

Mixed Dimensional Hierarchic Partitioned Analysis of Nonlinear Structural Systems

Gul Ahmed Jokhio

October 2012

A thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

Department of Civil and Environmental Engineering,
Imperial College London
London, SW7 2AZ

Abstract

The use of most accurate 3D modelling in Finite Element analysis is too computationally expensive to be practical. The alternate simplification of 1D elements may sometimes compromise the accuracy of results. However, the detailed modelling of the critical parts and approximate modelling of the non-critical parts of the structure is often sufficient.

A novel hierarchic domain partitioning approach has been developed in this study, which is being presented. The thesis begins with an introduction followed by a literature review. The domain partitioning approach is described next in which, parts of a structural system are removed and replaced by partition super elements. The removed parts are modelled separately with their partitioned boundary wrapped around by dual partition super elements. These partitions are analysed simultaneously using parallel computations. This domain partitioning approach increases the computational efficiency and allows the possibility of the use of differently dimensioned partitions.

Using the domain partitioning approach, the complex non-linear structural systems can be subjected to static time-history, proportional loading, and dynamic loading. For dynamic analysis, eigenvalue analysis is required. The eigenvalue problem for large matrices is itself computationally expensive; however, a new parallel implementation of eigenvalue analysis is developed here using the partition super elements.

In order to be able to use differently dimensioned partitions, a new dimensional coupling method has been developed. This is implemented with the help of a new master-slave element which has a single node as the master and all the nodes at the partitioned boundary as its slave nodes. This allows the possibility of using 3D brick elements inside a partition, whereas the other partitions at higher levels can use simplified 1D element models.

The domain partitioning approach has been further enhanced by making it hierarchical, where, the partitions are further subdivided by replacing their parts with partition super elements and the removed parts modelled at further lower levels of partitioning. The

hierarchical modelling is followed by a couple of case studies that demonstrate the applicability of the developed methods.

The thesis ends with discussion and with the conclusion that the mixed dimensional hierarchic partitioning methods have greatly increased the computational efficiency of the finite element analysis. Some directions for the future research related to this work have been suggested.

Acknowledgements

First and foremost I acknowledge the valuable guidance and immense help of my PhD advisor Prof. Bassam A. Izzuddin. He has been a great inspiration and throughout the duration of this work, he often times went out of his way to help and guide me. In addition, he provided great practical help with regards to the implementation of the developed methods.

I also acknowledge the great support provided by my wife Yasmeen Gul, my son Hazique Ushaque and my daughter Ashtar Sophia. They endured the hardships with great courage during the years of this work when I was practically non-existent for them. I also acknowledge the help and support provided by other members of the extended family.

The administrative support provided by Fionualla O. Donovan, Post Graduate Administrator, is duly acknowledged. In addition, I acknowledge the help and support provided by my colleagues and friends including: Dr. Lorenzo Macorini, Francesco, Ahmed Razin Zain-al-Abideen, Hamed Jahromi, Shirin Jowhari Moghadam, Miguel Parriera, Syed Kamran Shah, Mukesh Kumar, Dr. Arash Alfallah, and Imran Jokhio.

The research environment and support services provided by the Department of Civil and Environmental Engineering, Central Library, Graduate Schools of Engineering and Physical Sciences, HPC Services and Senior Common Room at the Imperial College London are duly acknowledged.

I acknowledge the Gwadar Development Authority, my employer, for allowing me to proceed for the higher education. The help and support of the former Director General Mr. Ahmed Bakhsh Lehri, and the incumbent Chief Engineer Dr. Sajjad Hussain are greatly acknowledged.

This research work would not have been possible without the financial help provided by the Higher Education Commission of Pakistan.

Dedication

This thesis is dedicated to the fond memories of my father Late Haji Yar Muhammad Jokhio and my brother Late Ahmed Ali Jokhio. Both these personalities have been a source of inspiration for me throughout my life.

Table of Contents

Abstract.....	2
Acknowledgements.....	4
Dedication	5
Table of Contents.....	6
List of Figures	13
List of Tables	21
Chapter 1 Introduction	23
1.1 Background.....	23
1.2 Challenges	24
1.3 Solution: Part Simplification, Partitioning and Parallelisation	25
1.4 Description of the Present Research.....	25
1.5 Objectives, Originality and Benefits	28
1.6 Scope	29
1.7 Thesis Outline.....	29
Chapter 2 Literature Review	31
2.1 Introduction.....	31
2.2 Analysis Methods	32
2.2.1 Finite Element Analysis	33

2.2.2	Nonlinear Static Analysis for Proportional Loading	36
2.2.3	Static/Dynamic Time-History Analysis	41
2.2.4	Iterative Solution Procedures	41
2.2.5	Frontal Method	44
2.2.6	Eigenvalue Analysis	45
2.3	Domain Decomposition.....	51
2.3.1	Staggered Approach.....	51
2.3.2	Iterative Coupling Methods	52
2.3.3	Lagrange Multiplier Techniques	54
2.3.4	Displacement Frame Methods.....	55
2.3.5	Multi-scale Modelling	56
2.3.6	Hierarchic Partitioning	58
2.4	Dimensional Coupling	59
2.5	Parallelisation Techniques.....	61
2.5.1	Computer Cluster Networks	61
2.5.2	Parallel Programming.....	64
2.5.3	Message Passing Interface (MPI) Library	65
2.5.4	MPI/OpenMP Hybrid Parallelisation.....	67
Chapter 3 Domain Decomposition		68

3.1	Introduction.....	68
3.2	Partitioning with Dual Super elements	69
3.2.1	Partitioning Method – Solution at Parent Level	71
3.2.2	Frontal Method – Solution at Child Level	74
3.2.3	Control and Convergence	77
3.2.4	Displacement Control	78
3.3	Parallelisation Efficiency with Partitioning Approach	79
3.3.1	4×4 Grid.....	80
3.3.2	2×8 Strip	86
3.3.3	Remarks on Efficient Partitioning	89
3.4	Implementation of Partitioning Method	90
3.4.1	Process 0: The Coordinator.....	91
3.4.2	Partition Process	91
3.4.3	Input Data Syntax.....	94
3.5	Verification	96
3.5.1	Lee’s Frame	96
3.5.2	Dynamic Analysis of Four-Storey Frame	99
3.5.3	Reinforced Concrete Beam	102
3.5.4	I-Beam Modelled with 3D Brick Elements	105

3.5.5	Square Slab	107
3.5.6	3D Frame.....	110
Chapter 4 Parallel Eigenvalue Analysis		114
4.1	Introduction.....	114
4.2	Monolithic Lanczos Algorithm.....	115
4.2.1	Lanczos Recursion	116
4.2.2	Convergence of Eigenvalues	118
4.2.3	Algorithm Steps.....	119
4.3	Partitioned Lanczos Algorithm	122
4.3.1	Initialisation.....	122
4.3.2	Lanczos Steps	124
4.3.3	Parallel Implementation of Lanczos Algorithm	126
4.4	Verification	129
4.4.1	Clamped Beam	129
4.4.2	23-storey Framed Structure.....	132
4.4.3	6-storey Plane Frame	135
4.4.4	I-beam with 3D Brick Elements.....	143
4.4.5	Space Frame made with 3D Brick Elements	145
Chapter 5 Dimensional Coupling		149

5.1	Introduction.....	149
5.2	Method Formulation	150
5.3	Inter-dimensional Transformations	153
5.3.1	Displacement Transformation	153
5.3.2	Forces Transformation	156
5.3.3	Tangent Stiffness Transformation	157
5.4	Implementation.....	158
5.5	Verification Examples.....	161
5.5.1	Cantilever Beam.....	161
5.5.2	Frame with Masonry Infill	170
5.5.3	3D Space Frame	172
Chapter 6 Hierarchical Partitioning		176
6.1	Introduction.....	176
6.2	Hierarchical Partitioning Approach	178
6.2.1	Method Formulation.....	181
6.3	Parallelisation Efficiency with Hierarchic Partitioning	183
6.4	Implementation.....	186
6.4.1	Intermediate Processes	187
6.4.2	Hierarchic Communicators	189

6.4.3	Input Data Syntax.....	195
6.5	Verification	195
6.5.1	Dynamic Analysis of Four-Storey Frame	195
6.5.2	Square Slab	197
Chapter 7 Case Studies		200
7.1	Introduction.....	200
7.2	Cellular Beam	201
7.3	Masonry Structures.....	207
7.3.1	Frame with Masonry Infill	207
7.3.2	Unreinforced Perforated Masonry Wall	215
7.4	Cable Stayed Bridge.....	217
7.4.1	Structural Model	218
7.4.2	Element Types.....	219
7.4.3	Partitioning Scheme	222
7.4.4	Analysis and Results.....	225
Chapter 8 Summary, Conclusions and Future Research.....		228
8.1	Summary	228
8.2	Conclusions.....	230
8.2.1	Reduction in Computational Time	231

8.2.2	Modelling of Complex and Large Scale Structures	232
8.2.3	Ease of Modelling.....	232
8.3	Recommendations for Future Research	233
8.3.1	Extended Dimensional Coupling Schemes.....	233
8.3.2	Multi-physics Modelling.....	233
8.3.3	MPI/OpenMP Hybrid Parallelisation.....	234
8.3.4	Partition Scheduling.....	234
8.3.5	Heterogeneous Solution Procedures.....	234
	References	236
	Appendix A.....	245
	Appendix B.....	250
	Appendix C.....	257

List of Figures

Figure 1-1: 1D and 3D elements used for various parts of a structure	26
Figure 1-2: Communication between the Control Process and Partition Processes.....	27
Figure 1-3: An example child partition defined by own elements, loads, restraints and partition boundaries	27
Figure 2-1: Equilibrium path (load-deflection diagram)	37
Figure 2-2: Newton-Raphson method	42
Figure 2-3: Modified Newton-Raphson method.....	42
Figure 2-4: Comparison of two assembly configurations for a two storey frame [Izzuddin, 1991].....	45
Figure 2-5: Staggered approach for a two domain system	52
Figure 2-6: Schematic diagram for iterative coupling	53
Figure 2-7: Two-level strategy for modelling a URM perforated wall [Izzuddin and Macorini, 2011]	57
Figure 2-8: Bus Network	62
Figure 2-9: Switching Network.....	62
Figure 2-10: 2D Mesh Network.....	63
Figure 2-11: Torus Network (a) 1D Torus (b) 2D Torus.....	63
Figure 2-12: Fat Tree Network.....	63
Figure 3-1: An illustrative structural domain	70

Figure 3-2: Partitioned structure (a) parent structure (b) child partitions.....	72
Figure 3-3: A mesh of 16 2D elements.....	80
Figure 3-4: Problem decomposed into 2 partitions of 8 elements each	82
Figure 3-5: Problem decomposed into 4 partitions of 4 elements each	84
Figure 3-6: Problem decomposed into 16 partitions of 1 element each	86
Figure 3-7: The mesh of 16 2D elements in strip shape	87
Figure 3-8: The strip partitioned in two.....	87
Figure 3-9: Flowchart of the coordinator process (PPs stands for partition processes)	92
Figure 3-10: Flowchart of a partition process.....	93
Figure 3-11 Lee's frame	97
Figure 3-12: Lee's frame after partitioning (a) parent structure, (b-d) partitions	98
Figure 3-13: Comparison of displacement of node 3	98
Figure 3-14: Four-storey frame structure with lumped mass	99
Figure 3-15: Ground acceleration record applied to four-storey frame	100
Figure 3-16: Partitioning of four-storey frame for parallel analysis.....	101
Figure 3-17: Comparison of displacement at the top.....	101
Figure 3-18: Comparison of deformed shapes	102
Figure 3-19: Reinforced concrete beam	103
Figure 3-20: Deflected shapes for RC beam (a) whole beam (b) 4 partitions	104

Figure 3-21: Load-deflection response of reinforced concrete beam.....	104
Figure 3-22: I-beam geometric configuration and alternative partitions	105
Figure 3-23: Comparison of wall-clock times for I-beam.....	107
Figure 3-24: Slab and alternative partitioned models	107
Figure 3-25: Wall-clock times for 8 and 27 Gauss points	109
Figure 3-26: Wall-clock times for 1000 Gauss points	110
Figure 3-27: Partition objects assembled to form plane/space frame.....	111
Figure 3-28: Modular approach of forming structure from partition objects.....	111
Figure 3-29: Final assembled 3D frame and deflected shape	112
Figure 4-1: Flow chart for the monolithic Lanczos Algorithm	121
Figure 4-2: Flow chart of the Lanczos algorithm in the coordinator process.....	127
Figure 4-3: Flow chart for the Lanczos algorithm in a partition process.....	128
Figure 4-4: Clamped beam model.....	130
Figure 4-5: Vibration mode shapes for the clamped beam.....	131
Figure 4-6: Plan and elevation of multi-storey frame.....	132
Figure 4-7: Comparison of mode shapes for mode 7 (a) full re-orthogonalisation (b) selective re-orthogonalisation	133
Figure 4-8: Comparison of mode shapes for mode 100 (a) full re-orthogonalisation (b) selective re-orthogonalisation	133
Figure 4-9: 6-storey frame	135

Figure 4-10: Cross-sections for different members of the 6-storey frame	136
Figure 4-11: Modes 3, 6, and 9 of 6-storey frame	137
Figure 4-12: Partitioning of 6-storey frame	139
Figure 4-13: Mode shapes of different partitions for mode no. 3	140
Figure 4-14: Mode shapes of different partitions for mode no. 6	141
Figure 4-15: Mode shapes of different partitions for mode no. 9	142
Figure 4-16: Comparison of monolithic and partitioned shapes for mode no. 1	144
Figure 4-17: Comparison of monolithic and partitioned shapes for mode no. 2	144
Figure 4-18: Comparison of monolithic and partitioned shapes for mode no. 3	144
Figure 4-19: Comparison of monolithic and partitioned shapes for mode no. 4	145
Figure 4-20: Mode shapes for frame: mode 2	146
Figure 4-21: Mode shapes for frame: mode 4	147
Figure 4-22: Comparison of wall-clock time	148
Figure 5-1: Master-slave (1D-3D) coupling element.....	150
Figure 5-2: An example child partition with 4 dimensional coupling elements	152
Figure 5-3: Global freedoms of 1D master node	154
Figure 5-4: Translations applied to slave nodes	155
Figure 5-5: Rotations applied to a slave node	155
Figure 5-6: Partitioned model of 3D beam with 2 cross-sections on partitioned boundary	159

Figure 5-7: Dimensional coupling interface elements used for a beam.....	160
Figure 5-8: Cantilever beam (a) cross-section (b) span and loading	163
Figure 5-9: Deflected shapes for Cases 1 and 2	164
Figure 5-10: Deflected shapes of Cases 3 to 6	164
Figure 5-11: Load-deflection of cantilever for different cases	166
Figure 5-12: Load-deflection comparison at Stage 1	167
Figure 5-13: Load-deflection comparison at Stage 2	167
Figure 5-14: Load-deflection comparison at Stage 3	168
Figure 5-15: Stress contours in X-Direction	169
Figure 5-16: Frame with infill masonry	171
Figure 5-17: Partitioned boundary of the masonry infill	171
Figure 5-18: Deflected shape and axial stress contours of the frame	172
Figure 5-19: Parent structure for mixed-dimensioned model of 3D space frame	173
Figure 5-20: Final deflected shape of the frame	175
Figure 5-21: Comparison of longitudinal stress contours at the column base.....	175
Figure 6-1: Hierarchic partitioning schematic diagram	177
Figure 6-2: Example hierarchical partitioning model and communication network.....	178
Figure 6-3: A general structural domain with 32 elements	179
Figure 6-4: Four levels of hierarchical partitioning.....	180

Figure 6-5: Parent structure for one level partitioning	184
Figure 6-6: Flow chart for the partitions at intermediate levels	189
Figure 6-7: Ranking schemes for hierarchic modelling.....	191
Figure 6-8: Communicator generation at one level.....	192
Figure 6-9: Flow chart of the communicator generating algorithm.....	194
Figure 6-10: Hierarchic partitioning of four-storey frame.....	196
Figure 6-11: Comparison of the top displacement for the four-storey frame	197
Figure 6-12: Hierarchical partitioning model for square slab.....	198
Figure 6-13: Comparison of wall clock time for hierarchic partition of the slab model.....	199
Figure 7-1: Side elevation of the cellular beam	202
Figure 7-2: Side elevation of a unit cell and cross-section of the cellular beam.....	202
Figure 7-3: Deflected shape of the cellular beam at limit point (Displacement scale = 5) ...	203
Figure 7-4: Deflected shape of the cellular beam at limit point (Displacement scale = 5) ...	203
Figure 7-5: Final deflected shape of the cellular beam (Displacement scale = 5)	204
Figure 7-6: Final deflected shape of the cellular beam (Displacement scale = 5)	204
Figure 7-7: Contours for normal traction in longitudinal direction at limit point for partitions 29 and 30 (Units: N/m as thickness is included in stress resultant)	204
Figure 7-8: Contours for normal traction in longitudinal direction at limit point for partitions 29 and 30 (Units: N/m as thickness is included in stress resultant)	205
Figure 7-9: Contours for normal traction in longitudinal direction at final stage for partitions 29 and 30 (Units: N/m as thickness is included in stress resultant)	205

Figure 7-10: Contours for normal traction in longitudinal direction at final stage for partitions 29 and 30 (Units: N/m as thickness is included in stress resultant)	206
Figure 7-11: Displacement at mid-span in the cellular beam.....	207
Figure 7-12: Frame with masonry infill	208
Figure 7-13: Cross-section of steel members and brick dimensions	208
Figure 7-14: Model for a single brick	209
Figure 7-15: Partitioning of the masonry infill.....	210
Figure 7-16: Intermediate partition (level 2) with dimensional coupling	211
Figure 7-17: Root partition (level 1).....	212
Figure 7-18: Deflected shape at 0.7 mm lateral drift (displacement scale = 100)	214
Figure 7-19: Contours showing plastic work at the interfaces	215
Figure 7-20: Mesh and partitions scheme used for representing a full-scale URM perforated wall [Izzuddin & Macorini, 2011]	216
Figure 7-21: (a) Crack development in full-scale URM perforated wall, (b) maximum equivalent stresses in solid elements [Izzuddin & Macorini, 2011]	217
Figure 7-22: Suspension bridge.....	218
Figure 7-23: Deck slab for one of the side spans	219
Figure 7-24: Element types for 'csl4' (I) left edge rib (II) cover (III) central rib (IV) right edge rib [Izzuddin <i>et al.</i> , 2004]	220
Figure 7-25: Local freedoms for element 'csl4' [Izzuddin <i>et al.</i> , 2004]	221
Figure 7-26 Stiffness parameters and local forces element type 'lnks' [Izzuddin, 2011].....	222

Figure 7-27: Root partition for the bridge	223
Figure 7-28: A typical level 2 partition for bridge deck and cables	224
Figure 7-29: Level 3 slab partitions	225
Figure 7-30: Deflected shape of the bridge	225
Figure 7-31: Deflected shape of the deck beams with contours showing axial force.....	226
Figure 7-32: Deflected shape for earthquake in transverse direction	227

List of Tables

Table 3-1: Reduction operations for a 4×4 grid using frontal method	81
Table 3-2: Reduction operations required for 2 partitions	83
Table 3-3: Reduction operations required for 4 partitions	84
Table 3-4: Reduction operations required for 16 partitions	85
Table 3-5: Summary of speed up gained	86
Table 3-6: Reduction operations for the strip of 16 2D elements.....	88
Table 3-7: Reduction operations required for the strip in 2 partitions.....	89
Table 3-8: Material properties used for reinforced concrete beam	103
Table 3-9: Comparison of equivalent DoFs.....	106
Table 3-10: Comparison of load steps performed in 72 hours.....	112
Table 4-1: Comparison of results for the clamped beam	130
Table 4-2: Natural frequencies of 23-storey framed structure	134
Table 4-3: Natural frequencies (cycles per second) for 6-storey frame.....	138
Table 4-4: Eigenvalues for I-beam modelled with 3D brick elements.....	143
Table 4-5: Natural frequencies for space frame	146
Table 5-1: Details of analysis cases for cantilever beam example	162
Table 5-2: Comparison of wall-clock times for cantilever beam	165
Table 5-3: Material properties for frame with masonry infill.....	171

Table 5-4: Computational efficiency of dimensional coupling compared to other methods	174
Table 6-1: Reduction operations for the single level partitioning	185
Table 6-2: Reduction operations for hierarchical partitioning	185
Table 7-1: Material properties for the mortar and brick interface elements	213

Introduction

This chapter presents the description of the current study, its objectives, significance, and scope.

1.1 Background

“Structural Engineering is the Art of moulding materials we do not wholly understand into shapes we cannot precisely analyse, so as to withstand forces we cannot really assess, in such a way that the community at large has no reason to suspect the extent of our ignorance” [Brown, 1967; Schmidt, 2009].

As the human society develops, it requires the construction of larger, stronger and more complex structures. The advent of computers and the development of computational methods have greatly aided engineers in their work in this regard. The most widely used tool for the analysis of structures is the Finite Element Method, which in effect is a mathematical approximation of the physical system.

The Finite Element Method involves many levels of simplification including material behaviour, mathematical modelling, and uncertainty of the applied load. The material model is formulated in light of the experiments carried out in order to determine its mechanical properties. The model, therefore, is an approximation that best represents the experimental data to start with, which often times is further simplified by approximating it with a linear, bilinear or trilinear curve.

The mathematical model representing the finite element is also an approximate formulation, which results in more accuracy with the use of higher order shape functions. In addition, it is not possible to know exactly what actions the structure would be subjected throughout its serviceable life. In this regard, predictions are made from analysing the historic data for example the prevailing use of different types of buildings in a particular society and recorded environmental events like windstorms, earthquakes etc.

These simplifications, while making it possible to carry out the analysis in the first place, are associated with inaccuracies that are in direct proportion to the extent of simplification used. When it comes to large and complex structural systems subjected to extreme loading, the details missed by simplified modelling might become significant thus compromising the usefulness of the undertaken computations.

1.2 Challenges

There are quite a few threats facing the construction industry at present. The biggest and most important challenge is the global environmental degradation. Another challenge, which is not very far behind, is the dwindling global economy. In addition, the accidental and intentional damage to structures from blasts, fires etc., is also a major concern in the modern world along with the effects of natural disasters such as earthquakes and hurricanes.

All these challenges demand the ability to design optimal structures. Ideally, the structures should use the least amount of material for the sake of environment and economy, but should be able to withstand the actions of accidental, human-created, and natural disasters. To achieve this, it is important that the response of the structures under specific actions can be predicted as accurately as possible.

The most accurate modelling approach is based on the use of 3D nonlinear finite elements, where, for example, the bond-slip characteristics between the steel reinforcement and the concrete are explicitly captured. This approach, however, is prohibitively expensive and is typically inapplicable to modelling the overall structural response, where the nonlinear analysis of relatively small structures can require several days of computing time. Moreover, this approach invariably suffers from a memory bottleneck, which prevents its application to realistic structural modelling problems. The alternative approach applicable to framed structures and based on 1D finite elements offers superior computational efficiency over 3D elements, but it suffers from inaccuracy in irregular regions of a frame, such as within joints. The 1D approach also requires an implicit treatment of the bond-slip response between reinforcement and concrete in reinforced concrete structures, which is often difficult to establish.

1.3 Solution: Part Simplification, Partitioning and Parallelisation

For all practical purposes, full 3D models of structures are not required. There are only certain parts of a structure that are critical. The joints and connections are such an example. As another example, while performing the earthquake analysis of a structure, the engineer may be more interested in the behaviour of particular columns in order to make the structures life safe or to undertake progressive collapse assessment. It would be very useful to model a structure using 3D elements for parts of interest and simplified 1D elements for the rest. This part-simplification, using elements of different dimensions for different parts of the structure, is termed as ‘mixed dimensional modelling’.

Another way to speed up the computations for large and complex systems is to divide the domain into different partitions and analyse them simultaneously using parallel processing. The partitioning and parallelisation would facilitate the implementation of part-simplification by mixed dimensional coupling.

1.4 Description of the Present Research

The present research is concerned with developing a new method for modelling structures under extreme loading, benefitting from the accuracy of 3D finite elements and from the computational efficiency of 1D and 2D elements through the use of mixed dimensional hierarchic partitioned analysis. Towards this objective, the structure is partitioned into several substructures: one substructure having the control of the analysis is termed as the *parent structure*, and the remaining substructures are called the *child partitions*. The parent structure may consist of the partitioned boundary nodes only or it may have some reduced dimensional elements, whereas the child partitions can be modelled using elements of higher dimensions (Figure 1-1) representing the parts of the structure that are either irregular or require detailed modelling to account for such effects as bond-slip etc.

The newly developed mixed dimensional hierarchic modelling has been implemented into the nonlinear structural analysis program ADAPTIC [Izzuddin, 1991], developed at Imperial College London. In this respect, multiple instances of ADAPTIC are run as parallel processes representing the parent structure and the child partitions (Figure 1-2), where the parent

process controls and communicates with the various child processes. The *control process* presents only a minor modification to a standard ADAPTIC nonlinear analysis procedure, where a few commands are embedded to control and communicate with the child processes. On the other hand, *child processes* operate only in a command mode, awaiting and acting upon control commands from the parent process. While the parent process is responsible for ensuring compatibility and equilibrium at the partition boundary and throughout the parent structure, the child processes are responsible for ensuring compatibility and equilibrium within the respective partitions. In this work, communication between the processes is undertaken via the Message Passing Interface (MPI) library subroutines, where the use of a distributed memory model for processes is scalable and avoids the memory bottleneck in the analysis of realistic large-scale problems.

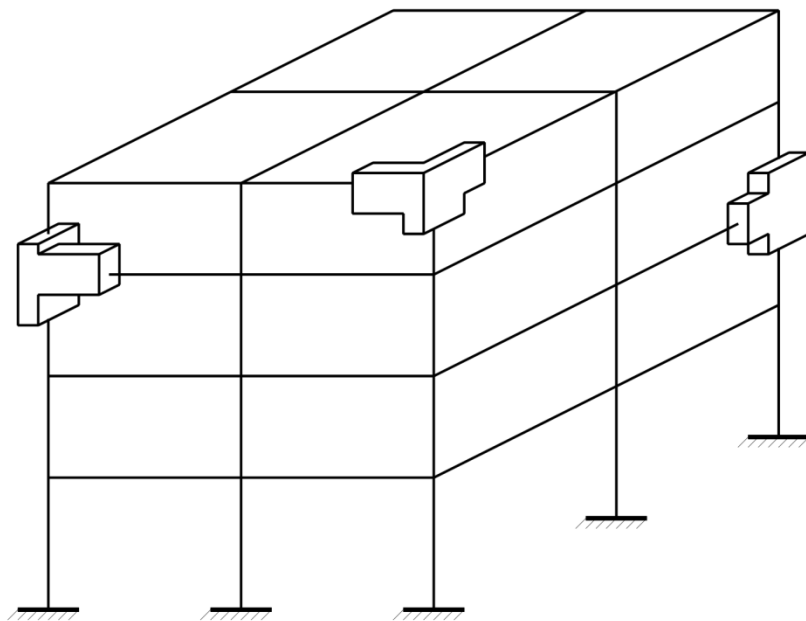


Figure 1-1: 1D and 3D elements used for various parts of a structure

Each substructure contains all the applied loads and restraints within the part of the structure that it represents (Figure 1-3), and is analysed as a normal structure with some nodes at the partition boundary, noting that the substructure analysis may or may not converge locally at a particular iteration of the nonlinear solution procedure. The resisting

forces and tangent stiffness at the partition boundary nodes is then communicated to the control process, which, considering global convergence as well as local convergence for each child structure, controls how the nonlinear analysis is progressed.

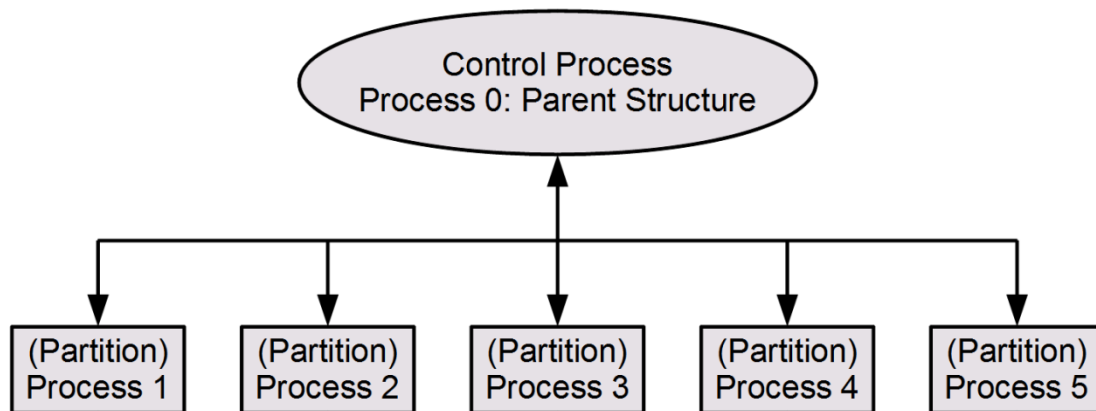


Figure 1-2: Communication between the Control Process and Partition Processes

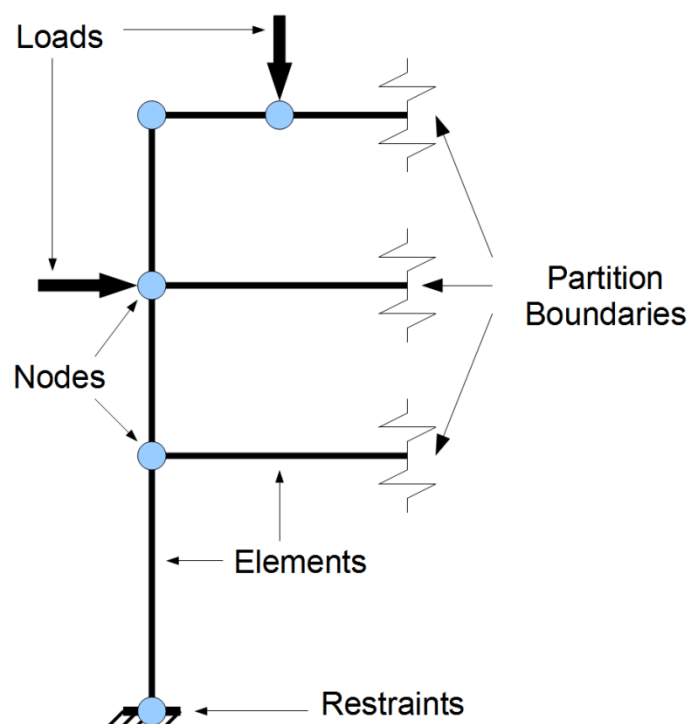


Figure 1-3: An example child partition defined by own elements, loads, restraints and partition boundaries

The developed structural domain decomposition approach as well as the dimensional coupling between elements of different dimensions has been verified at successive stages, by comparing the results to those obtained by conventional methods of nonlinear analysis.

1.5 Objectives, Originality and Benefits

The present study is aimed at developing a new scalable computational approach for nonlinear structural analysis based on hierarchic partitioning, which allows mixed dimensional coupling across partitions, and which employs parallel processing for computational efficiency.

The originality of this work lies in several features including:

1. The proposal and implementation of the concept of dual super elements which act as placeholders for interactions and communications between parent and child partitions.
2. The development of a partition control framework which can be applied to enhance existing monolithic nonlinear analysis tools for partitioned parallel processing, dealing with static, dynamic as well as eigenvalue analysis.
3. The formulation of an embedded interface at the boundary of 3D element partitions, enabling mixed dimensional coupling with 1D element parent partitions for effective part-simplification.
4. The enhancement of partitioned modelling with a hierarchic structure, which increases computational efficiency, simplifies the modelling of complex real-life problems, and can be potentially mapped onto hierarchic parallel processing hardware.

In addition to the benefits implied above, the developed hierarchic partitioning approach allows load-balancing amongst parallel processors to be practically and effectively undertaken with an appropriate choice of partitions. This avoids the need for a complex load-balancing algorithm, as would typically be required in procedure-based parallelisation. Furthermore, the developed approach potentially allows the use of different solution procedures with each partition, including different time-integration schemes, iterative

techniques and convergence criteria. Finally, this work can be potentially extended to deal with multi-physics problems, where the various partitions may represent a different physical system and modelled with a corresponding computational tool.

1.6 Scope

The scope of this work includes the following:

- Developing a computational framework for partitioned nonlinear structural analysis utilising MPI for parallel processing that:
 - can be used to analyse any number of partitions simultaneously running on processors connected via any network,
 - is hierarchic with unlimited levels of partitioning, and
 - is applicable to static, dynamic and eigenvalue analysis.
- Developing dual partition super elements that:
 - represent a partition in the parent structure,
 - wrap the partition boundary of the child structure, and
 - communicate the interactions between the parent and child partitions and information about local convergence.
- Developing an embedded interface formulation for dimensional coupling that:
 - models interactions between 1D and 3D element partitions, and
 - links a 1D master node to any number of 3D slave nodes.
- Verifying the hierarchic partitioning approach by comparing the results with those obtained using a monolithic treatment where possible.
- Verifying dimensional coupling of mixed 1D/3D models through comparison against full 3D models.

1.7 Thesis Outline

In Chapter 2, a literature review is presented on the mixed dimensional coupling of structures which has applications in multi-scale modelling, partitioning approaches based on domain decomposition, along with some of the computational methods and techniques that have been considered in this research.

The literature review is followed by two chapters related to the partitioning approach developed in this work. Chapter 3 describes the domain decomposition approach with application to nonlinear static analysis under proportional and non-proportional loading as well as dynamic time-history analysis. Chapter 4 extends the application of the partitioned approach to eigenvalue frequency analysis based on the Lanczos algorithm. These chapters also present verification examples which demonstrate the accuracy of the developed approach, and which demonstrate various benefits including reduction in wall clock analysis time via parallel processing as well as the ease of modelling with the modular partitioned approach.

Chapter 5 presents 1D/3D dimensional coupling based on a master-slave approach, where a coupling formulation utilising a single 1D master node with any number of 3D slave nodes is developed accounting for large displacements and rotations.

Chapter 6 presents the hierarchic extension of the partitioning approach presented in Chapters 3 and 4 and the dimensional coupling approach presented in Chapter 5.

Chapter 7 considers four case studies which illustrate the application of the developed hierarchic partitioning approach to the efficient and practical modelling of real nonlinear structural analysis problems.

Chapter 8 presents the conclusions drawn from this research and identifies areas for future research.

Literature Review

This chapter presents a review of partitioned modelling and the associated computational techniques used for nonlinear structural analysis

2.1 Introduction

The analysis of nonlinear structural systems has gained great importance over the past few decades. Significant research continues to be carried out in order to develop techniques that are able to analyse more complex structural systems with greater accuracy whilst requiring less computing time. This is because of two factors: on the one hand, the growth of human population has necessitated the construction of larger and more complex structures, while on the other hand, there is a greater need for understanding the structural response under extreme actions, both natural and human induced. It is therefore indispensable that the analysis should represent the response of the structure as accurately as possible so that it may be designed for safety.

The advent of computers has facilitated the development of numerous analysis techniques that were impractical to perform using pen and paper. Among a host of such techniques, some of which are discussed later in this chapter, finite element analysis is the most significant one because of the variety of analysis types, element formulations and constituent modelling options it offers. As a result, finite element analysis is being used in most of the commercial structural analysis software being employed in the design offices throughout the world.

The widespread use of computers for structural analysis notwithstanding, the computational demand of the accurate 3D analysis accounting for the material and geometrical nonlinearities is such that it is still impractical. In addition to the prohibitive problem size of the numerical model, its complexity makes the preparation of the model itself a daunting task. As a result, the use of such detailed analysis is mostly limited to very

few examples for the purposes of research in academic institutions. In this context, the present research is concerned with developing techniques that should enable the more practical and efficient nonlinear analysis of complex structural systems using detailed finite element modelling.

The above stated objective is achieved by enhancing computational efficiency through parallelisation of the solution after problem decomposition into subdomains in a hierarchic manner, and overcoming the memory bottleneck in the case of large problems through the use of distributed memory systems. The physical partitioning of the domain is undertaken through the development of dual partition super elements that facilitate not only the hierarchic partitioning but also the use of partition modules.

In addition to problem decomposition and hierarchic parallelisation, an additional approach of part simplification is used to further enhance computational efficiency. Part simplification is achieved with the use of lower dimension elements in parts of structure that are not critical while using higher dimension elements in the critical regions. This is enabled through the formulation of dimensional coupling between 1D and 3D elements.

This chapter presents a review of related literature in order to identify the prevailing methods and knowledge gaps. The topics reviewed in this chapter include analysis methods, domain decomposition, dimensional coupling, and parallelisation techniques.

2.2 Analysis Methods

Engineering computations are carried out to obtain information concerning the response of physical systems to certain imposed conditions, generally called *loads*, which is then used for making and/or justifying engineering decisions [Szabo & Babuska, 1991]. There are several mathematical and computational techniques available for this purpose; continuous models based on differential equations, finite element models, boundary element models, physical models and phenomenological models are just some examples of such techniques. From amongst these numerous analysis methods, the finite element method is probably the most commonly applied owing to its capacity for solving problems of enormous size and its applicability to a wide range of engineering problems.

2.2.1 Finite Element Analysis

Although originally developed to study stresses in complex airframe structures, the finite element analysis method has since been extended and applied to a wide spectrum of problems, including those in the field of continuum mechanics. Because of its diversity and flexibility as an analysis method, it has received much attention in engineering schools as well as industry [Huebner et al., 2001].

Individual finite elements can be visualized as small pieces of a structure. The word ‘finite’ distinguishes these pieces from infinitesimal elements used in differential calculus. In each finite element, a field quantity is allowed to have only a simple spatial variation, perhaps described by polynomial terms up to a certain order, generally referred to as the ‘shape function’. Since the actual variation in the region spanned by an element is typically much more complicated, finite element analysis provides an approximate solution [Cook et al., 2002]. The accuracy of finite element computations increases with the order of the polynomial being used as well as with the reduction in the finite element mesh size; the greater the number of degrees of freedom, the more closely the solution will approximate the true one [Zienkiewicz et al. 2005a].

2.2.1.1 Sources of Error in Finite Element Analysis

Apart from the mistakes made by users of finite element analysis tools, there are three main sources of error in finite element method [Cook et al., 2002], as listed below:

- Modelling/Formulation Errors
- Discretisation Errors
- Numerical Errors

Modelling or formulation errors occur when a physical system is represented as a simplified mathematical model, for example, idealisation of support boundary conditions as rigid, assuming that the material behaves linearly, simplification of the applied loads, etc. Discretisation errors arise when this mathematical model is converted into a finite element model with a finite number of degrees of freedom, integration rules, etc. Numerical errors

are mainly caused by the finite precision of the computing machine, though these are outside the scope of this work.

Discretisation errors are controlled mainly by increasing the number of elements or elements with higher order shape functions. Modelling errors, on the other hand, can be addressed, at least in part, by using higher dimensional elements as well as incorporating material and geometrical nonlinearities in the finite element formulations so as to capture the real structural response, as discussed in the following section.

2.2.1.2 Nonlinearities in Finite Element Analysis

The analysis of structures becomes more complicated with the introduction of nonlinearities. “In practical applications the limitation of linear elasticity, or more generally of linear behaviour, often precludes obtaining an accurate assessment of the solution because of the presence of ‘non-linear’ effects and/or because the geometry has a ‘thin’ dimension in one or more directions” [Zienkiewicz et al., 2005b]. There are two main sources of nonlinearity in structures: material and geometric.

Material nonlinearity comes into effect when, for example in elasticity, the stress is not linearly proportional to the strain. Another example is the elastic-plastic behaviour when the loading and unloading response of the material is different. The structure is said to be geometrically nonlinear when the deformations are no longer considered to be infinitesimal and the distinction is made between the initial ‘undeformed’ shape of the structure and the current ‘deformed’ shape after the loading is applied. Geometric nonlinearity arises due to the fact that equilibrium is expressed, and correctly so, in the unknown deformed configuration.

Several approaches have been developed to deal with the nonlinearities in the structural response. Spacone & El-Tawil (2004) discussed in detail the approaches used to tackle these nonlinearities while presenting the state of the art in nonlinear analysis of steel-concrete composite structures. The models used to represent material nonlinearity can be generally classified into ‘resultant section models’ and ‘fibre section models’.

Resultant section models are based on explicit definition of cross-section responses such as axial force-strain response and moment-curvature response. In the fibre section models on the other hand, the section is divided into a specific number of fibres; the stresses in these individual fibres are then integrated over the cross-section to obtain force and moment responses.

Elements used for frame analysis are typically 1D element and can be generally classified into two categories; one based on the force method and the other based on the displacement method. An alternative classification can be defined in terms of lumped models and distributed models. Lumped models use a physically impossible approach of concentrating all inelasticity at the element ends for the sake of conceptual simplicity and computational convenience. The distributed models are computationally more expensive although they are more rational and give more accurate results.

The approaches to tackle geometric nonlinearity can also be classified in two categories. One is co-rotated or inexact formulations and the other is geometrically exact formulations. The co-rotational approaches [Belytschko & Hsieh, 1973; Crisfield & Moita, 1996; Izzuddin, 2001] consider arbitrarily large displacements but are based on the assumption of small strains. The geometrically-exact formulations [Ibrahimbegovic, 1995; Jelenic & Crisfield, 1999; Kapania & Li, 2003], on the other hand, are based on full 3D continuum mechanics.

The most accurate approach to carrying out the finite element analysis of a structure is to use a fine mesh of 3D finite elements, where the nonlinear material response is represented by triaxial models and where the geometric nonlinearity is incorporated into the finite element formulations. This approach, however, is prohibitively expensive, impractical and typically inapplicable to modelling the overall structural response, where the nonlinear analysis of relatively small structures can take several days of computing time [Mata et al., 2009; Yue et al., 2010; Spacone & El-Tawil, 2004].

Before discussing methods of enhancing in the computational efficiency of finite element analysis, it is worth reviewing the different areas of analysis and common solution techniques.

2.2.2 Nonlinear Static Analysis for Proportional Loading

In nonlinear static analysis for proportional loading, the applied loads are always the product of a set of nominal loads and a load factor as represented by the equation:

$$P = \lambda P_0 \quad 2-1$$

where P is the vector of applied loads, λ is the load factor and P_0 is the vector of nominal loads. If R is a vector of the resistance forces offered by the structure when subjected to the applied load, then the structure is said to be in equilibrium if the resistance forces offered are equal to the applied load, or in other words, there are no out of balance forces acting on the structure. If the out of balance forces are represented by vector G then:

$$G = R - P \quad 2-2$$

Hence, the structure is in equilibrium if:

$$G = O \quad 2-3$$

where O is a zero vector.

The response of the structure is represented by a load-deflection curve consisting of all those points where Equation 2—3 is satisfied. The points on this curve are referred to as ‘equilibrium states’, and the curve is referred to as ‘equilibrium path’; Figure 2-1 shows one such equilibrium path. It is worth noting here that the different segments of the curve shown in Figure 2-1 are illustrative only. It is possible in some cases that the ‘initial linear response’ does not exist and the response is nonlinear from the beginning. In some other cases, it is also possible that at buckling the equilibrium path bifurcates into two or more paths. Further discussion on these aspects of the equilibrium path, however, is out of the scope of present research.

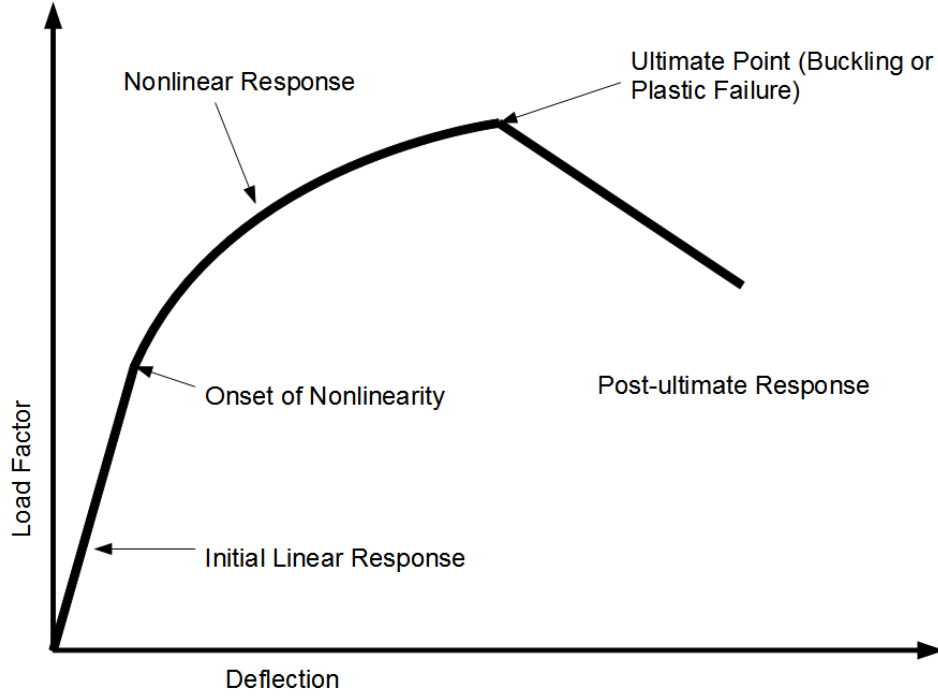


Figure 2-1: Equilibrium path (load-deflection diagram)

The slope of the load-deflection curve at any point along the equilibrium path is indicative of the ‘tangent stiffness’ of the system. If the response is linear, the tangent stiffness remains constant; in the nonlinear range, however, the tangent stiffness changes along the equilibrium path.

If the vector of nodal displacements is represented by U then the resistance forces R for a nonlinear system can be expressed as a nonlinear vector function of U :

$$R = R(U) \quad 2-4$$

Equation 2—4 combined with equations 2—1 to 2—3 gives a system of nonlinear equations as shown below that needs to be solved for the determination of an equilibrium state:

$$G = R(U) - \lambda P_0 = O \quad 2-5$$

At this point, the notion of a tangent stiffness matrix can be defined as:

$$K_{i,j} = \frac{\partial G_i}{\partial U_j} = \frac{\partial R_i}{\partial U_j} \text{ (for nominal loads } P_0 \text{ independent of } U) \quad 2-6$$

The solution of the nonlinear system of equations for U in (2-6) is undertaken through an iterative process. Up to the ultimate/limit point, a load control technique may be adopted where λ is increased at each loading step and U is determined iteratively. Beyond this point, however, the load control technique falters because a nearby equilibrium point cannot be found at a larger load factor. Therefore, the subsequent branch of the equilibrium path is determined using displacement control, where typically a specific nodal displacement is controlled while the remaining nodal displacements and the corresponding load factor are determined. The load control and the displacement control techniques are reviewed hereafter.

2.2.2.1 Load Control

In load control, a constant increment is applied to the load factor λ for each loading step, and the corresponding set of structural deflections U is obtained [Crisfield, 1991]. For any given iteration ' i ', at a given load step ' s ', the increased loading factor after the application of increment is:

$$\lambda^s = \lambda^{s-1} + \delta\lambda \quad 2-7$$

The out of balance is obtained as:

$$G^{s,i-1} = \lambda^s P_0 - R^{s,i-1} \quad 2-8$$

For the first iteration, the values of G and R correspond to the solution from the previous load step. The stiffness matrix can be used as a first-order guide to obtain iterative corrections of displacement:

$$\delta U^{s,i} = -K^{-1} G^{s,i-1} \quad 2-9$$

where K^{-1} is the inverse of the tangent stiffness matrix, where Equation 2—8 represents the solution of a system of simultaneous linear equations. This is computationally the most expensive part of the load control technique depending on the size and sparseness of the tangent stiffness matrix.

With the iterative displacement corrections obtained as above, the new displacements are obtained as:

$$U^{s,i} = U^{s,i-1} + \delta U^{s,i} \quad 2-10$$

The iterative process suggested by Equations 2—7 to 2—9 is continued until the out of balance becomes smaller than a predefined tolerance limit. When this is achieved, the structure can be considered to have attained the equilibrium state for the current load step. The next increment to the load factor is then applied as per Equation 2—6 and the iterative process is repeated for the new load step. If the new load factor is beyond the limit point, convergence of the iterative process will not be achieved and the solution may actually start diverging. In this case, the analysis may either be stopped or the control may be transferred from load control to displacement control, which is described in the following section.

2.2.2.2 Displacement Control

As suggested by the name, a specific nodal displacement, say U_f is controlled in this technique, by applying a constant increment to it, in return to making the load factor λ unknown. The iterative procedure for the displacement control is presented as follows [Batoz and Bhatt, 1979; Izzuddin, 1991]:

$$K\delta U^{s,i} = \lambda^{s,i} P_0 - R^{s,i-1} = (\lambda^{s,i-1} + \delta \lambda^{s,i}) P_0 - R^{s,i-1} \quad 2-11$$

$$\Rightarrow \delta U^{s,i} = K^{-1} [\delta \lambda^{s,i} P_0 + \lambda^{s,i-1} P_0 - R^{s,i-1}] \quad 2-12$$

$$\Rightarrow \delta U^{s,i} = K^{-1} [\delta \lambda^{s,i} P_0 - G^{s,i-1}] \quad 2-13$$

If the controlled nodal freedom is ' f ', and the increment to be applied to the controlled displacement U_f is ΔU^s , then the iterative procedure suggested by Equations 2—11 to 2—13 can be represented as follows:

The out of balance for the previous iteration using the load factor and resistance forces from that iteration is determined, followed by the solution of linear system of equations for the determination of the corresponding displacement correction:

$$G^{s,i-1} = R^{s,i-1} - \lambda^{s,i-1} P_0 \quad 2-14$$

$$\delta_g U^{s,i-1} = -K^{-1} G^{s,i-1} \quad 2-15$$

$$_p U^{s,i-1} = K^{-1} P_0 \quad 2-16$$

where $\delta_g U$ are the iterative displacements due to the out of balance forces, whereas $_p U$ are the displacements due to the nominal loads. Subsequently, the corrections as well as the new values of the load factor and the nodal displacements are obtained as:

$$\Delta U_f^{s,i} = \Delta U^s - (U_f^{s,i-1} - U_f^{s,0}) \quad 2-17$$

$$\delta \lambda^{s,i} = \frac{\Delta U_f^{s,i} - \delta_g U_f^{s,i-1}}{_p U_f^{s,i-1}} \quad 2-18$$

$$\delta U^{s,i} = \delta \lambda^{s,i} _p U^{s,i-1} + \delta_g U^{s,i-1} \quad 2-19$$

$$\lambda^{s,i} = \lambda^{s,i-1} + \delta \lambda^{s,i} \quad 2-20$$

$$U^{s,i} = U^{s,i-1} + \delta U^{s,i} \quad 2-21$$

2.2.2.3 Arc Length Control

Arc length control is an extension to displacement control [Riks, 1979; Hellweg & Crisfield, 1998], where instead of controlling a single displacement, the norm of the incremental nodal displacements vector ΔU is prescribed. This technique differs from the standard displacement control technique in the calculation of the increment of load factor, which involves the solution of a quadratic equation give as:

$$a(\delta \lambda^{s,i})^2 + b \delta \lambda^{s,i} + c = 0 \quad 2-22$$

where $a = _p U^{s,i} \cdot _p U^{s,i}$, $b = 2 _p U^{s,i} (\Delta U + \delta_g U^{s,i})$, $c = (\Delta U + \delta_g U^{s,i})(\Delta U + \delta_g U^{s,i}) - \Delta r^2$, and, Δr is the arc-length and $\Delta U = U^{s,i-1} - U^{s,0}$

The rest of the iterative process remains the same as the displacement control.

2.2.3 Static/Dynamic Time-History Analysis

In time-history analysis, which can be either static or dynamic, the loads may vary independently in the time or pseudo-time domain. If the values at a given time are identified with a superscript t , the load associated with the j^{th} degree of freedom is expressed as:

$$P_j^t = \lambda_j^t P_{0j} \quad 2-23$$

The out of balance is similarly expressed as:

$$G_j^{t,i-1} = R_j^{t,i-1} - \lambda_j^t P_{0j} \quad 2-24$$

where R includes inertia and damping resistance forces. Through the choice of an appropriate time-integration scheme, such as the implicit Newmark scheme [Newmark, 1959], R can again be expressed as a nonlinear vector function in terms of U .

Once the resistance and out of balance forces are obtained, the incremental displacements and the updated global displacements can then be determined in a manner similar to load control in static proportional loading:

$$\delta U^{t,i} = -K^{-1} G^{t,i-1} \quad 2-25$$

and:

$$U^{t,i} = U^{t,i-1} + \delta U^{t,i} \quad 2-26$$

2.2.4 Iterative Solution Procedures

As discuss before, to solve the nonlinear system of equations, presented for example by Equation (2-24), an iterative strategy is generally employed. The most popular amongst the various iterative strategies are the Newton-Raphson method and the modified Newton-Raphson method, which were discussed in some detail by Stricklin et al. (1971), and Powell and Simons (1981). In the Newton-Raphson method, the tangent stiffness matrix K is determined at every iteration as illustrated in Figure 2-2, whereas in the modified Newton-

Raphson method, K is determined only for the first iteration of each load step as shown in Figure 2-3.

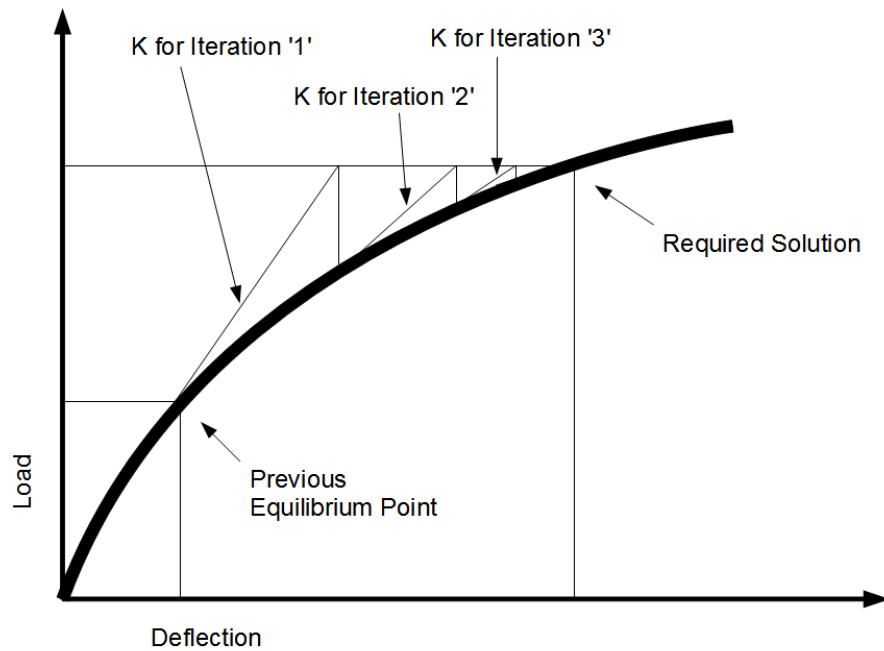


Figure 2-2: Newton-Raphson method

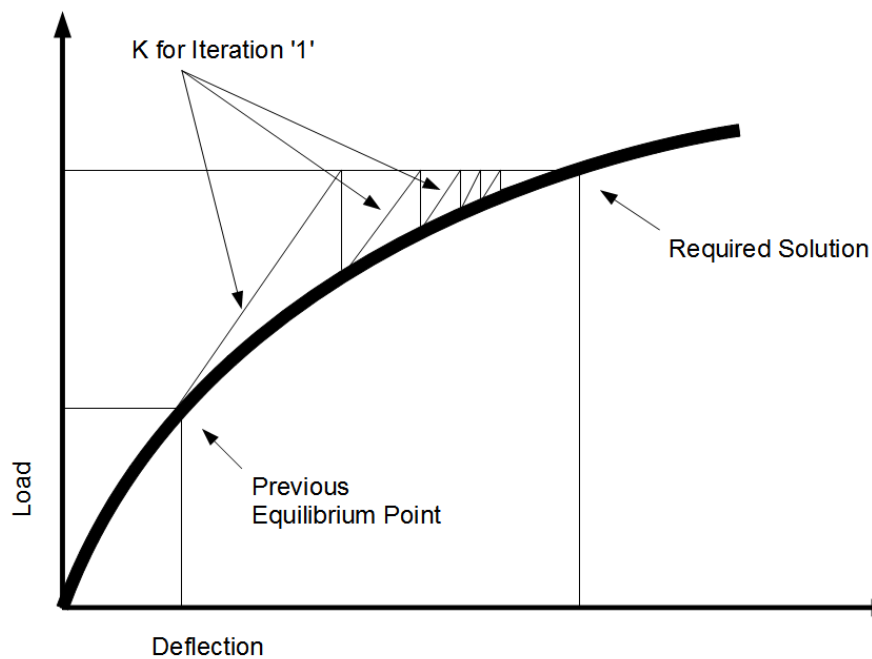


Figure 2-3: Modified Newton-Raphson method

The Newton-Raphson method while requiring fewer iterations, involves the computation and assembly of tangent stiffness matrix for each iteration, which can be computationally expensive. The modified Newton-Raphson method avoids the assembly of the tangent stiffness matrix for each iteration, though the total number of iterations required is more in this case.

Izzuddin (1991) adopted a general strategy in ADAPTIC of which the Newton-Raphson method and the modified Newton-Raphson method are special cases. According to this strategy, a number of initial stiffness reformations is pre-specified, meaning that the tangent stiffness is determined for this number of initial iterations in each load stage when the change in stiffness between iterations is more significant, where the last stiffness matrix is then used for the later iterations. Therefore, this approach combines the convergence and computational benefits of both Newton-Raphson and modified Newton-Raphson methods.

For the assembly of the structure resistance forces and the global tangent stiffness matrix, the contribution of individual elements is obtained according to the element formulation. Detailed discussion on the formulation of different types of elements and their relevant contributions to the global tangent stiffness matrix and resistance forces is outside the scope of this chapter. In general, the assembly process of the global tangent stiffness and the resistance forces can be represented by:

$$R = \sum_e r \quad 2-27$$

and:

$$K = \sum_e k \quad 2-28$$

For the solution of the linearised system of simultaneous equations, the assembled tangent stiffness matrix can be reduced to upper echelon form by the use of Gaussian Elimination. This can be performed by applying a series of row operations on the tangent stiffness matrix once the entire matrix has been assembled. This approach, however, is computationally very expensive both in terms of memory requirements for the tangent stiffness matrix as well as the number of mathematical operations required to perform the required row

operations. In order to reduce the computational cost of such an operation, different methods may be used, though the 'Frontal Method', proposed by Irons (1970), has particular benefits in respect of the present work, as discussed in Chapter 3. A review of this method is provided in the next section.

2.2.5 Frontal Method

It is possible to eliminate the freedoms associated with a node after the contributions of all the elements connected to that node have been accounted for, and these eliminated freedoms need not to be considered further in the rest of the elimination process. This fact offers a situation where instead of assembling the entire tangent stiffness matrix, only a small 'front', hence the name 'frontal method', is maintained where as soon as all the element contributions from elements attached to a node are obtained, that node is eliminated. After each such elimination, the size of the nodal front is reduced, which is expanded again by taking in the contribution from the next element in the order. Thus only a subset of the stiffness matrix, sometimes referred to as 'Grandpa', is being considered at any one time.

For computational efficiency, it is important that the size of the Grandpa should remain as small as possible throughout the process. This is because the elimination of one degree of freedom from a larger matrix is comparatively more expensive than that from a smaller matrix. The reduction of a 5×5 matrix to 4×4 , for example, requires 9 multiplication operations, whereas the reduction of a 4×4 matrix to 3×3 requires only 6 multiplication operations.

The size of the 'Grandpa' is dependent on the order in which the elements are taken into consideration. Two different element orders of the same structure as shown in Figure 2-4 result in different 'Grandpa' sizes, where the maximum frontal width changes from 4 to 8 between the two configurations.

where A is a square matrix, x is a vector and ψ is a scalar. Each value of ψ that satisfies the above equation is called an *eigenvalue* of the matrix A , and vector x corresponding to that eigenvalue is called an *eigenvector*. The total set of all eigenvectors is often given a collective name of *eigenspace*.

There are many applications of eigenvalue problem ranging over a wide spectrum of science and engineering fields like quantum physics, geology, glaciology, graphics and many more. The response of an airplane to air turbulence and the response of a power system network to a fault in the network are two examples of the application of eigenvalue analysis [Balle & Cullum, 1999]. The most widespread use of the eigenproblem is its application for finding out the natural frequencies and corresponding mode shapes in case of free vibration [Hughes, 2002; Parlett and Scott, 1979].

In the case of dynamic analysis of structures, the governing equilibrium equation for the undamped free vibration of a typical mode is given as:

$$([K] - \omega^2 [M])\{u\} = \{0\} \quad 2-30$$

where K is the stiffness matrix, M is the mass matrix, ω is natural frequency, and vector u is the associated mode shape.

The above equation is referred to in literature as ‘general eigenvalue problem’ in the context of structures [e.g. Wilson, 2002; Hughes, 2002]. It can be easily transformed into a standard eigenvalue problem of the form presented in Equation 2—32, where:

$$\psi = \frac{1}{\omega^2 - \mu} \quad 2-31$$

and:

$$A = (K - \mu M)^{-1} M \quad 2-32$$

The term μ in the above two Equations 2—31 and 2—32 is referred to as the ‘shift’ and the matrix A of the standardised eigenvalue problem is called a ‘matrix pencil’.

The standardised eigenvalue problem can either be solved directly or by the use of some iterative solution procedure. The direct solution of this problem involves the development of the characteristic polynomial for the determinant and obtaining the roots:

$$|K - \psi M| = 0 \quad 2-33$$

The roots of Equation 2—34 give the required set of eigenvalues for the given structure. The formation of such an equation, however, becomes impractical when matrices larger than say a 3×3 matrix are involved. In such cases, an iterative procedure is used. There are many iterative procedures available for the solution of such problems, the most significant among these procedures being the power iteration method, the inverse power iteration method, the QR and QL algorithms [Parlett, 1998], and Lanczos Algorithm [Lanczos, 1950].

2.2.6.1 Power and Inverse Power Iteration Methods

The power iteration method computes only the dominant eigenvalue of a given matrix assuming that a dominant eigenvalue exists i.e. if $\{\psi, \psi_1, \psi_2, \psi_3, \dots, \psi_n\}$ is a set of $n+1$ eigenvalues of matrix A then ψ is called the dominant eigenvalue of A if:

$$|\psi| > |\psi_i| \quad \forall \quad i=1 \rightarrow n \quad 2-34$$

The eigenvector corresponding to the dominant eigenvalue is called the dominant eigenvector and it is computed by starting with a random vector assuming that this random vector has some component along the desired eigenvector, that is if x is the desired vector, and x^0 is the starting random vector, then:

$$x \cdot x^0 \neq 0 \quad 2-35$$

If the above mentioned assumption is true, then the starting vector x_0 can be modified iteratively as below:

$$x^k = \frac{Ax^{k-1}}{\|Ax^{k-1}\|} \quad 2-36$$

where the superscript k represents the number of the current iteration.

