Parallel-Sparse Symmetrical/Unsymmetrical Finite Element Domain Decomposition Solver with Multi-Point Constraints for Structural/Acoustic Analysis

SIROJ TUNGKAHOTARA
INCA Engineers, Inc.
3850 N Causeway Blvd, Ste.210 New Orleans, LA 70002
USA
s.tungkahotara@incainc.com

WILLIE R. WATSON
NASA Langley Research Center
Computational Modeling & Simulation Branch; Mail Stop 128
Hampton, VA 23681
USA
willie.r.watson@nasa.gov

DUC T. NGUYEN
Old Dominion University
Department of Civil and Environment Engineering; 135 KAUF
Norfolk, VA 23529
USA
dnguyen@odu.edu

SUBRAMANIAM D. RAJAN
Arizona State University
Department of Civil, Environmental and Sustainable Engineering
Tempe, AZ 85287
USA
s.rajan@asu.edu

Abstract: - Details of parallel-sparse Domain Decomposition (DD) with multi-point constraints (MPC) formulation are explained. Major computational components of the DD formulation are identified. Critical roles of parallel (direct) sparse and iterative solvers with MPC are discussed within the framework of DD formulation. Both symmetrical and unsymmetrical system of simultaneous linear equations (SLE) can be handled by the developed DD formulation. For symmetrical SLE, option for imposing MPC equations is also provided.

Large-scale (up to 25 million unknowns involving complex numbers) structural and acoustic Finite Element (FE) analysis are used to evaluate the parallel computational performance of the proposed DD implementation using different parallel computer platforms. Numerical examples show that the authors' MPI/FORTRAN code is significantly faster than the commercial parallel sparse solver. Furthermore, the developed software can also conveniently and efficiently solve large SLE with MPCs, a feature not available in almost all commercial parallel sparse solvers.

Key-Words: - Domain Decomposition Solver, Multi-Point Constraints, Parallel Computation, Symmetrical/Unsymmetrical Simultaneous Linear Equation, Finite Element Analysis, Acoustic/Structural Engineering Applications, Iterative Algorithms, Sparse Assembly, Sparse Factorization.

1 Finite Element Analysis With Domain Decomposition (DD) Formulations

The finite element equilibrium equation (state equation) in terms of displacements, is given in [1]-[6]

$$\mathbf{K}\,\mathbf{z} = \mathbf{s} \tag{1}$$

where

$\mathbf{s}$ = vector of effective nodal loads on the structure

$\mathbf{z}$ = state variable vector of (e.g. nodal displacements)

$\mathbf{K}$ = global stiffness matrix, with dimension NxN

Using the DD concept, Eq. (1) can be re-written (in the partitioned form) as

$$\begin{bmatrix} \mathbf{K}_{BB} & \mathbf{K}_{BI} \\ \mathbf{K}_{IB} & \mathbf{K}_{II} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{z}_B \\ \mathbf{z}_I \end{bmatrix} = \begin{bmatrix} \mathbf{s}_B \\ \mathbf{s}_I \end{bmatrix} \tag{2}$$

where the subscripts B and I represent the boundary and interior terms, respectively.

The interior displacements $\mathbf{z}_\mathrm{I}$ are first eliminated from (2) and the following reduced equation is obtained ([1], [7]-[11])

$$\mathbf{K}_\mathrm{B}\mathbf{z}_\mathrm{B} = \mathbf{F}_\mathrm{B} \tag{3}$$

where

$$\mathbf{K}_\mathrm{B} = \mathbf{K}_\mathrm{BB} + \mathbf{K}_\mathrm{BI} \cdot \mathbf{Q} \tag{4}$$

$$\mathbf{F}_\mathrm{B} = \mathbf{s}_\mathrm{B} + \mathbf{Q}^T\mathbf{s}_I \tag{5}$$

$$\mathbf{Q} = -\left[\mathbf{K}_{II}\right]^{-1}\mathbf{K}_{IB} \tag{6}$$

Here $\mathbf{K}_\mathrm{B}$ is a boundary stiffness matrix for the entire structure and $\mathbf{F}_\mathrm{B} \in R^n$ is the vector of effective boundary forces. Efficient parallel (or serial) sparse numerical procedures discussed in [1], [12]-[20] can be used to decompose $\mathbf{K}_\mathrm{II}$ and to solve for $\mathbf{Q}_{m\times n}$ in (6).

The boundary stiffness $\mathbf{K}_\mathrm{B}$ and the effective boundary force vector $\mathbf{F}_\mathrm{B}$ are synthesized by considering contributions from all subdomains. For this purpose, the equilibrium equation for a sub-domain, which is considered as an isolated free-body, is also expressed in the partitioned form as

$$\begin{bmatrix} \mathbf{K}^{(r)}_{BB} & \mathbf{K}^{(r)}_{BI} \\ \mathbf{K}^{(r)}_{IB} & \mathbf{K}^{(r)}_{II} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{z}^{(r)}_B \\ \mathbf{z}^{(r)}_I \end{bmatrix} = \begin{bmatrix} \mathbf{s}^{(r)}_B \\ \mathbf{s}^{(r)}_I \end{bmatrix} \tag{7}$$

where the superscript r refers to the r^{th} sub-domain. Let n_r and m_r represent the number of boundary and interior degree-of-freedom (dof) of the r^{th} sub-domain, respectively. It may be noted that

$$m = \sum_{r=1}^{L} m_r$$

where L is the total number of subdomains. From the second equation in (7), one has

$$\mathbf{z}^{(r)}_\mathrm{I} = \left[\mathbf{K}^{(r)}_\mathrm{II}\right]^{-1}\left[\mathbf{s}^{(r)}_\mathrm{I} - \mathbf{K}^{(r)}_\mathrm{IB}\mathbf{z}^{(r)}_\mathrm{B}\right] \tag{8}$$

