# A Dual Super-Element Domain Decomposition Approach for Parallel Nonlinear Finite Element Analysis

G.A. Jokhio[1] and B.A. Izzuddin[2]

## Abstract

This paper presents a new domain decomposition method for nonlinear finite element analysis introducing the concept of dual partition super-elements. The method extends ideas from the displacement frame method and is ideally suited for parallel nonlinear static/dynamic analysis of structural systems. In the new method, domain decomposition is realised by replacing one or more subdomains in a 'parent system' each with a placeholder super-element, where the subdomains are processed separately as 'child partitions' each wrapped by a dual super-element along the partition boundary. The analysis of the overall system including the satisfaction of equilibrium and compatibility at all partition boundaries is realised through direct communication between all pairs of placeholder and dual super-elements. The proposed method has particular advantages for matrix solution methods based on the frontal scheme, and can be readily implemented for existing finite element analysis programs to achieve parallelisation on distributed memory systems with minimal intervention, thus overcoming memory bottlenecks typically faced in the analysis of large-scale problems. Several examples are presented in this paper, which demonstrate the computational benefits of the proposed parallel domain decomposition approach and its applicability to the nonlinear structural analysis of realistic structural systems.

## Keywords

Domain decomposition; Dual super-elements; Finite element analysis; Nonlinear analysis; Parallel processing; Distributed memory systems.

---

[1] Research Student, Department of Civil & Environmental Engineering, Imperial College London, SW7 2AZ, U.K.

[2] Professor of Computational Structural Mechanics, Department of Civil & Environmental Engineering, Imperial College London, SW7 2AZ, U.K., Corresponding author, email: b.izzuddin@imperial.ac.uk.

## 1 Introduction

It is well established that the computing time of nonlinear finite element analysis applied to large and complex structural systems can be significantly reduced by parallelisation. One way of achieving this is by dividing the problem into sub-problems and then processing these sub-problems simultaneously on parallel processors connected to a network. Within this context, parallelisation can either be based on shared memory systems or distributed memory computing systems. The use of shared memory systems, however, can lead to memory bottlenecks for problems of very large size, a shortcoming which is easily addressed with parallel domain decomposition on distributed memory systems.

Several domain decomposition techniques exist which can be divided into two main categories: mathematical and physical [1,2]. In mathematical domain decomposition, the various mathematical functions of the problem are divided into sub-problems. This kind of problem subdivision, however, requires the development of new solution methods and techniques if it is to be implemented for the purpose of nonlinear structural analysis. The most effective approach for problem decomposition which requires least interference with existing FEA codes is based on physical domain decomposition. The physical domain decomposition is often referred to as the data-decomposition, rather than problem decomposition, and involves the creation of subdomains or partitions with interfaces at the partition boundaries. In the context of structural analysis, it is then necessary to make sure that equilibrium and compatibility are not only satisfied within the partitions but also at the interfaces between partitions. Indeed, this latter issue tends to be the defining feature for the various domain decomposition techniques.

Different approaches used for solving the physical domain decomposition problems include the staggered approach and iterative coupling methods. The staggered approach is invariably applicable to dynamic analysis, where the partitioned subdomains utilise a predictor to predict the interface boundary conditions using values from the previous time-step [3,4]. Due to the fact that the displacements at the interface boundary of one subdomain are predicted without the communication of actual boundary interface displacements between subdomains, this approach does not necessarily satisfy compatibility. Another major disadvantage with the staggered approach is its conditional stability [5], which can require an excessively small time-step size, leading to considerable computational inefficiency.

To overcome the time-step size dependence of the staggered approach, corrective iterations have been introduced, resulting in the development of iterative coupling methods [4,6], which support parallel computations and can be applied to both static and dynamic problems. In these methods, the solution of the governing equations of the partitioned subdomains is carried out individually at each load or time step using predicted boundary conditions at the interface, which are successively subjected to iterative corrections until the satisfaction of equilibrium and compatibility. The convergence of such a procedure, however, is not guaranteed, particularly if the corrections to the interface boundary conditions are obtained without due consideration of the coupled system response. The solution dependency in this case shifts from time-step size to the type of iterative update employed for the displacement/force interface boundary conditions for each subdomain [4].

To improve the convergence of the iterative coupling methods, boundary conditions can be updated using an ‘interface relaxation approach’ [7,8], though the optimal relaxation parameter is problem dependent [4]. The convergence characteristics of iterative coupling methods can be significantly improved with the use of the condensed tangent stiffness matrix at the interface boundary, which can be approximated using such techniques as the reduced-order method [9]. Of course, even better convergence can be achieved with the use of the exact tangent stiffness matrix, and this will be sought in the present work.

Besides the iterative method used to ensure convergence to compatibility and equilibrium at the interface boundary, domain decomposition in finite element analysis also raises the question of linking subdomains with non-matching meshes at the interfaces. Kron [10] presented a method for linking subdomains for elasticity problems by using Lagrange multipliers, where discretisation of interface displacements in each of the linked subdomain and of the Lagrange multipliers, also referred to as interface tractions, yields a system of simultaneous equations. Based on this method, Farhat and Roux [11] presented a method of finite element tearing and interconnecting, more commonly known as the FETI method. Numerous researchers have since worked on the further development of the FETI method and its applications to different types of problem [12,13].

An alternative to domain decomposition methods based on Lagrange multipliers is an approach where subdomains are linked by a displacement field that is defined only on the interface. This approach is generally termed as the ‘frame method’, in which a displacement frame is made to surround the subdomain completely so that when all the internal variables are eliminated, the frame yields a stiffness matrix which can be used directly in coupling with any other element with similar displacement assumptions [14].

Based on a related concept, a new method is proposed for domain decomposition in nonlinear finite element analysis, which facilitates scalable parallel processing over distributed memory systems, thus overcoming memory bottlenecks for large scale problems. The proposed method introduces the concept of dual partition super-elements, where the term ‘dual’ takes here its literal meaning of ‘double/mirror image’ rather than the mathematical meaning of ‘duality’ as applied in such areas as linear mathematical programming. In this respect, a placeholder super-element represents a child partition in a parent subdomain, and this maps to a dual super-element that wraps the child partition at the interface boundary in a separate child process. The enforcement of compatibility and equilibrium at the interface boundary is thus achieved through communication between pairs of dual and placeholder super-elements, where multiple subdomains and scalable parallel processing are readily accommodated with a parent subdomain having multiple child partitions. It is worth noting that despite the conceptual similarity between the dual super-element and displacement frame methods [14], the discrete response of the dual super-element is readily recovered at the subdomain boundary following condensation operations, as opposed to the frame method in which the tractions are integrated over the boundary considering frame-specific shape functions.

An important benefit of the proposed method is that it allows the recovery of child partition forces and condensed tangent stiffness matrix at the interface boundary relatively easily, which can be achieved in a frontal solution method [15,16] by placing the child dual super-element at the end of the element ordering list. When all the other elements of the partition have been assembled and the associated interior freedoms are eliminated, the remaining equilibrium equations contain the forces and condensed tangent stiffness matrix for the dual super-element only, which can be communicated to the placeholder super-element that presents these as its forces and tangent stiffness in the parent partition. In this respect, the parent process treats the placeholder super-element similar to other finite elements, providing it with displacements that can be communicated to the child dual super-elements and receiving interface forces and the condensed tangent stiffness matrix in return. This results in a domain decomposition method which can be implemented with minimal intervention into existing nonlinear finite element analysis program, providing an overall convergence rate which is identical to the monolithic approach, and achieving very high speed-ups due to parallel processing.

It is worth noting here that MPI based parallel finite element approaches [17, 18] have also been developed for implicit nonlinear dynamic analysis utilising linear preconditioned conjugate gradient (PCG) solvers for the iterative analysis as opposed to the frontal solver, conventionally considered to be direct solution method [19]. The use of PCG solvers requires significant modification of existing finite element programs as most of these utilise direct solvers based on Gaussian elimination. Indeed, the typical requirement for a symmetric positive definite stiffness matrix with PCG solvers cannot be usually realised in nonlinear finite element analysis, which, in addition to the need for evaluating a conditioner matrix associated with a changing tangent stiffness matrix, discourages the use of PCG solvers in nonlinear finite element analysis. Furthermore, the domain decomposition in these approaches is carried out using graph partitioning techniques making it difficult to adapt domain decomposition to different mesh sizes or dimensions and to consider different time-integration schemes within the partitions. In the present approach, the solution of the nonlinear structural analysis problem can be developed using any solver based on Gaussian elimination, where specific consideration is given in this work to the frontal solver. Furthermore, the introduction in the present approach of the concept of parent-child partitions, with the parent partition acting also as a coordinator, creates the possibility of the use of mixed dimensional element coupling as well as mixed integration schemes across the partitions. Additionally, unlike the previous approaches [17, 81], neighbouring partitions in the present approach do not exchange boundary displacement/force entities directly; instead information flow is controlled by the parent coordinator, which is responsible for ensuring compatibility and equilibrium at the partition boundaries. A clear and natural extension of the proposed approach is hierarchic multi-level partitioning, which cannot be accommodated by previous approaches [17, 18], and which maps readily to hierarchic HPC architecture with ensuing reduction in inter-processor communication overheads.

It should be noted here that the present approach of domain decomposition is not limited to either 2D or 3D analysis. Rather, the approach is general and can be implemented for any existing finite element analysis program with different element libraries with only minor modifications. The approach can be also extended to include multi-physics modelling [20], which is becoming increasingly important role in various field of engineering.

Previous work by the authors [21] illustrated the application and overall benefits of the newly developed approach for parallel nonlinear finite element analysis. The current paper presents the complete method formulation, which may be directly used to upgrade virtually any monolithic finite element analysis program with parallel capabilities based on domain decomposition. This would not only enhance the speed of computations using such programs but would also overcome serious memory bottlenecks faced in the analysis of large scale problems. A further important aim of this paper is to demonstrate that the enhanced computing speed does not only arise in the proposed method from parallelisation of element computations but also from solving the simultaneous equations of the overall system. In this respect, the application of the proposed approach with a frontal solution procedure for individual child partitions presents a practical implementation and instance of a parallel multifrontal solution scheme [22] that is driven by physical partitioning. Besides these computational and practical benefits, the proposed partitioning approach also provides a natural framework for hierarchic partitioning, where a child partition can be decomposed into further partitions, and for mixed-dimensional coupling between heterogeneous subdomains [23].

The paper proceeds with presenting the components of the proposed approach, highlights the sources of computational efficiency, and provides several examples which demonstrate its excellent performance as a parallel domain decomposition method for nonlinear finite element analysis on distributed memory systems.

## 2 Partitioning with Dual Super-Elements

To facilitate the presentation of the developed partitioning approach, a structural domain (Figure 1) is considered subject to arbitrary restraint and loading conditions, and which is to be partitioned into three subdomains by introducing two partitions along the dotted lines. Without loss of generality, it is assumed here that each node has 2 degrees of freedom (DOF), hence this domain with 40 nodes has a total of 80 DOF when considering a monolithic treatment. In nonlinear finite element analysis, the tangent stiffness matrix relates the infinitesimal increments of resistance and displacements, thus for the overall structural domain under consideration:

$$\delta R_{\{80\}} = K_{[80\times 80]}\delta d_{\{80\}} \qquad \text{Eq. 1}$$

where $\delta d$ is infinitesimal increment of nodal displacements, $K$ is the tangent stiffness matrix, and $\delta R$ is the infinitesimal change in resistance. The subscripts in brackets indicate the respective size of arrays with curly brackets { } representing a column vector and square brackets [ ] representing a matrix. If $P_{\{80\}}$ is the vector of applied loads, and $R_{\{80\}}$ is the total resistance offered by the structure, then the out-of-balance $G_{\{80\}}$ is defined as:

$$G_{\{80\}} = R_{\{80\}} - P_{\{80\}} \qquad \text{Eq. 2}$$

The structure is said to be in equilibrium if the out-of-balance vanishes to within an acceptable tolerance:

$$G_{\{80\}} \cong O_{\{80\}} \qquad \text{Eq. 3}$$

where $O$ is a zero vector. Note that the treatment of restrained DOF at supports is easily accommodated by replacing the equilibrium conditions in $G$ with the essential nodal displacement conditions, though this is excluded from the following discussion without loss of generality to simplify the method presentation.

A non-zero $G$ for a given set of nodal displacements $d$ indicates lack of equilibrium and necessitates iterative correction of the displacements, which is obtained (for load control) as:

$$\delta d_{\{80\}} = K^{-1}_{[80\times 80]}\left(-G_{\{80\}}\right) \qquad \text{Eq. 4}$$

leading to:

$$d_{n\{80\}} = d_{\{80\}} + \delta d_{\{80\}} \qquad \text{Eq. 5}$$

where $d$ and $d_n$ are the previous and current iterative displacement vectors, respectively. The new resistance forces vector $R$ is evaluated again for the current displacements (i.e. $d = d_n$), and the iterative process is repeated until Eq. 3 is satisfied to within an acceptable tolerance.

It is noted here that Eq. 4 becomes different when the load at the end of the step is unknown, such as in displacement or arc length control as elaborated later in Section 2.4.

## 2.1 Partitioning Method – Solution at Parent Level

With the structural domain partitioning, let the three new subdomains be named as $\Omega_0$, $\Omega_1$ and $\Omega_2$, where the zero-indexed partition represents the parent structure, while the rest represent children partitions. The three partitions are illustrated in Figure 2.

After partitioning, the parent subdomain $\Omega_0$ has 37 nodes, of which 15 are connected to two partition placeholder super-elements. For the child partitions, partition $\Omega_1$ has 12 nodes with 10 nodes at the boundary connected to a dual super-element, while partition $\Omega_2$ has 9 nodes with 8 at the partitioned boundary connected to a dual super-element. Generally, the nodes in the parent subdomain that are connected to the partition super-elements may or may not be connected to conventional elements. In the example under consideration, one node is connected to partition super-elements only, and 14 nodes are connected to conventional finite elements in addition to partition super-elements. The parent subdomain is still complete with the response of the removed partitions being represented by the partition super-elements. The stiffness matrix for the parent subdomain can be assembled in the normal way by first accounting for the contribution from the conventional elements:

$$K_{[74,74]} = K_{\Omega_0[74,74]} \qquad \text{Eq. 6}$$

In the above equation the terms associated with node 31 are zero because there is no conventional element attached to this node. The contributions from the partition super-elements are then assemble into the stiffness matrix. The first partition super-element is connected to 10 nodes, which are numbered consecutively from 23 to 32 for presentational convenience, leading to the following additional assembled contribution:

$$K_{[45:64,45:64]} = K_{[45:64,45:64]} + K^{c}_{\Omega_1[20,20]} \qquad \text{Eq. 7}$$

where $K^{c}_{\Omega_1[20,20]}$ is the condensed stiffness matrix of child partition $\Omega_1$ after the internal nodes have been eliminated using the conditions of internal equilibrium. As discussed in the next section, the condensed stiffness matrix is recovered easily in a frontal solution method from the child analysis process as the ‘Grandpa’ [15] associated with the remaining active freedoms of the dual super-element on the partitioned boundary. Similarly the contribution from the second partition super-element, which is connected to 8 nodes numbered from 30 to 37, is assembled as:

$$K_{[59:74,59:74]} = K_{[59:74,59:74]} + K^{c}_{\Omega_2[16,16]} \qquad \text{Eq. 8}$$

It is evident that super elements are treated in an identical way to conventional finite elements with regard to assembly of element contributions, and therefore the order of element assembly in the parent subdomain may be varied from what is assumed in the above discussion without loss of generality. Indeed, in a frontal solution method, the order of assembly at the parent level, considering super-elements and conventional elements individually, would typically be determined as one that minimises the front width.

Considering the assembled tangent stiffness matrix, the change in resistance due to the iterative corrections of displacements, including the contribution from the partition super elements, may be approximated for the purpose of iteration as:

$$\delta R_{\{74\}} = K_{[74,74]} \delta d_{\{74\}} \qquad \text{Eq. 9}$$

where $\delta d$ and $\delta R$ refer here to finite iterative increments of nodal displacement and resistance at the parent level, respectively.

Similar to the tangent stiffness matrix, the resistance forces vector $R$ is assembled as contributions from conventional finite elements and super elements, where in the latter case the condensed resistance forces at the partitioned boundary $R^{c}_{\Omega_m}$ are considered. The out-of-balance at the parent level is then obtained as:

$$G_{\{74\}} = R_{\{74\}} - P_{\{74\}} \qquad \text{Eq. 10}$$

It is worth noting here that the load vector $P_{\{74\}}$ in the above equation does not contain the loads that are applied to the internal nodes of the partitions, as these are dealt with inside the child partitions. If there is any out-of-balance remaining, the correction to the displacements at the parent level is obtained as:

$$\delta d_{\{74\}} = K_{[74\times 74]}^{-1}\left(-G_{\{74\}}\right) \qquad \text{Eq. 11}$$

The corrections to the displacements for the nodes connected to the partition super-elements are communicated to the dual super-elements:

$$\delta d_{\Omega_1\{20\}}^{c} = \delta d_{\{45:64\}} \qquad \text{Eq. 12}$$

and:

$$\delta d_{\Omega_2\{16\}}^{c} = \delta d_{\{59:74\}} \qquad \text{Eq. 13}$$

so as to determine the displacement corrections for the internal nodes, as discussed in the next section. Once the partition displacements are updated with the iterative corrections, the condensed resistance forces and tangent stiffness from the dual super-elements are provided to the parent structure for the next iteration. The process is repeated until convergence to an acceptably small out-of-balance is achieved at both the parent and child levels.

## 2.2 Frontal Method – Solution at Child Level

The solution of the linearised system of equations at the parent level, as discussed in the previous section, may be obtained efficiently using a variety of techniques, such as skyline solution methods [24, 25] or the frontal method [15, 16, 26]. At the child level, however, there is a marginal benefit in using the frontal method, which is based on optimal element ordering for minimum front width. In this respect, the use of a wrapper element over the partitioned boundary, so-called dual super-element, and its placement as the last element in the frontal order of assembly allows the straightforward recovery of the condensed resistance forces and tangent stiffness matrix required at the parent level.