The iterative process represented by Equation 2—35 is repeated until x^k converges to x :

$$x^k \cong x \quad 2-37$$

Once the dominant eigenvector has been computed, the corresponding eigenvalue can be calculated using Rayleigh's quotient:

$$\psi = \frac{Ax \cdot x}{x \cdot x} \quad 2-38$$

The inverse power iteration method, on the other hand, uses the inverse of matrix A during the iterative process, which results in Equation 2—35 being modified to:

$$x^k = \frac{A^{-1}x^{k-1}}{\|A^{-1}x^{k-1}\|} \quad 2-39$$

If matrix A had the set of eigenvalues as $\{\psi, \psi_1, \psi_2, \psi_3, \dots, \psi_n\}$, then by definition the set of eigenvalues of A^{-1} is given as $\{1/\psi, 1/\psi_1, 1/\psi_2, 1/\psi_3, \dots, 1/\psi_n\}$. Consequently the iterative process presented by Equation 2—38 results into the eigenvector corresponding to the eigenvalue that is smallest in magnitude. Using a shift μ approximately equal to a desired eigenvalue, the inverse iteration method may be modified as below to get the corresponding eigenvector:

$$x^k = \frac{(A - \mu I)^{-1} x^{k-1}}{\|(A - \mu I)^{-1} x^{k-1}\|} \quad 2-40$$

Although the inverse power iteration method can possibly calculate any eigenvalue using the shift as represented by Equation 2—39, it requires a reasonable approximation of the desired eigenvalue to start with.

The convergence of both power and inverse power iteration methods is not guaranteed as it is subjected to the assumptions and initial estimations mentioned above. When these methods do converge, their convergence rate is slow resulting in a high computational demand. Moreover, they become even more computationally inefficient when more than one eigenvalue and corresponding eigenvector are required.

2.2.6.2 QR and QL Algorithms

The QR Algorithm is based upon ‘QR decomposition’, a factorisation in which a real, symmetric, square matrix ‘ A ’ is decomposed into an orthogonal matrix ‘ Q ’ and an upper triangular matrix ‘ R ’, sometimes also referred to as the ‘right triangular matrix’. These matrices are then multiplied again in reverse order to get an updated version of ‘ A ’, which is again decomposed in the similar manner. The resulting iterative process can be represented as:

$$Q^k R^k = A^k \quad 2-41$$

$$A^{k+1} = R^k Q^k \quad 2-42$$

The iterative process represented by Equations 2—40 and 2—41 is continued until A^{k+1} is reduced to a diagonal matrix. These diagonal values of A^{k+1} are the eigenvalues of A and their corresponding eigenvectors are calculated as:

$$S = Q^1 \times Q^2 \times \dots \times Q^k \quad 2-43$$

where, the i^{th} column of S is the eigenvector corresponding to the eigenvalue A_{ii}^{k+1} .

The QL Algorithm differs from QR Algorithm only in the fact that it is based on ‘QL decomposition’. In QL decomposition, instead of upper or right triangular matrix, the matrix A is decomposed into an orthogonal matrix Q and a lower triangular matrix L , sometimes also referred to as the ‘left triangular matrix’. The rest of the iterative process remains the same in both algorithms.

Both QR and QL algorithms calculate all the eigenvalues of a square, real, symmetric matrix. This capacity of these algorithms may be very beneficial in cases where the size of the matrix is not too large and where all the eigenvalues may be wanted. This is not typically the case in structural frequency analysis, where often the structure consists of thousands of degrees of freedom resulting in large stiffness and mass matrices. Moreover, not all the eigenvalues are sought in this case because only a few extreme eigenvalues are typically of significance.

The direct use of the QR or QL algorithm in structural analysis is, therefore, not advisable as it is not computationally cost effective.

2.2.6.3 Lanczos Algorithm

Lanczos (1950) presented an algorithm for swiftly calculating the extreme eigenvalues of large, sparse matrices. In the introduction to the ‘Lanczos Algorithm’, the author describes the benefits of this algorithm as:

1. The iterations are used in the most economical fashion, obtaining an arbitrary number of eigenvalues and eigensolutions by one single set of iterations, without reducing the order of the matrix.
2. The rapid accumulation of fatal rounding errors, common to all iteration processes if applied to matrices of high dispersion (large “spread” of the eigenvalues), is effectively counteracted by the method of “minimized iterations”.
3. The method is directly translatable into analytical terms, by replacing summation by integration. We then get a rapidly convergent analytical iteration process by which the eigenvalues and eigensolutions of linear differential and integral equations may be obtained.

The Lanczos algorithm based on a recursive process of orthogonal transformation of the given matrix was not very successful initially. This was because the successive Lanczos vectors generated in the process lost orthogonality due to the limited precision of the computing machines. Paige (1972), however, showed that ‘the loss of orthogonality and the convergence (of an eigenvalue) go hand in hand’. This find revived the Lanczos algorithm, and it has since been considered the most efficient algorithm for calculating the extreme eigenvalues of large, sparse matrices [Hughes, 2000].

Due to its efficiency compared to other algorithms, the Lanczos Algorithm is used for the parallelisation of eigenvalue analysis in this work. The algorithm and its implementation along with parallelisation are discussed in more detail in Chapter 4.

2.3 Domain Decomposition

The computational efficiency of nonlinear finite element analysis of large and complex structural systems can be enhanced by parallelisation. One way of achieving this is by dividing the problem into sub-problems and then running these sub-problems simultaneously on parallel processors connected to a network. As discussed in Section 2.5, parallelisation can either be based on shared memory systems or distributed memory systems. The use of shared memory systems, however, can cause memory bottlenecks for problems of considerably large sizes. To overcome this issue, it is necessary to use distributed memory systems with domain decomposition.

Several domain decomposition techniques exist which can be divided into two main categories: mathematical and physical. In mathematical domain decomposition, the various mathematical functions of the problem are divided into sub-problems. This kind of problem subdivision, however, will require 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 sub-structures with interfaces at the boundary. It is necessary to make sure that these interfaces at the boundaries of the subdomains do satisfy the requirements of compatibility and equilibrium.

Different approaches used for solving the physical domain decomposition problems include the staggered approach and iterative coupling methods. These approaches are briefly reviewed below.

2.3.1 Staggered Approach

The staggered approach is mainly applicable to dynamic analysis only where the partitioned subdomains utilise a predictor to predict the interface boundary conditions using those at the previous time step [Felippa and Park, 1980; Zolghadr Jahromi et al., 2007]. The solution procedure for a domain divided into two subdomains for example is shown in Figure 2-5.

Due to the fact that the displacements at the interface boundary of one domain are predicted without the communication of actual boundary interface displacements between domains, this approach does not necessarily satisfy compatibility. Another major disadvantage with the staggered approach is that achieving stability in the numerical procedure is extremely difficult without reformulation of the field equations of the original partitioned sub-domains [Felippa et al., 2001].

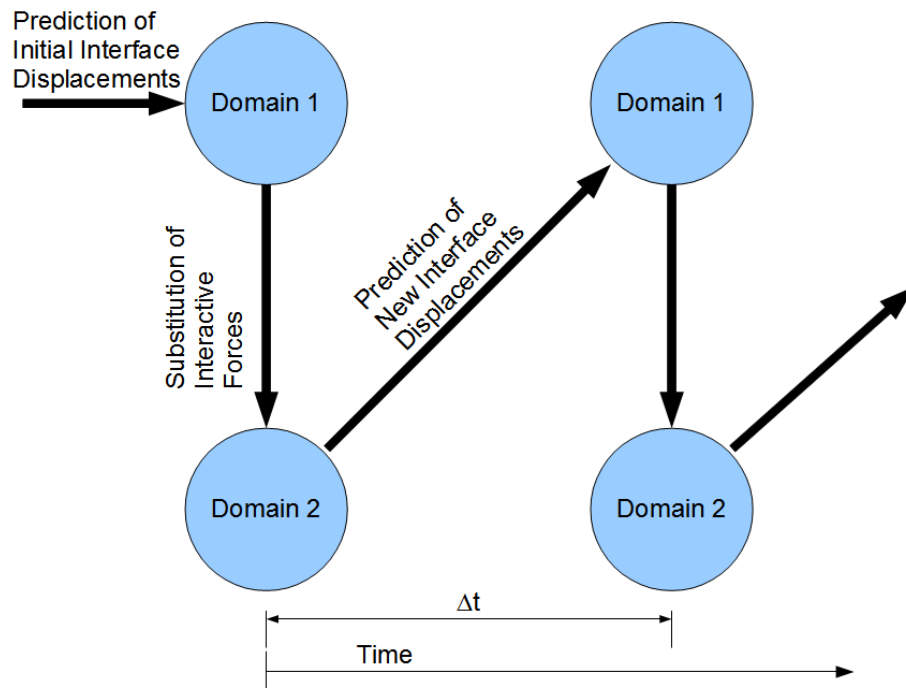


Figure 2-5: Staggered approach for a two domain system

Therefore, the staggered approach can only achieve a conditional stability which is dependent upon the size of the time-step, and is therefore computationally inefficient.

2.3.2 Iterative Coupling Methods

To overcome the time-step size dependence of the staggered approach, corrective iterations are introduced which results in the development of iterative sub-structuring or coupling methods [Quarteroni & Valli, 1999]. These methods support parallel computations and can be applied to both static and dynamic problems. The general procedure for iterative coupling methods is illustrated in Figure 2-6.

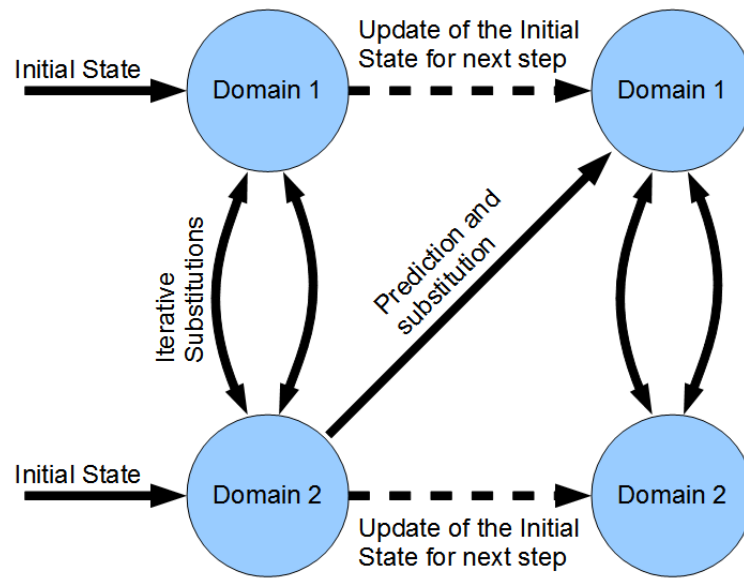


Figure 2-6: Schematic diagram for iterative coupling

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. The predicted boundary conditions 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. In this case, the solution dependency shifts from time-step size to the type of iterative update employed for the displacement/force interface boundary conditions for each subdomain.

To improve the convergence of the iterative coupling methods, boundary conditions can be updated using an 'interface relaxation approach' [Marini & Quarteroni, 1989; Mu, 1999]. The solution using the interface relaxation approach depends upon the relaxation parameter used which in turn is problem dependent.

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 [Zolghadr Jahromi et

al., 2009]. 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.

2.3.3 Lagrange Multiplier Techniques

Several domain decomposition techniques based on Lagrange multipliers for linking the interfaces were previously proposed. Kron (1939) presented a method for linking sub-domains for elasticity problems by using Lagrange multipliers. According to this method, if two domains Ω^1 and Ω^2 are to be linked, discretisation of displacements in each domain and of the Lagrange multipliers, also referred to as tractions on the interface, yields the system of equations as follows:

$$\begin{bmatrix} K^1 & 0 & Q^1 \\ 0 & K^2 & Q^2 \\ Q^{1T} & Q^{2T} & 0 \end{bmatrix} \begin{Bmatrix} u^1 \\ u^2 \\ \lambda \end{Bmatrix} = \begin{Bmatrix} f^1 \\ f^2 \\ 0 \end{Bmatrix} \quad 2-44$$

In the above expression, u^1 , u^2 , and λ are linked to the displacement fields u^1 , u^2 , and Lagrange multipliers λ by the shape functions N_1 , N_2 , and N_λ as:

$$u^i = N_i u^i; \lambda = N_\lambda \lambda \quad 2-45$$

and:

$$K^i = \int_{\Omega^i} B^{iT} D^i B^i d\Omega \quad 2-46$$

$$Q^i = (-1)^i \int_{\Gamma_I} N_i^T N_\lambda d\Gamma \quad 2-47$$

$$f^i = \int_{\Omega^i} N_i^T b^i d\Omega + \int_{\Gamma_I} N_1^T t^i d\Gamma \quad 2-48$$

where B , D , N , and b have the conventional meaning used in finite element methods, i.e. B is the strain-displacement transformation matrix, D is the constitutive matrix, N is the matrix of the shape functions, and superscript i represents the sub-domain number, Γ represents the sub-domain boundary and subscript I represents the interface between the two sub-domains.

Based on this method, Farhat and Roux (1991) presented a method of finite element tearing and interconnecting, more commonly known as the FETI method. Many researchers have since worked on the further development of the FETI method and its applications to different cases [Magoules & Roux, 2007; Kruis, 2007].

The basic domain decomposition approach based on Lagrange multipliers is subject to limitations imposed by the stability and consistency of the mixed patch test for the selection of an appropriate number of Lagrange multipliers. Furthermore, for a subdomain lacking rigid body constraints, the elimination of local sub-domain displacements is generally impossible [Zienkiewicz, et al. 2005b]. Moreover, this approach does not facilitate the use of differently dimensioned elements or heterogeneous solution procedures for example implicit/explicit time integration.

2.3.4 Displacement Frame Methods

An alternative to domain decomposition methods based on Lagrange multipliers is an approach where sub-domains 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 sub-domain 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.

Consider the frame displacements to be approximated as:

$$u = Nu^d \tag{2-49}$$

where u are the displacements on the sub-domain boundary, u^d are the nodal displacements for the interface frame, and N is the shape function of the sub-domain. If Γ_I is the interface boundary, virtual work can be used with discretisation to obtain the nodal forces as:

$$q = \int_{\Gamma_I} N^T t d\Gamma \tag{2-50}$$

where t are the tractions the sub-domain exerts on the displacement frame. When the tractions are expressed in terms of the frame parameters only, we get:

$$q = Ku^d + f \quad 2-51$$

where K is the condensed stiffness matrix of the sub-domain Ω , and f are its equivalent boundary forces. Global equilibrium can then be ensured by the condition of zero resultant forces from all the interface frames:

$$\sum_j q^j = 0 \quad 2-52$$

With the linearity implied by equation 2—51, this approach requires further enhancement for application to problems involving geometric and material nonlinearity. The present work borrows some of the concepts of the displacement frame technique to develop a new method for partitioned analysis of nonlinear systems including geometric and material nonlinearity.

2.3.5 Multi-scale Modelling

The numerical simulation of the nonlinear response of structures made of heterogeneous materials can be effectively carried out with multi-scale modelling. In a typical two-scale strategy, the modelling is carried out at two different scales often referred to as the ‘macroscale’ or the ‘structural scale’ and the ‘mesoscale’. There are several strategies of linking these two scales together [Markovic and Ibrahimbegovic, 2003] depending upon the degree of coupling used. When the two scales are coupled only weakly and when their characteristic lengths are very different, a two level finite element procedure [Feyel, 2003], the so-called FE^2 approach, is often used. In this approach, a finite element continuous model is used at structural scale, while a second level finite element model is formulated at mesoscale for the heterogeneous material. The mesoscale model is associated with a representative volume element (RVE), while in the finite element model at the structural level one RVE is linked to each integration point. One example of such modelling [Izzuddin

and Macorini, 2011] is the model for unreinforced masonry (URM) perforated wall as shown in Figure 2-7.

macroscale FE mesh for URM perforated wall

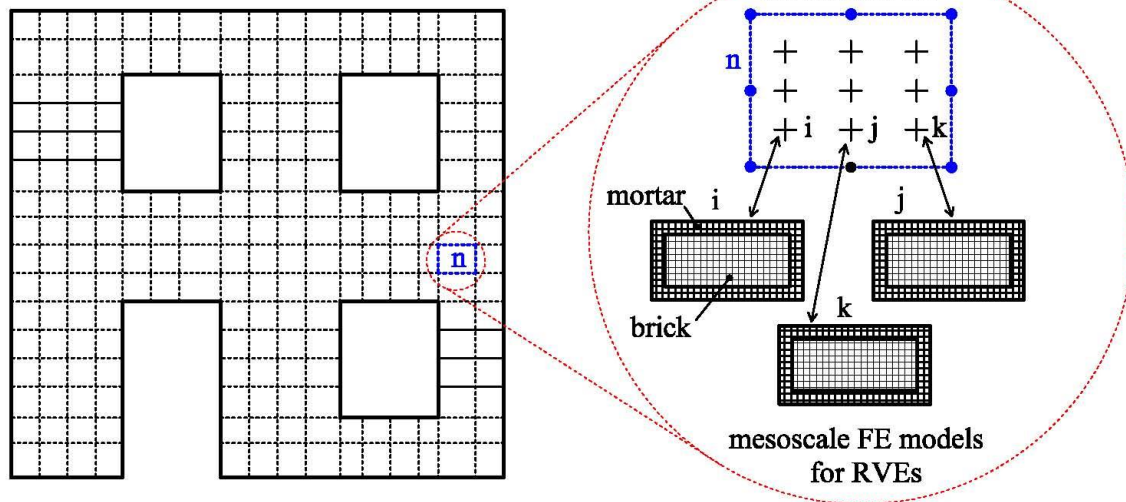


Figure 2-7: Two-level strategy for modelling a URM perforated wall [Izzuddin and Macorini, 2011]

The multi-scale approach has recently been used successfully to analyse the in-plane [Massart et al., 2007; Taylor & De Bellis, 2011] and out-of-plane response [Mercatoris & Massart, 2011] of running bond masonry structures. While this two-scale finite element formulation is accurate and suitable for the analysis of running bond masonry, its application in more general conditions for structures with complex textures and subjected to extreme loading is questionable. Especially in cases where the size of the RVE is comparable to the characteristic length of the finite elements used at the structural scale, the spacing of the integration points can be so small leading to the overlapping of different RVEs, which is unrealistic.

An alternative to the use of RVE for multi-scale modelling can be hierarchic domain decomposition, where the structural level is decomposed into partitions, each of which can be further decomposed into partitions, and so on. Hierarchic partitioning has conceptual

modelling advantages, but more importantly it can bring significant computational benefits, and is therefore chosen as one of the main topics for the present research.

2.3.6 Hierarchic Partitioning

Deuffhard et. al. (1989) presented a new adaptive hierarchical finite element code KASKADE, which applied to linear scalar second-order 2D elliptic problems with general domains. It was shown that the approach permitted a flexible balance among iterative solver, local error estimator, and local mesh refinement device. This approach did not use partitioned modelling.

Teresco et. al. (2000) described the design and implementation of parallel mesh systems within an adaptive framework. The hierarchical model represented heterogeneous processor and network speeds, and could be used to represent processes in any parallel computing environment. The hierarchy in this model consisted of machine/processor/partition, and it was based on the graph partitioning technique, which contrasts with physical domain partitioning as considered for the present work.

Li & Parashar (2004) presented hierarchical partitioning and load-balancing techniques for distributed structured adaptive mesh refinement (SAMR) applications. Their hierarchical partitioning algorithm partitioned the computational domains into subdomains and assigned them to hierarchical processor groups. Like the previous models, this method is also based on graph partitioning theory.

Whilst great progress has been made towards successful parallelisation by the use of graph partitioning theories based on load balancing, many challenges still remain due to some misconceptions and deficiencies within these theories [Hendrickson, 2000]. Some of the issues associated with load balancing techniques which are relevant and have been addressed in the present work are listed below:

1. Communication congestion: When many messages simultaneously compete for network resources, only those that use independent paths through the network can proceed at the same time while others must take turns. Thus, the communication pattern between processes can significantly affect the efficiency.

2. Adaptability: When an adaptive calculation requires re-meshing, the dynamic graph partitioning techniques invoke the repartitioners which results in a large amount of data being transferred between partitions. In addition, the repartitioners also compete with the main application for computing resources.
3. Graph partitioning techniques cannot easily deal with differently dimensioned partitions or allow different integration schemes therein. Moreover, in the context of nonlinear analysis, graph partitioning does not consider the potentially very different computational demands of elements of different types of element (e.g. elastic vs. elasto-plastic, 1D vs. 3D, etc.) used in the same structural model.

The hierarchical partitioning method developed in the present work is based on the physical partitioning of structures under user control. This method can avoid communication congestion by providing a separate communication channel for each parent to communicate with its children. Furthermore, dynamic remeshing does not affect the physical partitioning scheme. Finally, it is still possible with the proposed method to utilise an optional automatic partitioning tool to be used as a pre-processor.

2.4 Dimensional Coupling

Modelling of typical structures as an assembly of 3D continuum finite elements is impractical owing to the computational costs involved [Spacone & El-Tawil, 2004]. The most practical approach can be to combine the precision and generality of 3D continuum elements with the computational efficiency of 1D elements [Mata, Barbat & Oller, 2009].

Ziyaeifar & Noguchi (2000) presented a refined model for beam elements and beam-column joints. A new beam-solid transition element was introduced. To address the two-dimensional state of stress and strain in the high shear zone in beam and columns, a continuum approach was adopted. A refinement in first-order bending theory was made to relax the condition of planar cross-sections, and additional flexibility to the displacement field of the element was achieved by allowing cross-sectional warping. This allowed higher accuracy in strain field approximation.

A very significant development towards dimensional coupling was the presentation of mixed-dimensional coupling in finite-elements by McCune et al. (2000). They suggested a scheme for establishing compatibility and equilibrium at the dimensional interface when reduced or lower-dimensional element types are combined with higher-dimensional element types in a single model. For this purpose, the authors partitioned the model in two regions for meshing with elements of a given dimension. The coupling between dimensions was based on the work done by stresses in each part at the interface between dimensions. An assumed variation of stresses over the plate thickness or beam section was introduced for the reduced-dimensional part. Perturbation methods were used to derive the form of this variation. Four types of dimensional coupling were presented by the authors, including 2D-1D in plane, 3D solid-2D plate, general 2D-1D, and 3D-2D-1D coupling. In addition, error estimation in 2D-1D and 3D-2D mixed-dimensional models was proposed.

The dimensional coupling approach by McCune et al. (2000) was followed and generalised by Shim et al. (2002). Depending on the ability to determine the stress distribution due to any load at the interface, 1D-3D, 1D-2D or 2D-3D coupling could be achieved for transitions with arbitrary shapes or cross-sections.

Both the McCune, Armstrong & Robinson (2000) and Shim, Monaghan & Robinson (2002) approaches of mixed-dimensional coupling were developed for linear finite element analysis considering structures with isotropic homogenous materials. It is also noted that mixed-dimensional coupling has obvious accuracy shortcomings, since there is the possibility that spurious stresses may show up at the dimensional interface [McCune, Armstrong & Robinson, 2000; Shim, Monaghan & Armstrong, 2002].

Mata, Barbat & Oller (2009) presented a two-scale approach with local non-prismatic parts. This approach uses 1D elements at the global level. At the local level, however, if complex geometry appears, the corresponding elements are analysed using full 3D models. This approach uses the kinematics hypothesis of the 1D model on surface-interfaces of the 3D model for dimensional coupling between the two.

A new dimensional coupling approach based on a master-slave approach has been developed in this work. This dimensional coupling, which is presented in Chapter 5, is well

suited for use with the partitioning methods based on dual partition super elements presented in Chapter 3.

2.5 Parallelisation Techniques

With the advancement of computing technologies, it is possible to perform a huge amount of calculations in much less time. There are quite a few supercomputers that can perform calculations to the order of trillions of floating point operations per second (Tflops). The term 'High Performance Computing (HPC)' is generally associated with these supercomputers. This type of computing is largely used for scientific research.

The installation of a supercomputer is expensive and cannot be a solution for practical problems, for example performing structural analysis of a building. These types of problems are generally solved using workstations, and although the capacity of such workstations is increasing rapidly, the complexity of problems is also increasing at the same pace with the physical development in the world. In the field of structural engineering, for example, to provide better and safer structures there is a need to model the influence of complex phenomena such as fires, blasts, earthquakes, and other natural/manmade disasters.

Another HPC approach involves the use of grid computing, which employs multiple computer clusters. A computer cluster is a group of processing cores connected by means of a network, which work together and in parallel on a specific problem, thus increasing the overall computation ability. A group of workstations connected through a local area network is a very good example of a computer cluster. Such a cluster will have considerably more capacity compared to that of single workstations.

2.5.1 Computer Cluster Networks

The processing cores forming a computer cluster can be connected in several ways (Figures 2—8 to 2—12):

- Bus: all the processing cores are connected to a common bus or buses
- Switching network: processing cores are connected to each other through routing switches, similar to a telephone system

- Point to point: processing cores are connected directly to other processing cores. Mesh, torus, fat tree, and hypercube are a few examples of point to point networks.

Depending on the size of the area these networks cover, they are further classified into personal area network (PAN), local area network (LAN), campus area network (CAN), metropolitan area network (MAN), or wide area network (WAN). These networks can be wired or wireless.

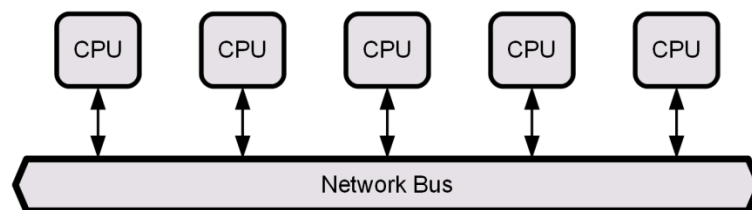


Figure 2-8: Bus Network

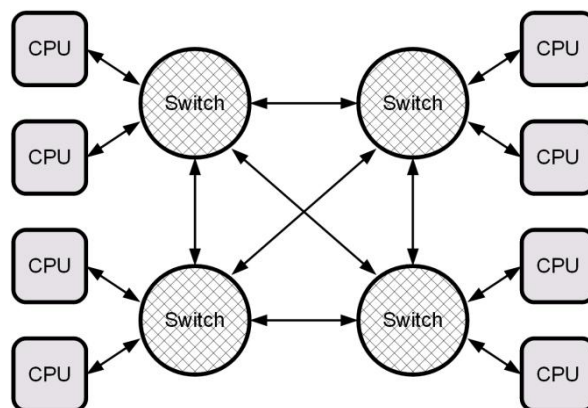


Figure 2-9: Switching Network

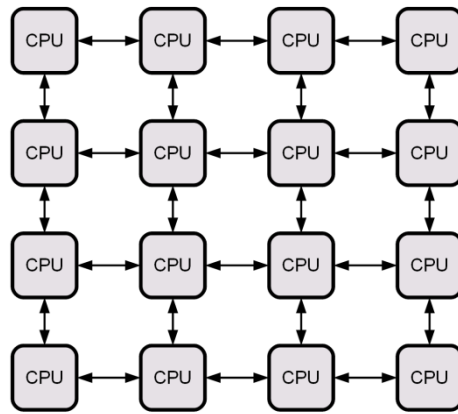


Figure 2-10: 2D Mesh Network

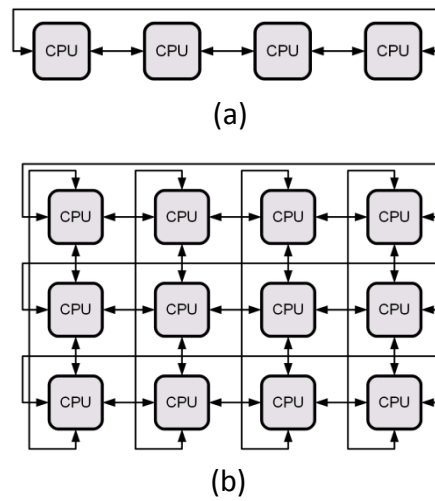


Figure 2-11: Torus Network (a) 1D Torus (b) 2D Torus

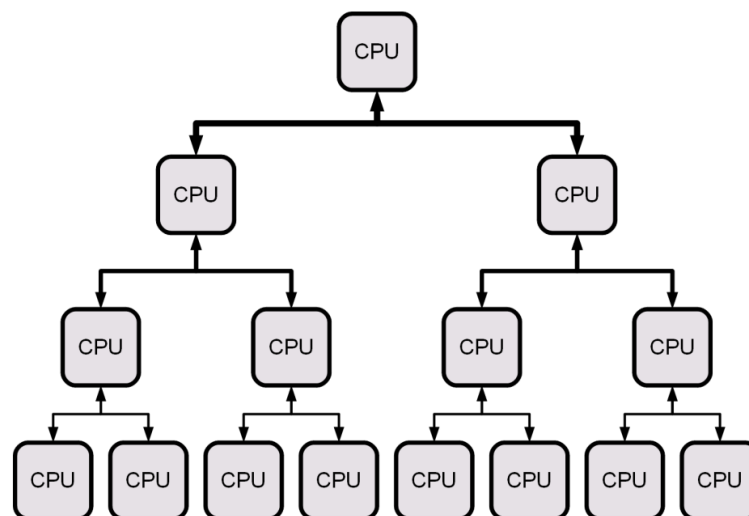


Figure 2-12: Fat Tree Network

The performance of a computer cluster depends on the efficiency of the network by which the processing cores are connected. Computer clusters and networks form the hardware side of the problem of HPC. On the software side, programs need to incorporate special instructions to utilise the additional capability gained from computer clusters. This can be done through Parallel Programming, which is briefly described in the following section.

2.5.2 Parallel Programming

Large problems can often be divided into smaller ones, which can then be solved concurrently in parallel using more than one processing cores. The programs developed for this purpose are mostly written in standard high-level languages, with additional calls to a parallel library and/or the use of compiler directives to facilitate the parallelism. Parallel programming forms an integral part of High Performance Computing (HPC).

The most commonly used language in HPC is FORTRAN, followed by C. The parallel programs written using any of these languages are then compiled with the standard compilers for the platform where they are to be run, thus providing the vital property of portability to these programs. A compiler optimised for the architecture is ideal to use.

The parallel decomposition of a large problem into several smaller ones is based on the assumption that the smaller problem can be solved more quickly than the larger one using the same processing capability. There can be two different ways to problem decomposition [Smith et al., 1996; Toselli & Widlund, 2005]. Domain decomposition means that the data is partitioned into subsets and each core is made responsible for the calculations on one or more of these subsets. Functional decomposition, on the other hand, means that the problem is split into sub-problems and each core is made responsible for solving one or more of these sub-problems.

Parallel programs can run on two different kinds of processing clusters, namely shared memory and distributed memory systems. In the case of a shared memory system, every core has direct access to the memory of every other core in the system. On the other hand, in a distributed memory system, a processing core holds all variables in local memory.

Different cores need to communicate with each other to send and receive data, which is done through message passing.

There are two main approaches to parallelisation with message passing [Lin & Snyder, 2008]:

- The same program is run on all the processing cores with conditional control within the program syntax that makes the cores able to perform their share of sub-problems.
- A program running on one of the cores can act as a parent starting different programs or identical copies of another program on other cores as its children. It can then decompose the problem into sub-problems and send each child its share of sub-problems and receive back the results.

The biggest issue facing parallel programming is portability. Another significant factor is that in a distributed memory environment, higher level abstractions are based on lower level message passing routines. These issues make it necessary to standardise message passing.

2.5.3 Message Passing Interface (MPI) Library

The most widely used standard in parallel programming is the Message Passing Interface (MPI) library, which provides a specification for the movement of data from the address space of one process to that of another through cooperative operations. It works with both shared and distributed memory systems, and its operations are expressed as functions, subroutines, or methods, according to the appropriate language bindings, which for C, C++, Fortran-77, and Fortran-90, are part of the MPI standard. The standard has been defined through an open process by a community of parallel computing vendors, computing scientists, and application developers [MPI Forum, 2009].

Historically, several message passing libraries have been used for parallel programming. The best known among them are the Parallel Virtual Machine (PVM) and the MPI. The former was once popular, however, at present it is mainly superseded by the latter, which has the following advantages for distributed memory systems [MPI Forum, 2009]:

- The MPI code is portable i.e. its implementations exist for most parallel platforms.
- Free, portable, downloadable versions of MPI are available.
- Considerable effort has been put into optimising the performance of the library and tuning it to specific hardware platforms and interconnects; this development is ongoing.
- The standard itself is also continually being refined and updated.

In terms of application, considering the solution of a linear system of equations that arises for example from discretisation in finite element analysis, this involves the following three main operations [Magoules & Roux, 2007]:

- Matrix-vector product
- Vector dot product
- $axpy$ (scalar a times vector x plus vector y)

To carry out the above operations in parallel, processes have to communicate with each other. The MPI library provides a wide range of communication subroutines; however, for the above operations, only a few of those subroutines are needed. The communication between processes can either be point to point, where a process either receives data from or sends data to another process, or it can be collective. Point to point communication can either be *blocking* (i.e. the sending process will not return from the send call before the receiving process enters the receive call) or *non-blocking* (i.e. the sending process will proceed regardless of whether the receiving process enters the call). In collective communication, a process sends or receives data from a group of processes or the entire group sends or receives data from each other. This type of communication may involve sending copies of the same data to the group, distributing an array among processes, collecting parts of an array from processes and assembling it. It may also involve performing algebraic operations such as collecting the contribution from each process, adding up, and sending the result back to a single process or the entire group. It is worth noting here that collective communications are always blocking and can be used for synchronisation between processes.

The default group in MPI includes every process, and the communicator associated with it is called `MPI_COMM_WORLD`. Any collective communication using this communicator involves all processes; however, if it is required to carry out collective communication between a certain number of processes, it is possible to create new groups and assign them different communicators. The collective communication in the newly created groups involves processes in that group only, a feature which is well suited for hierarchical parallelisation.

2.5.4 MPI/OpenMP Hybrid Parallelisation

While MPI is ideal for distributed memory systems, the OpenMP Application Program Interface [OpenMP Architecture Review Board, 2011], sometimes also referred to as Multi-threading, is designed for shared memory systems only and is a favourable option in that environment as it performs better than MPI. The OpenMP API is easier to program and debug and also supports gradual parallelisation while the code can still be run as a sequential program.

To get the advantages of both the MPI and OpenMP, hybrid parallelisation may be carried out, in which the processing cores within a node use OpenMP to run the program in parallel while they use MPI to communicate with processes on other nodes. This approach makes good use of shared memory systems by avoiding the extra communication overhead within the node.

Domain Decomposition

This chapter presents a new domain decomposition approach for nonlinear structural analysis based on a novel ‘dual partition super element’ concept.

3.1 Introduction

The term ‘domain decomposition’ has been often used in the literature to mean coupling of more than one sub-domain by the introduction of a formulation at the domain interfaces, most commonly using Lagrange multipliers. In some cases, however, the use of Lagrange multipliers can be computationally expensive as well as overly complicated. Another approach used in domain decomposition is the interface displacement frame method, where a displacement field is defined at the interface that links the subdomains with a standard stiffness matrix procedure after all the internal variables are eliminated. In general, domain decomposition implies partitioning of the stiffness matrix associated with the overall response into blocks and the communication of interface entities as elaborated in Section 2.3.

A new method has been developed in this work for structural domain decomposition, which facilitates scalable parallel processing over distributed memory networks, thus overcoming memory bottlenecks for large scale problems. The new method introduces the concept of dual partition super elements, where a child partition is wrapped at the interface boundary by a super element and is represented in the parent structure by a dual super element. With the parent structure and child partition attached to separate processes, one super element is used in the parent process and the dual super element is used as a wrapper around the partitioned boundary in the child process. It is worth noting here that this method can accommodate multiple child partitions for a specific parent structure. Furthermore, the method is hierarchic, in the sense that a child partition can be further partitioned similar to a parent structure, and it can thus be mapped to hierarchic parallel processing architectures. This chapter focuses on basic domain decomposition with one level of partitioning, while hierarchic partitioning is presented in detail in Chapter 6.

In addition to the benefits of traditional partitioning approaches, the current approach provides the facility for using mixed methods such as implicit-explicit integration schemes [Belytschko et al., 1979; Hughes et al., 1983] as well as dimensional coupling [McCune et al., 2000; Shim et al., 2002] between partitions. A further important benefit of the developed approach is that it allows the recovery of child partition forces and condensed tangent stiffness matrix at the interface boundary relatively easily via the dual super element, which can be achieved in a frontal solution method by placing the child super element at the end of the element ordering list. When all the other elements of the partition are assembled and the associated interior freedoms are eliminated, the remaining equilibrium equations contain the forces and condensed tangent stiffness matrix for the super element only, which can be communicated to the dual super element in the parent structure. The parent process treats the partition super element similar to other finite elements, providing displacements that can be communicated to child processes of the dual super elements and receiving interface forces and condensed tangent stiffness matrices in return. Thus, this approach is effectively identical in performance to the monolithic approach with high speed-ups due to parallel processing and ease of modelling.

In the following sections, the developed partitioning approach is presented in detail, and the abovementioned benefits are demonstrated through illustrative examples.

3.2 Partitioning with Dual Super elements

To elaborate the developed partitioning approach, a structural domain (Figure 3-1) is considered subject to arbitrary restraint and loading conditions, and which is to be partitioned into three subdomains by cutting out two partitions along the dotted lines.

For the sake of simplicity, it is assumed that each node has 2 degrees of freedom; hence this domain with 40 nodes has a tangent stiffness matrix of size 80x80. In nonlinear finite element analysis, the tangent stiffness matrix relates the infinitesimal increments of resistance and displacements. Therefore, for the overall structural domain under consideration:

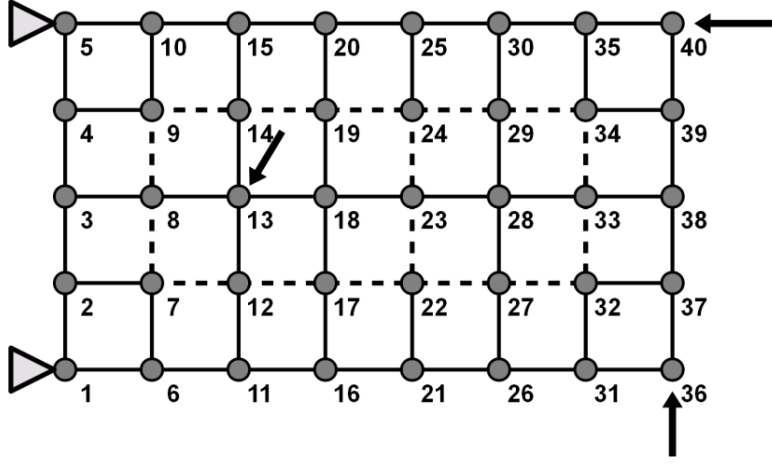


Figure 3-1: An illustrative structural domain

$$\delta R_{\{80\}} = K_{[80 \times 80]} \delta d_{\{80\}} \quad 3-1$$

where δd is infinitesimal increment of displacement, K is the tangent stiffness matrix, and δR 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:

$$G_{\{80\}} = R_{\{80\}} - P_{\{80\}} \quad 3-2$$

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

$$G_{\{80\}} \cong O_{\{80\}} \quad 3-3$$

where O is a zero vector.

A non-zero G on the other hand, indicates lack of equilibrium and necessitates iterative correction of the displacements, which is obtained (for prescribed loads) as:

$$\delta d_{\{80\}} = K_{[80 \times 80]}^{-1} (-G_{\{80\}}) \quad 3-4$$

leading to:

$$d_{\{80\}} = d_{0\{80\}} + \delta d_{\{80\}} \quad 3-5$$

where d_0 and d are the previous and current iterative displacement vectors, respectively. The new resistance forces vector R is evaluated again for the current d , and the iterative process is repeated until Equation 3—3 is satisfied.

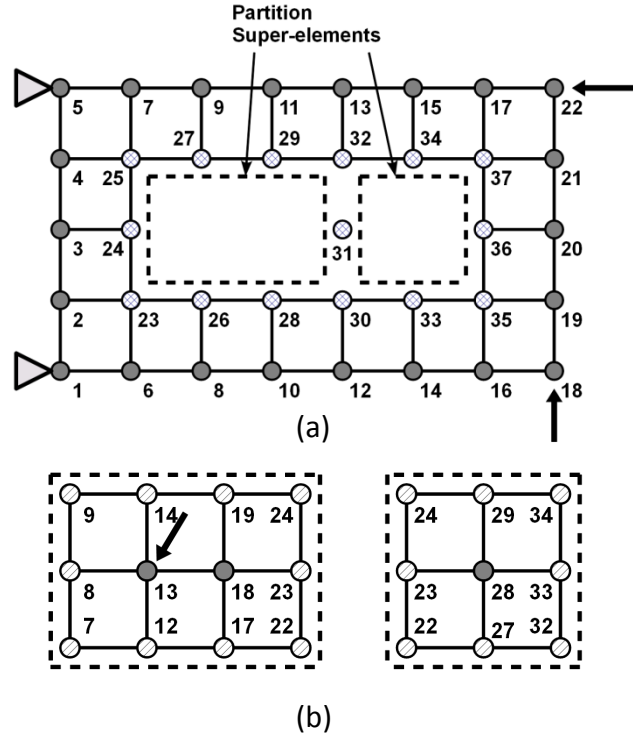
It is noted here that Equation 3—4 becomes different when the load at the end of the step is unknown, such as in displacement or arc length control as elaborated in Section 3.2.4.

3.2.1 Partitioning Method – Solution at Parent Level

With the structural domain partitioned, let the three new subdomains be named as Ω_0 , Ω_1 and Ω_2 , where the zero-indexed partition represents the parent structure, while the rest represent children partitions. The three partitions are illustrated in Figure 3-2.

After partitioning, the parent structure has 37 nodes, 15 of which are connected to two partition super elements. For the child partitions, partition 1 has 12 nodes with 10 nodes at the boundary connected to the dual partition super element, while partition 2 has 9 nodes with 8 at the partitioned boundary. It is worth noting here that the nodes in the parent structure that are connected to the partition super elements may or may not be connected to conventional elements, as is the case in the example under consideration where 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 structure is still complete with the response of the removed partitions being represented by the partition super elements. The stiffness matrix for the parent structure can be assembled in the normal way by first accounting for the contribution from the conventional elements in the parent structure:

$$K_{[74,74]} = K_{\Omega_0[74,74]} \quad 3-6$$



- Internal nodes
- ⊗ Nodes connected to two partition super elements
- Boundary nodes connected to dual super element

Figure 3-2: Partitioned structure (a) parent structure (b) child partitions

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

$$K_{[45:64,45:64]} = K_{[45:64,45:64]} + K_{\Omega_1[20,20]}^c \quad 3-7$$

where $K_{\Omega_1[20,20]}^c$ is the condensed stiffness matrix of child partition 1 after the internal nodes have been eliminated. As discussed in the next section, the condensed stiffness matrix is recovered easily in a frontal solution method as the 'Grandpa' associated with the remaining active freedoms 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:

$$K_{[59:74,59:74]} = K_{[59:74,59:74]} + K_{\Omega_2[16,16]}^c \quad 3-8$$

It should be noted 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 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 as:

$$\delta R_{\{74\}} = K_{[74,74]} \delta d_{\{74\}} \quad 3-9$$

where δd and δR refer here to finite iterative increments of displacement and resistance, 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 $R_{\Omega_m}^c$ are considered, as discussed in the next section. The out-of-balance at the parent level is obtained as:

$$G_{\{74\}} = R_{\{74\}} - P_{\{74\}} \quad 3-10$$

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 calculated as:

$$\delta d_{\{74\}} = K_{[74 \times 74]}^{-1} \left(-G_{\{74\}} \right) \quad 3-11$$

The correction to the displacements for the nodes connected to the partition super elements are provided to the dual partition super elements:

$$\delta d_{\Omega_1 \{20\}}^c = \delta d_{\{45;64\}} \quad 3-12$$

and:

$$\delta d_{\Omega_2 \{16\}}^c = \delta d_{\{59;74\}} \quad 3-13$$

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 at both the parent and child levels, as elaborated in Section 3.2.3.

3.2.2 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 [Felippa, 1975, Reid, 1971] or the frontal method [Irons, 1970; Sloan & Randolph, 1983, Izzuddin, 1991]. At the child level, however, there is a particular benefit in using the frontal method, which is based on optimal element ordering for minimum front width [Irons, 1970; Sloan & Randolph, 1983]. 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, as discussed later in this section.

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 linearised equations for the example under consideration can be represented as:

$$\begin{bmatrix} K_{1,1} & K_{1,2} & \cdots & K_{1,n} \\ K_{2,1} & K_{2,2} & \cdots & K_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ K_{n,1} & K_{n,2} & \cdots & K_{n,n} \end{bmatrix} \begin{Bmatrix} \partial d_1 \\ \partial d_2 \\ \vdots \\ \partial d_n \end{Bmatrix} = - \begin{Bmatrix} G_1 \\ G_2 \\ \vdots \\ G_n \end{Bmatrix} \quad 3-14$$

where n depends on the partition size.

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

$$\left[\begin{array}{cccc|c} K_{1,1} & K_{1,2} & \cdots & K_{1,n} & G_1 \\ K_{2,1} & K_{2,2} & \cdots & K_{2,n} & G_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ K_{n,1} & K_{n,2} & \cdots & K_{n,n} & G_n \end{array} \right] \begin{Bmatrix} \partial d_1 \\ \partial d_2 \\ \vdots \\ \partial d_n \end{Bmatrix} \quad 3-15$$

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:

$$\left[\begin{array}{cccc|c} K'_{1,1} & K'_{1,2} & \cdots & K'_{1,n} & G'_1 \\ 0 & K'_{2,2} & \cdots & K'_{2,n} & G'_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & K'_{n,n} & G'_n \end{array} \right] \quad 3-16$$

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

$$\delta d_n = \frac{-G'_n}{K'_{n,n}} \quad 3-17$$

$$\delta d_i = \frac{1}{K'_{i,i}} \left(-G'_i - \sum_{j=i+1}^n K'_{i,j} \delta d_j \right) \quad 3-18$$

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 structure. 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 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:

$$\left[\begin{array}{ccccccc|c} K'_{1,1} & K'_{1,2} & \cdots & \cdots & \cdots & \cdots & K'_{1,n} & G'_1 \\ 0 & K'_{2,2} & \cdots & \cdots & \cdots & \cdots & K'_{2,n} & G'_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & K'_{n-h+1,n-h+1} & K'_{n-h+1,n-h+2} & \cdots & K'_{n-h+1,n} & G'_{n-h+1} \\ \vdots & \vdots & \cdots & K'_{n-h+2,n-h+1} & K'_{n-h+2,n-h+2} & \cdots & K'_{n-h+2,n} & G'_{n-h+2} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & K'_{n,n-h+1} & K'_{n,n-h+2} & \cdots & K'_{n,n} & G'_n \end{array} \right] \quad 3-19$$

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 $K_{\Omega_m}^c$ for child partition m , while the associated right hand side vector $G'_{\{n-h+1:n\}}$ is returned as the corresponding condensed resistance forces $R_{\Omega_m}^c$.

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, as discussed earlier. 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 $K_{\Omega_m}^c$ with $R_{\Omega_m}^c$ to the parent and then receiving $\delta d_{\Omega_m}^c$ from the parent following

solution at the parent level. Once $\delta d_{\Omega_m}^c$ 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 Equation 3-18 using the augmented matrix of Equation 3-19 to establish the remaining iterative displacements starting from freedom $n-h$ to freedom 1.

3.2.3 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 λ , nonlinear analysis may be undertaken using load control, for which case λ 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 λ 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 undertaken with due consideration of the out-of-balance at the parent and child levels. There are different measures of convergence which can be used, such as the norm of the out-of-balance G , the norm of iterative displacements δd or the scalar product of G with δd . Convergence would then be considered to have been achieved when one or more of such measures is below a predefined tolerance ε . In this respect, convergence can be checked separately at the parent and child levels, with overall convergence required at all levels, as expressed by:

$$f(G_{\{1:n-h\}}, \delta d_{\{1:n-h\}}) \leq \varepsilon \quad 3-20$$

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.

3.2.4 Displacement Control

For static proportional loading, displacement control is typically applied to trace the equilibrium path following buckling, the formation of plastic mechanism or the initiation of softening. The simplest variant of displacement control [Izzuddin, 1991] prescribes over the current step an increment Δd_j for a specific freedom j , which can be at the parent or child levels. In return, the load factor λ 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:

$$\delta d = K^{-1}(-G + \delta\lambda P^o) \quad 3-21$$

where P^o is a vector representing the nominal (unscaled) 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 could then be obtained using the proposed partitioned approach using the same process of forward elimination followed by backward substitution, as already discussed in Sections 3.2.1 and 3.2.2, with:

$$\delta d^G = K^{-1}(-G) \quad 3-22$$

$$d^P = K^{-1}(P^o) \quad 3-23$$

$$\delta d = \delta d^G + \delta \lambda d^P \quad 3-24$$

Given that the increment of the controlled displacement increment δd_j is either the prescribed value Δd_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 to determine $\delta \lambda$ and hence δd .

A more sophisticated variant of displacement control is the arc-length method [Crisfield, 1991], 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 δd^G and d^P , which are still obtained as above.

3.3 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 processors with distributed memory. The following investigation aims at demonstrating through simple examples the conditions under which significant computational savings may be expected.

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 rarely decreases 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.

3.3.1 4×4 Grid

A basic mesh of 16 2D elements with 4 nodes each connected in the form of a 4×4 grid, as shown in Figure 3-3, is considered. For the sake of simplicity, it is assumed that there is only one degree of freedom per node. 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.

4	5	12	13
3	6	11	14
2	7	10	15
1	8	9	16

Figure 3-3: A mesh of 16 2D elements

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 the following expression:

$$MOps = \frac{(n-1)n(n+1)}{3} \quad 3-25$$

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 3-3, 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 3-1. 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 efficiency saving of approximately 86% demonstrates the superiority of the frontal method for the solution of a system of algebraic equations arising from finite element discretisation.

Reduction operation	Size of Grandpa before reduction	Size of Grandpa after reduction	Nodes eliminated	Multiplication operations
1	4x4	3x3	1	12
2	5x5	4x4	1	20
3	6x6	5x5	1	30
4	7x7	5x5	2	72
5	7x7	6x6	1	42
6	7x7	6x6	1	42
7	7x7	6x6	1	42
8	7x7	5x5	2	72
9	7x7	6x6	1	42
10	7x7	6x6	1	42
11	7x7	6x6	1	42
12	7x7	5x5	2	72
13	7x7	5x5	2	72
14	6x6	4x4	2	50
15	5x5	3x3	2	32
16	4x4	0	4	20
Total			25	704

Table 3-1: Reduction operations for a 4×4 grid using frontal method

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.

In the first case, the mesh is partitioned in 2 halves, each consisting of 2×4 element grids, as shown in Figure 3-4. 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 3.2.1 and 3.2.2. For solving each partition of 2×4 elements, 8 reduction operations are required (Table 3-2). Using the assembly order shown in Figure 3-4, 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 is 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.

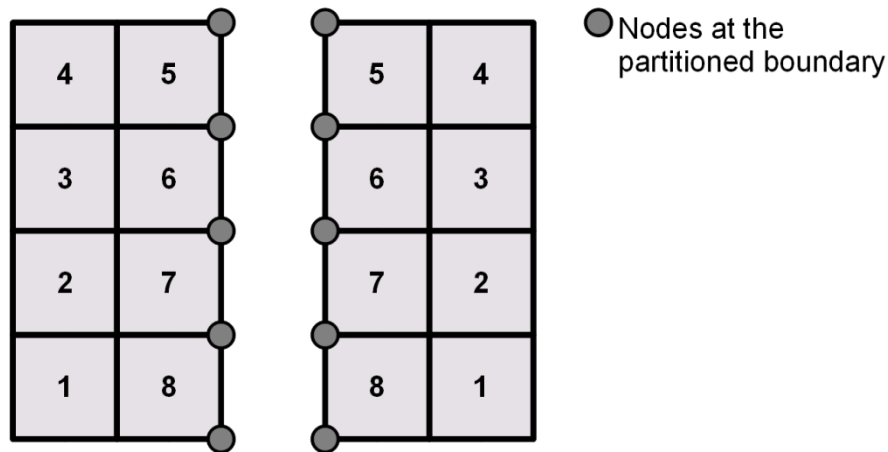


Figure 3-4: Problem decomposed into 2 partitions of 8 elements each

Reduction Operation	Size of Grandpa Before Reduction	Size of Grandpa After Reduction	Nodes Eliminated	Multiplication Operations
Operations at Partition Level				
1	4x4	3x3	1	12
2	5x5	4x4	1	20
3	6x6	5x5	1	30
4	7x7	5x5	2	72
5	7x7	6x6	1	42
6	7x7	6x6	1	42
7	7x7	6x6	1	42
8	7x7	5x5	2	72
Total			10	332
Operations at Global Level				
1	5x5	0	5	40

Table 3-2: Reduction operations required for 2 partitions

As an alternative partitioning scenario, the mesh is divided into 4 quarters of 2x2 element meshes, as shown in Figure 3-5. 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 3-3. 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 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. 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 74% and 39% of the wall-clock times of the cases with 2 partitions and no partitions, respectively.

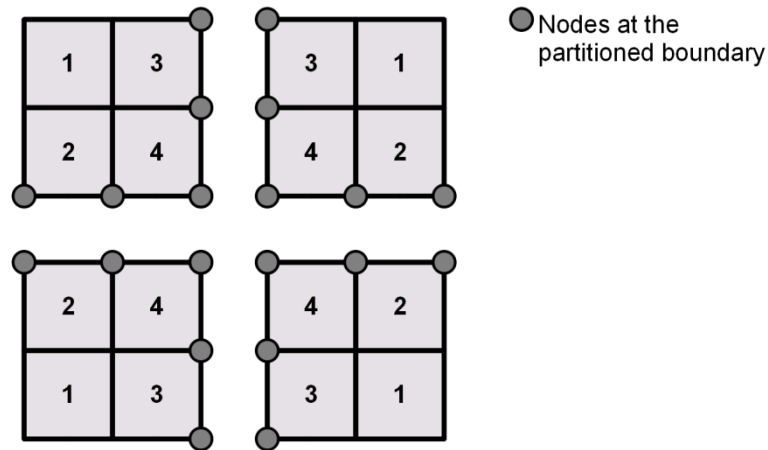


Figure 3-5: Problem decomposed into 4 partitions of 4 elements each

Reduction Operation	Size of Grandpa Before Reduction	Size of Grandpa After Reduction	Nodes Eliminated	Multiplication Operations
Operations at Partition Level				
1	4x4	3x3	1	12
2	5x5	4x4	1	20
3	6x6	5x5	1	30
4	6x6	5x5	1	30
Total			4	92
Operations at Global Level				
1	5x5	5x5	0	0
2	7x7	5x5	2	72
3	7x7	5x5	2	72
4	5x5	0	5	40
Total			9	184

Table 3-3: Reduction operations required for 4 partitions

The last scenario considered is partitioning the mesh into 16 parts each consisting of a single element, as shown in Figure 3-6. This option does not offer much advantage with regard to the forward elimination process because all of the original nodes except for 4 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 3-4. 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 63% and 120% more than that of the cases with 2 and 4 partitions, respectively.

Reduction Operation	Size of Grandpa Before Reduction	Size of Grandpa After Reduction	Nodes Eliminated	Multiplication Operations
Operations at Partition Level				
1	4x4	3x3	1	12
Total Multiplication Operations for Corner Partitions				12
Operations at Global Level				
1	3x3	3x3	0	0
2	5x5	4x4	1	20
3	6x6	5x5	1	30
4	6x6	5x5	1	30
5	7x7	6x6	1	42
6	7x7	6x6	1	42
7	7x7	6x6	1	42
8	7x7	5x5	2	72
9	7x7	6x6	1	42
10	7x7	6x6	1	42
11	7x7	6x6	1	42
12	7x7	5x5	2	72
13	6x6	5x5	1	30
14	6x6	4x4	2	50
15	5x5	3x3	2	32
16	3x3	0	3	8
Total Multiplication Operations at Global Level			21	596

Table 3-4: Reduction operations required for 16 partitions

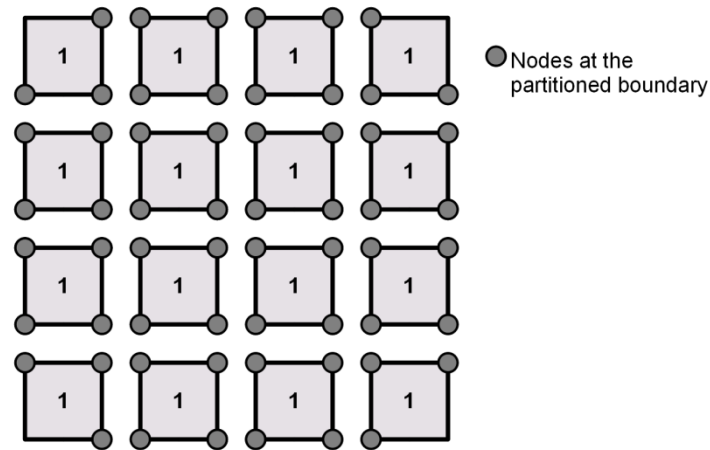


Figure 3-6: Problem decomposed into 16 partitions of 1 element each

A summary of the speedup, defined as the ratio of the wall-clock time without partitioning to that with partitioning, is provided in Table 3-5. 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. Further discussion of partitioning efficiency taking these factors into consideration is provided in Section 3.3.3.

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

Table 3-5: Summary of speed up gained

3.3.2 2×8 Strip

The same 16 2D elements used in the previous example are rearranged to constitute a structure in the shape of a strip as shown in Figure 3-7, where the computational efficiency is again illustrated by considering the full structure and then partitioning it into two.

If the assembly order is followed for the full structure as shown in Figure 3-7, there are 16 reduction operations required as detailed in Table 3-6. The total number of multiplication operations required in this case is 408.

2	3	6	7	10	11	14	15
1	4	5	8	9	12	13	16

Figure 3-7: The mesh of 16 2D elements in strip shape

If the strip is divided in two partitions, as shown in Figure 3-8, there are 8 reduction operations required for each partition, as detailed in Table 3-7. 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% of that of the full structure as against 53% when the 4x4 grid structure of the previous example was divided into two partitions.

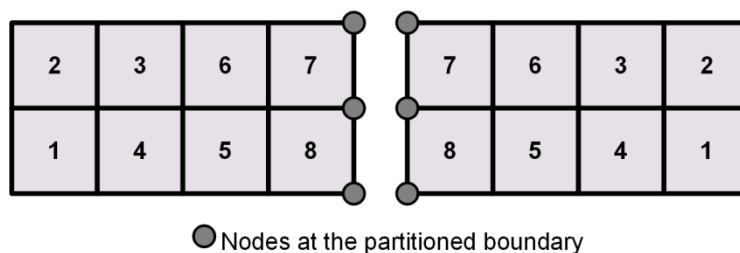


Figure 3-8: The strip partitioned in two

Reduction Operation	Size of Grandpa Before Reduction	Size of Grandpa After Reduction	Nodes Eliminated	Multiplication Operations
1	4x4	3x3	1	12
2	5x5	3x3	2	32
3	5x5	4x4	1	20
4	5x5	3x3	2	32
5	5x5	4x4	1	20
6	5x5	3x3	2	32
7	5x5	4x4	1	20
8	5x5	3x3	2	32
9	5x5	4x4	1	20
10	5x5	3x3	2	32
11	5x5	4x4	1	20
12	5x5	3x3	2	32
13	5x5	4x4	1	20
14	5x5	3x3	2	32
15	5x5	3x3	2	32
16	4x4	0	4	20
Total Number of Multiplication Operations for all Reductions				408

Table 3-6: Reduction operations for the strip of 16 2D elements

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 linearised system of equations and the inter-processor communication overhead.

No. of Reduction Operation	Size of Matrix Before Reduction	Size of Matrix After Reduction	Number of Nodes Eliminated	Multiplication Operations
Operations at Partition Level				
1	4x4	3x3	1	12
2	5x5	3x3	2	32
3	5x5	4x4	1	20
4	5x5	3x3	2	32
5	5x5	4x4	1	20
6	5x5	3x3	2	32
7	5x5	4x4	1	20
8	5x5	3x3	2	32
Total Multiplication Operations for the Partition				200
Operations at Global Level				
1	3x3	0	3	8

Table 3-7: Reduction operations required for the strip in 2 partitions

3.3.3 Remarks on Efficient Partitioning

It is well established that parallelisation over an increasing number of processors eventually becomes less effective with reducing speedup rates due to the inter-processor communication overhead [Xu and Hwang, 1996]. This fact aside, the above study has shown that even the solution of the linearised 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 [Amdahl, 1967], which states that the speedup of parallelisation is limited by the portion of the program that cannot be parallelised. In this respect, optimal partitioning is achieved when there is a balance between the number of interface freedoms and those internal to the partitions. However, if the computational demand is dominated by the evaluation of the element response, as opposed to the solution of equations or inter-processor communication, then fine partitioning would be expected to maximise speedup.

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, as illustrated in the case studies of Chapter 7, 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 allocated to the processor and scheduled for sequential processing using their frontal ordering at the parent level. This refinement, however, is outside the scope of the present work.

3.4 Implementation of Partitioning Method

The proposed partitioning approach has been implemented in ADAPTIC [Izzuddin, 1991], an advanced nonlinear structural analysis program developed at Imperial College London by Prof. 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 parallelisation, the shell script uses MPI commands 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 structure and serves as the ‘coordinator’, with the remainder serving as ‘partition processes’.

3.4.1 Process 0: The Coordinator

The process ranked 0 in the default MPI communicator is designated as the ‘coordinator’ and is responsible for processing the parent structure. 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’. The incremental solution procedure followed by the coordinator is illustrated in the flowchart of Figure 3-9 for static proportional under load control, with similar procedures used for displacement control and time-history static/dynamic loading.

Although the coordinator process is also the one that analyses the parent structure, it is not named ‘parent process’, since in the context of hierarchical partitioning, as introduced in Chapter 6, a partition process can be a parent process for other child partition processes. The present terminology is also consistent with the MPI terminology, where a parent process is one which starts other processes.

3.4.2 Partition Process

In the default MPI communicator, partition processes have a rank greater than zero and are each assigned a single partition. On start, each partition process reads the data for its partition, and then awaits instructions from the coordinator. As indicated in the previous section, the coordinator can send 7 instructions in total, which are described below for nonlinear static analysis under proportional loading using load control, and illustrated in Figure 3-9:

- 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, as discussed in Section 3.2.2, and sent back to the coordinator.

