Proceedings of the 9th WSEAS International Conference on Applied Mathematics, Istanbul, Turkey, May 27-29, 2006 (pp121-127)

UNSYMMETRICAL AND SYMMETRICAL SPARSE ITERATIVE ALGORITHM WITH MULTIPLE RIGHT-HAND - SIDES STRATEGIES

D.T. NGUYEN^, A.P. HONRAO", G. HOU", O. AKAN^, and O. BAYSAL~
^Civil & Env. Engineering Dept. 1319 ECSB, "Mechanical Engineering Dept. 238 KAUF, ~Aerospace Engineering Dept. 102 KAUF
Old Dominion University
Norfolk, VA-23508
USA

Abstract: - Unified unsymmetrical and symmetrical iterative solvers for handling multiple right-hand-side vectors are examined in this work. Efficient computer implementation strategies (to reduce computational time and in-core memory requirements) are proposed. In-core, out-of-core, linear, multiple right hand side (RHS) vectors, non-linear, symmetrical, and unsymmetrical capabilities of the developed software are demonstrated by solving variety of problems selected form different engineering disciplines. Results indicate that the developed algorithm and software is reliable and efficient.

Key-Words: - Sparse, Iterative, Linear, Nonlinear, Conjugate Gradient, Multiple RHS vectors.

1 Introduction

For large-scale engineering and science problems encountered in practical applications, finite element analysis (FEA) procedures have been widely accepted by the engineering communities as the efficient tool for obtaining the solution. The most time consuming (and computer memory requirement) part of the FEA procedures is to solve the large, discretized system of linear equations. Recent research works seem to indicate that FEA procedures, embedded inside the framework of Domain Decomposition (DD) formulation, where mixed "direct, sparse", and "iterative" algorithms are used, can exploit the computational efficiencies offered by modern parallel processing computer hardware. In this research work, the Lingen's unsymmetrical iterative algorithm [1] for efficient handling multiple right-hand-side (RHS) vectors is re-examined, and modified, so that both "unsymmetrical and symmetrical" system of linear equations [2, 6] can be handled in a sparse matrix environments. To validate the modified algorithms, and to evaluate the computational efficiency, applications in Structural, and Acoustic disciplines are discussed in this paper.

2 Generalized Conjugate Residual (GCR) Iterative Algorithms with Successive Right Hand Side Vectors

To keep the discussion more general, our objective here is to solve the system of "unsymmetrical," linear equations, which can be expressed in the matrix notations as:

$$[A]\,\vec{x} = \vec{b} \tag{1}$$

where [A] is a given N×N unsymmetrical matrix, and $\vec{b}$ is a given, single right-hand-side vector. System of "symmetrical" linear equations can be treated as a special case of "unsymmetrical" equations. However, more computational efficiency can be realized if specific algorithms (such as the Conjugate Gradient algorithms) which exploit the symmetrical property are used. Successive right-hand-side vectors will be discussed near the end of this section.

Solving Eq. (1) for the unknown vector $\vec{x}$ is equivalent to either of the following optimization problem:

$$\text{Minimize} \quad \psi_{1} \equiv \frac{1}{2} x^T A\, x - x^T b \qquad x \in R^N \tag{2}$$

or

$$\text{Minimize} \quad \psi_{2} \equiv (A\,\vec{x} - \vec{b})^T * (A\,\vec{x} - \vec{b}) \qquad x \in R^N \tag{3}$$

Equation (2) is preferred for the case matrix [A] is "symmetrical", while Eq. (3) is suggested if matrix [A] is "unsymmetrical".

The minimization problem, described by Eq. (2) or Eq. (3), can be iteratively solved by the following step-by-step procedures as indicated in Table 1.

Table 1 Step-by-Step Iterative Optimization Procedures

Step 0 Initial guessed vector:

$\vec{x} = \vec{x}^{(0)}$; and set iterarion count i = 0 (4)

Step 1 Set i=i+1 at the current design point, find the "search direction" to travel, $s^{(i-1)}$?

Step 2 Find the step-size, α(how far should we travel along a given direction $s^{(i-1)}$)?

