UNSW SYDNEY

High-performance computing for impact-induced fracture analysis exploiting octree mesh patterns

Author:
Ankit, Ankit

Publication Date:
2021

DOI:
https://doi.org/10.26190/unsworks/22788

License:
https://creativecommons.org/licenses/by-nc-nd/3.0/au/
Link to license to see what you are allowed to do with this resource.

Downloaded from http://hdl.handle.net/1959.4/71188 in https://unsworks.unsw.edu.au on 2024-07-02

High-Performance Computing for Impact-Induced Fracture Analysis exploiting Octree Mesh Patterns

Ankit

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

School of Civil and Environmental Engineering

The University of New South Wales

Sydney, Australia

November, 2021

Thesis submission for the degree of Doctor of Philosophy

Thesis Title and Abstract	Declarations	Inclusion of Publications Statement	Corrected Thesis and Responses

Thesis Title

High-Performance Computing for Impact-Induced Fracture Analysis exploiting Octree Mesh Patterns

Thesis Abstract

The impact-induced fracture analysis has a wide range of engineering and defence applications, including aerospace, manufacturing, and construction. An accurate simulation of impact events often requires modelling large-scale complex geometries along with dynamic stress waves and damage propagation. To perform such simulations in a timely manner, a highly efficient and scalable computational framework is necessary.

This thesis aims to develop a high-performance computational framework for analysing large-scale structural problems pertaining to impact-induced fracture events. A hierarchical grid-based mesh containing octree cells is utilised for discretising the problem domain. The scaled boundary finite element method (SBFEM) is employed, which can efficiently handle the octree cells by eliminating the hanging node issues.

The octree-mesh is used in balanced form with a limited number of octree cell patterns. The master element matrices of each pattern are pre-computed while the storage of the individual element matrices is avoided leading to a significant reduction in memory requirements, especially for large-scale models. Further, the advantages of octree cells are leveraged by automatic mesh generation and local refinement process, which enables efficient pre-processing of models with complex geometries.

To handle the matrix operations associated with large-scale simulation, a pattern-by-pattern (PBP) approach is proposed. In this technique, the octree-patterns are exploited to recast a majority of the computational work into pattern-level dense matrix operations. This avoids global matrix assembly, allows better cache utilisation, and aids the associated memory-bandwidth limited computations, resulting in significant performance gains in matrix operations.

The PBP approach also supports large-scale parallelism. In this work, the parallel computation is carried out using the mesh-partitioning strategy and implemented using the message passing technique. It is shown that the developed solvers can simulate large-scale and complex structural problems, e.g. delamination/fracture in sandwich panels with approximately a billion unknowns (or DOFs). A massive scaling can be achieved with more than ten thousand cores in a distributed computing environment, which reduces the computation time from months (on a single core) to a few minutes.

Thesis submission for the degree of Doctor of Philosophy

Thesis Title and Abstract	Declarations	Inclusion of Publications Statement	Corrected Thesis and Responses

ORIGINALITY STATEMENT

☑ I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

COPYRIGHT STATEMENT

☑ I hereby grant the University of New South Wales or its agents a non-exclusive licence to archive and to make available (including to members of the public) my thesis or dissertation in whole or part in the University libraries in all forms of media, now or here after known. I acknowledge that I retain all intellectual property rights which subsist in my thesis or dissertation, such as copyright and patent rights, subject to applicable law. I also retain the right to use all or part of my thesis or dissertation in future works (such as articles or books).

For any substantial portions of copyright material used in this thesis, written permission for use has been obtained, or the copyright material is removed from the final public version of the thesis.

AUTHENTICITY STATEMENT

☑ I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis.

Thesis submission for the degree of Doctor of Philosophy

Thesis Title and Abstract	Declarations	Inclusion of Publications Statement	Corrected Thesis and Responses

UNSW is supportive of candidates publishing their research results during their candidature as detailed in the UNSW Thesis Examination Procedure.

Publications can be used in the candidate's thesis in lieu of a Chapter provided:

- The candidate contributed **greater than 50%** of the content in the publication and are the "primary author", i.e. they were responsible primarily for the planning, execution and preparation of the work for publication.
- The candidate has obtained approval to include the publication in their thesis in lieu of a Chapter from their Supervisor and Postgraduate Coordinator.
- The publication is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in the thesis.

☑ The candidate has declared that **some of the work described in their thesis has been published and has been documented in the relevant Chapters with acknowledgement.**

A short statement on where this work appears in the thesis and how this work is acknowledged within chapter/s:

> The contents of Chapters 4 and 6 of the thesis were published in journals. The acknowledgement of the work of other authors has been made at the beginning of these chapters (see footnote) as well as in the Acknowledgement section. Additionally, Section 1.8 of the thesis also provides a list of publications.

Candidate's Declaration

I declare that I have complied with the Thesis Examination Procedure.

To my family

Jaie, Sushma, Sushil, Ankita, and Anshu

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Prof Chongmin Song, for his continuous support throughout my PhD study. His expert guidance and vast knowledge have greatly inspired me to strive for excellence in research. Further, his patience and kindness motivated me to move forward in the most difficult situations. Indeed, I could not have finished this thesis without his generous and persistent support.

I express my sincere thanks to Dr Sascha Eisenträger, who jointly supervised my PhD study. His valuable advice and comments were extremely helpful. I express my deep gratitude to A/Prof Hamid Valipour (PGC) for guiding me during the difficult phases. Further, I am grateful to Dr Ean Tat Ooi for his valuable suggestions and comments.

I also thank Dr Albert Saputra, Dr Junqi Zhang, Dr Hauke Gravenkamp, and Dr Weiwei Xing for their help and collaboration during my research. I had a great learning experience working with them.

A special thanks to NCI technical support team for their expert guidance that helped me work with Australia's largest supercomputer.

I would like to acknowledge the financial support I received from the University of New South Wales – University International Postgraduate Award (UIPA) – which allowed me to undertake my PhD study. I am also thankful to the university for extending the HDR completion scholarship and Prof Chongmin Song for the additional funding towards the end of my PhD study.

My heartfelt thanks to my wonderful friends, Lakshminarayanan, Saurav, Navjeet, Rakesh, Mandeep, Soumya, Krishna, Ramesh, Amolika, and everyone who supported me throughout my journey. Thank you Dr Nabanita Datta for guiding me throughout my research career.

Last but not least, I would like to extend my gratitude to my family. To my wife, for her love, support, patience, and sacrifices she has made. To my parents, sister, and everyone in the extended family for their endless support and encouragement. Certainly, my accomplishments would not be possible without them.

