method performance and speed in order to obtain reasonable running times for large problems. In the case of two-fluid codes due to the extended set of complex equations the necessity of a highly efficient numerical method and matrix solver is even more pronounced.

COBRA-TF [1] is a two-fluid, three-field code (continuous vapor, continuous liquid and entrained liquid drops) for two-phase flow problems. The three field conservation equations for multidimensional flow are discretized using semi-implicit finite-difference technique with donor cell differencing for the convected quantities. The formed in this way set of algebraic equations must be solved simultaneously over each cell as well as for the entire computational mesh in order to obtain parameter distributions for the different fields.

One of the most important steps in the solution process in terms of CPU time is the solution of a system of linear equations that relate pressure in each node of the computational mesh. Since the dimension of the matrix of this system is usually large the use of an optimized matrix solver could reduce the duration of the solution process significantly.

The purpose of this work was to investigate the application of a new optimized matrix solver to the analysis of large problems using both the direct inversion and Gauss-Seidel iterative methods for the solution of the system of pressure equations.

---

[*] Corresponding author, Tel. + 34-91-336-7177, FAX +34-91-544-2149, E-mail: dcuervo@etsin.upm.es. Visiting scholar at *Department of Mechanical and Nuclear Engineering, The Pennsylvania State University*

2. Numerical solution of the two-fluid equations in COBRA-TF

The two-fluid three-field semi-implicit finite difference equations must be simplified to be solved in a reasonable amount of time although they must converge to the correct solution.

2.1 Solution scheme of the two-fluid equations

The first stage in the solution process (outer iteration), performed at every time step, is to solve the momentum equations using currently known values for all the variables to obtain an estimate of the new time step flow for every cell. All explicit terms in the momentum equations are computed in this stage and are assumed to remain constant during the remainder of the time step. Tentative velocities can be now calculated in order to be introduced in the continuity and energy equations.

As second stage the block Newton-Raphson method is used to obtain the variation of each independent variable required to bring the residual errors to zero. The residual errors (E_{CL}, E_{EV}, E_{EL}, E_{CE}, E_{CV} and E_{CG}) appear when the new velocities are used to compute the convective terms in these equations. As example of equation residual form we show vapor mass equation for channel *i* and section *j*:

$$E_{CV} = \frac{\left[(\alpha_v\rho_v)_j^n - (\alpha_v\rho_v)_j\right]A_{cj}}{\Delta t} + \sum_{KA=1}^{NA}\frac{\left[(\alpha_v\rho_v)^*\tilde{U}_{vj}A_{mj}\right]_{KA}}{\Delta x_j} - \sum_{KB=1}^{NB}\frac{\left[(\alpha_v\rho_v)^*\tilde{U}_{vj-1}A_{mj-1}\right]_{KB}}{\Delta x_j}$$
$$- \sum_{L=1}^{NKK} S_L\left[(\alpha_v\rho_v)^*\tilde{V}_{vL}\right]_j - \frac{\Gamma_j}{\Delta x_j} - \frac{S_{cvj}}{\Delta x_j} \qquad (1)$$

where * symbol means donor cell quantities, *n* superscript means quantities at new time step and ~ symbol over the velocities U indicates that they are the tentative values computed from the momentum equations. ρ_v and α_v are the density and void fraction of vapor, A_c and A_m are the continuity and momentum cell cross section areas, Γ is the net rate of vapor generation, NKK number of transverse connections, NA and NB number of connections to top and bottom of cell, V transverse velocity and S_{cv} net entrainment rate of vapor.