Substituting (8) into the first equation in (7), one obtains

$$\mathbf{K}^{(r)}_B\mathbf{z}^{(r)}_B = \mathbf{F}^{(r)}_B \tag{9}$$

where

$$\mathbf{K}^{(r)}_B = \mathbf{K}^{(r)}_{BB} + \mathbf{K}^{(r)}_{BI}\mathbf{Q}^{(r)} \tag{10}$$

$$\mathbf{F}^{(r)}_B = \mathbf{s}^{(r)}_B + \left[\mathbf{Q}^{(r)}\right]^T\mathbf{s}^{(r)}_I \tag{11}$$

$$\mathbf{Q}^{(r)} = -\left[\mathbf{K}^{(r)}_\mathrm{II}\right]^{-1}\mathbf{K}^{(r)}_\mathrm{IB} \tag{12}$$

The boundary stiffness matrix $\mathbf{K}^{(r)}_B$ and the effective boundary force vector $\mathbf{F}^{(r)}_B$ for each sub-domain are computed from (10) and (11), respectively. Finally, $\mathbf{K}_B$ and $\mathbf{F}_B$ are assembled according to the equations

$$\mathbf{K}_B = \sum_{r=1}^{L}\left[\beta^{(r)}\right]^T\mathbf{K}^{(r)}_\mathrm{B}\left[\beta^{(r)}\right] \tag{13}$$

$$\mathbf{F}_B = \mathbf{s}_\mathrm{B} + \sum_{r=1}^{L}\left[\beta^{(r)}\right]^T\left[\mathbf{Q}^{(r)}\right]^T\mathbf{s}^{(r)}_\mathrm{I} \tag{14}$$

where $\beta^{(r)}_{n_r\times n}$ is a boolean transformation matrix.

Using the reduced equilibrium equation (3), the boundary displacements $\mathbf{z}_\mathrm{B}$ can be computed by a parallel-dense equation solver [21]-[24], which also fully exploits the cache available in most modern computer platforms. The parallel dense equation solver (for the solution of $\mathbf{z}_B$ in (3)) requires explicit computation of the dense matrix $\mathbf{K}_B$ in (13), which also requires the computation of $\sum \mathbf{K}_B^{(r)}$ (shown in (10)), and $\mathbf{Q}^{(r)}$. From (12), since $\mathbf{K}_{IB}^{(r)}$ is a matrix with number of columns as n_r. Therefore, the triple product of $\mathbf{K}_{BI}^{(r)} \cdot \left[\mathbf{K}_{II}^{(r)}\right]^{-1} \cdot \mathbf{K}_{IB}^{(r)}$ can be very expensive. For this reason, an iterative solver (such as preconditioned conjugate gradient algorithm) is recommended to use for solving $\mathbf{z}_\mathrm{B}$ in (3). Interior displacements are then simultaneously computed for each sub-domain, using (8). Lastly, member end forces for the rth sub-domain are computed from

$$\mathbf{P}^{(r)} = \mathbf{K}^{(r)}\mathbf{z}^{(r)} \tag{15}$$

where $\mathbf{P}^{(r)}$ is the vector of member forces, $\mathbf{K}^{(r)}$ is the stiffness matrix and $\mathbf{z}^{(r)}$ is the vector of nodal displacements for the rth sub-domain. Multiple loading conditions for the structure are routinely treated by taking $\mathbf{s}$ and $\mathbf{z}$ in (1) as matrices whose jth columns represent quantities associated with the jth loading condition.

Algorithms and software given in [3] and [17] can be used to automatically break the original, large-scale finite element domain into several smaller sub-domains.

Equations (8) and (12) requires factorization of a sparse, symmetrical matrix [$\mathbf{K}_{\mathrm{II}}^{(r)}$] for the rth sub-domain. Thus, algorithms and software for sparse symbolic, numerical factorization, (with unrolling techniques) and forward-backward solution given in [1] and [12]-[20] can be utilized. In this work, however, solution strategies presented in [1] are incorporated.

Equation (3) requires factorization of a dense, symmetrical matrix $\mathbf{K}_B$. Thus, efficient parallel dense solvers given in references [1] and [21]-[24] can be utilized. In this work, however, preconditioned conjugate gradient (or PCG) as explained in [1], [7] is used.

The remainder of this paper deals with issues related to an efficient implementation for handling MPCs within the general frame work of parallel DD formulation, efficient sparse assembly procedures, and generating matrix $\mathbf{K}_B$ for obtaining either symmetrical or unsymmetrical system matrices.

2 Multi-Point Constraints (MPC) in DD formulation

To explain the multi-point constraints capability within the framework of domain decomposition formulation, consider a planar truss structure as shown in Fig. 1. The truss is modeled with 4-nodes, and 5-elements. Node 2 is at an inclined roller support or a skew support.

