RELAP5 User Seminar

Sept. 12-14, 2000 Jackson, Wyoming

Improved Solution of Field Equations

Arthur Shieh
Idaho National Engineering and Environmental Laboratory
Idaho Falls, ID 83415-3880
ssa@inel.gov

1.0 Introduction

The solution of the finite difference equations in RELAP5-3D, whether in the semi-implicit or nearly-implicit method, involves an algebraic reduction from the full set of governing equations and correlations to a much smaller set of equations. The smaller system is solved, and the remaining unknowns obtained with back substitution and further algebra. The algebraic reduction to the smaller system is done by straightforward Gaussian Elimination with neither preconditioning nor pivoting. However, the reduction is applied to a matrix that is often ill-conditioned. Loss of accuracy from this ill-conditioning can result in not only inaccurate solutions, but also time-step reduction. Improving this algebraic reduction with a combination of scaling and pivoting should result in greater accuracy and removal of a source of time-step reductions.

2.0 Methodology for the Improved Solution Model

2.1 Technical Background of the Model

2.1.1 Derivation of the Matrix Equation

The following is a detailed mathematical description of the semi-implicit method for time advancement. The same techniques will also be applied to the nearly-implicit method, but the details are not given here.

The finite difference form of the governing equations in RELAP5-3D for the semi-implicit method are given in Reference **1** in Equations (3.1-87) - (3.1-91), (3.1-103), and (3.1-104). Equations (3.1-87) through (3.1-91) are grouped according to a control volume as represented in Equation (3.1-115). A slight generalization of this equation is given in Equation **(1)** below; it accounts for the fact that there may not be exactly 2 junctions, j, associated with volume L. Let J be the set of junctions attached to volume L. As defined in Equation (3.1-114), x is the vector of unknown differences associated with volume L, i.e. noncondensable density, vapor energy, liquid energy, vapor void fraction, and pressure.

$$Ax = b + \Sigma_{j\varepsilon J} g_j v_{g,j}^{n+1} + f_j v_{f,j}^{n+1} \qquad (1)$$

where

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} & A_{15} \\ A_{21} & A_{22} & A_{23} & A_{24} & A_{25} \\ A_{31} & A_{32} & A_{33} & A_{34} & A_{35} \\ A_{41} & A_{42} & A_{43} & A_{44} & A_{45} \\ A_{51} & A_{52} & A_{53} & A_{54} & A_{55} \end{bmatrix} \qquad (2)$$

The entries of A, b, g, and f are given in Reference **1** in Equations (3.1-117)-(3.1-158). The combined set of finite difference equations for the N_V volumes and N_J junctions in the flow region can be written in matrix form. Let A^1 be the $(5N_V)x(5N_V)$ matrix whose L-th main diagonal block is A. Then the matrix equation is

$$A^1x^1 = b^1 + B^1v^1 \qquad (3)$$

where x^1 is the $5N_V$ length vector of unknown differences, b^1 is a $5N_V$ length vector, B^1 is a $5N_Vx2N_J$ matrix, and v^1 is the $2N_J$ length vector of phasic velocities. The semi-implicit method then applies LU factorization without pivoting to form a factorization of the matrix, $A^1 = L^1U^1$. Define L as the lower triangular matrix obtained from the identity matrix by replacing row 5k with row 5k of L^1 for k = 1, 2,....., N. Multiplying Equation **(3)** by L^{-1} changes only every fifth equation and leaves the others unchanged. The product, $L^{-1}A^1$, has every fifth row entirely filled with zeroes except for the main diagonal entry. This means that the submatrix consisting of every fifth row and column can be decoupled from the rest via a permutation of the product $L^{-1}A^1$. Let P be the permutation matrix that reduces $L^{-1}A^1$ by renumbering every fifth equation (the pressure equation) last, then

$$P(L^{-1}A^{1})P^{T} = \begin{bmatrix} M_{11} & M_{12} \\ 0 & M_{22} \end{bmatrix} \tag{4}$$