The system should be linearized with respect to the independent variables α_v, $\alpha_v h_v$, $(1-\alpha_v)h_l$, α_e and pressure of the actual cell and those in contact with it, i.e, P_j, P_i with i=1,NCON, where NCON is the number of cells in contact with the one of interest. Applying the Newton-Raphson method one can obtain the matrix equation (2) for each cell.

$$\begin{bmatrix}
\frac{\partial E_{CG}}{\partial \alpha_g} & \frac{\partial E_{CG}}{\partial \alpha_v} & \frac{\partial E_{CG}}{\partial \alpha_v h_v} & \frac{\partial E_{CG}}{\partial (1-\alpha_v)h_l} & \frac{\partial E_{CG}}{\partial \alpha_e} & \frac{\partial E_{CG}}{\partial P_j} & \frac{\partial E_{CG}}{\partial P_{i=1}} & \cdots & \frac{\partial E_{CG}}{\partial P_{NCON}} \\
\frac{\partial E_{CL}}{\partial \alpha_g} & \frac{\partial E_{CL}}{\partial \alpha_v} & \frac{\partial E_{CL}}{\partial \alpha_v h_v} & \frac{\partial E_{CL}}{\partial (1-\alpha_v)h_l} & \frac{\partial E_{CL}}{\partial \alpha_e} & \frac{\partial E_{CV}}{\partial P_j} & \frac{\partial E_{CV}}{\partial P_{i=1}} & \cdots & \frac{\partial E_{CV}}{\partial P_{NCON}} \\
\frac{\partial E_{CV}}{\partial \alpha_g} & \frac{\partial E_{CV}}{\partial \alpha_v} & \frac{\partial E_{CV}}{\partial \alpha_v h_v} & \frac{\partial E_{CV}}{\partial (1-\alpha_v)h_l} & \frac{\partial E_{CV}}{\partial \alpha_e} & \frac{\partial E_{CV}}{\partial P_j} & \frac{\partial E_{CV}}{\partial P_{i=1}} & \cdots & \frac{\partial E_{CV}}{\partial P_{NCON}} \\
\frac{\partial E_{CE}}{\partial \alpha_g} & \frac{\partial E_{CE}}{\partial \alpha_v} & \frac{\partial E_{CE}}{\partial \alpha_v h_v} & \frac{\partial E_{CE}}{\partial (1-\alpha_v)h_l} & \frac{\partial E_{CE}}{\partial \alpha_e} & \frac{\partial E_{CE}}{\partial P_j} & \frac{\partial E_{CE}}{\partial P_{i=1}} & \cdots & \frac{\partial E_{CE}}{\partial P_{NCON}} \\
\frac{\partial E_{EL}}{\partial \alpha_g} & \frac{\partial E_{EL}}{\partial \alpha_v} & \frac{\partial E_{EL}}{\partial \alpha_v h_v} & \frac{\partial E_{EL}}{\partial (1-\alpha_v)h_l} & \frac{\partial E_{EL}}{\partial \alpha_e} & \frac{\partial E_{EL}}{\partial P_j} & \frac{\partial E_{EL}}{\partial P_{i=1}} & \cdots & \frac{\partial E_{EL}}{\partial P_{NCON}} \\
\frac{\partial E_{EV}}{\partial \alpha_g} & \frac{\partial E_{CG}}{\partial \alpha_v} & \frac{\partial E_{EV}}{\partial \alpha_v h_v} & \frac{\partial E_{EV}}{\partial (1-\alpha_v)h_l} & \frac{\partial E_{EV}}{\partial \alpha_e} & \frac{\partial E_{EV}}{\partial P_j} & \frac{\partial E_{EV}}{\partial P_{i=1}} & \cdots & \frac{\partial E_{EV}}{\partial P_{NCON}}
\end{bmatrix}
\cdot
\begin{Bmatrix}
\delta\alpha_g \\ \delta\alpha_v \\ \delta\alpha_v h_v \\ \delta(1-\alpha_v)h_l \\ \delta\alpha_e \\ \delta P_j \\ \delta P_{i=1} \\ \vdots \\ \delta P_{i=NCON}
\end{Bmatrix}
= -
\begin{Bmatrix}
E_{CG} \\ E_{CL} \\ E_{CV} \\ E_{CE} \\ E_{EL} \\ E_{EV}
\end{Bmatrix} \qquad (2)$$

This equation can be written in an operator form:

$$[R(x)]\{\delta(x)\} = -E \tag{3}$$

where [R(x)] is the Jacobian of the system of equations evaluated for the set of independent variables (x) and composed of analytical derivatives of each equation with respect to linear variation of independent variables, δ is the solution vector containing these linear variations and E is the errors' vector.

