

Solution of Flow Equations in the USNRC Consolidated Code

John Mahaffy

Applied Research Laboratory
The Pennsylvania State University
P.O. Box 30
State College, PA 16804
jhm@psu.edu

Thomas Downar, Quan Wang

1290 Nuclear Engineering Building
Purdue University
West Lafayette, Indiana, 47907-1290
downar@ecn.purdue.edu, qwang@ecn.purdue.edu

Keywords: sparse matrix, reactor simulation, two-phase flow

ABSTRACT

The USNRC consolidated reactor analysis code supports approximation of the time-dependent two-phase flow equations with both the semi-implicit and Stability Enhancing Two-Step (SETS) methods. Flow can be modeled in arbitrary networks of 1-D and 3-D regions. Regardless of the selected method, final solution always requires the solution of sparse systems of linear equations. The consolidated code has been structured to cleanly isolate the generation of these linear systems from their solution. Fortran 90 derived types are used to record the structure of the sparse matrices, and to store the non-zero coefficients. Two basic solution methods are provided in the code with automated selection for minimal run time based on the matrix configuration. The isolation of the solution subroutine permits developers to experiment with other solution methods, as technology or needs change. The consolidated code also supports an extended solution procedure that supports flows modeled with multiple processes, including use of other flow modeling programs as one or more of the other processes. This permits use of programs with special purpose flow models for components such as accumulators or core makeup tanks. It also opens the possibility for convenient use of 3-D CFD analysis of an area of special interest, while the consolidated code provides conditions in the balance of the system.

1. INTRODUCTION

This work supports two goals of the USNRC's reactor safety code consolidation effort (Mahaffy et. al., 2000). The first is to ease the path for future code enhancements. These could include: addition of an interfacial area concentration transport equation, or extra gas or liquid fields (more equations and state variables); installation of higher order difference methods; or use of advanced methods for solution of linear equations resulting from the numerical solution of the flow equations. We have supported this goal by cleanly isolating solution of equations from the creation of terms in these equations, and providing a relatively simple data structure for storing sparse matrices.

This work also supports the goal of producing a code that significantly reduces the time required to perform a transient relative to run times of predecessor codes (RELAP5, TRAC-P, or TRAC-B). One simple advance has been the introduction of a sparse linear system solver that has reduced execution times for AP600 analysis by at least 25%, primarily through significant improvements in cache utilization. Another significant improvement has been the implementation of the solution procedure in a way that supports solution of a linear system with equations generated on more than one processor. Mahaffy and Uhle (2000) have used this capability to demonstrate run times for a parallel flow solution close to two times faster than a serial job when using a dual process job, and over three and a half times faster when using four processors.

The linear systems that must be solved result from solution of a two-fluid model for two phase flow using a SETS Numerical method (Mahaffy, 1982). All stabilizer flow equations are linear. Stabilizer mass and energy equations have the useful feature that those associated with the liquid (liquid mass, liquid energy, and solute mass) all have the same coefficient matrix and differ only in the right hand side. The gas equations (total gas mass, gas energy, non-condensable gas mass) also share a coefficient matrix. This makes it worth considering solution procedures that deal efficiently with multiple right hand sides, such as lower-upper triangular matrix factorization (LU decomposition) and some implementations of Krylov subspace methods. The remaining linear system is associated with solution of the semi-implicit step in SETS or a pure semi-implicit method when that numerical method has been selected. The semi-implicit equations are nonlinear, and are solved with a Newton iteration. The linearized equations generated during each iteration are block reduced, resulting in an isolated set of equations coupling the pressure changes in each computational cell. This equation system, and all those from the SETS stabilizer equations are basically tridiagonal in 1-D flow regions, and add a pair of detached off-diagonal bands for each additional dimension in a vessel model. The matrix structure is complicated by the presence of tee's, plenums, and 1-D to 3-D connections, which create off-diagonal coefficients in locations only known after careful input processing.