Applying these equations to Equation **(3)** produces a system of the form

$$\begin{bmatrix} M_{11} & M_{12} \\ 0 & M_{22} \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} = PL^{-1}(b^{1} + B^{1}v^{1}) \tag{5}$$

where $y = Px^{1}$. It can be solved by first solving $M_{22}y_2 = r_2$ and then back substituting $M_{11}y_1 = r_1 - M_{12}y_2$. However, the right hand side of Equation **(5)** still involves unknown velocities that must first be algebraically replaced to produce a system involving only pressures. This is done by first solving the velocity Equations (3.1-103) and (3.1-104) of Reference **1** for the velocities differences $v_{g,\,j}^{n+1}$ and $v_{f,\,j}^{n+1}$. Denote the solution by v^{1} = Hp, where p is the vector of pressures. Substituting this into block row 2 of Equation **(5)** yields a system involving only pressures.

$$Cp = (M_{22} - (PL^{-1}B^{1}H)_2)p = r_2 \tag{6}$$

2.1.2 Error Estimate for the Gaussian Elimination Methods

Suppose we are solving Ax = b. Because of the presence of round-off errors, the real matrix equation that is solved is $A(x + \Delta x) = b + \Delta b$. It is known, Reference **2**, that the error Δx satisfies

$$\frac{\|\Delta x\|}{x} < k(A)\frac{\|\Delta b\|}{b} \tag{7}$$

where $\|x\|$ denotes the maximum norm of a vector x, k(A) is the condition number of A defined to be

$$k(A) = \|A\|\|A^{-1}\| \tag{8}$$

and $\|A\|$ is the maximum norm of the matrix A. A matrix A is said to be well conditioned if k(A), which mathematically cannot be smaller than one, is not much greater than one. A matrix is said to be ill conditioned if k(A) is on the order of 10^{10} or bigger.

Since it is obvious from the inequality **(7)** that the computed error is much smaller if A is well conditioned than if it is ill conditioned, it is easy to speculate that the Gaussian elimination method will give accurate results if the matrix A is well conditioned. This, however, is not always the case as the following example from Reference **2** shows. Consider the following system Ax = b, where

$$A = \begin{bmatrix} \varepsilon & -1 & 1 \\ -1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, A^{-1} = \frac{1}{4}\begin{bmatrix} 0 & -2 & 2 \\ -2 & 1-\varepsilon & 1+\varepsilon \\ 2 & 1+\varepsilon & 1-\varepsilon \end{bmatrix} \tag{9}$$

with $|\varepsilon| << 1$. This is a well conditioned system, k(A) = 3 in the maximum norm and Gaussian elimination is expected to yield accurate results. However, the choice of $A_{11} = \varepsilon$ as pivot will have a disastrous effect on the accuracy of the solution as shown in Reference **2**. This is because the LU factorization of the Gaussian reduction process can introduce arbitrarily large errors if no pivoting is used. Hence, pivoting must be used if accuracy is desired on a consistent basis. The example in Equation **(9)** also suggests that small elements on the main diagonal for a matrix with the maximum norm of each row and column vector equal to one quite often is an indication that bad pivots exist rather than ill-conditionness is present.

On the other hand, complete pivoting is not needed as the following results from Reference **2** shows. Let $\bar{L}$ and $\bar{U}$ be the computed triangular factors of a nxn matrix A, obtained by using Gaussian elimination with partial or complete pivoting. Then if floating-point arithmetic with rounding unit u has been used, there is a matrix E satisfying

$$\|E\| < un^2 g_n \|A\| \tag{10}$$

such that

$$\bar{L}\bar{U} = A + E \tag{11}$$

Here $g_n < 2^{n-1}$ for partial pivoting and $g_n < 1.8n^{0.25\log n}$ for complete pivoting. By analyzing the rounding errors involved in solving the triangular systems that involve $\bar{L}$ and $\bar{U}$, it is possible to derive a similar result for the computed solution $\bar{x}$. Let $\bar{x}$ denote the computed solution of the system Ax = b. Then there is a matrix ΔA, depending on both A and b, satisfying

$$\|\Delta A\| < (n^3 + 3n^2) g_n u \|A\| \tag{12}$$

such that

$$(A + \Delta A)\bar{x} = b \tag{13}$$

From the above results, it is clear that for n = 5 which is the order of the matrix in Equation **(2)**, partial and complete pivoting should yield similar results and hence there is no need to use complete pivoting. Partial pivoting applied to a well-conditioned matrix will give good results. The only question is how the accuracy of the solution should be improved if the matrix is ill-conditioned.