The solution technique in the frontal method, or in any other method based on Gaussian elimination, consists of two phases: forward elimination and backward substitution. In the forward elimination phase, the augmented matrix consisting of the coefficient matrix and the right hand side is converted into row echelon form by a succession of elementary row operations. The system of linearized equations for the example under consideration can be represented as:

$$\begin{bmatrix} K_{1,1} & K_{1,2} & \cdots & K_{1,n} \\ K_{2,1} & K_{2,2} & \cdots & K_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ K_{n,1} & K_{n,2} & \cdots & K_{n,n} \end{bmatrix} \begin{Bmatrix} \partial d_1 \\ \partial d_2 \\ \vdots \\ \partial d_n \end{Bmatrix} = - \begin{Bmatrix} G_1 \\ G_2 \\ \vdots \\ G_n \end{Bmatrix} \qquad \text{Eq. 14}$$

where *n* depends on the partition size.

The augmented matrix form for the above equation can be represented as:

$$\left[\begin{array}{cccc:c} K_{1,1} & K_{1,2} & \cdots & K_{1,n} & G_1 \\ K_{2,1} & K_{2,2} & \cdots & K_{2,n} & G_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ K_{n,1} & K_{n,2} & \cdots & K_{n,n} & G_n \end{array}\right] \begin{Bmatrix} \partial d_1 \\ \partial d_2 \\ \vdots \\ \partial d_n \end{Bmatrix} \qquad \text{Eq. 15}$$

where the displacement vector is tagged along in order to label the matrix rows.

When the contributions of all elements are available, such as at the parent level, a succession of elementary row operations, namely forward elimination, is used to convert this matrix into its upper triangular form:

$$\left[\begin{array}{cccc:c} K'_{1,1} & K'_{1,2} & \cdots & K'_{1,n} & G'_1 \\ 0 & K'_{2,2} & \cdots & K'_{2,n} & G'_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & K'_{n,n} & G'_n \end{array}\right] \qquad \text{Eq. 16}$$

which can immediately be followed by backward substitution to determine the iterative displacement corrections:

$$\delta d_n = \frac{-G'_n}{K'_{n,n}} \qquad \text{Eq. 17}$$

$$\delta d_i = \frac{1}{K'_{i,i}} \left( -G'_i - \sum_{j=i+1}^{n} K'_{i,j} \delta d_j \right) \qquad \text{Eq. 18}$$

For a child partition, however, the forward elimination process cannot include elimination of the freedoms at the partitioned boundary, since these are attached to the parent partition. In this respect, the use of a wrapper dual super element attached to the partitioned boundary nodes, and which is placed at the end of the frontal ordering list, allows the forward elimination process to be conveniently terminated at the point where only the partitioned boundary freedoms remain active. If *h* represents the number of freedoms on the partitioned boundary, and assuming without loss of generality that these are the last *h* of a total of *n* freedoms, the following augmented matrix is obtained at the point when the dual super element is being considered in the frontal solution:

$$\left[\begin{array}{ccccccc|c}
K'_{1,1} & K'_{1,2} & \cdots & \cdots & \cdots & \cdots & K'_{1,n} & G'_1 \\
0 & K'_{2,2} & \cdots & \cdots & \cdots & \cdots & K'_{2,n} & G'_2 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & K'_{n-h+1,n-h+1} & K'_{n-h+1,n-h+2} & \cdots & K'_{n-h+1,n} & G'_{n-h+1} \\
\vdots & \vdots & \cdots & K'_{n-h+2,n-h+1} & K'_{n-h+2,n-h+2} & \cdots & K'_{n-h+2,n} & G'_{n-h+2} \\
\vdots & \vdots & \cdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & K'_{n,n-h+1} & K'_{n,n-h+2} & \cdots & K'_{n,n} & G'_n
\end{array}\right] \qquad \text{Eq. 19}$$

The resulting 'Grandpa' associated with the active freedoms $K'_{[n-h+1:n,n-h+1:n]}$ is returned as the condensed tangent stiffness matrix $K^c_{\Omega_m}$ for child partition *m*, while the associated right hand side vector $G'_{\{n-h+1:n\}}$ is returned as the corresponding condensed resistance forces $R^c_{\Omega_m}$. In this respect, the condensed tangent stiffness and resistance forces for a child partition, which are obtained at the point of considering the dual super-element in the frontal solution, are communicated exclusively to the placeholder super-element to present these as its contributions at the parent level.

Once all child contributions are returned and assembled at the parent level, the iterative displacement corrections can be determined for the parent structure through a contiguous process of forward elimination followed by backward substitution. The iterative displacements at the partitioned boundary can then be provided to the child partitions via the dual super-elements, and only at this point can the iterative displacements inside the child partitions be determined via backward substitution. Clearly therefore, the forward elimination and backward substitution phases at the child level are interrupted by returning $K^c_{\Omega_m}$ along with $R^c_{\Omega_m}$ to the parent and then receiving $\delta d^c_{\Omega_m}$ from the parent following the solution at the parent level. Once $\delta d^c_{\Omega_m}$ is established, which corresponds at the child level to the iterative displacements of the active freedoms $\delta d_{\{n-h+1:n\}}$, backward substitution proceeds in accordance with Eq. 18 using the augmented matrix of Eq. 19 to establish the remaining iterative displacements starting from freedom $n-h$ to freedom 1.

It should be clear that the present partitioning method shares some concepts with the Schur Complement Method [27], where the assembled stiffness matrix $K$ following Eqs. (6-8) can be considered as the Schur Complement of the sub-matrix associated with freedoms of the parent sub-domain $\Omega_0$ in the stiffness matrix of the overall domain $\Omega$. However, unlike the Schur Complement Method, the assembled matrix $K$ not only relates to interface freedoms but also to interior freedoms of the parent sub-domain $\Omega_0$. In this respect, the adopted partitioning approach presents an effective generalisation of the Schur Complement Method that also facilitates hierarchic partitioning, in the sense that a child partition can be decomposed into further partitions leading to significant computational benefits on parallel systems.

## 2.3 Control and Convergence

The control of the incremental iterative solution procedure for tracing the equilibrium path is most effectively undertaken at the parent level. For proportional static loading, where all the loads are scaled by a single parameter $\lambda$, nonlinear analysis may be undertaken using load control, in which case $\lambda$ is prescribed for each increment at the parent level and propagated to child partitions. For non-proportional static loading, nonlinear analysis is most effectively undertaken over the time (or pseudo-time) domain, where the time $t$ is also prescribed for each increment at the parent level and propagated to child partitions. An almost identical solution procedure is adopted for dynamic loading using time-marching over the time domain. In all these cases, the loads are prescribed over the incremental step under consideration, and the expressions for the iterative displacement corrections are identical to those provided in the previous two sections for the parent structure and child partitions. For proportional static loading, however, displacement or arc-length control is often employed to trace the nonlinear response following buckling or softening, where the load factor $\lambda$ is unknown. This requires a modification of the expressions for iterative displacement corrections, as elaborated in the next section.

As mentioned before, overall convergence to equilibrium should be established with due consideration of the out-of-balance at the parent and child levels. Different measures of convergence may be used, such as the norm of the out-of-balance *G*, the norm of iterative displacements $\delta d$ or the scalar product of *G* with $\delta d$. Convergence would then be considered to have been achieved when one or more of such measures are below a predefined tolerance $\varepsilon$. In this respect, convergence can be checked separately at the parent and child levels, with overall convergence required at all levels, as expressed by:

$$f\left(G_{\{1:n-h\}}, \delta d_{\{1:n-h\}}\right) \le \varepsilon \qquad \text{Eq. 20}$$

where *f* is a scalar function that returns the desired convergence measure for the partition under consideration, *n* is the total number of freedoms for the partition, and *h* is the number of freedoms on the partition boundary, with $h = 0$ for the parent structure.

It should be noted that for a child partition the above expression deals with convergence to equilibrium inside the partition, since equilibrium at the partition boundary is dealt with at the parent level. This is reflected by the omission of the terms $G_{\{n-h+1:n\}}$, representing the resistance forces of the dual super element that are not typically zero at equilibrium, along with the corresponding $\delta d_{\{n-h+1:n\}}$ on the partition boundary.

## 2.4 Displacement Control

For static proportional loading, displacement control is typically applied to trace the equilibrium path following buckling, the formation of plastic mechanisms or the initiation of softening. The simplest variant of displacement control [24] prescribes over the current step an increment $\Delta d_j$ for a specific freedom *j*, which can be at the parent or child levels. In return, the load factor $\lambda$ becomes unknown and is therefore subject to iterative corrections

$\delta\lambda$ similar to displacements. Accordingly, the previous expressions for the iterative displacement corrections are modified at the parent and child levels to:

$$\delta d = K^{-1}\left(-G + \delta\lambda\, P^{o}\right) \qquad \text{Eq. 21}$$

where $P^{o}$ is a vector representing the nominal (un-scaled) proportional loads.

The solution of the resulting system of equations is most effectively undertaken by including $P^{o}$ as an additional vector along with *G* in the augmented matrix, at both the parent and child levels. The solutions associated with these vectors are then obtained using the proposed partitioned approach using the process of forward elimination followed by backward substitution, with:

$$\delta d^{G} = K^{-1}\left(-G\right) \qquad \text{Eq. 22}$$

$$d^{P} = K^{-1}\left(P^{o}\right) \qquad \text{Eq. 23}$$

$$\delta d = \delta d^{G} + \delta\lambda\, d^{P} \qquad \text{Eq. 24}$$

Given that the increment of the controlled displacement increment $\delta d_j$ is either the prescribed value $\Delta d_j$ at the start of the increment or 0 for subsequent iterations, with $\delta d_j = \delta d_j^{G} + \delta\lambda\, d_j^{P}$, this condition can be used with Eq. 24 to determine $\delta\lambda$ and hence $\delta d$ .

A more sophisticated variant of displacement control is the arc-length method [28], where the scalar product of the incremental displacements is typically controlled. The main difference between this method and basic displacement control is that $\delta\lambda$ is determined from a quadratic equation involving scalar products of $\delta d^{G}$ and $d^{P}$, which are still obtained as above.

## 3 Parallelisation Efficiency with Partitioning Approach

While the proposed partitioning approach can be implemented as a sequential procedure on a single computing processor, the main purpose behind its development is to achieve significant computational savings and reduction in memory demands via implementation on parallel processing systems with distributed memory. The computational savings that arise from undertaking element computations in parallel are obvious and are not addressed in this section. Instead, the following investigation aims at demonstrating through simple examples the conditions under which significant computational savings may be expected in the solution of the simultaneous equations of the overall system.

Focus is placed in this study on the computational cost of the forward elimination process using the frontal method, where the proposed partitioning approach is compared to the monolithic approach. The effectiveness of the frontal method is significantly enhanced when used with several partitions running in parallel on different processors. While the total CPU time required for the forward elimination phase does not always decrease as a result of partitioning, it is shown that the wall-clock time can be significantly reduced, as demonstrated in the following two examples. It should be noted that the influence of other computational aspects, such as backward substitution, element response calculation and inter-processor communication, are not considered here, though their implications are investigated later using the ADAPTIC implementation of the partitioned approach.

## 3.1 4×4 Grid

A basic mesh of 16 2D elements with 4 nodes each connected in the form of a 4×4 grid is considered, as shown in Figure 3(a). For presentational simplicity, it is assumed that each node has 1 DOF. Three possible scenarios are considered for the solution of this mesh, namely, the solution of full 25×25 matrix, frontal solution without partitioning, and frontal solution with partitioning. The computational efficiency of each scenario is considered in terms of both CPU time and wall-clock time. The exact time taken by the CPU depends upon numerous variables including the architecture of the particular machine and the communication network available for parallelisation; however, the count of multiplication operations required for the forward elimination phase of a system of equations provides a reasonably good comparison of efficiency.

In order to bring a matrix of size $n \times n$ to its upper triangular form, the number of multiplication operations required is approximately given by:

$$MOps = \frac{(n-1)n(n+1)}{3} \qquad \text{Eq. 25}$$

The full matrix for the grid under consideration is of size 25×25, thus the number of multiplication operations required to bring this matrix to its upper triangular form is 5200.

In the second scenario, the frontal method is considered for the solution of the same system. If the order of element assembly follows the numbering shown in Figure 3(a), there are a total of 16 reduction (or elimination) operations to be performed in order to bring the matrix to its upper triangular form, the details of which are shown in Table 1. The total number of multiplication operations required for carrying out all the 16 reduction operations is 704, which is approximately 14% of what is required for the full 25×25 matrix using conventional Gaussian elimination. This saving of approximately 86% demonstrates the efficiency of the frontal method in solving a system of algebraic equations arising from finite element discretisation.

In the third scenario, the frontal method is considered along with the proposed partitioning approach, where partitions are processed in parallel. Three cases of partitioning are considered: 2 partitions, 4 partitions, and 16 partitions, as illustrated in Figure 3(b,c,d).

In the first case, the mesh is partitioned in 2 halves, each consisting of 2×4 element grids, as shown in Figure 3(b). In this case, the 5 interface nodes on the partitioned boundary cannot be eliminated from the Grandpa at the child partition level but are eliminated at the parent level, as discussed in Sections 2.1 and 2.2. For solving each partition of 2×4 elements, 8 reduction operations are required (Table 2). Using the assembly order shown in Figure 3(b), the number of multiplication operations required is 332 per partition. In addition, 40 multiplication operations are required for the 5 interface nodes. Thus a total number of 704 multiplication operations are required for this method, which is identical in terms of CPU time to the frontal method using the 4×4 grid without partitioning. However, in terms of wall-clock time with parallel processing of the partitions, the number of multiplication operations undertaken sequentially is 372 (i.e. 332+40), which is only 53% of that required by the frontal solution without partitioning.

As an alternative partitioning scenario, the mesh is divided into 4 quarters of 2×2 element meshes, as shown in Figure 3(c). The solution of each partition following the depicted assembly order, and retaining the interface nodes for elimination at the parent level, consists of 4 reduction operations requiring 92 multiplications, as detailed in Table 3. The global solution in this case also requires 4 reduction operations, where the number of multiplication operations required for the global solution is 184. Thus the total number of multiplication operations in terms of CPU time required in this case is 552, which is interestingly much less than the CPU time required for the previous two cases. This highlights the absolute efficiency of the proposed partitioning approach as a practical implementation of a multifrontal solution scheme [22]. In terms of wall-clock time with parallel processing of the partitions, the number of multiplication operations undertaken sequentially is 276 (ie. 92+184), which is 39% and 74% of the wall-clock times of the cases with no partitions and with 2 partitions , respectively.

The last scenario considered is partitioning the mesh into 16 parts each consisting of a single element, as shown in Figure 3(d). This option does not offer much advantage with regard to the forward elimination process because all of the original nodes except for the four corner nodes are interface nodes and are carried into the global solution. The number of multiplication operations for solving all the partitions is 48, while that for the global solution is 596, as detailed in Table 4. Accordingly, the total number of multiplication operations related to CPU time is 644, and the sequential multiplication operations associated with wall-clock time is 608. The wall-clock efficiency is not commensurate in this case with the deployed computational resources consisting of 16 processors, since the wall-clock time reduces by a mere 14% compared to the frontal solution without partitioning. Importantly, the speedup deteriorates in comparison with the two previous partitioned cases, where the wall-clock time of the 16 partitions is 163% and 220% that of the cases with 2 and 4 partitions, respectively.

A summary of the speedup, defined as the ratio of the wall-clock time without partitioning to that with partitioning, is provided in Table 5. It should be emphasised again that this speedup is based on the computational demand of the forward elimination process, ignoring such factors as the inter-processor communication overhead and the computational demand of the element response calculations.

### 3.2 2×8 Strip

The same 16 2D elements used in the previous example are rearranged to constitute a mesh for a structure in the shape of a strip as shown in Figure 4(a), where the computational efficiency is again illustrated by considering the monolithic model and a partitioned model with two partitions. For the monolithic structure and considering an element assembly order as shown in Figure 4(a), there are 16 reduction operations required as detailed in Table 6. The total number of multiplication operations required in this case is 408.

For the partitioned model, as shown in Figure 4(b), there are 8 reduction operations required for each partition, as detailed in Table 7. In relation to CPU time, the total number of multiplication operations required is 408, which is same as for the full structure. In terms of wall-clock time, however, the number of sequential multiplication operations required in the partitioned analysis is 208 (ie. 200+8), which is around 51% that of the full structure as against 53% when the 4×4 grid structure of the previous example was divided into two partitions. This difference between the efficiency gained when the two example meshes are partitioned in two is partly because of their respective shapes. However, it also noted that for the same structure and number of partitions, improved wall-clock efficiency is typically achieved for a lower ratio between the number of interface nodes and the number of internal partition nodes, both in respect of solving the linearized system of equations and the inter-processor communication overhead.

## 3.3 Remarks on Efficient Partitioning

It is well established that parallelisation over an increasing number of processors eventually becomes less effective leading to a reduction in speedup rates due to the inter-processor communication overhead [29]. This fact aside, the above study has shown that even the solution of the linearized system of equations reaches a maximum wall-clock speedup at a specific level of partitioning, where excessive partitioning to the level of individual elements attains virtually no speedup. This result is consistent with Amdahl's Law [30], which states that the speedup of parallelisation is limited by the portion of the program that cannot be parallelised. From the perspective of solving the overall system of equations, optimal partitioning is typically achieved when there is a balance between the number of interface freedoms at the parent level $n_0$ and the number of freedoms internal to the child partitions $n_i - h_i$. For a rough guide, which assumes computational demand to be proportional to the number of eliminated freedoms, optimal partitioning is often realised when $n_0$ is equal to

$\max_i(n_i - h_i)$. However, there are real examples where at this point the front width at the parent may still be much less than that at the child level, in which case further partitioning can still enhance computational efficiency. This accords with the established benefits of the multifrontal solution method [22], and could even lead to computational efficiency that surpasses the optimal speedup (i.e. the number of processors). Indeed, this has already been demonstrated in practical applications of the proposed partitioning approach to the seismic assessment of multi-storey buildings [31], where a super-optimal speedup of 27 was achieved with 14 partitions running in parallel on an equal number of processors. Similarly, where the computational demand is dominated by the element response calculations, as opposed to the solution of equilibrium equations or inter-processor communication, optimal speedup may be achieved at relatively fine partitioning where $n_0 >> \max_i(n_i - h_i)$.