Step 3 Find the updated, improved design, $x^{(i)}$?

$$x^{(i)} = x^{(i-1)} + \alpha s^{(i-1)} \quad (5)$$

Step 4 Convergence test

Convergence is achieved if :

$$\left\|\nabla \psi_1^T\right\| \le \text{Tolerance} \quad (6)$$

$$\left\|\nabla \psi_2^T\right\| \le \text{Tolerance}$$

and / or

$$\left\|x^{(i)} - x^{(i-1)}\right\| \le \text{Tolerance} \quad (7)$$

If convergence is achieved (or, iteration # i=max. # of iterations allowed) then stop the process.

Else

Return to step1

End if

In Table 1, the 2 most important steps are to find "the search direction" $s^{(i-1)}$ to travel from the current design point (see Step 1), and to find "the step size" α (see Step 2).

2.1. How to find the "Step Size", α, along a Given Direction $\vec{s}$??

Assuming the search direction $\vec{s}^{(i)}$ has already been found, then the new, improved design point can be computed as (see Step 3 of Table 1):

$$x^{(i+1)} = x^{(i)} + \alpha s^{(i)} \quad (8)$$

Thus, Eqs. (2 – 3) become minimization problems with only 1 variable (=α), as following:

$$\text{Min. } \psi_1 = \frac{1}{2}\left(x^i + \alpha_1 s^i\right)^T A\left(x^i + \alpha_1 s^i\right) - \left(x^i + \alpha_1 s^i\right)^T b \quad (9)$$

or

$$\text{Min. } \psi_2 = \left[A\left(x^i + \alpha_2 s^i\right) - b\right]^T * \left[A\left(x^i + \alpha_2 s^i\right) - b\right] \quad (10)$$

In order to minimize the function values ψ_1 (or ψ_2), one needs to satisfy the following requirements:

$$\frac{d\psi_1}{d\alpha_1} = 0 \quad (11)$$

or

$$\frac{d\psi_2}{d\alpha_2} = 0 \quad (12)$$

and

$$\alpha_2 = \frac{-\left(s^i\right)^T A^T \left(r^i\right)}{\left(s^i\right)^T A^T A\, s^i} \quad (13)$$

2.2. How to find the "Search Direction," s^i?

The initial direction, s^0, is usually selected as the initial residual:

$$s^0 \equiv -r^0 = -(A\, x^0 - b) \quad (14)$$

The reason for the above selection of the initial search direction was because Eq. (14) represents the gradient $\nabla\psi_1(x^0)$, or "steepest descent direction," of the objective ψ_1, defined in Eq.(2).

The step size is selected such that

$$\frac{d\psi_1(x^{i+1})}{d\alpha_1} = 0 = \frac{d\psi_1(x^{i+1} + \alpha_1 s^i)}{d\alpha_1} = \frac{d\psi_1}{dx} * \frac{dx}{d\alpha_1} \quad (15)$$

$$0 = \nabla\psi_1(x^{i+1}) * s^i \quad (16)$$

or

$$\frac{d\psi_2(x^{i+1})}{d\alpha_2} = 0 = \frac{d\psi_2(x^{i+1} + \alpha_2 s^i)}{d\alpha_2} = \frac{d\psi_2}{dx} * \left(\frac{dx}{d\alpha_2} = s^i\right) \quad (17)$$

The above equation becomes:

$$\left(s^i\right)^T A^T * \left[A\left(x^i + \alpha_2 s^i\right) - b\right] = 0 \quad (18)$$

or

$$\left(s^i\right)^T A^T * \left[A\left(x^{i+1}\right) - b\right] = 0 \quad (19)$$

$$\left(s^i\right)^T A^T * \left[r^{i+1}\right] = 0 \quad (20)$$

Since Eq. (20) represents the "scalar quantity," hence it can also be presented in its "transposed" form, as following:

$$\left[r^{i+1}\right]^T * A\, s^i = 0 \quad (21)$$

Comparing Eq. (21) with Eq. (17), one concludes:

$$\nabla\psi_2 \equiv \frac{\partial\psi_2}{\partial x} \equiv \left[r^{i+1}\right]^T * A \quad (22)$$