2.1.3 Scaling of Linear Systems

As discussed in Reference **2**, in a linear system Ax = b the unknowns x_j are often physical quantities. If we change the units in which these are measured, this is equivalent to a scaling of the unknowns--say, $x_j = \alpha_j x_j'$. If we at the same time multiply the ith equation by β_i, then the original system Ax = b will be transformed into a system A'x' = b', where

$$A' = D_2AD_1, b' = D_2b, x = D_1x' \tag{14}$$

and

$$D_1 = \mathrm{diag}(\alpha_1, \alpha_2, \ldots, \alpha_n), D_2 = \mathrm{diag}(\beta_1, \beta_2, \ldots, \beta_n) \tag{15}$$

It appears that it is natural to expect that such a scaling should have no effect on the relative accuracy of the computed solution. This is in fact to a certain extent true, as the following theorem designed by F. L. Bauer on p. 181 in Reference **2** shows.

Denote by x and x' the computed solutions to the two systems Ax = b and $(D_2AD_1)x' = D_2b$. Assume that D_2 and D_1 are diagonal matrices, whose elements are even powers of the base in the number system used, so that no rounding errors are introduced by the scaling. Then, if Gaussian elimination is performed in floating point arithmetic on the two systems and if the same choice of pivots is used, all the results will differ only in the exponents, and we have exactly $x = D_1x'$.

It follows that essentially the only effect a scaling can have is to influence the choice of pivots. Now assume that we use the partial-pivoting strategy. Obviously, for any given sequence of pivots (which does not give a pivot exactly equal to zero) there exists a scaling of the equations such that partial pivoting will select these pivots. It is clear, then, that an unsuitable scaling of the equations may lead to a very poor choice of pivots.

It is therefore recommended in Reference **2** that if partial pivoting is used, then the equations should be balanced or equilibrated before the elimination. By this we mean that the matrix $A = (a_{ij})$ of the scaled system shall satisfy

$$\max_{i \le j \le n}|a_{ij}| = 1 \tag{16}$$

From Bauer's theorem, it follows that it is not necessary to perform the equilibration explicitly. Instead we can modify the partial pivoting strategy and in step k of the elimination process (Gaussian elimination process introduces zeroes to all the entries below the main diagonal up to the kth column of the reduced matrix) look for

$$\max_{k \le j \le n} \left| \frac{a_{ij}^{k}}{s_i} \right| \tag{17}$$

where

$$s_i = \max_{i \le j \le n} \left| a_{ij} \right| \tag{18}$$

While Bauer's theorem seems to suggest that scaling is of limited advantage except for influencing the choice of pivots, it is also true that if scaling by itself is helpful in reducing the condition number of the matrix, it is advantageous that the matrix is scaled first before the matrix equation is solved.

Hence, one approach is to determine the diagonal scaling matrices D_1 and D_2 so that the condition number of D_2AD_1 is minimized. However, it turns out that these optimal scaling matrices essentially depend on A^{-1}, which in practice is unknown. Another objection to this approach is that the scaling of the unknowns will change the norm in which the error is measured. Thus a sensible approach, p.183 of Reference **2**, in most cases is to choose the column scaling matrix D_1 in a way which reflects the importance of the unknowns and to use D_2 to equilibrate the matrix.

2.1.4 Improvement of the Accuracy of Solution for Ill-conditioned Systems by the Iterative Refinement Method

As discussed in Reference **2**, we have seen that when A is ill-conditioned, the computed solution $\tilde{x}$ may be inaccurate without any indication in the form of a large solution vector. Hence, it is important to have a good estimate of the condition number of matrix A.

Obviously, A^{-1} can be computed so that the condition number of matrix A can be estimated directly. If the condition number of matrix A is small, then the computed A^{-1} is also close to the true A^{-1}. The problem is that the computation of A^{-1} is n times more expensive than the LU factorization of the matrix A where n is the order of the matrix. Also if the condition number of matrix A is large, then the computation of A^{-1} is not accurate either. Hence, we propose an alternative method that is much cheaper and more reliable when A is ill-conditioned. Moreover, the method improves the accuracy of the solution while at the same time gives an estimate of the condition number.