Of course, the above discussion assumes that hardware resources in terms of the number of processors are unlimited. In practice, the number of processors that can be used is limited, and this might at first sight limit the number of partitions, since ideally only one partition should be attached to a specific processor so as to maximise the wall-clock speedup arising from parallelisation. However, there are numerous real problems where it is beneficial from a modelling perspective to employ more partitions than available processors, in which case more than one partition would be attached to a single processor. While this can have an adverse effect on speedup with the simultaneous processing of multiple partitions on one processor, this can be significantly ameliorated if these partitions are scheduled for sequential processing using their frontal ordering at the parent level. This refinement, however, is outside the scope of the present paper.

## 4 Implementation of Partitioning Method

The proposed partitioning approach has been implemented in ADAPTIC [16], an advanced nonlinear structural analysis program developed at Imperial College London by Izzuddin and co-workers over the past 25 years.

ADAPTIC consists of two programs: i) READ which reads, checks and processes the problem data file, and ii) ANALYSE which takes as input intermediate processed files generated by READ to perform the required analysis. A shell script is used to run ADAPTIC, which launches the READ program and, subject to the absence of errors in the data syntax and structural model, starts the ANALYSE program subsequently.

For the purpose of implementing the proposed partitioned approach on a parallel processing system with distributed memory, the Message Passing Interface (MPI) [32] paradigm is employed. In this respect, MPI commands are included within the ADAPTIC shell script to launch several parallel processes of ANALYSE after the execution of READ. The number of processes of ANALYSE is one more than the number of partitions supplied by the user, where the additional process represents the parent subdomain including the placeholder super-elements and serves as the ‘coordinator’, with the remainder serving as ‘child partition processes’.

The process ranked 0 in the default MPI communicator is designated as the ‘coordinator’ and is responsible for processing the parent subdomain with the child partitions represented by the placeholder super-elements. This process assumes the control of the analysis and issues instructions to all the other processes, which are responsible for one child structure each and are designated as ‘partition processes’ (PPs). The incremental solution procedure followed by the coordinator is illustrated in the flowchart of Figure 5 for static proportional loading under load control, with similar procedures used for displacement control and time-history static/dynamic loading. Clearly, the coordinator control procedure is almost identical to what is followed in a monolithic approach, where placeholder super-elements are processed similar to conventional elements via task instructions to the corresponding dual super-elements in child partition processes. These instructions can be easily introduced in the solution procedure of existing nonlinear finite element analysis programs with minimal intervention, where Task 2 is the main instruction by which the response of the placeholder super-element is determined in accordance with Section 2.2.

Child partition processes have a rank greater than zero in the default MPI communicator and are each associated with a single partition, where the flowchart for the child partition solution procedure is illustrated in Figure 6, On start, each partition process reads the data for its partition, and then awaits instructions from the coordinator. For nonlinear static analysis under proportional loading using load control, a partition process can receive 7 task instructions, which are processed as follows:

- *Task 0, New step*: This task specifies that a new load step is to be started, where the partition process then expects to receive the load factor from the coordinator for the new load step.
- *Task 1, Start*: This task specifies that the analysis for the new load step is to be started. The loads are as per the new load factor received and the partition boundary displacements are as per the last equilibrium state. The condensed tangent stiffness and resistance forces for the partition boundary (dual super element) nodes are obtained using forward elimination, and sent back to the coordinator.
- *Task 2, Iterate*: This is the main task which establishes the response of the dual super-element. It obtains the iterative incremental displacements for the partition boundary, employs backward substitution to obtain the internal partition displacements, and performs forward elimination to determine the corresponding tangent stiffness and resistance forces.
- *Task 3, Re-equilibrate*: After achieving overall convergence at parent and child levels, the coordinator can instruct partition processes to re-equilibrate the current load step. This is typically required following automatic mesh refinement [33] at either the parent or child levels.
- *Task 4, Post-equilibrium*: Under this task, the partition process updates the equilibrium state and stores the results for the current load step in a partition output file.
- *Task 5, Reset all*: Under this task, the partition process resets the increments of displacement to zero. This is typically performed after convergence and update, or to initiate a new iterative procedure following lack of convergence.
- *Task 6, Finish*: This instruction terminates the partition process.

The child partition procedure in Figure 6 makes use of the same component routines employed in the parent procedure of Figure 5, except for two features. The first feature relates to the determination of the condensed resistance forces and tangent stiffness at the partitioned boundary, which are evaluated for the dual super-element under Tasks 1 and 2 in accordance with Section 2.2. The second important feature concerns the fact that control fully rests with the parent process, with the child process simply acting on control instructions from the parent process. Accordingly, both the parent and child partitions could make use of the same nonlinear analysis program, including the same library of elements, materials, solution methods, etc., executed under different MPI processes, only the parent process takes active control (Figure 5) while child processes adopt passive control (Figure 6). This approach lends itself to implementation for existing finite element analysis programs with minimal modification using MPI.

## 5 Verification and Application

This section presents several examples employing ADAPTIC [16] with the newly developed partitioning approach, highlighting first the maintained accuracy of the proposed approach compared to the conventional monolithic treatment, followed by demonstrating the computational speedup and ease of modelling that it provides.

### 5.1 Accuracy

The following three examples are aimed at demonstrating the accuracy of the proposed domain decomposition approach in geometric and material nonlinear finite element analysis.

#### *5.1.1 Lee's Frame*

The first example used to demonstrate the accuracy of the proposed partitioning approach is the well-known Lee's frame [34] depicted in Figure 7(a). The frame is first analysed as a whole structure using conventional monolithic analysis, and then it is considered using 3 partitions within a parent structure as shown in Figure 7(b) In both conventional monolithic analysis and parallel partitioned analysis, the frame is subjected to proportional loading, and the analysis is carried out in two phases. First, a load control phase is applied with 20 equal load factor steps $\Delta\lambda = 0.1$. Then, automatic displacement control is used after the load control phase terminates with convergence problems near the limit point. The automatic displacement control phase is terminated with user-defined conditions on the values of the load factor and nodal displacements. Considering the predicted displacements of the loaded node 3 in the two directions in Figure 8, it is clear that there is an excellent match between the results of conventional monolithic analysis and the proposed partitioning method for this highly geometric nonlinear problem.

### *5.1.2 Four-Storey Frame*

The second example is a four-storey frame, as depicted in Figure 9, which is subject to ground excitation with the earthquake signal shown in Figure 10. The mass is lumped at nodes, and the frame is modelled using an adaptive elasto-plastic method, where elastic quartic elements are initially used, which are subsequently automatically subdivided into elasto-plastic cubic elements when and where necessary [33]. For the partitioned analysis, the frame is divided into one parent structure and 4 partitions as shown in Figure 11. Each storey is modelled as a separate partition, where the concept of modular modelling is utilised to take advantage of the fact that the top three stories are similar. The results obtained from partitioned analysis match exactly those obtained using the conventional monolithic approach, as shown in Figure 12. A similar favourable comparison of the deformed shapes is shown in Figure 13.

### *5.1.3 RC Beam*

The third example verifying the accuracy of the present approach is a reinforced concrete beam subjected to static flexural loading, as shown in Figure 14. The concrete part is modelled with 1D cubic elasto-plastic elements that use inelastic uniaxial material response, while the reinforcement is also modelled with 1D elements that are linked to the concrete elements using rigid link elements. A simple trilinear model is used for concrete ignoring the tensile strength, while a bilinear model is used for steel [33], where the associated material properties are given in Table 8. The beam is modelled monolithically and with 4 partitions, where a favourable comparison of the deflected shapes is shown in Figure 15. A further comparison of the load-deflection response, as depicted in Figure 16, provides an exact match demonstrating the accuracy of the proposed partitioned approach in geometric and material nonlinear analysis.

## 5.2 Computational and Modelling Benefits

Three examples are presented hereafter to demonstrate the computational efficiency of the proposed parallel partitioning approach, as well as the modelling advantages that arise from its modular features.

### *5.2.1 I-Beam*

The first example considers an elastic I-beam modelled with 20-noded 3D brick element (Figure 17), where the flanges and web are discretised using 5 brick elements each, with 10 element divisions used along the beam length leading to a total of 150 brick elements. The load is applied at mid-span in 200 steps with a constant increment of 60 kN to a total of 12 MN. The beam is analysed as a single monolithic structure and as a partitioned structure with using 2, 3, and 5 partitions, as illustrated in Figure 17. Since the present partitioning scheme processes the partitions in parallel and requires their response before completion of the solution at the parent level, it is expected that the time taken is approximately equal to that required for analysing the largest partition, subject to the order of the partitions in the frontal solution at the parent level, in addition to the time required by the parent structure and some communication overhead. A comparison of the equivalent degrees of freedom that are processed sequentially (i.e. the sum of the freedoms of the largest partition and the parent structure) can give a good estimate of the expected time saved. For the current example, the number of equivalent degrees of freedom is provided in Table 9 for all cases. It should be noted that the communication overhead will be in direct proportion to the size of the parent structure, leading to more time required for the case of 5 partitions than stipulated by the comparison based on equivalent freedoms in Table 9.

To investigate the correlation between the anticipated and actual wall-clock time savings, the various analyses are carried out 20 times for each case, and the time taken to perform each analysis is presented in Figure 18. The results show that the equivalent number of freedoms provides a reasonable first-order approximation of wall-clock time savings, as evidently the average savings of the partitioned cases are between 50% and 60%. Interestingly, the case with 2 partitions appears to be the most efficient in this case, contrary to the stipulation based on equivalent number of freedoms, which may be attributed to the lower communication overhead compared to the two other partitioned cases.

### *5.2.2 Slab*

The second example is a square slab consisting of 1024 (32×32×1) 3D 20-noded brick elements. The slab is simply supported and the load is applied uniformly at all the nodes on the top surface in 200 steps. The slab is analysed as a single structure and is also considered as a partitioned structure with 4 and 16 partitions, each consisting of 16×16×1 and 8×8×1 elements, respectively, as illustrated in Figure 19. This example also demonstrates the ease of modelling achieved by the present domain partitioning approach. As can be seen in Figure 19, the partitions are similar and can be modelled easily by making copies. In the case of 4 partitions for example, partition 2 can be easily modelled by copying partition 1 and then making the necessary changes in the support conditions and the partitioned boundary. In the case of 16 partitions, some partitions are identical in all respects and need just copying without any modifications (e.g. partitions 5 and 12, or partitions 6, 7, 10 and 11).

In order to study the effects of partitioning on computational efficiency in relation to the varying computational demand of evaluating the element response contributions compared to the solution of the system of equilibrium equations at structural level, the same slab models are analysed using different number of Gauss points. In the first set of runs, all cases (monolithic, 4 partitions, and 16 partitions) are analysed with 8 Gauss points per brick element for 10 times, where the comparison of results is shown in Figure 20. The average time taken by the monolithic models is 3587 seconds, which is about 7 times that taken by the structure modelled with 4 partitions that required an average of 495 seconds. The time taken by the structure modelled with 16 partitions is, however, greater than that of the 4 partitions, standing at an average of 798 seconds. This increase in the wall-clock time requirement is due to the increased size of the parent structure as the computational demand of the element response contributions is relatively low in this case. This fact is further verified by increasing the number of Gauss points from 8 to 27 in the second set of runs. The effects of this increase on the computational demand are not significantly visible, as the wall-clock time requirements of each case increased only slightly, as shown in Figure 20. A third set of runs is considered with 1000 Gauss points per element, representing a type of problem in which the evaluation of the element response is relatively computationally expensive. As observed from the results in Figure 21, the average time taken by the monolithic slab is now 7940 seconds, which is about 5 times that of the 4 partitions standing at 1507 seconds. The time taken by the slab modelled with 16 partitions now stands at 1042 seconds, which is actually lower than that of the 4 partitions. This indicates that the computational efficiency of parallelisation can continue to improve with partitioning when the evaluation of the element or partition response continues to be associated with a significantly high computational demand compared to that of the solution of system of equations at the parent structural level.

### *5.2.3 Cellular Beam*

The final example is a cellular beam, a type of structural member that has gained increased popularity because of its ability to withstand gravitational loads over large spans whilst allowing the integration of services within the beam depth. The presence of holes in the web of the beam, however, causes local buckling in the web-post and/or compression regions around the openings. Work on the simplified and detailed analysis of this type of structure has been recently undertaken at Imperial College London as part of an independent PhD research programme [35]. This example is presented for illustrative purposes, demonstrating that the modelling of such seemingly complex structures is simplified with the use of modular modelling, and highlighting the computational efficiency of the proposed partitioned approach.

The cellular beam under consideration spans over 30 meters and has a total of 32 holes in its web that are spaced 0.92 m apart centre to centre. The first hole is situated at a distance of 0.74 m from the left end of the beam as shown in Figure 22(a). The beam has 25.4 mm thick and 268 mm wide flanges, 15.6 mm thick web, a total depth of 1165.2 mm and hole diameters of 800 mm each as shown in Figure 22(b,c). The web posts between two consecutive holes have a minimum width of 120 mm.

The entire model consists of 9-noded shell elements [36] that are used to model the flanges and web regions. The domain partitioning for this example is relatively straightforward due to the fact that it consists of 31 identical unit cells in addition to the 2 end units. Therefore, it is advantageous to make each unit cell a child partition, resulting in the need to create only 3 data files in addition to the parent structure which consists of nodes for the 32 cross-sections, equally spaced at 0.92 m.

The cellular beam is subjected to a proportional vertical uniformly distributed load, with a nominal value of 10kN/m, which is specified internally at the partition level. Since this is a problem dominated by local buckling of the web-post, random imperfections are introduced via very small out-of-plane loads. Importantly, as the post-buckling response is associated with snap-back behaviour, the arc-length displacement control method is used beyond the limit point after an initial phase of load control.

Deflected shape at limit point is shown in Figure 23 whereas the final deflected shape is shown in Figure 24. For illustrative purposes, a close up of partition 29 with contours of the normal stress in the longitudinal direction is provided in Figure 25. It can be seen, as expected, that the web-posts buckle near the support due to significant shear forces combined with compression resulting from applying the UDL on top of the beam.

The load-deflection response of the beam is provided in Figure 26, where it is clear that the arc-length method is successful with the proposed partitioning approach in tracing the snap-back post-buckling response. Importantly, the whole analysis is undertaken on 34 processors in 23min 20sec of wall-clock time, while identical results are obtained with a single-processor monolithic model in 4hr 57min, thus representing a significant speed-up of around 13.

## 6 Conclusions

A new domain decomposition method for nonlinear finite element analysis introducing the concept of dual partition super-elements has been presented. The method is ideally suited for parallel nonlinear static/dynamic analysis of structural systems. The proposed method offers a practical approach which can be readily implemented for existing finite element analysis programs to achieve parallelisation on distributed memory systems with minimal intervention, thus overcoming memory bottlenecks typically faced in the analysis of large scale problems. The proposed partitioning method utilises the exact tangent stiffness matrix at the interface boundary, and therefore it has identical convergence characteristics to the monolithic approach.

The examples presented in Section 5.1 have demonstrated that the results obtained from the proposed parallel partitioning approach are identical to those obtained from conventional monolithic analysis. It has also been demonstrated in Section 5.2 that the computational efficiency is congruent with the equivalent number of total freedoms, taken as the size of the largest partition in addition to the parent structure. This is particularly the case when the solution of the system of simultaneous equilibrium equations dominates the computational demand. Together with the inter-processor communication overhead, this has the effect of diminishing return with increased partitioning, where the wall-clock time may increases from its optimal value for partitions exceeding a problem-specific number. On the other hand, it was shown that the optimal number of partitions can be significantly increased for problems where the computational demand is dominated by nonlinear finite element computations, simulated in a specific example by increasing the number of Gauss points.

The primary benefits of the proposed parallel partitioned approach include significant speedup due to parallelisation, overcoming memory bottlenecks on distributed memory systems, multi-frontal solution of the overall system of equations, and modelling benefits where identical partitions may be reused for modular/repetitive structures. The proposed parallel partitioning scheme also provides a natural framework to incorporate coupling of partitions with mixed element dimensions, mixed integration schemes and multi-level hierarchic partitioning, which are the focus of ongoing work.

## Acknowledgements

The authors acknowledge the Higher Education Commission of Pakistan for funding the PhD research of the first author, and the High Performance Computing (HPC) Services at Imperial College London for providing and supporting the required computing facilities.

## References

1. Smith, B., Bjorstad, P., and Gropp, W., (1996). Domain Decomposition: Parallel Multilevel Methods for Eliptic Partial Differential Equations. Cambridge University Press, United Kingdom.
2. Dawson, C.N., Du, Q., and Dupont, T.F., (1991). A finite difference domain decomposition algorithm for numerical solution of the heat equation. Mathematics of Computation, Americal Mathematical Society. Vol. 57. No. 195. P. 63-71.
3. Felippa, C.A., and Park, K.C., (1980). Staggered transient analysis procedures for coupled-field mechanical systems: Formulation. Computer Methods in Applied Mechanics and Engineering. Vol. 24. P. 61-111.
4. Zolghadr Jahromi, H., Izzuddin, B.A., & Zdravkovic, L. (2007). Partitioned analysis of nonlinear soil-structure interaction using iterative coupling. Interaction and Multiscale Mechanics. Vol. 1. No. 1. P. 33-51.

5. Felippa, C.A., Park, K.C., & Farhat, C., (2001). Partitioned analysis of coupled mechanical systems. Computer Methods in Applied Mechanics and Engineering. No. 190. P. 3247-3270.
6. Quarteroni, A., & Valli, A., (1999). Domain Decomposition Methods for Partial Differential Equations, Oxford, Clarendon press.
7. Marini, L. D., & Quarteroni, A., (1989). A relaxation procedure for domain decomposition methods using finite elements. Numerische Mathematik, No. 55, P. 575-598.
8. Mu, M., (1999). Solving composite problems with interface relaxation. SIAM Journal on Scientific Computing, No. 20, P. 1394-1416.
9. Zolghadr Jahromi, H., Izzuddin, B.A., & Zdravkovic, L. (2009). A domain decomposition approach for coupled modelling of nonlinear soil-structure interaction. Computer Methods in Applied Mechanics and Engineering. Vol. 198. No. 33. P. 2738-2749.
10. Kron, G. (1939). Tensor Analysis of Networks. John Wiley & Sons. New York.
11. Farhat, C., & Roux, F.X., (1991). A method of finite element tearing and interconnecting and its parallel solution algorithm. International Journal of Numerical Methods in Engineering. Vol. 32. P. 1205-1227.
12. Magoules, F. & Roux, F.X. (2007). Algorithms and Theory for Substructuring and Domain Decomposition Methods. In: Magoules, F. (ed). Mesh Partitioning Techniques and Domain Decomposition Methods. Saxe-Coburg Publications.
13. Kruis, J. (2007). The FETI-DP Method. In: Magoules, F. (ed). Mesh Partitioning Techniques and Domain Decomposition Methods. Saxe-Coburg Publications.
14. Zienkiewicz, O.C., Taylor, R.L., & Zhu, J.Z., (2005). The Finite Element Method: Its Basis and Fundamentals. The Fininte Element Method Set Volume 1. 6th Edition. Elsevier Ltd.
15. Irons, B.M., (1970). A frontal solution program for finite-element analysis, International Journal of Numerical Methods in Engineering, Vol. 2, No. 1, P. 5-32.
16. Izzuddin, B.A., (1991). Nonlinear dynamic analysis of framed structures, PhD Thesis, Civil Engineering Department, Imperial College of Science and Technology, London, December 1990.