Thus, Eq.(21) can also be presented as:

$$\nabla\psi_2(x^{i+1}) * s^i = 0 \quad (23)$$

One can built a set of "A conjugate" vectors (=s0.s1. .s$(= s^0, s^1, \ldots, s^i, s^{i+1})$ by applying the Gram-Schmidt procedures to the (new) residual vector:

$$\hat{r}^{i+1} = A\,x^{i+1} - b = 0 \quad (24)$$

for obtaining the search direction

$$\hat{s}^{i+1} = \hat{r}^{i+1} + \sum_{k=0}^{i} \beta_k\, \hat{s}^{k} \quad (25)$$

where

$$\beta_k = \frac{-\hat{r}^{i+1} A\, \hat{s}^{k}}{\left(\hat{s}^{k}\right)^T A\, \hat{s}^{k}} \quad (26)$$

and the following property of "A conjugate" vectors will be satisfied:

$$\left(\hat{s}^{i+1}\right)^T A\, \hat{s}^{k} = 0 \quad ; k = 0,1,2,L\ ,i \quad (27)$$

Polak-Rebiere Algorithm for β_i becomes:

$$\beta_i = \frac{\left(\hat{r}^{i+1}\right)^T * (\hat{r}^{i+1})}{\left(\hat{r}^{i}\right)^T * (\hat{r}^{i})} = \text{ Fletcher-Reeves Algorithms} \quad (28)$$

If [A] is an "unsymmetrical" matrix, then Reference [1] suggest to use "[A^T A] conjugate" vectors as:

$$\hat{s}^{i+1} = \hat{r}^{i+1} + \sum_{k=0}^{i} \beta_k\, \hat{s}^{k} \quad (29)$$

where

$$\beta_k = \frac{\left(-A\,\hat{r}^{i+1}\right)^T \left(A\,\hat{s}^{k}\right)}{\left(\hat{s}^{k}\right)^T A^T A \left(\hat{s}^{k}\right)} \quad (30)$$

with the following "Conjugate like" property:

$$\left(A\,\hat{s}^{i+1}\right)^T \left(A\,\hat{s}^{k}\right) = 0 \quad (31)$$

2.3. The Generalized Conjugate Residual (GCR) algorithm:

Based upon the discussions in previous sections, the Generalized Conjugate Residual (GCR) algorithms can be described in the following step-by-step procedures

Table 2 GCR Step-by-step Algorithms

Step 1 Choose an initial guess for the solution, x^0
Compute $r^0 = b - A\,x^0$

Step 2 Start optimization iteration, for j=1, 2...
Choose an initial search direction, $\hat{s}^{j}$
where $\hat{s}^{j} = r^{j-1}$ (32)
Compute $\hat{v}^{j} = A\,\hat{s}^{j}$; (portion of Eq.13)

Step 3 Generate a set of conjugate vectors $\cdot^{j}$,
The Gram-Schmidt Process

for $k = 1,2,L\ j-1$

$\beta = \left(\hat{v}^{j}\right)^T \left(v^{k}\right)$; see Eq.(30) (33)

$\hat{v}^{j} = \hat{v}^{j} - \beta\, v^{k}$ (34)

$\hat{s}^{j} = \hat{s}^{j} + \beta \hat{s}^{k}$; see Eq.(29) (35)

End for

Step 4 Normalize the vectors

$\gamma = \left(\hat{v}^{j}\right)^T \left(\hat{v}^{j}\right)$; complete the denominator of Eq.(13) (36)

$v^{j} = \frac{\hat{v}^{j}}{\gamma}$; "almost" completing Eq.(13) (37)

$s^{j} = \frac{\hat{s}^{j}}{\gamma}$ (38)

Step 5 Compute the step size

$\alpha = \left(r^{j-1}\right)^T \left(v^{j}\right)$; (39)

Step 6 Compute the updated solution

$x^{j} = x^{j-1} + \alpha\, s^{j}$ (See Eq.8)

Step 7 Update the residual vector

$r^{j} = r^{j-1} - \alpha\, v^{j}$ (40)

Step 8 Convergence check?