Abstract

The impact-induced fracture analysis has a wide range of engineering and defence applications, including aerospace, manufacturing and construction. An accurate simulation of impact events often requires modelling large-scale complex geometries along with dynamic stress waves and damage propagation. To perform such simulations in a timely manner, a highly efficient and scalable computational framework is necessary.

This thesis aims to develop a high-performance computational framework for analysing large-scale structural problems pertaining to impact-induced fracture events. A hierarchical grid-based mesh containing octree cells is utilised for discretising the problem domain. The scaled boundary finite element method (SBFEM) is employed, which can efficiently handle the octree cells by eliminating the hanging node issues.

The octree-mesh is used in balanced form with a limited number of octree cell patterns. The master element matrices of each pattern are pre-computed while the storage of the individual element matrices is avoided leading to a significant reduction in memory requirements, especially for large-scale models. Further, the advantages of octree cells are leveraged by automatic mesh generation and local refinement process, which enables efficient pre-processing of models with complex geometries.

To handle the matrix operations associated with large-scale simulation, a pattern-by-pattern (PBP) approach is proposed. In this technique, the octree-patterns are exploited to recast a majority of the computational work into pattern-level dense matrix operations. This avoids global matrix assembly, allows better cache utilisation, and

aids the associated memory-bandwidth limited computations, resulting in significant performance gains in matrix operations.

The PBP approach also supports large-scale parallelism. In this work, the parallel computation is carried out using the mesh-partitioning strategy and implemented using the message passing technique. It is shown that the developed solvers can simulate large-scale and complex structural problems, e.g. delamination/fracture in sandwich panels with approximately a billion unknowns (or DOFs). A massive scaling can be achieved with more than ten thousand cores in a distributed computing environment, which reduces the computation time from months (on a single core) to a few minutes.

Contents

Acknowledgments **vii**

Abstract **ix**

Contents **ix**

List of Figures **xv**

List of Tables **xxii**

Nonmenclature **xxix**

1 Introduction **1**

1.1 Computational modelling of structures 2

1.1.1 Finite-element based modelling 3

1.1.2 Spatial discretisation 4

1.2 Impact-induced fracture analysis 5

1.2.1 Cohesive element modelling 7

1.3 High-performance computational framework 7

1.4 Challenges 9

1.4.1 Automatic mesh generation 9

1.4.2 Local mesh refinement 10

1.4.3 Matrix operations in large-scale structural analyses 12

1.5 Proposed approach 13
1.6 Aim, objectives, and scope 16
1.7 Organisation of the thesis 18
1.8 List of publications 19
1.9 Notable citations 19

2 Literature review **20**
2.1 Computational modelling of structures 20
2.1.1 Mesh generation techniques 21
2.1.2 Numerical techniques 23
2.2 Time domain solutions in structural analysis 28
2.2.1 Explicit methods 29
2.2.2 Implicit methods 30
2.2.3 The EBE framework 33
2.3 Fracture analysis 34
2.3.1 Classical fracture theory 35
2.3.2 Cohesive zone models 36
2.3.3 Contact modelling 40
2.3.4 Compressive fracture analysis 42
2.3.5 Impact-induced fracture analysis 44
2.4 High performance computing 47
2.4.1 Memory architecture 48
2.4.2 Message passing interface (MPI) 50
2.4.3 Domain decomposition methods 52
2.5 Summary 57

3 Scaled boundary finite element method **61**
3.1 Formulation of SBFEM in two dimensions 61

3.1.1 Geometric modelling in scaled boundary coordinates 61
3.1.2 Formulation of linear elasticity in scaled boundary coordinates . 64
3.1.3 Derivation of the SBFE formulation for elastostatics 67
3.1.4 Solution of the SBFE equation 70
3.1.5 Derivation of SBFE formulation for elastodynamics 74
3.2 Formulation of SBFEM in three dimensions 76
3.2.1 Geometric modelling in scaled boundary coordinates 76
3.2.2 Formulation of linear elasticity in scaled boundary coordinates . 81
3.2.3 Derivation of the SBFE formulation for elastostatics 84
3.2.4 Solution of the SBFE equation 87
3.2.5 Derivation of SBFE formulation for elastodynamics 89
3.3 Summary 90

4 A parallel explicit solver for elastodynamic problems utilising octree mesh 92
4.1 Introduction 93
4.2 Octree-based mesh 95
4.2.1 Balanced octree mesh 96
4.2.2 Unique octree cell patterns 97
4.3 Explicit time integration 99
4.3.1 Central difference method 99
4.3.2 Mass lumping in SBFE framework 101
4.3.3 The octree-based pattern-by-pattern (PBP) approach 102
4.4 Parallel explicit solver for elasto-dynamics 107
4.4.1 Mesh partitioning strategy 107
4.4.2 The explicit solver 108
4.4.3 CPU time measurements and scaling 109
4.5 Numerical examples 115
4.5.1 Time-dependent excitations 115

4.5.2 Modal analysis 118
4.5.3 Transient analysis 120
4.5.4 Multi-storey building and ground 123
4.5.5 Mountainous region 131
4.5.6 Sandwich panel 135
4.6 Summary 146

5 An octree-based parallel implicit solver for elastostatic and elastodynamic analyses 149

5.1 Introduction 150
5.2 Implicit elastodynamics 152
5.2.1 Implicit time integration using HHT-α method 153
5.2.2 Parallel implementation 154
5.3 Linear solvers 156
5.3.1 Preconditioned conjugate gradient (PCG) solver 156
5.3.2 Octree-based matrix operations 159
5.3.3 Parallel implementation 160
5.4 Numerical examples 163
5.4.1 Prismatic beam under pure bending (Convergence study) 164
5.4.2 Concrete under uniaxial static loading 166
5.4.3 A building surrounded by trench 169
5.4.4 Seismic response of a castle on a mountain 178
5.5 Summary 184

6 Compressive fracture analysis in two dimensions 186

6.1 Introduction 187
6.2 The SBFE formulation for a cracked s-element in 2D 189
6.2.1 The stress intensity factor calculation 189

6.2.2 The sideface traction formulation 191
6.3 Cohesive and frictional-contact formulations 194
6.4 Numerical solution scheme and convergence criteria 199
6.5 Solution procedure 202
6.6 Numerical examples 204
6.6.1 Crack with frictional-contact 205
6.6.2 Cohesive crack propagation 210
6.6.3 Brazilian split test 214
6.6.4 Compressive fracture analysis in concrete 219
6.7 Summary 229