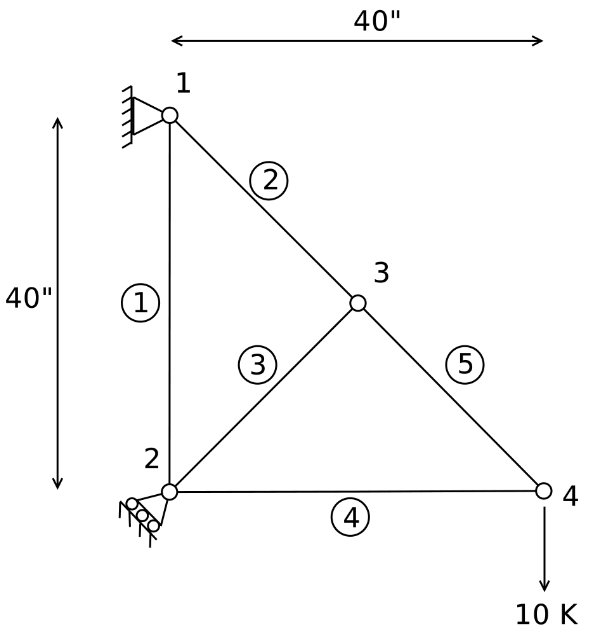


Fig.1: 4-node, 5-element truss example with an inclined roller support at joint 2

The MPC equation at the roller support joint 2 can be expressed as:

$$c_3 z_3 + c_4 z_4 = D \tag{16}$$

where z_3 and z_4 represent the horizontal and vertical displacements at node 2 in the global x and y directions, respectively and c_3, c_4 and D are known constants.

The single MPC equation (16) can be generalized to the following multiple MPC equations:

$$\begin{gathered}
c_{1,1}z_1 + c_{1,2}z_2 + \ldots + c_{1,i}z_i + c_{1,j}z_j + \ldots + c_{1,n}z_n = D_1 \\
c_{2,1}z_1 + c_{2,2}z_2 + \ldots + c_{2,i}z_i + c_{2,j}z_j + \ldots + c_{2,n}z_n = D_2 \\
\vdots \\
c_{i,1}z_1 + c_{i,2}z_2 + \ldots + c_{i,i}z_i + c_{i,j}z_j + \ldots + c_{i,n}z_n = D_i \\
c_{j,1}z_1 + c_{j,2}z_2 + \ldots + c_{j,i}z_i + c_{j,j}z_j + \ldots + c_{j,n}z_n = D_j \\
\vdots \\
c_{n,1}z_1 + c_{n,2}z_2 + \ldots + c_{n,i}z_i + c_{n,j}z_j + \ldots + c_{n,n}z_n = D_n
\end{gathered} \tag{17}$$

where $c_{i,j}$ and D_i are known constants. The total potential energy of the system described by (1) with MPC in (16) can be expressed as

$$\Pi(\mathbf{z}) = \frac{1}{2}\mathbf{z}^T\mathbf{K}\mathbf{z} - \mathbf{z}^T\mathbf{s} + \frac{1}{2}P\left(c_3 z_3 + c_4 z_4 - D\right)^2 \tag{18}$$

where P is a large penalty constant [25]. Experience has shown that using $P=10^4 \cdot max\left|K_{pq}\right|$ works reasonably well.

The terms appearing inside the parenthesis in the third term in (18) need be squared to guarantee a positive value (for a proper penalty term). The factor $1/2$ is used for convenience; the $1/2$ term disappears when the partial derivative of Π is computed.

From equation (18), it is seen that the total potential energy is minimum when $\frac{\partial \Pi}{\partial \mathbf{z}} = 0$. The derivative yields the usual total stiffness matrix and right-hand-side vector except the rows and columns associated with z_3 and z_4, in this case. The modified terms of rows and columns 3 and 4 are:

$$\begin{bmatrix} \ddots & \vdots & \vdots & \\ \cdots & k_{33}+Pc_3^2 & k_{34}+Pc_3c_4 & \cdots \\ \cdots & k_{43}+Pc_3c_4 & k_{44}+Pc_4^2 & \cdots \\ & \vdots & \vdots & \ddots \end{bmatrix} \tag{19}$$

and

$$\begin{bmatrix} \vdots \\ S_3+PDc_3 \\ S_4+PDc_4 \\ \vdots \end{bmatrix} \tag{20}$$

The additional terms could be considered as a fictitious, or artificial finite element stiffness matrix associated with each MPC. In other words, the MPC equations are treated in this work as additional artificial finite elements. The number of element nodes is the number of degrees of freedom in the MPC and each node has one degree of freedom. Since all MPC equations are treated as artificial (or fictitious) finite elements, they have to be included in the phase to find boundary degrees of freedom in order to avoid the coupling of interior degrees of freedom between two sub-domains.

As a quick example, suppose the following 2 MPC equations need to be implemented.

$$2z_3 - 8z_{17} + 4z_{25} = -6$$
$$-4z_8 + 12z_{23} = 5$$

Thus, the first artificial MPC finite element is created as

$$\left[\mathbf{k}_{MPC}^{(e=1)}\right] = \begin{bmatrix} P(2)^2 & P(2)(-8) & P(2)(4) \\ P(-8)(2) & P(-8)^2 & P(-8)(4) \\ P(4)(2) & P(4)(-8) & P(4)^2 \end{bmatrix}$$

and

$$\left\{\mathbf{s}_{MPC}^{(e=1)}\right\} = \begin{Bmatrix} P(-6)(2) \\ P(-6)(-8) \\ P(-6)(4) \end{Bmatrix}$$

and the element is associated with degrees of freedom 3, 17 and 25, respectively. Similarly, the second MPC element is created as

$$\left[\mathbf{k}_{MPC}^{(e=2)}\right] = \begin{bmatrix} P(-4)^2 & P(-4)(12) \\ P(12)(-4) & P(12)^2 \end{bmatrix}$$