17. Rao, A.A., (2005) MPI-based parallel finite element approaches for implicit nonlinear dynamic analysis employing sparse PCG solvers. Advances in Engineering Software Vol. 36, No. 3

18. B. Gullerud, Arne S. (2001), and Robert H. Dodds Jr. "MPI-based implementation of a PCG solver using an EBE architecture and preconditioner for implicit, 3-D finite element analysis. Computers & Structures Vol. 79, No. 5

19. Ramakrishnan C.V., and Kumar, S.R. (2002) Comparative Peroformance of Frontal (Direct) and PCG (Iterative) Solver Based Parallel Computations of Finite Element Analysis. Proceedings of the International Parallel and Distributed Processing Symposium, IEEE

20. Cross, M., Croft, T.N., Slone, A.K., Williams, A.J., Christakis, N., Patel, M.K., Bailey, C., and Pericleous, K. (2008) Computational Modelling of Multi-Physics and Multi-Scale Processes in Parallel. International Journal for Computational Methods in Engineering Science and Mechanics, Vol. 8, No. 2. P. 63-74.

21. Jokhio, G.A., & Izzuddin, B.A., (2013). Parallelisation of Nonlinear Structural Analysis using Dual Partition Super Elements. Advances in Engineering Software, Vols. 60-61, P. 81-88.

22. Amestoy, P.R., Duff, I.S., & L'Excellent, J.-Y., (2000). Multifrontal Parallel Distributed Symmetric and Unsymmetric Solvers. Computer Methods in Applied Mechanics and Engineering, Vol. 184, No. 2–4, P. 501–520

23. Izzuddin, B.A., & Jokhio, G.A., (2013). Mixed-Dimensional Coupling for Parallel Partitioned Nonlinear Finite Element Analysis. (*under review*).

24. Felippa, C.A., & Park, K.C., (1980). Staggered transient analysis procedures for coupled-field mechanical systems: Formulation. Computer Methods in Applied Mechanics and Engineering. Vol. 24. P. 61-111.

25. Reid, J.K., (1971). On the method of conjugate gradients for the solution of large sparse systems of linear equations. Proceedings of the Oxford conference of the Institute of Mathematics and its Applications held in April 1970. Academic Press. London. P. 283.

26. Sloan, S.W., & Randolph, M.F. (1982). Numerical prediction of collapse loads using finite element methods. International Journal for Numerical and Analytical Methods in Geomechanics. Vol. 6. P. 47-76.

27. Brezinski, C., (2005). Schur Complements and Applications in Numerical Analysis. In The Schur Complement and Its Application, F. Zhang (Ed.), Springer.

28. Cristfield, M.A., (1991). Non-linear Finite Element Analysis of Solids and Structures. Volume 1: Essential. Wiley: Chichester, U.K.

29. Xu, Z., & Hwang, K., (1996). Modelling communication overhead: MPI and MPL performance on the IBM SP2. Parallel & Distributed Technology: Systems & Applications, IEEE. Vol. 4, No. 1. P. 9-24.

30. Amdahl, G. (1967). Validity of the single processor approach to achieving large-scale computing capabilities. AFIPS Conference Proceedings No. 30. P. 483-485.

31. Izzuddin, B.A., Macorini, L., & Rinaldin, G., (2013). Partitioned Modelling for Nonlinear Dynamic Analysis of Reinforced Concrete Buildings for Earthquake Loading. Proceedings of 14th International Conference on Civil, Structural and Environmental Engineering Computing (CC2013), B.H.V. Topping and P. Iványi, (Editors), Civil-Comp Press, Stirlingshire, Scotland.

32. Message Passing Interface Forum, (2012). MPI: A Message Passing Interface Standard. Version 3.0. [Online]. September 2012. Available From: http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf[Accessed: 05 May 2013].

33. Izzuddin, B.A., and Elnashai, A.S., (1993). Adaptive space frame analysis – Part II: A distributerd plasticity approach. Proceedings of the Institute of Civil Engineers, Structures and Buildings, Vol. 99. P. 317-326.

34. Lee, S.L., Manuel, F.S., and Rossow, E.C., (1968). Large deflections and stability of elastic frames. J. Eng. Mech. Div., ASCE, Vol. 94. No. EM2. P. 521-547.

35. Zainal-Abidin, A.R., (2012). Modelling of Local Elastic Buckling for Steel Beams with Web Openings, PhD Thesis, Department of Civil and Environmental Engineering, Imperial College London.

36. Izzuddin, B.A., (2007). An Optimisation Approach towards Lock-Free Finite Elements, Proceedings of the 11th International Conference on Civil, Structural and Environmental Engineering Computing, Paper No. 105, B.H.V. Topping (Editor), Civil-Comp Press, Stirlingshire, Scotland.

| Reduction operation | Size of Grandpa before reduction | Size of Grandpa after reduction | Nodes eliminated | Multiplication operations |
|---|---|---|---|---|
| 1 | 4×4 | 3×3 | 1 | 12 |
| 2 | 5×5 | 4×4 | 1 | 20 |
| 3 | 6×6 | 5×5 | 1 | 30 |
| 4 | 7×7 | 5×5 | 2 | 72 |
| 5 | 7×7 | 6×6 | 1 | 42 |
| 6 | 7×7 | 6×6 | 1 | 42 |
| 7 | 7×7 | 6×6 | 1 | 42 |
| 8 | 7×7 | 5×5 | 2 | 72 |
| 9 | 7×7 | 6×6 | 1 | 42 |
| 10 | 7×7 | 6×6 | 1 | 42 |
| 11 | 7×7 | 6×6 | 1 | 42 |
| 12 | 7×7 | 5×5 | 2 | 72 |
| 13 | 7×7 | 5×5 | 2 | 72 |
| 14 | 6×6 | 4×4 | 2 | 50 |
| 15 | 5×5 | 3×3 | 2 | 32 |
| 16 | 4×4 | 0 | 4 | 20 |
| **Total** | | | **25** | **704** |

Table 1: Reduction operations for a 4×4 grid using frontal method

| **Reduction Operation** | **Size of Grandpa Before Reduction** | **Size of Grandpa After Reduction** | **Nodes Eliminated** | **Multiplication Operations** |
|---|---|---|---|---|
| Operations at Partition Level | | | | |
| 1 | 4×4 | 3×3 | 1 | 12 |
| 2 | 5×5 | 4×4 | 1 | 20 |
| 3 | 6×6 | 5×5 | 1 | 30 |
| 4 | 7×7 | 5×5 | 2 | 72 |
| 5 | 7×7 | 6×6 | 1 | 42 |
| 6 | 7×7 | 6×6 | 1 | 42 |
| 7 | 7×7 | 6×6 | 1 | 42 |
| 8 | 7×7 | 5×5 | 2 | 72 |
| Total | | | **10** | **332** |
| Operations at Global Level | | | | |
| 1 | 5×5 | 0 | 5 | 40 |

Table 2: Reduction operations required for 2 partitions

| Reduction Operation | Size of Grandpa Before Reduction | Size of Grandpa After Reduction | Nodes Eliminated | Multiplication Operations |
|---|---|---|---|---|
| Operations at Partition Level | | | | |
| 1 | 4×4 | 3×3 | 1 | 12 |
| 2 | 5×5 | 4×4 | 1 | 20 |
| 3 | 6×6 | 5×5 | 1 | 30 |
| 4 | 6×6 | 5×5 | 1 | 30 |
| **Total** | | | **4** | **92** |
| Operations at Global Level | | | | |
| 1 | 5×5 | 5×5 | 0 | 0 |
| 2 | 7×7 | 5×5 | 2 | 72 |
| 3 | 7×7 | 5×5 | 2 | 72 |
| 4 | 5×5 | 0 | 5 | 40 |
| **Total** | | | **9** | **184** |

Table 3: Reduction operations required for 4 partitions

| **Reduction Operation** | **Size of Grandpa Before Reduction** | **Size of Grandpa After Reduction** | **Nodes Eliminated** | **Multiplication Operations** |
|---|---|---|---|---|
| Operations at Partition Level | | | | |
| 1 | 4×4 | 3×3 | 1 | 12 |
| Total Multiplication Operations for Corner Partitions | | | | 12 |
| Operations at Global Level | | | | |
| 1 | 3×3 | 3×3 | 0 | 0 |
| 2 | 5×5 | 4×4 | 1 | 20 |
| 3 | 6×6 | 5×5 | 1 | 30 |
| 4 | 6×6 | 5×5 | 1 | 30 |
| 5 | 7×7 | 6×6 | 1 | 42 |
| 6 | 7×7 | 6×6 | 1 | 42 |
| 7 | 7×7 | 6×6 | 1 | 42 |
| 8 | 7×7 | 5×5 | 2 | 72 |
| 9 | 7×7 | 6×6 | 1 | 42 |
| 10 | 7×7 | 6×6 | 1 | 42 |
| 11 | 7×7 | 6×6 | 1 | 42 |
| 12 | 7×7 | 5×5 | 2 | 72 |
| 13 | 6×6 | 5×5 | 1 | 30 |
| 14 | 6×6 | 4×4 | 2 | 50 |
| 15 | 5×5 | 3×3 | 2 | 32 |
| 16 | 3×3 | 0 | 3 | 8 |
| **Total Multiplication Operations at Global Level** | | | **21** | **596** |

Table 4: Reduction operations required for 16 partitions

| **Number of Partitions** | **Sequential multiplications** | **Speedup** |
|---|---|---|
| 2 | 372 | 1.89 |
| 4 | 276 | 2.55 |
| 16 | 608 | 1.16 |

Table 5: Summary of speed up

| **Reduction Operation** | **Size of Grandpa Before Reduction** | **Size of Grandpa After Reduction** | **Nodes Eliminated** | **Multiplication Operations** |
|---|---|---|---|---|
| 1 | 4×4 | 3×3 | 1 | 12 |
| 2 | 5×5 | 3×3 | 2 | 32 |
| 3 | 5×5 | 4×4 | 1 | 20 |
| 4 | 5×5 | 3×3 | 2 | 32 |
| 5 | 5×5 | 4×4 | 1 | 20 |
| 6 | 5×5 | 3×3 | 2 | 32 |
| 7 | 5×5 | 4×4 | 1 | 20 |
| 8 | 5×5 | 3×3 | 2 | 32 |
| 9 | 5×5 | 4×4 | 1 | 20 |
| 10 | 5×5 | 3×3 | 2 | 32 |
| 11 | 5×5 | 4×4 | 1 | 20 |
| 12 | 5×5 | 3×3 | 2 | 32 |
| 13 | 5×5 | 4×4 | 1 | 20 |
| 14 | 5×5 | 3×3 | 2 | 32 |
| 15 | 5×5 | 3×3 | 2 | 32 |
| 16 | 4×4 | 0 | 4 | 20 |
| **Total Number of Multiplication Operations for all Reductions** | | | | **408** |

Table 6: Reduction operations for the strip of 16 2D elements

| **No. of Reduction Operation** | **Size of Grandpa Before Reduction** | **Size of Grandpa After Reduction** | **Nodes Eliminated** | **Multiplication Operations** |
|---|---|---|---|---|
| Operations at Partition Level | | | | |
| 1 | 4×4 | 3×3 | 1 | 12 |
| 2 | 5×5 | 3×3 | 2 | 32 |
| 3 | 5×5 | 4×4 | 1 | 20 |
| 4 | 5×5 | 3×3 | 2 | 32 |
| 5 | 5×5 | 4×4 | 1 | 20 |
| 6 | 5×5 | 3×3 | 2 | 32 |
| 7 | 5×5 | 4×4 | 1 | 20 |
| 8 | 5×5 | 3×3 | 2 | 32 |
| Total Multiplication Operations for the Partition | | | | **200** |
| Operations at Global Level | | | | |
| 1 | 3×3 | 0 | 3 | 8 |

Table 7: Reduction operations required for the strip in 2 partitions

| Concrete | |
|---|---|
| Secant compressive stiffness | 25,000 MPa |
| Compressive strength | 45 MPa |
| Compressive softening stiffness | -5,000 MPa |
| Residual compressive strength | 10 MPa |
| Steel | |
| Young’s modulus | 210,000 MPa |
| Strength | 300 MPa |
| Strain-hardening factor | 0.01 |

Table 8: Material properties used for reinforced concrete beam

| | Monolithic | 2 Partitions | 3 Partitions | 5 Partitions |
|---|---|---|---|---|
| **DoFs for the largest partition** | 1178 | 628 | 518 | 298 |
| **DoFs for the parent structure** | 0 | 78 | 156 | 312 |
| **Total Equivalent DoFs** | 1178 | 706 | 674 | 610 |
| **% of Monolithic** | **100** | **59.93** | **57.22** | **51.8** |

Table 9: Comparison of equivalent DoFs

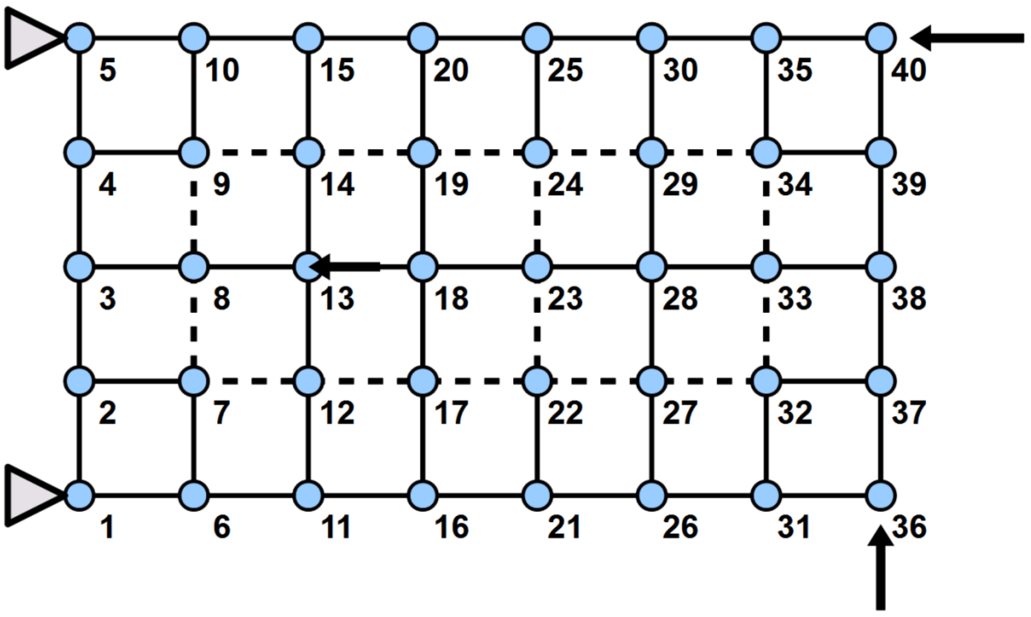
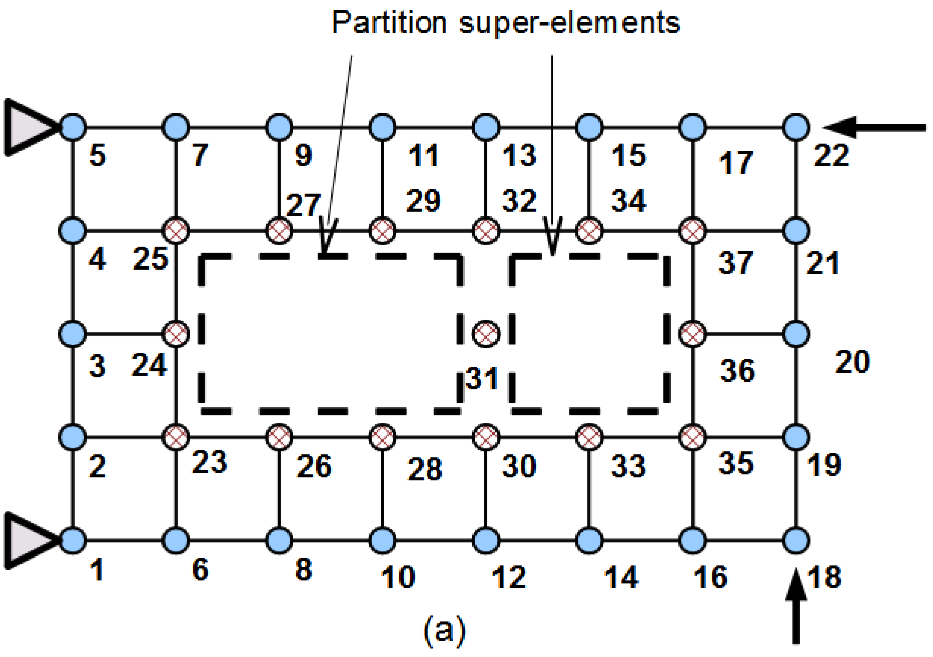
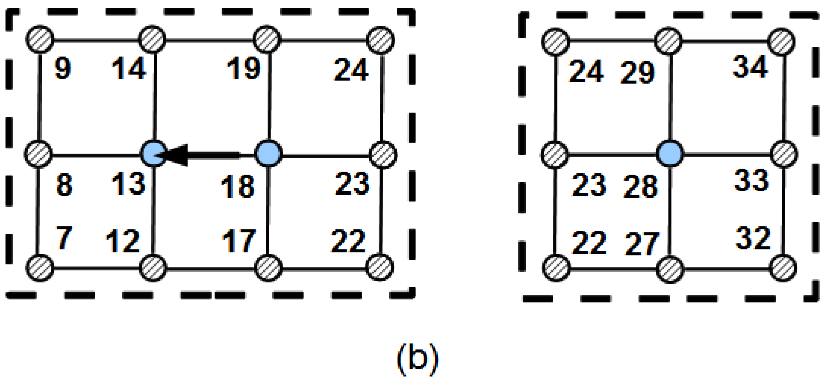