7 An octree-based parallel explicit solver for impact-induced fracture analysis 231
7.1 Introduction 231
7.2 Cohesive interface formulation 234
7.2.1 Single-mode loading 234
7.2.2 Mixed-mode loading 235
7.2.3 Contact formulation 239
7.3 Interface elements 241
7.3.1 Derivation of stiffness matrix 241
7.3.2 Unique interface element patterns 244
7.4 Explicit solver for cohesive-fracture simulation 245
7.4.1 Implementing interface element patterns 246
7.4.2 Parallel implementation 249
7.5 Numerical examples 249
7.5.1 Cubes under normal contact 250
7.5.2 Cantilever beam pair 259
7.5.3 Impact delamination of sandwich panel 268
7.6 Summary 280

8 Conclusions and recommendations **282**

8.1 Concluding statements 282

8.2 Recommendations for future work 288

Bibliography **291**

List of Figures

1.1 Meshing in FEM 3

1.2 Typical element shapes used in FEM 4

1.3 A sandwich panel with foam-core 6

1.4 Digital image of concrete: Blue and red colours indicate aggregate and matrix, respectively 10

1.5 A model of a castle with complex geometrical features 11

1.6 Octree-based meshing of the digital image of concrete 13

1.7 Scaling of an octree element 14

2.1 Barenblatt's model 37

2.2 Contact constraints 41

2.3 Initiation of wing cracks from interfacial flaw under compressive load . 43

2.4 A quadtree mesh and its dual graph 54

2.5 Direct communication of nodal interface force vectors between the workers 56

3.1 An s-element in 2D 62

3.2 An s-element in 3D 76

3.3 Isoparametric elements 78

4.1 Example of balanced octree mesh and octree cell patterns [1] 96

4.2 Face discretisation patterns for an octree cell. The number of hanging nodes on the edges of each face is indicated by n_{h}. 97

4.3 Transformation of octree cells 98
4.4 Lumping of the mass matrix 101
4.5 Lumped mass with Poisson's ratio $\nu = 0$ 103
4.6 Lumped mass with Poisson's ratio $\nu = 0.3$ 104
4.7 Scaling of an octree element 105
4.8 A sample of time distribution of parallel processing 111
4.9 Time-dependent excitations 116
4.10 Geometry and mesh of the cube 119
4.11 Relative error of first 100 eigenfrequencies 120
4.12 Geometry and mesh of the beam 121
4.13 Time histories of four selected points on the beam 122
4.14 Convergence curves of different types of meshes 122
4.15 Geometry of the building and ground 123
4.16 Dimension of the building (Unit: mm) 124
4.17 Octree mesh of the building and ground 125
4.18 The displacement and acceleration history obtained at the selected points in the building using the octree mesh (SBFEM) and ABAQUS (FEM) 126
4.19 Displacement and acceleration history obtained at the selected points on the ground surface obtained using the octree mesh (SBFEM) and ABAQUS (FEM) 127
4.20 Contour of displacement magnitude in the building and ground model at various time instances 128
4.21 Computational time study of the in-house code for the building and ground model 130
4.22 Geometry of the mountainous region 131
4.23 Boundary condition and dimensions of the mountainous region 131
4.24 Mesh partition of the mountainous region 132

4.25 Displacements and accelerations in y direction at five selected points in the mountainous region model 132
4.26 Contour of displacement magnitude of the mountain region model at four selected time instances t 133
4.27 Computational time study of the in-house code for the mountainous region 134
4.28 Geometry of the sandwich panel 136
4.29 Mirroring operation of the sandwich panel 136
4.30 Boundary condition of the sandwich panel with an enlarged view of the loading area (left) 137
4.31 Mesh of the sandwich panel 137
4.32 Mesh partition of the 1×1 sandwich panel 138
4.33 Response history of selected points of the sandwich panel 139
4.34 Contour of displacement magnitude at various time instances t 140
4.35 Computational time study of the in-house code for the sandwich panels 142
4.36 Memory usage of the in-house code for the explicit analysis of the sandwich panels 145

5.1 Prismatic beam geometry 164
5.2 Convergence study of beam in pure bending 165
5.3 The geometry of a 3D concrete obtained from X-ray CT scans 166
5.4 The octree mesh for concrete 167
5.5 The octree mesh for concrete under uniaxial static loading with the displacement contour 168
5.6 Computational time study (strong scaling) of the parallel solver for concrete under uniaxial static loading 169
5.7 Geometry of for the building surrounded by trench 170
5.8 Overall mesh of the building surrounded by trench 171
5.9 Ricker wavelet load for the building on a large soil block example . . . 171

5.10 Results at points A to F for the building surrounded by trench 172
5.11 Results at points G to L for the building surrounded by trench 173
5.12 Displacement contour plots u_{mag} for the building surrounded by trench at the given time stamps 174
5.13 Computational time study (strong scaling) of the parallel solver for the building surrounded by trench 175
5.14 Memory usage in the analysis of the building surrounded by trench . . 177
5.15 Geometry for the castle on a mountain example 178
5.16 Overall mesh of the castle on a mountain example 179
5.17 Half of the mesh (only showing elements within the positive $y-$axis) of the castle on a mountain example 180
5.18 Ricker wavelet load for the castle on a mountain example 180
5.19 Results at points A to F for the castle on a mountain example 181
5.20 Results at points G to K for the castle on a mountain example 182
5.21 Displacement contour plots u_{mag} for the castle on a mountain example at time snapshot 183

6.1 A cracked s-element in 2D 190
6.2 A cracked s-element in 2D with sideface tractions 191
6.3 Traction-separation relations along the crack-surface 196
6.4 Schematic representation of crack surface tractions implemented using interface elements (away from the crack-tip) and sideface tractions (near the crack-tip) 198
6.5 A cracked element showing linearized traction applied on the sidefaces . 200
6.6 Flow chart of the solution procedure 203
6.7 Infinite plate with circular hole 205
6.8 Finite-sized plate model 206
6.9 Mesh for the partially-cracked plate model 207

6.10 Variation of contact tractions along the crack surface for $\varphi = 45°$ (Analytical: [2]) 208
6.11 Convergence results of contact analysis of Mesh-A for $\varphi = 45°$ 209
6.12 Field variable contour plots for Mesh-A 210
6.13 SENS model under shear loading 211
6.14 Initial meshes for the SENS model 212
6.15 Results for cohesive crack propagation in SENS (Ref - Cendon (2000): [3], Ooi (2017): [4], Experiment: [5]) 213
6.16 Cohesive traction distribution along the crack in the deformed Mesh-A of SENS. (Deformation Scale: 100) 214
6.17 Specimen for the Brazilian split test 215
6.18 Initial meshes for the Brazilian test specimen 215
6.19 Results of the Brazilian test simulation (Ref - Rocco et al. (1999): [6], Experiment: [7]) 217
6.20 Cohesive traction distribution along the crack in the Brazilian test specimen of size $D/l_{ch1} = 0.9$ (Deformation Scale: 100) 218
6.21 Geometry and loading of the specimen 220
6.22 Initial meshes for various values of interfacial crack inclination angle α 221
6.23 Debonding load (P_D) (Experiment: [8]) 222
6.24 Crack surface traction distribution in the crack-tip region after debonding 223
6.25 Variation of normalised crack initiation load with r_0 and α (P.W. = Present Work, Exp. = Experiment [8]) 224
6.26 Wing crack initiation angle θ (Experiment: [8]) 225
6.27 Wing crack paths 226
6.28 Load-CMSD curves for the specimen (Experiment: [8]) 227
6.29 Various stages of wing crack propagation for specimen model for $\alpha = 36°$ (deformation scale = 100) 228