In this paper we discuss three aspects of the equation solution procedure. First is the base sparse matrix procedure derived from the solution method in TRAC-PF1 (Schnurr et. al. 1984). Next we discuss a more flexible, and at times much faster option, the SuperLU direct sparse matrix solver (Xiaoye, 1996). Finally we cover the modifications to the solution procedure, permitting solution of systems with evaluation spanning more than one process.

2. THE BASIC SOLUTION METHOD

During the design phase for the consolidated code, USNRC made a decision to use TRAC-PF1/MOD2 as a starting point for an evolutionary development process. The choice of this code was based on its modular, object-oriented structure. The choice to proceed with an evolutionary (rather than write from scratch) approach was based on a desire to have a fully functional product at all stages and a solid base for verification of new coding. In keeping with this evolutionary philosophy, the base implementation of consolidated code's linear solver was a close relative of the original TRAC-P network solution procedure. This permitted verification of the new solution procedure through use of test problems for which an exact match could be expected with calculations using original TRAC-PF1/MOD2 solver.

The original network solution method solves for unknowns within each user defined 1-D component as linear functions of the unknowns at (or adjacent to) the component junctions. This reduction is performed as each component evaluates its equations. A substitution of expressions for components' interior variables into flow equations at the junction, results in a closed set of "network junction" equations, which are solved for the network junction variables. If 3-D components are present, the network junction equations must be solved in terms of unknowns within the 3-D regions. Substitution of these expressions into the 3-D equation set leads to a closed set of equations and unknowns, which can be solved for the unknowns in the 3-D region. Back-substitution into the network equations, and reduced component equations results in the final solution over the full system.

The modified solution procedure follows a similar approach, but without direct consideration of component junctions. During initialization, each computational mesh cell in the system is given a unique index. Indices within all 3-D regions are grouped after those in 1-D regions. A similar indexing operation is performed for all cell edges (needed for stabilizer momentum equations). The indices are stored in a data structure associated with system components. When terms in the equations are evaluated, computational flow has been arranged so that any row of coefficients in the system can be evaluated independently of coefficient evaluation in other rows. Parallel distribution of work is possible by system component or at a finer grain by mesh cell. Currently we only distribute work by component.

The solution procedure scans the rows of the matrix representing 1-D equations for those rows with coefficients not contained in the basic tridiagonal band. These are designated as "network junction" rows, are noted, and are saved for later use in the solution. Variables with indices matching these row indices are designated "network junction" variables. The remaining rows are isolated in tridiagonal blocks and these systems of equations are solved for associated variables as functions of network junction variables. For example, a linear system associated with flow around a closed loop has the form:

$$\begin{pmatrix} a_{1,1} & a_{1,2} & 0 & 0 & 0 & 0 & 0 & a_{1,8} \\ a_{2,1} & a_{2,2} & a_{2,3} & 0 & 0 & 0 & 0 & 0 \\ 0 & a_{3,2} & a_{3,3} & a_{3,4} & 0 & 0 & 0 & 0 \\ 0 & 0 & a_{4,3} & a_{4,5} & a_{4,6} & 0 & 0 & 0 \\ 0 & 0 & 0 & a_{5,4} & a_{5,5} & a_{5,6} & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{6,5} & a_{6,6} & a_{6,7} & 0 \\ 0 & 0 & 0 & 0 & 0 & a_{7,6} & a_{7,7} & a_{7,8} \\ a_{8,1} & 0 & 0 & 0 & 0 & 0 & a_{8,7} & a_{8,8} \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \\ V_7 \\ V_8 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \end{pmatrix} \quad (1)$$

The equations from rows one and eight are isolated, since they include off-band coefficients.

$$\begin{aligned} a_{1,1} V_1 + a_{1,2} V_2 + a_{1,8} V_8 &= b_1 \\ a_{8,1} V_1 + a_{8,7} V_7 + a_{8,8} V_8 &= b_8 \end{aligned} \quad (2)$$