Suppose r = b - A$\tilde{x}$ is the residual vector to a computed solution $\tilde{x}$, then

$$A(x - \tilde{x}) = r \tag{19}$$

Now assume that Gaussian elimination has given the approximate triangular factors $\tilde{L}$ and $\tilde{U}$. From Equation **(11)**, we know that $\tilde{L}\tilde{U} = A + E$, where E is small. We can therefore approximate the correction x - $\tilde{x}$ with the solution to

$$\tilde{L}\tilde{U}(\Delta x) = r \tag{20}$$

This only takes $2n^{-}$ operations and is much cheaper than the computation of A^{-1} for large n (n in the test case is only five although it may expand to fourteen when more terms of the Taylor series expansion in the state variables are used).

New rounding errors are introduced in the computation of Δx, and $\tilde{x} + \Delta x$ may not be a more accurate solution than $\tilde{x}$. A more detailed analysis shows that, because of the cancellation which will take place in the computation of the residual vector r, it is essential that this vector be computed with sufficient accuracy. It is often advisable to proceed as follows. The components in r are

$$r_i = b_i - \Sigma_{k=1}^{n} a_{ik}\tilde{x}_k, \; i = 1, \ldots, n \tag{21}$$

If a_{ik} and $\tilde{x}_k$ are given with t digits, then the products $a_{ik}\tilde{x}_k$ contain at most 2t digits. We compute these products exactly and accumulate the sum using 2t digits. Finally, r_i is computed and rounded to t digits. This can be done very conveniently on most computers and will ensure that the error from this part of the calculation is small.

The improved solution $\tilde{x} + \Delta x$ can, of course, be corrected in the same way and hence the name iterative refinement. Unless the condition number is very large, one or two refinements are usually sufficient. The condition number can be estimated by

$$\kappa(A) \le \frac{1}{nu}\frac{\|\Delta x\|}{\|x + \Delta x\|} \tag{22}$$

where n is the order of the matrix and u is the rounding unit, (see p. 184 of Reference **2**). As long as the condition number satisfies

$$\kappa(A) \le \frac{0.1}{nu} \tag{23}$$

then the above procedure works (Reference **2**). If Inequality **(23)** is not satisfied, then enhanced precision should be used throughout the calculation to reduce the rounding unit u so that Inequality **(23)** is satisfied.

2.2 Enhanced Precision in the Matrix Solver

The dynamically dimensioned array is first mapped to a locally defined two-dimensional array in enhanced precision where the Gaussian elimination process and the forward and backward substitution is carried out. The scaled column pivoting algorithm is implemented in enhanced precision to eliminate any effect ill-conditionness may have in the solution of the matrix equation.

2.3 Choice of Algorithm

For problems that satisfy Inequality **(23)**, the scaled column pivoting algorithm with iterative refinement is used. Otherwise, the algorithm described in section

2.2 is used.

.

2.4 Implementation

2.4.1 Changes Required for the Semi-Implicit Scheme

Software implementation for the semi-implicit scheme is done primarily in the subroutine PRESEQ. Both partial pivoting and scaling will be introduced there. Also the iterative refinement process will be implemented in FORTRAN 90. For completeness, some details of implementation are given below.

In PRESEQ, the matrix A was first factorized and then the last row of A inverse was computed and stored in the last row of A. The elements of A are stored in dynamic scratch space, a11(ix), a12(ix),..., etc. Here ix is the volume index for a given volume. In order for Gaussian elimination with partial pivoting and equilibrating (or scaled column pivoting) to work efficiently, it is important that the matrix at the volume indexed with ix be first mapped to a 5x6 matrix locally. A published algorithm is then used to perform the Gaussian elimination with partial pivoting and equilibrating for the 5x6 locally dimensioned matrix.

The algorithm is applied to the matrix equation $A^T x = e_5$, where $e_5 = (0, 0, 0, 0, 1)^T$, the solution of which gives the last row of A inverse.