6.30 Variation of magnitude of contact traction along the initial crack interface with increasing CMSD (Δ_s) for $\alpha = 36°$ 229

7.1 Separation of top and bottom surfaces of the cohesive interface 234

7.2 Bi-linear traction-separation constitutive law in pure modes, $i \in (s, t, n)$ 235

7.3 Bi-linear traction-separation constitutive law in mixed mode 239

7.4 Linear interface elements 241

7.5 Face discretisation patterns for an octree cell with the number of hanging nodes on the edges of each face indicated by n_{h}. 245

7.6 Master interface element patterns 247

7.7 Cubes under normal static contact 250

7.8 Meshes for the cube with interface plane 251

7.9 Convergence study of the cubes under frictionless static-contact 252

7.10 Two impacting cubes 253

7.11 Linear vs 2nd order penalty regularisation 254

7.12 Results of penalty regularisation of Hex5 mesh (Analytical: [9]) 255

7.13 Convergence study of hexahedral mesh (Analytical: [9]) 256

7.14 Comparision of results obtained with hexahedral and octree meshes (Analytical: [9]) 257

7.15 Variation in energy with time (K.E. = Kinetic Energy, P.E. = Potential Energy, T.E. = Total Energy (P.E.+K.E.) 258

7.16 A pair of cantilever beams with contact interface 260

7.17 Meshes for the cantilever beam pair 260

7.18 Convergence study of the contact tractions 261

7.19 A pair of cantilever beams with cohesive interface 262

7.20 Loading conditions 263

7.21 Convergence study of the decohesion for various loading conditions . . 264

7.22 Impact loading condition 265
7.23 Results of dynamic decohesion process for various loading conditions . . 266
7.24 Displacement (deformation scale: 50) and interface damage value obtained for $p_0 = 5.0$ MPa at various time steps using Oct65 mesh 267
7.25 Geometry of the 1×1 sandwich panel 268
7.26 Mirroring operation of the sandwich panel 269
7.27 Boundary and loading conditions of the sandwich panel 269
7.28 Mesh of the 1×1 sandwich panel showing interfaces 270
7.29 Response history of selected points in 4×4 sandwich panel 271
7.30 Displacement contours obtained at selected time steps for 4×4 sandwich panel 273
7.31 Damage contours at the interface of foam-core and top cover sheet shown at selected time steps for 4×4 sandwich panel 274
7.32 Damage contours at the interface of foam-core and bottom cover sheet shown at the selected time steps for 4×4 sandwich panel 275
7.33 Computational time study for cohesive delamination of the sandwich panel 276
7.34 Memory usage in the cohesive delamination analysis of the sandwich panel 279

List of Tables

4.1 Computational time of the in-house code and ABAQUS for the building and ground model 129

4.2 Computational time of the in-house code for the mountainous region . 135

4.3 Computational time of the in-house code for the sandwich panel 141

4.4 Strong scaling results of the sandwich panel 143

4.5 Weak scaling results of the sandwich panel 143

4.6 Memory usage of the in-house code for the explicit analysis of the sandwich panels 146

5.1 List of meshes employed in the prismatic beam example 164

5.2 Strong scaling study for concrete under uniaxial static loading 168

5.3 Material properties for the building surrounded by a trench 171

5.4 Strong scaling study for the building surrounded by trench 176

5.5 Memory usage in the analysis of the building surrounded by trench . . 177

5.6 Material properties for the castle on a mountain example 179

7.1 Mesh properties for the cube 253

7.2 Mesh properties of the cantilever beam pair 260

7.3 Mesh properties of the cantilever beam pair 266

7.4 Mesh properties of the sandwich panels 270

7.5 Computational time for the cohesive delamination of the sandwich panel 277

7.6 Strong scaling results of the sandwich panel 277
7.7 Weak scaling results of the sandwich panel 278
7.8 Memory usage in the cohesive delamination analysis of the sandwich panel 279

List of Algorithms

4.1 Pseudo-code for the parallel explicit solver 110

5.1 Pseudo code for parallel HHT-α algorithm 155

5.2 Preconditioned conjugate gradient (PCG) method to solve a system of linear equation with a diagonal preconditioner 158

5.3 Pseudo-code for the parallel preconditioned conjugate gradient (PCG) method with a diagonal preconditioner 162

Nomenclature

$(\,)_b$	Subscript b denotes boundary of the s-element
$(\,)_{,x}$	$\frac{d}{dx}$
$\dot{(\,)}$	$\frac{d}{dt}$
$\ddot{(\,)}$	$\frac{d^2}{dt^2}$
$(\mathbf{B}_1,\ \mathbf{B}_2)$	Strain-displacement matrices in s-element
$\mathbf{C}$	Global damping matrix
$\mathbf{c}$	Integration-constant vector in SBFE solution
$\tilde{\mathbf{c}}$	Interface adhesion vector
$\mathbf{c}_f^e$	Element adhesion load vector
$\mathbf{D}$	Elasticity matrix
D	Scalar damage variable
$(\mathbf{E}_0,\ \mathbf{E}_1,\ \mathbf{E}_2)$	Coefficient matrices of s-element
E	Young's modulus
E_s^{rel}	Relative efficiency in strong scaling
E_w^{rel}	Relative efficiency in weak scaling