The remaining equations are arranged in the following form:

$$\begin{pmatrix} a_{2,2} & a_{2,3} & 0 & 0 & 0 & 0 \\ a_{3,2} & a_{3,3} & a_{3,4} & 0 & 0 & 0 \\ 0 & a_{4,3} & a_{4,4} & a_{4,5} & 0 & 0 \\ 0 & 0 & a_{5,4} & a_{5,5} & a_{5,6} & 0 \\ 0 & 0 & 0 & a_{6,5} & a_{6,6} & a_{6,7} \\ 0 & 0 & 0 & 0 & a_{7,6} & a_{7,7} \end{pmatrix} \begin{pmatrix} V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \\ V_7 \end{pmatrix} = \begin{pmatrix} b_2 \\ b_3 \\ b_2 \\ b_2 \\ b_2 \\ b_2 \end{pmatrix} - \begin{pmatrix} a_{2,1} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} V_1 - \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ a_{7,8} \end{pmatrix} V_8 . \quad (3)$$

As a first reduction step in the overall solution, Eq. 3 is solved in the form:

$$V_i = v_i + c_{i,1} V_1 + c_{i,8} V_8 \quad (4)$$

These equations are substituted into equations from rows one and eight (Eq. 2) and values obtained for V_1 and V_8 . Back-substitution into Eq. (4) completes the solution for a purely 1-D system. In a normal 1-D flow system, more than one tridiagonal equation block is isolated, and the tridiagonal blocks could be solved in parallel to reduce overall solution time.

For systems with 3-D regions, the solution contains a final reduction step that is nearly identical to the old TRAC-P network solution procedure. The network junction equations analogous to Eq. system (2) above contain terms involving unknowns within the 3-D region. Network junction variables are solved as linear functions of the 3-D variables, and these expressions are substituted into equations within the 3-D region as needed. The resulting closed system of 3-D equations and unknowns is solved and two back-substitution stages, produce first values for the network junction variables, then for the remaining 1-D variables.

Unfortunately the 3-D solution method inherited from TRAC-PF1 is very inefficient when a large number of 1-D connections are present, as is the case with the standard AP600 model. In addition, the network junction equations are solved as a general system of equations with no further consideration of sparsity. We reached a point in the development cycle where it was clear that speed advantages could be obtained by improving this solution procedure. However, effort required to further upgrade the existing solution coding would be very substantial. It was time to test our design goal of making it easy to install better equation solvers developed by others.

3. AN IMPROVED SPARSE SYSTEM SOLVER

A project was initiated at Purdue University to survey available sparse matrix solvers. It began with a study of typical problems expected for the consolidated code, concluding that equations resulting from NRC applications could be solved more efficiently with direct rather than iterative methods. Next speed implications of modern computer architectures were considered. As pointed out by many previous researchers (Bertsekas and Tsitsiklis, 1989, Grant and Ramsden, 1995, Kumar et. al. 1994), a primary reason for poor performance on modern CPU's can be excessively large data access times. In particular, the failure to properly utilize cache and to account for the large differences in the access times to L1 cache, L2 cache, and main memory become particularly noticeable as processor speeds increase. Initial efforts to improve consolidated code performance for large problems were therefore focused on implementing a new "SuperLU" matrix solution algorithm which was specifically designed to optimize cache performance (Xiaoye, 1996, Demmel et. al., 1997). The revised Consolidated Code structure made installation of this existing solution package very simple.