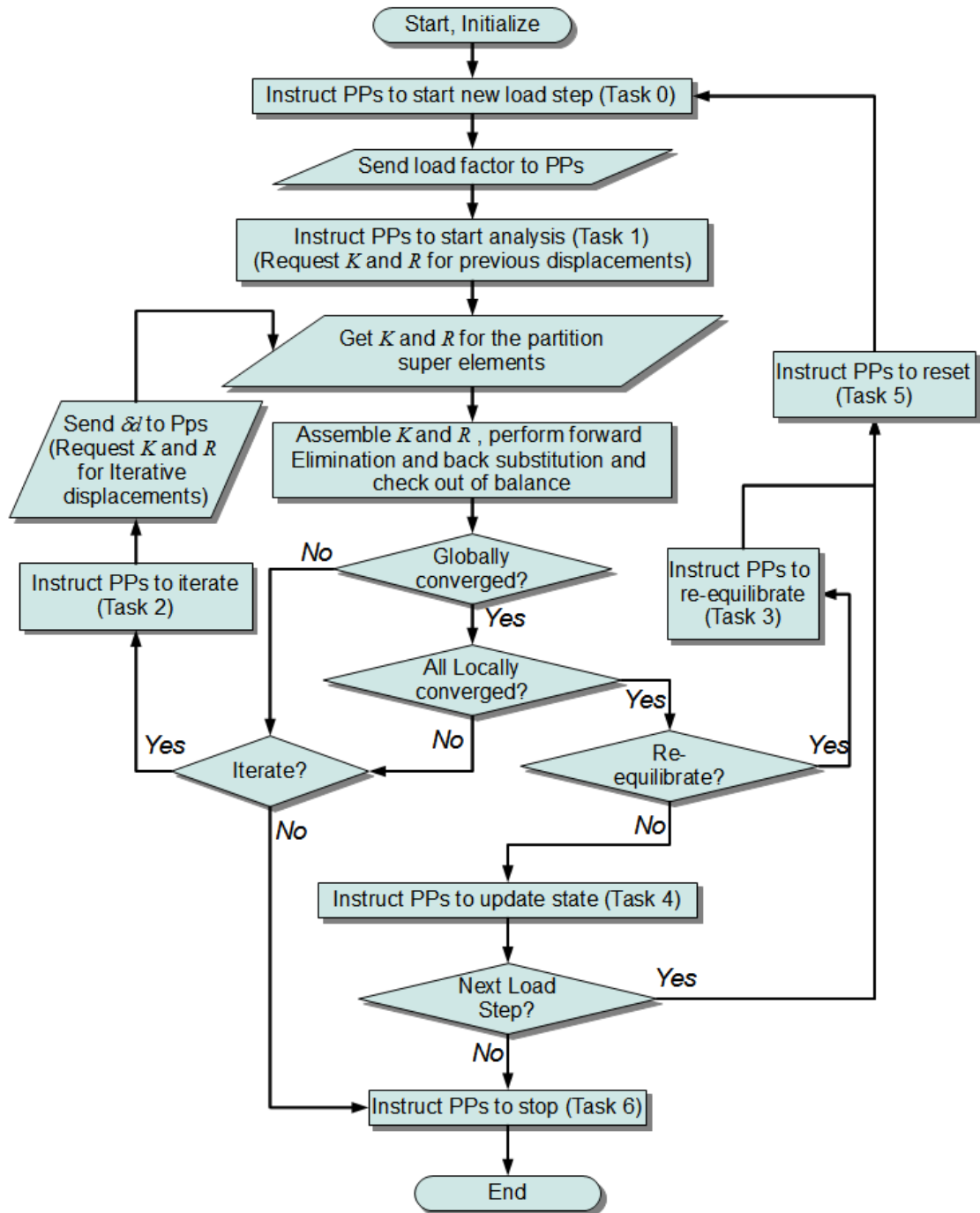


Figure 3-9: Flowchart of the coordinator process (PPs stands for partition processes)

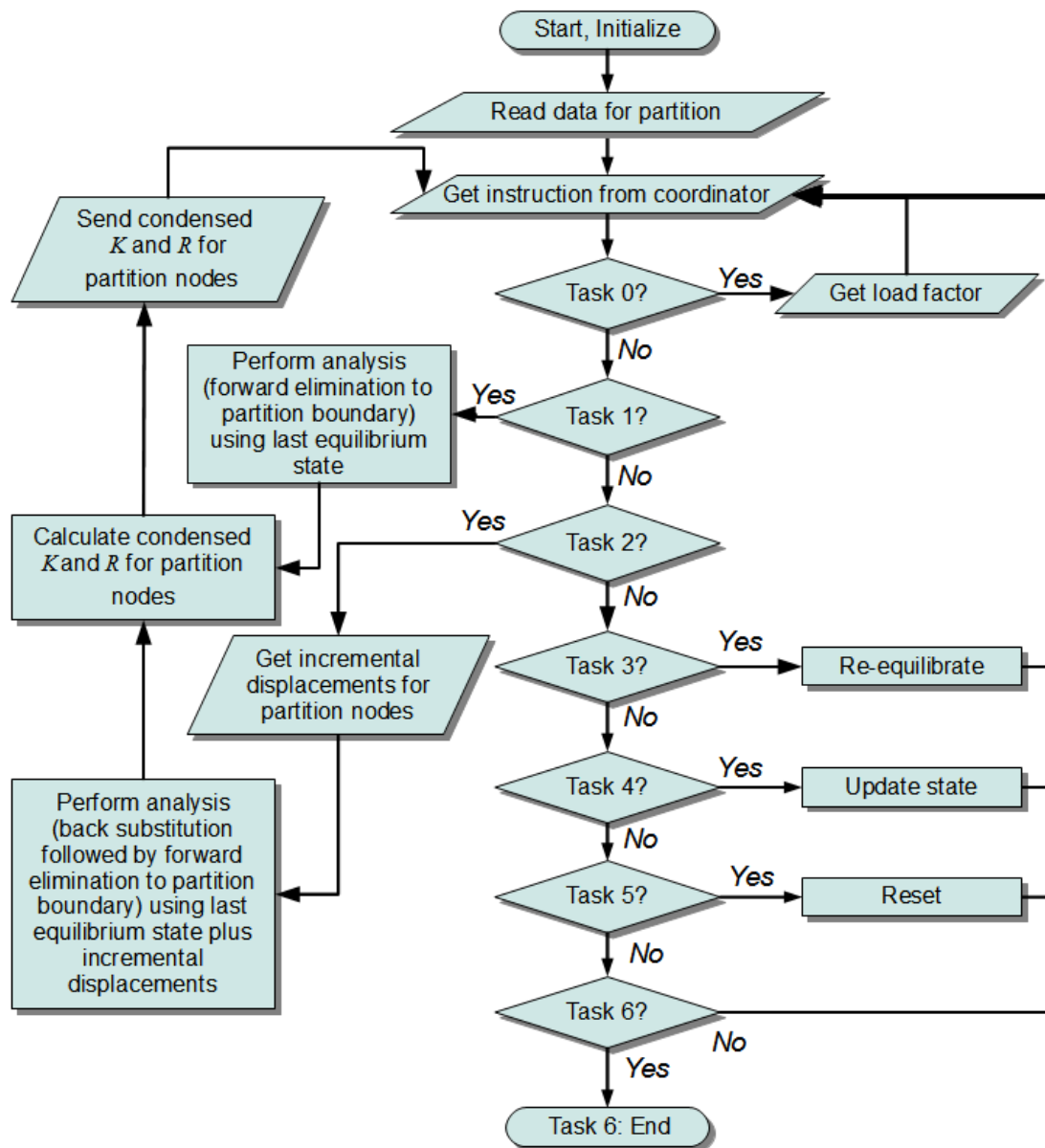


Figure 3-10: Flowchart of a partition process

- Task 2, Iterate: This task 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, as discussed in Section 3.2.2.

- Task 3, Re-equilibrate: After achieving overall convergence at parent and child levels, the coordinator can issue instruction to a partition process to check whether automatic mesh refinement is to be undertaken in the partition for the current step. In this case, the partition process returns a flag which indicates that the overall model should be re-equilibrated for the current load step.
- Task 4, Post Equilibrium: This task instruction is received after overall equilibrium has been achieved, in which case the partition process updates the equilibrium state and stores the results in a partition output file.
- Task 5, Reset All: During analysis, the coordinator can instruct the partition processes to reset the increments of displacement to zero, for example after convergence and update, or to initiate a new iterative procedure following lack of convergence.
- Task 6, Finish: This instruction terminates the partition process.

3.4.3 Input Data Syntax

For parallel partitioned analysis in ADAPTIC, one input file is used for each partition including the parent structure. The parent structure file is named in the usual manner with a '.dat' extension at the end (i.e. *basename.dat*), whereas the partition number is added to a '#' following the base name for all partition files (i.e. *basename#001.dat* for the data file of the first partition).

The format of the input files remains largely the same as that for ADAPTIC [Izzuddin, 2009]. The new partition super element used in parent structure is named 'part' and is defined in the same manner as regular elements. The main data that needs to be specified for each group of this element is the number of nodes on the partitioned boundary, while the actual nodes connected to individual elements are defined in the element connectivity module. The data syntax for this element is given below:

groups	
type.of.element =	part
grp.name	nodes
<token>	<int>
⋮	⋮

The data for the partition super element groups used in the parent structure for the example described in Section 3.2, Figure 3-2, is specific in the *basename.dat* file as follows:

groups	
type.of.element =	part
grp.name	nodes
grp1	10
grp2	8

grp1 and grp2 are arbitrary names used here to represent partition super elements for partition 1, which is connected to 10 nodes, and partition 2, which is connected to 8 nodes, respectively.

With reference to Figure 3-2, the nodal connectivity of the super elements in the parent structure is defined in the *basename.dat* file as follows:

element.connectivity		
elm.name	grp.name	nod.name(s)
1	grp1	23 24 25 26 27 28 29 30 31 32
2	grp2	30 31 32 33 34 35 36 37

The dual partition super element is defined as the partitioned boundary in the partition data files. A data module is added to the input files in order to define this boundary as follows:

partitioned.boundary
nod.name
<token>
< token>
⋮

where 'partitioned.boundary' is the module header followed by the node names belonging to the dual super element listed on separate lines.

For partition No. 1 in the example structure shown in Figure 3-2, this module is instantiated in the *basename#001.dat* file to:

partitioned.boundary

nod.name

7

8

9

12

14

17

19

22

23

24

Note that the order of node specification on the partitioned boundary in the child partition (dual super element) must be identical to that specific for the super element in the parent structure. Example data files for other partitioned problems, including files for the parent structure and partitions, are presented in Appendix A.

3.5 Verification

This section presents several examples employing ADAPTIC with the newly developed partitioning approach. The first 3 examples demonstrate the expected perfect match between the monolithic and partitioned methods of analysis along with the accuracy of the current implementation, while the last 3 examples focus on the speedup and ease of modelling that are realised. All the examples presented here employ geometric nonlinearity.

3.5.1 Lee's Frame

The first example used to demonstrate the accuracy the proposed partitioning approach is the well-known Lee's frame depicted in Figure 3-11. 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 3-12. 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 $\Delta\lambda = 0.1$, and 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 3-13, it is clear that there is an excellent match between the results of conventional monolithic analysis and the proposed partitioning method.

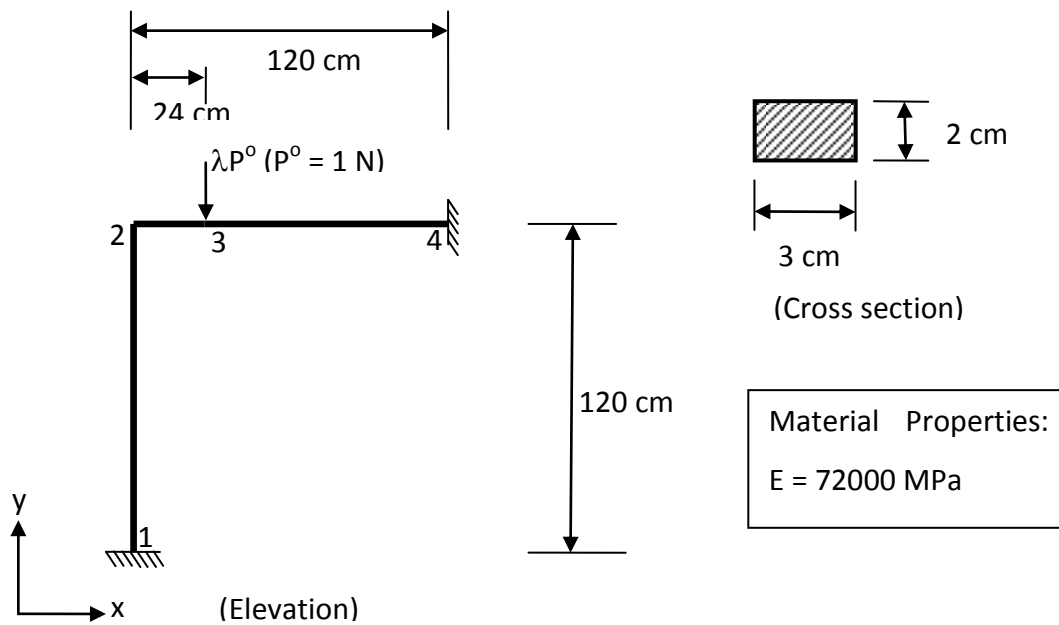


Figure 3-11 Lee's frame

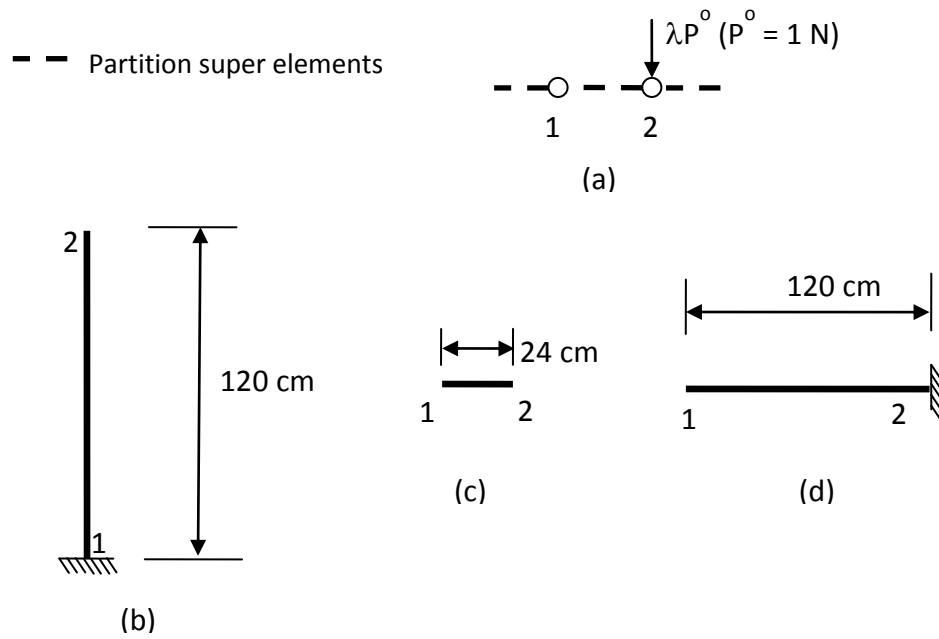


Figure 3-12: Lee's frame after partitioning (a) parent structure, (b-d) partitions

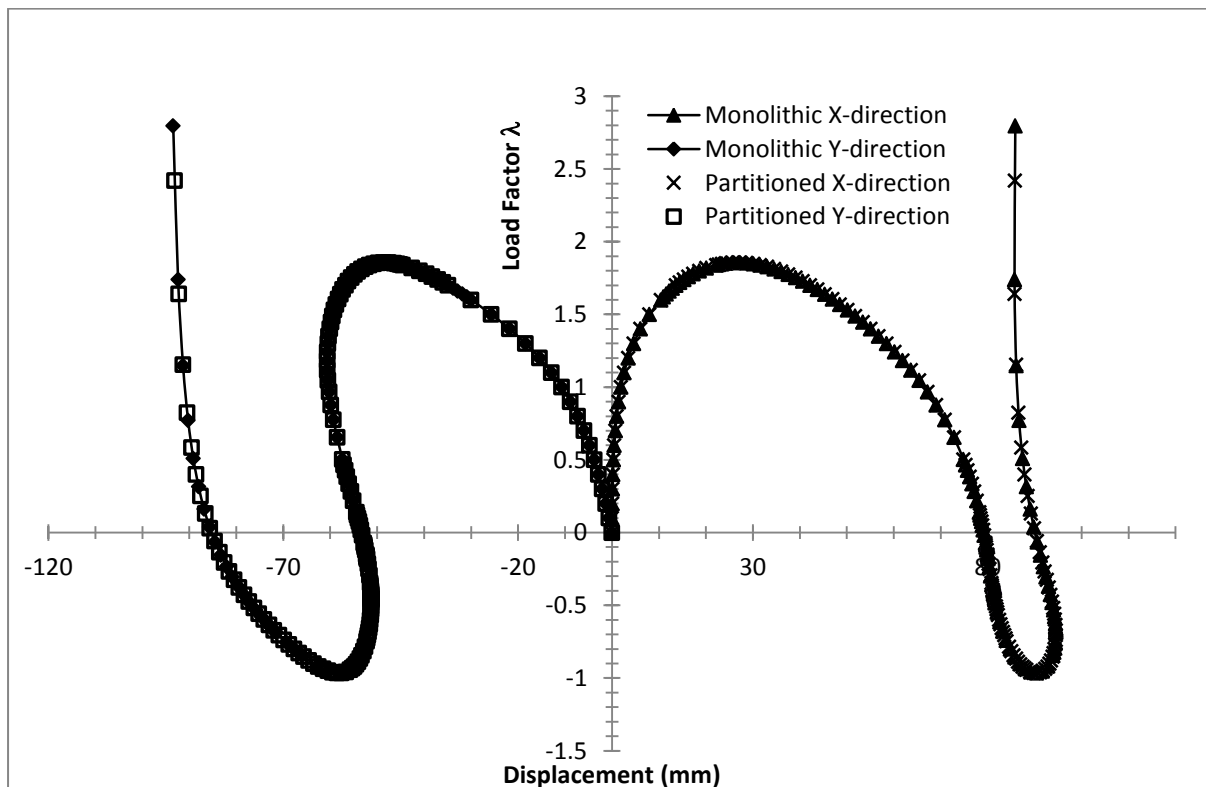


Figure 3-13: Comparison of displacement of node 3

3.5.2 Dynamic Analysis of Four-Storey Frame

The second example used for the verification of the accuracy the proposed partitioning approach is a four-storey frame, as depicted in Figure 3-14, which is subject to ground excitation with the earthquake signal shown in Figure 3-15. 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 [Izzuddin, 1991].

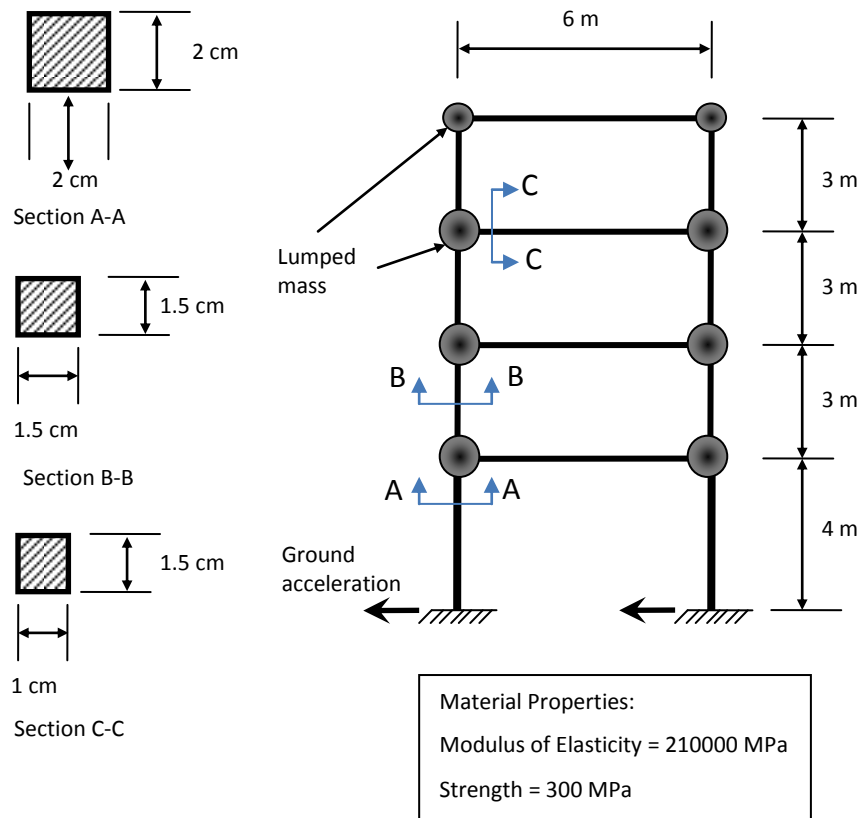


Figure 3-14: Four-storey frame structure with lumped mass

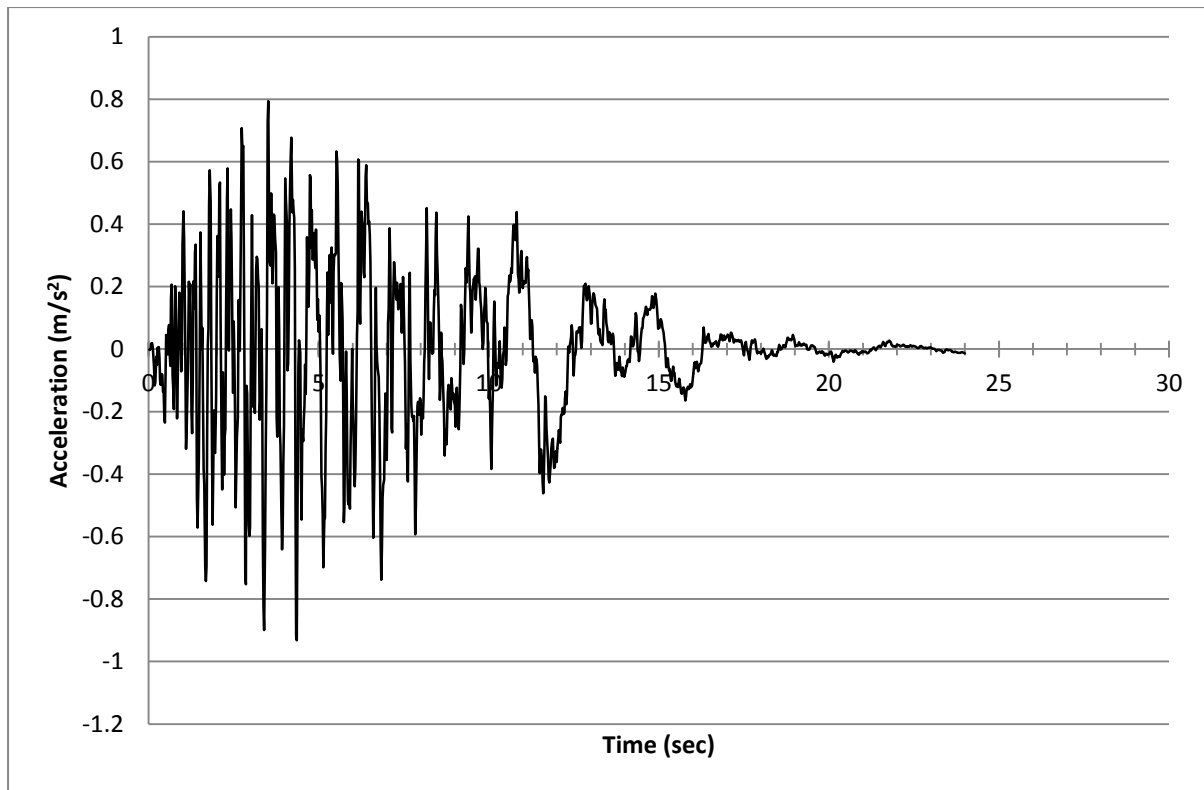


Figure 3-15: Ground acceleration record applied to four-storey frame

For the partitioned analysis, the frame is divided into one parent structure and 4 partitions as shown in Figure 3-16. Each storey is modelled as a separate partition, where the concept of modular modelling, which is demonstrated in more detail later in Example 6, is utilised to take the advantage of the fact that the top three stories are similar. The results obtained from partitioned analysis match exactly those obtained by the conventional monolithic approach, as shown in Figure 3-17 . A similar favourable comparison of the deformed shapes is shown in Figure 3-18.

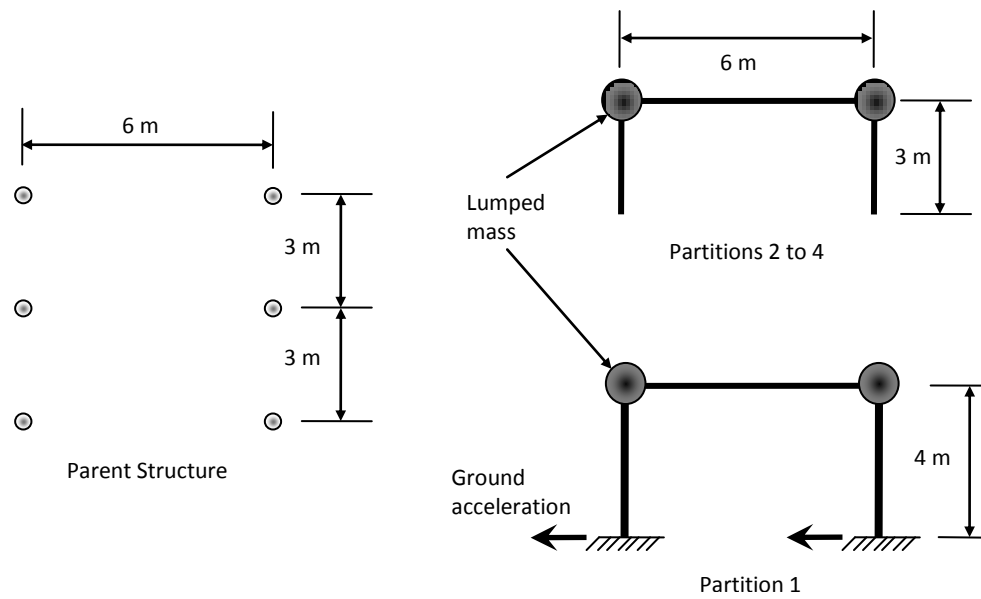


Figure 3-16: Partitioning of four-storey frame for parallel analysis

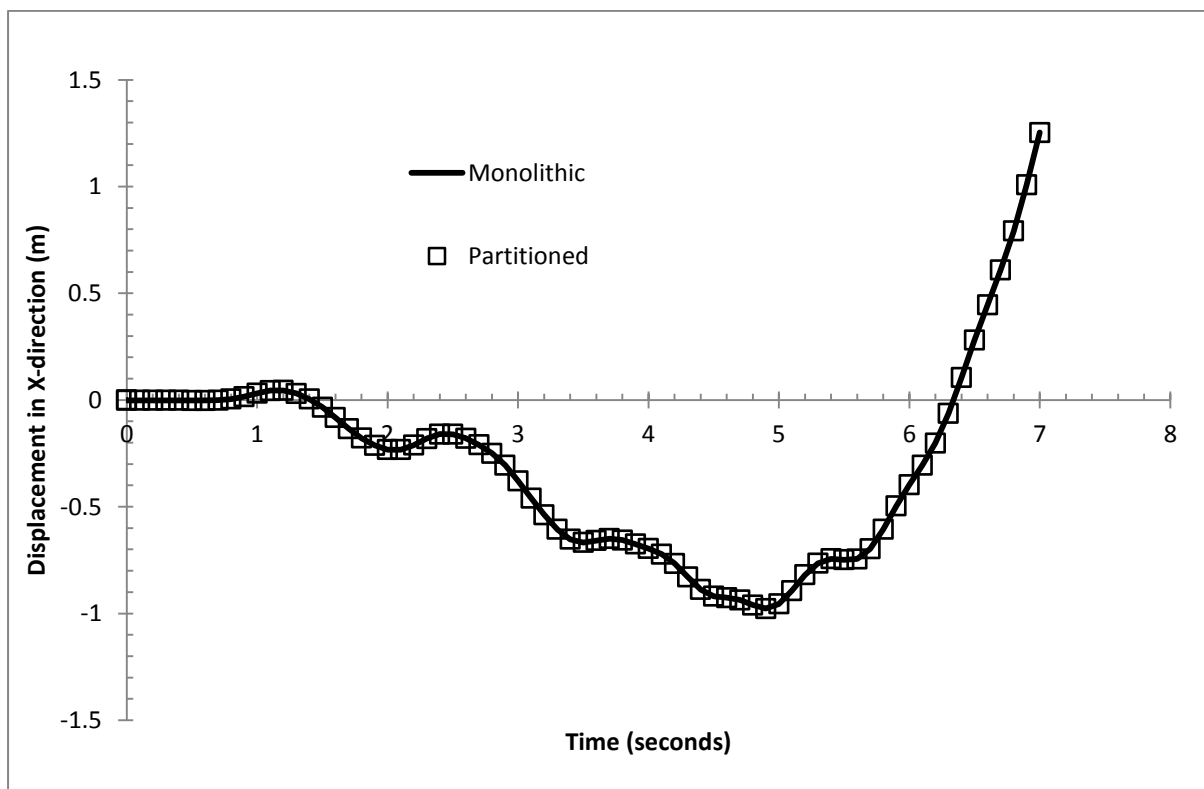


Figure 3-17: Comparison of displacement at the top

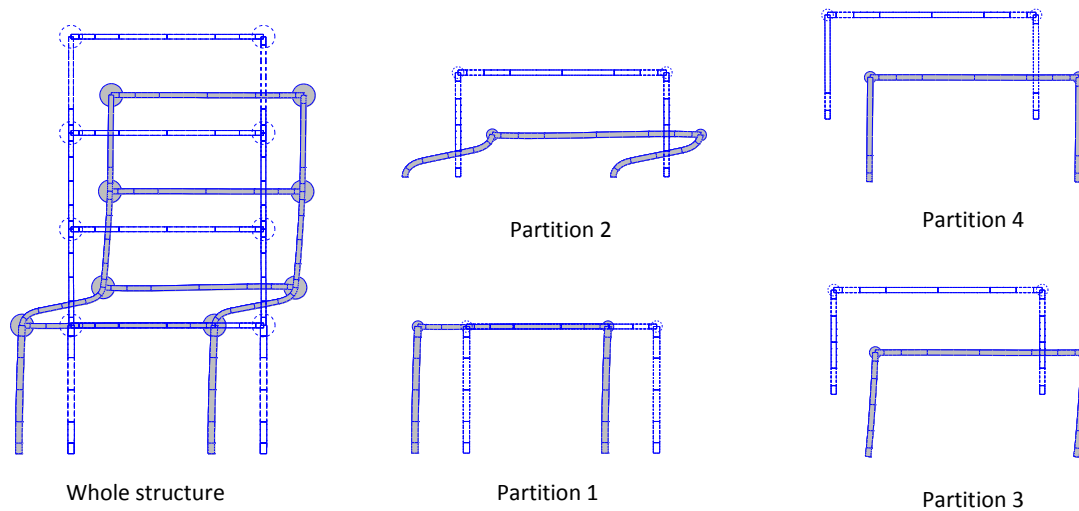


Figure 3-18: Comparison of deformed shapes

3.5.3 Reinforced Concrete Beam

The third example that verifies the accuracy of the present approach is a reinforced concrete beam subjected to static flexural loading, as shown in Figure 3-19. 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 [Izzuddin, 2009], where the associated material properties are given in Table 3-8.

The beam is modelled monolithically and using 4 partitions, where a favourable comparison of the deflected shapes is shown in Figure 3-20. A further comparison of the load-deflection response, as depicted in Figure 3-21, demonstrates an exact match.

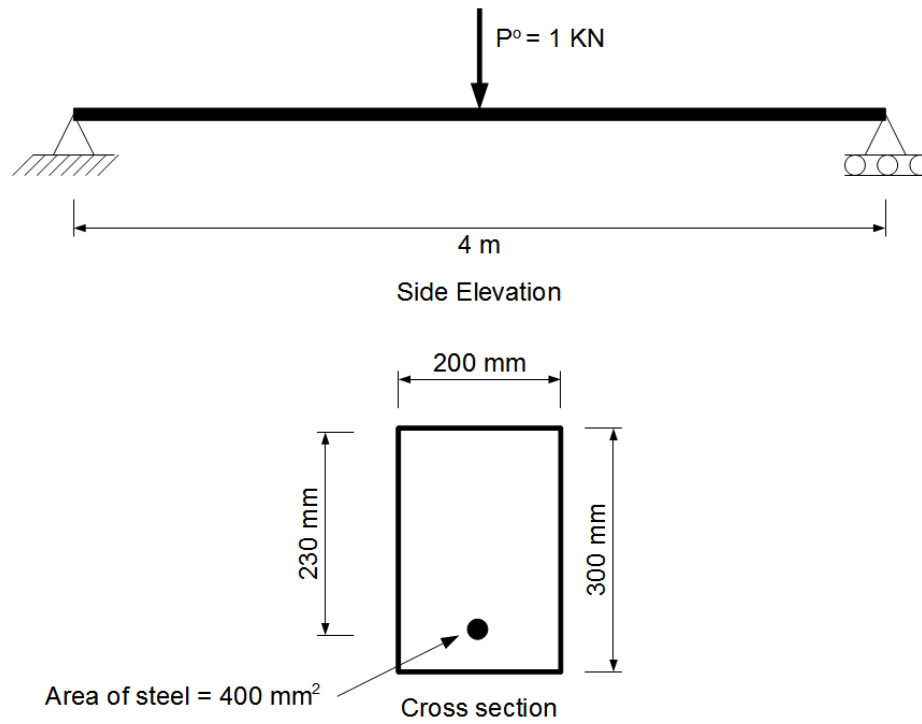


Figure 3-19: Reinforced concrete beam

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 3-8: Material properties used for reinforced concrete beam

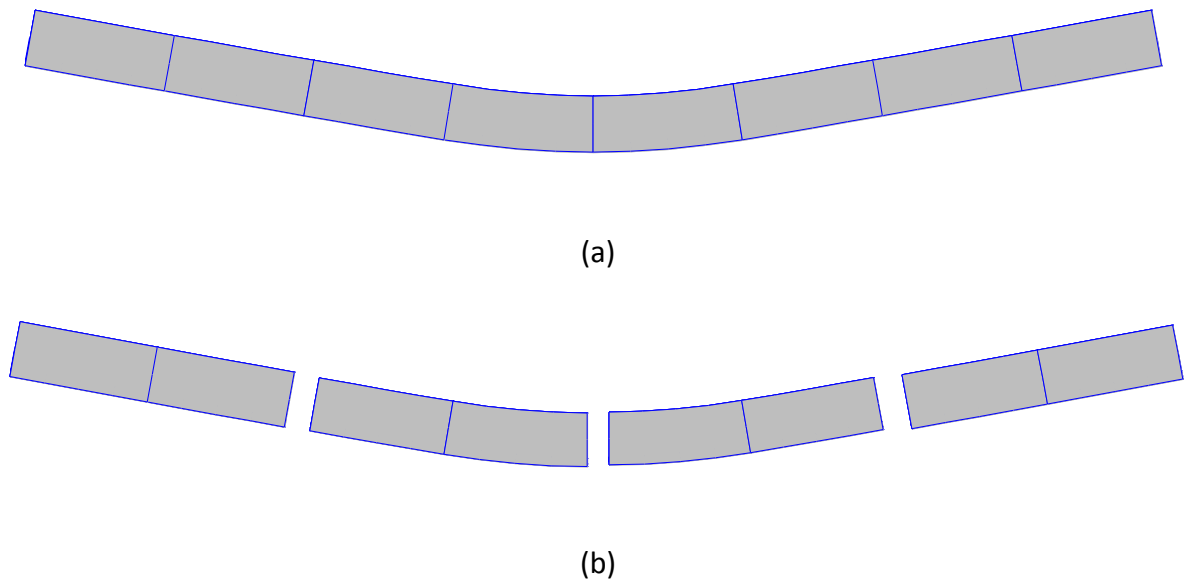


Figure 3-20: Deflected shapes for RC beam (a) whole beam (b) 4 partitions

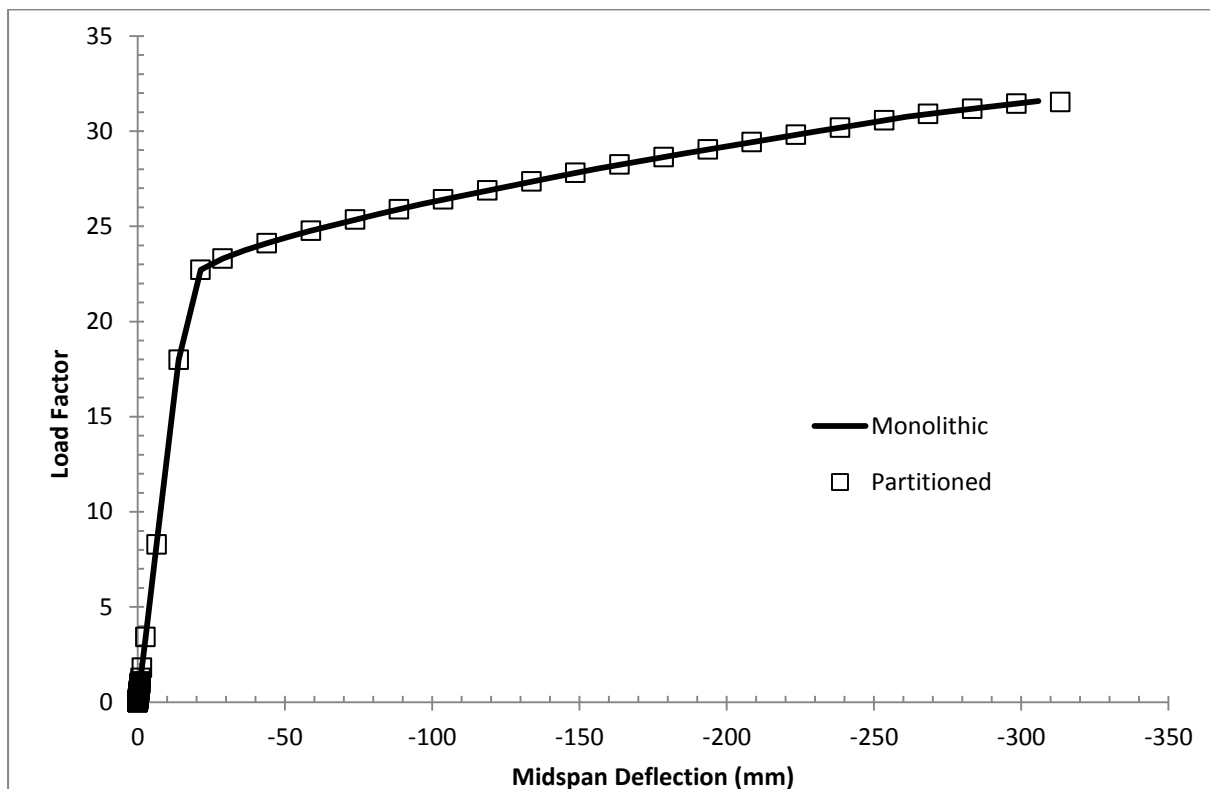


Figure 3-21: Load-deflection response of reinforced concrete beam

3.5.4 I-Beam Modelled with 3D Brick Elements

This is the first of three final examples used to demonstrate the computational efficiency of the proposed partitioned approach, considering an elastic beam made of 20-noded 3D brick elements. The beam cross-section is I-shaped (Figure 3-22) with the flange and web discretised using 5 brick elements each, with 10 element divisions used along the length of the beam, 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.

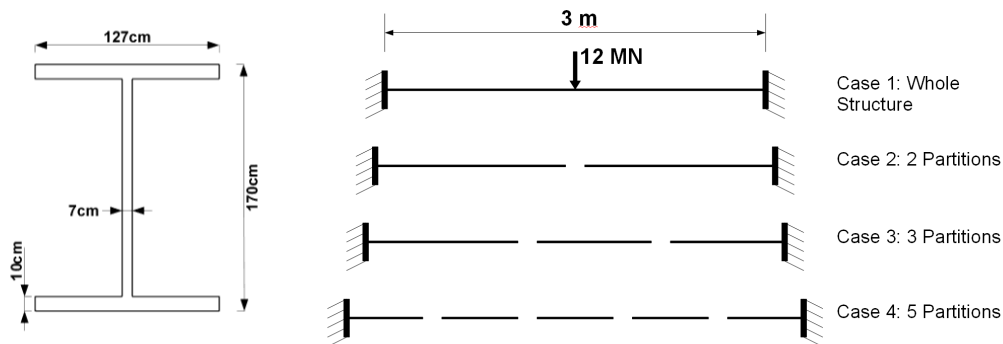


Figure 3-22: I-beam geometric configuration and alternative partitions

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 3-22. 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 parent structure plus 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 3-9 for all cases. It should be noted that the communication overhead will be in direct proportion to the size of the

parent structure resulting in more time required for the 5 partitions case than stipulated by the comparison based on equivalent freedoms in Table 3-9.

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 3-23. 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, 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.

	Single	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 Single	100	59.93	57.22	51.8

Table 3-9: Comparison of equivalent DoFs

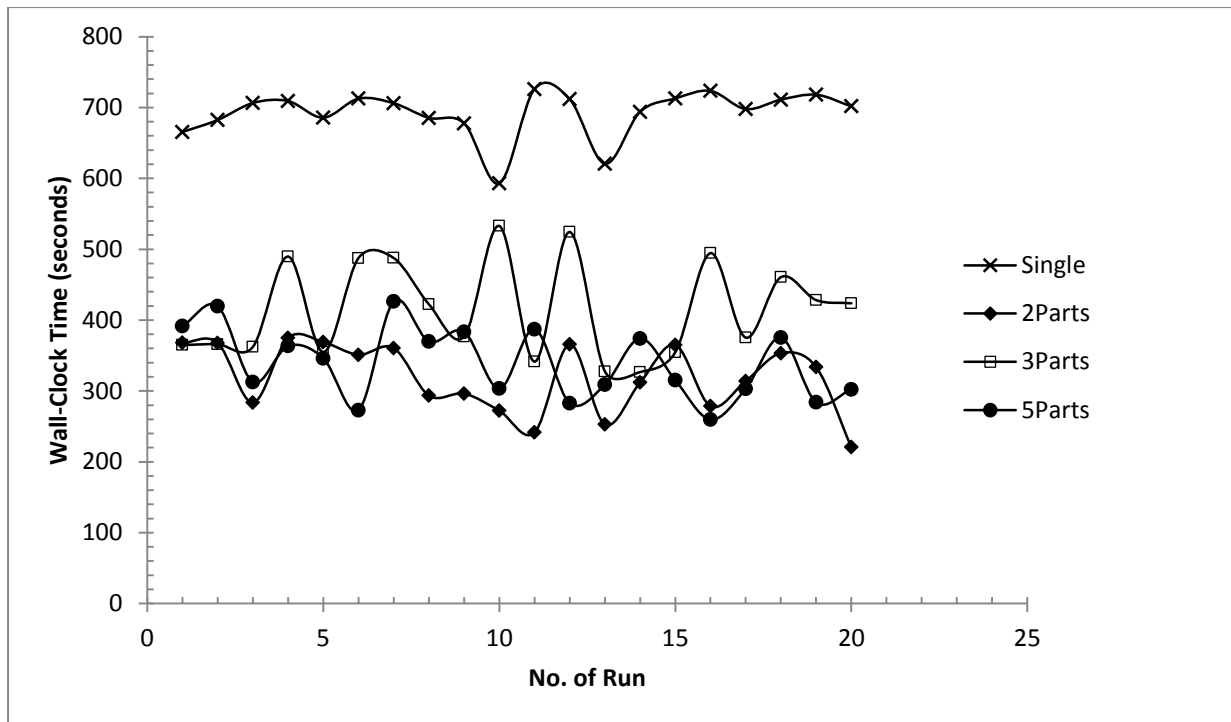


Figure 3-23: Comparison of wall-clock times for I-beam

3.5.5 Square Slab

The second example demonstrating computational efficiency of the partitioned approach is a square slab consisting of 1024 ($32 \times 32 \times 1$) 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 partitioned into 4 and 16 partitions, each consisting of $16 \times 16 \times 1$ and $8 \times 8 \times 1$ elements respectively, as illustrated in Figure 3-24.

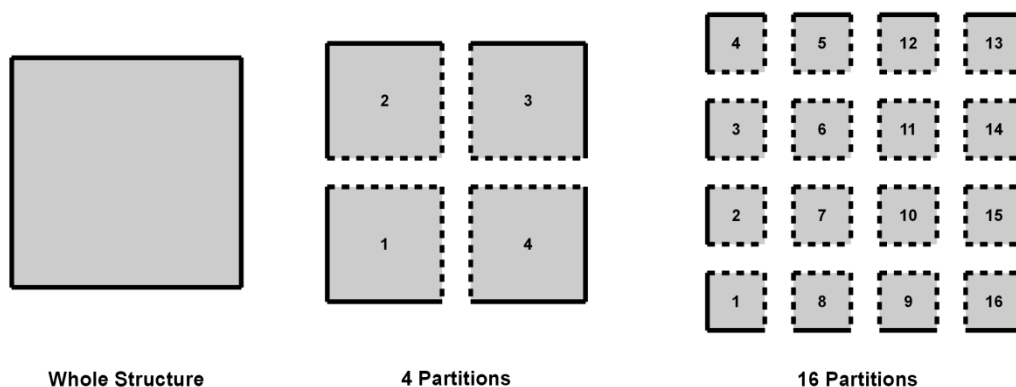


Figure 3-24: Slab and alternative partitioned models

This example also demonstrates the ease of modelling achieved by the present domain partitioning approach. As can be seen in Figure 3-24, the partitions are similar to one another 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 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 and the comparison of results is shown in Figure 3-25.

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 3-25.

To consider the influence of the demand of the element response evaluation on the efficiency of parallelisation, a third set of runs is considered with 1000 Gauss points per element. As observed from the results in Figure 3-26, 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 partition 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.

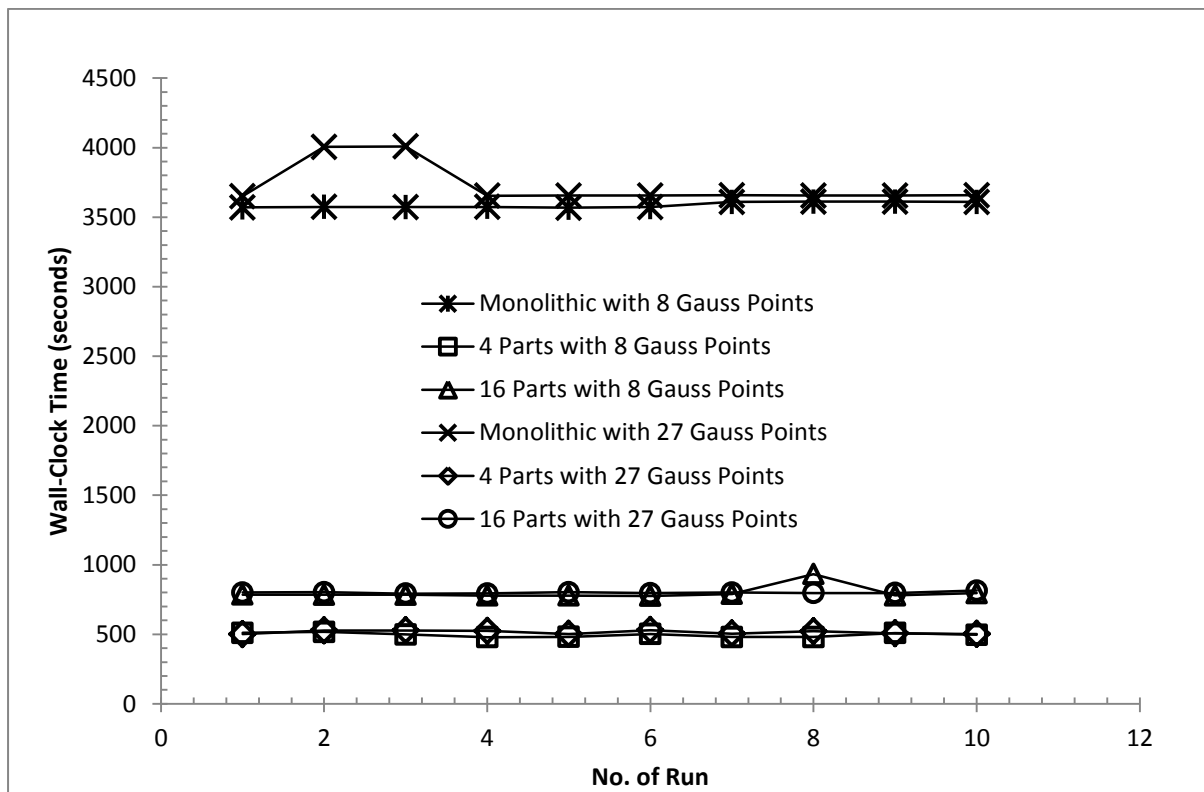


Figure 3-25: Wall-clock times for 8 and 27 Gauss points

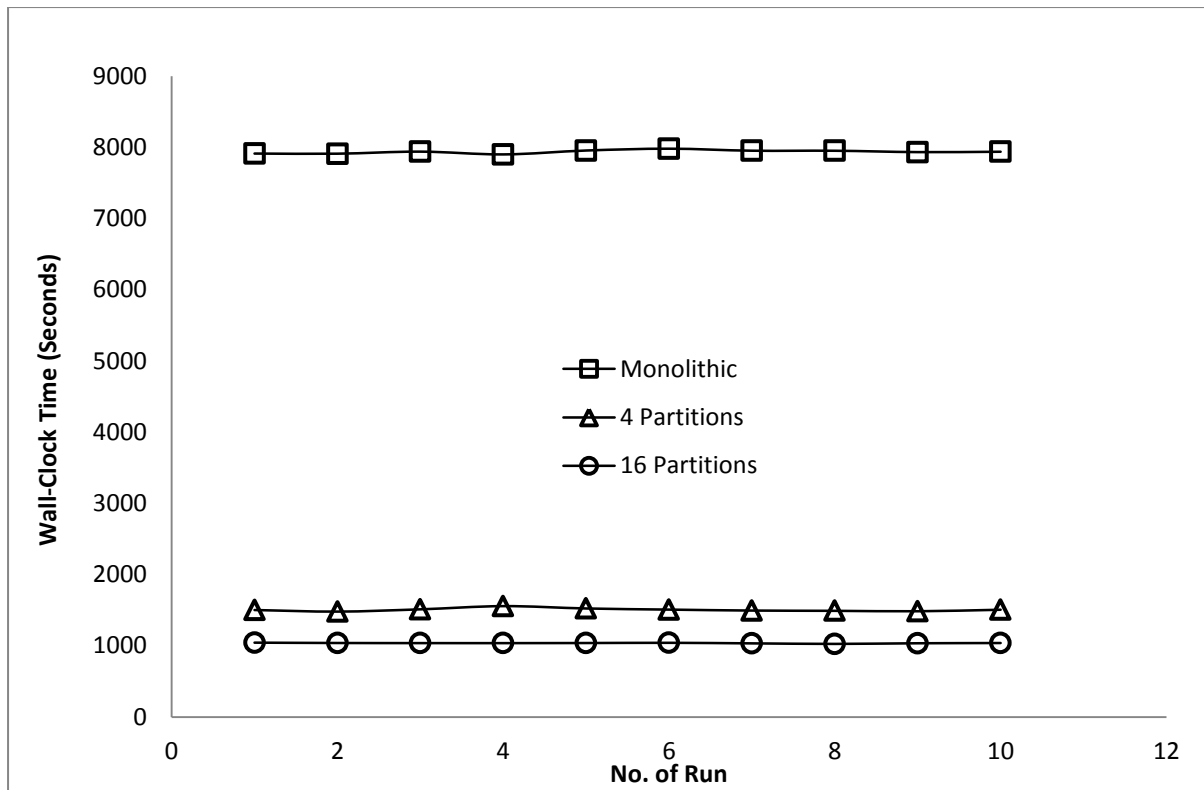


Figure 3-26: Wall-clock times for 1000 Gauss points

3.5.6 3D Frame

This final example is used to demonstrate the ease of modelling using the proposed partitioned approach in addition to the advantage gained in terms of the wall-clock time. Consideration is given to a 3D steel frame, where 4 partition objects are developed respectively for two different beam lengths, a column and a joint, using 3D brick elements, as shown in Figure 3-27. These objects are then combined to form the 2-storey 3D frame depicted in Figure 3-28. The frame structure is subjected to out-of-plane loads at one of its top corners, where the resulting deflected shape is shown in Figure 3-29.

Compared to the modelling of large monolithic structures, the modular approach facilitated by the developed partitioning approach requires much less modelling time, since identical partition objects have to be developed only once, with the rest becoming a question of connecting the corresponding super elements at the parent structure level. The analysis for this example was run for 72 hours on the HPC system at Imperial College London for both the monolithic and partitioned cases, where the number of loading steps successfully completed within the allowed time is shown in Table 3-10.

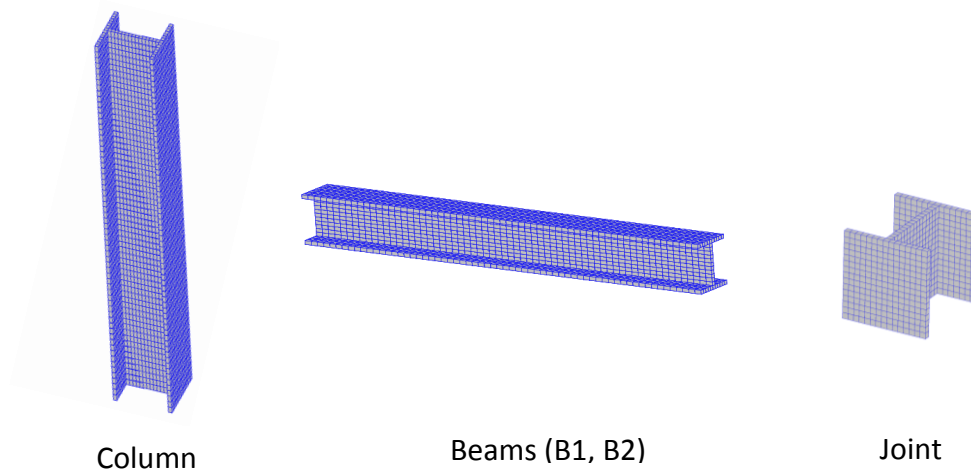


Figure 3-27: Partition objects assembled to form plane/space frame

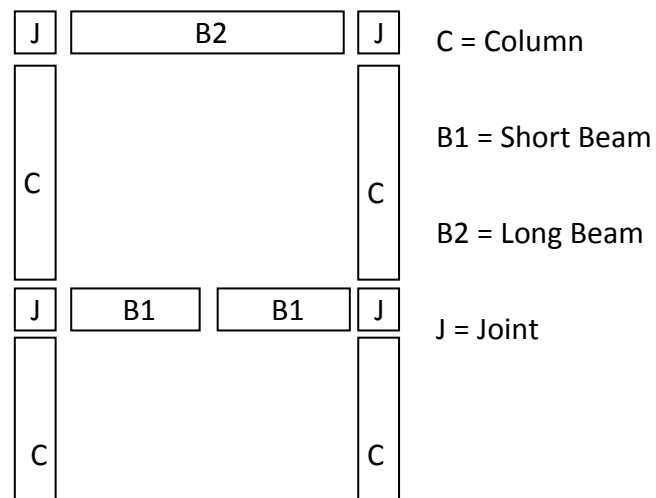


Figure 3-28: Modular approach of forming structure from partition objects

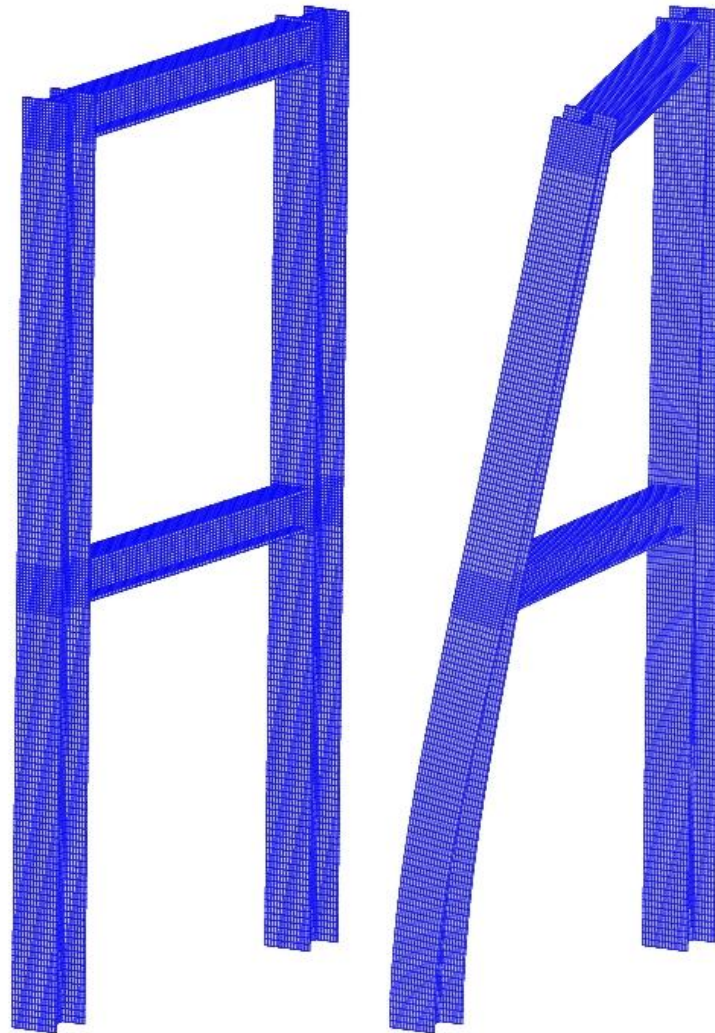


Figure 3-29: Final assembled 3D frame and deflected shape

Load steps completed for:	Initial Loading	Time-history	Total
Monolithic analysis	1	9	10
Partitioned analysis	1	120	121

Table 3-10: Comparison of load steps performed in 72 hours

It can be seen that the 11 partitions plus the parent structure have impressively exceeded a linear speedup factor of 11, which in any case is only possible if all partitions have identical computational demands, by around 10%. It can therefore be concluded that the communication overhead is negligible relative to the partition computations, and that in the monolithic analysis more time is taken for solving a large system of equations that exceeds RAM capacity with reliance on swap memory operations. These results again show that in addition to the ease of modelling offered by the proposed modular approach, significant savings can be made in terms of wall-clock time for the nonlinear analysis of real structures.

Partitioned Eigenvalue Analysis

This chapter extends the partitioned approach to eigenvalue analysis using the Lanczos Algorithm.

4.1 Introduction

The word ‘eigenvalue’ originates from a German word ‘eigen’ meaning ‘own’. In the context of linear algebra, eigenvalues represent a set of characteristic value of a matrix. For each of these characteristic values, there is a corresponding vector which when multiplied by the eigenvalue gives the same result as when it is multiplied by the matrix. These vectors are termed as ‘eigenvectors’. Mathematically, an eigenvalue problem can be stated as:

$$Ax = \psi x \quad 4-1$$

Where, A is a square matrix, x is an eigenvector and ψ is a scalar eigenvalue.

The applications of eigenvalue analysis are wide ranging covering a broad spectrum of science and engineering fields. The most important application in the context of this study is for frequency analysis, where the natural frequencies and associated mode shapes of structures are defined in terms of a generalised eigenvalue problem involving the stiffness and mass matrices. In addition to application in serviceability design and assessment considering structural vibrations, this type of analysis has wider application in the evaluation of the structural response under dynamic loading (e.g. earthquakes) using spectral analysis techniques.

The general eigenvalue problem for frequency analysis is stated as:

$$K u = \omega^2 M u \quad 4-2$$

where K is the stiffness matrix, M is the mass matrix, ω is the natural frequency, and u is the associated vibration mode vector.

Typically, the mass and the stiffness matrices for real discretised structures are large, sparse, symmetric, and at least positive semi-definite. For finding the associated eigenvalues, the Lanczos Algorithm [Lanczos 1950] is often cited as the most effective [Hughes 2000] compared to other solution methods, including power iteration, inverse iteration, QR algorithm, etc.

Although the Lanczos Algorithm is very efficient in calculating a few extreme eigenvalues of large matrices fairly quickly, it can still require significant computing time to determine the eigenvalues of very large discrete systems. In this respect, the extension of the partitioned approach developed in the previous chapter to eigenvalue analysis with the Lanczos Algorithm is expected to bring significant computational efficiencies. Furthermore, there is a need to enable eigenvalue analysis for partitioned models created for dynamic analysis, since it would be hugely inefficient to recreate equivalent monolithic models just for the purpose of frequency analysis.

In the following sections, the monolithic application of the Lanczos Algorithm is first presented. Subsequently, the previously developed partitioned approach is extended to enable parallel eigenvalue analysis with the Lanczos Algorithm. Finally, several examples are provided to demonstrate the features and computational benefits of the developed parallel eigenvalue analysis approach.

4.2 Monolithic Lanczos Algorithm

The general eigenvalue problem for frequency analysis of Equation 4—2 can be converted into the standard eigenvalue problem of the form of Equation 4—1 as:

$$(K^{-1}M)u = \left(\frac{1}{\omega^2}\right)u \quad 4-3$$

It can be seen that the eigenvalues of the above standard eigenvalue problem are the reciprocals of the square of the natural frequencies of the structure. Here, the term $(K^{-1}M)$ is often referred to as the ‘matrix pencil’. The Lanczos transformation of this matrix pencil is based on a sequence of orthogonal vectors $Q = [q_1, q_2, \dots, q_k]$ such that:

$$A = Q^T (K^{-1}M)Q \quad 4-4$$

where, A is a tridiagonal matrix of size $k \times k$.

Using the Lanczos transformation therefore, the standard eigenvalue problem can be converted into a reduced eigenvalue problem as given by Equation 4—1. It has been proved that the eigenvalues of this reduced problem converge towards the eigenvalues of the original matrix pencil as size of the tridiagonal increases [Lanczos, 1950; Paige 1976; Parlett and Scott, 1979], i.e.:

$$\psi \rightarrow \left(\frac{1}{\omega^2} \right) \quad \text{as} \quad k \rightarrow n \quad 4-5$$

where, $n \times n$ is the size of the matrix pencil.

A very important characteristic of the Lanczos transformation is that some of the eigenvalues of the tridiagonal matrix A converge fairly quickly to the extreme values of the original matrix pencil $(K^{-1}M)$. Normally, to obtain the largest eigenvalues of the matrix pencil, or the smallest natural frequencies of the structure, a very few Lanczos steps are required (i.e. $k \ll n$) when the reduced eigenvalue problem is used. For the solution of the reduced eigenvalue problem, any computational tool based on some standard procedure (e.g. QR Algorithm) can be used, which results in relatively little computational effort compared to the solution of the original general eigenvalue problem.

4.2.1 Lanczos Recursion

The Lanczos transformation and the generation of the tridiagonal matrix are based on the Lanczos recursion. The basic Lanczos recursion procedure is presented as follows:

Let B be a real square symmetric matrix of size $n \times n$ and let q_1 be a unit starting vector, which is usually generated randomly. Then for $j = 1 \rightarrow p$, where $p \leq n$, a series of Lanczos tridiagonal matrices A_j is generated by defining $q_0 = 0$ and $B_1 = 0$ followed by the calculation of q_i , α_i , and β_{i+1} for $i = 1 \rightarrow p$ as:

$$\beta_{i+1}q_{i+1} = Bq_i - \alpha_iq_i - \beta_iq_{i-1} \quad 4-6$$

$$\alpha_i = q_i^T B q_i \quad 4-7$$

$$\beta_{i+1} = q_{i+1}^T B q_i \quad 4-8$$

where α and β are the diagonal and off-diagonal terms of the tridiagonal Lanczos matrix A_j , respectively:

$$A_{j[i,i]} = \alpha_i \quad \forall \quad i=1 \rightarrow j \quad 4-9$$

$$A_{j[i,i+1]} = A_{j[i+1,i]} = \beta_{i+1} \quad \forall \quad i=1 \rightarrow j-1 \quad 4-10$$

By definition therefore, for every step ' i ' the new Lanczos vector q_{i+1} can be obtained by orthogonalising the vector Bq_i with respect to q_i and q_{i-1} . This also means that $\alpha_i q_i$ and $\beta_i q_{i-1}$ are the orthogonal projections of Bq_i over the two previous Lanczos vectors q_i and q_{i-1} . These definitions stay true throughout the Lanczos steps when computations are performed with infinite precision. With limited precision, however, the orthogonal projections are not purged completely from the new generated Lanczos vectors at every step. The small errors gradually magnify and as a result the new generated vectors lose orthogonality completely.