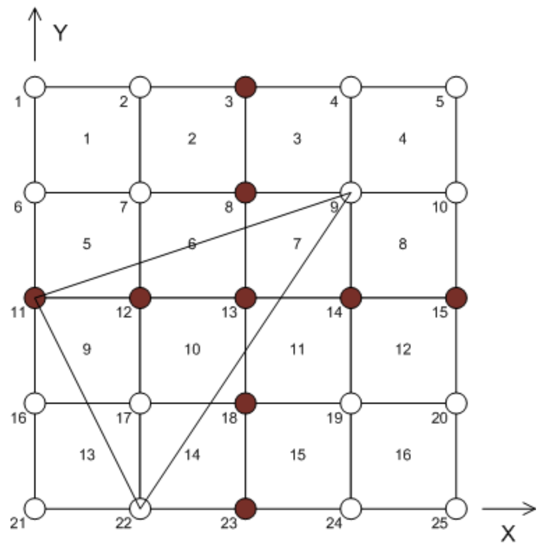
and

$$\left\{\mathbf{s}_{MPC}^{(e=2)}\right\} = \begin{Bmatrix} P(5)(-4) \\ P(5)(12) \end{Bmatrix}$$

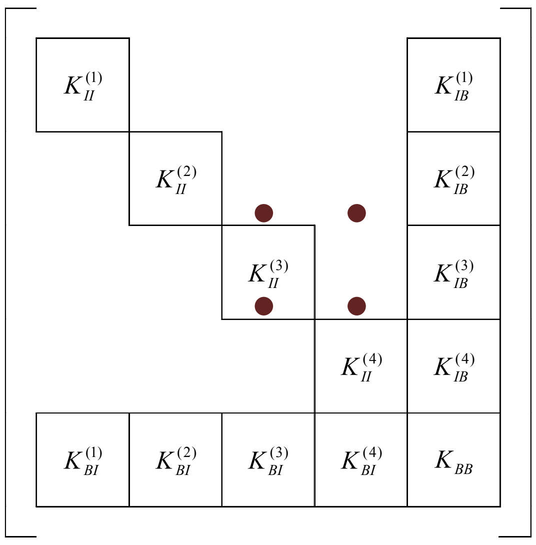
and the element is associated with degrees of freedom 8 and 23, respectively.

The DD formulation (for 2-dimensional FE model) with MPC equation(s) can be further explained by referring to Fig.2, and Fig.3. In Fig.2(a), the 2-D FE mesh is partitioned (or divided) into 4 sub-domains I, II, III, and IV. Rectangular finite elements #1, 2, 5, 6 belong to sub-domain I (upper left corner), while rectangular finite elements # 3, 4, 7, 8 belong to sub-domain II (upper right corner). Similarly, rectangular elements #9, 10, 13, 14 belong to sub-domain III (lower left corner), and rectangular finite elements #11, 12, 15, 16 belong to sub-domain IV (lower right corner), respectively. To simplify the explanation, it is assumed that there is only one (1) degree-of-freedom (or dof) at each node. Also, we assume only 1 MPC equation related to degrees of freedom 9, 11, and 22 as shown in Fig.2(a). Since the entire FE mesh (or entire system of SLE) is partitioned into 4 sub-domains, the boundary nodes can be identified as nodes 11-15, 3, 8, 13, 18, and 23 (see Fig.2(a)), while the remaining nodes are considered as interior nodes. The single MPC equation (related to degrees of freedom 9, 11, 22) will be treated like a 3-node (triangular) finite element in our DD formulation. This fictitious MPC finite element, however, does create some undesirable features. Node 9 (belongs to sub-domain II) and 22 (belongs to sub-domain III) are both interior nodes. As a consequence, there is an undesirable coupling effect between sub-matrices $\mathbf{K}_{II}^{(2)}$ and $\mathbf{K}_{II}^{(3)}$, as indicated in Fig.2(b). Because of this coupling effect, inverting (or factorizing) $\mathbf{K}_{II}^{(r)}$ for each r^th^ sub-domain cannot be concurrently done by independent/parallel processors.

Using our DD formulation, we also need to identify which sub-domain should be the owner of these fictitious n-noded MPC finite elements. In this study we have used the sub-domain that has the most interior nodes of the MPC finite element as the owner. This strategy helps to reduce the total boundary degrees of freedom for the entire domain. Since sub-domain II is the owner of one interior node (9), and sub-domain III is the owner of one interior node (22), there is a "tie" in this particular example. Hence, we arbitrarily assign this fictitious MPC finite element to sub-domain III. Thus the interior node 9 of the MPC element (originally belonging to sub-domain II) is now considered as a boundary node, and now belongs to both sub-domains II and III (see Fig.3(a) and 3(b)). As the consequence of the abovementioned elegant strategy, the coupling effect between $\mathbf{K}_{II}^{(2)}$ and $\mathbf{K}_{II}^{(3)}$ of sub-domains II and III now disappears (see Fig.3b)