After calculation of terms, the former system is analytically reduced using the Gaussian elimination to obtain solutions for the independent variables. Void fraction related variables and pressure for the actual cell are dependent on the pressure of adjacent cells (Equation 4).

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{14} & r_{15} & r_{16} & r_{17} & \cdots & r_{1(6+NCON)} \\ 0 & r_{22} & r_{23} & r_{24} & r_{25} & r_{26} & r_{27} & \cdots & r_{2(6+NCON)} \\ 0 & 0 & r_{33} & r_{34} & r_{35} & r_{36} & r_{37} & \cdots & r_{3(6+NCON)} \\ 0 & 0 & 0 & r_{44} & r_{45} & r_{46} & r_{47} & \cdots & r_{4(6+NCON)} \\ 0 & 0 & 0 & 0 & r_{55} & r_{56} & r_{57} & \cdots & r_{5(6+NCON)} \\ 0 & 0 & 0 & 0 & 0 & r_{66} & r_{67} & \cdots & r_{6(6+NCON)} \end{bmatrix} \cdot \begin{Bmatrix} \delta\alpha_g \\ \delta\alpha_v \\ \delta\alpha_v h_v \\ \delta(1-\alpha_v)h_l \\ \delta\alpha_e \\ \delta P_j \\ \delta P_{i=1} \\ \vdots \\ \delta P_{i=NCON} \end{Bmatrix} = -\begin{Bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{Bmatrix} \tag{4}$$

After reducing the system an equation of the form:

$$\delta P_j = a + \sum_{i=1}^{NCON} g_i \delta P_i \tag{5}$$

is derived for each cell. Thus a system with a number of equations equal to the number of computational cells should be solved in order to obtain the pressure variation for every cell. After this step the linear variation in the other independent variables is unfolded.

A flow diagram of the process is shown in Figure 1.

2.2 Solution of the matrix of pressure system

In regard to the pressure equation, the size of this system depends on the number of cells in the problem. The dimension of the matrix is the square of the number of cells. For a case with a small number of cells the equation set may be solved by direct inversion. For a case with a large number of mesh cells a Gauss-Seidel iterative technique is recommended. This technique is based on splitting the mesh cells in groups of cells that influence greatly each other. These groups of cells are called *simultaneous solution groups*. Equation set is split in the same way. When a solution group is being solved the values of δP_j for the cells that do not belong to the group are set to the previously calculated values. The multiplication of the pressure matrix by the independent variables' vector produces a linear system with the same number of equations as the number of cells in the solution group. This system is solved by the Gaussian elimination. A Gauss-Seidel iteration (inner iteration) is carried out over the groups of cells to obtain the new pressure vector. Convergence is reached when the change in δP_j for each cell from one iteration to the next fulfills a convergence criteria. The process is shown in Figure 2.