Paige (1976) proved that the loss of orthogonality of the Lanczos vectors signifies the convergence of an eigenvalue. For the continuation of Lanczos recurrence procedure, however, it is necessary to re-orthogonalise the new generated vectors against the previous Lanczos vectors. One such method is full re-orthogonalisation in which every new generated Lanczos vector is orthogonalised against all the previous Lanczos vectors thus diminishing the effects of limited precision computation [Parlett & Scott, 1979].

The process of full re-orthogonalisation was, however, considered computationally expensive because of the limited size of random access memory (RAM) available in early computers. The issue of limited RAM was overcome by 'sending the previous Lanczos vectors to secondary storage' a term generally found in literature to mean saving this data into virtual memory. The vectors were therefore retrieved from virtual memory one by one and their projections were purged from the new generated Lanczos vector. This time-consuming and cumbersome process led some researchers to adopt 'selective re-

orthogonalisation' [Hughes et al., 1983; Horst, 1984; Hughes, 2000]. In this technique, the new generated Lanczos vector is orthogonalised only against those previous vectors that had a projection of a significant size.

In the present work, however, full re-orthogonalisation has been adopted due to the fact that modern computing machines have enough RAM to keep all the previous vectors readily accessible. Furthermore, the use of partitioned analysis reduces the RAM requirements for individual processors. Indeed, it is shown in Section 4.4.2 that the use of full re-orthogonalisation produces more accurate/detailed results without incurring significant computational expenditure.

4.2.2 Convergence of Eigenvalues

Normally, the convergence of a quantity being calculated through an iterative process is checked by ensuring that the difference from the previous iteration is within a tolerable limit:

$$\frac{|\psi_j - \psi_{j-1}|}{|\psi_j|} < \varepsilon \quad 4-11$$

This criterion, however, cannot be applied to the eigenvalues of the tridiagonal [Hughes, 2000], but is instead applied to the eigenvectors. The eigenvectors of the full matrix 'u' are calculated by multiplying the eigenvectors of the tridiagonal 'x' with the Krylov subspace Q generated during Lanczos recursion as shown in Equation 4—28 later. The change in i^{th} eigenvector of the full matrix at step 'j' is:

$$\delta u_i = q_j x_{i\{j\}} \quad 4-12$$

It can, therefore, be concluded that there is no significant change in the i^{th} eigenvector of the full matrix if the bottom element of the corresponding eigenvector of the tridiagonal is negligible in value:

$$|x_{i\{j\}}| < \varepsilon \quad 4-13$$

Hence any eigenvalue of the tridiagonal is considered to be a good approximation of an eigenvalue of the full matrix if the bottom element of the corresponding eigenvector of the tridiagonal satisfies Equation 4—13.

4.2.3 Algorithm Steps

The procedure followed by the monolithic Lanczos algorithm is presented below and illustrated in the flow chart of Figure 4-1.

1. Assemble the mass matrix M and the stiffness matrix K .
2. Start with an initial vector t , which is either given or generated randomly.
3. Use t to obtain a starting trial vector which satisfies the essential boundary conditions:

$$r_0 = K^{-1} M t \quad 4-14$$

4. Calculate:

$$\bar{p}_0 = M r_0 \quad 4-15$$

$$\beta_1 = \sqrt{\bar{p}_0^T r_0} \quad 4-16$$

5. For each of j from 1 to maximum number of Lanczos steps:

$$q_j = \frac{1}{\beta_j} r_{j-1} \quad 4-17$$

$$p_j = \frac{1}{\beta_j} \bar{p}_{j-1} \quad 4-18$$

$$\bar{r}_j = K^{-1} p_j \quad 4-19$$

$$\hat{r}_j = \bar{r}_j - q_{j-1} \beta_j \quad 4-20$$

$$\alpha_j = p_j^T \hat{r}_j \quad 4-21$$

noting that q_0 , required in Eq. 4—20 for the first Lanczos step, is a zero vector.

6. If the next Lanczos step is needed, obtain the next trial vector:

$$r_j = \hat{r}_j - q_j \alpha_j \quad 4-22$$

otherwise, proceed to the final step 16.

7. Perform full reorthogonalisation of r_j :

$$r_j = r_j - \sum_{k=1}^{j-1} (q_k^T M r_j) q_k \quad 4-23$$

8. Determine the terms required for the calculation of the next Lanczos vector:

$$\bar{p}_j = M r_j \quad 4-24$$

$$\beta_{j+1} = \sqrt{\bar{p}_j^T r_j} \quad 4-25$$

9. Formulate the tridiagonal Lanczos submatrix:

$$A_j = \begin{bmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \alpha_3 & \ddots & & \\ & & & \ddots & \ddots & \\ & & & & \alpha_{j-2} & \beta_{j-1} \\ & & & & \beta_{j-1} & \alpha_{j-1} & \beta_j \\ & & & & & \beta_j & \alpha_j \end{bmatrix} \quad 4-26$$

10. Solve the tridiagonal eigenvalue problem (e.g. using subroutine DSTEVD in LAPACK library), and return eigenvectors x_i along with eigenvalues ψ_i .
11. Check the bottom element of each of the eigenvectors x_i to be within an acceptable tolerance for convergence of the corresponding eigenvalue ψ_i .
12. If enough eigenvalues have converged, proceed to the final step 16.

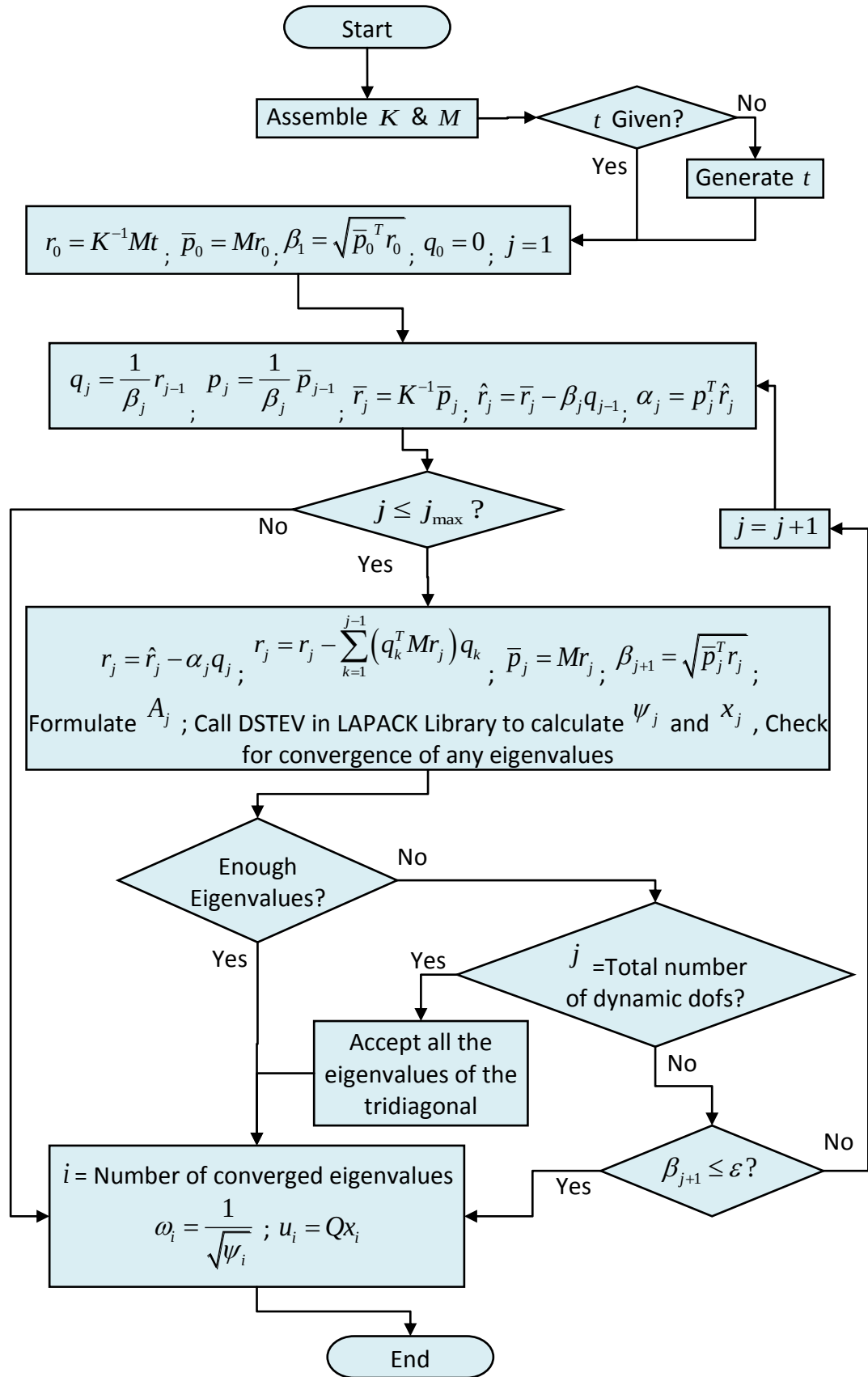


Figure 4-1: Flow chart for the monolithic Lanczos Algorithm

13. If the size of the tridiagonal (i.e. the current Lanczos step) is equal to the total dynamic degrees of freedom of the structure (i.e. number of freedoms associated with non-zero mass), accept all the eigenvalues and eigenvectors of the tridiagonal as converged, and proceed to the final step 16.
14. If the next Lanczos vector will be trivial, as reflected by a very small β_{j+1} , the maximum number of possible eigenvalues with the current starting trial vector has been reached, thus proceed to step 16.
15. Continue to the next Lanczos step, setting $j = j + 1$ and repeating from step 5.
16. Return the natural frequencies and associated vibration modes of the original problem, considering the accepted/converged eigenvalues ψ_i and eigenvectors x_i of the reduced tridiagonal problem:

$$\omega_i = \frac{1}{\sqrt{\psi_i}} \quad 4-27$$

$$u_i = Q x_i \quad 4-28$$

$$\text{where } Q = [q_1, \dots, q_j].$$

4.3 Partitioned Lanczos Algorithm

The domain partitioning approach presented in Chapter 3 is extended for use with the Lanczos algorithm to enable parallel eigenvalue analysis. As before, a structure is partitioned into parent structure and child structures, where the parent structure is handled by the coordinator process and the child structures are handled by processes referred to as partition processes. The main changes to the monolithic Lanczos algorithm and the procedures followed in the coordinator and child processes are presented in following subsections.

4.3.1 Initialisation

In the serial implementation, an initial vector t , which is either user-defined or randomly generated, is used to obtain a starting trial vector according to Eq. 4—14 . With partitioning, however, the initial vector may be defined independently in different partitions. If none of

the partitions is supplied with an initial vector, all partitions generate their own initial vectors randomly; otherwise, if at least one partition is supplied with an initial vector, then initialisation proceeds with the initial vectors in the remaining partitions taken as zero.

The determination of the starting trial vector according to Eq. 4--14 involves solving the static equilibrium equations with an out-of-balance vector equal to $-M t$. This can be dealt with in a partitioned manner at the child and parent levels, using the dual super element approach as presented in Section 3.2. For a child partition Ω_m , the stiffness matrix and out-of-balance vector are defined as:

$$K = K_{\Omega_m} \quad 4-29$$

$$G = -M_{\Omega_m} t_{\Omega_m} \quad 4-30$$

Forward elimination is first employed up to the partitioned boundary nodes, as discussed in Section 3.2.2, at which point the child partition condensed tangent stiffness matrix $K_{\Omega_m}^c$ and resistance forces $R_{\Omega_m}^c$ are returned to the parent partition as the contribution of the child super element.

At the parent level, the stiffness matrix involves the assembly of element contributions in the parent structure K_{Ω_0} and condensed super element contributions $K_{\Omega_m}^c$, while the out-of-balance vector involves $-M t$ evaluated for the parent structure and assembled with condensed super element contributions $R_{\Omega_m}^c$, as expressed by:

$$K = K_{\Omega_0} + \sum_{m \geq 1} K_{\Omega_m}^c \quad 4-31$$

$$G = -M_{\Omega_0} t_{\Omega_0} + \sum_{m \geq 1} R_{\Omega_m}^c \quad 4-32$$

noting that ' $\sum$ ' here indicates assembly over the relevant freedoms instead of direct summation. Once K and G are obtained for the parent partition, forward elimination is immediately followed by backward substitution, as discussed in Section 3.2.1, to determine the starting trial vector $r_{o,\Omega_0} = \delta d$ at the parent level.

Once the solution for the trial vector is obtained for the parent partition, backward substitution can proceed for the child partitions given the values of δd at the partition boundary, as discussed in Section 3.2.2, leading to $r_{o,\Omega_m} = \delta d$.

Once the trial vector is determined for all partitions, vector $\bar{p}_0$ is obtained at the partition level and β_1 is assembled from partition contributions:

$$\bar{p}_{0,\Omega_m} = M_{\Omega_m} r_{0,\Omega_m} \quad 4-33$$

$$\beta_1 = \sqrt{\sum_{m \geq 0} \bar{p}_{0,\Omega_m}^T r_{0,\Omega_m}} \quad 4-34$$

4.3.2 Lanczos Steps

In the proposed partitioning approach, the various vectors associated with nodal freedoms are partitioned and attached to individual parent/child partitions. For a parent/child partition Ω_m ($m \geq 0$), the local vector allocation includes the trial and intermediate vectors $(r_{j,\Omega_m}, \bar{r}_{j,\Omega_m}, \hat{r}_{j,\Omega_m})$, the Lanczos vectors (q_{j,Ω_m}) , as well as the loading and intermediate vectors $(p_{j,\Omega_m}, \bar{p}_{j,\Omega_m})$.

Following initialisation as discussed in the previous section, the same steps 5 to 8 of the monolithic algorithm presented in Section 4.2 are followed for all partitions in parallel. Most of the operations are performed locally at the partition level, specifically those involving basic scaling of the local partition vectors in Eqs. 4—17, 4—18, 4—20, 4—22 and 4—24, which become:

$$q_{j,\Omega_m} = \frac{1}{\beta_j} r_{j-1,\Omega_m} \quad 4-35$$

$$p_{j,\Omega_m} = \frac{1}{\beta_j} \bar{p}_{j-1,\Omega_m} \quad 4-36$$

$$\hat{r}_{j,\Omega_m} = \bar{r}_{j,\Omega_m} - q_{j-1,\Omega_m} \beta_j \quad 4-37$$

$$r_{j,\Omega_m} = \hat{r}_{j,\Omega_m} - q_{j,\Omega_m} \alpha_j \quad 4-38$$

$$\bar{p}_{j,\Omega_m} = M_{\Omega_m} r_{j,\Omega_m} \quad 4-39$$

The solution for $\bar{r}_j$, as expressed by Eq. 4—19, involves the same process of forward elimination and backward substitution over the parent/child partitions, as discussed in Section 4.3.1, with Mt substituted by p_j . The determination of the tridiagonal matrix terms, as given previously for the monolithic algorithm by Eqs. 4—21 and 4—25, involves now aggregation over all partitions:

$$\alpha_j = \sum_{m \geq 0} p_{j,\Omega_m}^T \hat{r}_{j,\Omega_m} \quad 4-40$$

$$\beta_{j+1} = \sqrt{\sum_{m \geq 0} \bar{p}_{j,\Omega_m}^T r_{j,\Omega_m}} \quad 4-41$$

The remaining operation which involves all partitions is the reorthogonalisation of the trial vector, as expressed for the monolithic algorithm by Eq. 4—23, which becomes:

$$r_{j,\Omega_m} = r_{j,\Omega_m} - \sum_{k=1}^{j-1} \left(\sum_{s \geq 0} q_{k,\Omega_s}^T M_{\Omega_s} r_{j,\Omega_s} \right) q_{k,\Omega_m} \quad 4-42$$

The formation of the tridiagonal matrix given by Eq. 4—26 and its solution are performed in the coordinator process, which also determines whether convergence to the required number of eigenvalues has been achieved.

Finally, the converged natural frequencies are determined as before using Eq. 4—27, while the corresponding vibration modes, obtained previously from Eq. 4—28, are now determined in the individual child partitions:

$$u_{i,\Omega_m} = Q_{\Omega_m} x_i \quad 4-43$$

where $Q_{\Omega_m} = [q_{1,\Omega_m}, \dots, q_{j,\Omega_m}]$.

4.3.3 Parallel Implementation of Lanczos Algorithm

Similar to the general partitioning approach presented in Chapter 3, the partitioned Lanczos Algorithm is implemented within ADAPTIC [Izzuddin, 1991] to enable parallel eigenvalue analysis. For this purpose, two separate subroutines have been written, one for the coordinator process and another for the child partition processes. The procedures followed by the two subroutines are described below and illustrated in Figure 4-2 and Figure 4-3.

At the start, each partition process broadcasts whether it is supplied with an initial vector t , and if this is not the case for any of the partitions, an instruction is broadcast to all partitions to generate a random initial vector.

Once the initial vector is defined, each partition determines its corresponding G vector, and forward elimination is followed by backward substitution, as discussed in Section 4.3.1, to determine the starting trial vector r_o in all partitions. Subsequently, $\bar{p}_o$ is obtained by each partition process according to Eqs. 4—33, and β_1 is obtained in all partitions from partition contributions.

Following initialisation, the coordinator process takes control of the Lanczos steps, and sends an instruction to child processes to commence a new Lanczos step. The coordinator and child processes then proceed in parallel with steps 5 to 8 of the Lanczos algorithm, as discussed in Sections 4.2.3 and 4.3.2, after which the coordinator continues to the end of one Lanczos cycle at step 15. This procedure is repeated until convergence to the required number of eigenvalues or achieving the maximum number of Lanczos steps.

With regard to the parallel steps 5 to 8, the operations involved in obtaining q_j , p_j , $\hat{r}_j$, r_j and $\bar{p}_j$, according to Eqs. 4—35 to 4—39, are local to the individual partition process. On the other hand, the determination of $\bar{r}_j$ involves forward elimination and backward substitution over all partition processes. Similarly, the determination of α_j and β_{j+1} from Eqs. 4—40 and 4—41, respectively, and the reorthogonalisation of r_{j,Ω_m} according to 4—42 require aggregation of contributions from all processes.

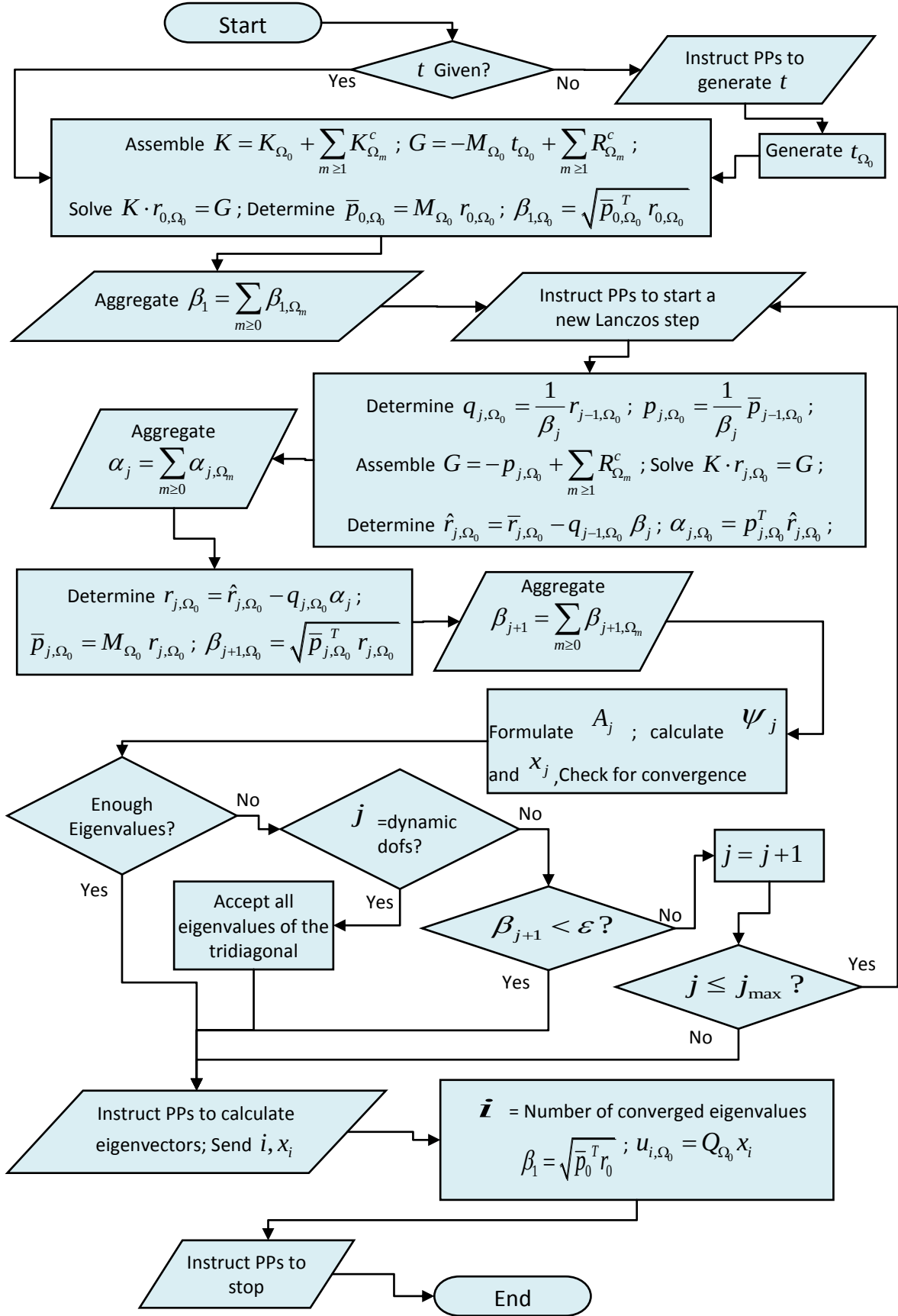


Figure 4-2: Flow chart of the Lanczos algorithm in the coordinator process

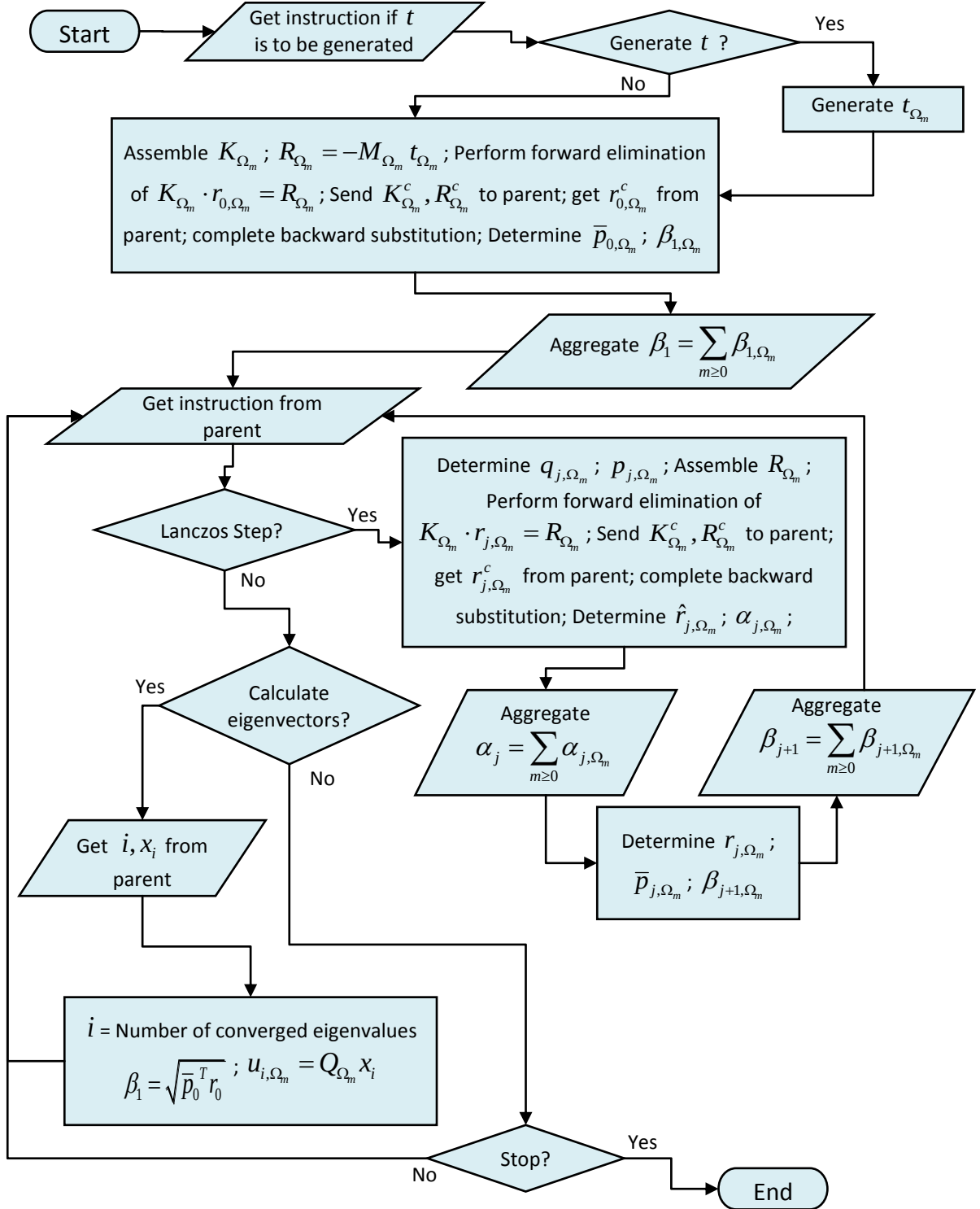


Figure 4-3: Flow chart for the Lanczos algorithm in a partition process

Once the coordinator processes establishes that convergence or the maximum number of Lanczos steps has been achieved, it sends an instruction to child processes to terminate the Lanczos steps. At this point, the child processes receive from the coordinator process the converged solution of the tridiagonal eigenvalue problem, which allows the converged natural frequencies and corresponding vibration modes to be obtained in all partitions according to Eqs. 4—27 and 4—43.

4.4 Verification

This section presents several examples which demonstrates the accuracy of the monolithic Lanczos Algorithm, including the specific features adopted for this work, and the significant computational benefits arising from its incorporation within the developed partitioning approach.

4.4.1 Clamped Beam

This example is used to demonstrate the accuracy of the monolithic Lanczos Algorithm, where consideration is given to a steel beam which is clamped at both ends and with lumped mass at the nodes, as depicted in Figure 4-4. The beam is modelled with 10 linear elastic Euler-Bernoulli beam elements, where frequency analysis is undertaken using the built-in eigenvalue solution capability of the symbolic computing tool Maple as well as using the Lanczos algorithm implemented within ADAPTIC. Perfect comparison is achieved between the two sets of results for all 18 modes, as demonstrated in Table 4-1 where the resulting vibration modes are depicted in Figure 4-5.

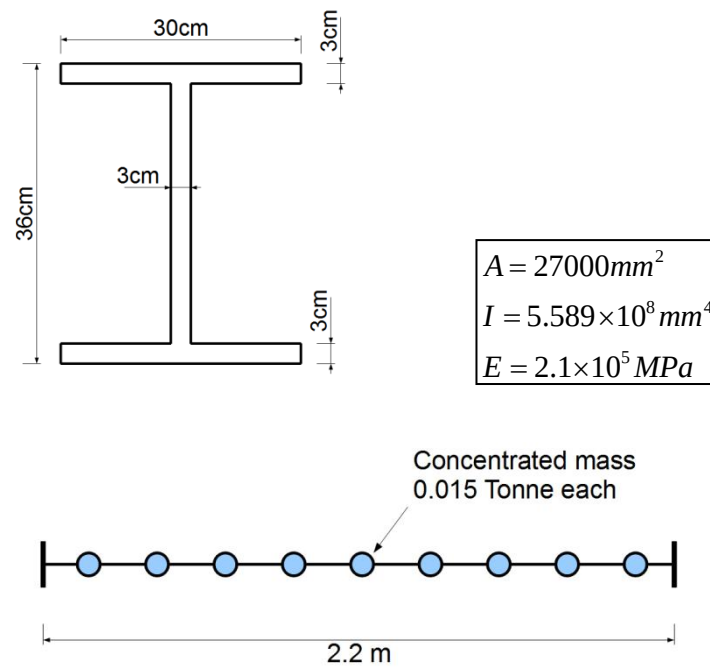


Figure 4-4: Clamped beam model

Mode	Natural Frequency (rad/sec)	
	Lanczos Algorithm/ADAPTIC	Maple Solution
1	59.1853	59.23705127
2	126.562	126.561714
3	163.092	163.1067248
4	250.007	250.0070577
5	319.298	319.3031968
6	367.296	367.2964013
7	475.542	475.541682
8	525.735	525.7362453
9	572.078	572.0775539
10	654.527	654.5269762
11	720.86	720.8597699
12	769.443	769.4426075
13	777.681	777.6810716
14	799.079	799.0792121
15	1062.78	1062.77735
16	1354.34	1354.341609
17	1608	1607.996211
18	1775.79	1775.791741

Table 4-1: Comparison of results for the clamped beam

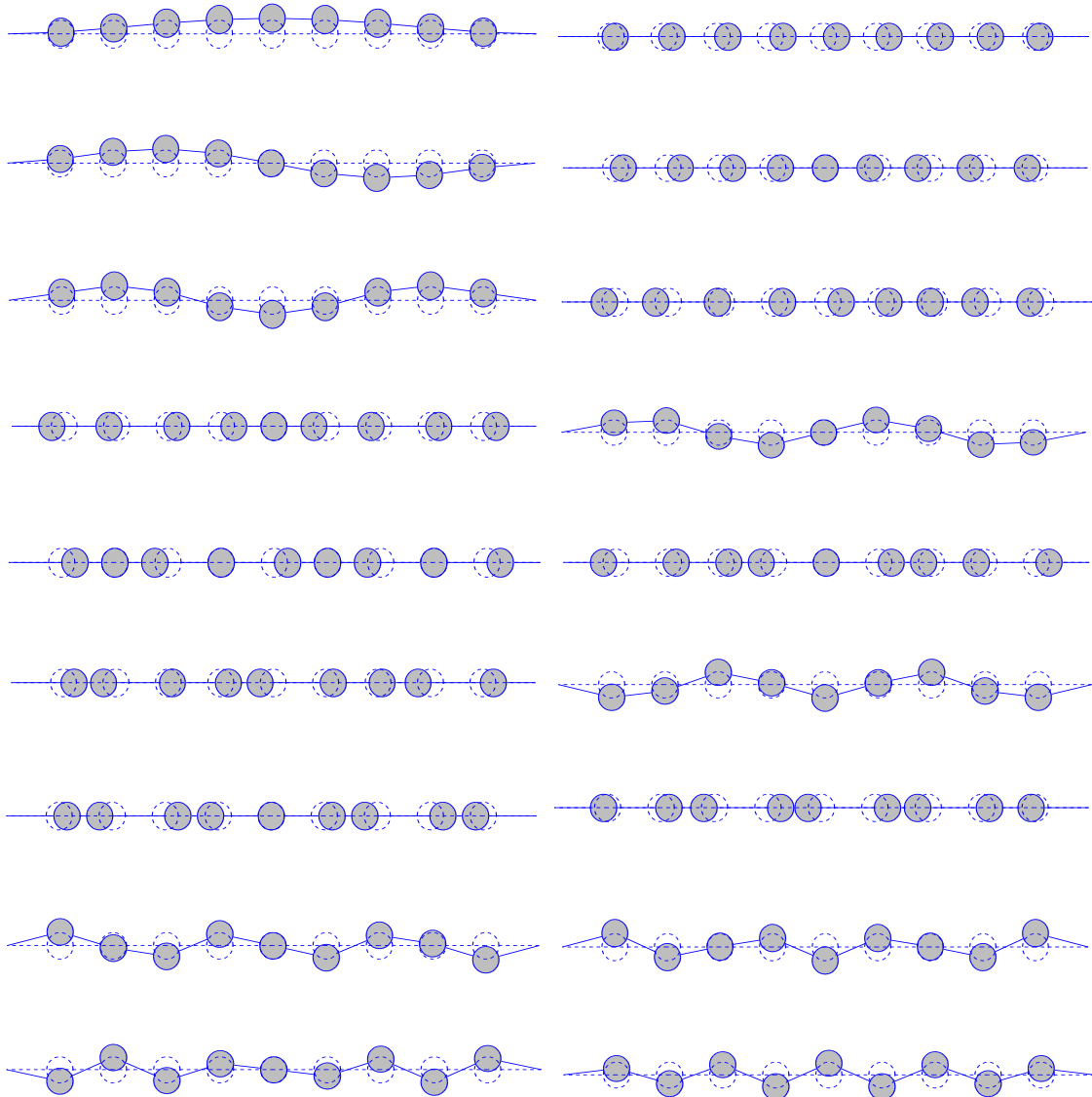


Figure 4-5: Vibration mode shapes for the clamped beam

4.4.2 23-storey Framed Structure

The second example is a 23-storey frame used to demonstrate that full re-orthogonalisation of the trial vector achieves accuracy at no significant additional cost compared to selective re-orthogonalisation. The structure, shown in Figure 4-6, is modelled with elastic beam-column elements, where lumped mass is considered at all nodes.

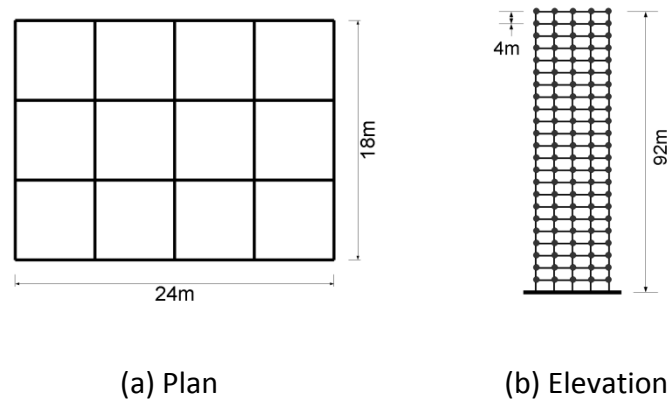


Figure 4-6: Plan and elevation of multi-storey frame

A total of 100 eigenvalues and the corresponding mode shapes are obtained with the Lanczos algorithm firstly using full re-orthogonalisation of the trial vectors and secondly using selective re-orthogonalisation [Hughes, 2000]. The total computational time required in the first case is 0.882 seconds compared to 0.791 seconds in the second case, representing an increase of only 11.5%. Two vibration modes, specifically modes 7 and 100, are illustrated in Figure 4-7 and Figure 4-8 for the considered re-orthogonalisation methods, while the natural frequencies for selected modes are shown in Table 4-2.

It can be observed that both the approaches give identical results for the initial 92 modes. Thereafter, however, selective re-orthogonalisation misses some of the eigenvalues and returns higher ones, whereas full re-orthogonalisation continues to return the subsequent modes accurately. The relevant values are highlighted in Table 4-2, where it is observed that modes 95 and 99-100 returned with full re-orthogonalisation are returned as modes 93-95

by selective re-orthogonalisation. Modes 93-94 and 96-98 returned by full re-orthogonalisation are however completely missed by the selective approach.

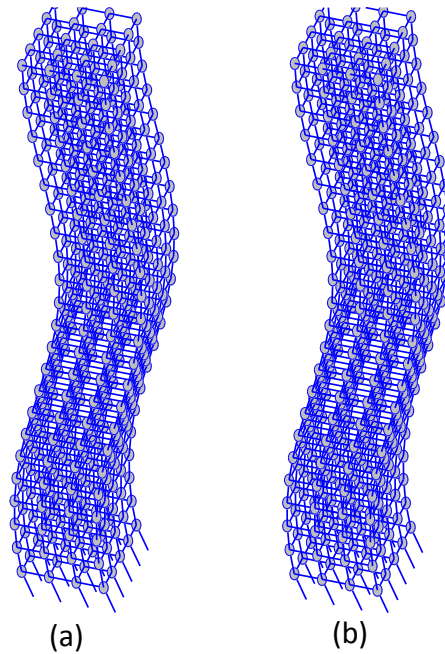


Figure 4-7: Comparison of mode shapes for mode 7 (a) full re-orthogonalisation (b) selective re-orthogonalisation

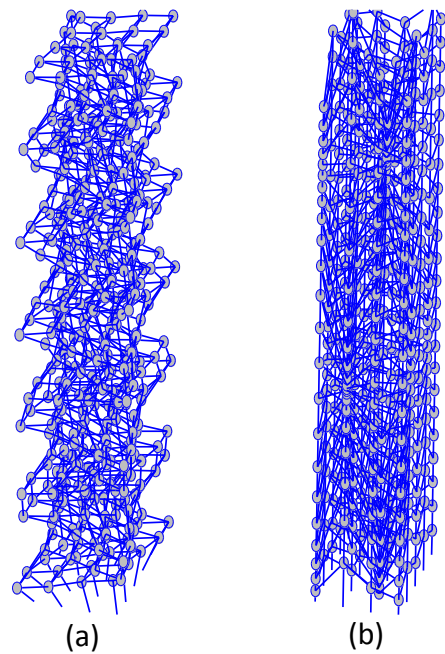


Figure 4-8: Comparison of mode shapes for mode 100 (a) full re-orthogonalisation (b) selective re-orthogonalisation

Mode	Natural Frequency (rad/sec)	
	Full re-orthogonalisation	Selective re-orthogonalisation
1	0.144481	0.144481
10	1.05196	1.05196
20	1.75359	1.75359
30	2.19296	2.19296
40	2.61026	2.61026
50	3.0717	3.0717
60	3.50087	3.50087
70	3.85463	3.85463
80	4.1111	4.1111
90	4.40594	4.40594
91	4.41586	4.41586
92	4.54566	4.54566
93	4.55877	4.57674
94	4.55949	4.76349
95	4.57764	4.84091
96	4.60671	10.8277
97	4.62015	12.2156
98	4.6275	32.435
99	4.76349	53.9007
100	4.84091	54.0404

Table 4-2: Natural frequencies of 23-storey framed structure

The above results indicate that full re-orthogonalisation provides accurate results without adding significantly to computational cost. Clearly, the computational demand of the operations associated with full re-orthogonalisation is not significant relative to other operations (e.g. forward elimination/backward substitution). Moreover, with the availability of sufficient random access memory (RAM), the Lanczos vectors can be stored and retrieved efficiently, which renders the related benefits of selective re-orthogonalisation marginal. Indeed, one of the main benefits of the proposed partitioning method is the reduction in memory demands per partition, which enables the storage of partition data within RAM instead of requiring virtual memory, consequently allowing full re-orthogonalisation to be used without an undue increase in computing time.

4.4.3 6-storey Plane Frame

The 6 storey plane frame in this example is used to demonstrate the accuracy of the partitioned Lanczos algorithm using the proposed approach with dual partition super elements. The frame, shown in Figure 4-9, has been modelled with quartic elastic 2D beam column elements, the cross-sections of which are shown in Figure 4-10.

A total of 10 eigenvalues and the corresponding mode shapes are first obtained using the monolithic Lanczos algorithm, the accuracy and reliability of which have been demonstrated in the two previous examples. The mode shapes for selected modes (3, 9 and 6) are presented in Figure 4-11.

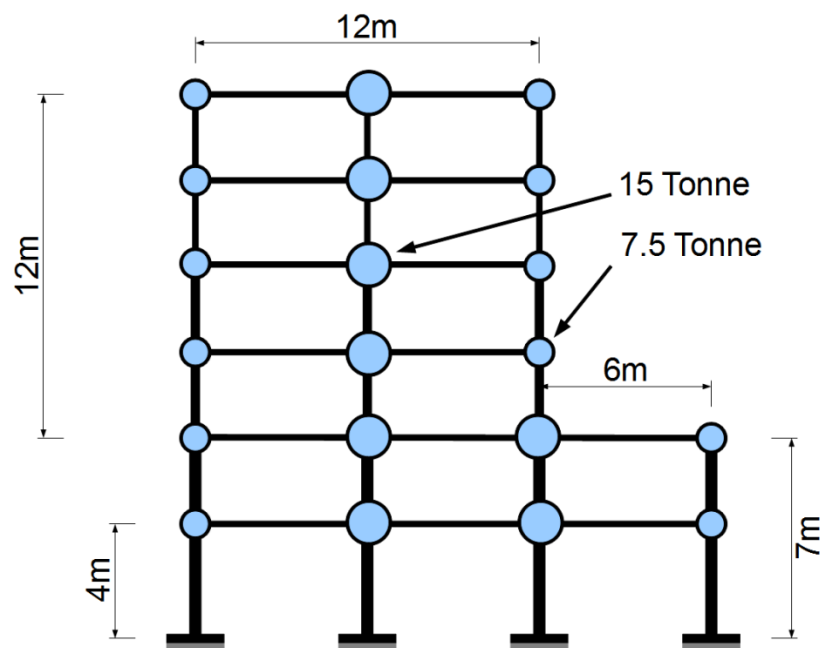


Figure 4-9: 6-storey frame

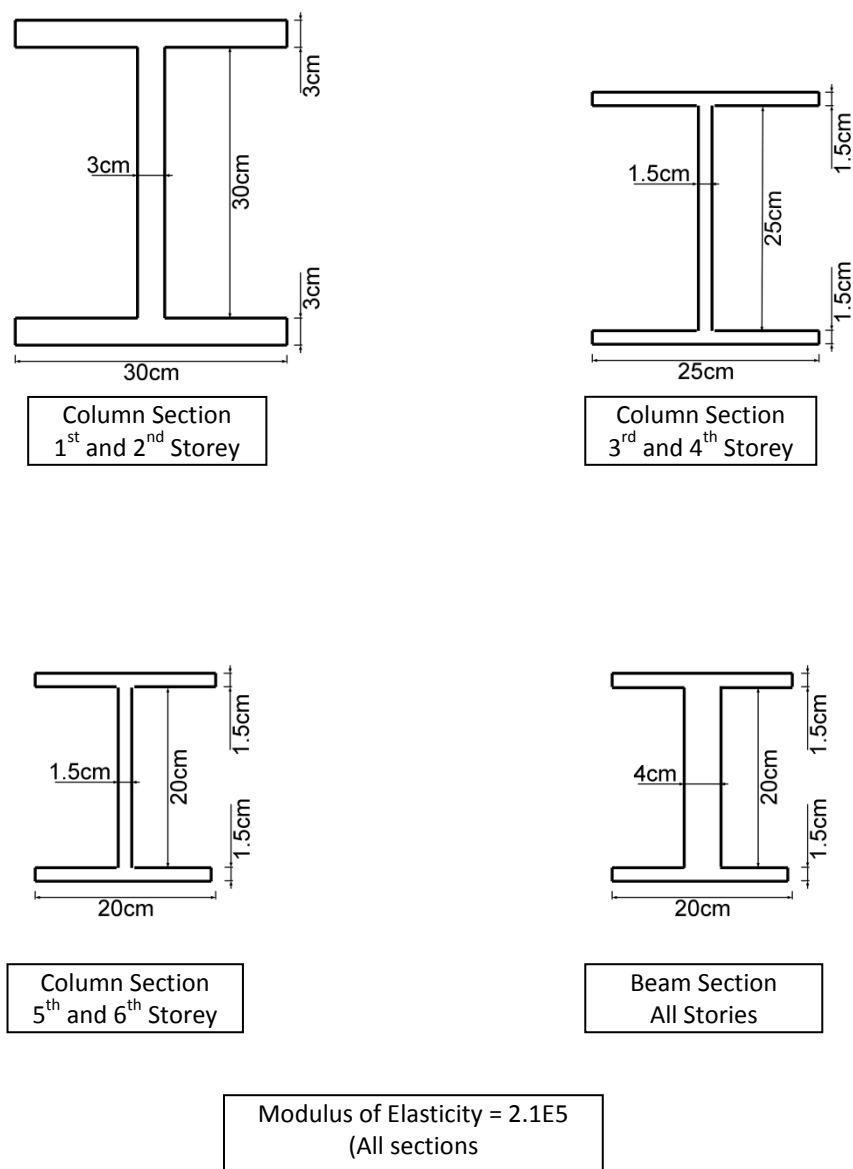
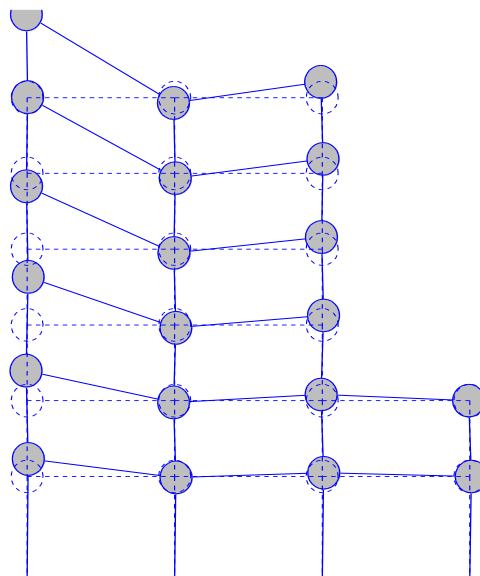
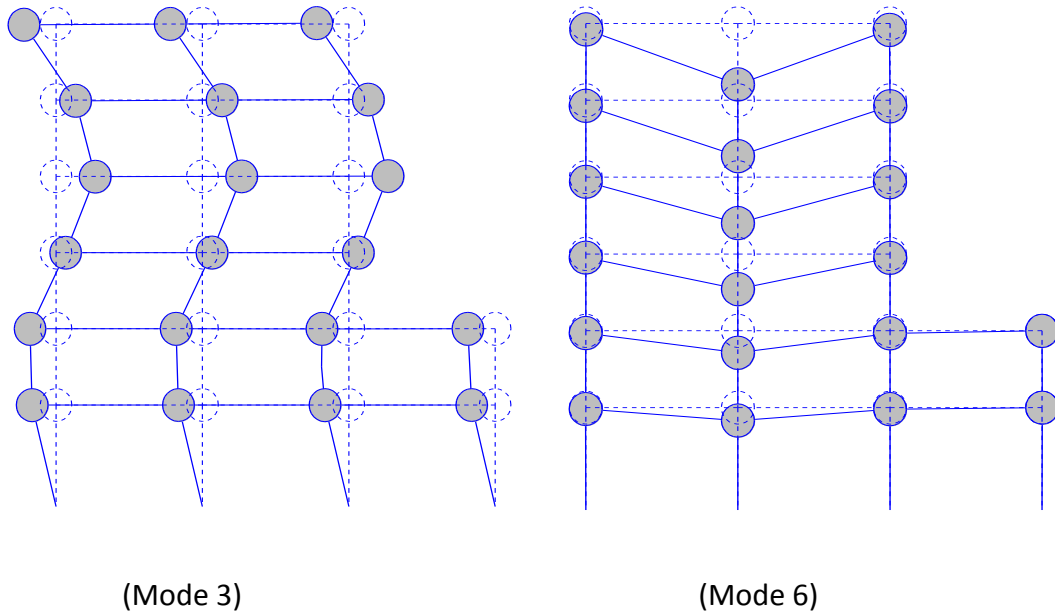


Figure 4-10: Cross-sections for different members of the 6-storey frame



(Mode 9)

Figure 4-11: Modes 3, 6, and 9 of 6-storey frame

For partitioned eigenvalue analysis, the frame domain is decomposed into 3 partitions in addition to the parent structure, and is therefore analysed using 4 parallel processors. The models for the parent structure and the 3 partitions are shown in Figure 4-12. The frame is again analysed for 10 eigenvalues, and the comparison of these values to those obtained the monolithic Lanczos algorithm is shown in Table 4-3.

The results in Table 4-3 clearly show that there is an exact match between the results of the monolithic and partitioned Lanczos algorithm. This fact is further verified by the comparing the mode shapes of the monolithic approach (Figure 4-11) to those from the partitioned approach Figure 4-13 to Figure 4-15. The mode shapes from partitioned analysis are presented separately for clarity, while the grey lines in the mode shape of the parent structure represent the child partition super elements.

Mode	Monolithic Analysis	Partitioned Analysis
1	6.56844	6.56844
2	16.9082	16.9082
3	28.4689	28.4689
4	41.4806	41.4806
5	56.9723	56.9723
6	61.7345	61.7345
7	73.3057	73.3057
8	82.8308	82.8308
9	85.494	85.494
10	161.817	161.817

Table 4-3: Natural frequencies (cycles per second) for 6-storey frame

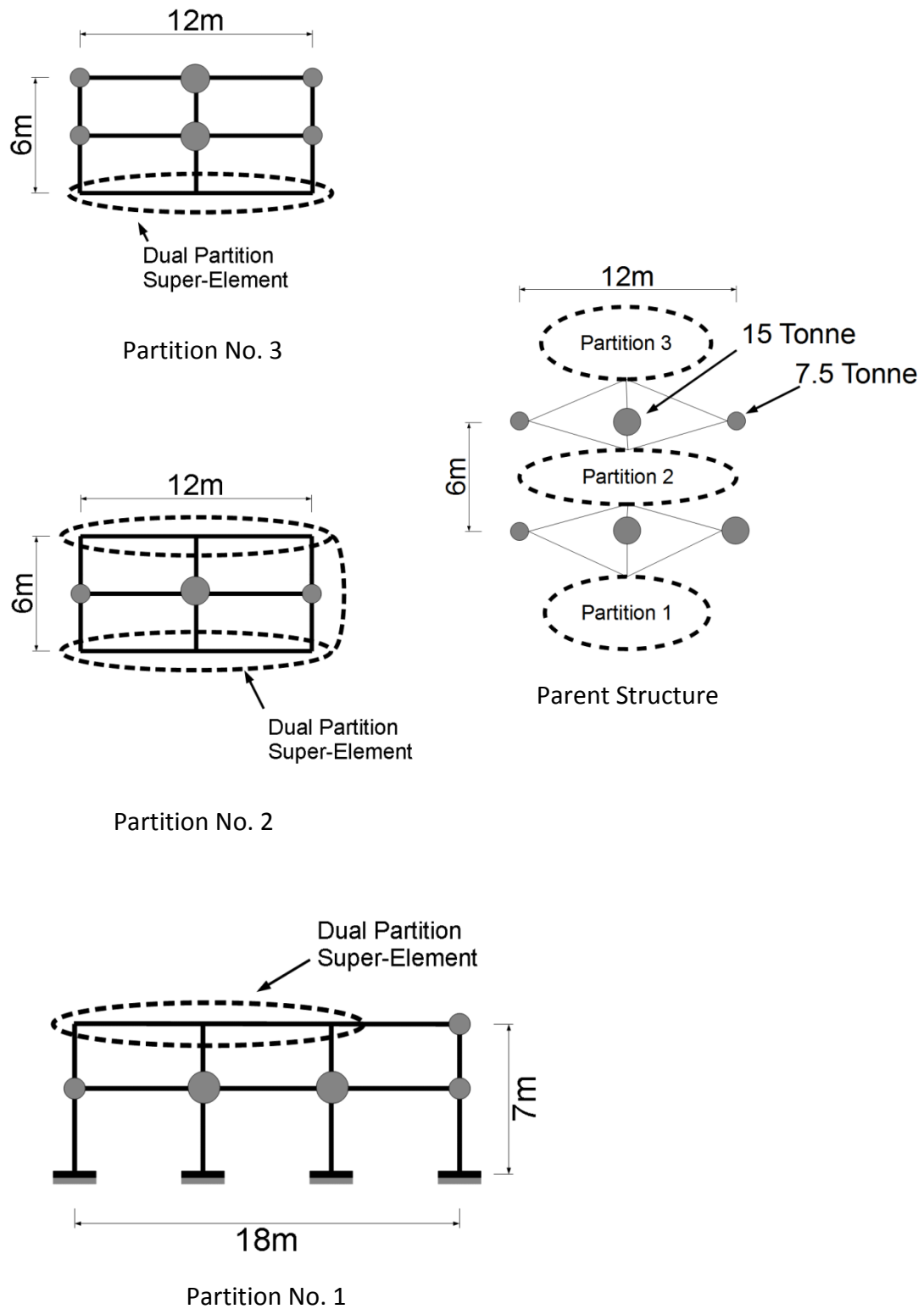
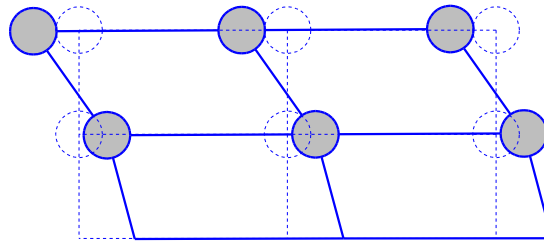
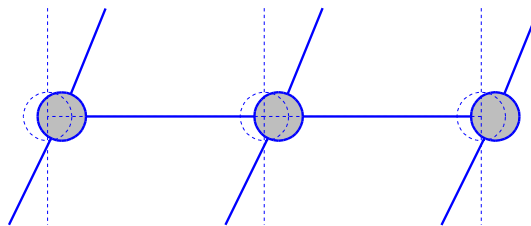


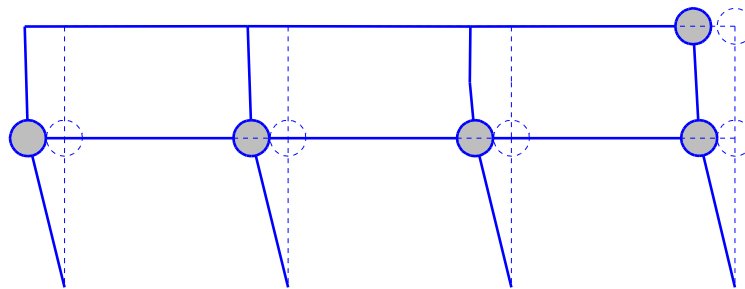
Figure 4-12: Partitioning of 6-storey frame



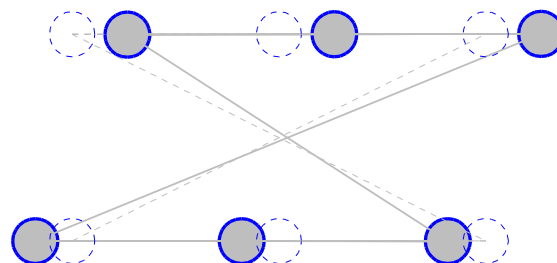
Partition 3



Partition 2

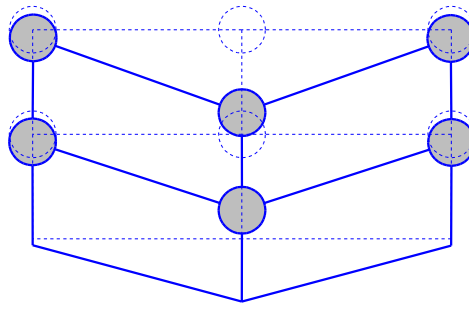


Partition 1

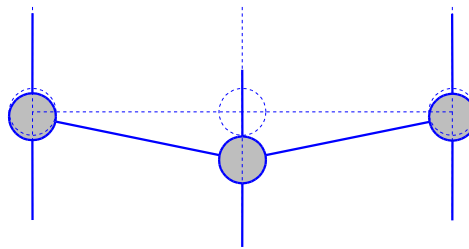


Parent Structure

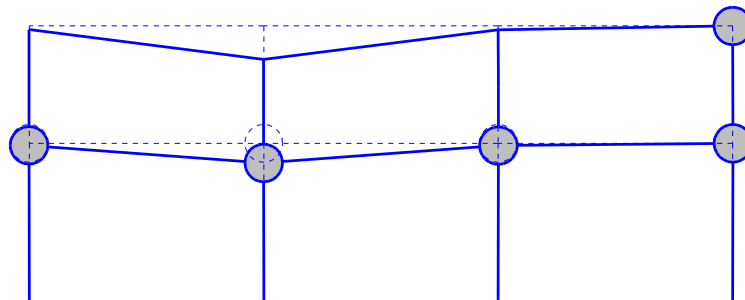
Figure 4-13: Mode shapes of different partitions for mode no. 3



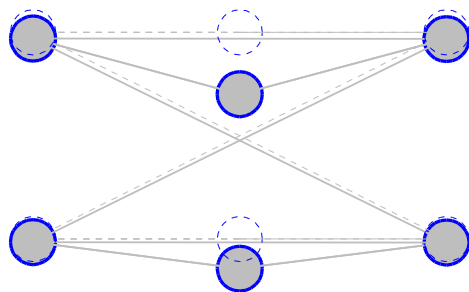
Partition 3



Partition 2

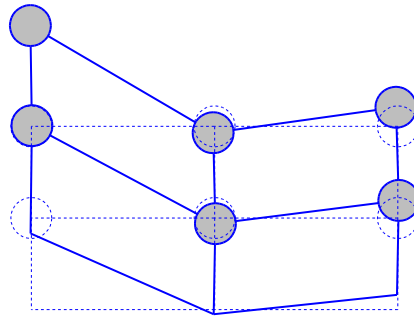


Partition 1

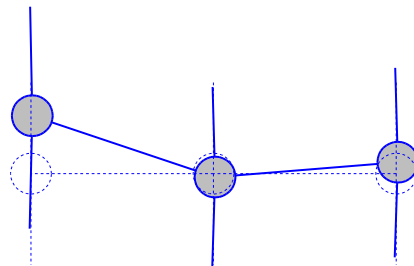


Parent Structure

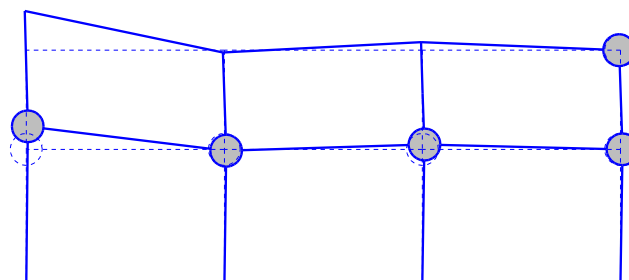
Figure 4-14: Mode shapes of different partitions for mode no. 6



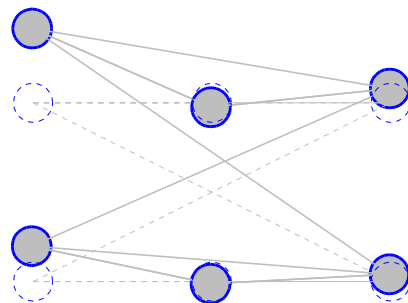
Partition 3



Partition 2



Partition 1



Parent Structure

Figure 4-15: Mode shapes of different partitions for mode no. 9

4.4.4 I-beam with 3D Brick Elements

This example is used to demonstrate the accuracy of the partitioned eigenvalue analysis for 3D continuum analysis, where the same I-beam modelled with 3D brick elements in Example 4 of Section 3.5 is considered (Figure 3—22). Here, however, the beam is subjected to eigenvalue analysis, and two cases are considered. In the first case, the beam is analysed as a monolithic structure computing 20 eigenvalues. In the second case, the beam is divided into 5 partitions and then analysed in parallel computing also 20 eigenvalues.

The comparison of natural frequencies in Table 4-4 shows very minor discrepancies in the higher modes. These discrepancies are attributed to the tolerance used for numerical convergence of the eigenvalues, combined with the use of different randomly generated initial vectors. Figure 4-16 to Figure 4-19 show a favourable comparison of the first 4 mode shapes from the monolithic and partitioned eigenvalue analysis.