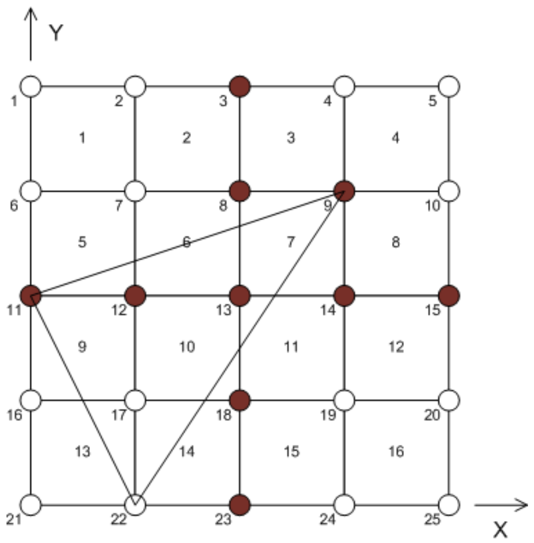


(a) FE Mesh with four sub-domains and one MPC element

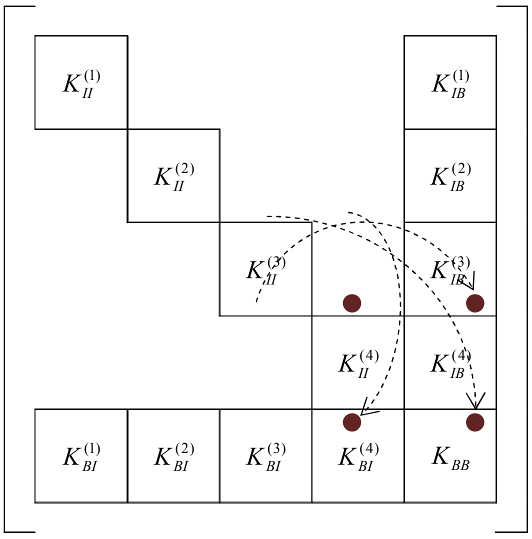


(b) Partitioned FE matrices from the 4 sub-domains and the 3-noded MPC element

Fig.2 Two-Dimensional Finite Element (FE) Domain Decomposition (DD) with 1 MPC equation



(a) FE Mesh with four sub-domains and one MPC element



(b) FE Mesh with four sub-domains and one MPC element

Fig.3 Two-Dimensional Finite Element (FE) Domain Decomposition (DD) with 1 MPC equation

3 Step-by-Step Numerical Procedures for Parallel/Sparse DD with MPC Equation Solver [1],[4],[14]

There exists vast amount of research literature on DD (iterative) algorithms for solving large sparse system of SLE [1], [7-11]. Most research articles, however, have been focused on solving system of symmetrical linear equations that are not burdened with MPCs.. We summarize the following step-by-step numerical procedure for parallel-sparse DD with MPC equation solver [1].

Step 1: The familiar, classical finite element input data (such as nodal coordinates, element connectivity, number of dof per node, materials properties, system right-hand-side load (or source) vector, Dirichlet (or geometrical) boundary conditions, etc.) for the entire domain is assumed to be known. Multi-Point Constraint (MPC) equations, if they exist, are also assumed to be known.

Step 2: Assuming the number of processors (NP) available is, ParMETIS domain partitioning algorithms [17], [26] is used to determine which joints (or nodes) belong to a particular rth processor (or rth sub-domain). In this step, only element connectivity information (for the entire domain [26] including MPC finite elements), and the total unknowns (or, degrees of freedom) N are used as input parameters.

Step 3: Using ParMETIS output, an interface subroutine (or module) [1], [26] can be written to identify the boundary, and interior nodes, and the corresponding elements for each rth processor (or rth sub-domain).

Step 4: Using METIS reordering algorithms [17] obtain an integer, mapping array that relates the old degree of freedom number to the new degree of freedom number. This step is very helpful for minimizing the number of fill-in terms during the symbolical and numerical factorization phases for factorizing matrices $\mathbf{K}_{ii}^{(r)}$.

Step 5: In parallel computation, every rth processor will generate finite element stiffness matrices (including the MPC finite elements [26], if they exist), and assemble the matrices $\mathbf{K}_{BB}^{(r)}$, $\mathbf{K}_{BI}^{(r)}$, $\mathbf{K}_{IB}^{(r)}$ and $\mathbf{K}_{II}^{(r)}$, as shown in (7).

It should be noted that, for symmetrical system of SLE, $\mathbf{K}_{IB}^{(r)} = \left(\mathbf{K}_{BI}^{(r)}\right)^T$. For unsymmetrical system of SLE, $\mathbf{K}_{IB}^{(r)} \neq \left(\mathbf{K}_{BI}^{(r)}\right)^T$.

Step 6: In parallel computation, each rth processor will perform sparse symbolical factorization of matrices $\mathbf{K}_{II}^{(r)}$, as shown in (8).

Step 7: In parallel computation, super-nodes (or super degrees of freedom) corresponding to each rth sub-domain are identified (for efficient unrolling techniques employed in Step 8) [1], [12], [20], [26].

Step 8: In parallel computation, each rth processor performs sparse numerical factorization of matrices $\mathbf{K}_{ii}^{(r)}$, as shown in (8).

In actual computer implementation, $\mathbf{K}_{ii}^{(r)}$ is not inverted (as explicitly shown in (8)). Instead, efficient symbolical and numerical factorization (with unrolling strategies) is implemented. Thus, direct sparse methods (such as Cholesky, or LDL^T algorithms are used for symmetrical SLE, or LU algorithm is used for unsymmetrical SLE.

Step 9: Use iterative algorithms, such as Preconditioned Conjugate Gradient (for symmetrical SLE), or m-GMRES [26] (for unsymmetrical SLE) to solve for the unknown boundary vector $\mathbf{z}_B$ as shown in (3).