Any direct solution of sparse matrices depends strongly on the pattern of the matrix and the amount of fill-in required during LU factorization. Reordering the sparse matrix before computing can decrease the fill-in of the matrix and significantly decrease the number of floating point operations. The SuperLU package first utilizes a symbolic factorization to build the data structures necessary to compute and store elements of the factorized matrix. It then improves the data access speed by exploiting dense submatrices (supernodes) in the lower and upper triangular matrices and by managing the data locality in the process of computing. The algorithm is based on a left looking Cholesky factorization (Rice, 1981). A supernode is defined as a group of columns with the same non-zero structure relative to the matrix diagonal. It consists of a filled lower triangular block, below which all rows are either full or zero (see. Fig. 1). The supernodes allow multiple columns to be updated simultaneously, providing benefits in both storage and the ability to utilize higher level BLAS routines (Dongarra et. al., 1989) for the resulting dense submatrices. We use processor specific BLAS libraries to obtain optimal speed. The compact storage pattern used for supernodes, enables efficient use of cache.

Results for the SuperLU sparse solver have been mixed. On a DEC Alpha, it is always faster than the original solution. On a PC it is somewhat slower for all applications except those with very large numbers of vessel connections such as the AP600 model. The AP600 deck has a large number of vessel connections, which drastically slows the

original method. The impact of these vessel connections on the AP600 matrix structure can be seen Figure 2 which has several large off-diagonal stripes.