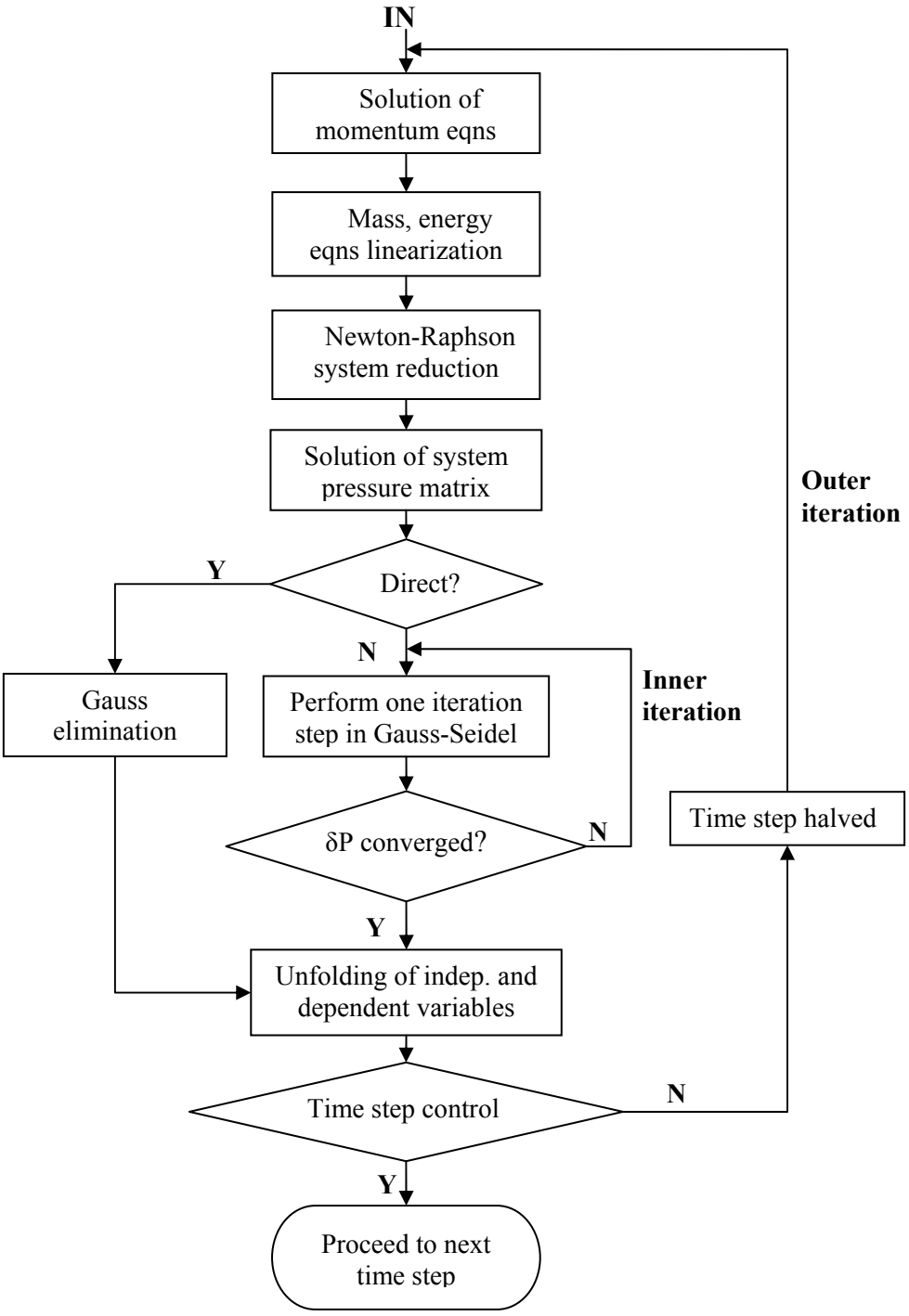

Fig.1 COBRA-TF solution flow diagram.