The advantage of this approach is that indirect addressing can be used to simulate row interchanges (see the algorithm below). For computers with parallel processors, several locally dimensioned matrices may be needed. Instead the algorithm is coded in a subroutine called by PRESEQ with ix as the main argument. Each thread is then assigned a different ix. For maximum efficiency, PRESEQ should contain only two or three such subroutines or loops where ix is incremented. The following algorithm is taken from Reference **3**. This choice is made mainly because of the clarity of the algorithm and there is a good one to one correspondence between each step of the algorithm and the published software in Reference **4**. In the following algorithm, the n x (n+1) matrix A = (a_{ij}) is set to be the transpose of the matrix aij(ix) in PRESEQ, i.e. a_{ij} = aji(ix), and the right hand side vector is stored in the (n+1)th column of A. The LU factorization is carried out in steps 1-7 below with L stored in the lower triangular part of the matrix.

Algorithm for Gaussian elimination with partial pivoting and equilibrating

Step 1. For i= 1,...,n set $s_i = \max_{1 \le j \le n} |a_{ij}|$; set NROW(i) = i

Step 2. For i= 1,...,n-1 do steps 3-5. (Elimination process)

Step 3.Let p be the smallest integer with $i \le p \le n$ and

$$\frac{|a(NROW((p), i))|}{s(NROW(p))} = \max_{i \le j \le n} \frac{|a(NROW((j), i))|}{s(NROW(j))}$$

Step 4.If NROW(i) not = NROW(p), then set NCOPY = NROW(i); NROW(i) = NROW(p); NROW(p) = NCOPY. (simulated row interchanges)

Step 5.For j=i+1,....,n do steps 6 and 7.

Step 6.Set m(NROW(j),i) = a(NROW(j),i)/a(NROW(i),i)

Step 7.Perform $E_{NROW(j)} - m(NROW(j), i)E_{NROW(i)} \rightarrow E_{NROW(j)}$

Step 8. Set x_n = a(NROW(n),n+1)/a(NROW(n),n) (Start backward substitution)

Step 9. For i = n-1,.....,1, set

$$x_i = \frac{a(NROW(i), n+1) - \Sigma_{j=i+1}^{n} a(NROW(i), j)x_j}{a(NROW(i), i)}$$

Step 10.Estimate the condition number of the matrix A relative to the nth component of the solution by computing

$$\kappa(A)_n = n(\max_{i=1}^{n} s_i)(\Sigma_{i=1}^{n} |x_i|)$$

Step 11.If $\kappa(A)_n < 10^7$, stop. Otherwise, if $\kappa(A) < 10^{13}$, compute the residual vector $r = e_n - Ax$ where x is the computed solution, and e_n is the nth unit vector. (Start iterative refinement process)

Step 12.Solve Az = r by using the factored matrix to do forward and backward substitution; set $z_1 = r_1$. (Start forward substitution process)

Step 13.For i = 2,...,n, set

$$z_{NROW(I)} = a(NROW(i), n+1) - \Sigma_{j=1}^{i-1} a(NROW(i), j)z_j$$

Step 14.Repeat steps 8 and 9. Compute x = x + z. Stop.

If the condition number of the matrix A is bigger than 1.0e13, then only do steps 1 through 9 in the above algorithm in enhanced precision.,

2.4.2 Changes Required for the Nearly-Implicit Scheme

Software implementation for the nearly-implicit scheme is done primarily in the subroutine VIMPLT. Both partial pivo ting and scaling will be introduced there. The algorithm is applied to the matrix equation $A^T x = e_5$, where $e_5 = (0, 0, 0, 0, 1)^T$, the solution of which gives the last row of A inverse and to the matrix equation $A^T x = e_4$, where $e_4 = (0, 0, 0, 1, 0)^T$, the solution of which gives the fourth row of A inverse.

3.0 Developmental Verification Problems and Results

The test problems include the Edwards pipe problem and the typical PWR prob lem. Because the equations $A^T x = e_5$ and $A^T x = e_4$ are solved for the nearly implicit scheme, a good method of measuring the accuracy of the methods is to compute the combined cumulative sum of the Euclidean norm of the residuals for both equations. The combined cumulative norms for both the old and the new methods of solving the matrix equation are compared in **Figure 3** and **Figure 4** for both the Edwards pipe problem and the typical PWR problem.