E_s	Efficiency in strong scaling
E_w	Efficiency in weak scaling
$(f_s,\ f_t,\ f_n)$	Allowable-stress (strength) components in interface elements
$\mathbf{f_t}(\xi)$	Nodal sideface force function in s-element
$\mathbf{f}^{\text{ext}}$	Global external force vector
$\mathbf{f}^{\text{int}}$	Global internal force vector
$\mathbf{f}_b$	Boundary nodal force vector in s-element
$\mathbf{f_t}$	Equivalent nodal boundary sideface force vector
G^c	Fracture toughness
$(\mathbf{i},\ \mathbf{j},\ \mathbf{k})$	Unit vectors of the Cartesian coordinate system
$\mathbf{J}$	Jacobian matrix
$(K_I,\ K_{II})$	Stress intensity factors
$(K_s,\ K_t,\ K_n)$	Penalty stiffness components in interface elements
$\mathbf{K}$	Global stiffness matrix
$\mathbf{K}^e$	Element stiffness matrix
$\tilde{\mathbf{K}}$	Constitutive matrix of interface element
$\mathbf{K}_f^e$	Interface element stiffness matrix
$\mathbf{K}^p$	Stiffness matrix of master element of pattern p
$\mathbf{L}$	Linear elastic differential operator
$(\mu,\ c)$	Mohr-Coloumb friction parameters

$\mathbf{M}$	Global mass matrix
$\mathbf{M}^e$	Element mass matrix
$\mathbf{M}_{\text{diag}}$	Diagonal mass matrix
$\mathbf{M}^p$	Mass matrix of master element of pattern p
M_S	System memory requirement
M_D	Model data memory requirement
M_T	Total memory requirement
$\mathbf{N}$	Shape function matrix of line/surface element in s-element
N_{cores}	Numer of computer cores
N_E	Number of elements of pattern p
$\mathbf{N}_f$	Shape function matrix for interfacial elements
N_P	Number of unique patterns in the octree mesh
$\mathbf{N}_u$	Interpolation function matrix in s-element
$\mathbf{N}_v$	Shape function matrix of the s-element
$\%T_C$	Percentage calculation time
$\mathbf{q}(\xi)$	Nodal force function in s-element
$\mathbf{r}$	Position vector with respect to the scaling centre of s-element
$(s,\, t,\, n)$	Local coordinate system in interface elements
S_s^{rel}	Relative speed-up in strong scaling
S_w^{rel}	Relative speed-up in weak scaling

S_s	Speed-up in strong scaling
S_w	Speed-up in weak scaling
$(t_s(\xi),\, t_n(\xi))$	Sideface traction components in the normal and shear direction to the sideface
$(t_s,\, t_t,\, t_n)$	Traction components in interface elements
$\mathbf{T}$	Standard transformation matrix
$t(\xi)$	Sideface traction
T_T	Total CPU time
T_C^m	Mean calculation time
T_W^m	Mean communication+waiting time
$\tilde{\mathbf{u}}$	Displacement field vector in s-element
$\mathbf{u}$	Global displacement vector
$\mathbf{u}(\xi)$	Nodal displacement function in s-element
$\mathbf{u}_b$	Boundary nodal displacement vector in s-element
$\mathbf{u_t}(\xi)$	Nodal sideface displacement function in s-element
$\%T_W$	Percentage communication+waiting time
$(x,\, y,\, z;\, t)$	Global Cartesian and time coordinate system
$\mathbf{Z}_p$	Hamiltonian matrix
$(\alpha,\, \beta,\, \gamma)$	Parameters in HHT-α time integration
$(\delta_s,\, \delta_t,\, \delta_n)$	Separation components in interface elements

δ^0	Separation at the onset of delamination in interface elements
δ_M	Mixed mode separation in interface elements
δ^f	Separation at total delamination in interface elements
Δt	Time-step size
Δt_{cr}	Critical time-step size in the explicit time-integration scheme
$\boldsymbol{\varepsilon}$	Strain field vector in s-element
$\langle \lambda_b \rangle$	Diagonal matrix containing the eigenvalues in SBFE solution
ν	Poisson's ratio
$\mathbf{\Phi_t}$	Modal sideface displacement (SBFE) solution
$\mathbf{\Phi}_b^q$	Force sub-matrix of eigenvector in SBFE solution
$\mathbf{\Phi}_b^u$	Displacement sub-matrix of eigenvector in SBFE solution
ρ	Mass density
$\boldsymbol{\sigma}$	Stress field vector in s-element
$(\xi,\, \eta,\, \zeta)$	Scaled boundary coordinates

Chapter 1

Introduction

Impact-induced fracture analysis is critical in many engineering and defence applications. Examples include composite panels used in aircraft structures subjected to collision with runway debris or bird-strike, the infrastructure components under the impact of falling objects such as hailstone or blast-induced flying debris, etc. An accurate simulation of impact-damage mechanisms during such events is highly desirable so as to predict the effect of damage on structural performance and avoid life-threatening accidents. Over the last few decades, the advancements in high-performance computing (HPC) facilities and numerical techniques have opened new pathways for capturing complex and transient dynamic failure modes in such materials resulting in highly durable and damage-tolerant structural designs.

This chapter starts with the discussion on computational modelling of structures in Section 1.1. In Section 1.2, an overview of the impact-induced fracture analysis is provided. Section 1.3 briefly introduces HPC for large-scale structural analyses. Section 1.4 outlines the challenges faced by existing numerical techniques relevant to this thesis. In Section 1.5, an overview of the developed approach to overcome the identified challenges is presented. Section 1.6 discusses the aim, objectives, and scope of the thesis. The organisation of the thesis is presented in Section 1.7. Finally, Sections 1.8

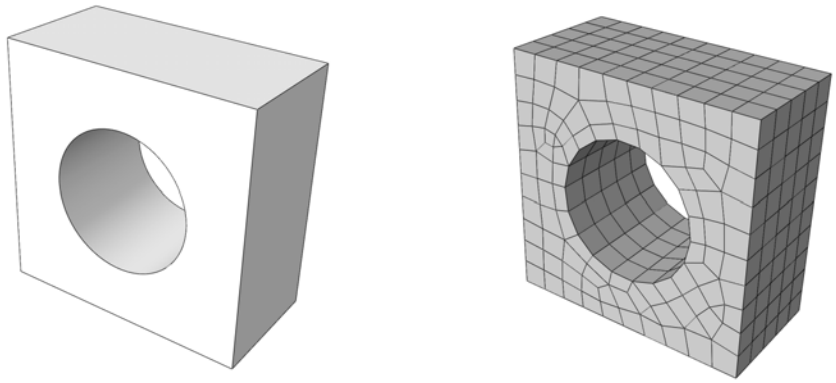
and 1.9 provide a list of publications and notable citations related to the current work.

1.1 Computational modelling of structures

The field of solid mechanics deals with the physical behaviour of solids and structures under applied loading and boundary conditions. It typically assumes a continuum nature of physical quantities such as mass density, displacement, stresses, etc., defined in the problem domain (or the structure under consideration). The continuum assumption allows describing the physical characteristics of the problem domain analytically using scientific laws. Such analytical descriptions often contain a set of complicated differential/integral equations and are referred to as mathematical models [10].