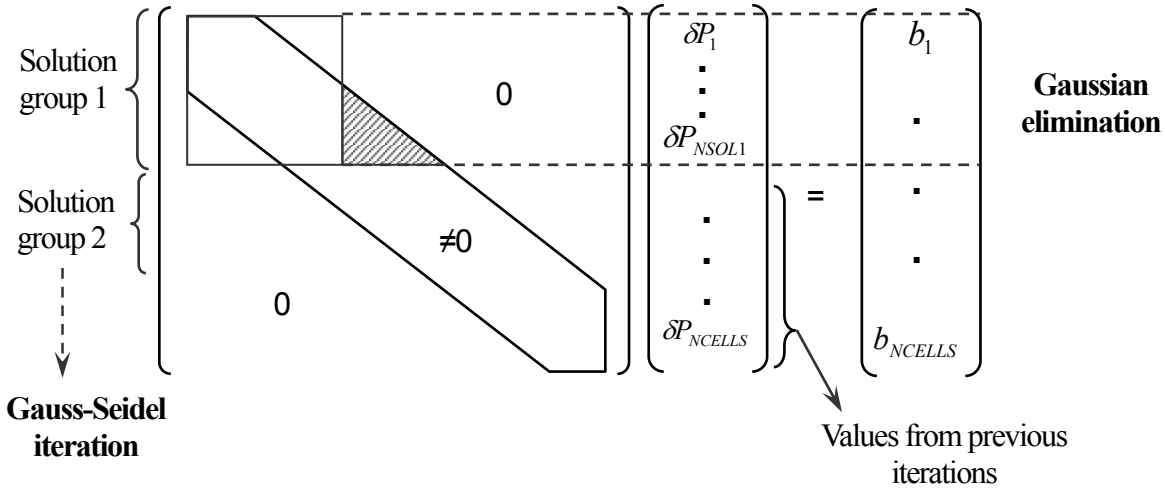

Fig. 2. Gauss-Seidel iteration for the *simultaneous solution groups*.

As it was shown, in both numerical methods, currently implemented in the code, one or more linear equation systems must be solved by Gaussian elimination. It is obvious that the performance of this part of the solution process will influence greatly the code computation efficiency.

3. Library for efficiency improvement of direct inversion of matrices

COBRA-TF is using the Gaussian elimination to solve the system pressure matrix in both solution alternatives existing in the code. In order to improve the solution process performance some subroutines from SuperLU library were utilized.

3.1 SuperLU library

SuperLU [2] is a general purpose library for the direct solution of large sparse systems of linear equations on high performance machines. The library is written in C and is callable from either C or Fortran. The library routines will perform a LU decomposition with partial pivoting and solution of the triangular system through forward and backward substitution.

Three libraries exist - sequential, multithreaded, and distributed, and all of them use variations of the LU decomposition optimized to take advantage of both the sparsity and the computer architecture [3]. Sequential SuperLU was selected as the most adequate for our use. The library provides a large set of subroutines to perform different steps of LU decomposition as well as other tasks like performance statistics or error bounds computing. Two standard driver algorithms are included but the user should design an optimized one for the problem being solved. All subroutines are available in single and double precision and for real and complex input data.

SuperLU depends on having a high performance BLAS (Basic Linear Algebra Subroutine) in order to be able to show high performance characteristics. In case this library is not available, SuperLU includes a default BLAS library with necessary subroutines. It is important to note that this library is not optimized for all machine architectures.

The above mentioned subroutines have to be compiled with C compiler. In order to embed them in a Fortran program, bridge subroutines (as well in C language) have to be coded. The system matrix has to be stored in the Harwell-Boeing sparse matrix format, i.e. compressed column storage.

3.2 Library implementation in COBRA-TF

Two new subroutines were developed as part of the matrix solver implementation. The first one, coded in Fortran, performs the transformation of the matrix of pressure into a Harwell Boeing format matrix in order to apply SuperLU subroutines to solve the system. Originally COBRA-TF only stores

the diagonal band of the matrix of pressure to save memory space due to the rest of matrix elements are zero. Thus this subroutine must carry out two tasks, recovering of the complete matrix of pressure and storage of this matrix in the Harwell-Boeing format. These two tasks must be done in only one step in order to save time in the solution process.

The second subroutine, coded in C, performs the solution procedure steps to obtain the solution array for the system.