Mode	Natural Frequency (rad/sec)		
	Monolithic Analysis	Partitioned Analysis	% Difference
1	347.796	347.78	0.0046
2	439.471	439.449	0.005006
3	860.199	860.022	0.020577
4	1167.26	1167.08	0.015421
5	1233.55	1233.12	0.034859
6	1319.67	1319.25	0.031826
7	1359.86	1356.47	0.24929
8	1585.22	1584.25	0.06119
9	1724.12	1723.18	0.054521
10	2234.48	2233.72	0.034012
11	2243.5	2241.26	0.099844
12	2445.58	2444.23	0.055202
13	2809.37	2808.08	0.045918
14	2983.42	2978.11	0.177984
15	3003.69	3000.33	0.111862
16	3126.03	3116.7	0.298462
17	3308.46	3295.48	0.392328
18	3588.78	3582.33	0.179727
19	3760.36	3755.78	0.121797
20	4072.67	4068.9	0.092568
Average Difference %			0.10435
Standard Deviation			0.105929

Table 4-4: Eigenvalues for I-beam modelled with 3D brick elements

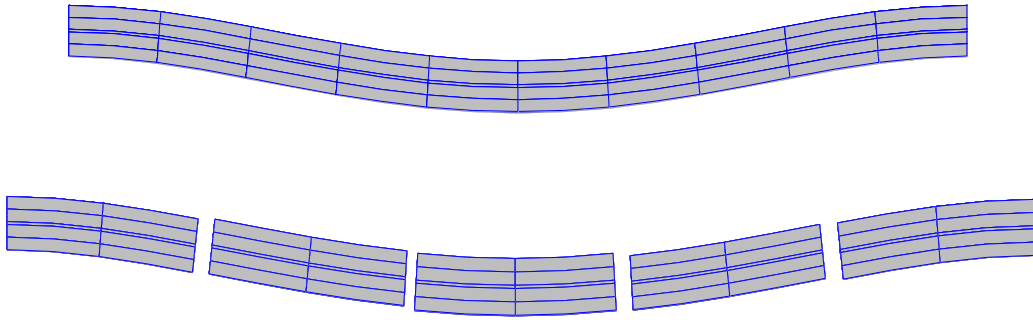


Figure 4-16: Comparison of monolithic and partitioned shapes for mode no. 1

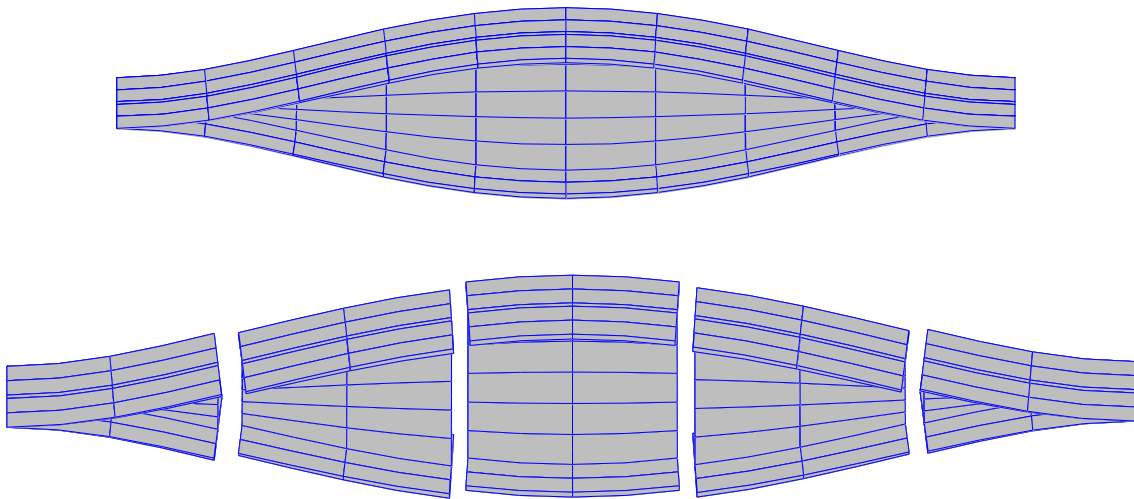


Figure 4-17: Comparison of monolithic and partitioned shapes for mode no. 2

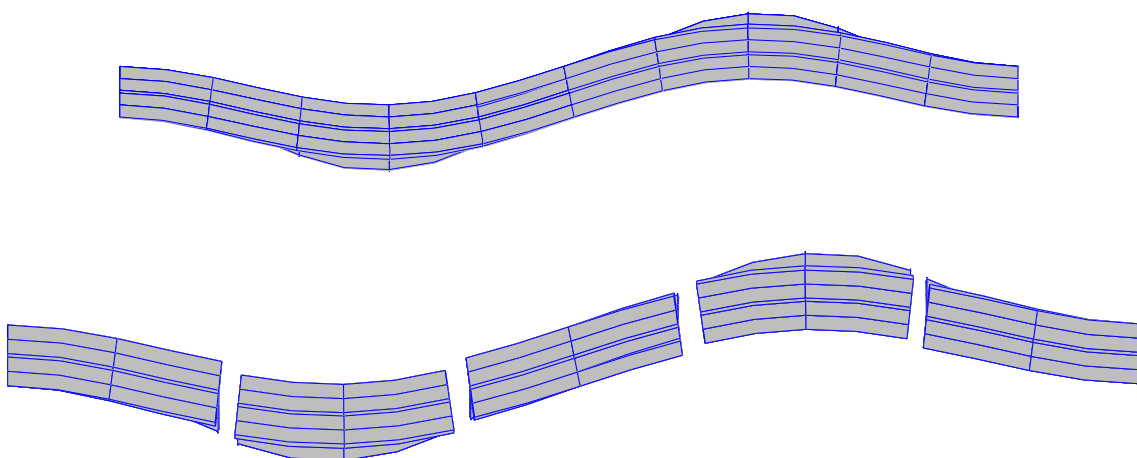


Figure 4-18: Comparison of monolithic and partitioned shapes for mode no. 3

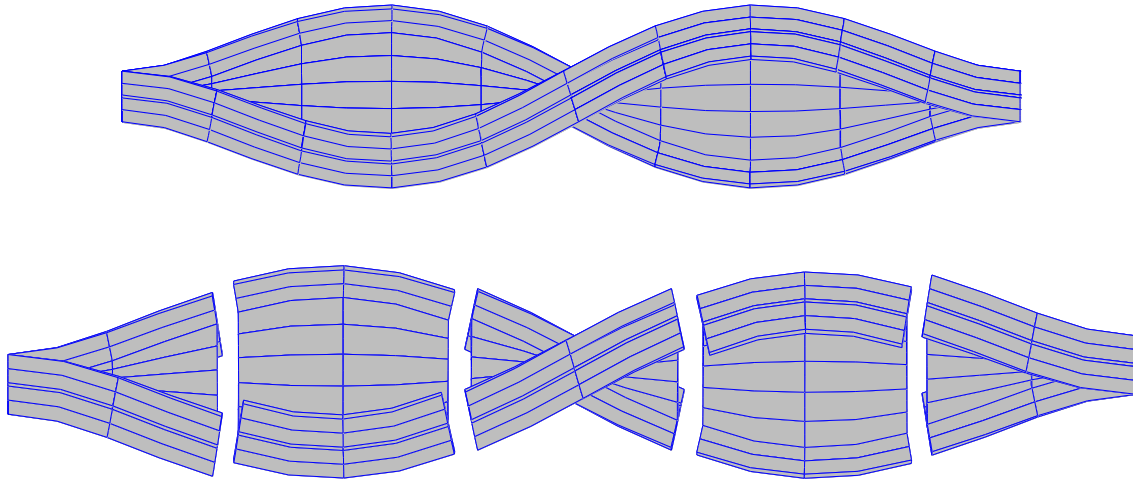


Figure 4-19: Comparison of monolithic and partitioned shapes for mode no. 4

4.4.5 Space Frame made with 3D Brick Elements

This final example is used to demonstrate the computational efficiency of the partitioned Lanczos Algorithm using parallel processing in comparison with the monolithic algorithm running of a single processor. This is the same frame used in Example 6 of Section 3.5.6 to demonstrate the ease of modelling achieved by the domain partitioning approach, where 11 partitions using modular objects with 3D brick elements are employed. Here, however, eigenvalue analysis is undertaken, where the first 10 natural frequencies and corresponding mode shapes are obtained. Both the accuracy of the results and the time taken for the monolithic and partitioned eigenvalue analyses are considered.

The comparison of the natural frequencies obtained by partitioned and monolithic eigenvalue analysis is shown in Table 4-5, where favourable comparison with minor discrepancies is observed. Further favourable comparison is demonstrated for two mode shapes (modes 2 and 4) in Figure 4-20 and Figure 4-21.

Mode	Natural Frequency (rad/sec)		
	Monolithic Analysis	Partitioned Analysis	% Difference
1	173.501	174.436	0.538902
2	217.687	219.851	0.994088
3	605.163	605.168	0.000826
4	977.309	977.636	0.033459
5	1265.12	1266.01	0.070349
6	1474.35	1477	0.17974
7	2023.72	2024.03	0.015318
8	2207.77	2210.22	0.110972
9	2313.08	2313.05	0.001297
10	2867.46	2869.65	0.076374
Average Difference %			0.2
Standard Deviation			0.32

Table 4-5: Natural frequencies for space frame

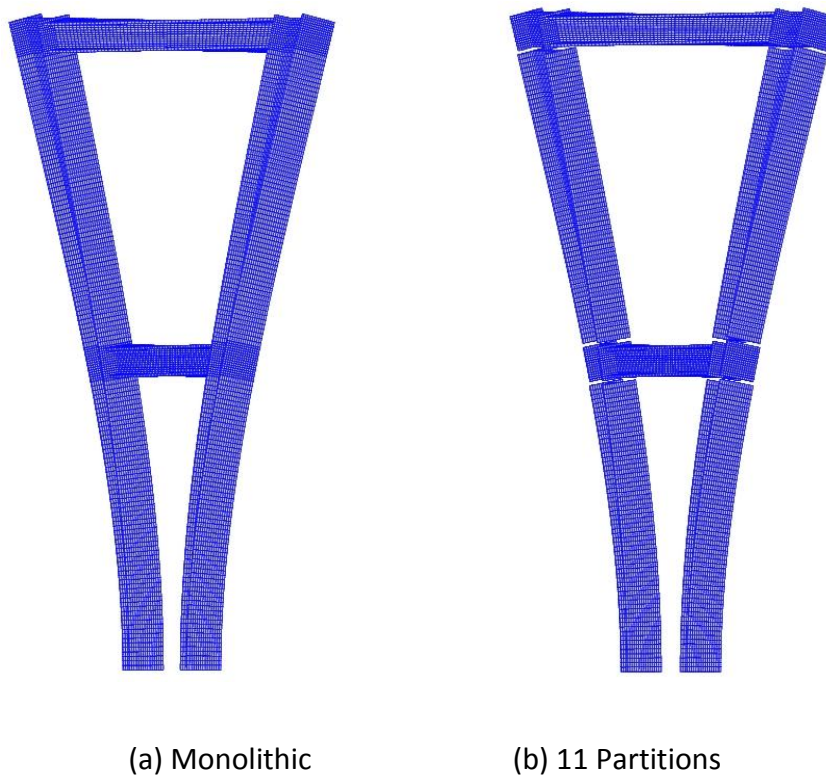


Figure 4-20: Mode shapes for frame: mode 2

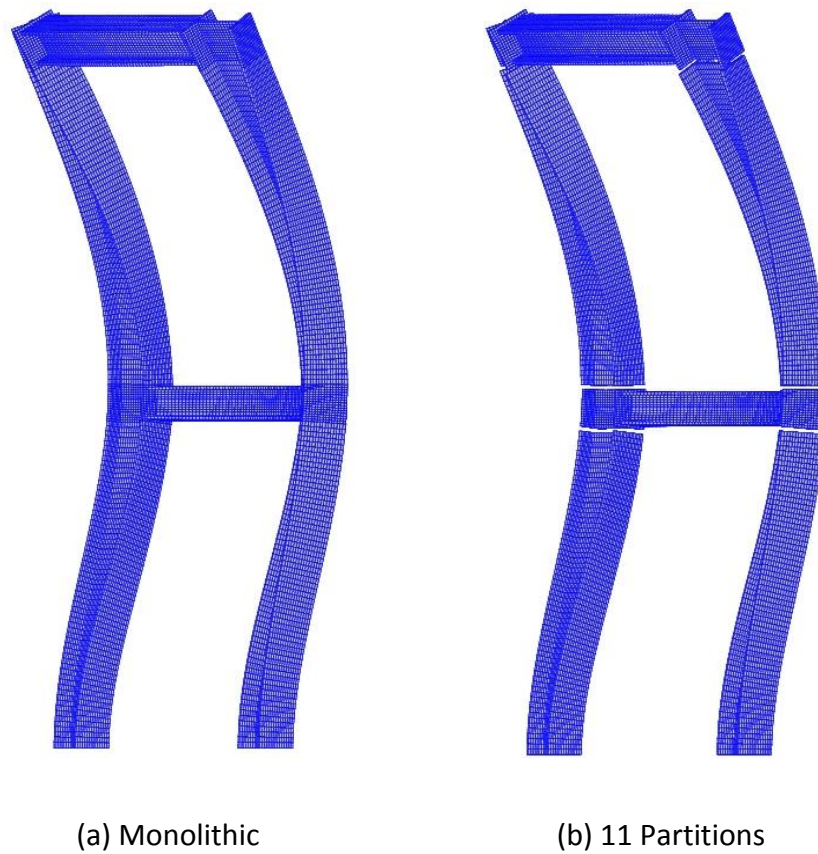


Figure 4-21: Mode shapes for frame: mode 4

It is worth noting that the partitioned model in both Figure 4-20 and Figure 4-21 is depicted so that the partitions are visible separately by using coordinates translation. This property is used for drawing purposes only, and the analysis is not affected.

Finally, a comparison of the wall-clock time taken by the partitioned and monolithic eigenvalue analysis is shown in Figure 4-22, where a speedup factor of over 5.5 is clearly demonstrated. It is worth noting that the eigenvalue analysis is undertaken with 1000 Gauss points per element to magnify the computational demand of eigenvalue analysis and facilitate comparison between monolithic and partitioned analysis.

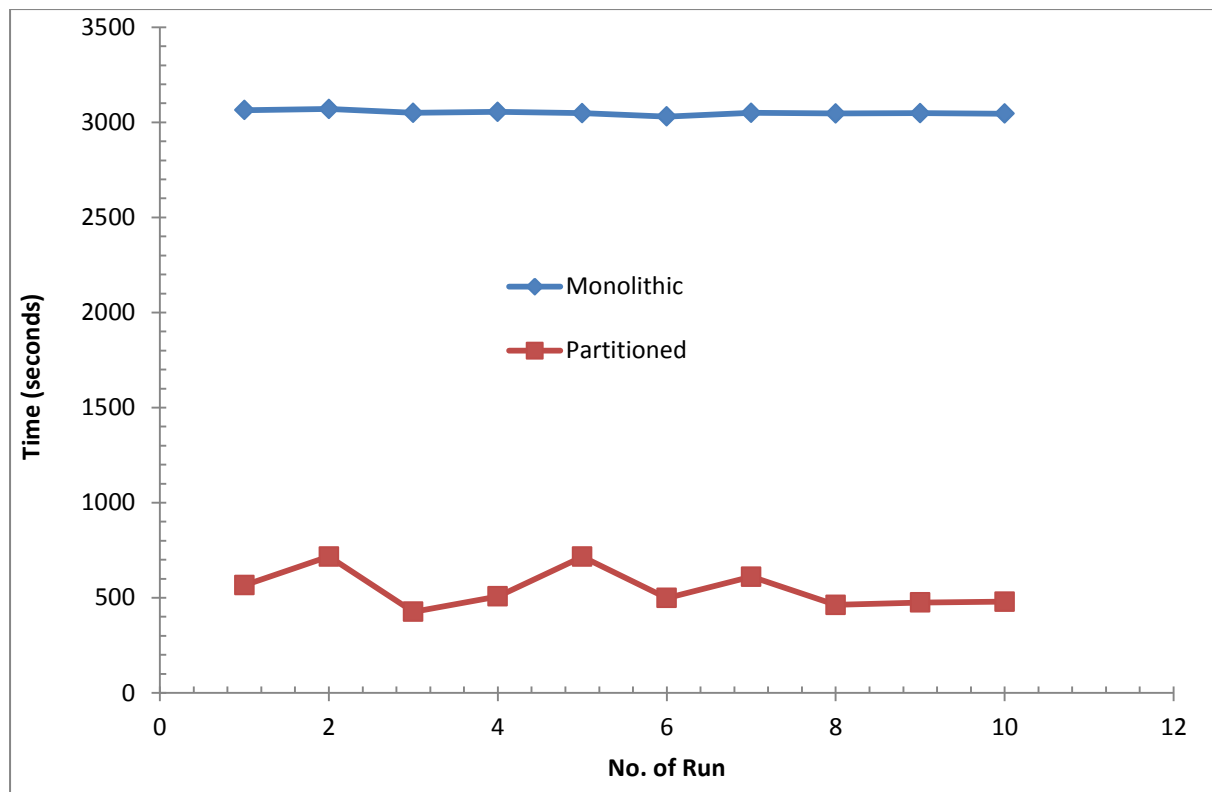


Figure 4-22: Comparison of wall-clock time

Dimensional Coupling

This chapter presents a novel method of dimensional coupling between 1D and 3D elements using a master-slave approach.

5.1 Introduction

The partitioned modelling approach using dual super elements presented in the previous two chapters greatly enhances the computational efficiency of nonlinear structural analysis. Additionally, it offers other benefits such as the possibility to use different time integration schemes, physical field formulations, dimensions, and/or scales inside various partitions. Each of these benefits suits a particular type of structure or a particular type of analysis and is expected to result in further enhancement to computational efficiency and ease of modelling.

It is impractical to formulate methods for all of the above mentioned options in a single study; however, dimensional coupling is considered in this work because it is an essential part of mixed-dimensional partitioned modelling. Using dimensional coupling, a framed structure is largely modelled with reduced dimension (1D) elements, except for the more critical parts which are modelled with elements of higher dimension (3D). It is demonstrated with the help of examples that the use of such dimensional coupling produces results closer in accuracy to the full 3D models with a computational cost that is comparable to that of reduced 1D models.

For the use of differently dimensioned elements in different parts of a structure, it is necessary to develop interface elements between such elements. These interface elements may include a wide range of dimensional couplings (i.e. 1D-2D, 1D-3D, 2D-3D). The scope of this study is limited to the coupling between 3D brick elements and 1D beam-column elements, hence an interface element for 1D-3D coupling which accounts for geometric nonlinearity is presented here. This interface element has a single ‘master’ 1D node with 6

degrees of freedom on one side. On the other side, it is connected to all the nodes on a cross-section made of 3D brick elements regardless of the shape of the cross-section. These nodes on the 3D side have 3 degrees of freedom each and are referred to as the ‘slave nodes’.

Although the 1D-3D coupling element can be implemented with conventional monolithic analysis using serial computations, in order to achieve its full potential in terms of computational efficiency, it is formulated for use with parallel analysis in conjunction with the domain partitioning approach presented in Chapter 3. In this case, the 3D nodes in a child partition that would have been previously assigned to an original partitioned boundary are now connected to one or more 1D-3D coupling elements with the 1D master nodes designated as the new partitioned boundary nodes, as shown in Figure 5-1.

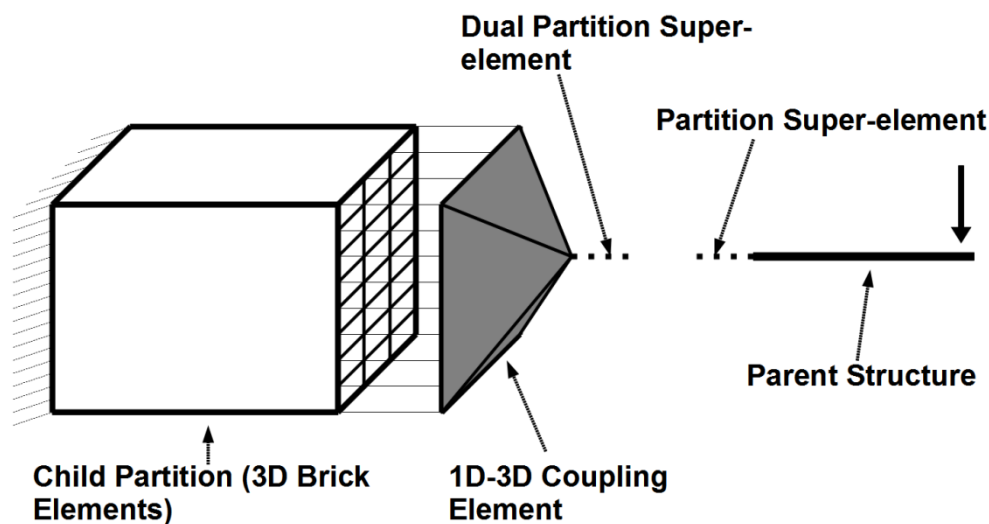


Figure 5-1: Master-slave (1D-3D) coupling element

5.2 Method Formulation

Since the dimensional coupling elements are present inside the child partition and the 1D master nodes of these elements are now connected to the partitioned boundary, there is no change in the solution method as far as the parent structure is concerned. It assembles the

tangent stiffness matrix and the resistance forces vector using partition super elements as elaborated in Section 3.2.1. The solution method for the child partitions on the other hand now involves inter-dimensional transformations at the end of forward elimination before the tangent stiffness matrix and the resistance forces vectors are communicated to the parent structure. Similarly, the incremental displacements received from the parent structure are transformed into those for the original partitioned boundary nodes before completion of the backward substitution. This change in the solution method for the child partition is elaborated below.

For the sake of generality, assume that the 3D child partition consists of ' n ' degrees of freedom, of which ' h ' degrees of freedom are associated with the original partitioned boundary. Furthermore, assume that there are ' k ' dimensional coupling elements each having a single 1D master node and several slave nodes with a total of ' h_i ' degrees of freedom, where $i=1 \rightarrow k$. An example child partition using four dimensional coupling elements is shown in Figure 5-2, where $k=4, i=1 \rightarrow 4$. It is worth noting here that the overall ' h ' degrees of freedom would have been assigned to the partitioned boundary if dimensional coupling is not used. The frontal solution can proceed in the same way as before, where the 3D nodes on the original partitioned boundary are retained in the Grandpa at the end of the forward elimination process. In this sense, the 1D master nodes of the coupling elements, which are at the new partitioned boundary, are not considered as part of the frontal solution in the 3D child partition.

When all the internal degrees of freedom of the child partition are eliminated in the forward pass, the system of equations for the partition is given as:

$$\begin{bmatrix} {}_s K'_{n-h+1,n-h+1} & {}_s K'_{n-h+1,n-h+2} & \cdots & {}_s K'_{n-h+1,n} \\ {}_s K'_{n-h+2,n-h+1} & {}_s K'_{n-h+2,n-h+2} & \cdots & {}_s K'_{n-h+2,n} \\ \vdots & \vdots & \ddots & \vdots \\ {}_s K'_{n,n-h+1} & {}_s K'_{n,n-h+2} & \cdots & {}_s K'_{n,n} \end{bmatrix} \begin{Bmatrix} \delta_s U_{n-h+1} \\ \delta_s U_{n-h+2} \\ \vdots \\ \delta_s U_n \end{Bmatrix} = \begin{Bmatrix} \delta_s R'_{n-h+1} \\ \delta_s R'_{n-h+2} \\ \vdots \\ \delta_s R'_n \end{Bmatrix} \quad 5-1$$

where the left subscript indicates that these are the values for the slave nodes on the 3D partition, each of which has 3 degrees of freedom.

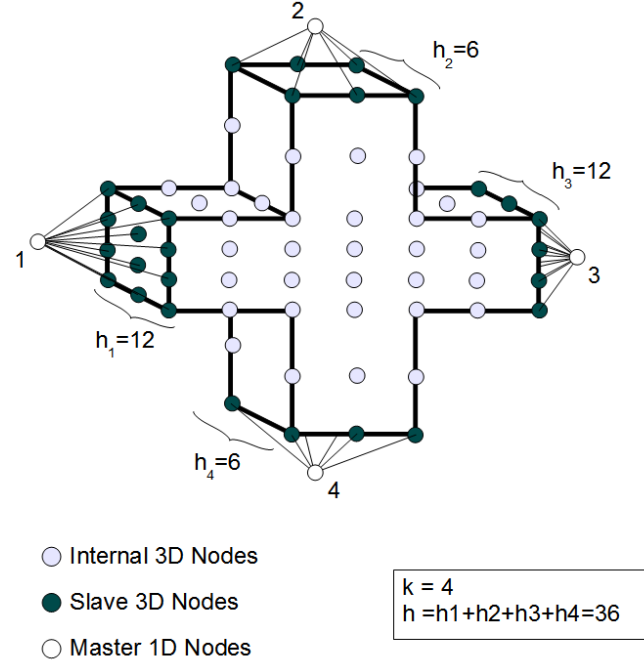


Figure 5-2: An example child partition with 4 dimensional coupling elements

Each dimensional coupling element has a distinct set of slave nodes normally associated with a cross-section; by definition slave nodes can have only one master node, and as such the sets of slave nodes associated with the different coupling elements are mutually exclusive.

The coupling elements transform the condensed stiffness matrix and the right hand side vector into those for the 1D master nodes:

$${}_s K_{i,j}^c[h_i \times h_j] \Rightarrow {}_m K_{i,j}^c[6 \times 6] \quad 5-2$$

and,

$${}_s R_{i\{h_i\}}^c \Rightarrow {}_m R_{i\{6\}}^c \quad 5-3$$

where the left subscript indicates slave or master node, and the symbol ' $\Rightarrow$ ' means 'transformed into'. This transformation is elaborated upon in the next section.

The dual partition super element that wraps around the 1D master nodes then sends ${}_mK$ and ${}_mR$ to the partition super element that represents this partition in the parent structure. The parent structure completes the assembly of its stiffness matrix and performs a forward elimination and a backward substitution. It then sends the iterative increment of displacements $\delta_m U$ for the nodes of the relevant partition super element to the dual partition super element. Upon receiving the iterative increment of displacements, the coupling element transforms those to iterative increments of displacements for the 3D slave nodes $\delta_s U$:

$$\delta_m U_{i_{\{6\}}} \Rightarrow \delta_s U_{i_{\{h_i\}}} \quad 5-4$$

This transformation is again elaborated upon in the next section.

The child partition then carries out backward substitution for the iterative displacements of the internal 3D nodes in the same way as discussed in Section 3.2.2.

5.3 Inter-dimensional Transformations

As indicated in the previous section, there are a total of 3 inter-dimensional transformations required by Eq. 5—2 to 5—4 for the formulation of the 1D-3D dimensional coupling element. These transformations are presented below.

5.3.1 Displacement Transformation

As stated above, there exists a master-slave relationship between the 1D node and the 3D nodes resulting in the determination of the displacements of the 3D nodes as a function of the displacements of the 1D node, denoted for the iterative increments by:

$$\delta_s U = f(\delta_m U) \quad 5-5$$

The transformation of the 1D master node displacements into those of the slave nodes is based on a large displacement analysis method for 3D frames, proposed by Izzuddin (1993). The nodal displacements and rotations of the 1D master node describe compatible displacements of the 3D nodes, presenting an approximation of the displacements of the original 3D partitioned boundary.

In the global reference system, the displacement vector for a 1D master node consists of 6 nodal freedoms, 3 translations and 3 rotations:

$${}_mU = \begin{Bmatrix} {}_mU^t \\ {}_mU^r \end{Bmatrix}, \quad {}_mU^t = \begin{Bmatrix} u \\ v \\ w \end{Bmatrix}, \quad {}_mU^r = \begin{Bmatrix} \alpha \\ \beta \\ \gamma \end{Bmatrix} \quad 5-6$$

which are incremental relative to the previous equilibrium configuration, as shown in

Figure 5-3.

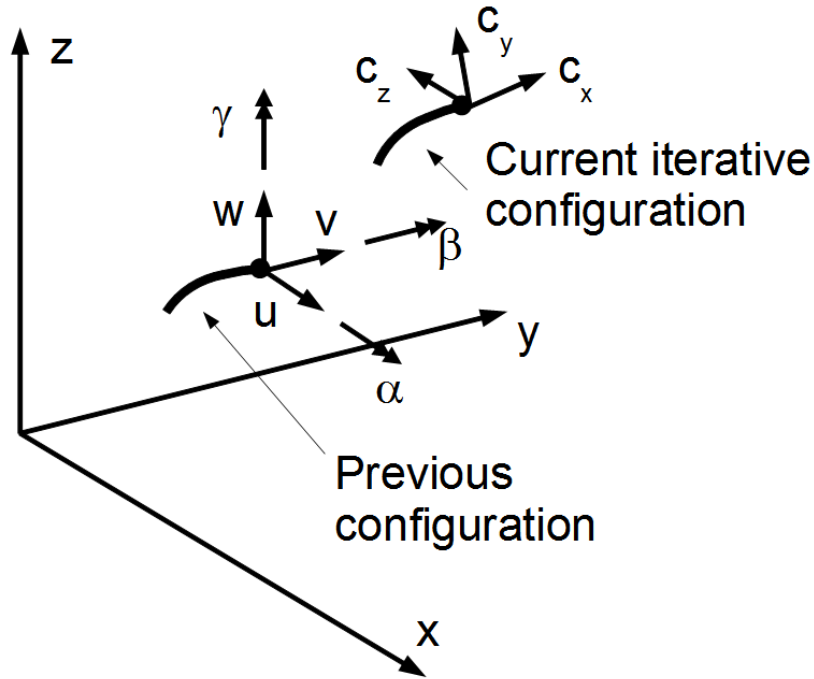


Figure 5-3: Global freedoms of 1D master node

Treating the 3D partitioned boundary similar to a rigid cross-section, the translations/rotations of the 1D master node could be identified with translations/rotations of the 3D cross-section. Accordingly, the 3D cross-section is subject to translations (Figure 5-4) and rotations (Figure 5-5), leading to the following evaluation of the displacements for a 3D slave node k :

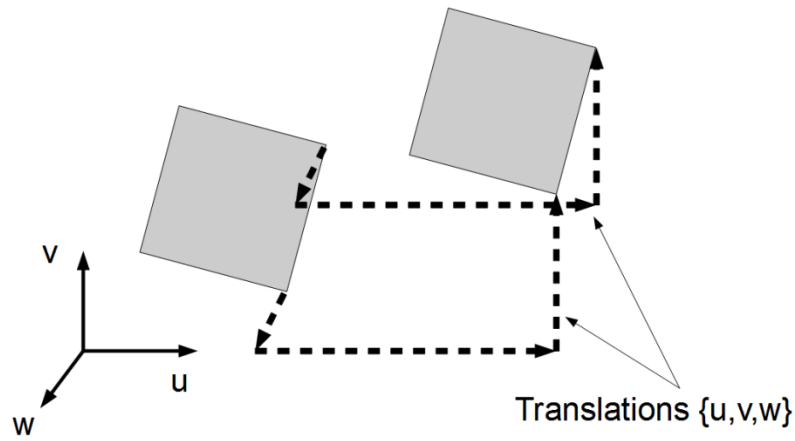


Figure 5-4: Translations applied to slave nodes

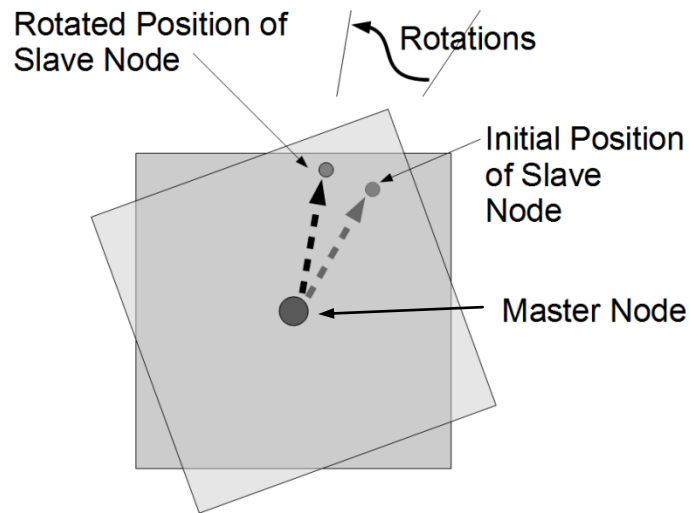


Figure 5-5: Rotations applied to a slave node

$${}_sU_k = {}_mU^t + {}_mT^r \begin{Bmatrix} {}_sX_k \\ {}_sY_k \\ {}_sZ_k \end{Bmatrix} \quad 5-7$$

where $\{{}_sX_k, {}_sY_k, {}_sZ_k\}$ are the global coordinates of slave node k relative to the master node, and ${}_mT^r$ is a second-order rotation transformation matrix proposed by Izzuddin (1993, 2001) which depends on the three master rotational freedoms ${}_mU^r$:

$${}_mT^r(\alpha, \beta, \gamma) = \begin{bmatrix} -\frac{\beta^2 + \gamma^2}{2} & -\gamma + \frac{\alpha\beta}{2} & \beta + \frac{\alpha\gamma}{2} \\ \gamma + \frac{\alpha\beta}{2} & -\frac{\alpha^2 + \gamma^2}{2} & -\alpha + \frac{\beta\gamma}{2} \\ -\beta + \frac{\alpha\gamma}{2} & \alpha + \frac{\beta\gamma}{2} & -\frac{\alpha^2 + \beta^2}{2} \end{bmatrix} \quad 5-8$$

With the total incremental displacements of the 3D slave node given by Equation 5-7, the iterative corrections can be readily obtained from those of the master node as:

$$\delta_s U_k = \delta_m U^t + \delta_m T^r \begin{Bmatrix} {}_sX_k \\ {}_sY_k \\ {}_sZ_k \end{Bmatrix} \quad 5-9$$

where:

$$\delta_m T^r = {}_mT^r(\alpha + \delta\alpha, \beta + \delta\beta, \gamma + \delta\gamma) - {}_mT^r(\alpha, \beta, \gamma) \quad 5-10$$

It is noted that since the rotation transformation in Equation 5-10 is based on incremental rotations relative to the last equilibrium configurations, it is important that the global coordinates of the slave node relative to the master node are updated after convergence at the end of the current equilibrium step:

$$\begin{Bmatrix} {}_sX_k \\ {}_sY_k \\ {}_sZ_k \end{Bmatrix} = \begin{Bmatrix} {}_sX_k \\ {}_sY_k \\ {}_sZ_k \end{Bmatrix} + {}_sU_k - {}_mU^t \quad 5-11$$

5.3.2 Forces Transformation

In the system local to the 3D partition and hence the coupling element, the resistance forces associated with a slave node k contribute to the resistance forces of the master node, which are thus determined from the principle of virtual work:

$${}_mR^c = \sum_{k=1}^{n_s} T_k^T {}_sR_k^c \quad 5-12$$

where n_s is the number of slave nodes associated with the considered master node, and T_k represents the derivative of the displacements of slave node k with respect to the displacements/rotations of the master node:

$$T_k = \left[\frac{\partial_s U_k}{\partial_m U^T} \right] = \left[I_{[3 \times 3]} \quad \vdots \quad {}_r T'_k [3 \times 3] \right] \quad 5-13$$

In which I is the identity matrix and ${}_r T'_k$ can be obtained from Equations 5-7 and 5-8:

$${}_r T'_k = \left[\frac{\partial_s U_k}{\partial_m U^T} \right] = \begin{bmatrix} \frac{\beta_s y_k + \gamma_s z_k}{2} & -\beta_s x_k + \frac{\alpha_s y_k}{2} + {}_s z_k & -\gamma_s x_k - {}_s y_k + \frac{\alpha_s z_k}{2} \\ \frac{\beta_s x_k}{2} - \alpha_s y_k - {}_s z_k & \frac{\alpha_s x_k + \gamma_s z_k}{2} & {}_s x_k - \gamma_s y_k + \frac{\beta_s z_k}{2} \\ \frac{\gamma_s x_k}{2} + {}_s y_k - \alpha_s z_k & -{}_s x_k + \frac{\gamma_s y_k}{2} - \beta_s z_k & \frac{\alpha_s x_k + \beta_s y_k}{2} \end{bmatrix} \quad 5-14$$

5.3.3 Tangent Stiffness Transformation

The condensed tangent stiffness matrix is defined as the derivative of the condensed resistance forces with respect to the corresponding nodal displacements at the partitioned boundary. For the slave nodes at the original partitioned boundary, the condensed tangent stiffness matrix is defined as:

$${}_s K^c = \left[\frac{\partial_s R^c}{\partial_s U^T} \right] \quad 5-15$$

while for the master nodes at the new partitioned boundary, the reduced condensed tangent stiffness matrix is:

$${}_m K^c = \left[\frac{\partial_m R^c}{\partial_m U^T} \right] \quad 5-16$$

Considering Equation 5-16, using chain differentiation principles, and noting that ${}_r T'$ is dependent on master node rotations, it can be shown that the 6x6 submatrix of ${}_m K^c$ related to master nodes i and j on the new partitioned boundary can be obtained as the

transformation of the associated 3x3 submatrices of ${}_s K^c$ related to the associated slave nodes:

$${}_m K_{i,j}^c = \left(\sum_{k=1}^{n_{s,i}} \sum_{l=1}^{n_{s,j}} T_k^T {}_s K_{k,l}^c T_l \right) + {}_g K_{i,j}^c \quad 5-17$$

In this expression, $n_{s,i}$ and $n_{s,j}$ are the number of 3D slave nodes associated with 1D master nodes i and j , respectively, T is given by Equation 5-14, and ${}_s K_{k,l}^c$ is a 3x3 submatrix associated with slave node k of master node i and slave node l of master node j . On the other hand, ${}_g K_{i,j}^c$ is a geometric stiffness matrix which arises from the quadratic kinematics presented by Equation 5-12 and 5-13:

$${}_g K_{i,j \neq i}^c = 0_{[6 \times 6]}, \quad {}_g K_{i,i}^c = \begin{bmatrix} 0_{[3 \times 3]} & 0_{[3 \times 3]} \\ 0_{[3 \times 3]} & {}_r K_{i,i}^c \end{bmatrix} \quad 5-18$$

where:

$${}_r K_{i,i}^c = \sum_{k=1}^{n_{s,i}} \begin{bmatrix} -{}_s y_k {}_s R_{k,2} - {}_s z_k {}_s R_{k,3} & \frac{{}_s y_k {}_s R_{k,1} + {}_s x_k {}_s R_{k,2}}{2} & \frac{{}_s z_k {}_s R_{k,1} + {}_s x_k {}_s R_{k,3}}{2} \\ \frac{{}_s y_k {}_s R_{k,1} + {}_s x_k {}_s R_{k,2}}{2} & -{}_s x_k {}_s R_{k,1} - {}_s z_k {}_s R_{k,3} & \frac{{}_s z_k {}_s R_{k,2} + {}_s y_k {}_s R_{k,3}}{2} \\ \frac{{}_s z_k {}_s R_{k,1} + {}_s x_k {}_s R_{k,3}}{2} & \frac{{}_s z_k {}_s R_{k,2} + {}_s y_k {}_s R_{k,3}}{2} & -{}_s x_k {}_s R_{k,1} - {}_s y_k {}_s R_{k,2} \end{bmatrix} \quad 5-19$$

5.4 Implementation

The dimensional coupling interface element has been implemented within ADAPTIC [Izzuddin, 1991]. It can be wrapped around the boundary nodes of any 3D child partition and, its master node is then assigned to the new 1D partitioned boundary, where it is associated with a single node in the parent structure. It is worth noting here that one dimensional coupling interface element is sufficient per cross-section of the 3D child partition, provided such cross-sections can be assumed to remain rigid.

If a 3D child partition has two or more cross-sections at the partitioned boundary, as illustrated for example in Figure 5-6, the original partitioned approach considers direct coupling between the 3D nodes in the parent and child partitions at the partition boundary.

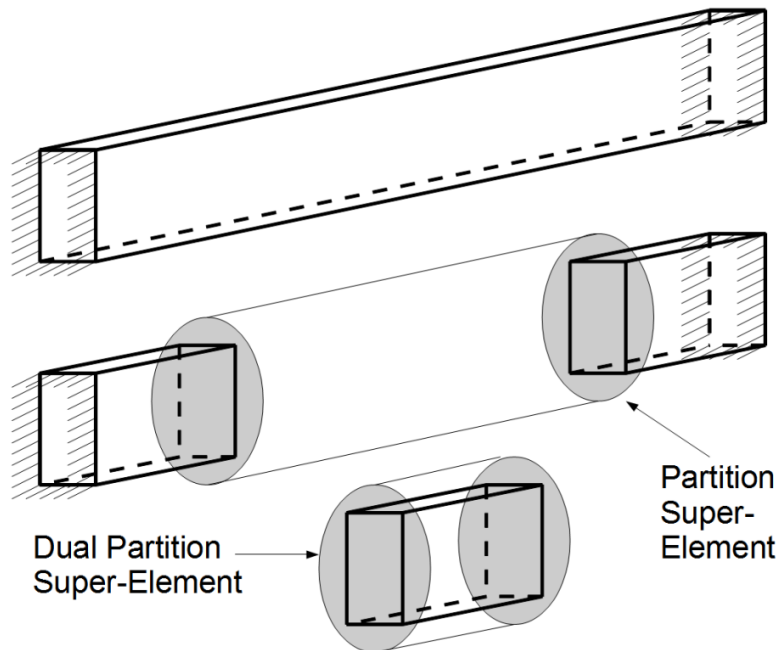


Figure 5-6: Partitioned model of 3D beam with 2 cross-sections on partitioned boundary

Using the mixed dimensional coupling interface element, the same beam can be modelled as a 1D parent structure in combination with a child partition still modelled using 3D continuum elements. In this case, the 1D parent structure model would have a partition super element connected to two 1D nodes, with dual super element connected to the corresponding 1D nodes of two 1D-3D dimensional coupling interface elements, each connected to one cross-section as shown in Figure 5-7.

As stated in Section 5.1, the dimensional coupling interface element is connected to all the nodes at the cross-section on the one side and a single master node on the other side. This single master node is then connected to the dual partition super element. For the implementation with ADAPTIC, this single master node is defined as a non-structural node, and its coordinates are ideally chosen as the geometric centroid of the cross-section.

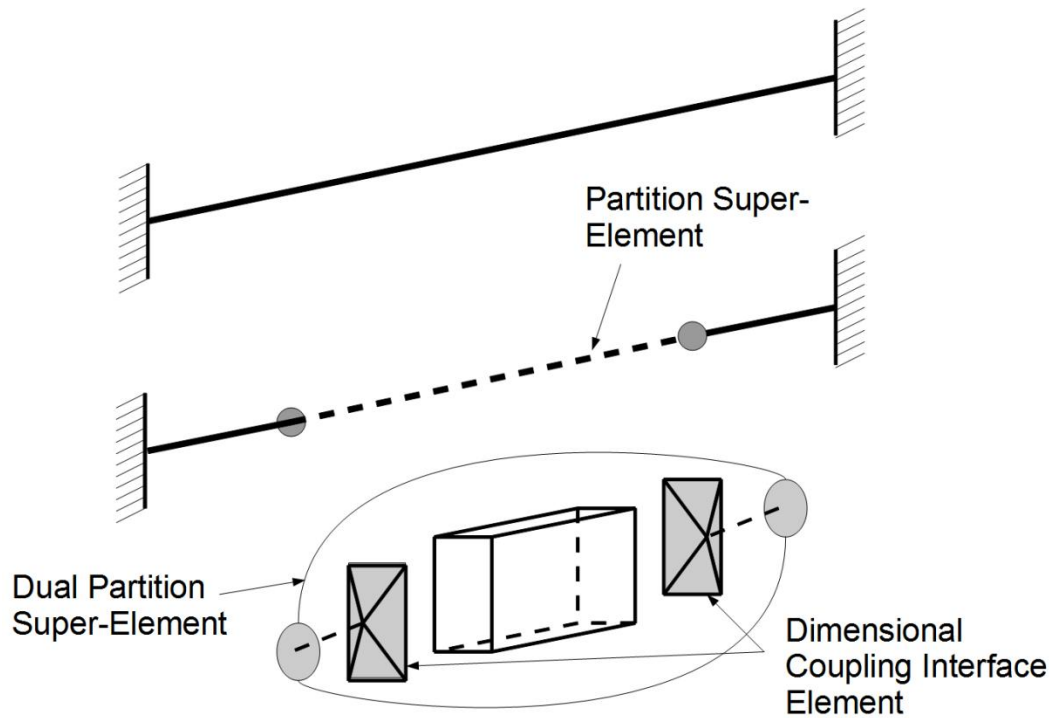


Figure 5-7: Dimensional coupling interface elements used for a beam

The data syntax for defining groups of dimensional coupling interface elements is given below:

groups	
type.of.element =	mslv
grp.name	slave.nodes
<token>	<int>
⋮	⋮

The element type 'mslv' is used to designate the dimensional coupling interface element. 'grp.name' is the name of the element group and can be an arbitrary string consisting of alpha-numeric characters, whereas 'slave.nodes' is an integer value of the number of nodes on the 3D cross-section. This number does not include the master node as it is known by default that every element of this type will have one and only one master node. The master node is assigned along with the slave nodes to the coupling element in the element connectivity module.

The `partitioned.boundary` module has also been modified slightly, where for the case employing dimensional coupling interface elements, the partitioned boundary is defined in terms of the master-slave coupling elements instead of the partitioned boundary nodes. Since the connectivity of the coupling element to slave nodes is already defined, this approach easily identifies the slave nodes on the original partitioned boundary.

```
partitioned.boundary
  master.slave.elm.name
  <token>
  < token>
  ⋮
```

Example input files using dimensional coupling are presented in Appendix B.

The use of dimensional coupling interface element is not limited to models where the parent structure is modelled using 1D elements and partitions are modelled using 3D brick elements. This coupling element can also be used in cases where the parent structure consists entirely of partition super elements. In such cases, the dimensional coupling is used to reduce the size of the parent structure as well as the communication overhead between different processes. This can attain significant computational efficiency, as demonstrated in the following section.

5.5 Verification Examples

This section presents several verification examples that demonstrate that the proposed mixed dimensional coupling approach provides an accuracy which is close to that offered by full 3D models but at a much reduced computational demand. The examples presented in the following subsections are all geometrically nonlinear.

5.5.1 Cantilever Beam

The first example is a simple cantilever beam, which is considered to study the accuracy and computational efficiency of the proposed mixed dimensional coupling approach. As shown

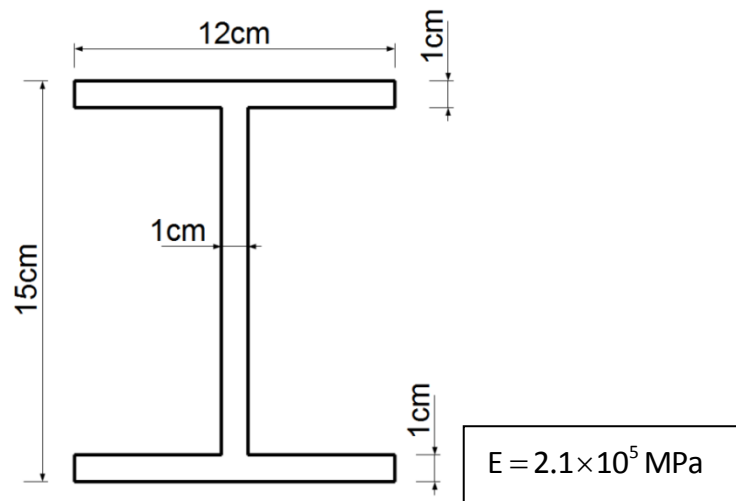
in Figure 5-8, the cantilever span is 5 meters, with the self-weight applied as an initial load and a tip point load applied as a varying load.

The beam was modelled in 6 different ways ranging from a full 1D model to a full 3D model, with the other models having different lengths modelled in 3D and the remaining length modelled in 1D. For all cases, the length of the 1D elements or the 3D partitions is 0.5 m. The details of the analysis cases along with the portions of length modelled in 1D or 3D are provided in Table 5-1. It should be noted that the difference between Case 2 and Cases 3-7 is that the latter cases use dimensional coupling (i.e. the parent structure consists of a single 1D node per cross-section), whereas Case 2 does not use dimensional coupling (i.e. partitioned boundaries at cross-sections consist of 3D nodes). It should also be noted that whenever a portion of the beam is modelled with 1D elements they are always part of the 1D parent structure, and that the 3D partitions always start from the fixed end.

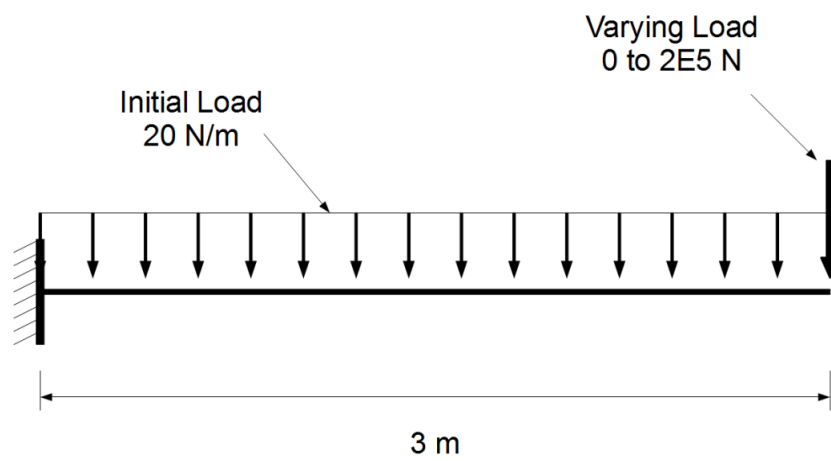
The final deflected shapes for Cases 1 to 6 are shown in Figure 5-9 and Figure 5-10. The deflected shape for Case 7 was exactly the same as that for Case 2. Figure 5-11 shows a comparison of the tip load-deflection response, and Table 5-2 shows the respective wall-clock times taken for the nonlinear analysis using 384 load steps.

Case	Portion modelled in 1D (m)	Portion modelled in 3D (m) [No. of partitions]
1	3	0 [0]
2	0	3 [6]
3	2.5	0.5 [1]
4	2	1 [2]
5	1.5	1.5 [3]
6	1	2 [4]
7	0	3 [6]

Table 5-1: Details of analysis cases for cantilever beam example



(a)



(b)

Figure 5-8: Cantilever beam (a) cross-section (b) span and loading

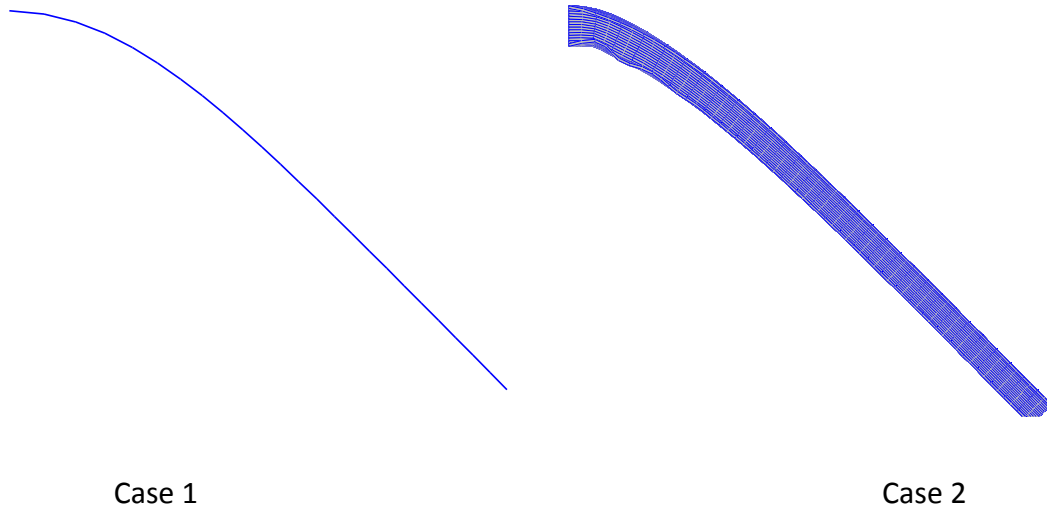


Figure 5-9: Deflected shapes for Cases 1 and 2

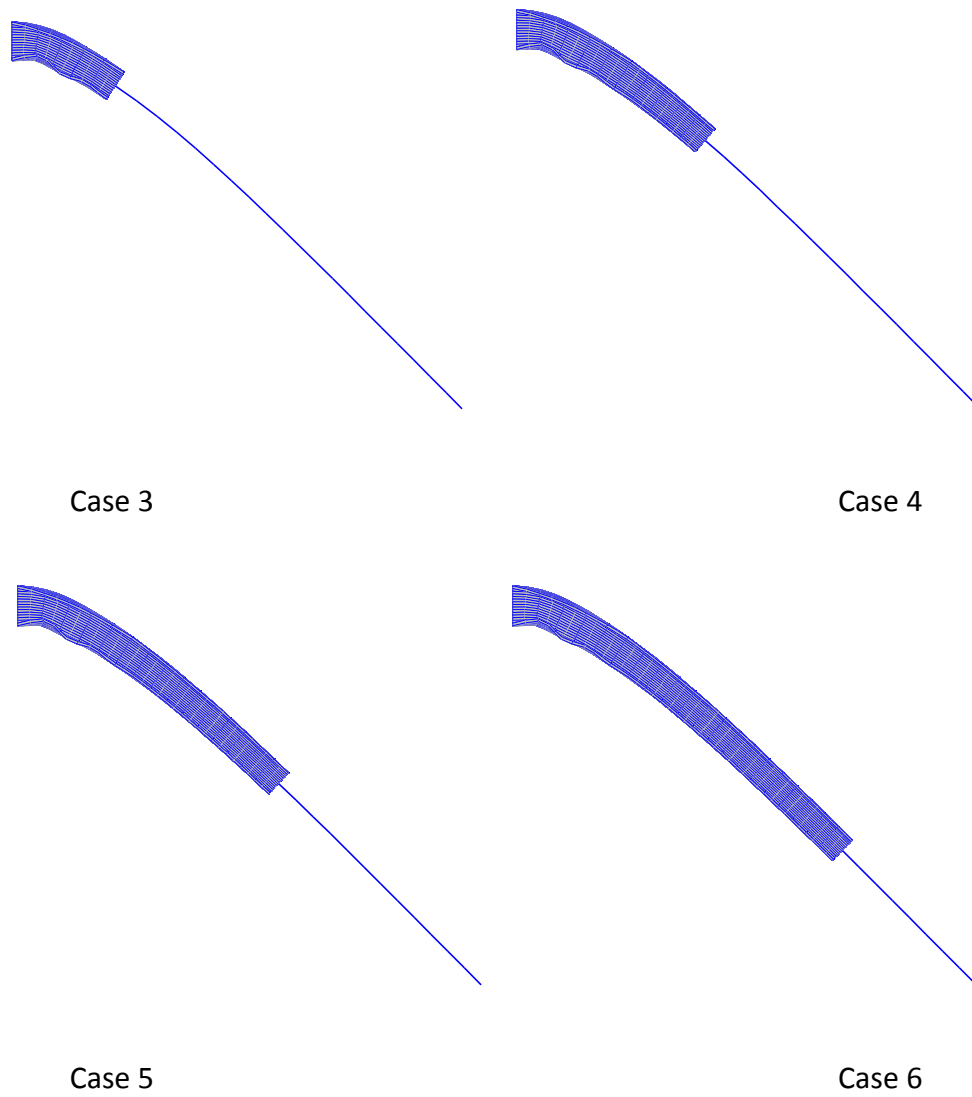


Figure 5-10: Deflected shapes of Cases 3 to 6

Case	Wall-clock time (sec)
1	2.8
2	13275.5
3	1985.9
4	5826.2
5	5997.0
6	6307.7
7	5970.3

Table 5-2: Comparison of wall-clock times for cantilever beam

With a linear elastic material response being considered, the only nonlinearity involved in this example is geometric nonlinearity. This nonlinearity is demonstrated by the load-deflection responses in Figure 5-11. Interestingly, the model for Case 1, which is made entirely from 1D elements, exhibits a progressively stiffening response due to large displacement and the transition of the beam from flexural to axial response. However, the other cases, namely Case 2 utilising a full 3D model and Cases 3 to 7 with dimensional coupling, are in agreement and do not progress under load control beyond a load factor of approximately 0.65.

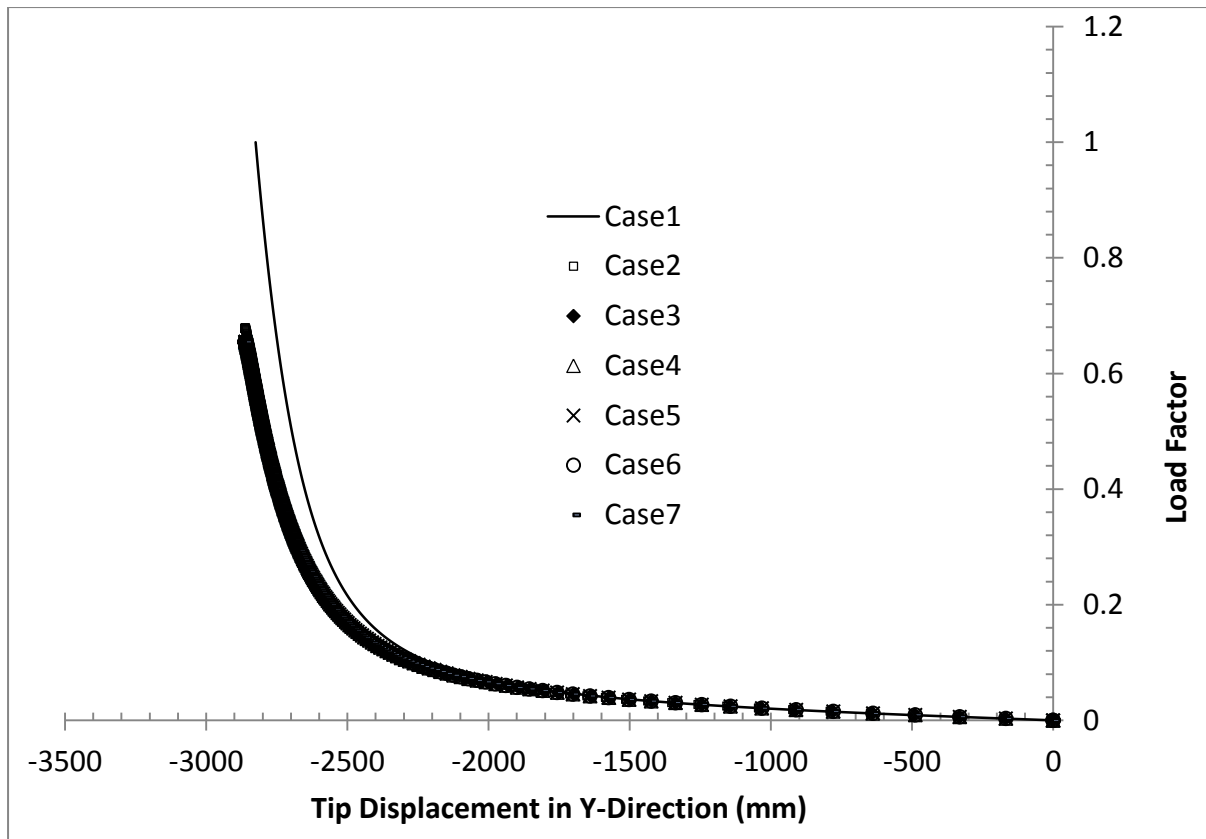


Figure 5-11: Load-deflection of cantilever for different cases

This inability of Cases 2 to 7 to progress beyond a load factor of 0.65, even with a linear elastic material response, is attributed to local buckling, which is triggered by compression buckling of the lower flange, as can be clearly seen near the fixed end in Figure 5-10. This phenomenon is successfully captured by the full 3D model as well as the models using dimensional coupling, but is missed by the 1D model of Case 1.

The comparison of the results for the different models at different stages of response presents interesting observations. In this respect, the results are compared at 3 different stages, where Stage 1 is set very early in the analysis when the load factor is about 0.03, Stage 2 is set slightly later when the load factor is about 0.06, and Stage 3 is set much later when the load factor is about 0.21. The comparison of the load-deflection response at these three stages is presented in Figure 5-12 to Figure 5-14, respectively.

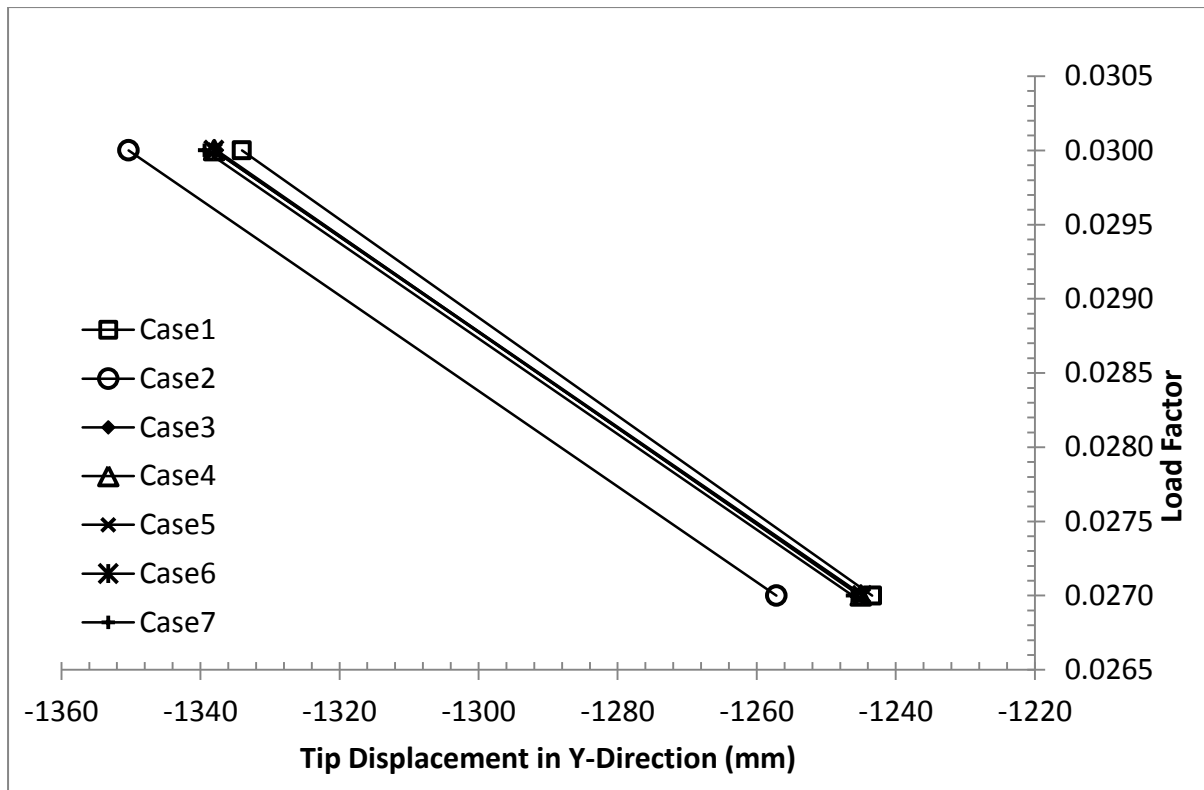


Figure 5-12: Load-deflection comparison at Stage 1

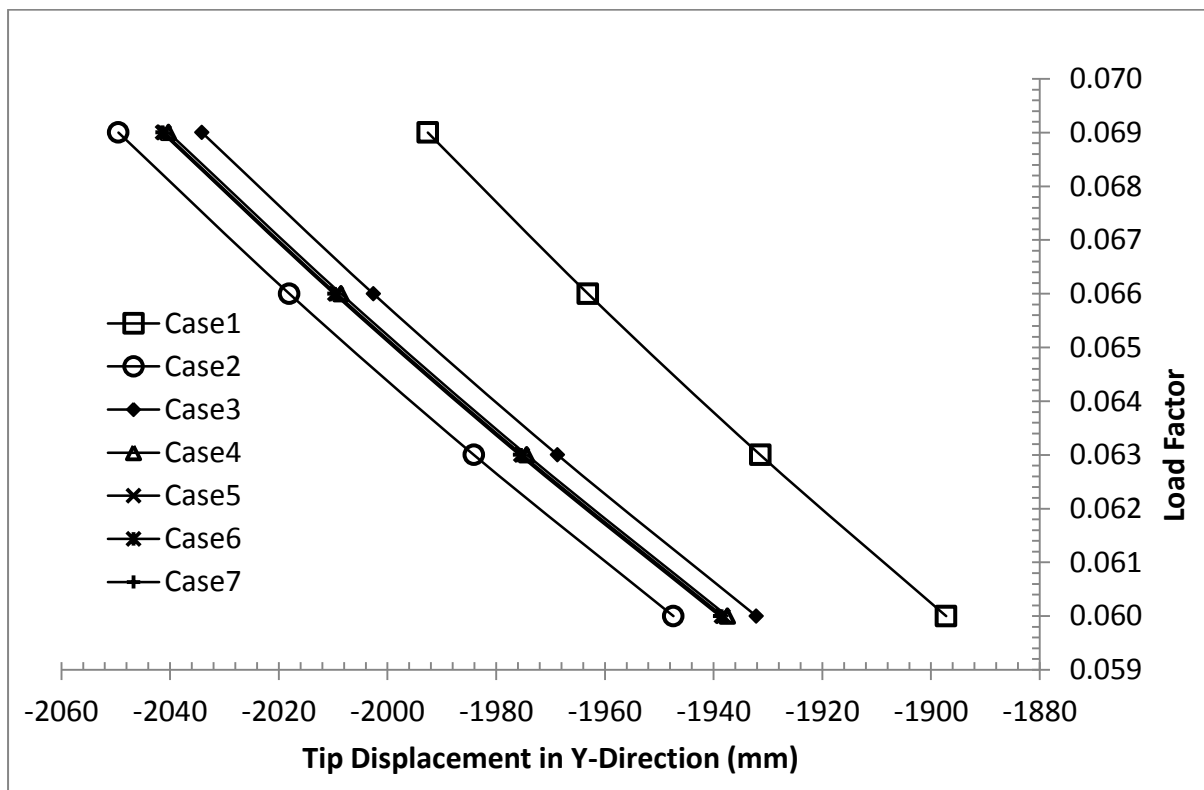


Figure 5-13: Load-deflection comparison at Stage 2

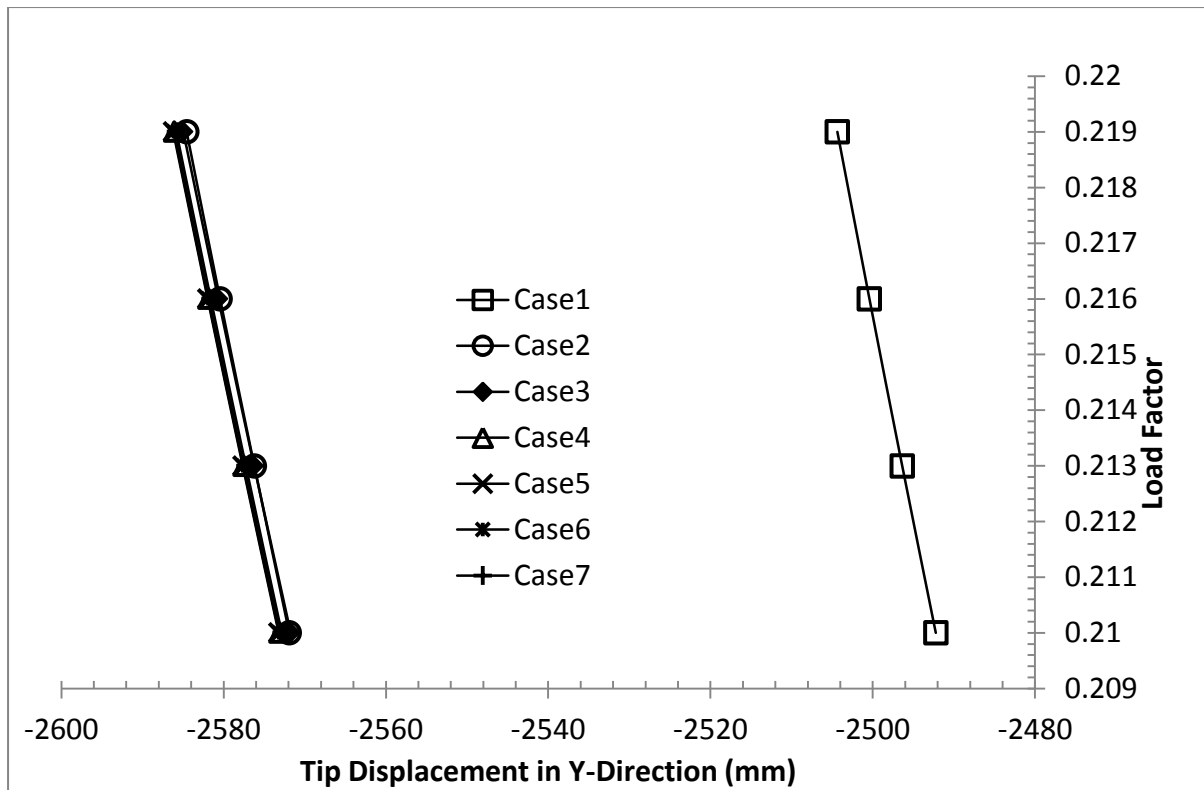
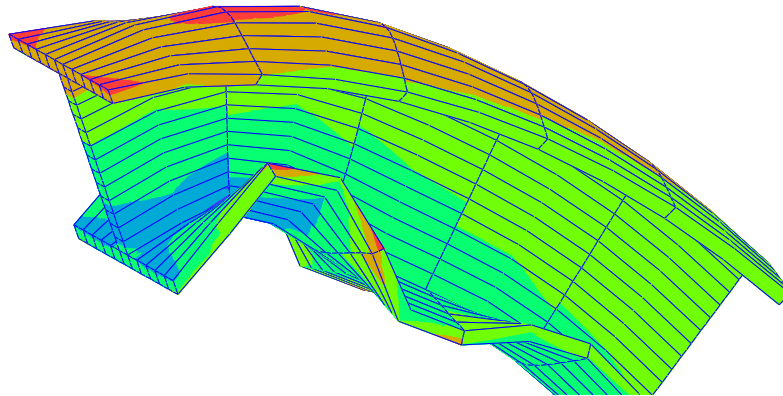


Figure 5-14: Load-deflection comparison at Stage 3

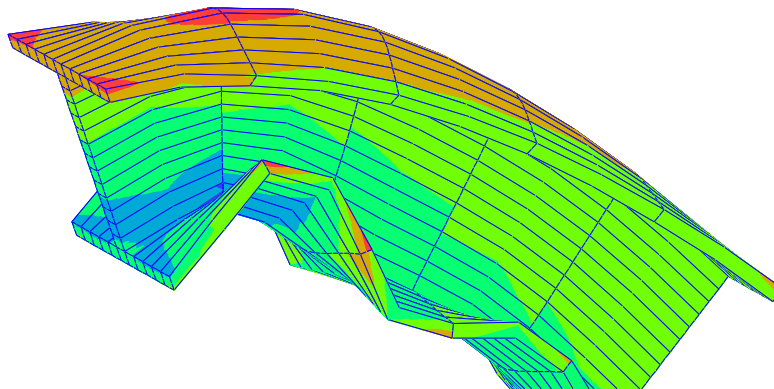
As can be seen in Figure 5-12, earlier in the analysis when the difference in the tip deflection predicted by Cases 1 and 2 is about 20 mm, the results of Cases 3 to 6 are closer to those of Case 1. The discrepancy between Case 2 and Cases 3 to 7 is attributed to the assumption of rigid cross-sections at the 1D-3D coupling interfaces. In Stage 2, while the difference between the deflections predicted by Case 1 and Case 2 increases to about 40 mm, Cases 3 to 7 move away from Case 1 and get closer towards Case 2. Further on in Stage 3 when the difference between Cases 1 and 2 increases to about 80 mm, Cases 3 to 7 agree totally with Case 2.