Because the equation $A^T x = e_5$ is solved for the semi-implicit scheme, a good method of measuring the accuracy of the method is to compute the cumulative sum of the Euclidean norm of the residuals. The cumulative norms for both the old and the new methods of solving the matrix equation are compared in **Figure 3** and **Figure 4** for both the Edwards pipe problem and the typical PWR problem. The new method generally computes a smaller residual. The number of advancements required by the new method for the typical PWR problem is 4310. The number of advancements required by the old method is 4720. Both methods use the same number of advancements for the Edwards pipe problem. The time history of the pressures computed by both methods are aboout the same.

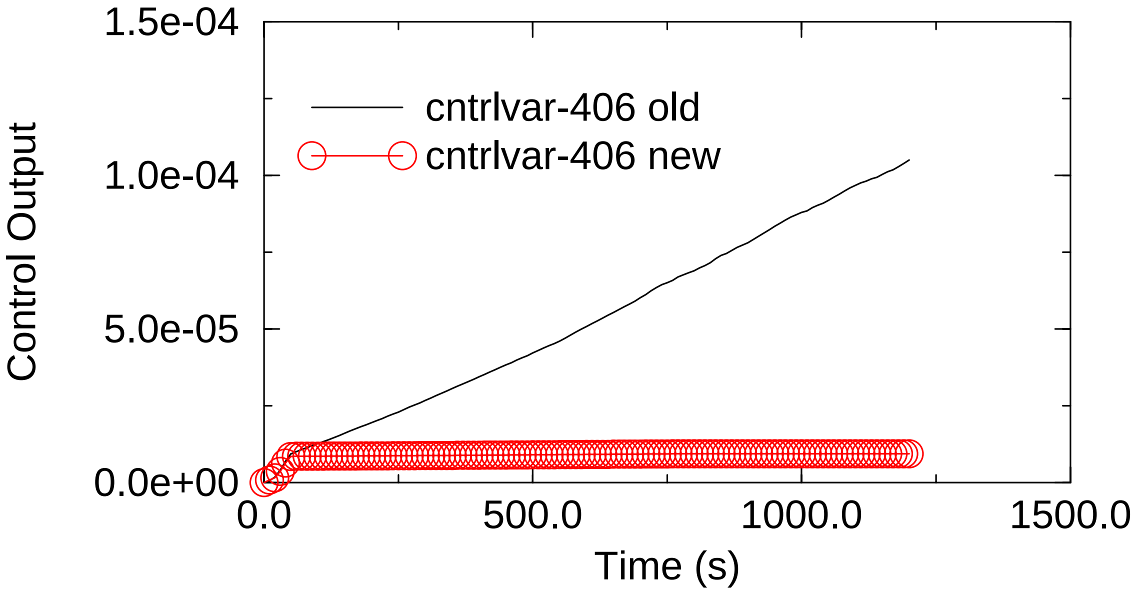

Figure 1 Comparison of the cumulative sum of the combined norms of the residualls for the typ1200n.i problem