In the factorization process two matrices, the *column permutation matrix* and the *row permutation matrix* (Pr A Pc=LU), must be calculated. These matrices do not change if the matrix A has the same sparsity structure and similar numerical entries for every time step as it is the case in subchannel analysis. Thus the first stage consists in the calculation of the permutation matrices if the code is solving for the first time step. After that, the code performs the calculation of L and U matrices at each time step and solution of the system through forward and backward substitution. The permutation matrices are stored to be reused at every time step. Memory allocation for L and U matrices is performed only at the first time step. This approach is designed not only to use an optimized matrix solver with better performance than the one implemented originally in the code, but as well to avoid unnecessary steps that need to be carried out only once.

It was observed that the number of Gauss-Seidel iterations is strongly dependent on the time step size. At this moment COBRA-TF selects time step size to fulfill finite-difference equations convergence criteria. However, this time step selection algorithm can be optimized to improve inner iteration convergence.

4. Application of COBRA-TF with the implemented matrix solver

Two cases have been calculated to test the implementation performance. These calculations were carried out on a Pentium 4 machine with 1.2 GHz processor and 512 MB RAM.

The first one is a flow reduction transient hot subchannel analysis of PWR (157 17x17 fuel assemblies). An embedded mesh [4] was used for this analysis. The interior channels are real thermalhydraulic channels within the assembly. From interior to exterior different size lumped channels were modeled until reaching a quarter of assembly size (Figure 3). The model contains 233 channels and 34 axial nodes per channel that means a matrix dimension of 7920×7920. Boundary conditions are inlet flow and enthalpy and outlet pressure. Axial and radial power distributions are obtained from 3D

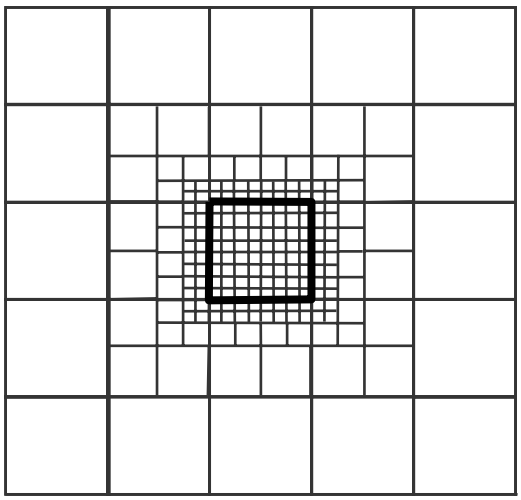

Fig. 3. Embedded mesh configuration for subchannel analysis problem.

neutronic analysis. The calculation was performed using both methods, direct and iterative. For the iterative method, 34 simultaneous solution groups were selected for the calculation - one group per axial level. The results are shown in Table 1. The number of inner iterations in this case was up to 4.

In the second case a PWR (157 17x17 fuel assemblies - FA) main steam line break (MSLB) scenario was simulated [5]. The inlet mass flow rate and enthalpy as inlet boundary condition at the core bottom and the system pressure as boundary condition at the top of the core are applied. The number of channels is 157 (one channel per FA) with 40 axial nodes per channel, which constitutes a pressure matrix of dimension 6280×6280. Both methods were again applied using 40 simultaneous groups for the iterative method. Due to the high number of iterations observed in this case that reduces in great manner the efficiency of the code a second calculation with time step optimization was performed. In this last case the optimization was carried out in a tentative manner without selection algorithm to reduce the inner iteration number up to 7 in most of the time steps. The CPU time results are shown in Table 1.

Table 1 CPU time results

Subchannel analysis	*Matrix solver implemented*	*Standard code*
Direct solution	6h 03min	13h 25min
Gauss-Seidel iteration	2h 36min	5h 24min

MSLB scenario	*Matrix solver implemented*	*Standard code*
Direct solution	3h 24min	7h 19min
Gauss-Seidel iteration	3h 16min	3h 12min

To prove that the matrix solver implementation do not introduce deviations to the COBRA-TF results two variables have been selected for analysis: pressure drop and phasic velocities. Table 2 shows the obtained axial distributions for a selected channel in the subchannel analysis case. It can be observed that the deviations are negligible in all comparisons. Similar situation with even smaller deviations is found in the MSLB scenario calculations.