Most engineering problems are associated with complex geometry, material constitution, loading/boundary condition and non-linear behaviour. Such problems require complicated mathematical modelling whose analytical solutions are very difficult to obtain. With the advancement in electronic computation, the numerical solutions of these problems have gained much popularity among scientists and engineers. The numerical methods transform the complicated mathematical models into a set of simple algebraic equations using a suitable discretisation process. In this way, all the relevant features of the physical problem can be effectively modelled, and the solution can be obtained with the desired approximation.

As of today, several numerical techniques are available to solve the mathematical models pertaining to complex engineering problems. These include the finite element methods [10, 11], extended finite element method (XFEM) [12], scaled-boundary finite element method (SBFEM) [13, 14], iso-geometric analysis (IGA) [15], etc (reviewed in Chapter 2). Among these, the most commonly used numerical technique is the finite element method [11] explored in the following section.



(a) A geometry defining the problem domain

(b) Meshed geometry

Figure 1.1: Meshing in FEM

1.1.1 Finite-element based modelling

The finite element method (FEM) [10, 11, 16] is a powerful numerical technique widely used to solve real-world applications involving complex geometries and non-linear physical behaviour. The basic idea is to discretise or divide the problem domain into small-sized elements to form a mesh, as shown in Fig. 1.1. The discretisation process allows expressing complicated functions describing the behaviour of the problem domain in the form of a collection of simple polynomials. The polynomial functions are defined over each element and satisfy the desired level of continuity with the neighbouring elements at the connecting points. Each element is processed individually and later assembled in a matrix form to represent the required physical characteristics of the problem domain. This technique has been applied to various engineering problems concerning fracture mechanics [17–19], vibration analysis [20, 21], elastoplasticity [22, 23], contact mechanics [24, 25], etc.

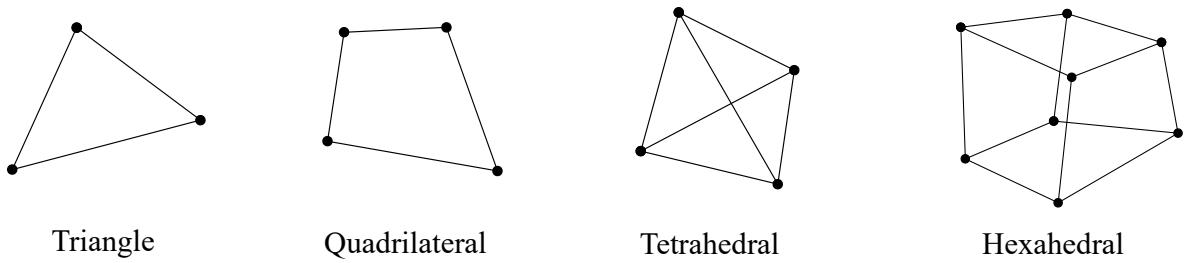


Figure 1.2: Typical element shapes used in FEM

1.1.2 Spatial discretisation

Though the finite element methods can capture the complex physical behaviour of the problem domain, the numerical solution obtained is only approximately equal to the exact solution. The errors are introduced in several stages of analyses as listed by Reddy [10].

One of the most significant sources of error is the spatial discretisation process or meshing. The complex problem domain is discretised using simpler element shapes, such as triangle/quadrilateral in two-dimensions and tetrahedral/hexahedral in three-dimensions (Ref: Fig. 1.2), which may not exactly represent the original domain. Generating a high-quality mesh is critical in numerical simulations for maintaining the fidelity of the underlying physics and obtaining solutions with high accuracy. The discretisation-induced errors may be reduced by refining the mesh (e.g., reducing the element size) to better fit around the complicated geometric shapes of the problem domain. In this case, a global refinement of the mesh domain may significantly increase the computational cost, and therefore, a local refinement is preferred.

The efficient generation of high-quality meshes is still an active topic among researchers worldwide. Several meshing techniques have been developed, including mapping [26, 27], sweeping [28–30], Delaunay triangulation [31, 32], advancing front [33, 34], and grid-based methods [35–37].

Among these techniques, the grid-based method is a promising technique for automating and speeding up computational simulations for solid mechanics. In this tech-

nique, the problem domain is discretised using a background grid (e.g. hexahedral elements), which can be directly used in the analysis [38]. The hierarchical grids with quadtree/octree decomposition are more popular, as they have an efficient data structure that can capture the complex geometrical features of different scales. Further, they allow fast mesh size transition in both CAD-based and image-based modelling leading to efficient local refinement of the mesh. However, to capture the shape of the domain, the grid is often trimmed at the boundary, resulting in arbitrary polytope elements. This requires additional treatments in conventional FEM since the shape function for arbitrary element shapes is not available [39].

The grid-based meshes can be effectively handled by numerical techniques, such as the scaled boundary finite element method (SBFEM), that provide shape functions for arbitrary polytope elements. Further, the grid-based meshes typically contain many geometrically similar elements that can be exploited for reducing the memory footprints and performing efficient simulations of large-scale structural problems. These advantages will be further discussed in the later sections of the present chapter.

1.2 Impact-induced fracture analysis

The impact-induced fracture analysis is crucial in many engineering applications. Here, the impact loading conditions can take various forms, ranging from a dropped tool travelling at a few metres per second to blast-induced debris travelling at more than a hundred meters per second. These extreme events, typically categorised as low- and high-velocity impacts [40–42], can induce different levels of damage, which subsequently affect the residual load-carrying capacities of the target structures.

Under low-velocity impact loading, the time of contact between the projectile and target is much longer. Therefore, the whole structure responds in such a way that the transmitted kinetic energy is accommodated well beyond the point of impact. Many

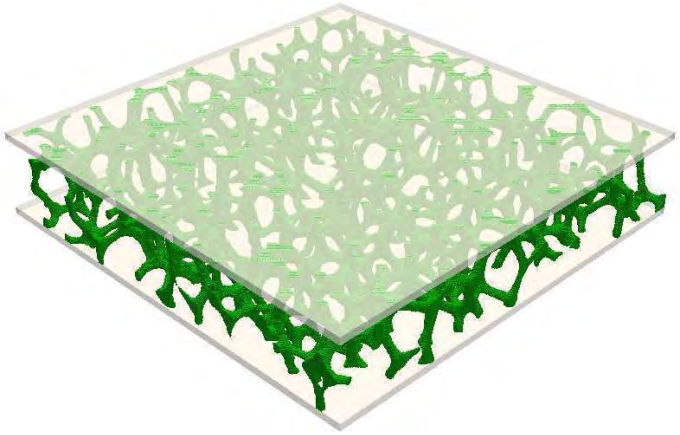


Figure 1.3: A sandwich panel with foam-core

factors, such as geometrical configuration (e.g., length of fibres), boundary conditions, etc., affect the energy absorbing capacity of the target. On the other hand, a high-velocity impact loading induce a highly localised response in the target, which results in energy dissipating over a much smaller region.