Interestingly, the results of Cases 3 to 7, initially closer to Case 1 and later converging to Case 2, are close throughout the response. This indicates that small discrepancy in the initial linear response compared to the full 3D model of Case 2 is mainly due to the modelling of the 3D region in the vicinity of the support, and the associated inaccuracy of 1D-3D coupling. This small discrepancy is reduced at the later stages of the response with the occurrence of local flange buckling, indicating that Case 3 with only a partial 3D model near

the support is sufficiently accurate in this respect. This fact is further highlighted by the comparison of stress contours in the X-direction in the partition closest to the fixed end in Case 2 to that in Case 3 (Figure 5-15), which shows a close match at the same load factor.



(a) Case 2



(b) Case 3

Figure 5-15: Stress contours in X-Direction

Importantly for this example, the results of the accurate model of Case 2 are obtained in approximately 3 ¾ hours, while 1D-3D dimensional coupling as in Case 3 provides results of almost the same accuracy in a much reduced wall-clock time of around ½ hour, presenting a computational saving of over 85%.

5.5.2 Frame with Masonry Infill

The second example consists of a frame with masonry infill, and is used to illustrate the general applicability of the mixed dimensional modelling approach to problems in which parts of the structure are naturally modelled with 1D elements, with other parts more realistically modelled with 3D elements. While the masonry infill can be accurately modelled with 3D continuum elements for the masonry bricks and 2D interface elements for the mortar joints [Macorini and Izzuddin, 2011], the most realistic model for frame members is combining accuracy and computational efficiency is based on 1D beam-column elements. The partitioning approach combined with mixed dimensioned coupling between the 1D and 3D elements developed in this work provides a unique capability for modelling such structural systems with relative ease.

The frame considered for this example has a square geometry of size 180 cm, measured from the centre line of the frame members on opposite sides. The frame members are 20 cm deep, which leads to an outer dimension of 200 cm for the system, as illustrated in Figure 5-16. For illustration purposes, the infill is modelled with an 8x8 mesh of 20-noded 3D elastic brick elements with a thickness of 9 cm. A more realistic representation of masonry walls is presented as a case study in Chapter 7. The material models used for both the frame and the masonry infill are linear elastic with the moduli of elasticity and Poisson's ratios as shown in Table 5-3

The parent structure for the mixed dimensioned model of this frame consists of a square with 17 nodes on each side, making a total of 64 nodes. The child partitions are connected to master-slave elements, which then form their partitioned boundary as shown in Figure 5-17.

The translational freedoms of the left corners of the frame in all 3 directions are restrained in addition to the rotation about the vertical axis, whereas the right corners are subjected to increasing point loads in addition to the weight of the bricks applied as an initial load. The final deflected shape of the frame as well as the contours of the axial stresses in the frame elements are shown in Figure 5-18.

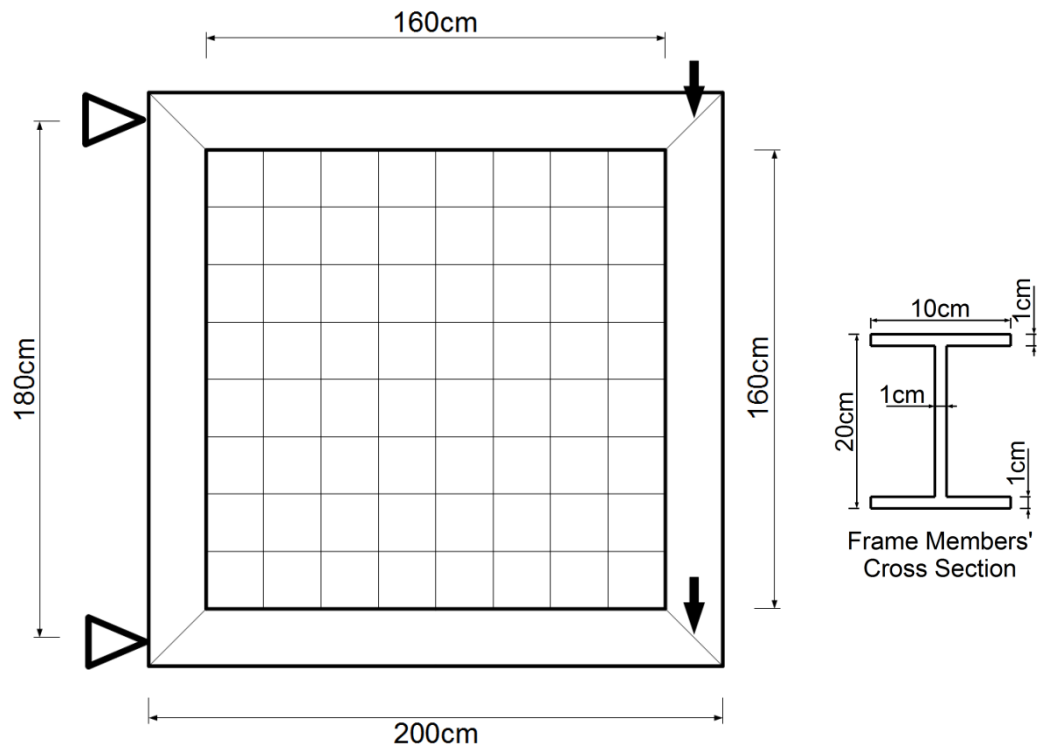


Figure 5-16: Frame with infill masonry

Materials	Modulus of Elasticity	Poisson's Ratio
Frame Members	2.0×10^5	0.3
Block Masonry	16.7×10^3	0.15

Table 5-3: Material properties for frame with masonry infill

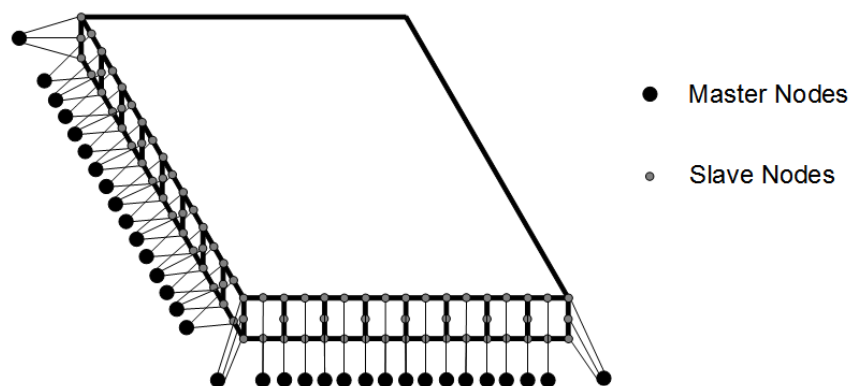


Figure 5-17: Partitioned boundary of the masonry infill

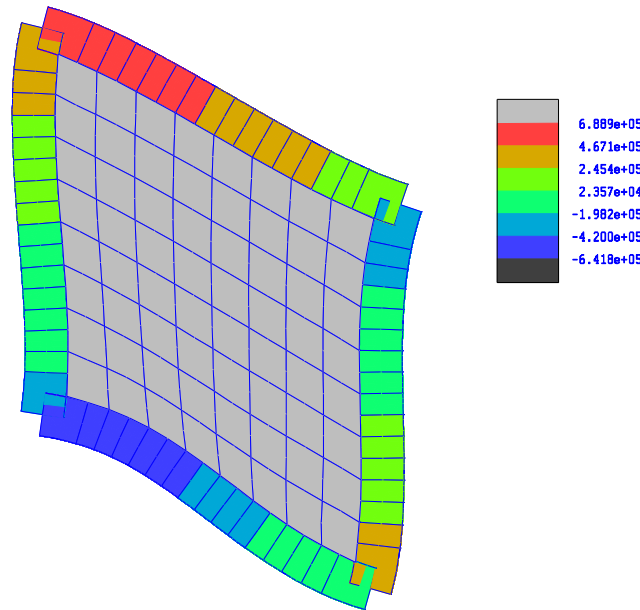


Figure 5-18: Deflected shape and axial stress contours of the frame

This example is presented only to illustrate the applicability of the developed dimensional coupling scheme to such structures. The detailed discussion on the results, being out of context here, is therefore skipped. The analysis time required for this frame with infill masonry is merely 19 seconds for 10 loading steps, as this is a simplified frame for illustrative purposes only. Obviously, more elaborate models would require more computing time, though this example shows that dimensional coupling can be successfully used for this type of problem.

5.5.3 3D Space Frame

The third example used to illustrate the application of dimensional coupling is the same space frame considered in Section 3.4.2.3. Here, however, the frame is modelled using the mixed dimensioned coupling, where the parent structure consists of one node per cross-section, where it is connected to the partitions as illustrated in Figure 5-19.

The two lower columns are connected to one node only, whilst the rest of the columns and all the beams are connected to two nodes each. The joints at the top are connected to two nodes and those at the lower level are connected to three nodes each. This setting makes

the modelling process very easy as only one data file for each kind of partition needs to be created. Once these files are created, the modelling of the parent structure becomes similar to that of a 2D plane frame with line elements, although the underlying super elements correspond to child partitions with 3D brick elements with 1D-3D coupling at the partition boundary.

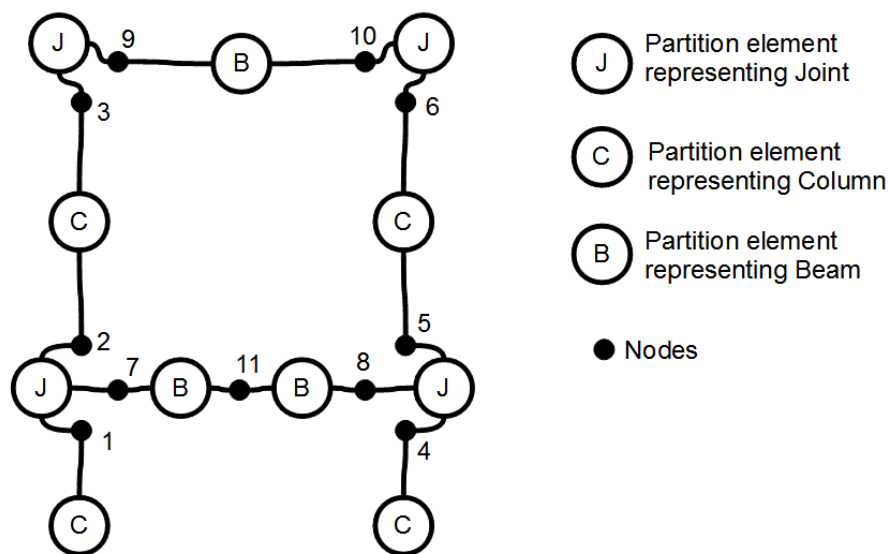


Figure 5-19: Parent structure for mixed-dimensioned model of 3D space frame

Although the number of nodes at the original partitioned boundary for the partitions remains the same with the additional master-slave elements involving dimensional coupling process, the quantity of information being communicated between processes is now greatly reduced, since only the entities associated with the 1D boundary nodes are communicated. The size of the stiffness matrix being communicated for the top of the first column between the parent structure and child partition no. 1 is reduced from 467856 values for 228 nodes with 3 degrees of freedom each to a mere 36 values for a single node with 6 degrees of freedom. The extent of communication between processes becomes therefore negligible compared to the previous case when partitioning is performed without dimensional coupling.

With this great reduction in communication overhead, the computational efficiency of the partitioned model has further increased as expected. For the same time of 72 hours, the dimensionally coupled model was able to carry out 259 steps as opposed to 121 by the partitioned model without dimensional coupling, as shown in Table 5-4.

In the previous case, the parent structure consisted of a total of 2208 nodes, while with dimensional coupling it is reduced to 11 nodes. As was argued in Section 3.5.4, the computational time required is largely related to the size of the largest partition, the size of the parent structure, and the communication overhead. Although the size of the largest partition remains the same in the case with dimensional coupling, the great reduction in the other two clearly results in an increase of computational efficiency to more than twice.

The final deflected shape for the frame at loading step 259 is shown in Figure 5-20 .

Load steps completed in 72 hours for:	Initial Loading	Time-history	Total
Monolithic analysis	1	9	10
Partitioned analysis without dimensional coupling	1	120	121
Partitioned analysis with dimensional coupling	1	258	259

Table 5-4: Computational efficiency of dimensional coupling compared to other methods

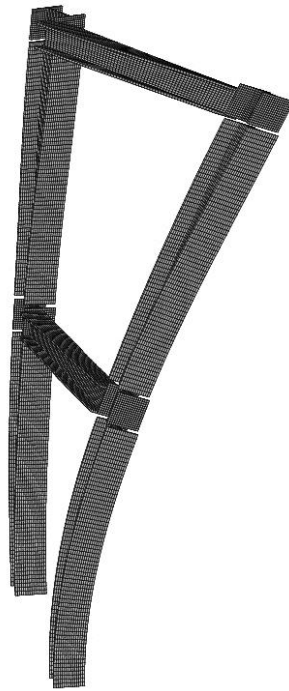


Figure 5-20: Final deflected shape of the frame

Importantly, this enhancement of computational efficiency with dimensional coupling does not come at a compromise in accuracy. This is illustrated in Figure 5-21, where the stress contours at the column base in partition no. 1 are favourably compared for the two cases with and without dimensional coupling at the same time step.

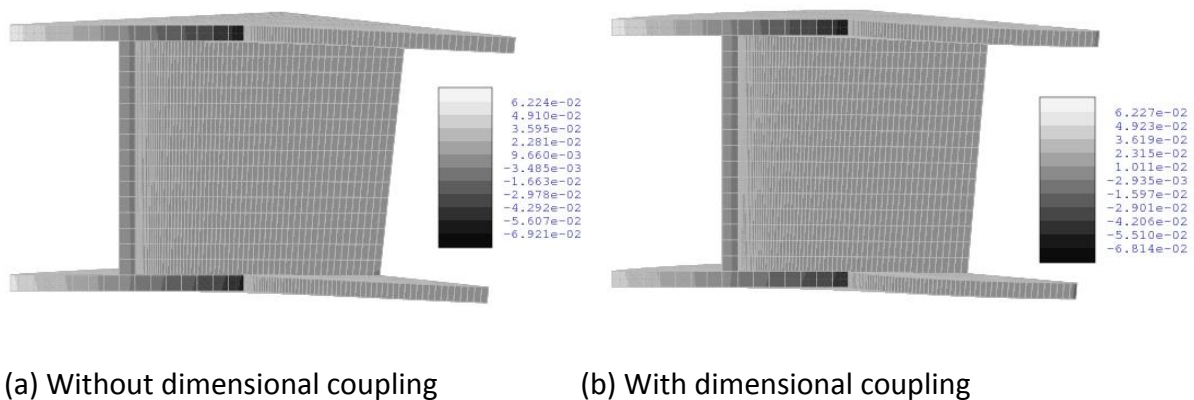


Figure 5-21: Comparison of longitudinal stress contours at the column base

Hierarchical Partitioning

This chapter extends the mixed-dimensional partitioned approach to hierarchic partitioned modelling

6.1 Introduction

The domain partitioning approach presented in Chapters 3 and 4 in conjunction with the dimensional coupling based on master-slave approach presented in Chapter 5 has greatly speeded up the analysis time required for large and complex structures. However, in some cases where the size of the structure is very large resulting in a large number of partitions, the modelling process, although still more effective in comparison to modelling the entire structure without partitioning, becomes increasingly cumbersome.

It was discussed in Section 3.3.3 that the computational efficiency of the partitioning approach is greatly affected by the size of the parent structure. For structures where the computational requirements of the element contributions are much less compared to those for the global solution, increasing the number of partitions beyond a certain number may not be beneficial as was demonstrated in the example of Section 3.5.5.

Both issues of modelling complexity and increasing size of the parent structure can be resolved with the use of hierarchic partitioning. The domain decomposition approach presented in Chapter 3 can be readily extended to include hierarchic partitioning.

To achieve hierarchical partitioning, the domain decomposition approach is further enhanced by making further subdivisions of the child partitions, parts of which can be removed and replaced by partition super elements. The removed parts are then modelled as further lower level partitions with the dual partition super elements surrounding their partitioned boundary. This partitioning scheme is illustrated in Figure 6-1, where it is noted that intermediate and the lowest level partitions can use dimensional coupling on their

partitioned boundary. With this scheme, an unlimited number of intermediate levels can be created and dimensional coupling can be used at any level of the hierarchy.

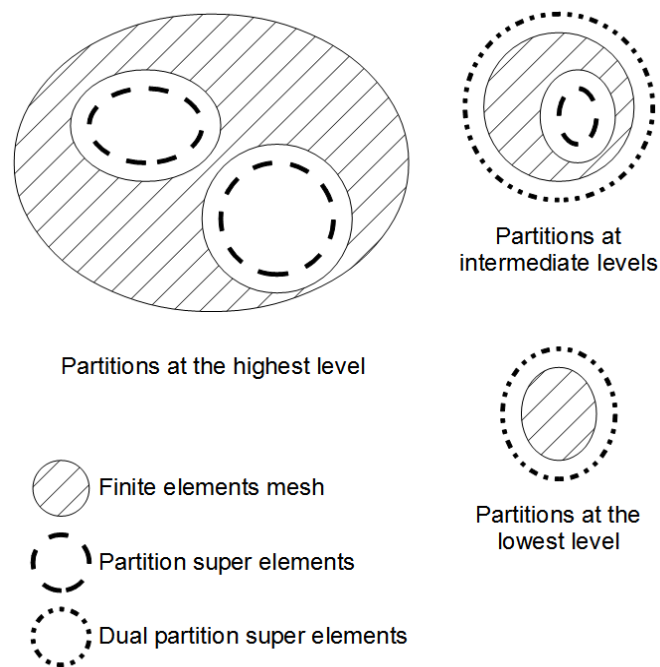


Figure 6-1: Hierarchic partitioning schematic diagram

A further advantage gained by the use of hierarchical modelling is the improved facility of using a modular modelling approach. In the case of large structures, such as a multi storey building, it is possible to make partition modules consisting of whole stories or parts thereof having further partitions within consisting, for example, of detailed member or connection models.

With hierarchical modelling being multi-level, partitions at one level are children to those at the upper level and parent to those at the lower level. There is only one partition at the highest level, referred to as the 'root partition', which makes the shape of the hierarchical partitioning model similar to a fat tree network. On the other hand, the partitions at the lowest level in the hierarchy are given a general collective name of 'terminal partitions'. An example of a hierarchic partitioned model is shown in Figure 6-2, where 'L' stands for the level number and 'P' stands for the partition number which is also the rank of the process

that runs that particular partition. Each partition has a communication link with its children connected at the bottom and one with its parent connected at the top.

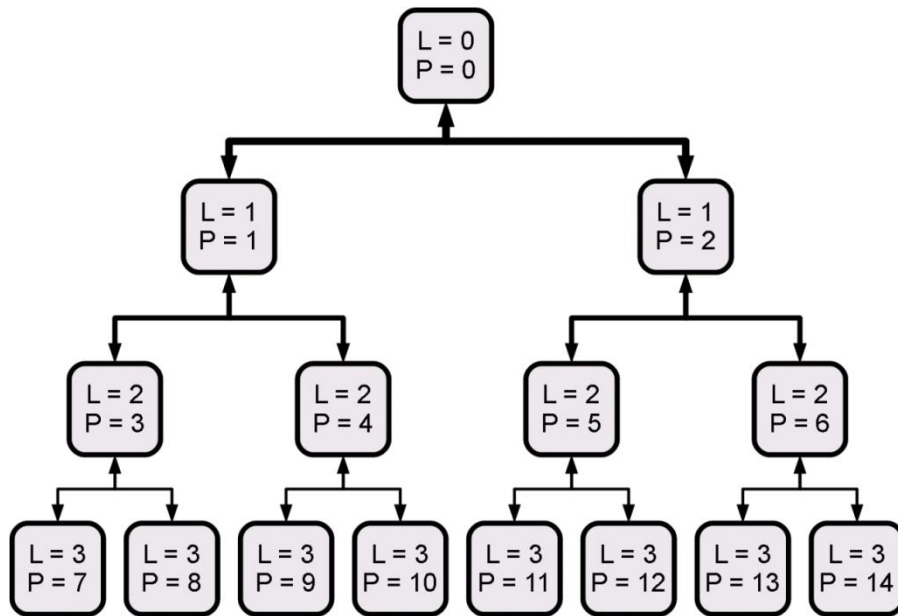


Figure 6-2: Example hierarchical partitioning model and communication network

As required during analysis, each partition sends tangent stiffness and resistance forces to and receives incremental displacements from its parent, as elaborated in the following section. A coordinator process, however, is required, which maintains the overall flow of the analysis and ensures that all partitions are running the same load/time steps, etc. For this purpose, the process running the root partition is designated as the coordinator.

6.2 Hierarchical Partitioning Approach

The domain decomposition approach presented in Section 3.2 has been modified for the purpose of hierarchical partitioning. The modified approach is explained here with the use of a similar example to that used in Chapter 3 (Figure 3—1), which consists of a 2D grid of 32 elements as shown in Figure 6-3, with the assumption that there are two degrees of freedom per node.

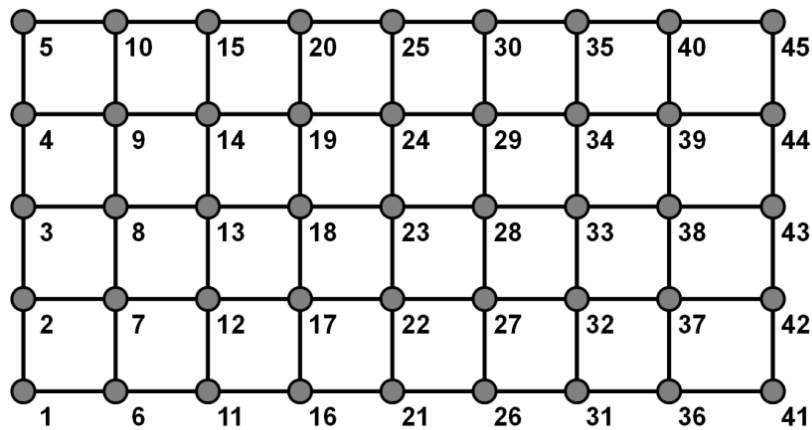
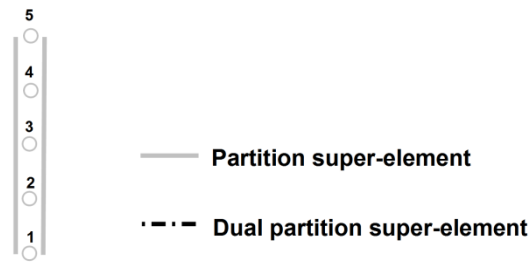
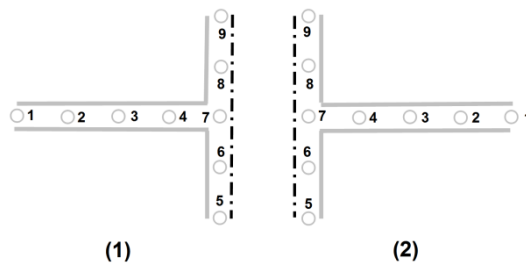


Figure 6-3: A general structural domain with 32 elements

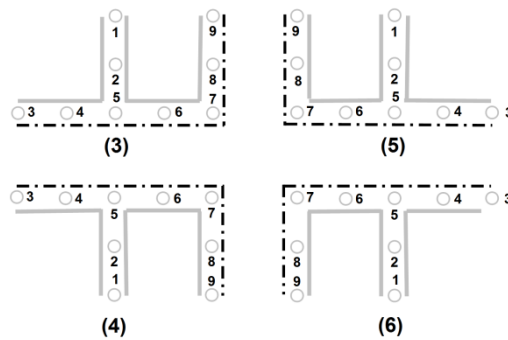
The solution method for the monolithic system was explained in Section 3.2, though in this case the sizes of the stiffness matrix and load/resistance/displacement vectors in Equations 3—1 to 3—5 are 90 instead of 80. For the purpose of illustrating hierarchical partitioning, the structural domain is first divided into two partitions. Each of these partitions is further divided to make third level partitions, which are again divided to make a fourth level of partitions. The partitions at the last level consist of 4 elements each, as shown in Figure 6-4. It is worth noting here that the present hierarchical approach does not put any limits on the number of partitions as well as partitioning levels. The number of partitioning levels in this illustration is limited, without loss of generality, to 4 for the sake of simplicity.



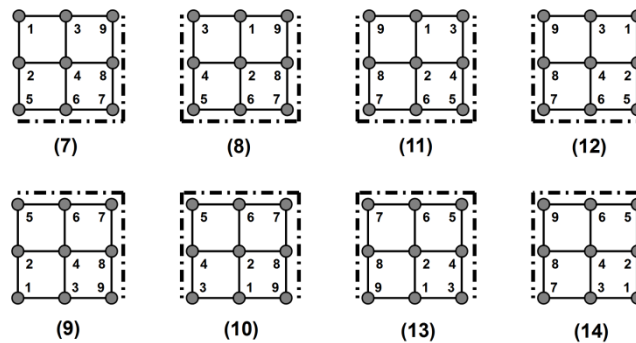
(a) Parent structure (Level 0)



(b) Level 1 partitions



(c) Level 2 partitions



(d) Level 3 partitions

Figure 6-4: Four levels of hierarchical partitioning

6.2.1 Method Formulation

The partitioned domains in the illustrative example are named as Ω_0 to Ω_{14} in accordance with their partition numbers as shown in brackets below the respective partitions in Figure 6-4. Ω_0 is the root partition and Ω_7 to Ω_{15} are the terminal partitions, thus partitions Ω_1 to Ω_6 are at intermediate levels.

Since Ω_0 has two partition super elements both connected to all of its 5 nodes, its stiffness matrix is simply the sum of the condensed stiffness matrices returned by the dual super elements of the child partitions:

$$K_{\Omega_0[10,10]} = K_{\Omega_1[10,10]}^c + K_{\Omega_2[10,10]}^c \quad 6-1$$

Partitions Ω_1 and Ω_2 do not have any conventional elements either; therefore, their stiffness matrices also consist only of the condensed stiffness matrices from their child partitions. The assembly of the stiffness matrix in these partitions is done according to the element connectivity of the partition super elements. Equations 6-2 to 6-5 show the assembly of the stiffness matrix for Ω_1 , and similar equations can be written for Ω_2 by replacing the relevant partition numbers:

$$K_{\Omega_1[1:10,1:10]} = K_{\Omega_3[1:10,1:10]}^c + K_{\Omega_4[1:10,1:10]}^c \quad 6-2$$

$$K_{\Omega_2[1:14,1:14;1:14,1:10]} = K_{\Omega_4[1:14,1:14;1:14,1:10]}^c \quad 6-3$$

$$K_{\Omega_3[1:10,15:18;15:18,1:10;15:18,15:18]} = K_{\Omega_3[1:10,1:14;1:14,1:10;1:14,1:14]}^c \quad 6-4$$

$$K_{\Omega_4[1:14,15:18;15:18,1:14]} = 0 \quad 6-5$$

Similarly, the third level partitions Ω_3 to Ω_6 do not have conventional elements, and their tangent stiffness matrices also consist of condensed stiffness entries received from their children. These partitions are similar to each other, and the assembly Equations 6-6 to 6-9 presented for Ω_3 below can also be re-written for Ω_4 , Ω_5 , and Ω_6 , replacing the relevant parent and child partition numbers:

$$K_{\Omega_3[1:4,1:4;9:10,1:4;1:4,9:10,9:10,9:10]} = K_{\Omega_7[1:4,1:4;9:10,1:4;1:4,9:10;9:10,9:10]}^c + K_{\Omega_8[1:4,1:4;5:6,1:4;1:4,5:6;5:6,5:6]}^c \quad 6-6$$

$$K_{\Omega_3[5:8,1:10;1:4,5:8;9:10,5:8]} = K_{\Omega_7[5:8,1:10;1:4,5:8;9:10,5:8]}^c \quad 6-7$$

$$K_{\Omega_3[1:18,1:4;1:4,1:18;1:18,9:18;9:10,1:18]} = K_{\Omega_8[7:14,1:4;1:4,7:14;7:14,5:14,5:6,7:14]}^c \quad 6-8$$

$$K_{\Omega_3[1:18,5:8;5:8,1:18]} = 0 \quad 6-9$$

It is clear from the above description of the assembly of the tangent stiffness matrix that it is done from bottom up. In general, each partition at the lower level performs the forward elimination until it reaches its partitioned boundary as explained in Section 3.2.2. It then sends the condensed tangent stiffness matrix for its partitioned boundary to its parent process at the level above, which assembles its own tangent stiffness matrix, performs the forward elimination until the partitioned boundary, and then sends the partitioned boundary values further up.

Once the tangent stiffness matrix and the out of balance have been received at the root level, Ω_0 calculates the iterative corrections to the displacements for the next iteration.

$$R_{\Omega_0\{1:10\}} = R_{\Omega_1\{1:10\}}^c + R_{\Omega_2\{1:10\}}^c \quad 6-10$$

$$G_{\Omega_0\{10 \times 1\}} = R_{\Omega_0\{10 \times 1\}} - P_{\Omega_0\{10 \times 1\}} \quad 6-11$$

$$\delta d_{\Omega_0\{10 \times 1\}} = K_{\Omega_0[10 \times 10]}^{-1} \left(-G_{\Omega_0\{10 \times 1\}} \right) \quad 6-12$$

The solution of Equation 6—12 involves a complete process of forward elimination followed by backward substitution. The root partition then sends the values of the iterative displacement corrections to its child partitions for their partitioned boundary nodes through partition super elements:

$$\delta d_{\Omega_1\{9:18\}}^c = \delta d_{\Omega_2\{9:18\}}^c = \delta d_{\Omega_0\{1:10\}} \quad 6-13$$

The second level partitions Ω_1 and Ω_2 perform their backward substitution step according to Equation 3—18 for the rest of their nodes to calculate the corresponding iterative displacement corrections. These partitions then send the values for the partitioned

boundaries of their children through partition super elements, which in turn perform their backward substitution step. The process thus continues as the iterative corrections to displacements are propagated down the hierarchic levels. This transaction for Ω_8 , for example, is given as:

$$\delta d_{\Omega_8\{5:8;9:18\}}^c = \delta d_{\Omega_3\{1:4;9:18\}} \quad 6-14$$

When the terminal partitions have received the iterative corrections and performed their backward substitution, the iteration is completed. If there are further iterations required, the entire process is repeated by the terminal partitions performing their forward elimination, sending the condensed tangent stiffness and resistance forces at the partitioned boundary to their parents and so on. This iterative process continues until the out of balance G and/or the iterative displacement corrections δd are within a specific tolerance for all partitions as expressed by Equation 3—20.

It should be evident at this point that the proposed hierarchic partitioning method described above is both general and scalable, which is an important feature for parallelisation and overcoming memory bottlenecks.

6.3 Parallelisation Efficiency with Hierarchic Partitioning

In addition to ease of modelling, hierarchical partitioning is also computationally efficient in cases where the size of the parent structure becomes large. This fact is demonstrated with the help of the same example which was used in Section 6.2, Figure 6-4. As was done in Chapter 3, focus is placed on the computational cost of the forward elimination process using the frontal method, where the hierarchic partitioning approach is compared to the single level partitioning approach. While the total CPU time required for the forward elimination phase may increase because of the introduction of additional levels of partitioning, the wall clock time can reduce significantly.

The number of sequential multiplication operations reflecting the wall clock time for this model is compared to that when the same structure is modelled with only one level of partitioning, with partitions similar to those at level three in Figure 6-4(c), as illustrated in Figure 6-5.

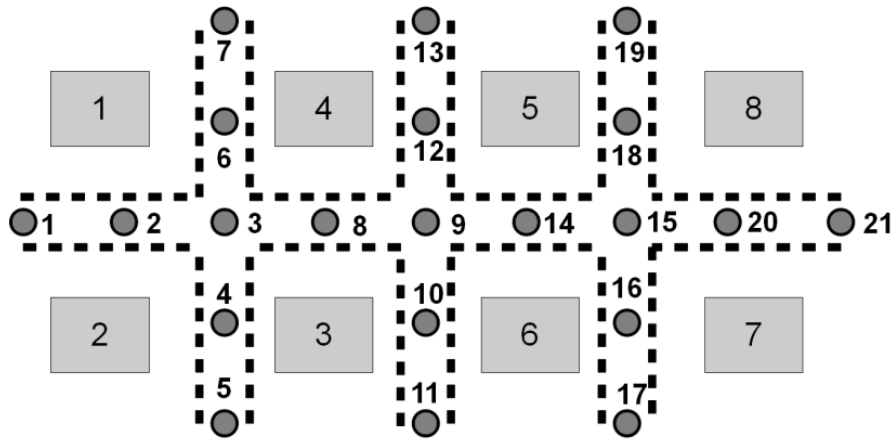


Figure 6-5: Parent structure for one level partitioning

Assuming an element frontal ordering as shown in Figure 6-5, the details of the reduction operations required at the partition levels and at the global level are given in Table 6-1. The number of multiplication operations for the parent structure in the case of single level partitioning is 6762 and that for the most expensive single partition is 1028. It should be noted that this number is for any partition from No. 3 to 6 and that the number of multiplication operations for other partitions is lower. Therefore, the total number of sequential multiplication operations reflecting wall clock time in this case is 7790.

In case of the hierarchical partitioning, since the last level partitions are the same, their required number of multiplication operations also remains the same. Going in reverse order from level 2 to level 0, the number of multiplication operations required at each level is 1028, 1608 and 330, respectively, as shown in Table 6-2. This gives a total of sequential number of multiplication operations reflecting wall clock time as 3994, which is 51.27% of the single level partitioning.

Reduction Operation	Size of Grandpa Before Reduction	Size of Grandpa After Reduction	Nodes Eliminated	Multiplication Operations
Operations at Partition Level (for Partition No. 1, 2, 7, and 8)				
1	8x8	6x6	1	98
2	10x10	8x8	1	162
3	12x12	10x10	1	242
4	12x12	10x10	1	242
Total			4	744
Operations at Partition Level (for Partition No. 3 to 6)				
1	18x18	14x14	2	1028
Operations at Global Level				
1	14x14	10x10	2	580
2	18x18	14x14	2	1028
3	18x18	10x10	4	1608
4	18x18	14x14	2	1028
5	18x18	10x10	4	1608
6	14x14	10x10	2	580
7	10x10	0	5	330
Total at Global Level			21	6762
Total sequential multiplication operations reflecting wall clock time				7790

Table 6-1: Reduction operations for the single level partitioning

Partitioning Level	Size of Grandpa Before Reduction	Size of Grandpa After Reduction	Nodes Eliminated	Multiplication Operations
3	18x18	14x14	2	1028
2	18x18	14x14	2	1028
1	18x18	10x10	4	1608
0	10x10	0	5	330
Total sequential multiplication operations reflecting wall clock time				3994

Table 6-2: Reduction operations for hierarchical partitioning

In addition to the computational efficiency in terms of multiplication operations, the communication overhead is also expected to reduce in the case of hierarchical partitioning. Although a larger number of processors is being used in this example (15 as opposed to 9 in case of the single level), some of the communications are being done in parallel with different parents, thus the total time required for sending data is expected to reduce. To

investigate this, the movement of the tangent stiffness matrix from the terminal partitions to the root partition is compared as follows.

In the case of single level partitioning, 4 of the child partitions send 100 entries of the tangent stiffness matrix each and the remaining 4 send 196 entries each to the parent structure. Therefore, the parent structure receives a total of 1184 entries. Although the partitions calculate the condensed stiffness matrix in parallel and send their contribution as soon as it is calculated, the parent structure receives these values in the order of the partition super elements in the frontal ordering list. In the case of hierarchical partitioning, however, this communication is parallelised to different parents. Each partition at level 2 receives a total of 296 entries from its level 3 child partitions and sends 196 entries to its level 1 parent partition. Accordingly, the total entries received by each level 1 partition is 392, and each 100 entries to the root partition which thus receives a total of 200 entries.

Considering the fact that each of the partitions at a level can receive these values from its children at the same time as the other partitions at that level, the equivalent number of tangent stiffness matrix entries moving sequentially from the lowest level partitions to the highest level partition is 888, which is 75% that of the single level partitioning model.

It is worth noting here that there may be other factors involved as well; for example, communication between any two processes might be slower than that between another pair due to their physical location. However, it is likely that these factors will affect both single level and hierarchical modelling; therefore, the comparison presented above can be considered to offer a reasonable indication of the communication efficiency gained by hierarchical partitioning.

6.4 Implementation

The proposed hierarchical modelling approach represents an extension of the single level partitioning method presented in Chapter 3, and has also been implemented with ADAPTIC [Izzuddin, 1991].

The root partition and the terminal partitions behave as the parent structure and child partitions, respectively, as described in Section 3.4. The only difference in the procedure for

the root partition process from that of the coordinator process described in Section 3.4.1 is that it sends task instructions and load factor/time to all the partitions in the model. On the other hand, the condensed tangent stiffness matrix, resistance forces and incremental displacements vectors are communicated only between it and its children according to the element connectivity of partition super elements.

Similarly, the only difference in the procedure for the terminal partition processes from that of the partition process described in Section 3.4.2 is that these receives task instructions and load factor/time from the root partition instead of the parent. The partitions at the intermediate levels, however, are slightly different as they are now both child and parent partitions at the same time. The procedure followed by the processes running the intermediate level partitions is presented next.

6.4.1 Intermediate Processes

Before the procedure followed by the intermediate partitions in terms of the tasks is explained it is important to discuss the frontal ordering of these partitions. The nodal ordering is done in such a way that their partitioned boundary nodes that they share with the partitions at higher levels are in the very end of nodal ordering list. The partition super elements representing the child partitions and conventional elements are ordered as per the ordering scheme that aims at minimising the frontal width.

Upon start, the intermediate level processes read the partition data assigned to them and then awaits the instructions from root partition. Similar to the single level partitioning method, there are a total of 7 instructions issued by the root partition:

- Task 0, New Step: This task specifies that a new load step is to be started, where the partition process 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 tangent stiffness matrix and the resistance forces vector are assembled by taking into account the

contributions from the child partitions. The condensed tangent stiffness and resistance forces for the partition boundary nodes are obtained using forward elimination and sent back to the coordinator.

- Task 2, Iterate: This task obtains the iterative incremental displacements for the partition boundary, employs backward substitution to obtain the internal partition displacements. Unlike in terminal partitions, this task then sends the iterative incremental displacements for the child partition super elements followed by the assembly of tangent stiffness and resistance forces by receiving the contributions from the children. Finally, it performs forward elimination to determine the corresponding tangent stiffness and resistance forces or its own partitioned boundary.
- Tasks 3 to 6 remain the same as described in Section 3.4.2.

communication channels are referred to as ‘communicators’ in MPI terminology. The creation of more communicators in hierarchic partitioning is necessitated by the requirement of a partition communicating with its parents and/or children separately. At the same time, the coordinator process needs to communicate with all the partitions at all levels for the purpose of sending load factor and task instructions.

In parallel processing with MPI, there are two types of communications. The first type is direct communication between two processes, sometimes also referred to as point-to-point communication, and the second type is collective communication which is done between all the processes in a group. The hierarchic partitioning approach uses both of these types of communication. The coordinator process sends task instructions through collective communication, whereas the exchange of iterative corrections to displacements, resistance forces, and condensed tangent stiffness is performed between individual partition processes.

When MPI starts, a communicator is available by default which includes all the processes, which can be used for global level collective communication. For the communication between individual processes, the use of the default communicator becomes rather complicated. To communicate with the parent using the default communicator, for example, a partition requires its parent’s rank whereas a parent has to maintain a list consisting of the ranks of its children. There are also potential inefficiencies with all processes using the same communicator for inter-process communication.

This complication is avoided by creating more communicators. For each partition that is parent to at least one other partition, a communicator is created which consists of that partition and all its children. In this communicator, the partition in question is ranked 0, whereas all the child partitions are ranked successively maintaining the order in which they are ranked in the default communicator at the start.

For creating new communicators, it is necessary that any parent partition has a rank lower than all of its children, which do not necessarily have to have consecutive ranks. If the values shown in the middle of the squares in Figure 6-7 are the ranks of these partitions in

the default communicator for example, then ‘Scheme A’ is erroneous whereas ‘Scheme B’ is fine.

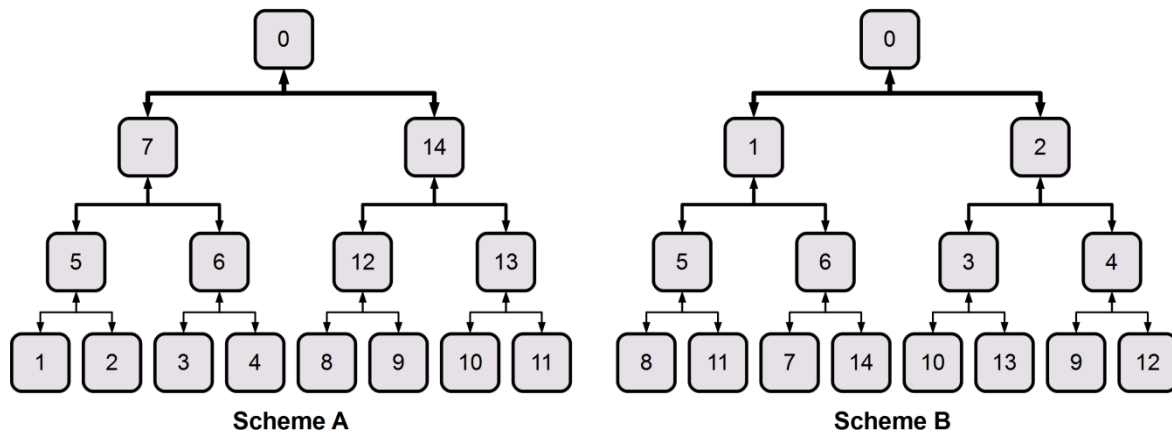


Figure 6-7: Ranking schemes for hierarchic modelling

In ‘Scheme A’ in Figure 6-7, the communicator intended for partition no. 6 will not have that partition ranked zero, instead, maintaining the original order, it will have the child partition 3 ranked zero in the new communicator. Thus partition 3 will end up being treated as the parent structure in that particular communicator, which is incorrect. In ‘Scheme B’ on the other hand, partition 6 will be ranked zero in the new communicator and partitions 7 and 14 will be locally ranked as 1 and 2 respectively.

In the current implementation, the ranking order is maintained by the use of successive input file numbers for the partitions, where the number provides the process rank in the default communicator. Therefore, it is necessary that vertical hierarchy is maintained in the numbering of the partition input files, that is partition files at a particular level must have greater numbers than the respective parent files at the upper levels.

New communicators are created by ‘splitting’ the existing default communicator. The term ‘splitting’ is used in MPI to mean the creation of a new group among some or all of the member processes of an existing communicator. The original communicator is retained during this process. The new group of processes is then assigned to a new communicator

name. All processes that are required to be part of the new group need to send an identifier termed as the 'colour' in MPI terminology. If more than one value of identifier is sent then as many new groups and associated communicators are created [MPI Forum, 2009].

At every level, one parent structure and all its children send the same value as identifier, resulting in the creation of a new group for that parent structure and its children. All the other parent structures at the same level and their children also send their respective identifiers and get their own new groups. The partitions on higher or lower levels do not send any identifier.

At level 2 in Figure 6-8 for example, when the new communicators are created, processes ranked 1,5, and 6 send identifier 'i1' and processes ranked 2,3, and 4 send identifier 'i2' whereas the rest of the processes send 'undefined identifier' denoted by 'U' in the figure. The MPI environment thus creates two new groups of processes and associates them with new communicators, one each for i1 and i2.

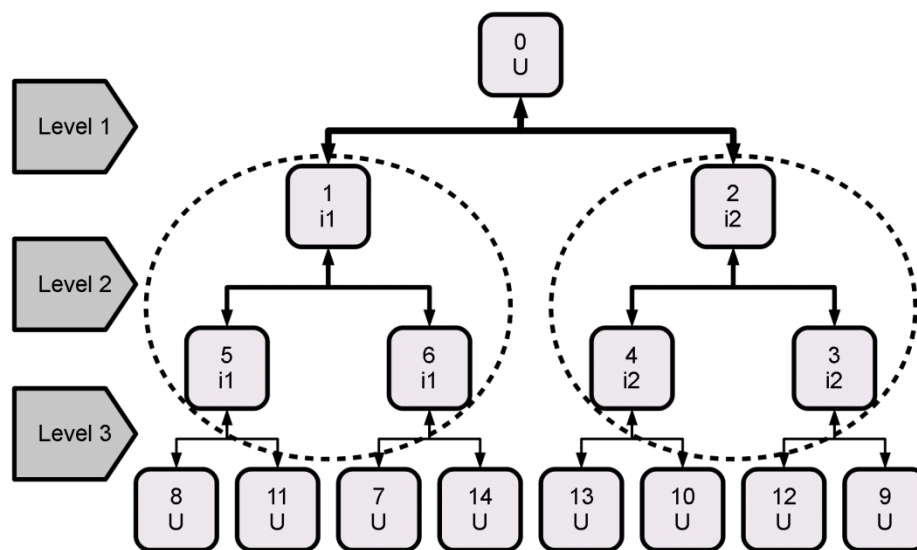


Figure 6-8: Communicator generation at one level

The identifier sent by a parent structure and its children is in fact the rank of the parent structure in the default communicator, which means that the value of i1 is 1 and that of i2 is 2 in this case. For this purpose, every child structure needs to know the rank of its parent.

The parent however, does not need to know the ranks of its children. When the new communicator is created, the parent partition can check its total number of processes, if required. Since the parent structure is ranked lower in the default communicator, it is ranked zero in the new communicator and all the children are ranked in the same order as they are in the default communicator.

In the example of Figure 6-8, in the next call for level 3, partitions 1 and 2 along with partition 0 will send identifier value as 'U' whereas partitions 5, 8, and 11 will send 5, partitions 6, 7, and 14 will send 6, partitions 4, 10, and 13 will send 4, and partitions 3, 9, and 12 will send 3 as identifier, resulting in the creation of 4 new groups and associated communicators.

In the hierarchical communicators, every partition is part of 2 communicators in addition to the default communicator except the root and the terminal partitions, which are part of only one additional communicator. The root partition is the parent in its additional communicator and the terminal partitions are the children in their additional communicators. All the partitions at the intermediate levels are parents in the communicator that they share with the partitions at the lower level and children in the communicator that they share with the partitions at the higher level.

With this arrangement, the additional communicators are named either the 'parent communicator' or the 'child communicator' depending upon the designation of a partition in that communicator. It means that the same communicator is named differently inside different partitions. The first `i1` communicator returned at level 2 in the example of Figure 6-8 is named 'child communicator' in partition 1 and 'parent communicator' in partitions 5 and 6.

The above discussion results in the development of an algorithm for the generation of new communicators for the hierarchical modelling. In this respect, it is necessary that every process takes part in the call to create new communicators; therefore, there is no distinction between processes with respect to their rank for this algorithm. Every process takes part in the algorithm and is party to every call that is made to MPI for the creation of new communicators with the only difference being in the identifier that they send to this

call. A flow chart for the communicator generating algorithm is shown in Figure 6-9, where, the communicator names 'CCOMM', 'PCOMM', and 'DCOM' stand for 'child communicator', 'parent communicator', and 'dummy communicator', respectively. The dummy communicator name is used just as a place holder as the identifier value of 'undefined' means that the MPI will return null communicator for that process. The term 'processed' is used to determine the current level under consideration where, all processes above the current level are 'processed' while those below are not.

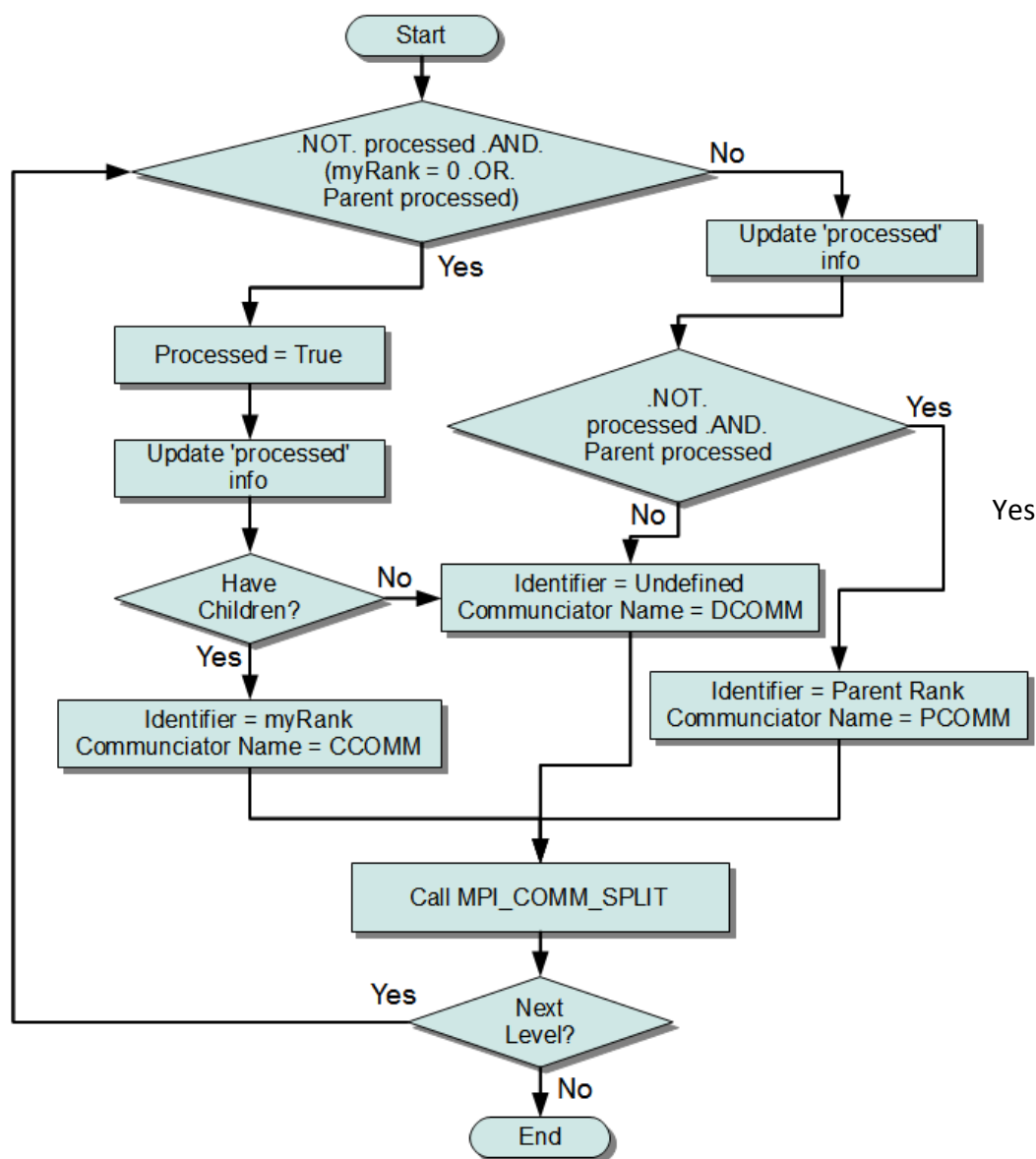


Figure 6-9: Flow chart of the communicator generating algorithm

6.4.3 Input Data Syntax

The input data files for the hierarchical partitioning remain virtually the same as described for the general domain decomposition and mixed dimensional modelling using master slave elements in Chapters 3 and 5, respectively. The only addition required to the input data for hierarchical partitioning is that the partitions at lower levels must specify their parent rank. Therefore, a single entry 'parent' is added in the 'partitioned boundary module as shown below:

```
partitioned.boundary
  parent.rank = <int>
  nod.name/master.slave.elm.name
  <token>
  < token>
  ⋮
```

The integer value of '*parent.rank*' is obviously the rank of the parent process. The rest of the data syntax is similar to that presented in Section 3.4.3. Example input files for hierarchic models are presented in Appendix C.

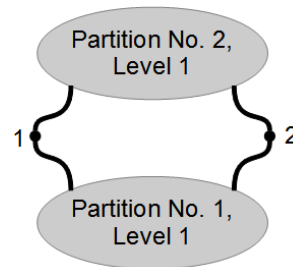
6.5 Verification

In this section, the two examples presented in Sections 3.5.2 and 3.5.5 are reproduced with hierarchical partitioned modelling, investigating what effects this has on the computational efficiency as well as accuracy.

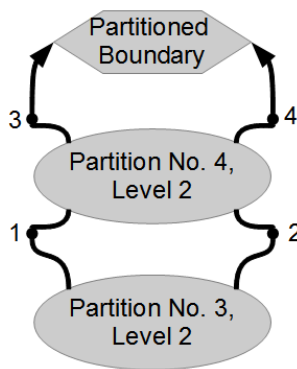
6.5.1 Dynamic Analysis of Four-Storey Frame

The first example considered is the four-storey frame presented before in Section 3.5.2, which is now modelled in a hierarchical setting. The analysis of this frame is not computationally expensive, therefore, no significant gain in computational efficiency is anticipated; however, it is being presented as an illustrative example to demonstrate the applicability and the accuracy of results of the hierarchical partitioning.

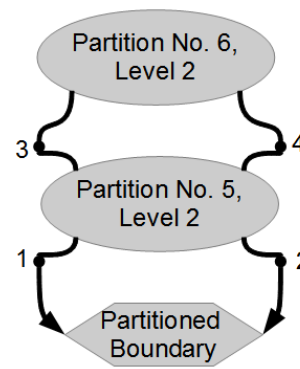
In the hierarchical model, the level 0 partition consists of just 2 nodes as shown in Figure 6-10, with two partitions at level 1 connected to both these nodes. Each level 1 partition is further subdivided into two level 2 partitions resulting in a total of seven partitions with three levels of partitioning.



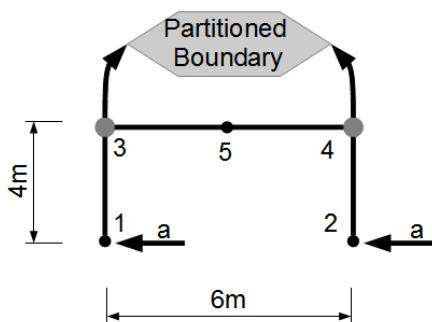
(a) Root Partition



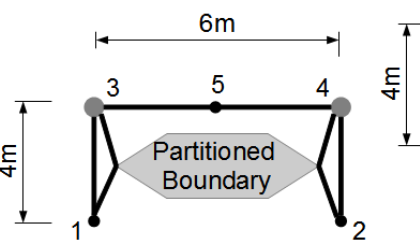
(b) Partition 1



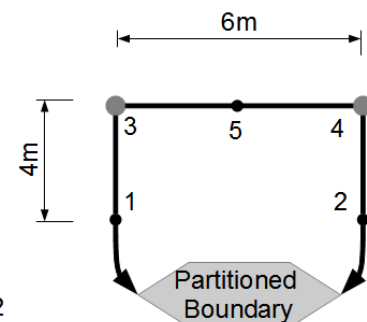
(c) Partition 2



(d) Partition 3



(e) Partition 4 and 5



(f) Partition 6

● Ordinary Nodes ● Nodes with lumped mass

Figure 6-10: Hierarchic partitioning of four-storey frame

The material properties as well as the element types used in the level 2 partitions, partitions 3 to 6, are the same as used in Chapter 3 for the respective parts of the frame. The base acceleration given by the record of Figure 3—15 is applied at the base nodes in partition 3. The comparison of the results obtained with hierarchical partitioning against those obtained with monolithic and single level partitioning is shown in Figure 6-11, where an exact match is observed.

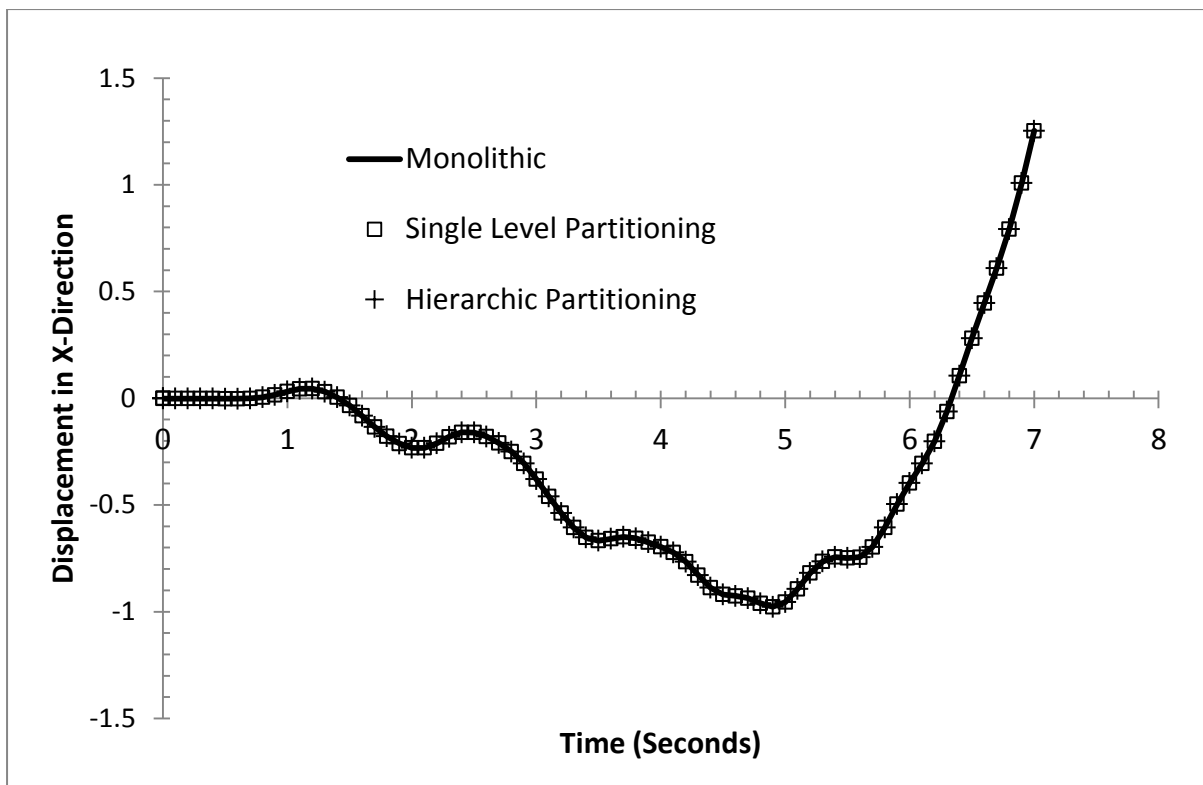


Figure 6-11: Comparison of the top displacement for the four-storey frame

6.5.2 Square Slab

The square slab used in the example of Section 3.5.5 is considered with hierarchical partitioned modelling using a total of 3 levels of partitioning. The structure is first partitioned into four partitions at the second level, each of which is further partitioned into four partitions to create the third level. The partitions at the first two levels consist of

partition super elements only, whereas the terminal partitions consist of 3D brick elements with their partitioned boundaries connected to the dual partition super elements, as shown in Figure 6-12.