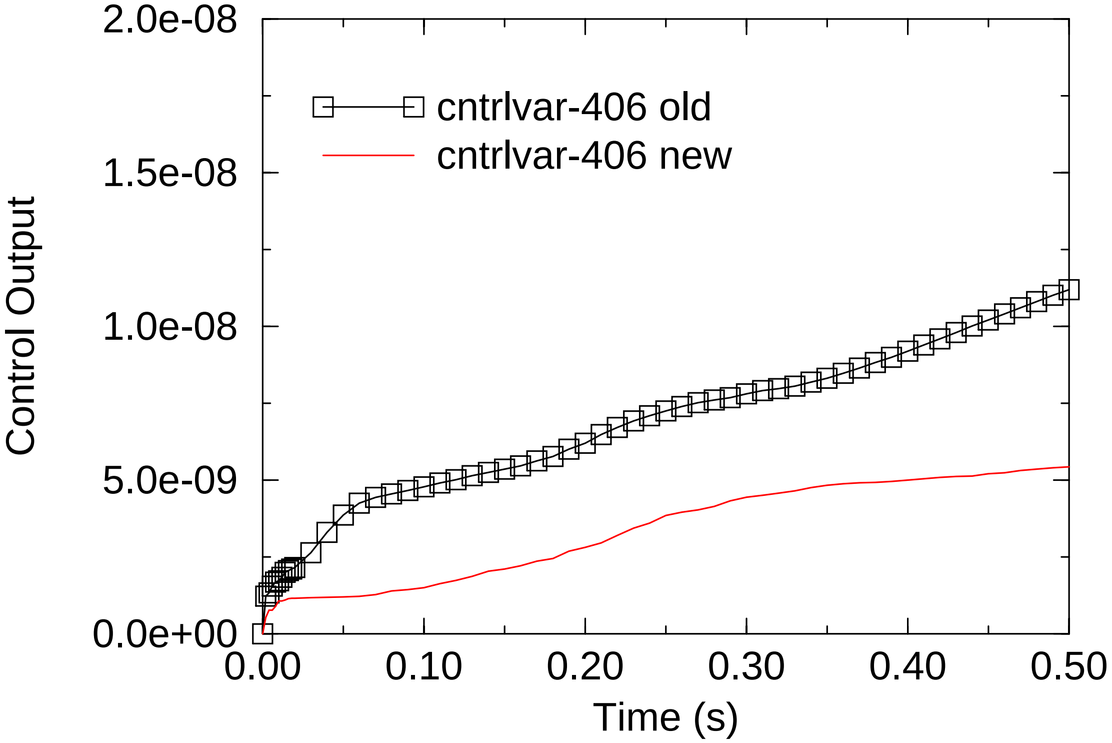

Figure 2 Comparison of the cumulative sum of the norm of the residualls between the old and the new methods for the nearly-implicit scheme for the Edwards pipe problem