Figure 1: An illustrative structural domain





● Internal nodes

⊗ Nodes connected to two super-elements

▨ Boundary Nodes connected to dual super-elements

Figure 2: Partitioned structure (a) parent structure (b) child partitions

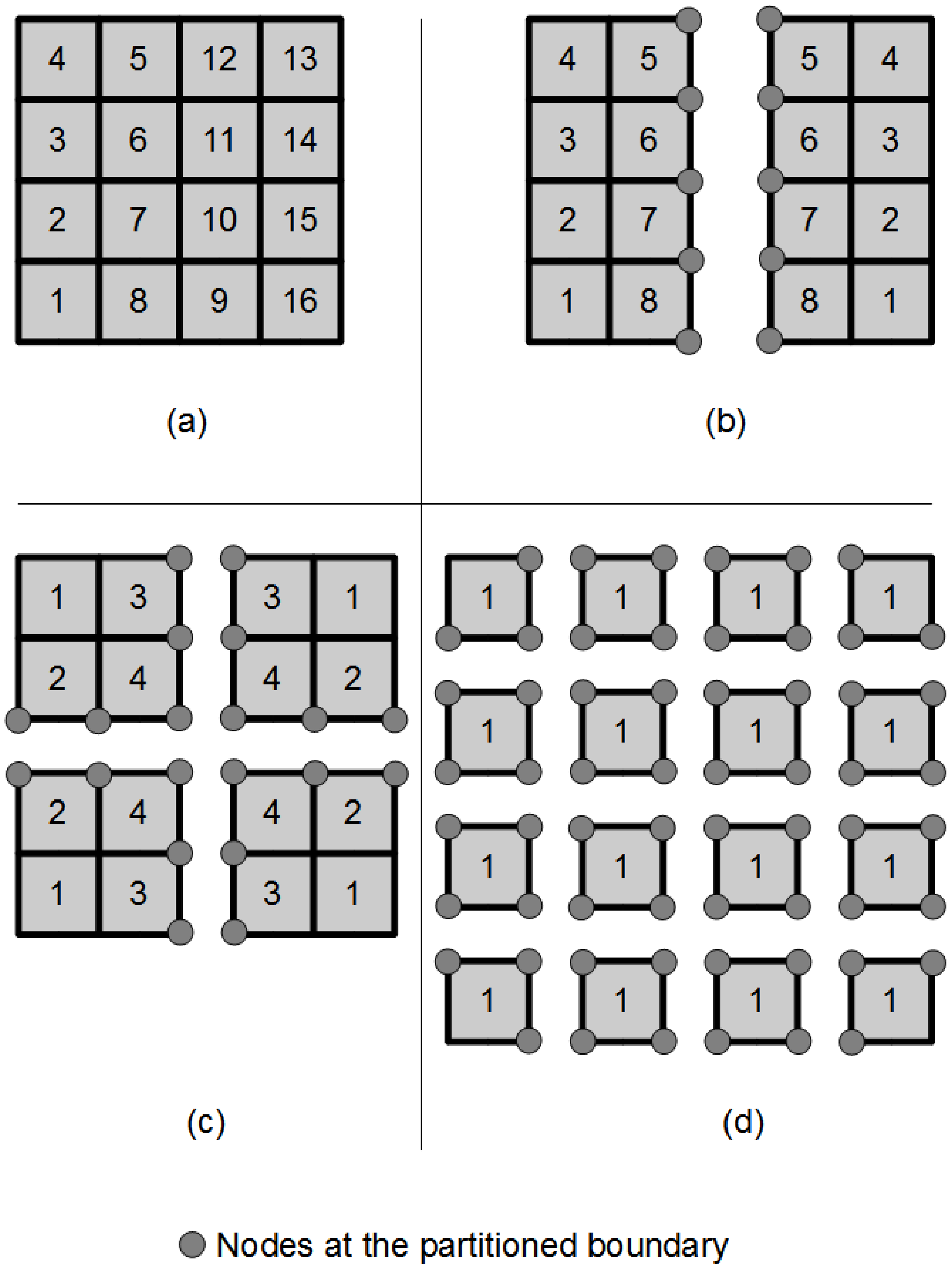
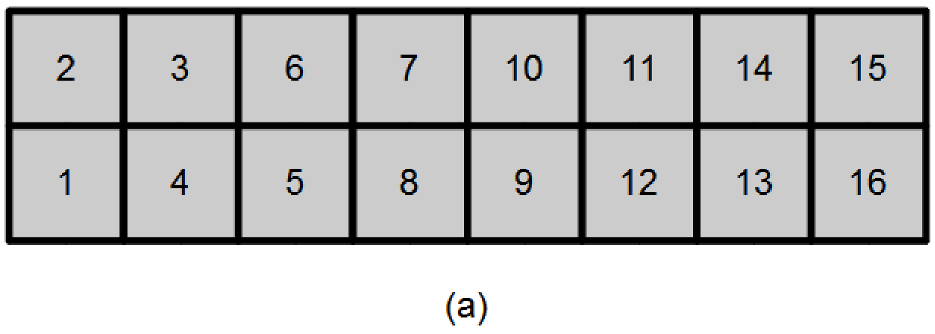
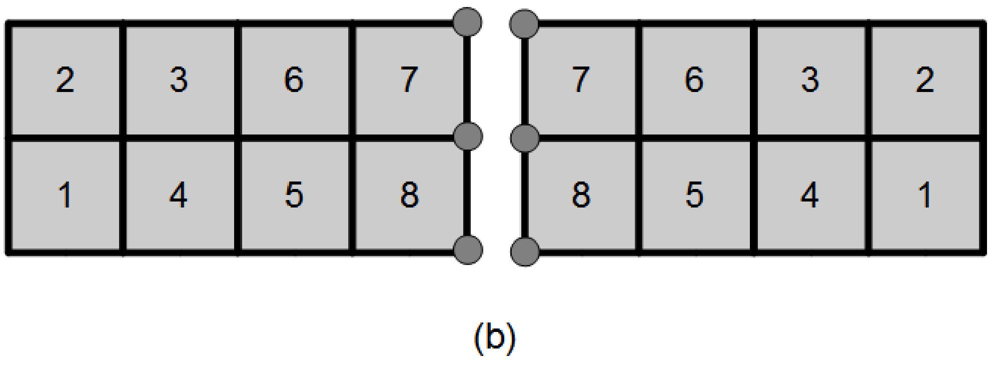


Figure 3: A mesh of 16 2D elements (a) Original mesh (b) 2 partitions (c) 4 partitions (d) 16 partitions





● Nodes at the partitioned boundary

Figure 4: Mesh of 16 2D elements (a) in the shape of a strip (b) the strip partitioned in two

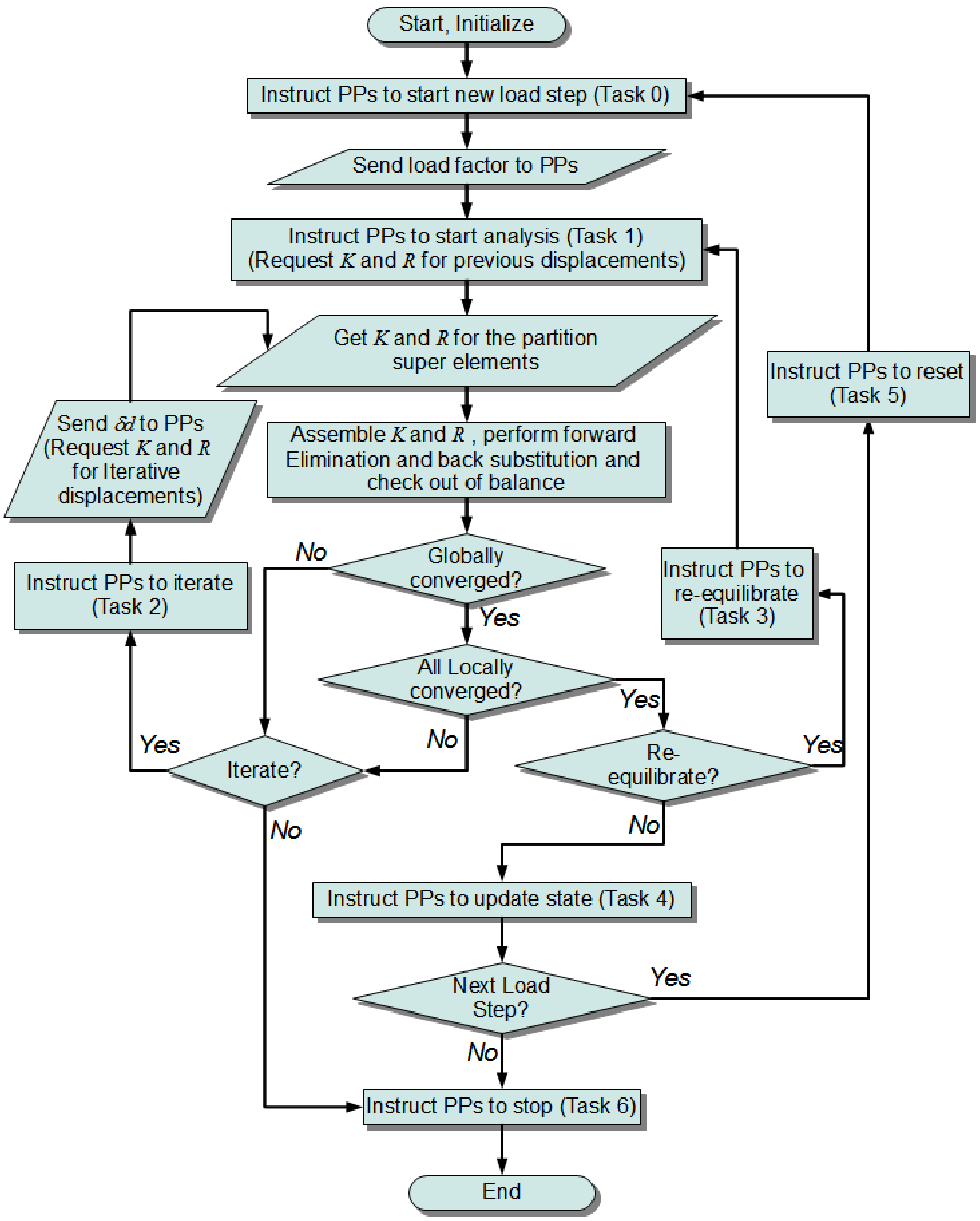

Start, Initialize
Instruct PPs to start new load step (Task 0)
Send load factor to PPs
Instruct PPs to start analysis (Task 1) (Request $K$ and $R$ for previous displacements)
Get $K$ and $R$ for the partition super elements
Assemble $K$ and $R$ , perform forward Elimination and back substitution and check out of balance
Globally converged?
No
Yes
All Locally converged?
Yes
No
Iterate?
Yes
No
Re-equilibrate?
Yes
No
Instruct PPs to iterate (Task 2)
Send $\delta d$ to PPs (Request $K$ and $R$ for Iterative displacements)
Instruct PPs to re-equilibrate (Task 3)
Instruct PPs to reset (Task 5)
Instruct PPs to update state (Task 4)
Next Load Step?
Yes
No
Instruct PPs to stop (Task 6)
End

Figure 5: Flow chart of the coordinator process (PPs stands for partitions processes)

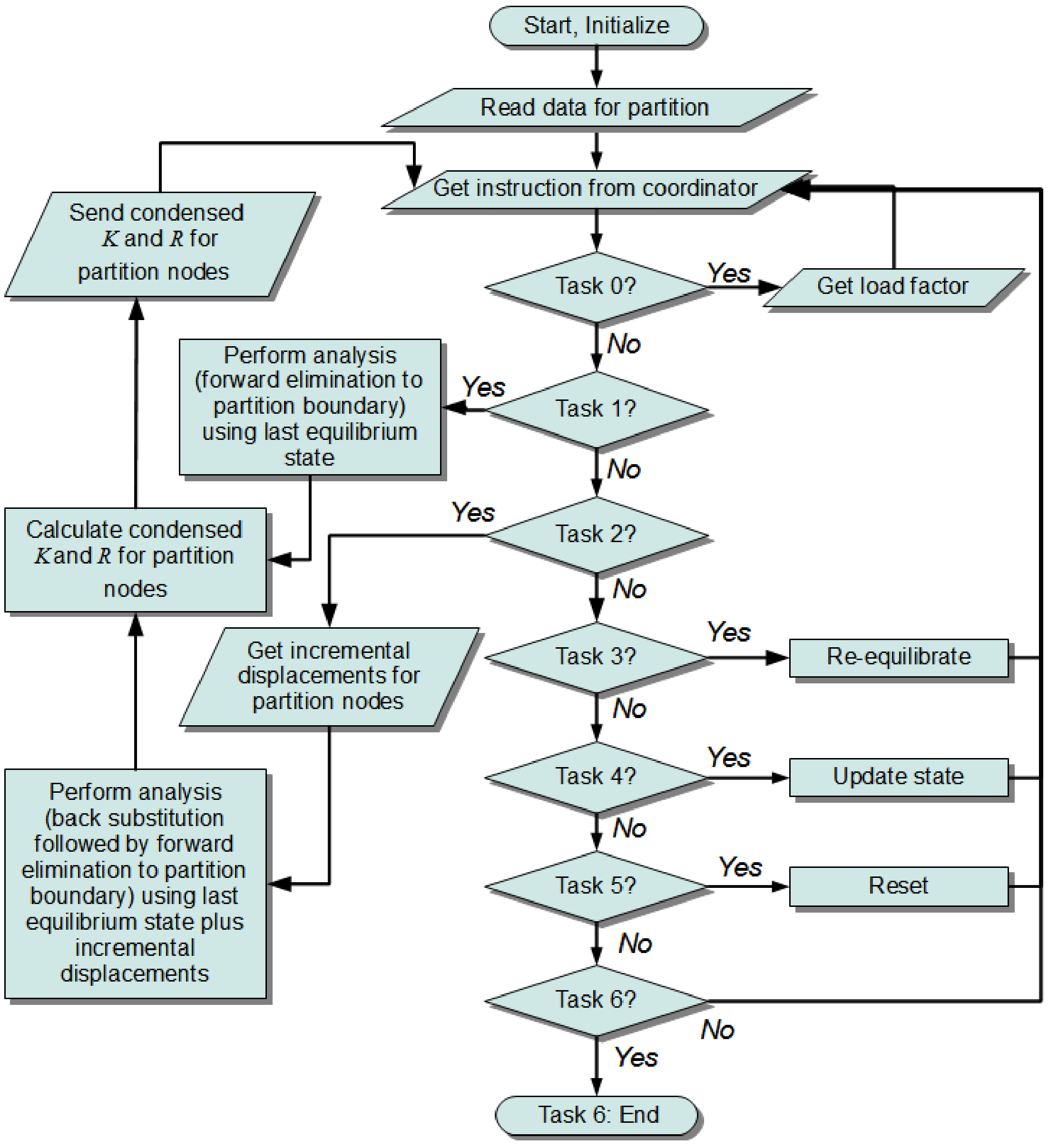

Start, Initialize
Read data for partition
Get instruction from coordinator
Send condensed $K$ and $R$ for partition nodes
Task 0?
Yes
Get load factor
No
Perform analysis (forward elimination to partition boundary) using last equilibrium state
Yes
Task 1?
No
Calculate condensed $K$ and $R$ for partition nodes
Yes
Task 2?
No
Get incremental displacements for partition nodes
Task 3?
Yes
Re-equilibrate
No
Task 4?
Yes
Update state
No
Perform analysis (back substitution followed by forward elimination to partition boundary) using last equilibrium state plus incremental displacements
Task 5?
Yes
Reset
No
Task 6?
No
Yes
Task 6: End