Since an iterative method is used to solve (8), the system matrix $\mathbf{K}_B$ in (13), and the sub-domain's matrices $\mathbf{K}_B^{(r)}$ in (10) are never explicitly computed. Instead, only sub-domains' matrices $\mathbf{K}_{BB}^{(r)}$, $\mathbf{K}_{BI}^{(r)}$, $\mathbf{K}_{IB}^{(r)}$ and $\mathbf{K}_{II}^{(r)}$ are used. This step has been implemented in parallel computing environment with some inter-processor communication required [1].

Step 10: In parallel computing environment, each sub-domain's unknown interior degree of freedom can be solved by the familiar forward and backward solution phases (in conjunction with the Cholesky, or LDL^T algorithm, or LU algorithm), as shown in (8).

The sub-domain's $\mathbf{z}_{\mathrm{B}}^{(r)}$, shown in (8) is merely a subset of the vector $\mathbf{z}_{\mathrm{B}}$ that has already been solved in Step 9.

4 Numerical (Acoustic/Structural) Applications

Based on the implementation of the developed parallel DD formulation, several large-scale (acoustic and structural) engineering applications are solved. These problems have different features (such as real and complex numbers, symmetrical and unsymmetrical matrices, without and with MPC equations, etc.) and are considered in this section for evaluating the numerical performance of the developed algorithms and software. The software system is called DIPSS (domain decomposition formulation with mixed Direct-Iterative Parallel Sparse Solver). Since DIPSS has been coded with the standard MPI/FORTRAN language, it can be ported to different computer platforms without any change to the source code.

(A) Example 1 – Three Dimensional Acoustic Finite Element Model (Symmetrical Case)

The 3-D finite element acoustic model considered in this example involves 751,513 symmetrical equations involving complex numbers. Due to the 3-D nature of this example, there exist a very large number of fill-in (non-zero) terms during the factorization phase. Incidentally, for these reasons, this is the largest problem size that can be solved by NASA Langley Research Center's SGI parallel computer using the best commercial SGI sparse solver (subroutine ZPSLDLT). Using 8 SGI processors, subroutine ZPSLDLT took 6.5 hours to obtain the solution. Using the domain decomposition formulation with mixed Direct-Iterative Parallel Sparse Solver (DIPSS) developed by the authors, it took only 2.44 hours to obtain the same solution once again using 8 SGI processors.

DIPSS code was used to solve even larger 3-D acoustic finite element model involving 1,004,400 degree-of-freedom (a problem that cannot be solved using SGI's code). The problem was solved using SUN 10000 Processor cluster at Old Dominion University (64 nodes with 64 GB of memory). Timing information is detailed in Table 1.

Table 1: 3-D Hard Wall Duct Acoustic Finite Element Analysis with 1,004,400 degrees of freedom (complex numbers)

# Processors	**1**	**2**	**4**	**8**
Sparse Assembly Time (seconds)	19.38	10.00	5.08	2.49
Sparse Factorization (seconds)	131,229	58,976	26,174	10,273
Total time (entire FEA)	131,846	61,744	27,897	11,751
Total Speed-Up Factor	1.00	2.14	4.73	11.22

Cont'd

# Processors	**16**	**32**	**64**
Sparse Assembly Time (seconds)	1.26	0.70	0.27
Sparse Factorization (seconds)	3,260	909	56
Total time (entire FEA)	3,817	1,967	1,534
Total Speed-Up Factor	34.54	67.03	85.95

It should be noted that superlinear speedup is obtained for all runs. For example, with 64 processors the speedup is nearly 86. This can be explained as follows.

(a) The large finite element model has been divided into 64 sub-domains. Since each processor is assigned a smaller sub-domain, the number of operations (proportional to n^3 for dense matrix, or $n \cdot BW^2$ for banded, sparse matrix, BW represents the half band width of the coefficient stiffness matrix) performed by each processor is greatly reduced.

(b) When the entire finite element model (with 1,004,400 degrees of freedom) is analyzed by conventional formulation (using only direct sparse solver), due to large problem size, more computer paging is required as compared to the DD formulation.

(B) Example 2 - Three Dimensional Structural Bracket Finite Element Model (symmetrical case, with 194,925 degrees of freedom)

The developed MPI-DIPSS code has also been applied to solve a 3-D structural bracket on a cluster of Intel PCs (Pentium 4-1.75 GHz with 512 MB RAM) running Window XP OS. Timing information is shown in Table 2. Once again, superlinear speedup is obtained.

Table 2: 3-D Structural Bracket Model with 194,925 degrees of freedom (real numbers)

# Processor (Intel PC @ ASU)	1	2	3	4	5	6
Total Wall Clock Time (seconds)	2,670	700	435	405	306	258
Total Speed-Up Factor (seconds)	1.00	3.81	6.14	6.59	8.73	10.35

(C) Example 3 – Three Dimensional Acoustic Finite Element Model (Unsymmetrical Case)

In this example, an unsymmetrical finite element acoustic model with 6 million (complex numbers) degrees of freedom is solved. Due to the size of this problem, and the incore memory available on the ODU SUN 10000 cluster, at least 28 processors need to be used. The numerical performance of the developed parallel DD solver is summarized in the Table 3.