The fracture simulation of the high-velocity impact in composites laminates, sandwich panels (e.g., Fig. 1.3), and other heterogeneous materials like concrete is a fairly complicated process. The instantaneous contact condition between the impacting bodies induces stress wave propagation through the material. The boundary conditions do not influence the response as the impact event gets over before the stress waves reach the edge of the structure. Due to the heterogeneous nature of these materials, the failure can occur in various modes. In case of composites, failure modes include matrix/core crushing and cracking, delamination produced by interlaminar stresses, fibre buckling and failure, and full penetration with the projectile perforating through the target. On the other hand, failure modes in concrete can include mortar and aggregate cracking, decohesion at ITZ, partial penetration and full fragmentation of the target.

1.2.1 Cohesive element modelling

To capture the failure modes involving matrix cracking and delamination (or debonding) under impact loading, the cohesive interface-element modelling has been extensively utilised within the framework of finite-element based methods [43–47]. These interface elements are based on the cohesive zone approach, initially proposed by Dugdale and Barenblatt [48, 49]. A traction-separation relation is utilised, which allows modelling the onset of the fracture using the strength-based criteria and capturing the propagation using energy-based criteria. The main advantage of the cohesive elements is the ability to predict both the initiation and propagation of crack without previous knowledge of the crack location and propagation direction [50, 51]. Further, these interface elements can also model the contact interaction during the fracture process using the penalty-stiffness approach.

In general, impact-induced fracture/delamination simulations require a fine mesh discretisation to capture the non-linear behaviour arising from contact and cohesive interactions along the crack-path and material interfaces. As these non-linear interactions induce high-stress gradients over a localised region, an efficient local-refinement procedure can help reduce the computational burden. Additionally, capturing the realistic stress wave propagation in the dynamic analysis of heterogeneous materials like composites and concrete often require three-dimensional modelling. This can lead to a large number of unknowns (or DOFs) that need to be solved over many time steps. Therefore, developing an efficient parallel solver is inevitable for simulating such problems in a feasible time.

1.3 High-performance computational framework

Large-scale simulations with millions (or billions) of DOFs have gained much attention in the field of solid and structural mechanics [52–55]. Such simulations are useful in

many engineering applications that demand extensive three-dimensional modelling and sufficient resolution to accurately capture complex geometrical shapes and material-dependent length scales [56]. However, handling a vast number of nodes/elements in these problems can be computationally challenging and the serial execution of their solution procedure may have unrealistic CPU time requirements. Therefore, the development of a high-performance parallel computational framework is crucial for accurately and efficiently solving large-scale problems.

The large-scale simulations are usually carried out on a cluster of multiprocessor systems called the supercomputer. The supercomputers can have different architectural models such as shared memory, distributed/hybrid memory, etc. The shared-memory systems provide global memory access to all the processors for high-speed data sharing. However, the memory contention limits the performance of such systems as the number of processors increase. Most modern supercomputers are based on distributed/hybrid memory systems, which are highly scalable. Here, the memory is physically local to each processor (or a group of processors in a compute node). The processors communicate over a high-speed network for exchanging data using algorithms such as message-passing. A review of different memory systems is covered in Chapter 2.

There are several ways of exploiting parallelism ranging from the arithmetic level to solving decoupled equations in parallel. In the context of FE based analyses, the mesh-partitioning strategy [57] is commonly utilised. Here, the problem domain is decomposed into relatively smaller subdomains (or mesh partitions). At each step, the governing equations are solved in parallel over each partition, synchronising the data at the boundary of the partition. The mesh-partitioning strategy is highly scalable and complementary to the distributed memory architectures, making it a popular approach for solving large-scale structural problems. However, there are several limitations faced by the existing FE based techniques in carrying out large-scale simulations, as explored in the next section.

1.4 Challenges

In the field of computation mechanics, several challenges are encountered with large-scale modelling and analysis of structures (including fracture), especially when the existing mesh-based numerical methods such as FEM are employed. Within the scope of the thesis, the following issues are identified and discussed.

1.4.1 Automatic mesh generation

Over the past few decades, many researchers have extensively developed the theory and applications of meshing technologies [16, 58–61]. The current meshing technologies can automatically produce reasonably good approximations for computer-aided-design (CAD) based geometries using tetrahedral elements. However, it is well-known that the tetrahedral elements provide lower accuracy in analyses and are inefficient in modelling high-stress gradients. Thus, non-uniform hexahedral elements are usually employed for producing high-quality meshes. However, as the size and complexities of geometry increase, it becomes difficult to produce hexahedral meshes automatically using the current commercial mesh generators. Therefore, considerable time and manual labour are required by engineers to produce ideal meshes, especially in the case of large-scale problems [62].

Highly heterogeneous materials such as concrete, foam-core sandwich panels, etc., require modelling complicated geometric shapes to realistically capture the structural and fracture response. Such issues can be solved using image-based analyses. Here, the digital images (obtained from noninvasive imaging technologies such as X-ray CT scans, MRI, etc.) use colour intensities to provide the geometrical information and the material distribution of the problem domain, as illustrated in Fig. 1.4. The simplest automatic meshing approach utilises an all-hex grid-based technique, which models each voxel in the image as a hexahedral element [63]; however, it creates a large number of elements

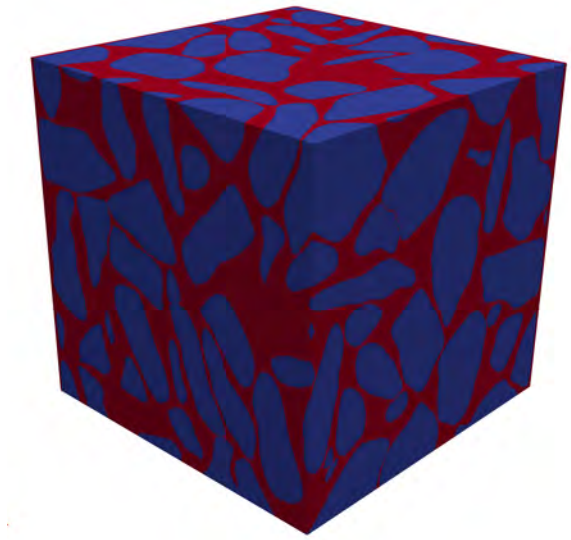


Figure 1.4: Digital image of concrete: Blue and red colours indicate aggregate and matrix, respectively

that significantly increase the computational burden. Another approach [64] utilises CAD-based meshing techniques for images; however, it still faces the aforementioned CAD-related meshing issues.