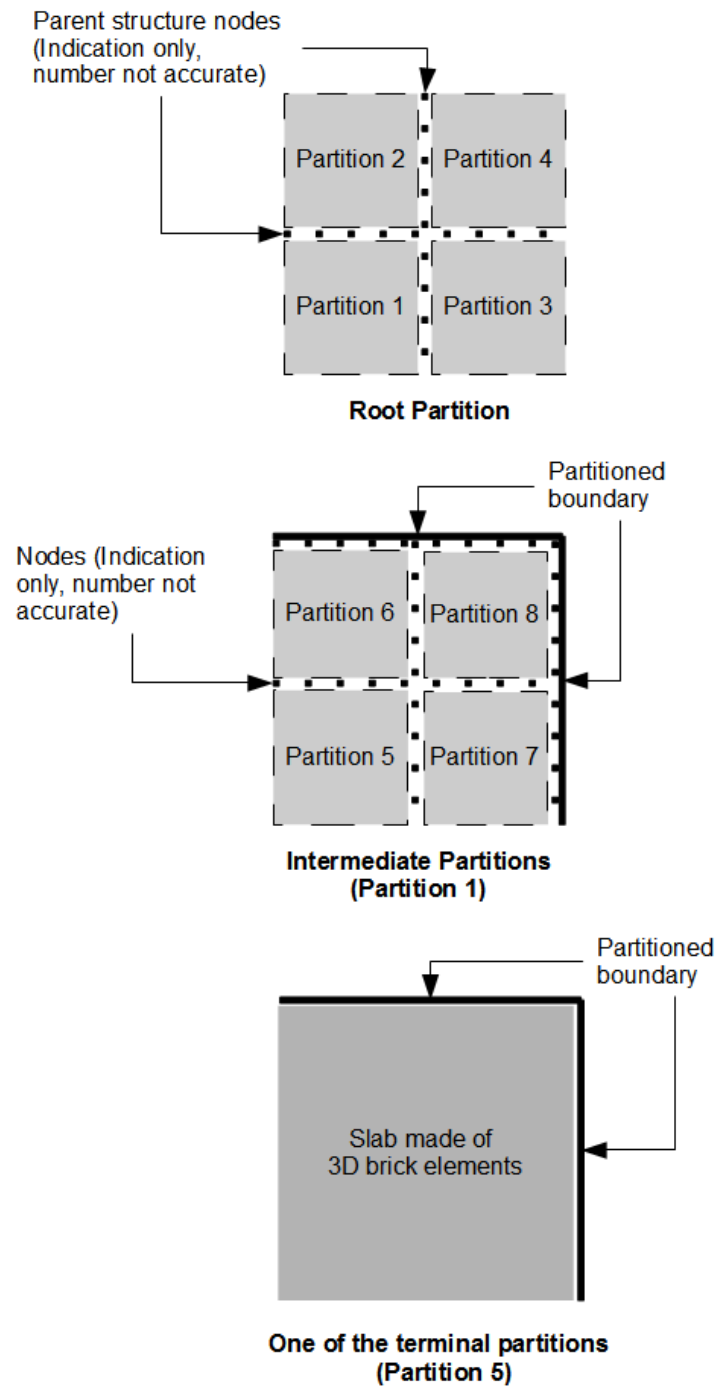


Figure 6-12: Hierarchical partitioning model for square slab

A comparison of the wall clock time required in the case of 16 terminal partitions with hierarchic partitioning to that in the case of 16 partitions with a single level partitioning as well as 4 partitions is shown in Figure 6-13. It can be seen that the time required for the hierarchic partitioned model is less than half that of the single level partitioned model. It requires less time than the model with 4 partitions, which emphasises the benefits of hierarchic partitioning even for relatively fine decomposition. It is noted that only 27 Gauss points are used here for the 20-noded brick elements, hence the finite element computations within the partitions are not expensive, and most of the computational efficiency with hierarchic modelling arises as a result of enhancement in the frontal solution.

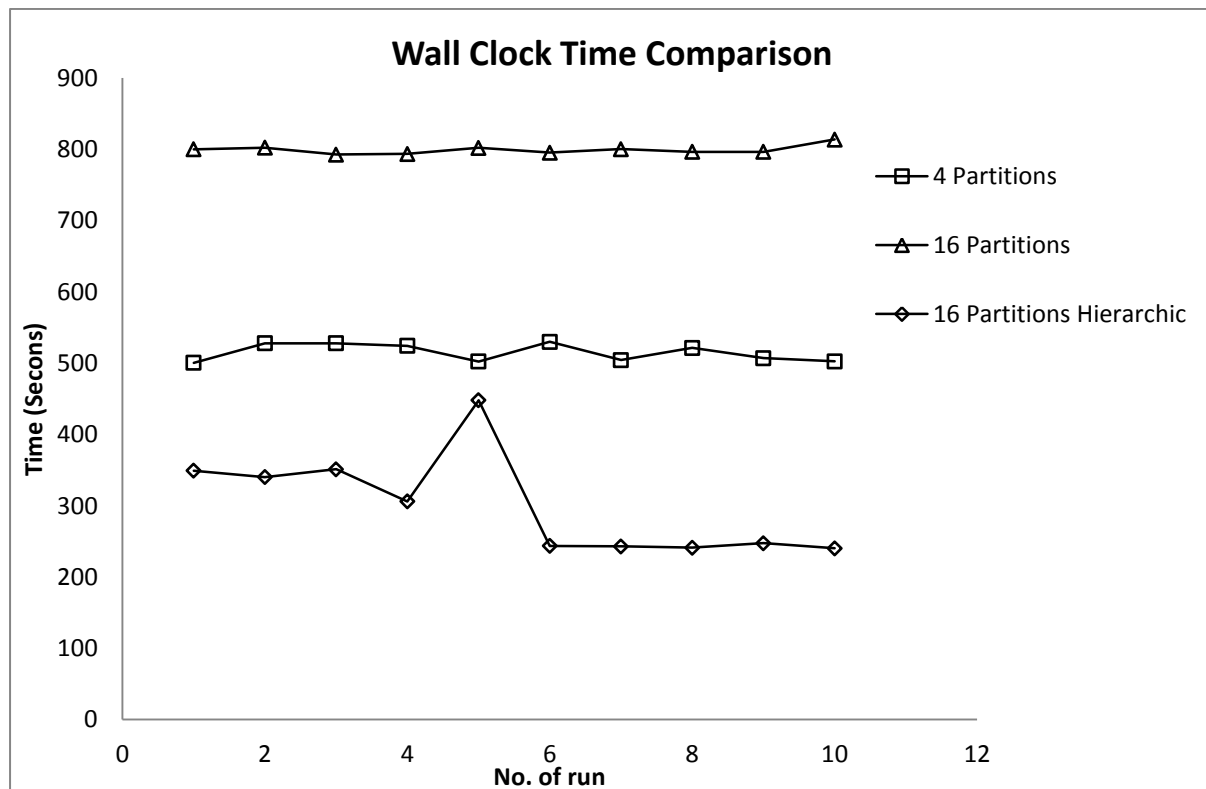


Figure 6-13: Comparison of wall clock time for hierarchic partition of the slab model

Case Studies

This chapter presents four case studies that demonstrate some of the applications of the mixed dimensional hierarchic partitioned modelling approaches developed in this work.

7.1 Introduction

The application of the hierarchic domain decomposition methods and the mixed dimensional coupling is not limited to particular types of structure, elements, or materials. These methods can be used with any of the analysis methods, element types, and material models available with the software they are implemented with, which is ADAPTIC [Izzuddin, 1991] in this case. This chapter illustrates some of the many ways in which the developed methods can be used. A total of four case studies are presented that demonstrate the application of single level partitioning, hierarchic partitioning, and dimensional coupling.

The first case study is a cellular beam which demonstrates the use of single level partitioning as well as the ease of modelling achieved through modular modelling. The beam is divided into 32 partitions with every partition being identical except the two partitions at either end of the beam, making the modelling relatively easy.

The second case study is a square frame with masonry infill, which is modelled using 3D brick elements in conjunction with interface mortar elements. This case study uses dimensional coupling, where the parent structure consists of nodes with 6 DOF each and contains the frame elements, while the partitions consist of 3D continuum elements with 3 DOF nodes and contain the brick and interface mortar elements. Another related case study is presented in the same section which consists of a realistic brick masonry wall with openings. This example illustrates the application of partitioned modelling to real problems imposing a memory bottleneck.

The fourth and final case study is an approximate model of a suspension bridge where the bridge deck is modelled with 2D composite steel-decked concrete slab elements, and the

beams, columns, and cables are modelled with 1D beam-column elements. This case study illustrates the use of dimensional coupling along with hierarchic partitioning, where initially the bridge is divided into piers and 4 segments. The deck slab of each segment is further divided into 16 partitions, thus making a total of 69 partitions with 3 levels of partitioning.

7.2 Cellular Beam

The first case study considered 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, resulting in floors of a much smaller overall depth compared to conventional steel beams. 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 is currently ongoing at Imperial College London as part of an independent PhD research programme [Zainal Abidin & Izzuddin, 2011; Zainal Abidin, 2012]. 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 when utilised with parallel processing.

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 7-1. 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 7-2. The web posts between two consecutive holes have a minimum width of 120 mm.

The entire model is made of elements of type 'cvs9', which are shell elements that can be used to model 2D slabs as well as flanges and webs of frame elements. Further details about this element can be found elsewhere [Izzuddin, 2007].

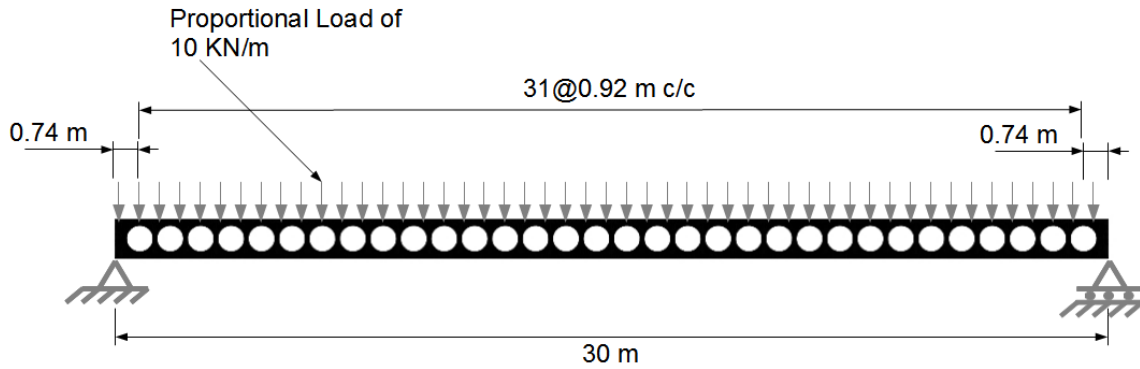


Figure 7-1: Side elevation of the cellular beam

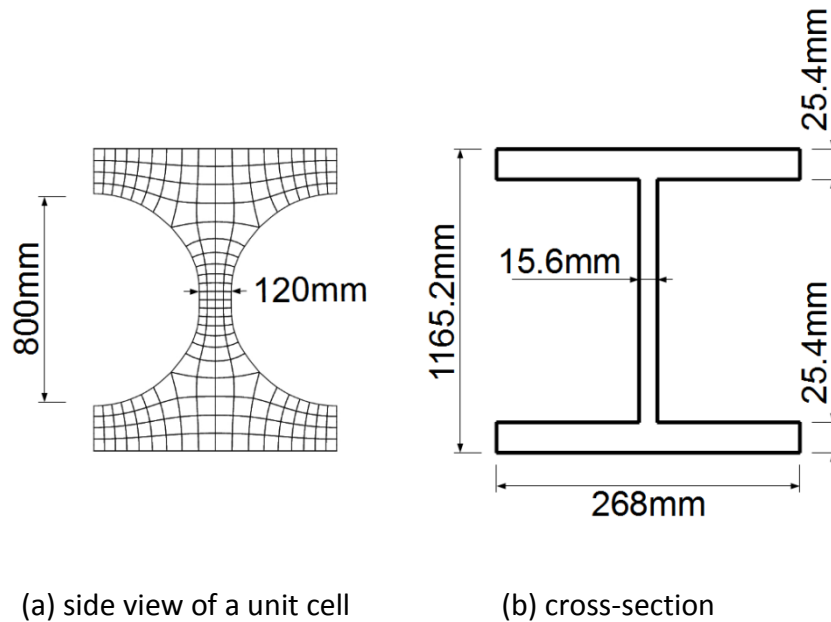


Figure 7-2: Side elevation of a unit cell and cross-section of the cellular beam

The domain partitioning for this example is quite simple 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 mesh used for a typical unit cell can be seen in Figure 7-2 (a).

The cellular beam is subjected to a proportional 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 in the web posts 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 shapes at limit point are shown in Figure 7-3 and Figure 7-3 whereas the final deflected shapes are shown in Figure 7-5 and Figure 7-6. For illustrative purposes only a close up of partitions 29 and 30 with contours of the normal stress in the longitudinal direction is provided in Figure 7-7 to Figure 7-10. It can be seen that, 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.

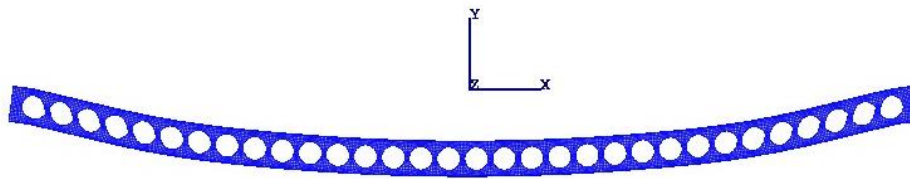


Figure 7-3: Deflected shape of the cellular beam at limit point (Displacement scale = 5)

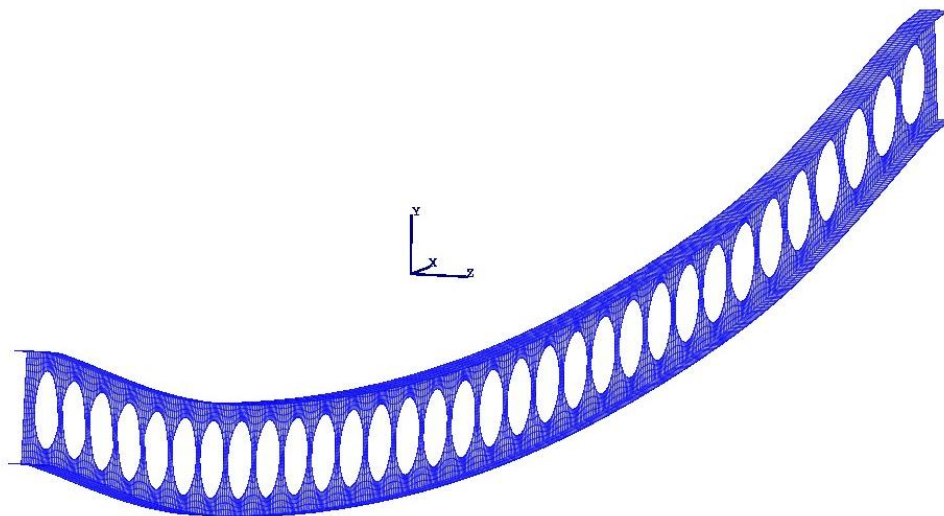


Figure 7-4: Deflected shape of the cellular beam at limit point (Displacement scale = 5)

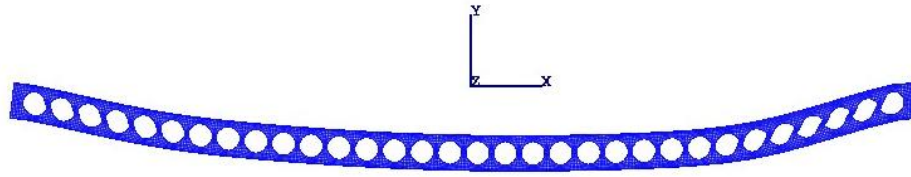


Figure 7-5: Final deflected shape of the cellular beam (Displacement scale = 5)

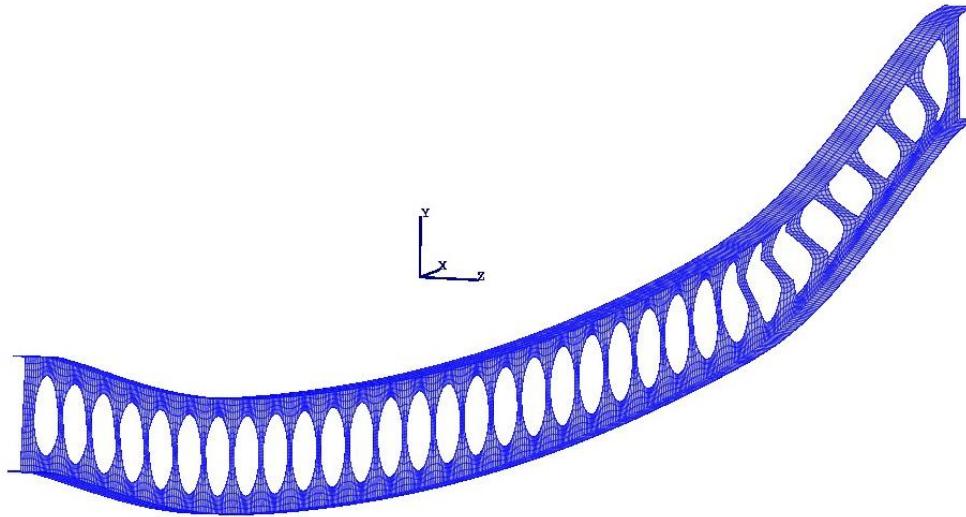


Figure 7-6: Final deflected shape of the cellular beam (Displacement scale = 5)

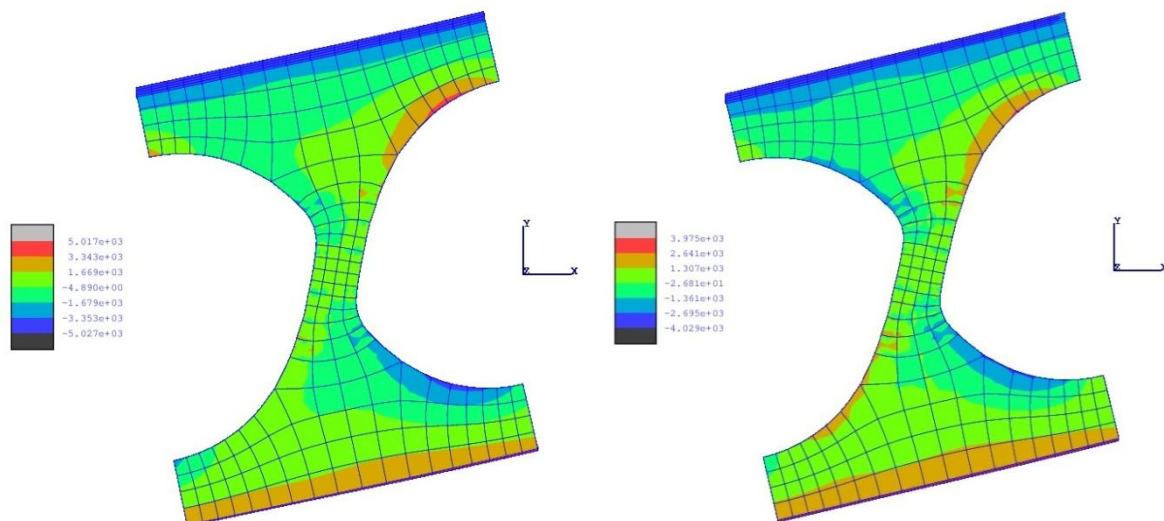


Figure 7-7: Contours for normal traction in longitudinal direction at limit point for partitions 29 and 30 (Units: N/m as thickness is included in stress resultant)

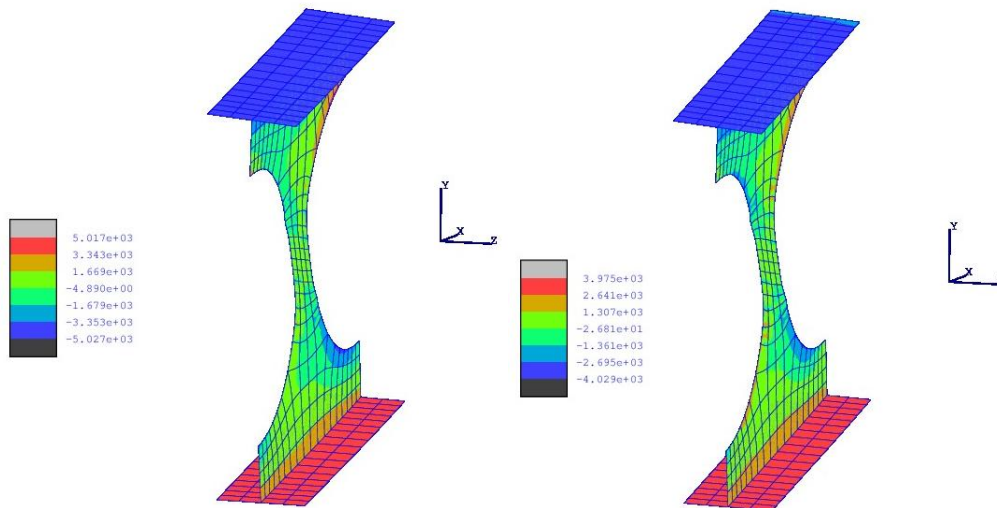


Figure 7-8: Contours for normal traction in longitudinal direction at limit point for partitions 29 and 30 (Units: N/m as thickness is included in stress resultant)

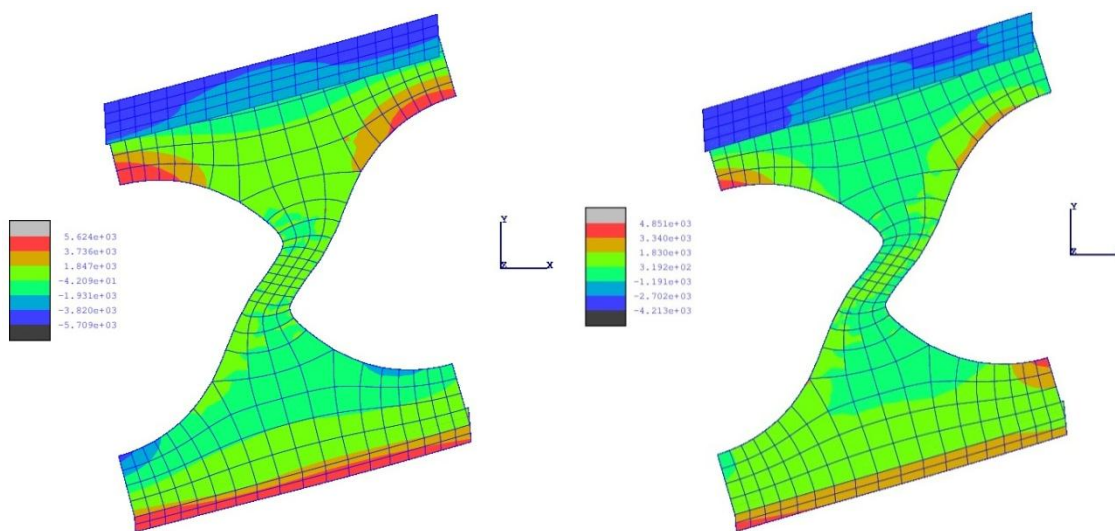


Figure 7-9: Contours for normal traction in longitudinal direction at final stage for partitions 29 and 30 (Units: N/m as thickness is included in stress resultant)

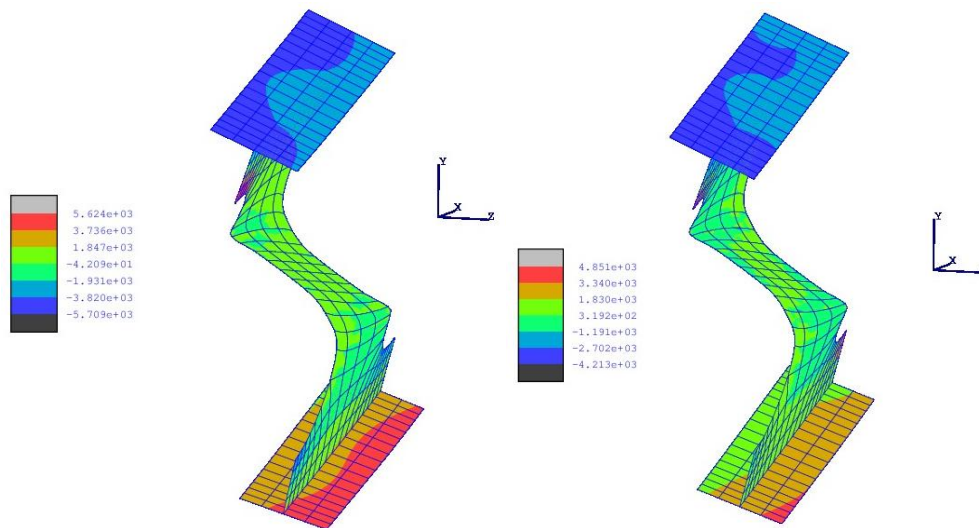


Figure 7-10: Contours for normal traction in longitudinal direction at final stage for partitions 29 and 30 (Units: N/m as thickness is included in stress resultant)

The load-deflection response of the beam is provided in Figure 7-11, where it is clear that the arc-length method is successful with the proposed partitioned approach in tracing the snap-back post-buckling response. Importantly, the whole analysis is undertaken on 34 processors in 30 minutes of wall-time when it would have taken over 16 hours with a monolithic model besides the excessive memory requirements that such a model demands.

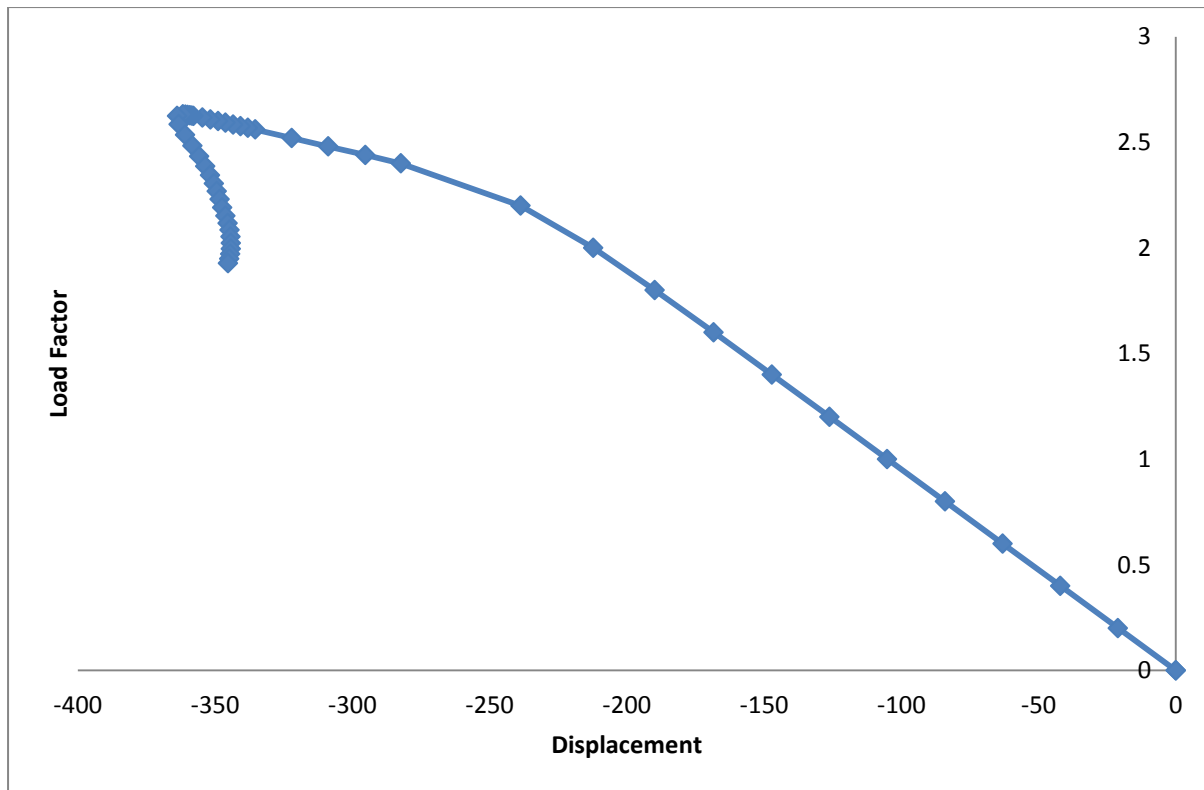


Figure 7-11: Displacement at mid-span in the cellular beam

7.3 Masonry Structures

7.3.1 Frame with Masonry Infill

The case study presented here relates to a steel frame with masonry infill, as shown in Figure 7-12. The frame members consist of rectangular solid steel cross-sections, and the masonry infill consists of bricks as illustrated in Figure 7-13.

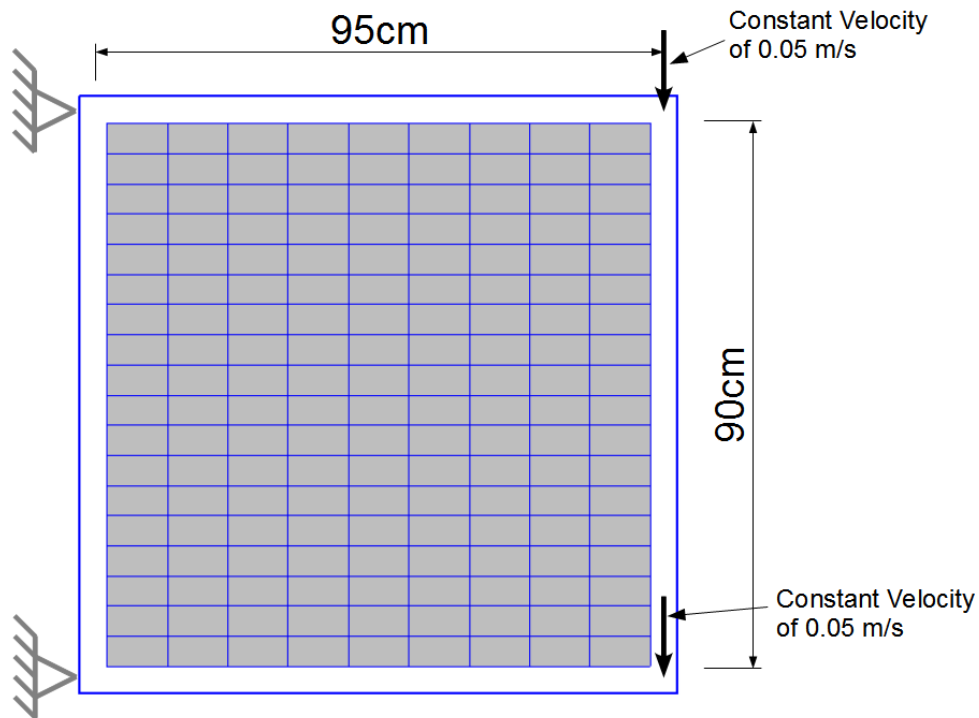


Figure 7-12: Frame with masonry infill

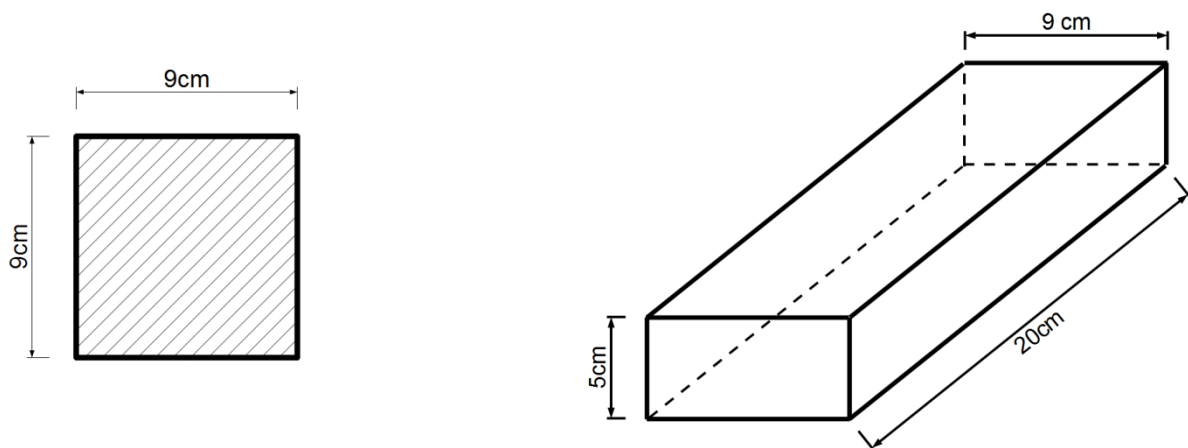


Figure 7-13: Cross-section of steel members and brick dimensions

7.3.1.1 Structural Model and Element Types

The beams and the columns of the frame are modelled with the element type 'cbp3', which is a cubic elasto-plastic 3D beam column element [Izzuddin & Elnashai, 1993]. Normally 6 or

more elements are required per member to capture the spread of plasticity along the member length; however, due to the nature of the problem as described below, the number of elements required here is significantly greater.

The masonry infill is divided into 16 partitions and joined to the surrounding frame through the use of dimensional coupling. Each partition is modelled with a 20-noded 3D brick elements of type 'bk20' [Izzuddin, 2009], and the mortar joining the bricks together is modelled with the 16-noded interface element type 'in16' [Macorini & Izzuddin, 2010]. Each brick is modelled using 2 elements with an additional interior interface element, which has different properties to the mortar interface elements, so as to allow possible crack development inside the brick, as shown in Figure 7-14.

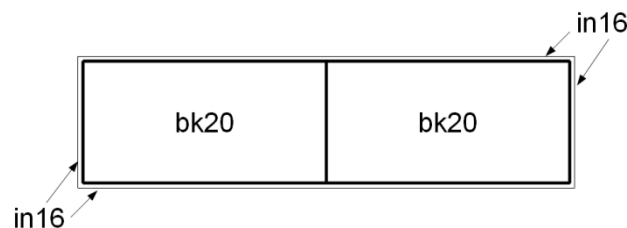


Figure 7-14: Model for a single brick

Although the interface elements modelling mortar 'in16' [Macorini & Izzuddin, 2010] are shown outside the brick model for clarity, these elements are in fact zero-thickness to start with and coincide with the brick faces on either side. Later, as the analysis progresses, these elements allow the separation between adjoining bricks due to cracks or slip planes developing in the masonry. When these nonlinear interface elements are used, convergence difficulties can arise in static analysis due to the softening characteristics of cracked mortar [Macorini & Izzuddin, 2010]. In order to reduce these difficulties and determine the solution up to a significant level of deformation, especially for the case of high normal pressure, a dynamic analysis procedure is utilised [Macorini & Izzuddin, 2010], allowing the sudden release of elastic energy to be balanced by kinetic and viscous energy. This approach is being used in this example as well, where a small constant velocity is applied at the two

right hand corner nodes in downward direction leading to a linearly increasing displacement with time.

7.3.1.2 Partitioning with Dimensional Coupling

The masonry infill is divided into 16 partitions, with 4 different partition sizes, as shown in Figure 7-15.

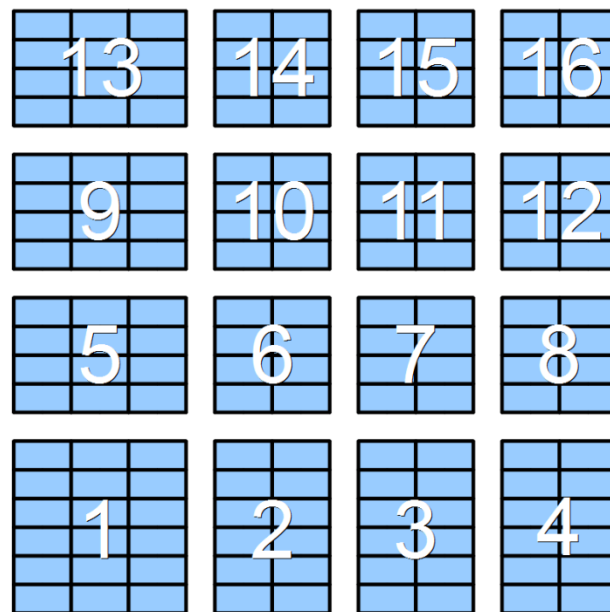


Figure 7-15: Partitioning of the masonry infill

All 16 partitions of the masonry infill are children to a single higher level partition which in turn is the child to the root partition that contains the frame elements. The intermediate partition, shown in Figure 7-16, is used to employ dimensional coupling between the frame elements and the masonry infill whilst the partitioned boundaries of the masonry parts are joined without dimensional coupling. For clarity, the locations of the master nodes in Figure 7-16 are not shown to scale; in reality, these nodes are located along the centre line of the frame members. In this example, one dimensional coupling element is connected to each

row of 3D nodes; however, it is also possible to reduce the number of dimensional coupling elements by connecting each to a larger group of 3D nodes.

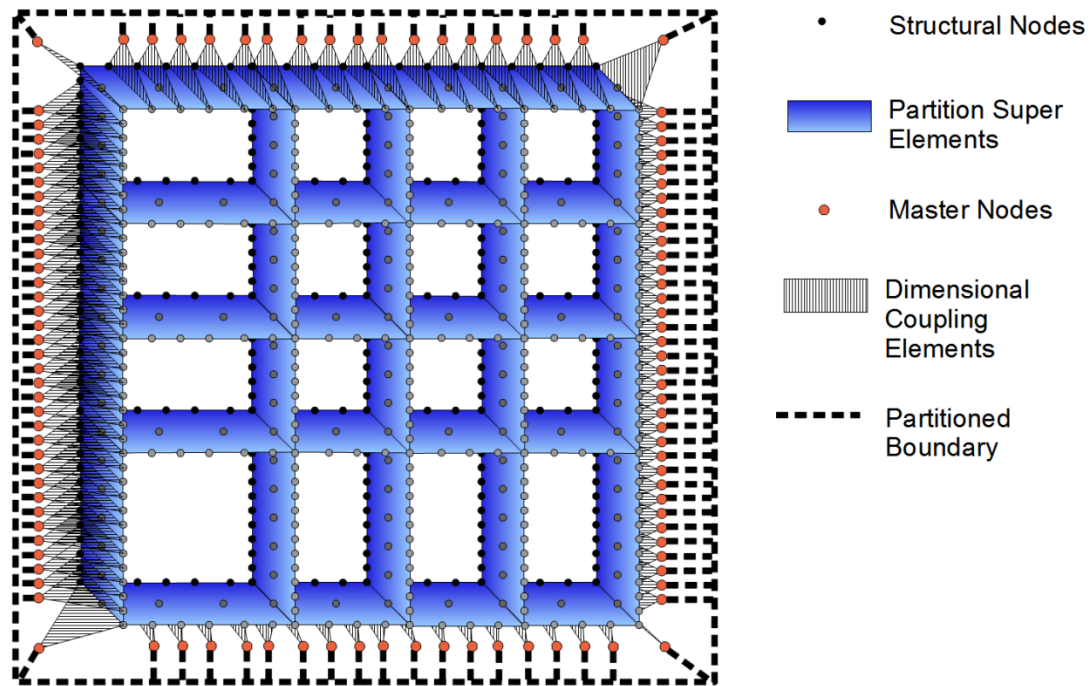


Figure 7-16: Intermediate partition (level 2) with dimensional coupling

It is noted that the use of an intermediate partition can be avoided by using dimensional coupling between the masonry infill parts as well. The root partition for the current example consists of 1D beam column elements and a single partition super element as shown in Figure 7-17.

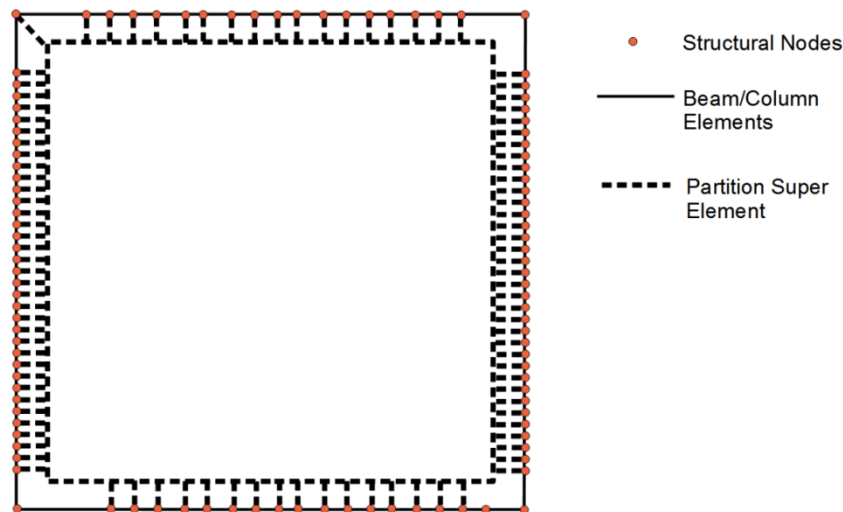


Figure 7-17: Root partition (level 1)

7.3.1.3 Material Model

The material model used with the brick elements is elastic isotropic with a modulus of elasticity of 2.5×10^5 MPa and a Poisson's ratio of 0.15. The material properties for the mortar and brick interface elements are shown in Table 7-1, a detailed explanation of which can be found elsewhere [Macorini & Izzuddin, 2010].

7.3.1.4 Analysis and Results

All partitions are subjected to an initial load of $19 \text{ kN} / \text{m}^3$ as self-weight of the bricks. An initial velocity of $0.05 \text{ m} / \text{s}$ is applied to the free corner nodes (Figure 7-12) of the parent structure. As a dynamic load, an acceleration of $0.0 \text{ m} / \text{s}^2$ is applied, which means that the initial velocity remains constant, effectively resulting in a linearly increasing displacement applied to these nodes.

The analysis ran for a total of 10,539 loading steps in just about 38 hours, where the final deflected shape at a lateral drift of 0.7 mm is shown in Figure 7-18.

	Elastic properties	Surface F_1	Surface Q_1	Surfaces F_2, Q_2
Mortar interface	$k_n = 48N / mm^3$ $k_{t1} = 21N / mm^3$ $k_{t2} = 21N / mm^3$	$\sigma_{t0} = 0.15MPa$ $C_0 = 0.35MPa$ $\tan \phi_0 = 0.6$ $\tan \phi_r = 0.6$ $G_{f.I} = 0.02N / mm$ $G_{f.II} = 0.05N / mm$	$\sigma_{t0} = 0.15MPa$ $C_0 = 0.35MPa$ $\tan \phi_0 = 0.0$ $\tan \phi_r = 0.0$	$\sigma_{c0} = 6.0MPa$ $\sigma_{cr} = 0.0MPa$ $D = 5.0MPa$ $\tan \phi_0 = 1.0$ $\tan \phi_r = 1.0$ $G_c = 5.0N / mm$
Internal brick interface	$k_n = 1 \times 10^4 N / mm^3$ $k_{t1} = 1 \times 10^4 N / mm^3$ $k_{t2} = 1 \times 10^4 N / mm^3$	$\sigma_{t0} = 2.0MPa$ $C_0 = 2.8MPa$ $\tan \phi_0 = 1.0$ $\tan \phi_r = 1.0$ $G_{f.I} = 0.08N / mm$ $G_{f.II} = 0.5N / mm$	$\sigma_{t0} = 2.0MPa$ $C_0 = 2.8MPa$ $\tan \phi_0 = 1.0$ $\tan \phi_r = 1.0$	

Table 7-1: Material properties for the mortar and brick interface elements

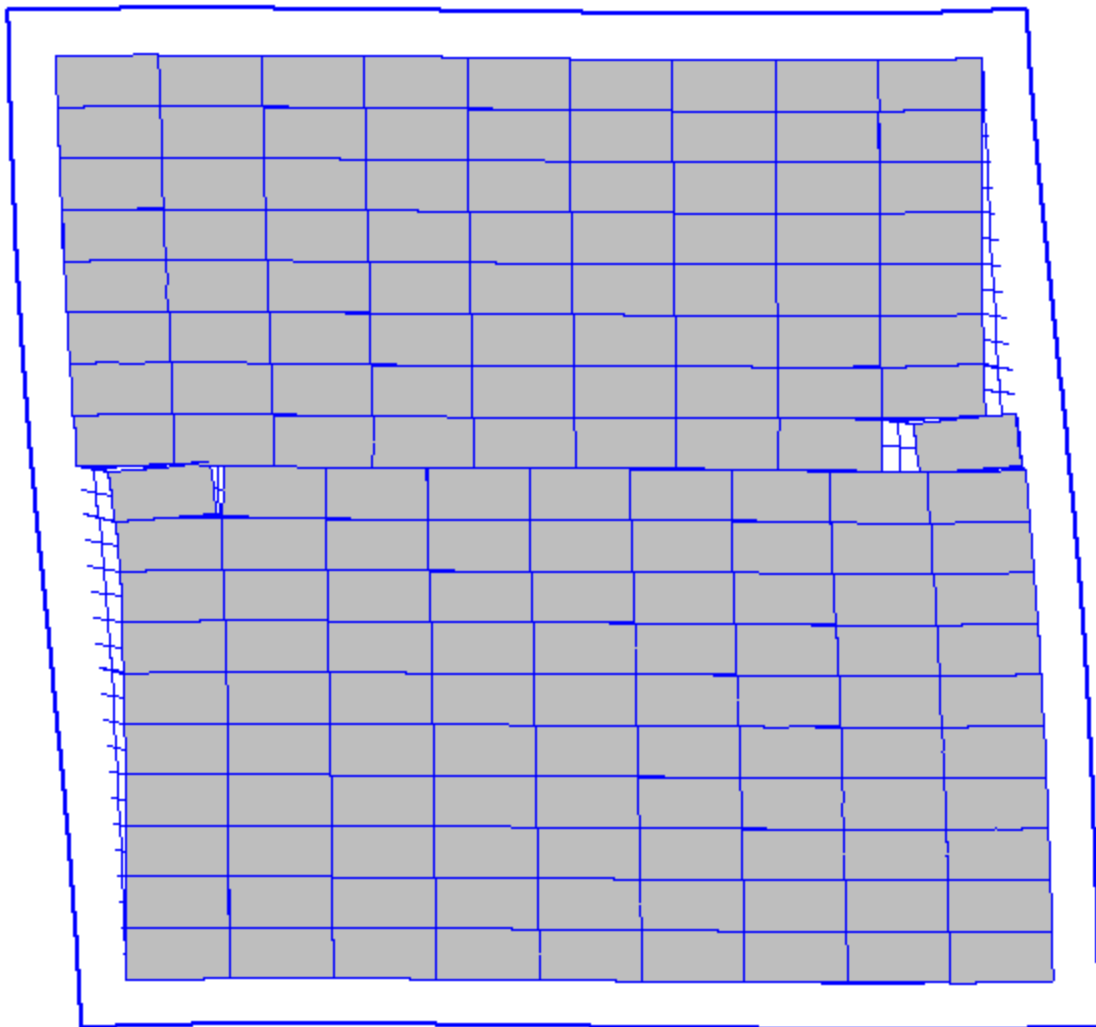


Figure 7-18: Deflected shape at 0.7 mm lateral drift (displacement scale = 100)

Figure 7-19 shows the plastic work at the interfaces due to the applied lateral drift. Further detailed analysis of the behaviour of this structure is out of the scope of this study. However, the example has illustrated the use of dimensional coupling along with domain partitioning, which has rendered the nonlinear dynamic analysis of such a structure not only practical but computationally feasible.

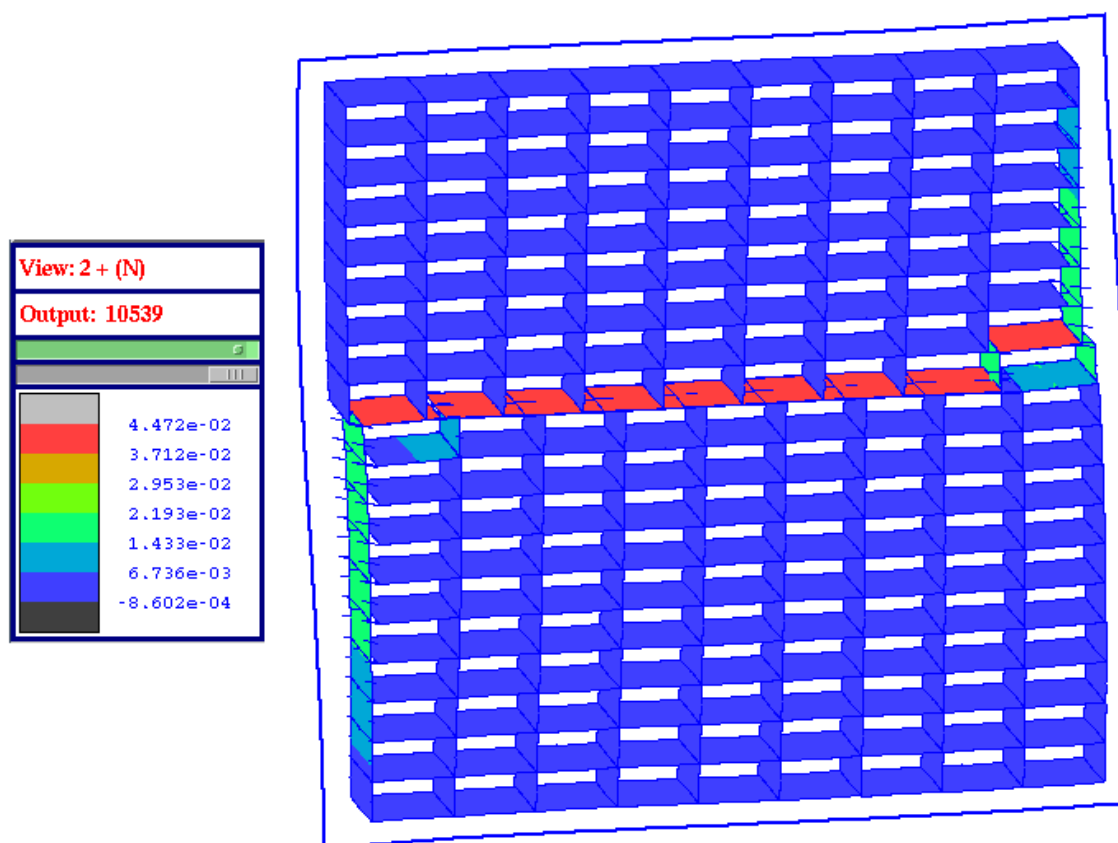


Figure 7-19: Contours showing plastic work at the interfaces

7.3.2 Unreinforced Perforated Masonry Wall

The domain decomposition techniques developed in this work have been used by Izzuddin and Macorini (2011), where they have performed multi-scale modelling of unreinforced masonry (URM) structures. The present example is being reproduced with permission from their paper as referred to above. In this example, an URM structure is partitioned into 62 subdomains, where each partition is modelled by a detailed 3D mesoscale model, similar to the example in Section 7.3.1, accounting for material and geometric nonlinearity. The use of strong coupling between the macro and the mesoscales instead of the representative volume element (RVE) has made the relatively accurate analysis of such a complex structural model practical.

The full-scale perforated URM wall has been analysed under in-plane loading. Due to the large dimension of the structure as shown in Figure 7-20, the detailed 3D mesoscale strategy, which requires 162840 nodes to represent the whole URM perforated wall, cannot be employed in a monolithic model using ordinary memory and CPU resources, thus requiring the use of the developed domain decomposition strategy with parallelisation.

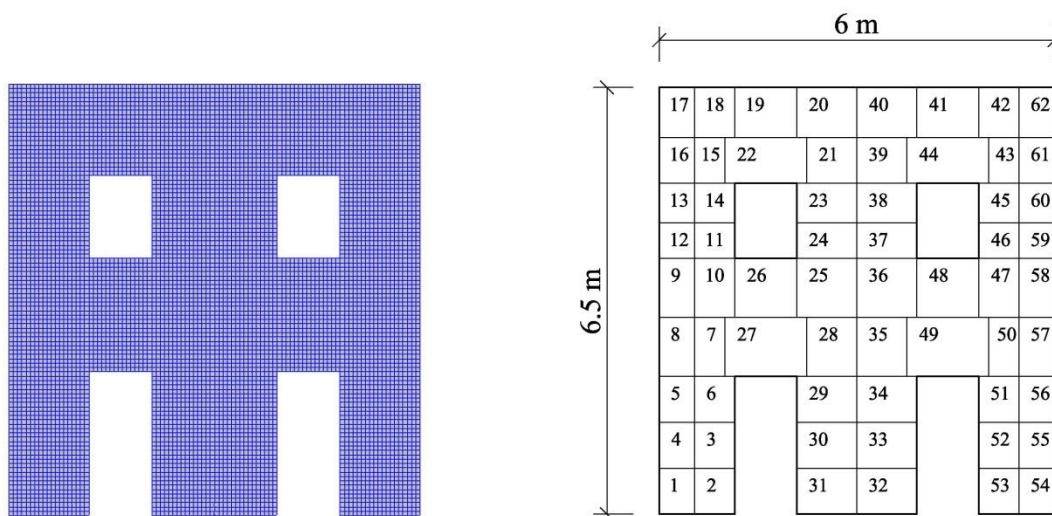


Figure 7-20: Mesh and partitions scheme used for representing a full-scale URM perforated wall [Izzuddin & Macorini, 2011]

Figure 7-20 shows the mesh with solid and interface elements for the whole structure and the arrangement for the 62 partitions employed for representing the structural system. The deformed shape including the cracks in the wall with details of damage at interface elements is depicted in Figure 7-21(a), while the contours of maximum principal stresses in solid elements are shown in Figure 7-21(b). The results achieved confirm the potential of the proposed domain decomposition approach, which allows the investigation of full-scale URM structures, while maintaining the accuracy characteristics of mesoscale models for a realistic representation of cracks and damage.

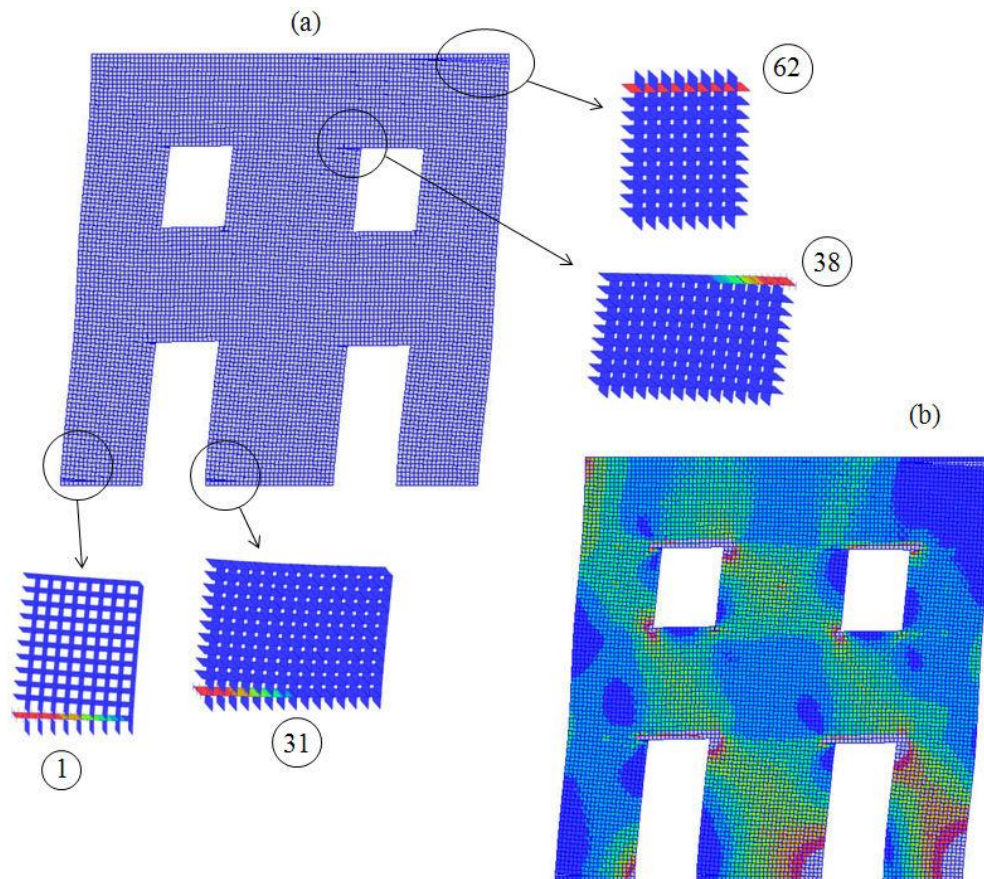


Figure 7-21: (a) Crack development in full-scale URM perforated wall, (b) maximum equivalent stresses in solid elements [Izzuddin & Macorini, 2011]

7.4 Cable Stayed Bridge

Cable Stayed and suspension bridges have grown in popularity over the recent years because of their ability to cover large spans without the need for closely spaced piers. The Akashi-Kaiko Bridge in Japan has the world's largest single span that covers approximately 2 km.

With extremely large spans and a load carrying mechanism in which the cables supporting the bridge deck can resist tension only, these bridges face a lot of challenges in resisting non-gravitational loads such as earthquakes, wind, blast, etc. Several solutions have been developed to overcome these challenges ranging from the design of the deck inspired by aircraft wings to the introduction of dampers which keep the deck firmly in place when subjected to wind loads but allow it to move relatively freely in the case of earthquakes.

These advanced solutions to the problems facing these bridges owing to non-gravitational loads make the analysis of such structures more complicated requiring detailed modelling. The example presented here is a hierarchic model of a cable stayed bridge which consists of reinforced concrete deck slab, steel cables, steel beams and steel piers. Although this example reflects only the general structural features of the bridge, it does illustrate how the mixed dimensional hierarchic partitioning method developed in this work may be used for the detailed modelling of such structures.

The bridge under consideration consists of a single central span of 72 meters and two cantilever side spans of 36 meters each (Figure 7-22).

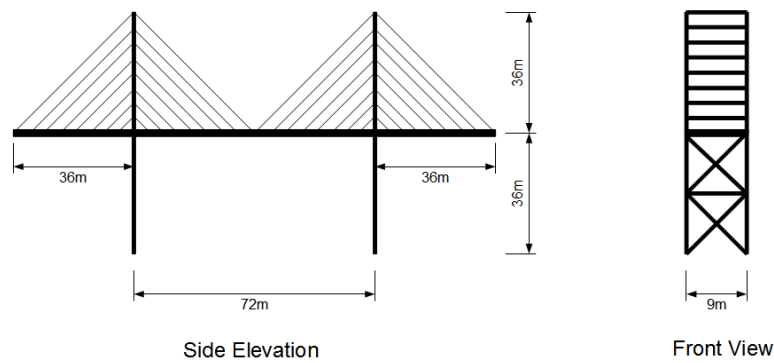


Figure 7-22: Suspension bridge

7.4.1 Structural Model

The deck is modelled as a floor system consisting of a reinforced concrete slab supported by a grid of steel beams that are tied to the cables as shown in Figure 7-23 for one of the side spans. In real bridges, box girders are typically used to support the deck; however, the present scheme is adopted because the focus is on application of the partitioned approach more than on the actual mechanical behaviour of the structure.

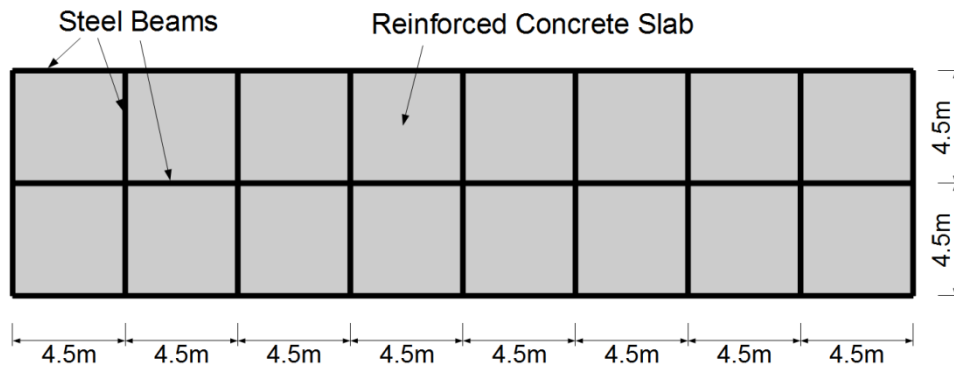


Figure 7-23: Deck slab for one of the side spans

Each end of the transverse beams in the floor system is tied to a cable, the other end of which is tied to the bridge column. All the cables are at 45° with the horizontal deck slab. The piers are braced with horizontal struts in the portion above the deck slab and with cross bracings in the portion below as an approximate imitation of the Golden Gate bridge, although the cabling system in this example is kept simple.

The two side spans are assumed to act as cantilevers, whereas the centre span is a continuous one with the transverse beam in the middle tied to the top of both columns through cables. The weight of the deck slab is transferred via tension cables to the columns which act in compression. The columns are assumed to be fixed at the base in all directions, except for the direction in which the ground acceleration is applied when the bridge is subjected to an earthquake.

7.4.2 Element Types

The reinforced concrete slab is modelled with the elements type 'csl4' [Izzuddin *et al.*, 2004]. These are 2D flat shell elements which can be used for the realistic modelling of composite floor slabs under extreme loading conditions including fire. There are a total of 4 element sub-types available for this element including left edge rib, cover, central rib, and right edge rib; however, only the element type 'cover' is used for the present example.

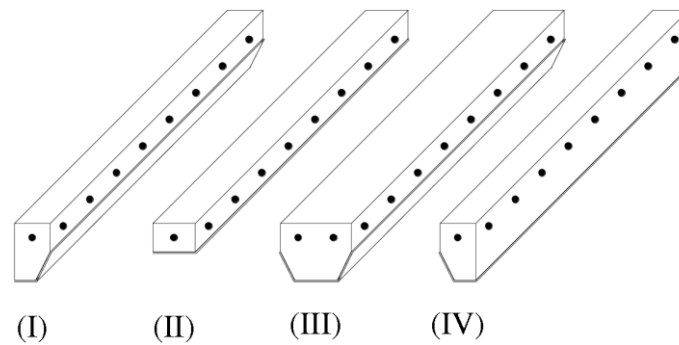


Figure 7-24: Element types for 'csl4' (I) left edge rib (II) cover (III) central rib (IV) right edge rib [Izzuddin *et al.*, 2004]

This element type 'csl4' has up to 32 additional freedoms associated which need to be considered if one of its sides is at the partitioned boundary in case of partitioned analysis. In the bilinear form only 8 additional freedoms are required, whereas the remaining 24 additional freedoms, as shown within the dotted lines in Figure 7-25, are required only if the element is employed in the full quadratic form. In either case, additional freedoms which are on the partitioned boundary should be coupled across the partitions. For example, in its bilinear form, if nodes 2 and 3 of one such an element (Figure 7-25) in partition 1 are on the partitioned boundary which is shared by another such element in another partition, say partition 2, which has its nodes 1 and 4 on the partitioned boundary, then additional freedom $\bar{w}_2$ of the first element should be coupled with the additional freedom $\bar{w}_4$ of the second element. This coupling of additional freedoms is indicated at the level of the parent structure, while the actual additional freedoms to be coupled are specified on the partitioned boundary of the dual super elements. Typical data files enabling the specification of coupled additional freedoms at the child and parent partition levels are provided in Appendix C.