Table 3: 3-D Unsymmetrical Acoustic model with 6 million degrees of freedom

No. of CPUs	28	56
Time (seconds)	1965	1154
Relative Speedup	1.0	1.7

(D) Example 4 – Three Dimensional Acoustic Finite Element Model with 25 Million Degrees of Freedom

In this example, the generic aero-engine duct is modeled as a rectangular duct by cutting it along the axis and unwrapping it into a rectangular geometry. When unwrapped, the nacelle engine duct has a 317.5 cm x 63.5 cm rectangular cross-section and is 219.5 cm in length. Thus, the volume of our generic aero-engine duct is slightly more than 2,075 times that of the Flow Impedance Test Facility investigated in the previous example, and requires many more grid points for accurate resolution of the acoustic field. The highest frequency of interest (5.0 kHz) is roughly equivalent to four to six times the blade passage frequency (BPF) for a typical large commercial engine. Just to illustrate the capability of the hybrid solver we have used a (NNX, NNY, NNZ) = (100x100x2501) uniformly spaced grid (N=NNX*NNY* NNZ=25,010,000 unknown degrees of freedom involving complex numbers). This example was run on NASA Columbia supercluster (512 nodes of SGI Altix 3000 1.5 GHz with 1TB of RAM). Such large number of equations is far beyond what can be solved using direct sparse solvers such as the SGI solver.

The numerical performance of the developed parallel DD solver is summarized in Fig.4(a), and 4(b).

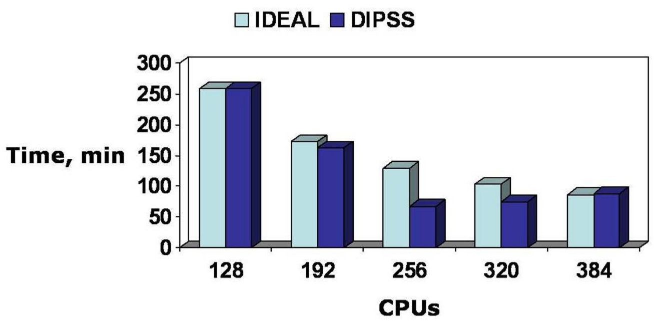


Fig.4(a). Example 4 - Wall Clock Time of DIPSS Solver (25 million points, 3.5 kHz, plane wave source, GIT)

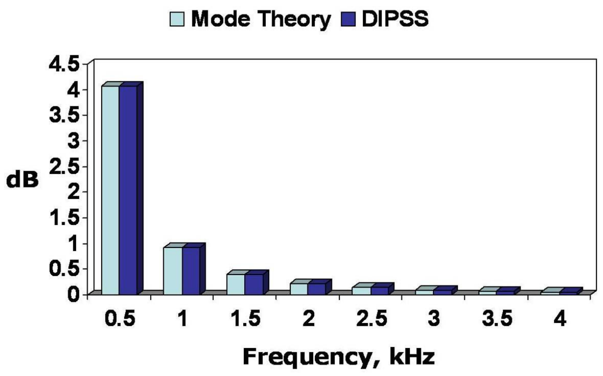


Fig.4(b). Example 4 - Linear Attenuation in Aeroengine Duct (25 million points, 192 CPUs, 1990 sec Wall Clock)

(E) Example 5 – Two Dimensional Acoustic Finite Element Model with 40 MPCs

In this example, a 3-D symmetrical acoustic FE model with 2.5 million degrees of freedom is considered. However, 40 MPC equations are included [26] in this example. More details of these 40 MPC equations can be obtained from [26]. The model was run on ODU Wilbur Cluster (64 nodes of AMD Opteron 1.8 GHz with 2 processors and 4GB of memory per node).Numerical performance of the developed parallel-sparse FE-DD solver is summarized in Table 4. There is a dramatic reduction in both computational time and computer memory requirements as the number of processors is increased.

Table 4: Timing information for 2.5 million degrees of freedom 3-D symmetrical acoustic with 40 MPC equations

2.5M (MPC)	20	30	40	50
Time(sec)	761	402	253	218
Ideal Speedup	1.00	1.50	2.00	2.50
Actual Speedup	1.00	1.89	3.01	3.49
Memory (MB)	1409	799	560	416

5 Conclusions

Details of the highly efficient parallel/sparse mixed direct-iterative Domain Decomposition (DD) formulation, including an elegant treatment of multi-point constraint (MPC) equations, have been presented in this work. The developed numerical procedures and the associated software has the capabilities to solve both symmetrical and unsymmetrical system of simultaneous linear equations (SLE), with (or without) imposing MPC equations. The entire formulation has been built around several key modules and concepts as follows.

(a) Efficient sparse assembly (to obtain $\mathbf{K}_{II}^{(r)}$, shown in (8)), based on [1].

(b) Efficient algorithms to break a large domain into smaller sub-domains, based on ParMETIS [17] and with special procedures to efficiently identify which boundary/interior nodes (and which finite elements) belong to sub-domain (or processor) [25].

(c) Efficient reordering algorithms (to minimize fill-in terms, during the symbolical and numerical sparse factorization phases for $\mathbf{K}_{II}^{(r)}$), based on METIS [17].

(d) Efficient sparse solver that takes full advantage of unrolling techniques and maximizes the usage of limited computer cache. (to solve for $\mathbf{z}_{\mathrm{I}}^{(r)}$ in Eq. (8)), based on [1].

(e) Efficient parallel pre-conditioned [1], [25] iterative solver (within the general frame work of DD formulation) which also exploits the developed cache based "matrix times vector" subroutine.