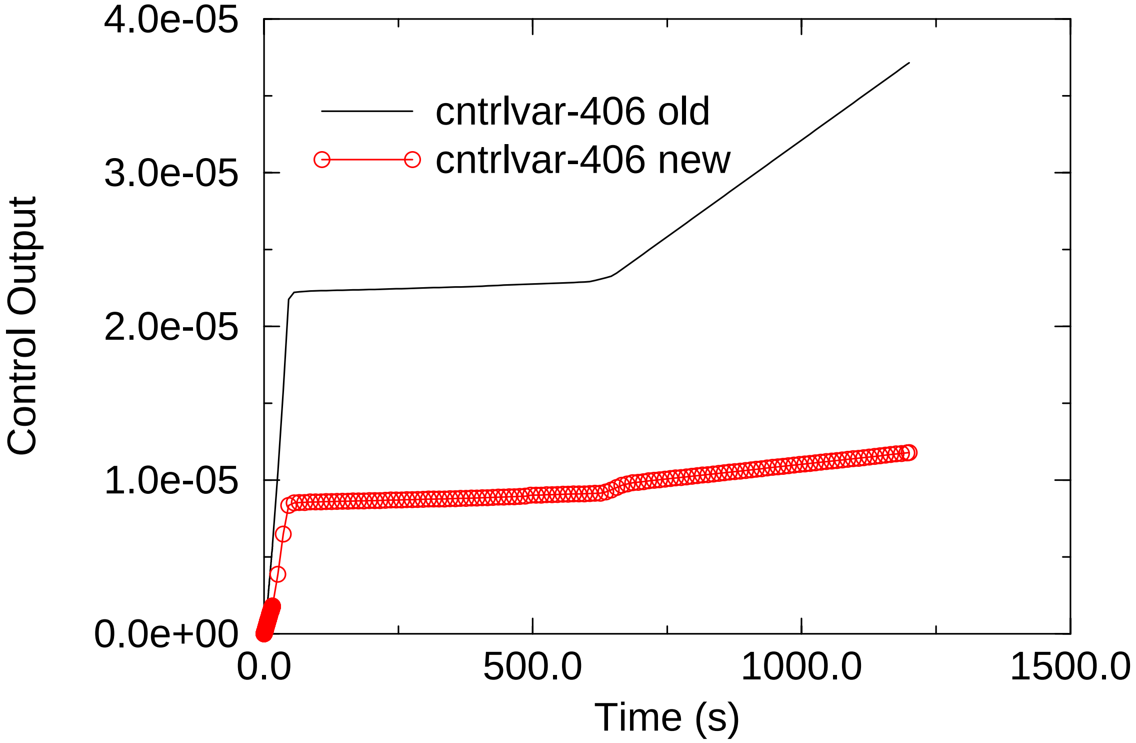

Figure 3 Comparison of cumulative sum of norm of residuals for the typ1200.i problem for the semi-implicit scheme.

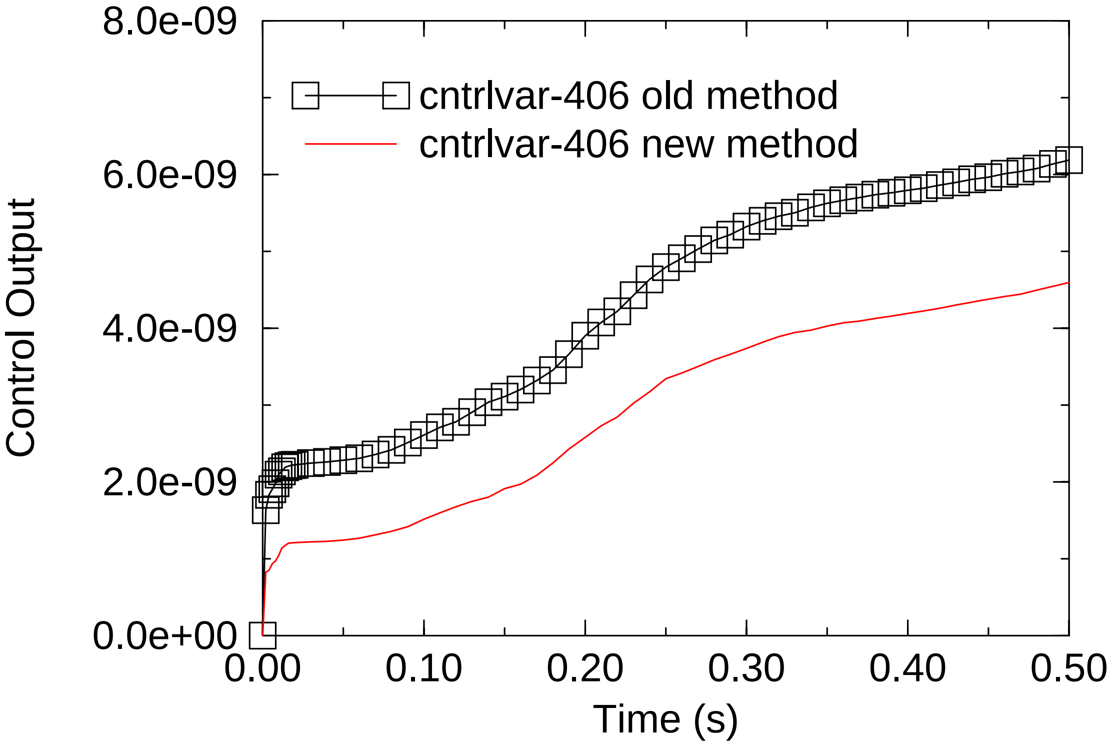

Figure 4 Comparison of cumulative sum of norm of residuals for the Edwards pipe problem for the semi-implicit scheme.

4.0 References

1 The RELAP5-3D Development Team, “RELAP5-3D Manual”, INEEL-EXT-98-00834, Revision 1.1b, July 1999.

2 G. Dahlquist, A. Bjorck, and N. Anderson, “Numerical Methods”, Prentice-Hall, New Jersey, 1976.

3 R. Burden, J. Faire, and A. Reynolds, “Numerical Analysis”, Prindle, Weber, and Schmidt, 1981.

4 R. Burden, J. Faire, and A. Reynolds, “Teacher’s Manual for Numerical Analysis”, Prindle, Weber, and Schmidt, 1981.