If $\left[\left\|r^{j}\right\| \Big/ \|b\|\right] \le \varepsilon_{Tol}$; Then (41)

Stop

Else

$j = j + 1$

Go to Step 2

2.4 How to Efficiently Handle Successive Right-Hand-Side Vectors?

Assuming we have to solve for the following problems:

$$[A]\vec{x}_i = \vec{b}_i \quad (42)$$

In Eq. (42), the right-hand-side (RHS) vectors are NOT all available at the same time, but that $\vec{b}_i$ depends on $\vec{x}_{i-1}$. There are 2 objectives in this section, which will be discussed in subsequent paragraphs:

(1) Assuming the 1st solution $\vec{x}_1$, which corresponds to the 1st RHS vector $\vec{b}_1$ has already been found in "n_1" iterations. Here, we would like to utilize the first "n_1" generated (and orthogonal)

vectors $s_{i=1}^{j=1,2,L\ ,n_1}$ to find "better initial guess" for the 2nd RHS solution vector $\vec{x}_{i=2}$. The new, improved algorithms will select the initial solution $\vec{x}_{i=2}^{0}$, to minimize the errors defined by ψ_1 (see Eq.2), or ψ_2 (see Eq.3) in the vector space spanned by the already existing (and expanding) "n_1" conjugate vectors. In other words, one has:

$$\vec{x}_{i=2}^{0} \equiv [s_1^1, s_1^2, L\ , s_1^{n_1}]_{n\times n_1} * \{P\}_{n_1\times 1} = [S_1] * \{P\} \quad (43)$$

Eq.(43) expresses that the initial guess vector $\vec{x}_2^{0}$ is a linear combinations of columns $s_1^1, s_1^2, L\ , s_1^{n_1}$. By minimizing (with respect to {P}) ψ_2 (defined in Eq.3), one obtains:

$$\nabla_p(\psi_2) \equiv \frac{\partial \psi_2}{\partial \vec{P}} = \vec{0} \quad (44)$$

which will lead to:

$$\{P\}_{n_1\times 1} = \frac{[s_1^T]_{n_1\times n} * [A^T]_{n\times n} * \{\vec{b}_2\}_{n\times 1}}{[s_1^T][A^T][A][s_1]} \quad (45)$$

For the "symmetrical" matrix case, one gets:

$$\nabla_p(\psi_1) \equiv \frac{\partial \psi_1}{\partial \vec{P}} = \vec{0} \quad (46)$$

which will lead to:

$$\{P\} = \frac{[s_1^T] * \{\vec{b}_2\}}{[s_1^T][A][s_1]} \quad (47)$$

The step-by-step algorithms to generate "good" initial guess $\vec{x}_2^{0}$ for the 2nd RHS vector can be given as shown in Table 3.

Table 3 Step-by-step Algorithms to Generate "Good" Initial Guess for RHS Vectors

$$\vec{x}_2^{0} = \vec{0} \quad (48)$$

$$\vec{r}_2^{0} = \vec{b}_2 \quad (49)$$

$$\text{for } k = 1, 2, L\ , n_1 \quad (50)$$

$$P = \left(r_2^0\right)^T \left(v_1^k\right) \quad (51)$$

$$x_2^0 = x_2^0 + P\, s_1^k \quad (52 \text{ also see Eq.43})$$

Recalled: $v_1^k = A\, s_1^k$ (see Eq.32, where s_1^k has already been normalized according to Eq.38). Furthermore, kept in mind that v_1^k and s_1^k vectors had already been generated and stored when the 1st-RHS had been processed.