(f) Each MPC equation is conveniently treated as an artificial MPC finite element [25] with the associated known right-hand-side vector. Symmetry and positive definiteness are preserved without losing efficiency of the solution algorithm.

Medium (194,925 structural unknowns, real numbers, and 751,513 acoustic unknowns, complex numbers) to large-scale (ranging from 1 million to 25 million acoustic unknowns, complex numbers) examples considered in this study show that the developed MPI parallel DD code (with MPC equations imposed) is highly efficient (in terms of reduction of computational time, and computer memory requirements) in both sequential and parallel computer environments.

Acknowledgements

Prof. Duc T. Nguyen would like to acknowledge the research and financial support provided by NASA Langley Research Center.

References:

[1] D.T. Nguyen, *Finite Element Methods: Parallel-Sparse Statics and Eigen-Solutions*, Springer, 2006.

[2] K.J. Bathe, *Finite Element Procedures*, Prentice-Hall, Inc., 1996.

[3] M.A. Moayyad and D.T. Nguyen, An Algorithm For Domain Decomposition in Finite Element Analysis, *Journal of Computers and Structures*, Vol.39, No.1-4, 1991, pp. 277-290.

[4] W.R. Watson, Three-Dimensional Nacelle Aeroacoustic Code With Application to Impedance Eduction, *AIAA paper 2000-1956*, June 2000.

[5] D.T. Nguyen, *Parallel-Vector Equation Solver For Finite Element Engineering Applications*, Kluwer Academic/Plenum Publishers, 2002.

[6] J. Qin, C.E. Gray, Jr., C. Mei, and D.T. Nguyen, A Parallel-Vector Equation Solver for Unsymmetric Matrices on Supercomputers, Computing Systems in Engineering, *An International Journal*, Vol.2, No.2/3, 1991, pp. 197-290.

[7] D.T. Nguyen, S. Tungkahotara, W.R. Watson, S.D. Rajan, Parallel Finite Element Domain Decomposition For Structural/Acoustic Analysis, *Journal of Computational and Applied Mechanics*, Vol.4, No.2, 2003, pp. 1-12.

[8] C. Farhat, and F.X. Roux, Implicit Parallel Processing In Structural Mechanics, *Computational Mechanics Advances*, Vol.2, 1994, pp. 1-124.

[9] J. Mandel, Balancing Domain Decomposition, *Comm. Appl. Num. Meth. Engr.*, Vol.9, 1993, pp. 233-241.

[10] R. Glowinski, G.H. Golub, G.A. Meurant, and J. Periaux, Editors, *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM, 1988.

[11] T.F. Chan, T.P. Mathew, Domain Decomposition Algorithms, *Acta Numerica*, Vol.3, 1994, pp. 61-143.

[12] E. Ng, and B. Peyton, Block Sparse Choleski Algorithm on Advanced Uniprocessor Computer, *Society for Industrial and Applied Mathematics Journal of Scientific Computing*, Vol.14, 1993, pp. 1034-56

[13] I. Duff and J. Reid, *MA27-A set of Fortran Subroutines for Solving Sparse Symmetric Sets of Linear Equations*, AERE Technical Report, R-10533, Harwell, 1982.

[14] I. Duff and J. Reid, The Multifrontal Solution of Indefinite Sparse Symmetric Linear Systems, *Association for Computing Machinery Transactions Mathematical Software*, Vol.9, 1983, pp. 302-325.

[15] A. George, and J. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Inc., 1981, chap. 5 & chap. 10.

[16] S. Pissanetzky, *Sparse Matrix Technology*, Academic Press (AP), Inc1984.

[17] G. Karypis and V. Kumar, METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering, Version 2.0, *University of Minnesota*, 1995.

[18] M. Papadrakakis, S. Bitzarakis and A. Kotsopulos, Parallel Solution Techniques in Computational Structural Mechanics, B.H.V. Topping (Editor), *Parallel and Distributed Processing for Computational Mechanics: Systems and Tools*, Saxe-Coburg Publications, 1999. pp. 180-206.

[19] X.S. Li, *Sparse Gaussian Elimination on High Performance Computers*. Technical Report,

UCB/CSD-96-919, Computer Science Division, U.C. Berkeley, September 1996, Ph.D. dissertation.

[20] J.W. Demmel, J.R. Gillbert, and X.S. Li, *SuperLU and SuperLU_MT*, 1997.

[21] L.S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R.C. Whaley, *ScaLAPACK Users' Guide*, Society for Industrial and Applied Mathenatics, 2007.

[22] S. Tungkahotara, D.T. Nguyen, W.R. Watson and H.B. Runesha, Simple and Efficient Parallel Dense Equation Solvers, *Ninth International Conference on Numerical Methods and Computational Mechanics*, Univ. of Miskolc, July 15-19, 2002.

[23] E. Anderson, Z. Bai, C. Bischof, L.S. Blackford and James W. Demmel, Lapack Users' Guide (Software, Environments and Tools, 9), *Society for Industrial & Applied Mathematics* 3rd pkg edition, February 2000.

[24] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, James W. Demmel, Scalapack Users' Guide, *Society for Industrial & Applied Mathematics*, Bk&Cd r edition, July 1997.

[25] S.D. Rajan, *Introduction to Structural Analysis and Design*, John Wiley & Sons, Inc., 2001.

[26] S. Tungkahotara, *Parallel MPI/FORTRAN Finite Element Symmetrical/Unsymmetrical Domain Decomposition*, Old Dominion University, Civil & Env. Engineering Department, May 2008, Ph.D. dissertation.