5. Conclusions

One can see from the results presented in Table 1 that the new matrix solver implementation demonstrates a capability to reduce CPU times to half of the standard CPU times (without using the matrix solver) for both steady state and transient cases if the pressure matrices that must be solved are large enough to reach the size of matrix solver optimum performance. The use of the Gauss-Seidel iterative method reduces the dimension of matrices that must be solved by direct inversion to the square of number of simultaneous groups times the dimension of complete matrix of pressure system. Thus the dimension of these smaller matrices can be situated bellow the solver optimum performance limit. As conclusion it seems obvious that the larger the case is that must be solved the greater would be the reduction in CPU time compared to the standard CPU time. The combination of the Gauss-Seidel iterative method with the implemented matrix solver could allow analyzing much larger cases with the same CPU time.

Table 2 Deviations (in %) of selected predicted variables from values obtained from original methods for flow reduction transient hot subchannel analysis case.

	Gauss Seidel (a)			Matrix solver imp. (b)			Matrix solver imp. in G.S. (c)		
Axial location (ft)	*Press. Drop (psia)*	*Liquid Velocity (ft/s)*	*Vapor Velocity (ft/s)*	*Press. Drop (psia)*	*Liquid Velocity (ft/s)*	*Vapor Velocity (ft/s)*	*Press. Drop (psia)*	*Liquid Velocity (ft/s)*	*Vapor Velocity (ft/s)*
12	0.03	0	-0.05	0.04	0	-0.05	0.02	0.12	0.38
11.8	0.04	0.06	0	0.04	0.06	-0.27	0.02	0.06	0.22
11.61	0.04	0.06	-0.11	0.04	0.12	0.27	0.02	0	0.06
11.41	0.03	0	0.17	0.04	0.12	0.55	0.03	0.06	0.22
11.21	0.04	-0.06	0.45	0.04	0	0	0.03	0.06	-0.22
11.04	0.03	-0.06	-0.17	0.04	-0.06	-0.33	0.03	0.06	0.22
10.89	0.03	0	0	0.03	0	-0.15	0.02	0	-0.05
10.52	0.04	0.06	0.06	0.04	0.06	-0.06	0.02	0	0
10.07	0.03	0.06	0.17	0.03	0.06	0.17	0.02	-0.06	-0.11
9.62	0.03	0.07	0.30	0.03	0.07	0.18	0.02	-0.07	-0.18
9.18	0.03	0	-0.18	0.04	0	0.18	0.02	0.07	0.35
8.8	0.02	0	0.25	0.03	0	0.13	0.02	0.07	-0.13
8.36	0.02	0	-0.31	0.03	0	-0.31	0.02	0.07	0.37
7.91	0.02	0	-0.21	0.03	0.07	0.20	0.03	0.07	0.21
7.47	0.02	0	0.07	0.03	0	0.07	0.02	0.07	0.07
7.09	0.02	0	0.07	0.03	0.07	0.07	0.02	0.07	0
6.64	0.01	0	0.07	0.03	0.07	0.15	0.02	0	0
6.2	0	0	0	0.02	0.07	0.07	0.02	0.07	0.07
5.75	0.01	0	0	0.03	0	0	0.01	0	0
5.38	0.01	0	0	0.03	0	0	0.01	0.08	0.07
4.93	0	0	0	0.02	0	0	0.03	0	0
4.49	0	0	0	0.02	0	0	0.02	0	0
4.04	0	0	0	0.04	0	0	0.02	0	0.08
3.67	0	0.08	0	0.05	0.08	0	0.02	0	0
3.22	0	0	0	0.02	0	0	0.02	0	0.08
2.77	0	0	-0.08	0.03	0.08	-0.72	0.03	0	-0.16
2.33	-0.04	0	0	0	0	-0.49	0.04	0	-0.49
1.95	-0.05	0	0.40	0	0.08	0.24	0.09	0	-0.24
1.64	0	-0.08	0.08	0.05	-0.08	0.57	0.05	0	-0.08
1.33	0	-0.08	-0.33	0.06	-0.08	0.08	0.06	0.08	0.08
1.02	0	0	0.25	0.07	0	0.16	0	0	0.41
0.72	0	-0.08	0	0.08	0	-0.33	0.08	0.16	-0.08
0.42	-0.09	-0.08	-0.08	0	0	-0.67	0.09	0.08	-0.50
0.19	0.03	0	0.41	0.04	0.08	0	0.02	0.08	-0.08