Thus, Eq.(51) represents the implementation of Eq.(45), but corresponding to "ONLY 1 column" of matrix $[S_1]$. That's why we used to have a "do loop index k" (see Eq.50) to completely execute Eq.(45) & Eq.(43).

$$r_2^0 = r_2^0 - P\, v_1^k \quad (53)$$

Eq.(53) can be derived as following :

First, pre-multiplying both sides of Eq.(52) by (-A), one obtains (note :P=scalar, in Eq.51) :

$$(-A)\, x_2^0 = (-A)\, x_2^0 + P\,(-A)\, s_1^k$$

Then, adding (b_2) to both sides of the above equation, one gets:

$$-A\, x_2^0 + b_2 = -A\, x_2^0 + b_2 - P\left(A\, s_1^k\right)$$

or

$$r_2^0 = r_2^0 - P\left(A\, s_1^k\right)$$

or, referring to Eq.(32), then the above Eq. becomes:

$$r_2^0 = r_2^0 - P\left(v_1^k\right)$$

which is the same as indicated in Eq.(53).

End for

 # (2) For successive RHS vectors, the "search vectors" s^j (see Eq.35, in Table 2) need to be modified, so that these vectors s^j will not only orthogonal amongst themselves (corresponding to the current 2nd RHS vector), but they also will orthogonal with the existing n_1 vectors [corresponding to ALL "previous" RHS vector(s)]. Thus, the total number of (cumulative) conjugate vectors will be increased, and hence "faster convergence" in the GCR algorithm can be expected. Obviously, there will be a "trade-off" between "faster convergence rate" versus the "undesired" increase in computer memory requirement. In practical computer implementation, the user will specify how many "cumulative" vectors s^j and v^j that can be stored in RAM. Then, if these vectors s^j and v^j have already filled up the user's specified incore memory available and convergence is still NOT YET achieved, one may have to "restart" the process (by starting with the new initial guess etc ...), or using the out of core strategies. In this work, out of core strategies are used. The amount of work and the amount of memory that is required are proportional to the total number of search vectors that have been generated. To make the extended GCR algorithm competitive with direct solution algorithms, it is essential to keep the total number of generated search vectors as small as possible. This can be achieved by preconditioning the system of equations (42). Another way to reduce the amount of work and the memory is to use the extended GCR algorithm in

combination with the Domain Decomposition (DD) approach [3-5].

3 Numerical Applications

Example 1: Two dimensional Frame structure (small change at all locations for consecutive RHS vectors, but dimension of the problem is 1200x1200).

The basic system has five nodes in horizontal and five nodes in vertical direction. The corresponding nodes are connected with vertical and horizontal members as shown in the Figure 1. Since 2-D frame elements are used in this example, each node has three degree's of freedom: two translational (T_x,T_y) and one rotational (θ_z), dof. This basic structure, therefore, has 75 degree's of freedom. All the nodes in the bottom floor are fixed support (Dirichlet type) boundary conditions.

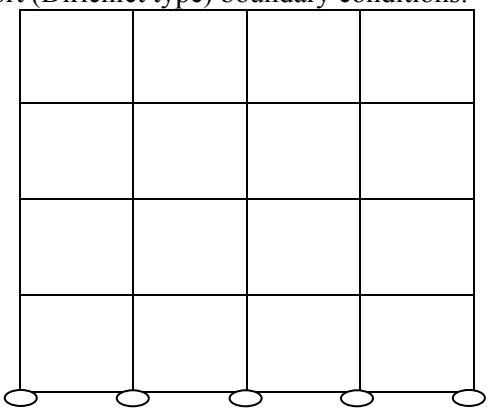

Figure 1: A typical 2-D frame structure (5x5 grid points)

The length of each element is 10 in., Young's modulus is $2x10^9$ Psi. Moment of inertia of the cross-section is $1x10^{-10}$ mm^4 and the area of cross section is $1x10^{-3}$ mm^2. Multiple right hand sides or the several load cases are applied on the frame structure.

In this example, the size of the 2-D frame structure (see Figure 1) is increased to 400x400 grid points, in each of the x and y directions. This structure, therefore, has 1200 degrees of freedom. Three right hand sides are used in this example. The unified GCR proposed in the paper converges in {844,13,14} iterations, corresponding to the right hand side vectors {1,2,3}, respectively. The absolute error norms for the solutions are {8.908E-12, 5.618E-12, 6.492E-12}, respectively.