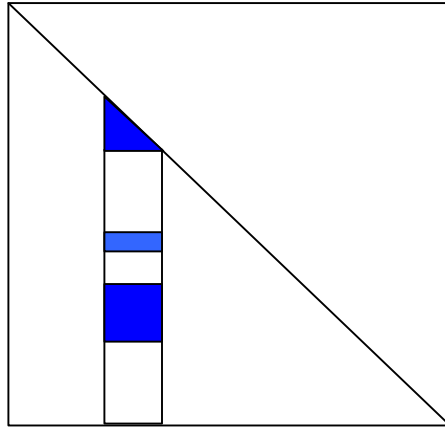


Fig. 1 A Supernode Substructure

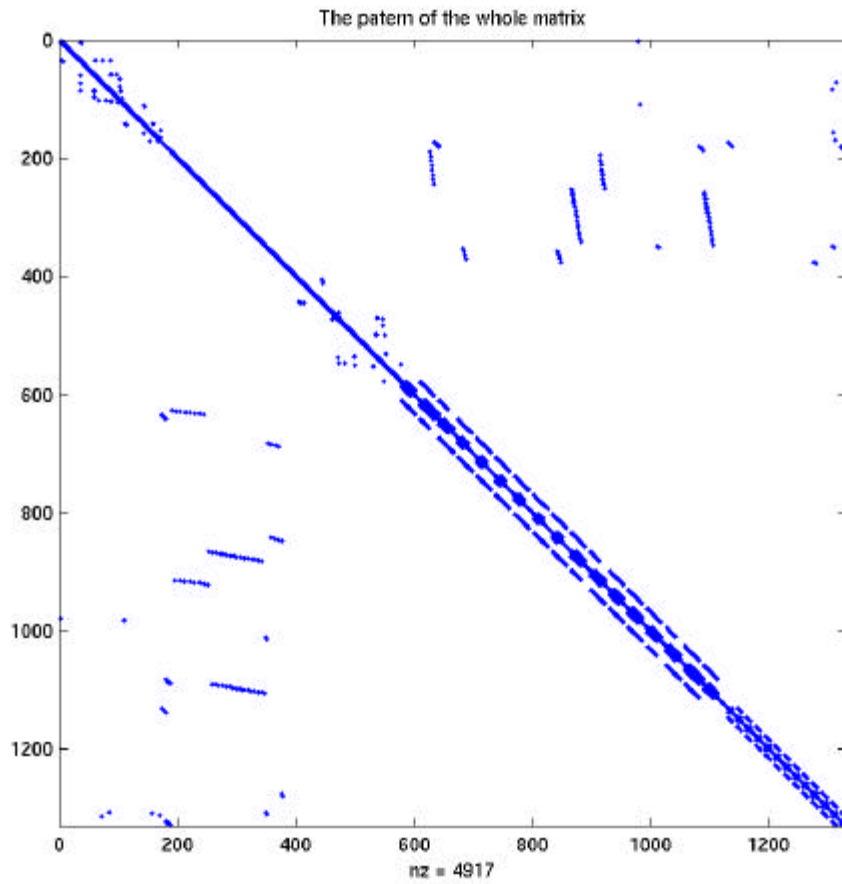


Fig. 2 Pressure Matrix Structure for AP600

With processor specific BLAS libraries SuperLU is an order of magnitude faster in solving the AP600 pressure matrix on a PC, and provides an overall run time reduction over 25% for the AP600 LBLOCA case. Work is underway to enhance its speed. Once completed it is anticipated that SuperLU will replace the old TRAC-M solver, but currently SuperLU is a user option. Work is also underway to exploit parallelism in the new matrix solver. Each of the supernodes can be distributed to different processors using standard threads interfaces such as OPENMP.

4. LINEAR SOLUTIONS IN PARALLEL COMPUTATIONS

When more than one task is responsible for fluid flow modeling, some form of implicit coupling should be maintained between the tasks to avoid numerical instabilities. If proper data naming conventions are followed, the Consolidated Code will correctly couple the solution of Semi-Implicit or SETS numerical methods across all processes contributing to the fluid flow. This is enabled by the Exterior Communications Interface (ECI) described by Mahaffy and Uhle (2000) elsewhere in these conference proceedings. Provisions also exist within the communications interface for another process to take control of the fluid equation solution. This represents a much more difficult programming effort for the application developer, but might be necessary for a special process which provides detailed CFD modeling of a specific region within the full system.

Parallel solution of the flow equations is based upon an agreement between tasks on the location of “exterior variables” to be used in a two stage reduction of the equations. Figure 3 illustrates special variable assignments performed during a multiprocessor solution of flow equations. The upper half of the loop is modeled with one process, and the lower half is simulated with a flow model in a parallel process. To shorten notation in the following discussion, cells in the figure have been given an absolute numbering convention. In terms of component numbering, cells 1-4 in this figure can be considered cells 1-4 of PIPE 1 (modeled by the first process), and cells 5-8 in the figure are cells 1-4 of PIPE 2 (modeled on the second process). For the multiprocessor solution, one variable is assigned on each side of each junction between tasks. For the semi-implicit pressure equation the exterior variables $*p_{Ext 1}$, $*p_{Ext 2}$, $*p_{Ext 3}$, and $*p_{Ext 4}$ correspond directly to the solution variables $*p$ at volumes with absolute indices 1, 4, 8, and 5 respectively.

Selection of exterior variables for cell edges associated with the SETS stabilizer momentum equations is somewhat more complex. Again two exterior variables are associated with each inter-task flow junction. At a given junction, the task responsible for evaluation of the junction momentum equations associates its exterior variable with that junction’s velocity. The task not responsible for the inter-task junction momentum equations evaluates its exterior variable at the next face in from the junction face. For the example in Figure 1, the exterior velocity variables $v_{Ext 1}$, $v_{Ext 2}$, $v_{Ext 3}$, and $v_{Ext 4}$ correspond directly to the stabilizer velocities at edges with absolute indices 1, 5, 8, and 6 respectively.