Figure 6: Flow chart of a child partition process

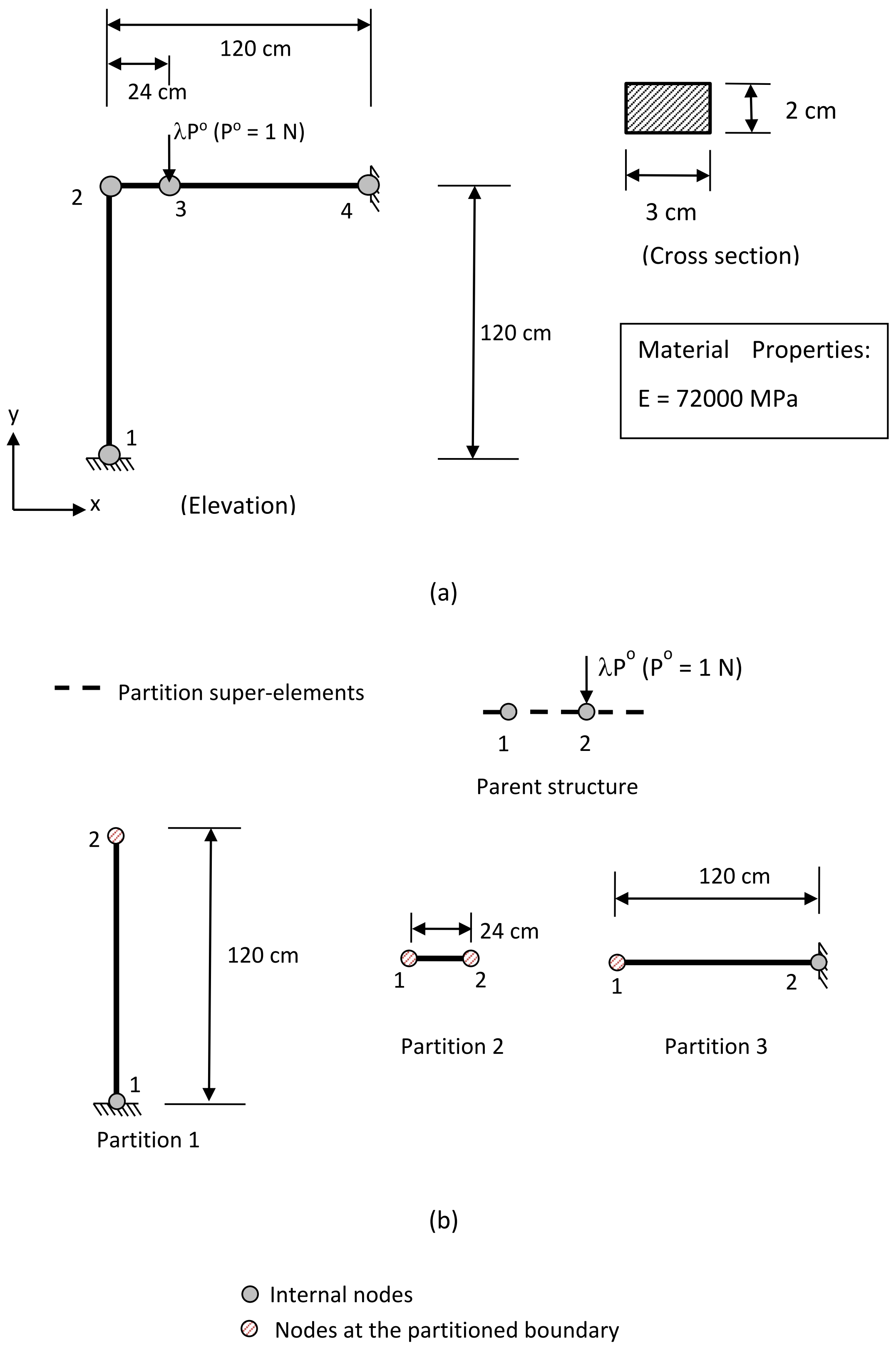


Figure 7: Lee's frame (a) monolithic (b) partitioned

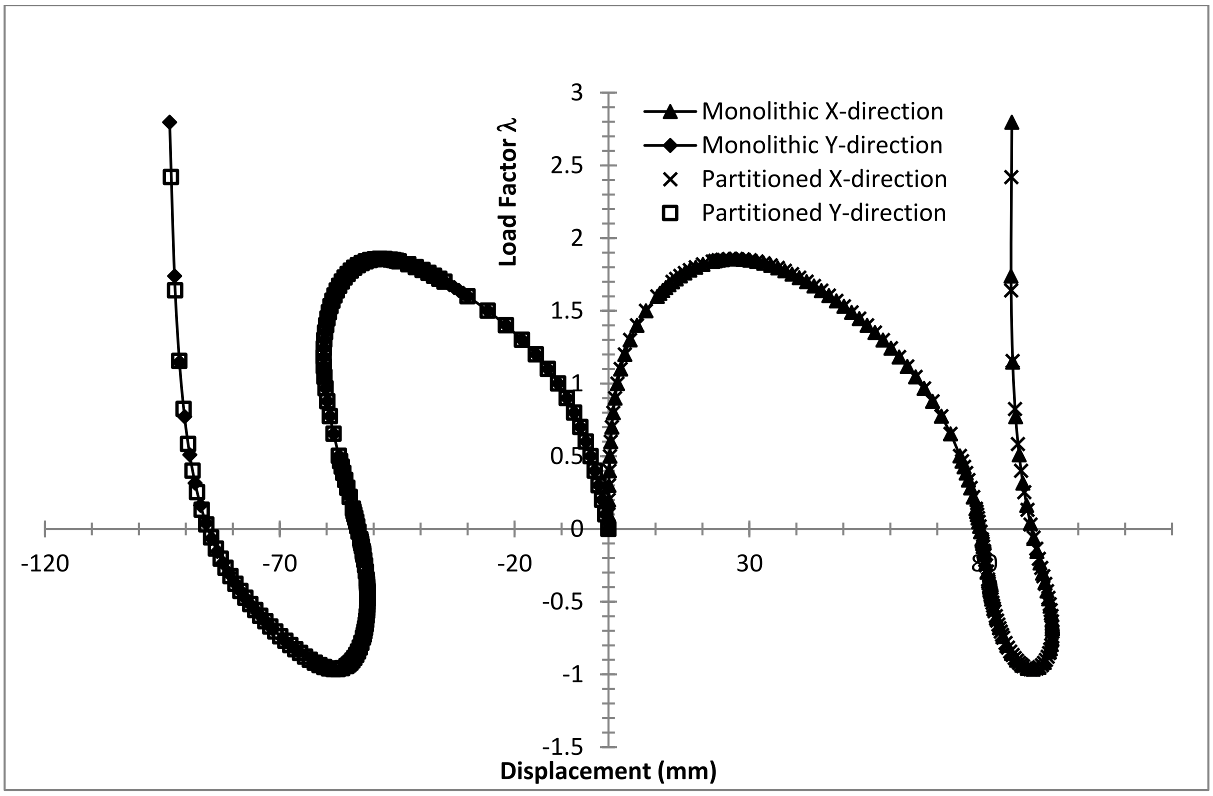
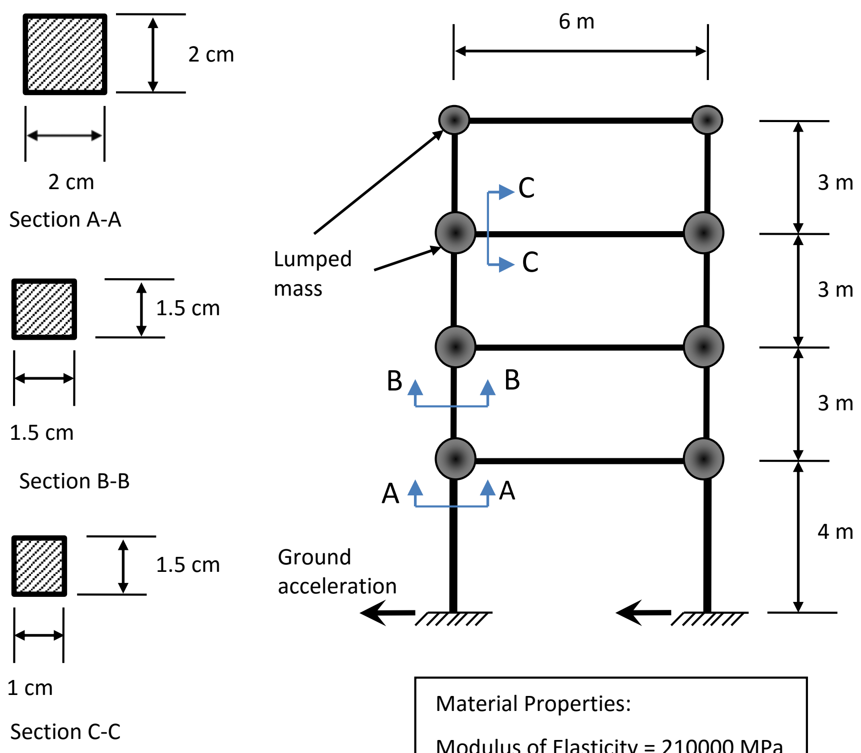


Figure 8: Comparison of displacement of node 3



Material Properties:

Modulus of Elasticity = 210000 MPa

Strength = 300 MPa

Figure 9: Four storey frame structure with lumped mass

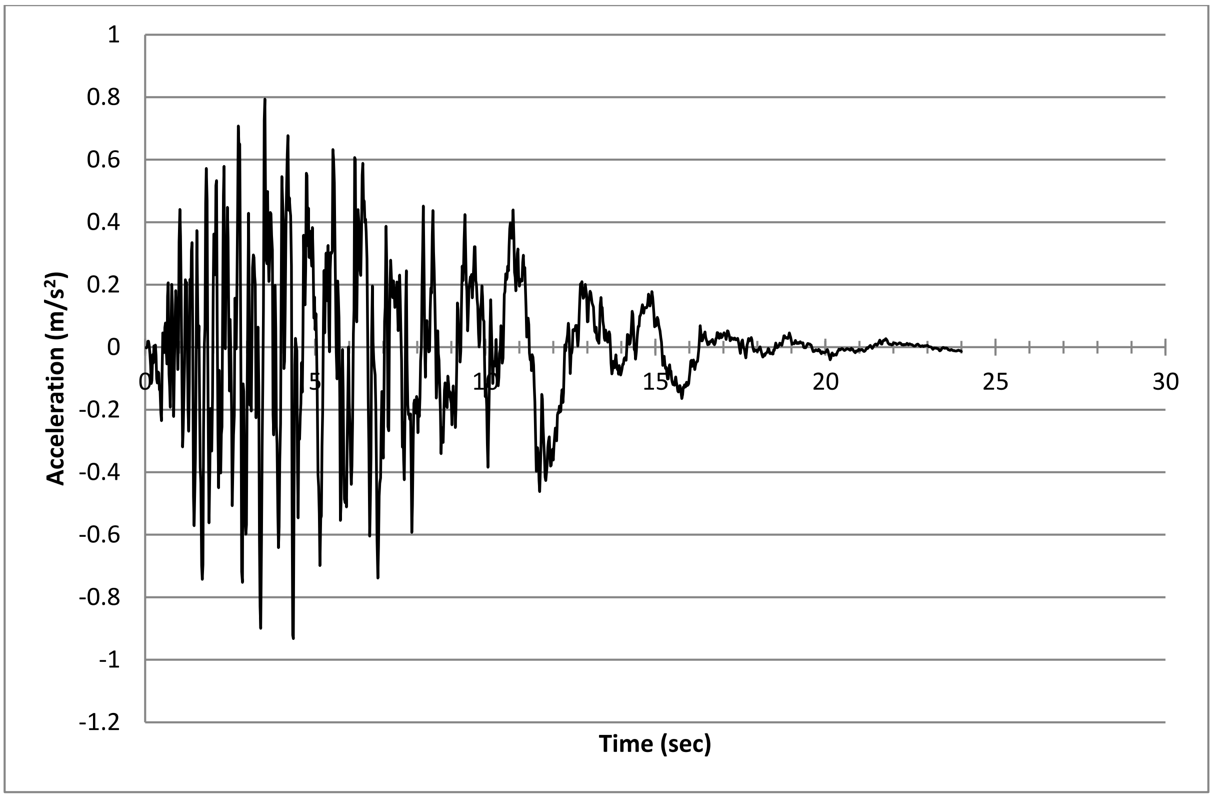


Figure 10: Ground acceleration record applied to 4 storey frame

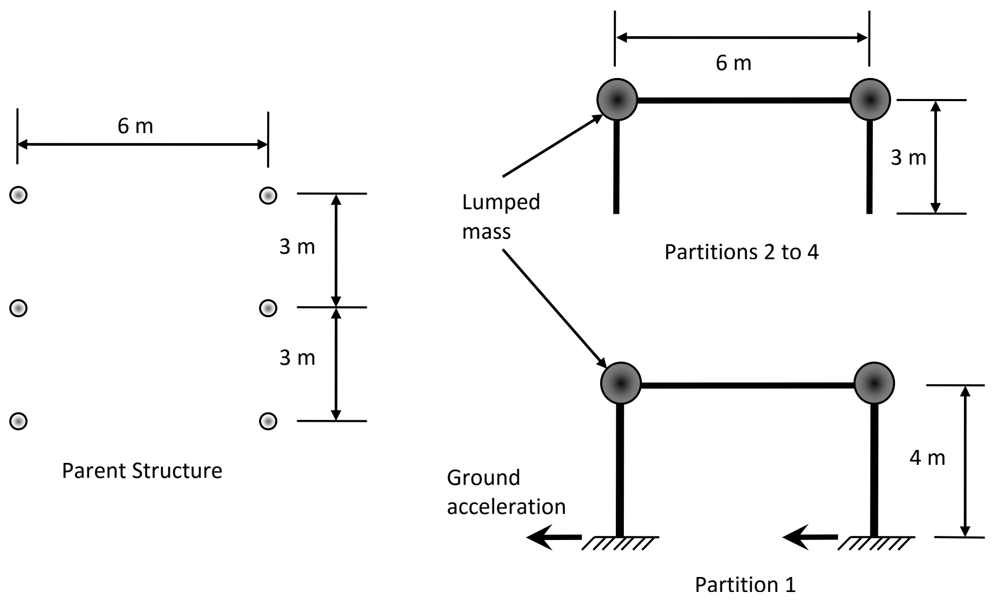


Figure 11: Partitioning of 4 storey frame for parallel analysis

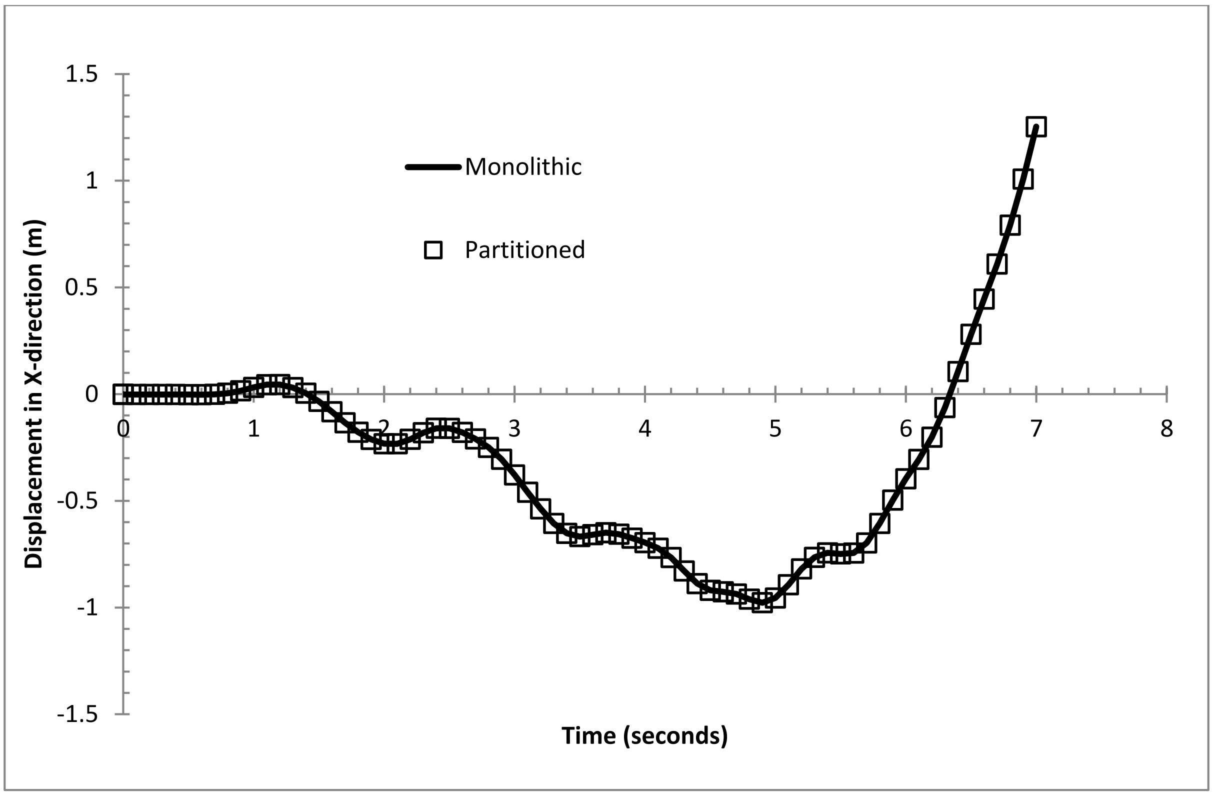


Figure 12: Comparison of displacement at the top

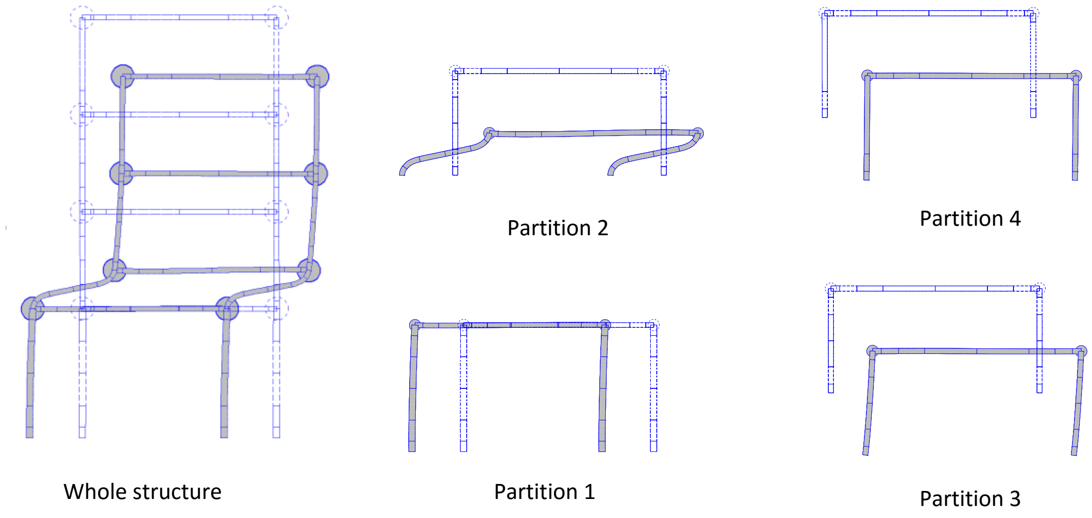


Figure 13: Comparison of deformed shapes

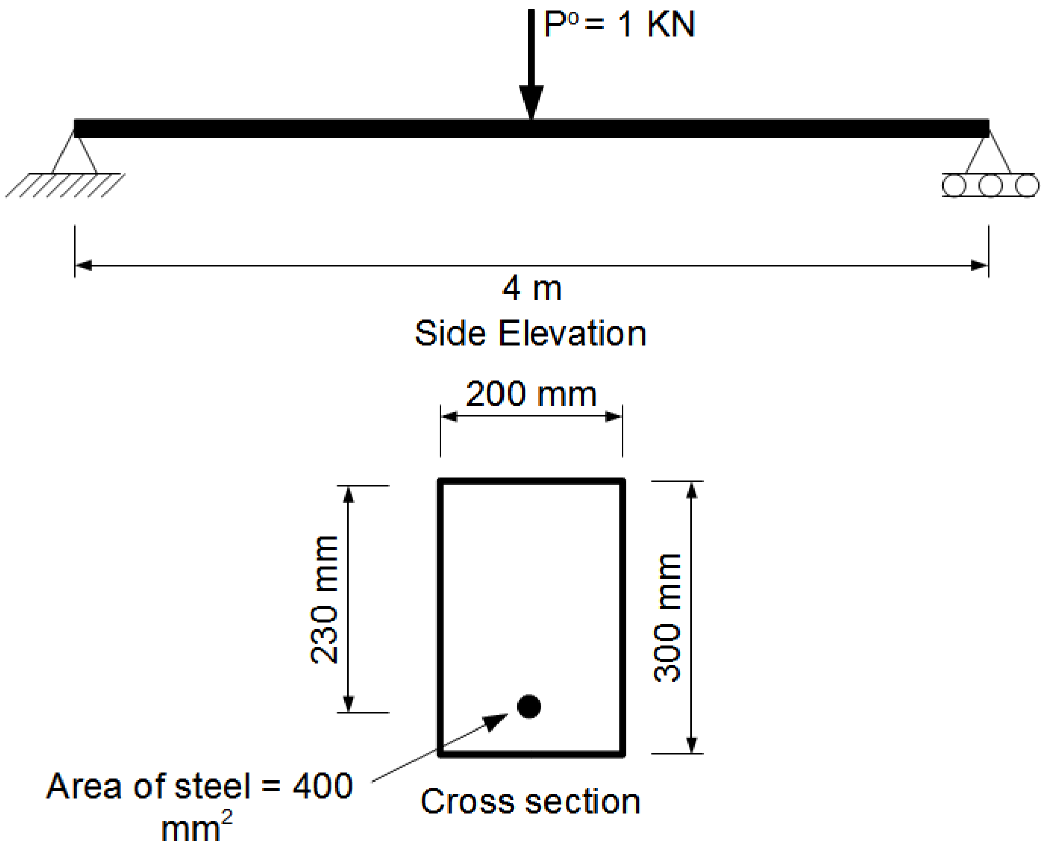


Figure 14: Reinforced concrete beam

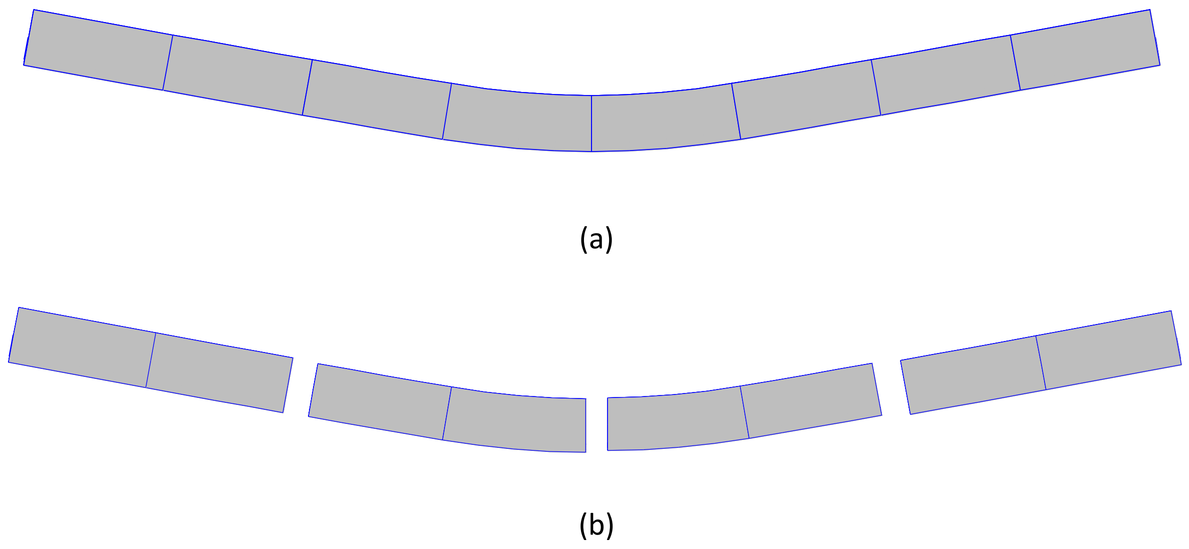


Figure 15: Deflected shapes for RC beam (a) whole beam (b) 4 partitions

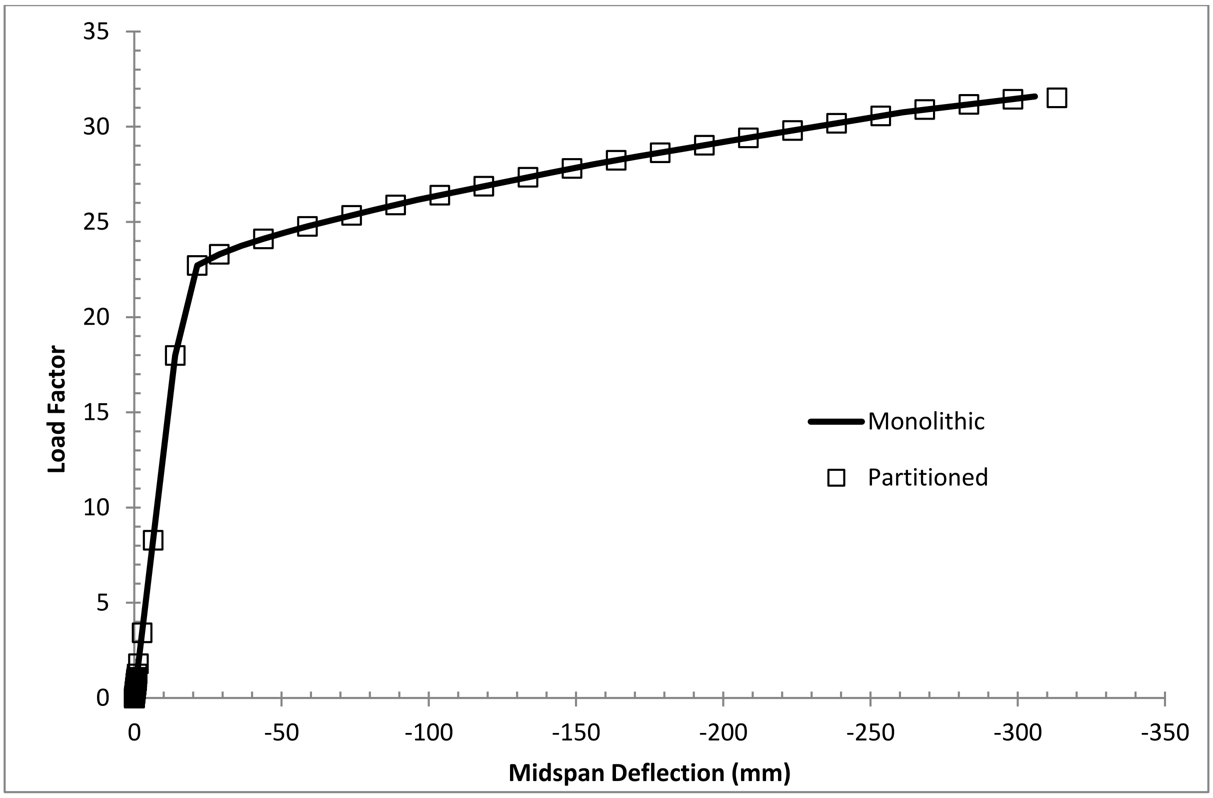


Figure 16: Load-deflection response of reinforced concrete beam

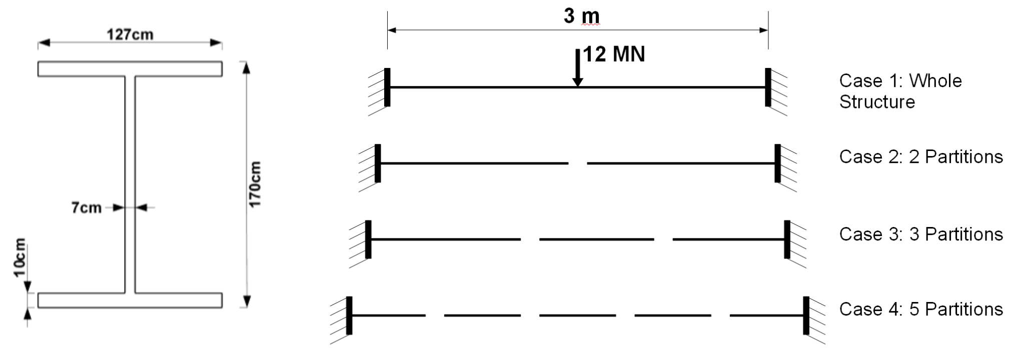


Figure 17: I-Beam geometric configuration and alternative partitions

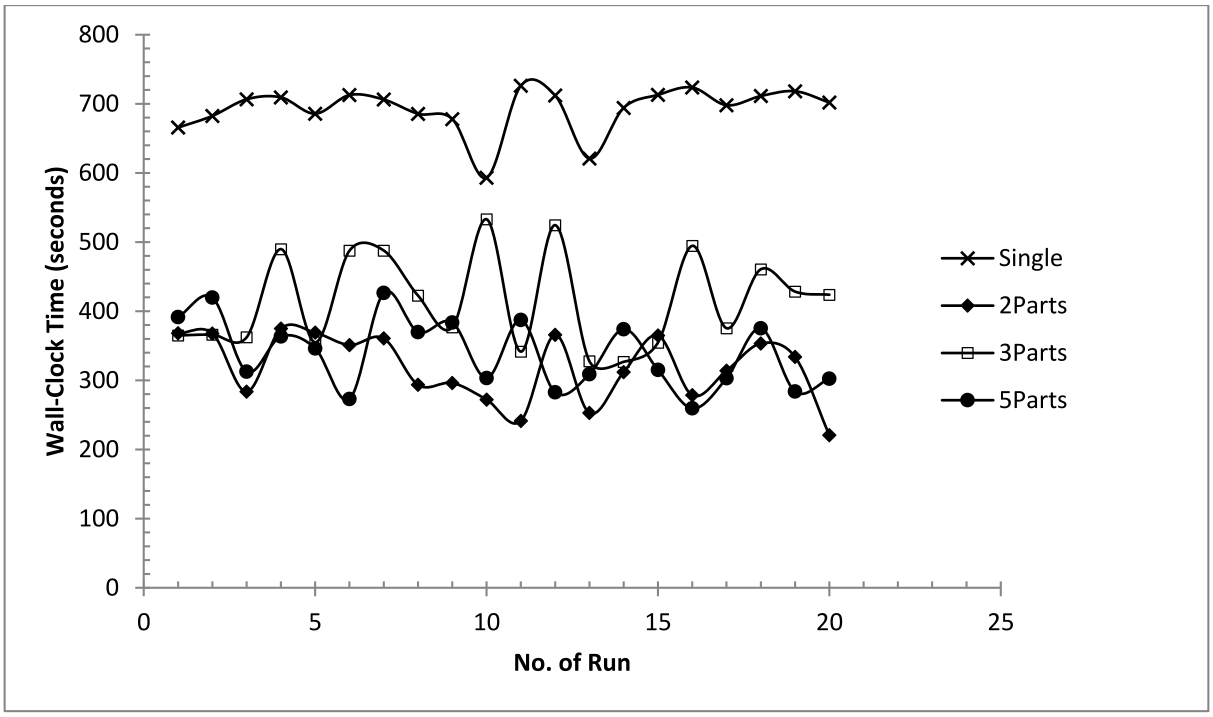


Figure 18: Comparison of wall clock times for I-beam