(a) Deviation of results using Gauss-Seidel iterative method from values obtained by direct solution using Gauss elimination.
(b) Deviation of results using the direct solution with matrix solver implemented from values obtained by direct solution using Gauss elimination.
(c) Deviation of results using the Gauss-Seidel iterative method with matrix solver implemented from values obtained using Gauss-Seidel iterative method.

Another important conclusion of the study carried out for this work is that the efficiency of the method used by the code to solve the set of system pressure linear equations will affect the code efficiency in a great manner. So far the code is using either Gauss elimination or a Gauss-Seidel iterative technique. During last years, considerable research has been performed in a class of techniques known as Krylov subspace methods or non-stationary iterative methods which include the classical conjugate gradient method. The superior performance of

Krylov solvers as compared to the stationary iterative methods (to which Gauss-Seidel method belongs) has been well documented for the few-group neutron diffusion equation solution [6] as well as for phasic continuity equation solution in VIPRE-02 [7], which is a two-fluid two-field code for subchannel analysis. The implementation of a non-stationary method for the solution of the matrix of pressure system in COBRA-TF would increase further the efficiency of the code and this is the subject of planned future work in this direction. The use of the matrix solver, implemented during this work, in the solution of linear systems that must be solved in most of the Krylov subspaces methods with the preconditioner matrix could represent an additional increase in the method efficiency that will be studied.

Acknowledgements

The authors wish to thank SuperLU library developers, James W. Demmel, John R. Gilbert and Xiaoye S. Li, for the work accomplished and freely distribution of the library.

References

1) C.Y. Paik et al., "Analysis of FLECHT-SEASET 163-Rod Blocked Bundle Data Using COBRA-TF", NUREG/CR-4166 (1985).
2) J.W. Demmel, J.R. Gilbert, S.L. Xiaoye, "SuperLU User's Guide", pp. 25-30 (2003).
3) J.W. Demmel et al., "A Supernodal Approach to Sparse Partial Pivoting", SIAM J. Matrix Anal. Appl. , **20** (3),pp. 720-755 (1999).
4) J.M. Aragones, C. Ahnert, O. Cabellos, "Methods and Performance of the Three-Dimensional Pressurized Water Reactor Core Dynamics SIMTRAN On-Line Code", Nuclear Science and Engineering, **124**, pp. 111-124 (1996)
5) K. Ivanov et al, "A PWR MSLB Benchmark, Final Specifications", NEA/NSC/DOC(97)15, OECD Nuclear Energy Agency, (1997)
6) H. Joo, T. Downar, "An Incomplete Domain Decomposition Preconditioning Method for Nonlinear Nodal Kinetics Calculating", Nuclear Science and Engineering, **123**, pp. 403-414 (1996)
7) T. Downar, H. Joo, "A Preconditioned Krylov Method for Solution of the Multi-dimensional, Two-fluid Hydrodinamics Equations", Annals of Nuclear Energy, **28**, pp. 1251-1267 (2001)