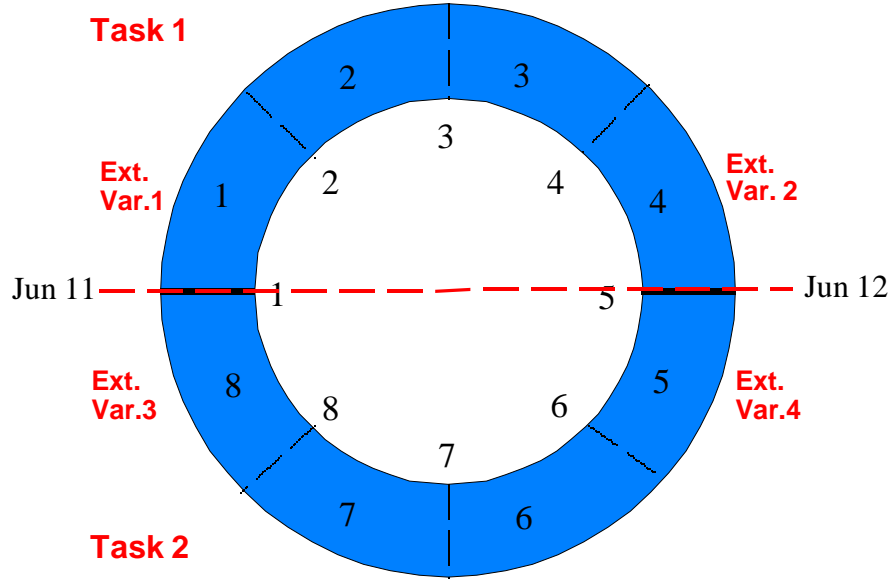


Fig. 3 Cell Centered Exterior Variables.

The first stage of the solution is local to each process, and follows the normal equation solution procedure with the addition of terms on the right hand side of each equation to account for the unknown effects of variables in adjacent processes. For Task 1 in the Figure 3 example the system of pressure equations would have the form:

$$\mathbf{A} \mathbf{p} = \mathbf{b} - \mathbf{a}_{\text{Ext } 3} p_{\text{Ext } 3} - \mathbf{a}_{\text{Ext } 4} p_{\text{Ext } 4}, \quad (5)$$

which is solved to give:

$$p_i = p_{0,i} - a_{i,3} p_{\text{Ext } 3} - a_{i,4} p_{\text{Ext } 4}. \quad (6)$$

A similar set of solutions are generated by Task 2 in the form:

$$p_i = p_{0,i} - a_{i,1} p_{\text{Ext } 1} - a_{i,2} p_{\text{Ext } 2}. \quad (7)$$

Operation of the matrix (A) on multiple right hand side vectors reinforces the selection of direct solution methods based upon LU decomposition.

For the second stage of the solution, the reduced equations adjacent to each inter-process connection are collected by the solver process:

$$\begin{aligned} p_1 &= p_{01} - a_{1,3} p_{\text{Ext } 3} - a_{1,4} p_{\text{Ext } 4} \\ p_4 &= p_{04} - a_{4,3} p_{\text{Ext } 3} - a_{4,4} p_{\text{Ext } 4} \\ p_8 &= p_{08} - a_{8,1} p_{\text{Ext } 1} - a_{8,2} p_{\text{Ext } 2} \\ p_5 &= p_{05} - a_{5,1} p_{\text{Ext } 1} - a_{5,2} p_{\text{Ext } 2}. \end{aligned} \quad (8)$$

Recognizing the identity relationships between local variables and exterior variables, and gathering unknowns on the left side, these equations become:

$$\begin{aligned}
 *p_{Ext\ 1} + a_{1,3} *p_{Ext\ 3} + a_{1,4} *p_{Ext\ 4} &= *p_{o,Ext\ 1} \\
 *p_{Ext\ 2} + a_{2,3} *p_{Ext\ 3} + a_{2,4} *p_{Ext\ 4} &= *p_{o,Ext\ 2} \\
 *p_{Ext\ 3} + a_{3,1} *p_{Ext\ 1} + a_{3,2} *p_{Ext\ 2} &= *p_{o,Ext\ 3} \\
 *p_{Ext\ 4} + a_{4,1} *p_{Ext\ 1} + a_{4,2} *p_{Ext\ 2} &= *p_{o,Ext\ 4}
 \end{aligned}
 \tag{9}$$

This closed set of equations is solved, and the final values of the exterior variables are transmitted back to the separate tasks where a final substitution step generates values for all local pressure variations.

A similar series of algebraic operations is used for all equations associated with SETS stabilizer steps. Schemes can be devised that cut the number of “exterior variables”. This one was selected for relative simplicity, and associated ease of coding necessary data transfers.