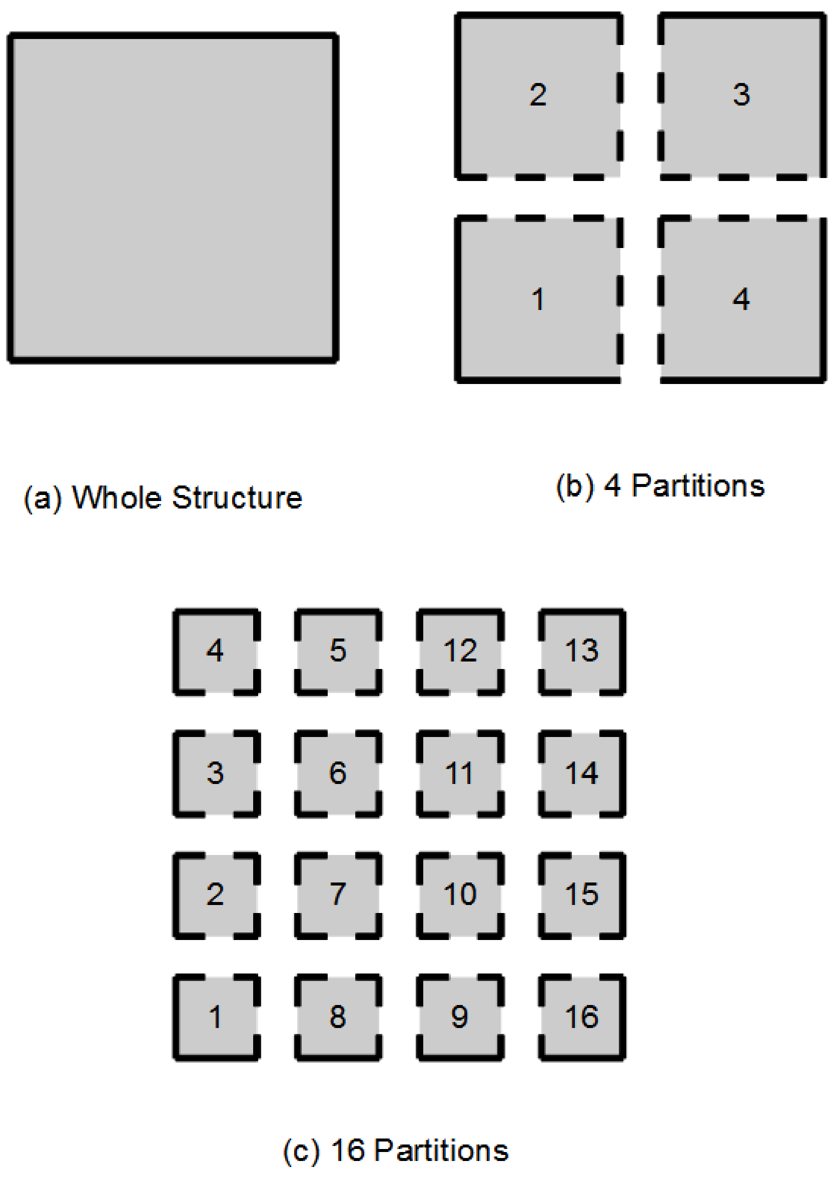


Figure 19: Slab and alternative partitioned models

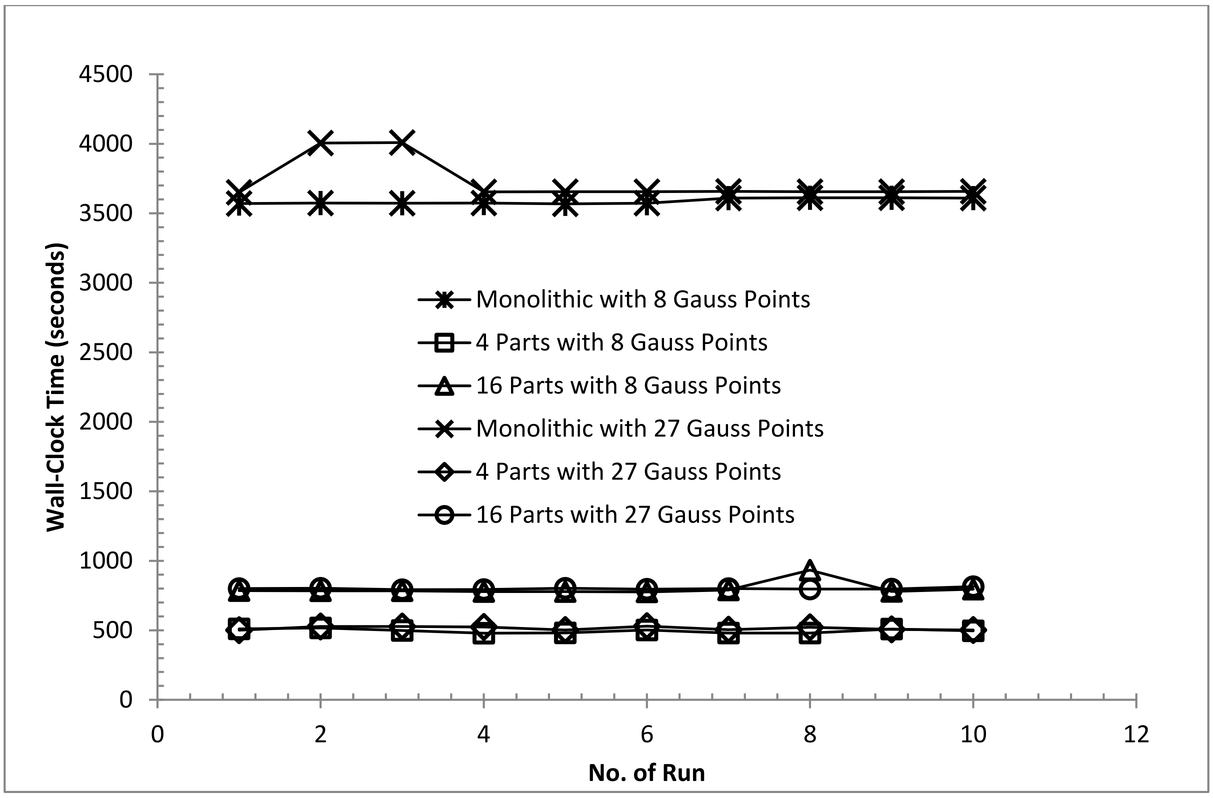


Figure 20: Wall clock times for 8 and 27 Gauss points

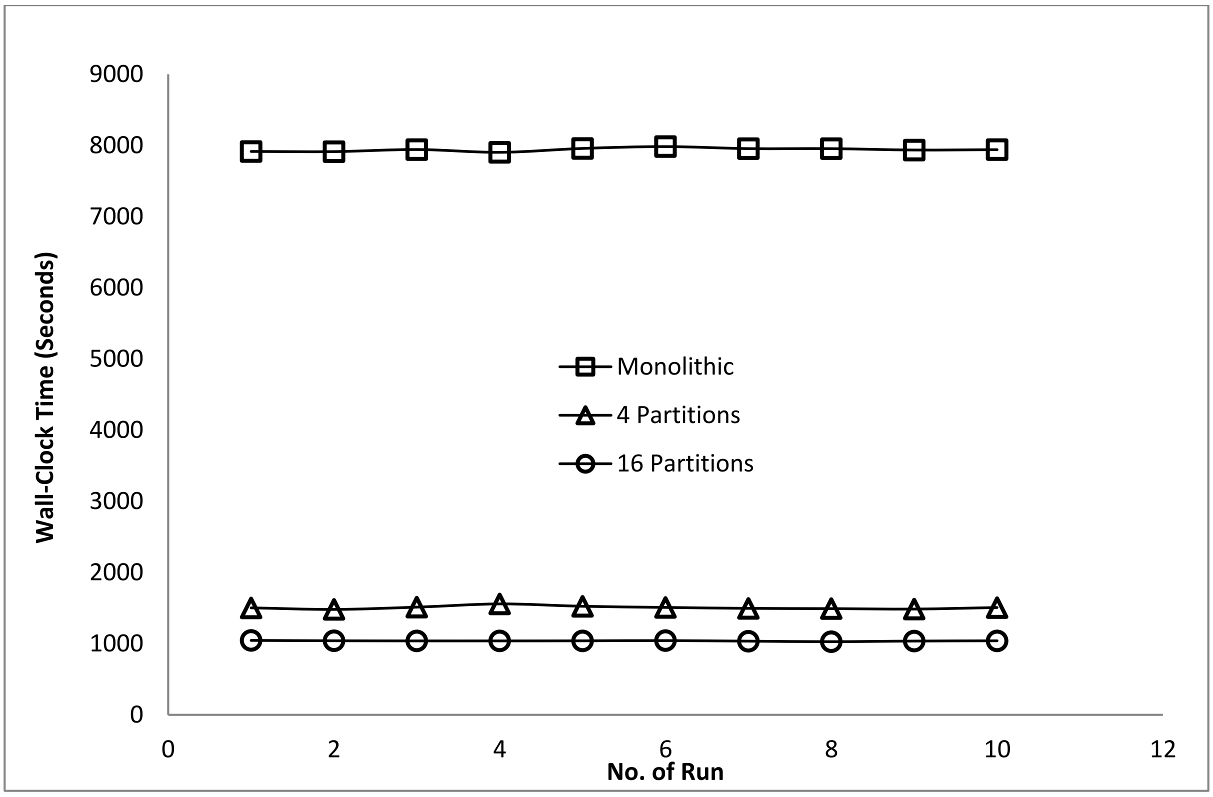


Figure 21: Wall clock times for 1000 Gauss points

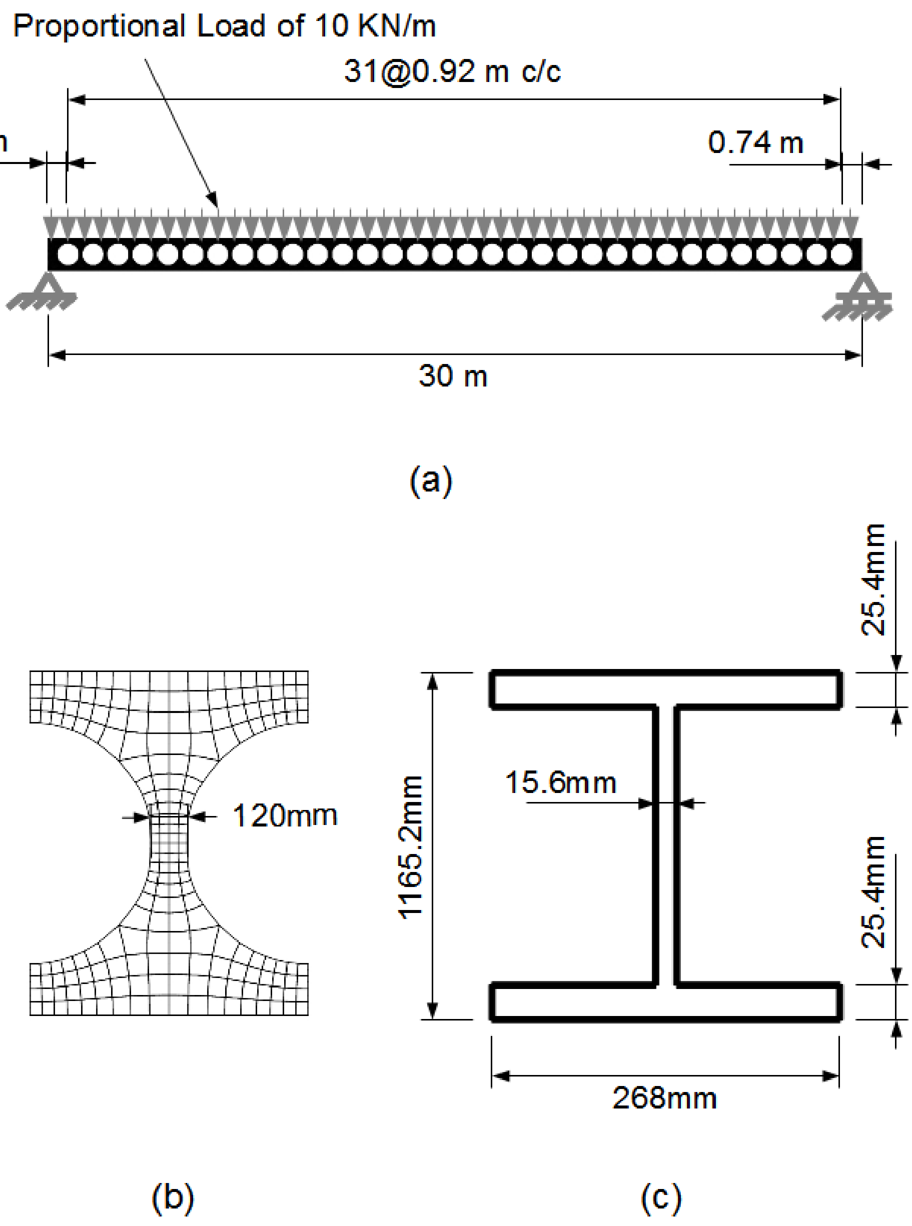


Figure 22: Cellular beam (a) side elevation of the entire beam (b) side elevation of a typical unit cell (c) beam cross section

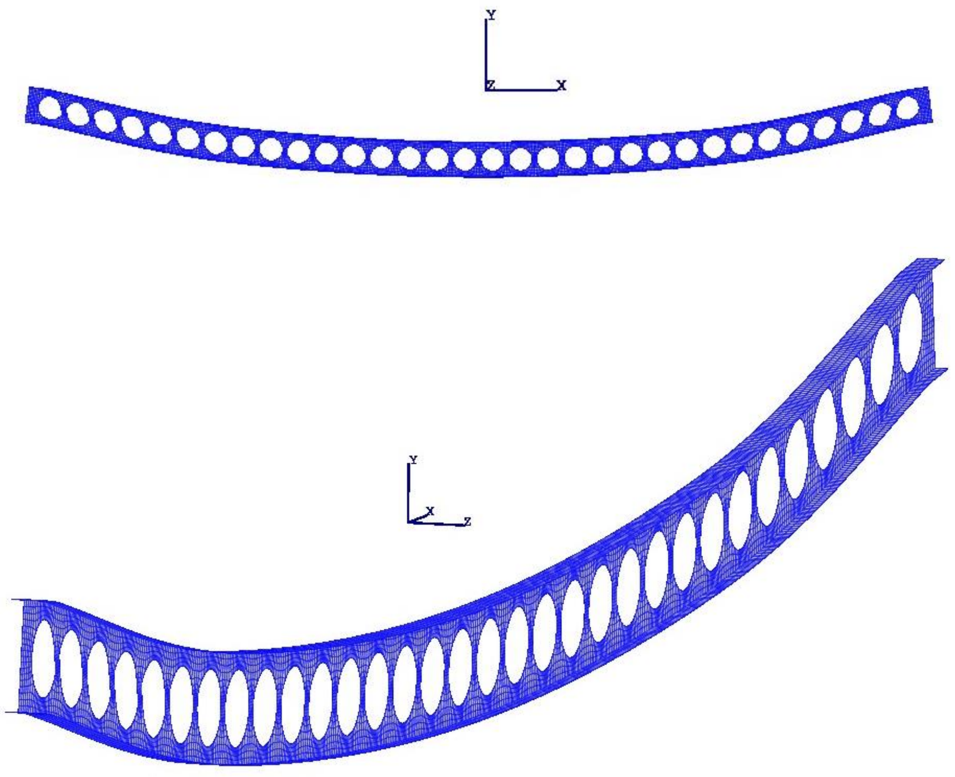


Figure 23: Deflected shape of the cellular beam at limit point (displacement scale = 5)

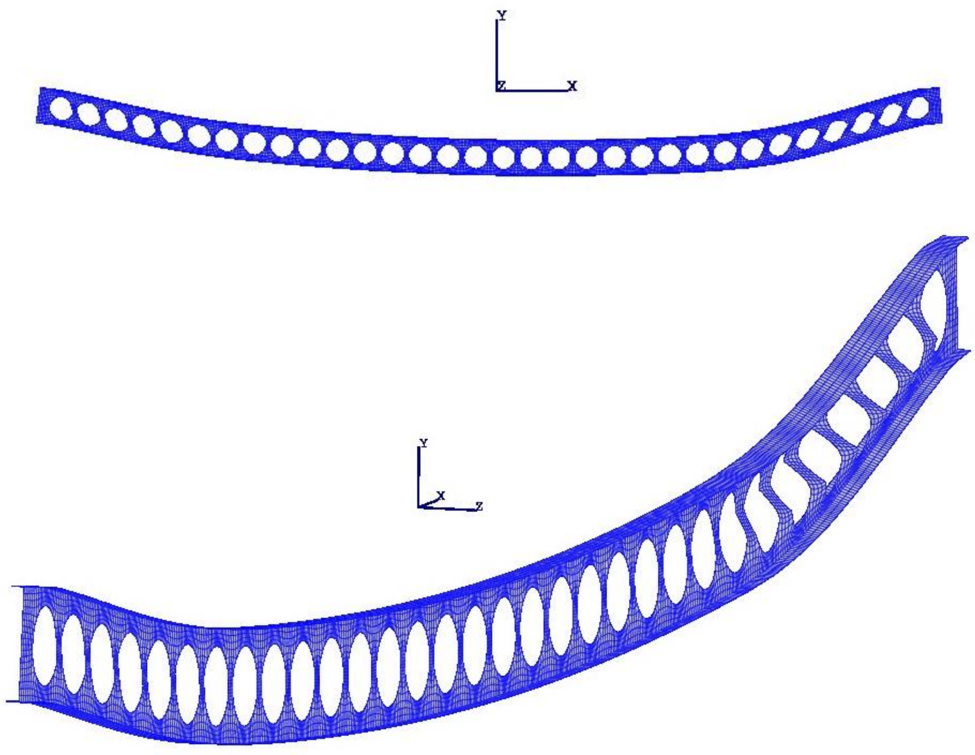
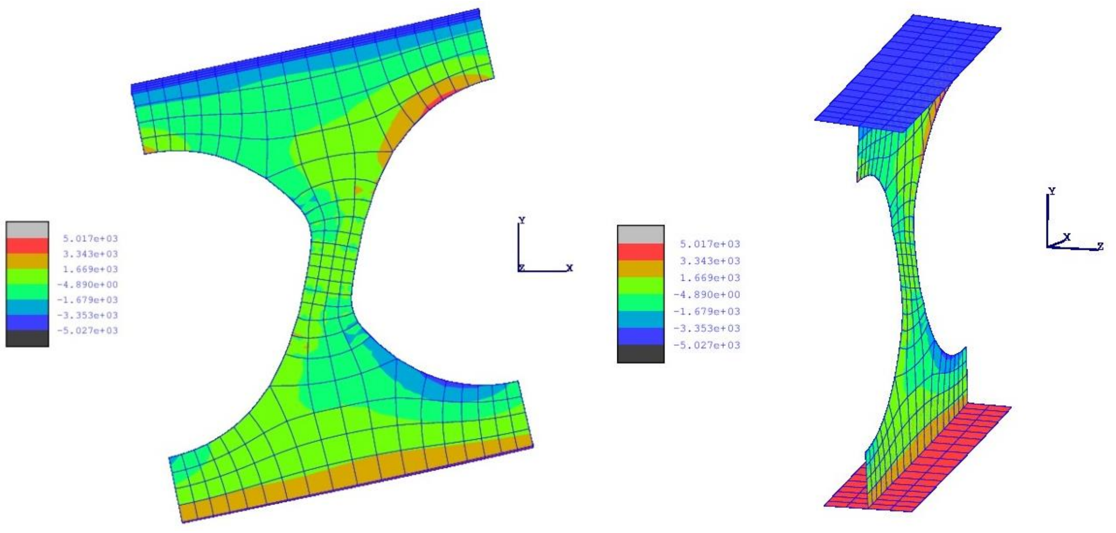
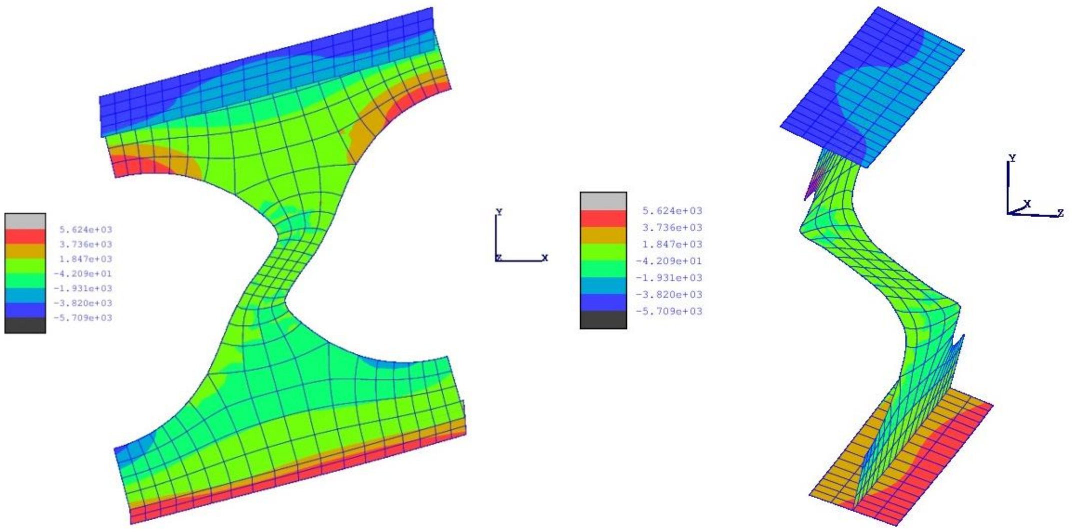


Figure 24: Final deflected shape of the cellular beam (displacement scale = 5)



(a) Limit point



(b) Final

Figure 25: Contours of normal traction in longitudinal direction for partition 29 (Units: N/m)

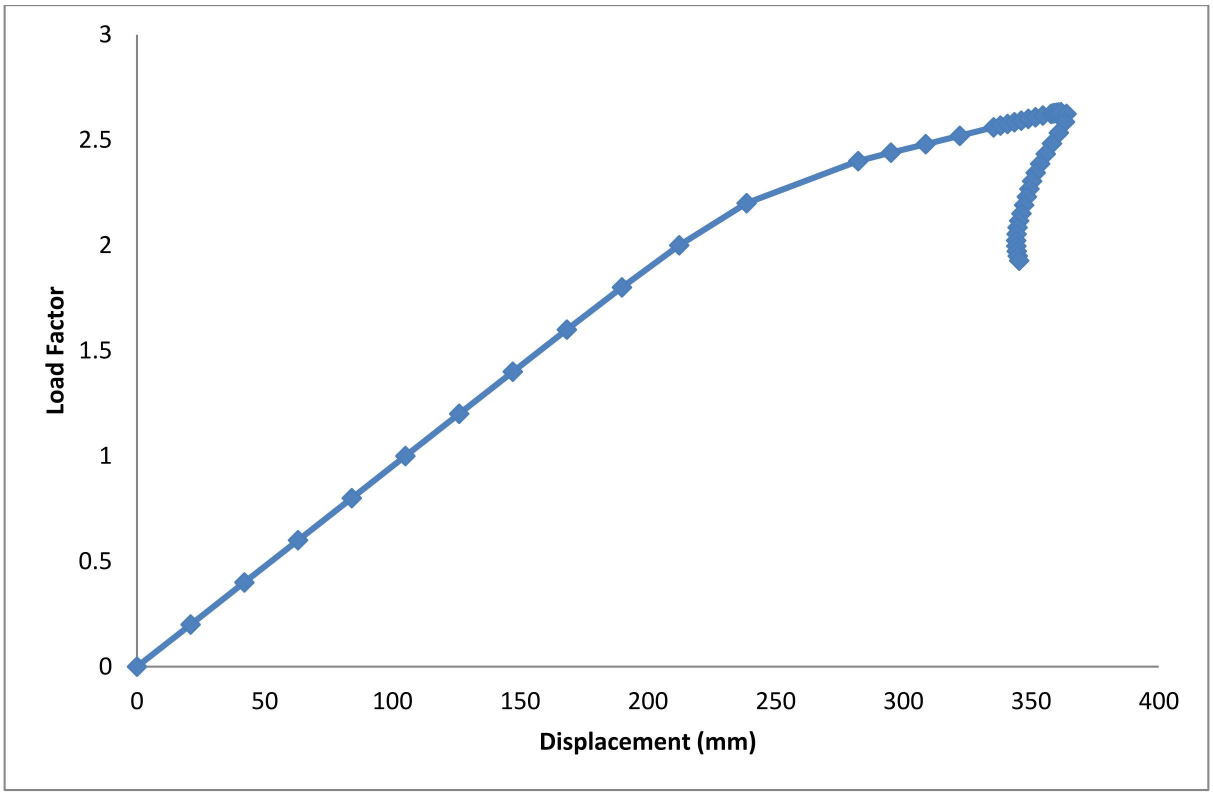


Figure 26: Displacement at mid-span in the cellular beam