Example 2: Aero-acoustics Application (400 degree of freedom, 2-D un-symmetric system) [5]

The developed algorithm will be exercised to study the propagation of acoustic pressure waves in a three-dimensional duct lined with sound absorbing materials (acoustic liners) as depicted in Figure 2.

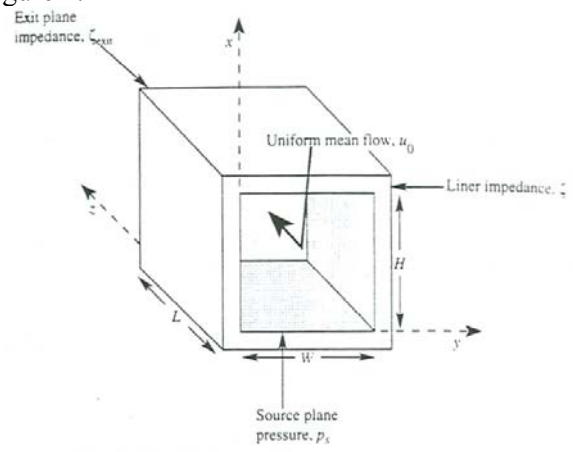

Figure 2: Aero-acoustics application

The duct is spanned by an axial coordinate z, transverse coordinate x, and span wise coordinate y. the source plane is located at z = 0, and the source plane acoustic pressure p_s is assumed to be known. At the exit plane the dimensionless exit acoustic impedance ζ_{exit} is assumed to be known. In the duct, air is flowing along the positive z axis at a sub-sonic speed of $u_0$and the duct has acoustic liners along its upper, lower and the two side walls. The duct walls are assumed to be locally reacting so that the absorbing properties of the acoustic liners results from the dimensionless wall impedance ζ that is assumed to be known. The sound source pressure, dimensionless wall impedance is assumed functions of position along their respective boundaries.

In this example the finite element system of 400 degree of freedom, unsymmetrical (complex numbers) system, and five generated right hand side vectors are solved in the similar fashion as for the previous case. The unified GCR algorithm converges in {218,46,19,14,16} iterations, corresponding to the right hand side vectors {1,2,3,4,5}, respectively. The absolute error norms for the solutions are {2.182E–4, 3.469E–4, 5.836E–4, 9.054E–4, 8.778E–4}, respectively.

Example 3: Non linear system of equations (100 degree of freedoms system)

In this example, Newton-Raphson and the unified GCR algorithm are used to solve a system of non-linear equations. The system is based upon a building block of the following three nonlinear equations.

$$\mathbf{F} \equiv \begin{Bmatrix} \mathbf{F}_1(x_1,x_2,x_3) \\ \mathbf{F}_2(x_1,x_2,x_3) \\ \mathbf{F}_3(x_1,x_2,x_3) \end{Bmatrix} = \begin{Bmatrix} 3x_1^3+3x_1-\cos(x_2x_3)-0.5 \\ x_1^2+81(x_2^3+0.1)+81x_2+\sin(x_3)+1.06 \\ e^{(-x_1x_2)}+20x_3^3+20x_3+(10\text{pi}-3)/3 \end{Bmatrix}$$

where x_1,x_2,x_3 are the local variables. Then three integers are randomly generated in order simulate elements of a larger non-linear system. Then the local variables are converted into global variables and the system is built to get the 100 non-linear equations.

Once the F vector is assembled, we calculate the tangent stiffness matrix and solve for the unknown displacement-vector, using the Newton-Raphson method. The increment in the **x** i.e. **dx** is solved using the unified GCR algorithm. The 100 non-linear equations problem was converged in 6 Newton-Raphson iterations.

Table 4: GCR and Newton-Raphson Algorithms to solve 100 non-linear equations

N-R iteration no.	N-R error	Unified GCR iterations	GCR residue
1	2.7992	29	388e-005
2	0.3444	26	4.2716e-005
3	0.0242	26	6.0358e-005
4	9.61e-4	21	9.6438e-005
5	1.72e-4	16	9.7629e-005
6	3.78e-5	16	7.6250e-005

Example 4: GCR with Out-of-core Strategies for cumulative vectors:

Out of core strategies used: The user will specify how much space (or in-core computer memory) is available. The storage scheme will be generated accordingly. The algorithm is written in such a way that it divides the available computer memory into two parts (i.e. two matrices). One matrix is used for in-core available vectors and the other matrix is used to read the out of core vectors.