The default is for the consolidated code to take responsibility for the above solution procedure. If only Consolidated Code processes contribute to the simulation, the whole solution procedure is transparent to the developer, or user. If a developer writes an application to participate in a flow solution, he or she need only pay attention to the naming conventions used by the ECI for terms in the above equations, to let the central process automatically generate the solution. However, this default mode is not always the most efficient. If one process is a detailed CFD simulation its system equivalent to Eq. 5 will be solved by an iterative procedure. In this instance the CFD process can claim control of the final solution step. It gathers the expressions in the form of Eq. 9, makes necessary substitutions into its local equations and solves for the local variables. Finally, it solves the equations for the exterior variables and distributes the results to the other processes. The ECI permits the CFD process to claim control of this solution step without being the central process for the system simulation.

Examples provided in this section reflect the current first order accurate difference methods used by the Consolidated Code. We have recognized the future option of implementing higher order numerical methods. Data structures used for the basic sparse matrix storage, and for equations passed between processes have been designed to adapt to larger primary band widths.

5. CONCLUSIONS

The USNRC's consolidated reactor safety analysis code now has solution methods and supporting code structure that permit significant run time improvements, and allow quick adoption of better solution methods as they are needed. The greatest speed advantage for users has been provided through the features permitting execution on parallel computers. However, in the important case of AP600 analysis, we have seen significant speed gains even with serial computing. We have demonstrated the adaptability of the code by installing one sophisticated off-the-shelf solution package. This SuperLU solver significantly reduces the execution time for problems with a large number of vessel

connections by better utilizing the sparsity of the linear system, and efficiently using processor specific implementations of BLAS libraries.

ACKNOWLEDGMENTS

This work has been supported by the US Nuclear Regulatory Commission.

REFERENCES

- Bertsekas, D. P., Tsitsiklis, J. N., 1989, *Parallel and Distributed Computation Numerical Methods*, Prentice Hall, NJ.
- Demmel, J. W., Gilbert, J. R., Xiaoye S. Li, SuperLU users' guide, <http://acts.nersc.gov/superlu/main.html>, Nov. 1997.
- J.J. Dongarra, I. Duff, J. DuCroz, and S. Hammarling, A set of level 3 basic linear algebra subprograms, *ACM Trans. On Math. Soft.*, 1989.
- Grant, A. and Ramsden, S., 1995, *An Introduction to High Performance Computing*, Manchester and Noth HPC T&EC.
- Kumar, V., Grama, A., Gupta, A., and Karypis, G., 1994, *Introduction to Parallel Computing Design and Analysis of Algorithms*, The Benjamin/Cummings Publishing company, Inc., RedWood City, CA.
- Mahaffy, J. H., 1982, A stability-enhancing two-step method for fluid flow calculations, *J. Comput. Phys.* **46**, 329-341.
- Mahaffy, J. H., Uhle, J., Dearing, J., Downar, T., Johns, R., and Murray, C., 2000, Architecture of the USNRC Consolidated Code, 2000, Proceeding of ICONE 8, 8th International Conference on Nuclear Engineering, Baltimore, MD USA, April 2-6, 2000.
- Mahaffy, J. H. and Uhle, J., 2000, Distributed/Parallel reactor simulations using the USNRC consolidated code, International Meeting on "Best-Estimate" Methods in Nuclear Installation Safety Analysis (BE-2000) Washington, DC, November, 2000.
- Schnurr, N. M. et al., TRAC-PF1/MOD2 Theory manual, Los Alamos National Laboratory Report LA-12031-M, US Nuclear Regulatory Commission Report, NUREG/CR-3664, 1984.
- Xiaoye S. Li, Sparse Gaussian Elimination on High Performance Computers, Ph.D. dissertation, Tech Report CSD-96-919, Computer Science, UC Berkeley, Sept. 1996.