In the present example, however, no coupling of additional slab element freedoms is required, since the slab is connected at the partitioned boundary directly to the beams in the parent partitions. Since slab nodes have 5 DOF (3 translations and 2 rotations), whereas the beam nodes in the parent partitions have 6 DOF, a special link element is used which connects at one node to the slab, while its other node is at the partitioned boundary. This type of model imposes compatibility at the nodes only, and therefore there are some

negligible compatibility defaults arising within the length of the beam element and side of the slab element due to the different shape functions of the transverse displacement field (i.e. cubic function for the beam element vs quadratic function for the basic slab element).

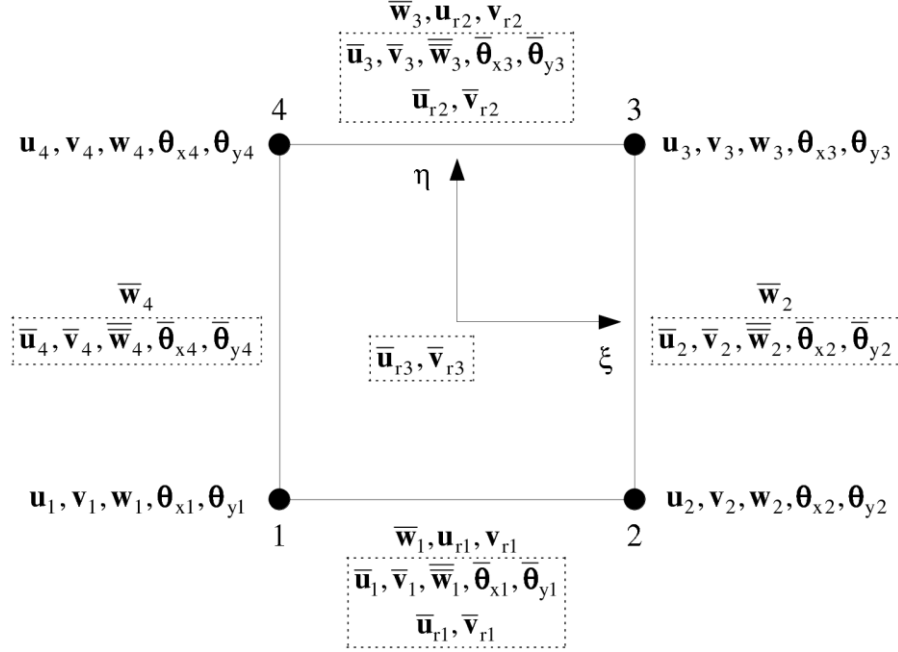


Figure 7-25: Local freedoms for element 'csl4' [Izzuddin *et al.*, 2004]

The link element used at the partitioned boundary of the slab partitions is element type 'lnks' [Izzuddin, 2003, 2011]. This is a 3D link element which can link a 6 DOF node to a 5 DOF node, and which requires specification of 5 stiffness parameters (Figure 7-26), all taken in this case as '**rigid**'.

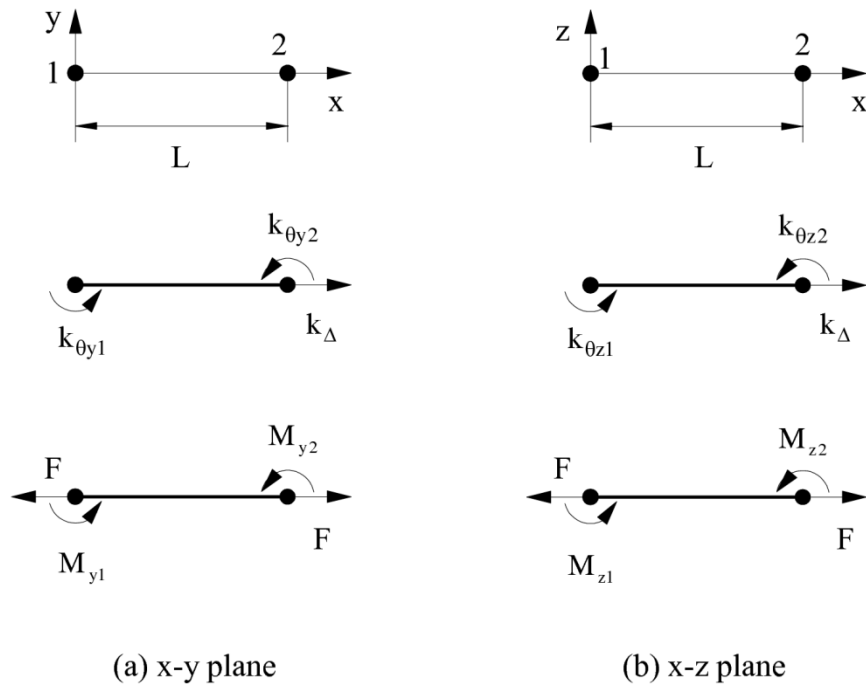


Figure 7-26 Stiffness parameters and local forces element type 'lnks' [Izzuddin, 2011]

The beams and columns as well as the cables are all modelled using the quartic elastic 3D beam-column element 'qdp3' [Izzuddin & Elnashai, 1993; Izzuddin, 2011] that utilises automatic mesh refinement and subdivides into cubic elasto-plastic 3D beam-column element 'cbp3' where and when material inelasticity is detected.

7.4.3 Partitioning Scheme

The bridge is divided into a total of 69 partitions with 3 levels of hierarchic partitioning. The root partition (Figure 7-27) consists of the bridge piers and 4 partition super elements that represent the segments of the bridge deck along with the cables. Each of the 4 bridge segments is modelled as a separate partition, which consists of the beams of the bridge floor system and the tension cables that join the bridge deck to the columns.

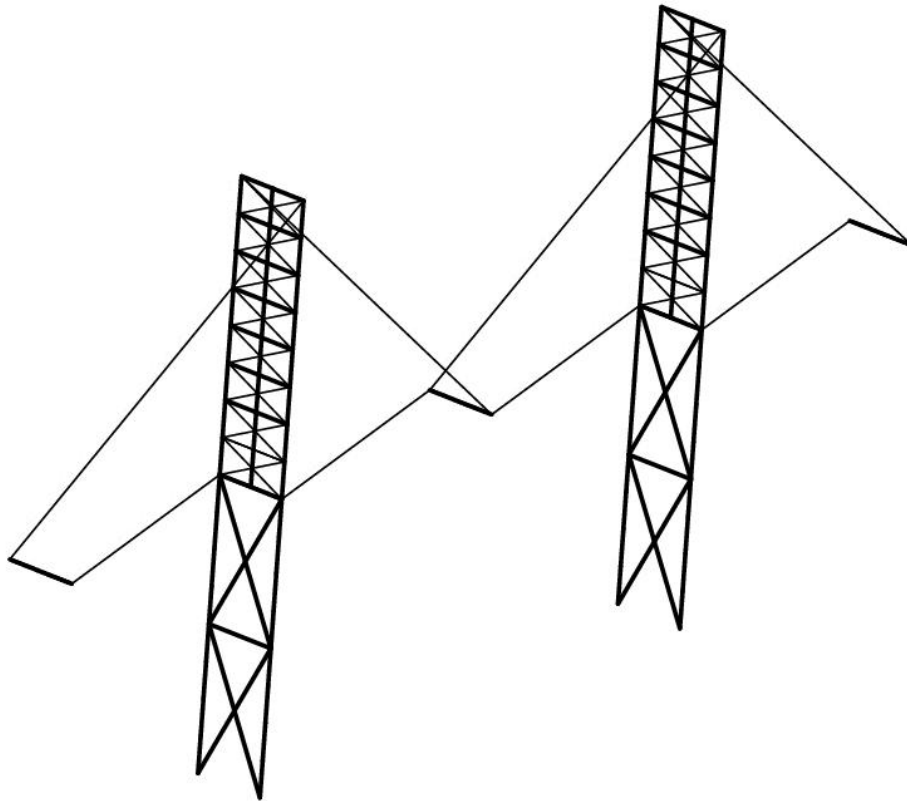


Figure 7-27: Root partition for the bridge

Since the 4 level 2 partitions modelling the segments of the bridge deck are exactly the same (Figure 7-28) with the only difference being their location and orientation relative to the parent structure, they were created by making 4 copies of the same data file, as illustrated in Appendix A, and then making the appropriate rotational/translational transformations via the 'coordinates.transformation' module.

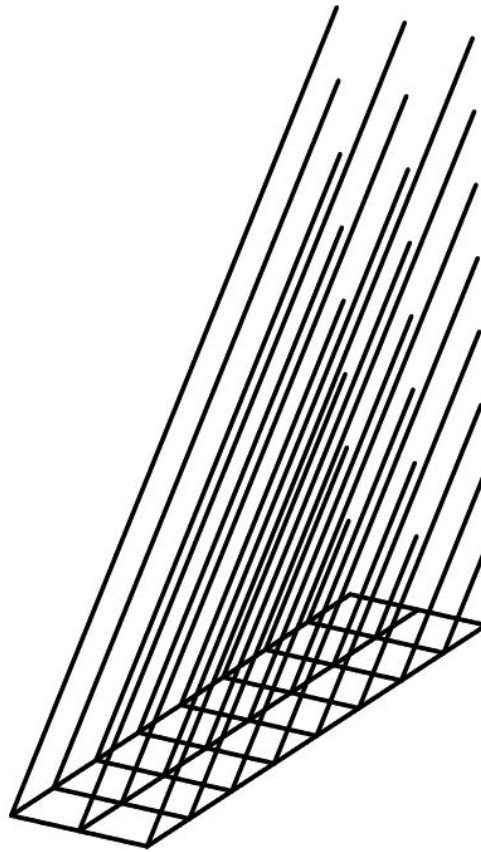


Figure 7-28: A typical level 2 partition for bridge deck and cables

Each of these second level partitions utilise 16 level 3 child partitions thus completing the hierarchic model. The last level partitions are all exactly the same (Figure 7-29) consisting of a square slab made with 'csl4' elements that are joined with the link 'lnks' elements at the edges, with the other nodes of the link elements forming the partitioned boundary. All 64 of these partitions were created by making copies of a single data file and then again making the appropriate geometric translational transformation via the 'coordinates.transformation' module.

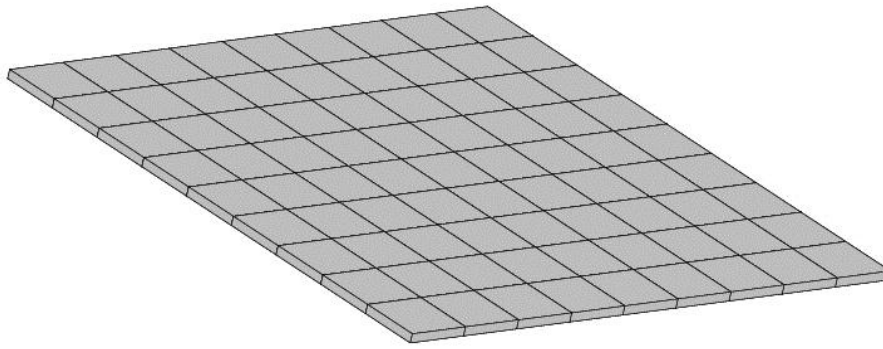


Figure 7-29: Level 3 slab partitions

7.4.4 Analysis and Results

The bridge is subjected to an earthquake by applying accelerations to the base of piers according to the record used in Chapter 3 (Figure 3—15). While a detailed discussion on the results and response of the bridge is not the aim of this study, the deflected shape of the bridge is provided as an illustration in Figure 7-30.

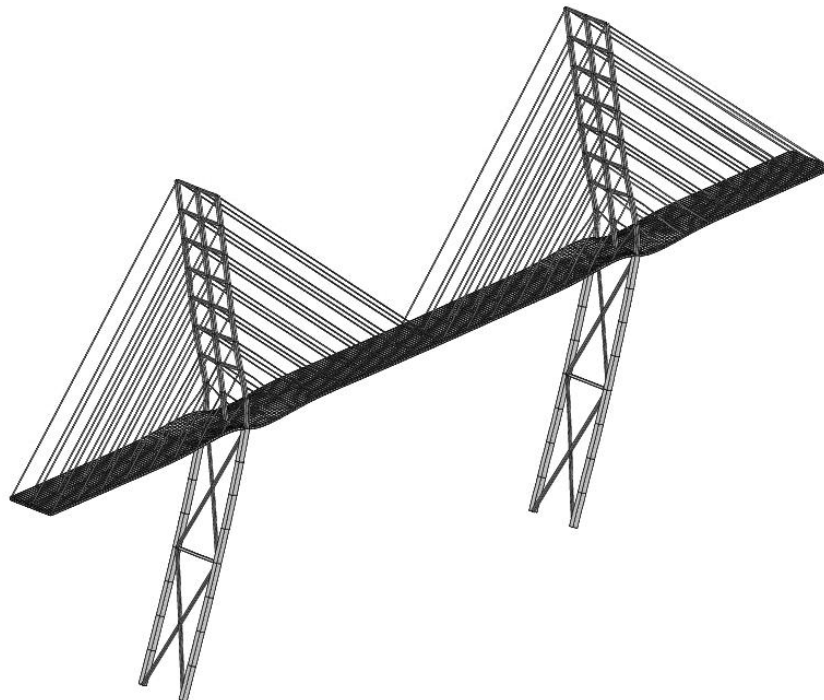


Figure 7-30: Deflected shape of the bridge

Figure 7-31 depicts the deflected shape of a bridge segment showing the axial force in the beams.

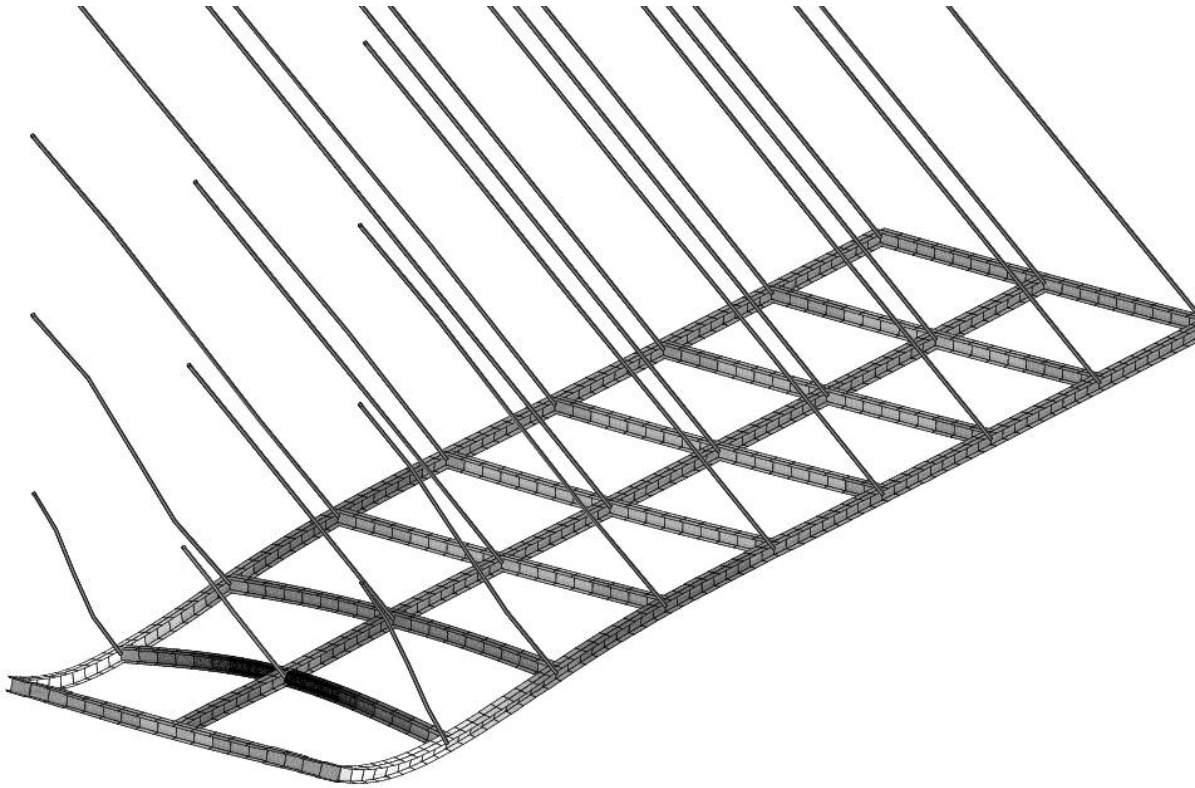


Figure 7-31: Deflected shape of the deck beams with contours showing axial force

The same bridge is also subjected to the same earthquake but this time in the transverse direction. In this case, the bridge behaves differently as shown in Figure 7-32. It can be observed that the bridge piers are bent in the direction of the earthquake, while the bending of the bridge deck due to gravitational loads is also evident.

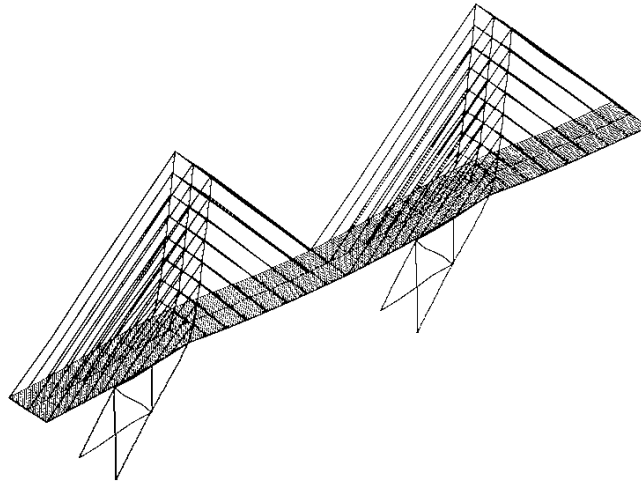


Figure 7-32: Deflected shape for earthquake in transverse direction

The analysis time for this bridge was a mere 18 minutes and 25 seconds for the 200 time steps carried out, despite the fact that only 24 processors were used for running the 69 partitions simultaneously. If every partition were to run on a separate processor, it is expected that the wall-clock time required for the analysis would be even smaller.

Summary, Conclusions and Future Research

This chapter summarises and concludes this thesis and presents recommendations for future research.

8.1 Summary

The present research work was started with the objective to enable computationally efficient nonlinear analysis of complex structural systems. At present, several issues including limited processing capability, memory bottlenecking and exceedingly complicated modelling process make such an analysis impractical. To overcome these issues, it was decided to use parallel computations where different partitions of the structure should be able to use different dimensional elements.

In order to achieve this objective, a review of the literature was carried out in which different partitioning methods were reviewed. In addition, different partitioning techniques and available tools were also reviewed. This literature review was summarised in Chapter 2 of this thesis. After this review of the literature, the domain decomposition method presented in Chapter 3 and its application to eigenvalue analysis presented in Chapter 4, was developed. A dimensional coupling approach based on the master-slave coupling technique was developed to be used with the domain decomposition. These methods were further enhanced to include hierarchic partitioning as presented in Chapter 6. Finally, some case studies were carried out as presented in Chapter 7 to demonstrate the applicability of the developed mix dimensional hierarchic partitioning methods.

During the development and implementation of the above mentioned methods, several technical challenges were faced that are presented below:

The first challenge was to determine a partitioning method to be used along with parallel computations. It was critical that the partitioning method should be such that it is portable

and scalable. The partitioning technique also had to be flexible so that it is not limited to any particular types of structures or particular types of analyses.

The prevailing partitioning methods are either limited to linear elastic analyses types for example the 'displacement frame method' or involve complicated pre-conditioners at the partitioned boundary for example domain decomposition methods based on Lagrange multipliers. The implementation of the latter type of methods requires extensive modifications to the finite element code.

To overcome these difficulties the dual partition super elements were developed. The domain decomposition method based on these elements is portable in the sense that it can be implemented with any finite element software. It is also scalable and inherently hierarchic; both these properties of the developed method have been demonstrated with the help of case studies.

The domain decomposition methods presented in this study are applicable to all kinds of finite element analyses as demonstrated. It is the property of the dual partition super elements that the parent structure is a complete structural model without any limit on the number of partitions or hierarchic levels. Using the load control technique for such a model was straight forward as the load factor was determined at the parent level and communicated down the partitioned hierarchy. Similarly the displacement as well as arc length control, as discussed in Section 3.2.4, were also used successfully and demonstrated with various examples throughout the thesis.

Once the dual partition super elements and a domain decomposition method based on these elements were developed, the next challenge was to implement these for the purpose of parallel computations. For this purpose Message Passing Interface (MPI) library subroutines were chosen because of their portability. MPI can be used for both shared memory and distributed memory systems and can be implemented with Fortran as well as C language. MPI library offers a wide variety of inter-processor communications including point to point, collective; blocking and non-blocking communications. For the most effective implementation of the developed methods both point to point and collective communications were used. The communication between processors for hierarchic

partitioning was another challenge as it was possible that different processors communicate through the same channel may cause bandwidth overlapping issues. To overcome this problem a novel procedure for the generation of additional communication channels was developed which has been presented in Section 6.4.2.

For the use of different dimensions in different partitions it was necessary to use dimensional coupling at the partitioned boundary. For this purpose an interface element was needed. There were several options i.e. the interface element could be placed on the parent structure side or it could be placed on the partition side as has been done. The former approach would have increased the inter processor communications volume as well as the size of the non-parallelisable portion of the program.

The interface element for dimensional coupling not only has different number of nodes on its two ends i.e. a single master node on one side and any number of slave nodes on the cross-section on the other side but also these nodes have different number of degrees of freedom. The single master node has 6 degrees of freedom whereas the slave nodes have 3 degrees of freedom on the other side. The transformation translational displacement between these two element ends was straight forward, however, the transformation of the rotational displacements and the tangent stiffness matrix posed significant challenge due to the non-commutativity of finite rotations about fixed axes and the dual issue of nonconservative moments about fixed axes.

Despite these challenges and difficulties, the objectives of this work have been successfully achieved as described in the following section.

8.2 Conclusions

The objectives of this research have been achieved by the development of a novel mixed dimensional hierarchic partitioned modelling method for the analysis of nonlinear structural systems. A new domain decomposition approach has been proposed for hierarchic partitioned modelling of structures. This approach enables partitions at various levels to use elements of different dimension from those at other levels. As a result, the computational time required for the analysis of complex structural systems subjected to extreme loading is

greatly reduced. In addition, the novel hierarchic partitioning method is shown to offer great ease in the modelling process through the use of modular modelling.

8.2.1 Reduction in Computational Time

The reduction in computational time is achieved in three ways, i) by domain partitioning, ii) by dimensional coupling, and iii) by hierarchic modelling, as elaborated in the following subsections. It is emphasised that the reduction in computational time in these three manners has been achieved without compromising the accuracy of analysis. This fact has been demonstrated by the use of examples throughout Chapters 3 to 6.

8.2.1.1 Domain Partitioning

By partitioning the structural domain and processing the partitions in parallel, the computational time has been greatly reduced, as shown in Sections 3.5.4-3.5.6. In Section 3.5.4, the I-beam with partitioned modelling using 2 partitions was analysed in about 46% of the time taken by monolithic analysis. In Section 3.5.5, it was shown that extensive partitioning with 16 partitions can continue to reduce the computing time for computationally demanding partitions, where the analysis time was reduced to about 13% of that required for monolithic analysis. Similarly, it was shown in Section 3.5.6 that the analysis time taken by the monolithic analysis is about 12 times more than that taken by partitioned analysis running on 12 parallel processors.

In the above mentioned examples, it was concluded that when the computational requirement inside the elements is significantly higher compared to that at the global structural level, more partitions typically result in further reduction in computational time. However, when the computational requirement of the elements is trivial compared to that at the structure level, the efficiency of partitioned modelling is governed by the size of the parent structure, which determined an optimum number of partitions.

Similar computational efficiency was achieved with the parallelisation of eigenvalue analysis using the domain partitioning method, as illustrated in Section 4.4.5.

8.2.1.2 Dimensional Coupling

Through the use of dimensional coupling, further reduction in analysis time has been recorded. In Section 5.5.1, the computational time of the I-beam with dimensional coupling between 1D and 3D element partitions has resulted in computational savings of 85% compared to partitioned modelling using only 3D brick elements. The gain in efficiency due to mixed-dimensional coupling has been further elaborated in Section 5.5.3, where only 3D elements are used. In this case, dimensional coupling is used only to reduce the size of the parent structure, which doubled the computational efficiency compared to the case without dimensional coupling at no significant loss in accuracy.

8.2.1.3 Hierarchic Modelling

The use of hierarchic modelling has further reduced computational demand, since in addition to reducing the size of the parent structure, hierarchic modelling also parallelises the communication between partitions. As shown in Section 6.5.2, the computational time for a slab with 16 partitions ran at an average of 300 seconds with hierarchic partitioning compared to an average of about 800 seconds when single level partitioning was used.

8.2.2 Modelling of Complex and Large Scale Structures

Whilst the proposed domain partitioning method has greatly reduced computational demand, this method has also created new opportunities for modelling real structures. In addition to the mixed dimensional and hierarchic features of the new method, parallelisation using distributed memory has overcome the memory bottleneck previously faced with large models, and has allowed previously intractable nonlinear analysis problems to be considered. Examples of such problems have been provided in Chapter 7, such as the brick masonry wall with openings and the cable stayed bridge with detailed modelling of the deck slab.

8.2.3 Ease of Modelling

The proposed hierarchic domain partitioning method has made the modelling process of complex structural systems relatively straightforward through the use of modular modelling.

In this context, partitioned models for similarly substructures/components can be developed and incorporated within the parent structure in a manner similar to conventional elements through coordinate transformations including translation and rotation. This capability has been demonstrated in the example of Section 3.5.6 and in the two case studies of Sections 7.2 and 7.4.

8.3 Recommendations for Future Research

The mixed-dimensional hierarchic partitioned method developed in this study has been successfully implemented within ADAPTIC [Izzuddin, 1991], allowing the nonlinear analysis of structural systems to be undertaken with great computational efficiency, and overcoming the memory bottleneck which previously prohibited the consideration of large models of real structures. While the original objectives of this research have been largely achieved, there are several fronts on which future research may be pursued towards further enhancements, as outlined hereafter.

8.3.1 Extended Dimensional Coupling Schemes

The dimensional coupling scheme developed and presented in this study considers 1D-3D element coupling and is based on a master-slave approach. Although, this kind of dimensional coupling approach is sufficient for many types of problem, e.g. frames, there are other types of problem for which the assumed relative rigidity of the slave nodes can compromise the actual solution. An alternative dimensional coupling scheme for 1D-3D elements based on Lagrange multipliers which impose average best-fit constraints on the 3D nodes is a promising direction for future research.

Further research on mixed dimensional coupling could also consider 1D-2D and 2D-3D element coupling, which can be achieved in the first instance with a master-slave approach and later enhanced with the Lagrange multiplier approach.

8.3.2 Multi-physics Modelling

Another research direction that could build on the achievement of this research relates to the modelling of multi-physics problems. For this purpose, dual partition super elements

may be used across different types of physical domains, where for example the parent partition could be a structure, while the child partition could be a soil or fluid subdomain.

8.3.3 MPI/OpenMP Hybrid Parallelisation

The domain decomposition and hierarchic partitioning methods developed in this work have been implemented with MPI for parallelisation. This provides them with the portability and flexibility of running on shared as well as distributed memory systems. This approach, however, does not utilise the additional advantage in case of shared memory systems.

In order to utilise the advantage of shared memory in a multi-threaded environment, work can be carried out on the MPI/OpenMP hybrid parallelisation as was discussed in Chapter 2. In this scheme, the hierarchic parallelisation of the current methods may be enhanced to be mapped according to the machine hierarchy.

8.3.4 Partition Scheduling

Whilst the hierarchic partitioning methods developed in this work do not put any limit on the total number of partitions that can be done, it may not always be physically possible to have as much number of processors available as there are the partitions in the analysis problem. In such situations, more than one partition can be assigned to a single processor as was demonstrated with the help of case study 4 in Chapter 7. This set up, while still more efficient than the monolithic case, may be subjected to latency costs where the frontal solution may have to wait for the next partition in the ordering list to return its values. The efficiency of such a system can be significantly increased if a technique is developed for scheduling of partitions in such a way that the partitions that occur earlier in the frontal ordering list in their parent structure are analysed first. This kind of partition scheduling can potentially achieve the same level of efficiency with less number of processors in cases that have large number of child partitions.

8.3.5 Heterogeneous Solution Procedures

As was discussed in Chapter 1, the developed methods can potentially allow the use of different solution procedures with each partition, including different time-integration

schemes, iterative techniques and convergence criteria. Due to the limitations of time and scope, such heterogeneous solution procedures were not implemented in the current work. It is suggested that the implementation of the same can be a viable future research direction.

References

- Amdahl, G. (1967). Validity of the single processor approach to achieving large-scale computing capabilities. *AFIPS Conference Proceedings* No. 30. P. 483-485.
- Balle, S., & Cullum, J., (1999). A parallel algorithm for computing eigenvalues of very large real symmetric matrices on message passing architectures. *Applied Numerical Mathematics*. Vol 30. P. 341-365.
- Belytschko, T., Hsieh, B.J. (1973). Non-linear transient finite element analysis with convected co-ordinates. *International Journal for Numerical Methods in Engineering*. Vol. 7. P.255-271.
- Belytschko, T., Yen, H.J., & Mullen, R. (1979). Mixed methods for time integration. *Computer Methods in Applied Mechanics and Engineering*. Vol. 17/18. P. 259-275.
- Brown, E.H., (1967). *Structural Analysis*. Volume 1. Longman Publishers London, United Kingdom.
- Chaboche, J.L., Feyel, F., & Monerie, Y., (2001). Interface debonding models: a viscous regularization with a limited rate dependency. *International Journal of Solids and Structures*. Vol. 38. P. 3127-3160.
- Chen, S.H., Lian, H.D., & Yang, X.W., (2003). Interval eigenvalue analysis for structures with interval parameters. *Finite Elements in Analysis and Design*. Vol. 39. P. 419-431.
- Cook, R.D., Malkus, D.S., Plesha, M.E., & Witt, R.J., (2002). *Concepts and Applications of Finite Element Analysis*. 4th Edition. John Wiley & Sons, Inc.
- Cristfield, M.A., (1991). *Non-linear Finite Element Analysis of Solids and Structures. Volume 1: Essential*. Wiley: Chichester, U.K.
- Cristfield, M.A., Moita G.F. (1996). A unified co-rotational framework for solids, shells and beams. *International Journal of Solids and Structures*. Vol. 33. P. 2969-2992.
- Demmel, J.W., Dhillon, I., & Ren, H., (1995). On the correctness of some bisection-like parallel eigenvalue algorithms in floating point arithmetic. *Electronic Transactions on Numerical Analysis*. Vol. 3. P. 116-149.

- Deuflhard, P., Leinen, P., Yserentant, H., (1989). Concepts of an adaptive hierarchical finite element code. *Impact of Computing in Science and Engineering*. Vol. 1. Issue 1. P. 3-35.
- Duff, I.S. (1996). A Review of Frontal Methods for Solving Linear Systems. *Computer Physics Communications*. Vol. 97. Issues 1-2. P. 45-52.
- Farhat, Ch., & 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.
- 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.
- 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.
- Fu, F., Lam, D., & Ye, J., (2007). Parametric study of semi-rigid composite connections with 3-D finite element approach. *Engineering Structures*. Vol. 29. P. 888-898.
- Gimenez, D., Hernandez, V., & Vidal, A.M., (2006). *Efficient Jacobi algorithms on multicomputers*. Springer eBook. Springer Berlin/Heidelberg.
- Gotze, J., Paul, S., & Sauer, M., (1993). An efficient Jacobi-line algorithm for parallel eigenvalue computation. *IEEE Transactions on Computers*. Vol. 42. No. 9. P. 1058-1065.
- Granat, R., Kagstroem, B., & Kressner, D., (2009). *A novel parallel QR algorithm for hybrid distributed memory HPC systems*. Research Report No. 2009-15. Swiss Federal Institute of Technology. Zurich.
- Hellweg, H.B. & Crisfield, M.A. (1998). A new arc-length method for handling sharp snap-backs. *Computers & Structures*. Vol. 66. P. 705-709.
- Hendrickson, B., (1998). Graph partitioning and parallelsolvers: Has the emperor no clothes? In: Solving Irregularly Structured Problems in Parallel, Irregular '98, *Lecture Notes in Computer Science*, Vol. 1457, Springer, Berlin. P. 218-225.
- Hendrickson, B., (2000). Load balancing fictions, falsehoods and fallacies. *Applied Mathematical Modelling*. Vol. 25. P. 99-108.

- Horst, S.D., (1984). The Lanczos Algorithm with Partial Reorthogonalization. *Mathematics of Computation*. Vol. 42. No. 165. P. 115-142.
- Huebner, K.H., Dewhirst, D.L., Smith, D.E., & Byrom, T.G., (2001). *The Finite Element Method for Engineers*. 4th Edition. John Wiley & Sons Inc.
- Hughes, T.J.R., (2000). *The Finite Element Method: Linear Static and Dynamic Analysis*. New York: Dover Publications.
- Hughes, T.J.R., Levit, I., & Winget, J. (1983). An element-by-element solution algorithm for problems of structural and solid mechanics. *Computer Methods in Applied Mechanics and Engineering*. Vol. 36. P. 241-254.
- Ibrahimbegovic, A. (1995). On finite element implementation of geometrically nonlinear Reissner's beam theory: Three-dimensional curved beam elements. *Computer Methods in Applied Mechanical Engineering*. Vol. 122. P. 11-26.
- Irons, B.M. (1970). A Frontal Solution Program for Finite Element Analysis. *International Journal for Numerical Methods in Engineering*. Vol. 2. No. 1. P. 5-32.
- Izzuddin, B.A., (1991). *Nonlinear Dynamic Analysis of Framed Structures*. A thesis submitted for the degree of Doctor of Philosophy in the Faculty of Engineering of the University of London. London: Imperial College of Science and Technology.
- Izzuddin, B.A., & Elnashai, A.S., (1993). Eulerian formulation for large-displacement analysis of space frames. *Journal of Engineering Mechanics*. ASCE. No. 117. P. 549-569.
- Izzuddin, B.A., & Lloyd Smith, D., (2000). Effective nonlinear analysis of elasto-plastic three dimensional reinforced concrete frames using adaptive techniques. *Computers & Structures*. Vol. 78. No. 4. P. 549-573.
- Izzuddin, B.A., (2001). Conceptual issues in geometrically nonlinear analysis of 3D framed structures. *Computer Methods in Applied Mechanics and Engineering*. Vol. 191. Issues 8-10. P. 1029-1053.
- Izzuddin, B.A., (2003). Integration of Beam-Column and Shell Elements in Large Displacement Structural Analysis, *Proceedings of the International Symposium on New Perspectives for Shell and Spatial Structures*. IASS-APCS. Taipei, Taiwan.

- 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.
- Izzuddin, B.A. (2009). *ADAPTIC Manual*. Revision 1.3b. Systems and Mechanics Section. Department of Civil and Environmental Engineering. Imperial College London.
- Izzuddin, B.A., and Elnashai, A.S., (1993). Adaptive Space Frame Analysis- Part II: A Distributed Plasticity Approach. *Proceedings of the Institution of Civil Engineers, Structures and Bridges*, Vol. 99. P. 317-326.
- Izzuddin, B.A., & Macorini, L., (2011). Multiscale modelling of masonry structures under extreme loading, *Proceedings of the XI-International Conference on Computational Plasticity. Fundamentals and Applications (COMPLAS-XI)*. Onate & Owen. Eds.
- Izzuddin, B.A., Tao, X.Y., and Elghazouli, A.Y., (2004). Realistic Modelling of Composite and R/C Floor Slabs under Extreme Loading – Part I: Analytical Method. *Journal of Structural Engineering*. ASCE. Vol. 130. No. 12. P. 1972-1984.
- Jelenic G., Crisfield, M.A. (1999). Geometrically exact 3D beam theory: Implementation of a strain-invariant finite element for static and dynamics. *Computer Methods in Applied Mechanical Engineering*. Vol. 171. P. 141-171.
- Kapania, R.K., & Li, J. (2003). On a geometrically exact curved/twisted beam theory under rigid cross-section assumption. *Computer Methods*. Vol. 30. P. 428-443.
- Karpenko, S.N. (1989). Modified Computational Scheme of the Frontal Method. *Strength of Materials*. Vol. 21. Number 7. P. 942-948.
- Katagiri, T., & Kanada, Y., (2001). An efficient implementation of parallel eigenvalue computation for massively parallel processing. *Parallel Computing*. Vol. 27. P. 1831-1845.
- Kron, G. (1939). *Tensor Analysis of Networks*. John Wiley & Sons. New York.
- Kruis, J. (2007). The FETI-DP Method. In: Magoules, F. (ed). *Mesh Partitioning Techniques and Domain Decomposition Methods*. Saxe-Coburg Publications.

- Lanczos, C., (1950). An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bur. Standards*. Vol. 45. P. 255-282.
- Lee, S.S., & Moon, T.S., (2007). Moment-rotation model of semi-rigid connections with angles. *Engineering Structures*. Vol. 24. P. 227-237.
- Li, X., & Parashar, M., (2004). Hierarchical Partitioning Techniques for Structured Adaptive Mesh Refinement Applications. *The Journal of Supercomputing*. Vol. 28. No. 3. P. 265-278.
- Lin, C., & Snyder, L. (2008). *Principles of Parallel Programming*. Pearson/Addison-Wesley.
- Macorini, L., & Izzuddin, B.A., (2010). A non-linear interface element for 3D mesoscale analysis of brick-masonry structures. *International Journal for Numerical Methods in Engineering*. Vol. 85. P. 1584-1608.
- Macorini, L., & Izzuddin, B.A., (2011). Nonlinear analysis of masonry structures using mesoscale partitioned modelling, PARENG2011, Ajaccio, France.
- 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.
- Markovic, D., & Ibrahimbegovic, A., (2003). On micro-macro interface conditions for micro scale based FEM for inelastic behaviour of heterogeneous materials, *Computer Methods in Applied Mechanics*. Vol. 193. P. 5503-5523.
- Massart, T.J., Peerlings, R.H.J. & Geers, M.G.D., (2007). An enhanced multi-scale approach for masonry wall computations with localizations of damage, *International Journal of Numerical Methods in Engineering*, Vol. 69. P. 1022-1059.
- Mata, P., Barbat, A.H., & Oller, S., (2009). Two-scale approach for the non-linear dynamic analysis of RC structures with local non-prismatic parts. *Engineering Structures*. Vol. 30. P. 3667-3680.
- McCune, R.W., Armstrong, C.G., & Robinson, D.J., (2000). Mixed-dimension coupling in finite element models. *International Journal for Numerical Methods in Engineering*. Vol. 49. P. 725-750.

- Mercatoris, B.C.N., & Massart, T.J., (2011). A coupled two-scale computation scheme for the failure of periodic quasi-brittle thin planar shells and its application to masonry. *International Journal of Numerical Methods in Engineering*. Vol. 85. P. 1177-1206.
- Message Passing Interface Forum (2009). *MPI: A Message Passing Interface Standard*. Version 2.2. [Online]. September 2009. Available From: <http://www.mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf> [Accessed: 31 May 2011].
- Morgan, R.B., (1996). On restarting the Arnoldi method for large non-symmetric eigenvalue problems. *Mathematical Computations*. Vol. 65. 1213-1230.
- Netlib, (2011). *LAPACK – Linear Algebra PACKage*. [Online]. Available from: <http://www.netlib.org/lapack/> [Accessed: 6 June, 2011].
- Newmark, N.M. (1959). A method of computation for structural dynamics. *Journal of the Engineering Mechanics Division, ASCE*. Vol. 85(EM3). P. 67-94.
- OpenMP Architecture Review Board (2011). *OpenMP Application Program Interface*. Version 3.1. <http://www.openmp.org/mp-documents/OpenMP3.1.pdf> [Accessed: August 7, 2012]
- Paige, C.C., (1971). *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices*. Thesis submitted for the degree of Ph.D. University of London.
- Paige, C.C., (1972). Computational variants of the Lanczos method for the eigenproblem. *J. Inst. Math. Appl.* Vol. 10. P. 373-381.
- Paige, C.C., (1976). Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix. *J. Inst. Math. Appl.* Vol. 18. P. 341-349.
- Parlett, B.N. (1998). *The Symmetric Eigenvalue Problem*. Classics in Applied Mathematics 20. Society for Industrial and Applied Mathematics. 3600 University City Science Centre, Philadelphia. PA. 19104-2688.
- Parlett, B.N., & Scott, D.S., (1979). The Lanczos Algorithm with Selective Orthogonalization. *Mathematics of Computation*. Vol. 33. No. 145. P. 217-238.
- Platonov, A.D. & Trivailo, P.M. (1979). An Algorithm for Excluding Unknowns in the Frontal Method. *Strength of Materials*. Vol. 11. No. 8. P. 841-842.

- Quarteroni, A., & Valli, A., (1999). *Domain decomposition methods for partial differential equations*, Oxford, Clarendon press.
- Reissner, E., (1973). On one-dimensional large-displacement finite-strain beam theory. *Studies in Applied Maths*. Vol. 11. P. 87-95.
- Riks, E. (1979). An incremental approach to the solution of snapping and buckling problems, *International Journal of Solids and Structures*. Vol. 15. P. 529-551.
- Royo, D., Gonzalez, A., & Valero-Garcia, M., (1999). Low communication overhead Jacobi algorithms for eigenvalues computation on hypercubes. *The Journal of Supercomputing*. Vol. 14. P. 171-193.
- Saad, Y., & Malevsky, A., (1995). *A portable library of distributed memory sparse iterative solvers*. Report umsi-95-180. Minnesota Supercomputer Institute.
- Schmidt, J.A., (2009). The definition of structural engineering. *Structure*. A joint publication of NCSEA, CASE and SEI. www.structuremag.org [Accessed 16/01/12].
- Shim, K.W., Monaghan, D.J., & Armstrong C.G., (2002). Mixed dimension coupling in finite element stress analysis. *Engineering with Computers*. Vol. 18. P. 241-252.
- Simo, J.C., (1985). A finite strain beam formulation. The three-dimensional dynamic problem. Part I. *Computational Methods in Applied Mechanics and Engineering*. Vol. 49. P. 55-70.
- 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.
- Smith, B., Bjorstad, P., & Gropp, W. (1996). *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press.
- Spacone, E., & El-Tawil, S., (2004). Nonlinear analysis of steel-concrete composite structures: State of the art. *Journal of Structural Engineering, ASCE*. Vol. 126. No. 2. P. 159-168.

- Suda, R., Nishida, A., & Oyanagi, Y., (1999). A high performance parallelization scheme for the Hessenberg double shift QR algorithm. *Parallel Computing*. Vol. 25. P. 729-744.
- Szabo, B., & Babuska, I., (1991). *Finite Element Analysis*. John Wiley & Sons Inc.
- Taylor, R.L. & De Bellis, M.L., (2011). Multi-scale analysis of masonry structures, CMM 2011 – Computer Methods in Mechanics, Poland.
- Teresco, J.D., Beall, M.W., Flaherty, J.E., and Shephard, M.S., (2000). A hierarchical partition model for adaptive finite element computation. *Computer Methods in Applied Mechanics and Engineering*. Vo. 184. P. 269-285.
- Teresco, J.D., Faik, J., and Flaherty, J.E., (2006). Hierarchical Partitioning and Dynamic Load Balancing for Scientific Computation. In: Applied Parallel Computing, State of the Art in Scientific Computing. *Lecture Notes in Computer Science*. Vol. 3732/2006. P. 911-920.
- Toselli, A., & Widlund, O. (2005). *Domain Decomposition Methods – Algorithms and Theory*. Springer Series in Computational Mechanics No. 34, Springer-Verlag Berlin Heidelberg.
- von Bertalanffy, L., (1968). *General System Theory: Foundations, Developments, Applications*. New York: Braziller.
- Walonick, D.S., (1993). *General System Theory*. <http://www.statpac.org/walonick/systems-theory.htm> [Accessed 16/01/12].
- Watkins, D.S., (1982). Understanding QR Algorithm. *SIAM Review*. Vol. 24. No. 4. P. 427-440.
- 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.
- Yue, J., Fafitis, A., & Qian, J., (2010). On the kinematic coupling of 1D and 3D finite elements: a structural model. *Interaction and Multiscale Mechanics*. Vol. 3. No. 2. P. 193-212.
- 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. (to be submitted)

- Zainal-Abidin, A.R. and Izzuddin, B.A., (2011). Buckling Analysis of Cellular Beams using the Rotational Spring Analogy, *Proceedings of the 19th International Conference on Computer Methods in Mechanics*, Warsaw.
- Zienkiewicz, O.C., Taylor, R.L., & Zhu, J.Z., (2005a). *The Finite Element Method: Its Basis and Fundamentals*. The Finite Element Method Set Volume 1. 6th Edition. Elsevier Ltd.
- Zienkiewicz, O.C., Taylor, R.L., & Zhu, J.Z., (2005b). *The Finite Element Method in Solid and Structural Mechanics*. The Finite Element Method Set Volume 2. 6th Edition. Elsevier Ltd.
- Zolghadr Jahromi, H., (2009). *Partitioned Analysis of Nonlinear Soil-Structure Interaction*. PhD Thesis. Imperial College London.
- 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.
- 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.

Appendix A

Input files for the example presented in Section 3.5.1 are presented as below. These files illustrate the single level partitioning of a simple structure without dimensional coupling.

Parent Structure

```
analysis 2d statics

materials
  mat.name      model properties
  mat1          stl1  0.720e3  0.100e1  0.00

groups
type.of.element = part
  grp.name      nodes
  grpp1         1
  grpp2         2

coordinates.transformation
  translation.vector
  0.0  120.0

structural
  nod.n      x      y
  1          0.00  0.00
  2          24.00  0.00

element.connectivity
  elm.name    grp.name    nod.name
  1           grpp1 1
  2           grpp2      1  2
  3           grpp1  2

applied.loading
  proportional.loads
    nod.name    direction    type    value
    2           y           force  -0.10e+1

condition
  lf.cnd.name    limits
  1             -2.0 2.0
  disp.cnd.name  nod.name    direction  limits
  2             2           x          -0.12e+3 0.12e+3
  3             2           y          -0.12e+3 0.12e+3

phases
load.control
  increment      path  steps
  0.2e+1      k      20
automatic.control
```

```

      type              path  cnd.name
      nodal      translation      c      1 2 3
iterative.strategy
number = 5
initial.reformations = 5
step.reduction = 5
divergence.iteration = 4
maximum.convergence = 0.1e3

convergence.criteria
  tolerance = 0.1e-5
  force = 0.2e+1
  mome = 0.1e+3

output
stress

end

```

Partition No. 1

```

analysis 2d statics

materials
  mat.name      model properties
  mat1          stl1  0.720e3  0.100e1 0.00

sections
  type = rss
  sec.name      mat.name      dimensions
  sect1         mat1          3.0 2.0

groups
type.of.element = qel2
grp.name  sec.name
  grp1      sect1

structural
  nod.n      x      y
  1          0.00   0.00
  2          0.00  120.00
  3          0.00   60.0

restraints
  nod.name direction
  1      x+y

element.connectivity
  elm.name  grp.name  nod.
  1      grp1      1 3
  2      grp1      3 2

```

```
partitioned.boundary
  nod.name
  2

applied.loading
  proportional.loads
    nod.name  direction  type  value
      3         y      force -0.0

iterative.strategy
number = 5
initial.reformations = 5
step.reduction = 5
divergence.iteration = 4
maximum.convergence = 0.1e3

convergence.criteria
  tolerance = 0.1e-5
  force = 0.2e+1
  mome = 0.1e+3

output
frequency 0

end
```

Partition No. 2

```
analysis 2d statics

materials
  mat.name      model properties
  mat1          stl1  0.720e3  0.100e1 0.00

sections
  type = rss
  sec.name  mat.name  dimensions
  sect1     mat1      3.0 2.0

groups
type.of.element = qel2
grp.name  sec.name
  grp1      sect1

coordinates.transformation
  translation.vector
    0.0  120.0

structural
  nod.n    x    y
    1      0.00 0.00
```

```

      2      24.00  0.00
      3      12.00  0.00

element.connectivity
  elm.name   grp.name   nod.
      1      grp1      1  3
      2      grp1  3  2

partitioned.boundary
nod.name
  1
  2

applied.loading
proportional.loads
  nod.name   direction   type   value
      3             y     force  -0.0

iterative.strategy
number = 5
initial.reformations = 5
step.reduction = 5
divergence.iteration = 4
maximum.convergence = 0.1e3

convergence.criteria
  tolerance = 0.1e-5
  force = 0.2e+1
  mome = 0.1e+3

output
frequency 0

end

```

Partition No. 3

```

analysis 2d statics

materials
  mat.name   model properties
  mat1      stl1  0.720e3  0.100e1  0.00

sections
  type = rss
  sec.name   mat.name   dimensions
  sect1      mat1      3.0 2.0

groups
type.of.element = qel2
grp.name   sec.name

```

```

      grp1          sect1

coordinates.transformation
translation.vector
    24.0 120.0

structural
  nod.n      x      y
    1        0.00    0.00
    2       96.00    0.00
    3       48.00    0.00

restraints
  nod.name direction
    2      x+y

element.connectivity
  elm.name  grp.name  nod.
    1      grp1      1  3
    2      grp1      3  2
partitioned.boundary
  nod.name
    1

applied.loading
  proportional.loads
    nod.name  direction  type  value
        3          y      force -0.0

iterative.strategy
number = 5
initial.reformations = 5
step.reduction = 5
divergence.iteration = 4
maximum.convergence = 0.1e3

convergence.criteria
  tolerance = 0.1e-5
  force = 0.2e+1
  mome = 0.1e+3

output
frequency 0

end

```

Appendix B

Input files for the case 4 of example presented in Section 5.5.1 are presented as below. These files illustrate the single level partitioning of a cantilever beam using dimensional coupling.

Parent Structure

```
#
analysis 3d statics
#
materials
  mat.name model properties
  mat1          st11  2.10E+05  2e9  0.3
#
sections
  type = isec
  mat.name = mat1
  sec.name  dimensions
  sec1      120   10   120   10   130   10
#
groups
  type = cbp3
  grp.name      sec.name  monitoring.points
  gp1      sec1          99
  type = part
  grp.name nodes
  gpp      1
#
coordinates.transformation
  translation.vector
    500.0 75.0 60.0
#
structural.nodal
  nod.name  x    y    z
  f    1    0.0  0.0  0.0
  r    1          500.0 0.0  0.0  5
#
non.structural.nodal
  nod.name  x    y    z
  f    11    0.0  500.0 0.0
  r    1    500.0 0    0.0  4
#
element.connectivity
  elm.name  grp.name  nod.name
  f  1          gp1    1    2    11
  r  1          -      1    1    1    4
#
    11    gpp    1
#
```

```

#restraints
# nod.name
# f 1
# r 1 20
# r 200 2
# r 2400 1
# f 101
# r 2 10
# r 200 1
# r 2400 1
# f 509
# r 2 2
#r 200 9
#f 609
# r 1 4
# r 200 8
#
#
linear.curves
start.time = 0
crv.name = c1
time load.factor
1 1
#
applied.loading
initial.loads
nod.name direction type value
f 1 y f -178 #-75
r 1 - - 0.0 5
##
time.history.loads
nod.name direction type crv.name value
6 y f c1 -1.78e7 #-1e6
#r 200 - - - 0.0 2
#r 2400 - - - 0.0 1
#f 1000101 y f c1 -1e6
#r 2 - - - 0.0 10
#r 200 - - - 0.0 1
#r 2400 - - - 0.0 1
#f 1000509 y f c1 -1e6
#r 2 - - - 0.0 2
#r 200 - - - 0.0 9
#f 1000609 y f c1 -1e6
#r 1 - - - 0.0 4
#r 200 - - - 0.0 8
#
equilibrium.stages
end.of.stage steps
0.3 100
0.5 100
0.8 200
1.0 200
#
iterative.strategy
number = 10
initial.reformations = 10

```

```

step.reduction = 10
divergence.iteration = 8
maximum.convergence = 1e7
#
convergence.criteria
  tolerance = 0.1e-4
  force.ref = 0.1e9
  moment.ref = 0.1e8
#
output
  frequency 0 stress
#
end

```

Child Partition

```

#
analysis 3D.Continuum Statics
#
materials
  mat.name model properties
  mat1          beth 2.10E+05 0.3 0.1E-5
#
groups
  type = bk20
  grp.name mat.name gauss.points density damping.parameters
  gp1 mat1 27 1.9E-9 20 0
  type = mslv
  grp.name slave.nodes
  gpms 178
#
coordinates.transformation
  translation.vector
    0 0 0
#
structural.nodal
  nod.name x y z
  f 1 0.0 0.0 0.0
  r 2 0.0 0.0 11.0 5
  r 12 0.0 0.0 65.0 1
  r 200 0.0 10.0 0.0 1
  r 2800 0.0 140.0 0.0 1
  r 10000 50.0 0.0 0.0 10
#
  f 2 0.0 0.0 5.5
  r 2 0.0 0.0 11.0 4
  r 12 0.0 0.0 65.0 1
  r 200 0.0 10.0 0.0 1
  r 2800 0.0 140.0 0.0 1
  r 20000 100.0 0.0 0.0 5
#

```

```

f      12      0.0    0.0    60.0
r      200     0.0    10.0   0.0    15
r      20000  100.0   0.0    0.0    5
#
f      101     0.0    5.0    0.0
r       2      0.0    0.0    11.0   5
r      12      0.0    0.0    65.0   1
r      2800    0.0   140.0   0.0    1
r      20000  100.0   0.0    0.0    5
#
f      311     0.0   15.0   55.0
r       2      0.0    0.0   10.0   1
r      200     0.0   10.0   0.0   12
r      20000  100.0   0.0    0.0    5
#
f      411     0.0   20.0   55.0
r       2      0.0    0.0   10.0   1
r      200     0.0   10.0   0.0   11
r      10000  50.0   0.0    0.0   10
#
non.structural.nodal
nod.name  x      y      z
25        500.0  75.0   60.0
#
element.connectivity
elm.name  grp.name  nod.name
f      1      gp1    3      20003 20001 1      &
                203    20203 20201 201    &
                10003 20002 10001 2      &
                10203 20202 10201 202    &
                103    20103 20101 101
r      1      -      2      2      2      2      &
                2      2      2      2      &
                2      2      2      2      &
                2      2      2      2      &
                2      2      2      2      10
r      200    -      2800 2800 2800 2800 &
                2800 2800 2800 2800 &
                2800 2800 2800 2800 &
                2800 2800 2800 2800 &
                2800 2800 2800 2800 1
r6:6 10      -      200   200   200   200 &
                200   200   200   200 &
                200   200   200   200 &
                200   200   200   200 &
                200   200   200   200 13
r      1000   -      20000 20000 20000 20000 &
                20000 20000 20000 20000 &
                20000 20000 20000 20000 &
                20000 20000 20000 20000 &
                20000 20000 20000 20000 4
#
12      gpms  100001      100002      100003      100004      100005
                100006      &
                100007      100008      100009      100010      100011
                100012      &

```

100013	100014	100015	100016	100017
100018 &				
100019	100020	100021	100022	100023
100101 &				
100103	100105	100107	100109	100111
100113 &				
100115	100117	100119	100121	100123
100201 &				
100202	100203	100204	100205	100206
100207 &				
100208	100209	100210	100211	100212
100213 &				
100214	100215	100216	100217	100218
100219 &				
100220	100221	100222	100223	100311
100313 &				
100411	100412	100413	100511	100513
100611 &				
100612	100613	100711	100713	100811
100812 &				
100813	100911	100913	101011	101012
101013 &				
101111	101113	101211	101212	101213
101311 &				
101313	101411	101412	101413	101511
101513 &				
101611	101612	101613	101711	101713
101811 &				
101812	101813	101911	101913	102011
102012 &				
102013	102111	102113	102211	102212
102213 &				
102311	102313	102411	102412	102413
102511 &				
102513	102611	102612	102613	102711
102713 &				
102801	102802	102803	102804	102805
102806 &				
102807	102808	102809	102810	102811
102812 &				
102813	102814	102815	102816	102817
102818 &				
102819	102820	102821	102822	102823
102901 &				
102903	102905	102907	102909	102911
102913 &				
102915	102917	102919	102921	102923
103001 &				
103002	103003	103004	103005	103006
103007 &				
103008	103009	103010	103011	103012
103013 &				
103014	103015	103016	103017	103018
103019 &				
103020	103021	103022	103023	25

#

```

restraints
  direction = x+y+z
nod.name
f    1
r    1    22
r   200    1
r  2800    1
f    101
r    2    11
r  2800    1
f   311
r    2    1
r   200   12
f   411
r    1    2
r   200   11
#
partitioned.boundary
master.slave.elm.name
12
#
linear.curves
start.time = 0
crv.name = c1
time load.factor
1    1
#
applied.loading
initial.loads
  nod.name direction  type  value
  10001    y          f    -0.0 #-75
#
##
time.history.loads
  nod.name direction  type  crv.name  value
  40001    y          f    c1      -000 #-75
#
equilibrium.stages
end.of.stage  steps
0.3          100
0.5          100
0.8          200
1.0          200
#
iterative.strategy
number = 10
initial.reformations = 10
step.reduction = 10
divergence.iteration = 8
maximum.convergence = 1e7
#
convergence.criteria
tolerance = 0.1e-4
force.ref = 0.1e9
moment.ref = 0.1e8
#

```

```
output
  frequency 0 stress
#
end
```

Appendix C

Input files for 3 partitions of the example presented in Section 6.5.1 are presented as below. These files illustrate hierarchic partitioning of a frame structure. Each of the 3 partitions presented here represents a different level of partitioning. These files also illustrate the coupling of additional freedoms for element 'csl4' as well as the use of 'coordinates.transformation' module for the purpose of hierarchical partitioning.

Root Partition

```
#
analysis 2d dynamics
#
patterns
  pat.name      ratios
    p1          1 2 3 3 2 1
#
groups
#
type = part
  grp.name  nodes
  gpp2      2
#
coordinates.transformation
  translation.vector
  0.0  7.0
#
structural.nodal.coordinates
  nod.name      x      y
    1            0      0
    2            6      0
#
element.connectivity
  elm.name  grp.name  nod.name
  1  gpp2  1  2
  2  gpp2  1  2
#
linear.curves
  start.time = 0
  crv.name = c1
  file = earthquake1      # obtain the record from file "earthquake1"
  first.line = 1
  last.line = 1200
  format = (23x,2(e15.8,2x))
#
integration.scheme
  scheme = hilber
  alpha = -0.3
#
```

```

applied.loading
  initial.loads
    nod.name direction type value
      1      y      f  -49.050e4    # mass*g applied as initial
      2      y      f  -49.050e4    # gravity loads; modify as
necessary
#
equilibrium.stages
  end.of.stage  steps
      7          350      # time step 7/350=0.02; modify as
necessary
#
iterative.strategy
  number.of.iterations = 10
  initial.reformations = 10
  step.reduction = 10
  divergence.iteration = 10
  maximum.convergence = 0.1e8
#
convergence.criteria
  tolerance = 1e-5
  force.ref = 1e5
  moment.ref = 1e5
#
output
  frequency 1
#
end

```

Intermediate Level Partition (1)

```

analysis 2d dynamics
#
patterns
  pat.name      ratios
    p1          1 2 3 3 2 1
#
groups
#
type = part
  grp.name  nodes
  gpp2      2
  gpp4      4
#
coordinates.transformation
  translation.vector
    0.0 4.0
#
structural.nodal.coordinates
  nod.name      x      y
    1            0.004    0
    2            6.004    0

```

```

      3      0      3
      4      6      3
#
element.connectivity
  elm.name   grp.name   nod.name
  1   gpp2  1     2
  2   gpp4  1     2     3     4
#
partitioned.boundary
parent.rank = 0
nod.name
3
4
#
linear.curves
start.time = 0
crv.name = c1
file = earthquakel      # obtain the record from file "earthquakel"
first.line = 1
last.line = 1200
format = (23x,2(e15.8,2x))
#
integration.scheme
scheme = hilber
alpha = -0.3
#
applied.loading
initial.loads
  nod.name direction type value
    1       y      f  -49.050e4    # mass*g applied as initial
    2       y      f  -49.050e4    # gravity loads; modify as
necessary
#
equilibrium.stages
end.of.stage  steps
    7          350      # time step 7/350=0.02; modify as
necessary
#
iterative.strategy
number.of.iterations = 10
initial.reformations = 10
step.reduction = 10
divergence.iteration = 10
maximum.convergence = 0.1e8
#
convergence.criteria
tolerance = 1e-5
force.ref = 1e5
moment.ref = 1e5
#
output
  frequency  1
#
end

```

Terminal Partition (3)

```

analysis 2d dynamics
materials
  mat.name      model      properties
  mat1          stl1       210e9  300e6  0.01
#
sections
  type = rss
  mat.name = mat1
  sec.name      dimensions
  sec1          0.1  0.15
  sec2          0.20 0.20
  sec3          0.15 0.15
#
patterns
  pat.name      ratios
  p1            1 2 3 3 2 1
#
groups
type = cbp2
  grp.name      sec.name      monitoring.points
  grp1c         sec1          20
  grp2c         sec2          20
  grp3c         sec3          20
#
type = qdp2
  grp.name      cbp2.grp.name  pat.name
  grp1          grp1c         p1
  grp2          grp2c         p1
  grp3          grp3c         p1
#
type = cnm2
  grp.name      mass
  gpm1          5e4      # modify the masses
  gpm2          2.5e4    # as required
#
structural.nodal.coordinates
  nod.name      x      y
  1              0      0
  2              6      0
  3              0.004  4
  4              6.004  4
  5              3.0    4
#
restraints
  nod.name      direction
  1              y+rz      # note no x restraint
  2              y+rz      # to enable base earthquake excitation
#
element.connectivity
  elm.name      grp.name      nod.name
  1              grp1          3      5
  2              grp1          5      4
  11             grp2          1      3

```

```

12      grp2      2    4
51      gpm1      3
52      gpm1      4
#
partitioned.boundary
parent.rank = 1
nod.name
3
4
#
linear.curves
start.time = 0
crv.name = c1
file = earthquake1      # obtain the record from file "earthquake1"
first.line = 1
last.line = 1200
format = (23x,2(e15.8,2x))
#
integration.scheme
scheme = hilber
alpha = -0.3
#
applied.loading
initial.loads
nod.name direction type value
5      x      f      -0.0      # mass*g applied as initial
#
dynamic.loads
nod.name direction type crv.name value
1      x      a      c1      9.81      # record scaled by g
2      x      a      c1      9.81
#
equilibrium.stages
end.of.stage steps
7      350      # time step 7/350=0.02; modify as
necessary
#
iterative.strategy
number.of.iterations = 10
initial.reformations = 10
step.reduction = 10
divergence.iteration = 10
maximum.convergence = 0.1e8
#
convergence.criteria
tolerance = 1e-5
force.ref = 1e5
moment.ref = 1e5
#
output
frequency 1
#
end

```