Storage scheme: At any time during the unified GCR process, if the in core matrix has utilized all the available space, the algorithm writes the data in binary formats into a file and all the in-core vectors are destroyed. The in-core matrix is again free to store another user specified number of vectors. This process repeats itself until the convergence is achieved.

Reading scheme: When the out of core vectors are required, the out-of-core matrix reads the appropriate file and the vectors are made available for computation. After the vectors in the file are used, the out of core matrix is destroyed.

The developed out-of-core strategies will be validated by solving 2-D symmetrical (real numbers) frame structure problem (with 675 degrees of freedom and 5 RHS), and 2-D unsymmetrical (complex numbers) Aero-Acoustic problem (with 576 degrees of freedom and 7 RHS vectors). The key data and results are summarized in Table 5.

Table 5: In-core and Out-of-core GCR strategies.

Problem Description	2-D Symmetrical Frame	2-D Unsymmetrical Aero-Acoustic
Total # dof	675	576
Total # RHS vectors	5	7
Maximum # accumulated vectors allowed	(a) 675 (in-core strategies)	(a) 576 (in-core strategies)
	(b) 65 (out-of-core strategies)	(b) 80 (out-of-core strategies)
Number of converged iteration, for each RHS vector	(a) {483,13, 12,12,12}	(a) {353,59,25, 16,12,6,6}
	(b) {483,13, 12,12,12}	(b) {353,59,25,16, 12,6,6}
Elapsed time	(a) 301.41 seconds	(a) 181.46 seconds
	(b) 307.94 seconds	(b) 184.07 seconds

4. Summary

In this paper, the unsymmetrical GCR presented in Ref. [1] has been modified so that the following enhanced capabilities and conclusions can be stated:

- Both symmetrical and unsymmetrical matrix equations can be treated under a unified framework (including multiple RHS strategies).
- The suggested algorithm only involves with one "fairly large" incore memory array (instead of using two fairly large incore memory arrays, as presented in Ref. [1]).
- Both real and complex numbers can be treated.
- Through extensive test cases: linear/ nonlinear analysis, symmetrical/ unsymmetrical matrices, real/complex numbers, generic mathematical/structural/acoustic applications, the unified GCR seems to be robust and efficient, especially for handling multiple RHS load vectors.

References:

[1] F.J. Lingen, "A Generalized Conjugate Residual Method for the Solution of Non-Symmetric Systems of Equations with Multiple Right Hand Sides", IJNM in Engineering, Vol-44, pp. 641-656(1999).

[2] J.J. Dongarra, I.S. Duff, D.C. Sorensen, and H.A. Van der Vorst, "Numerical Linear Algebra for High-Performance Computers", SIAM (1998), ISBN # 0-89871- 428-1.

[3] C. Farhat, M. Lesoinne, P. LeTallec, K. Pierson, and D. Rixen, "FETI-DP: A Dual-Primal Unified FETI Method-Part I: A Faster Alternative to the 2 Level FETI Method," IJNME, Vol. 50, pp. 1523-1544 (2001).

[4] R. Kanapady, and K.K. Tamma, "A Scalablility and Space/Time Domain Decomposition for Structural Dynamics-Part I: Theoretical Developments and Parallel Formulations," Research Report UMSI 2002/188 (November 2002).

[5] D.T. Nguyen, S Tungkahotara, W.R. Watson, and S.D. Rajan "Parallel Finite Element Domain Decomposition for Structural/Acoustic Analysis," Journal of Computational and Applied Mechanics, Volume 4, no. 2 pp.189-201 (2003).

[6] Symbolic Math Toolbox: for use with MATLAB, user's guide, Version 3, The MathWorks Inc., 3 Apple Hill Drive, Natick, MA-01760-2098.