1.4.2 Local mesh refinement

Locally modifying a mesh region without modifying the whole structure can be tricky in conventional FEM, especially when quadrilateral or hexahedron elements are employed. This issue creates several difficulties in analysing complicated geometric models, (e.g., Fig 1.5), which require careful treatment to capture all the geometric features accurately. Additionally, local mesh refinement is also critical in fracture analyses and surface interaction problems, such as decohesion/contact phenomena, where a denser mesh is required over a localised area to capture the non-linear interaction process accurately. Further, in case of multi-scale modelling, the local refinement process play a crucial role in satisfying the discretisation requirements that can vary significantly in a single model to capture various physical phenomena at different scales.

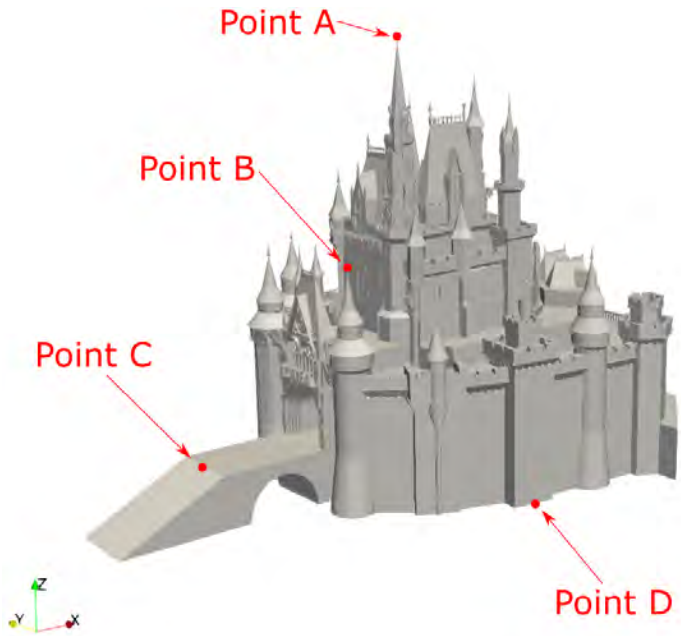


Figure 1.5: A model of a castle with complex geometrical features

The hierarchical grid-based technique such as octree-based decomposition (Ref: Section 1.1.2) is very suitable for achieving fast mesh transition. In this technique, the problem domain is systematically divided into octree cells of varying sizes that can efficiently capture discontinuities and complicated shapes at different scales using local mesh refinement (without modifying the global mesh). However, when two neighbouring octree cells have different sizes, they introduce hanging/incompatible nodes when FEM is utilised. To alleviate this issue, additional treatment such as triangulation/tetrahedralisation of the mesh is usually required [65, 66], which may not be an optimal solution given the drawbacks associated with the triangular/tetrahedral (constant strain) elements. Another approach is to use additional algebraic constraints on the hanging nodes to impose compatibility with the neighbouring cells [67]. However, it incurs additional computational effort proportional to the number of hanging nodes, and thus, can be significant.

1.4.3 Matrix operations in large-scale structural analyses

The numerical modelling of large-scale structural problems is inherently challenging due to the large number of nodes/elements involved in these problems.

The FE based techniques typically require the computation of the global system matrices (e.g., stiffness and mass matrices). This involves computing the element matrices using numerical integration and assembling them into the global system matrices using neighbour connectivity. However, calculating the global system matrices in large-scale structural problems with several million (or billion) elements can dramatically increase the memory requirements and computational burden.

An assembly-free method (also known as the element-by-element or EBE method) proposed by Hughes et al. in 1983 [68, 69] can effectively handle this issue. In the EBE method, element-level matrix operations are performed, which avoids the assembly of global matrices and reduces the associated memory requirements. However, using the EBE technique may not be very advantageous for large structural problems. This is because the overall memory consumption can still be very high due to the storage requirements of individual element matrices. The memory footprint can be reduced by recomputing the element-stiffness matrices at each step within the solver [70]; however, it induces additional computational expenses.

An alternative approach employs the congruency of the elements having the same geometrical and material properties [53, 71]. As the congruent elements have identical element stiffness matrices, only the distinct element matrices need to be computed and stored. This approach greatly reduces the memory footprint of the EBE-based solvers without increasing the computational burden. However, the unstructured meshes typically used in FE techniques may contain a large number of incongruent elements, and therefore, the efficacy of such an approach is only problem-dependent for these meshes. On the other hand, the hierarchical grid-based meshes containing quadtree/octree cells have a large number of congruent elements; however, these meshes are difficult to ana-

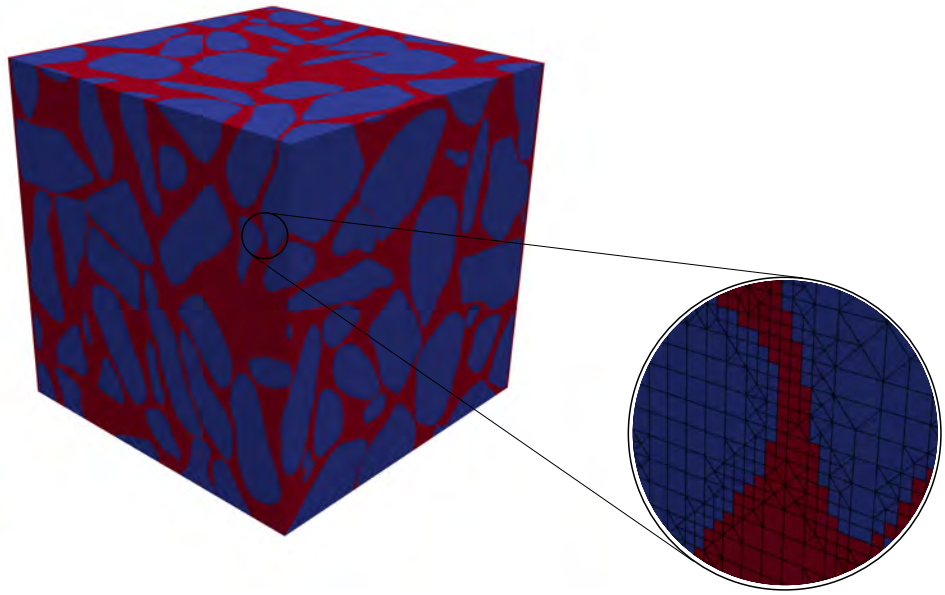


Figure 1.6: Octree-based meshing of the digital image of concrete

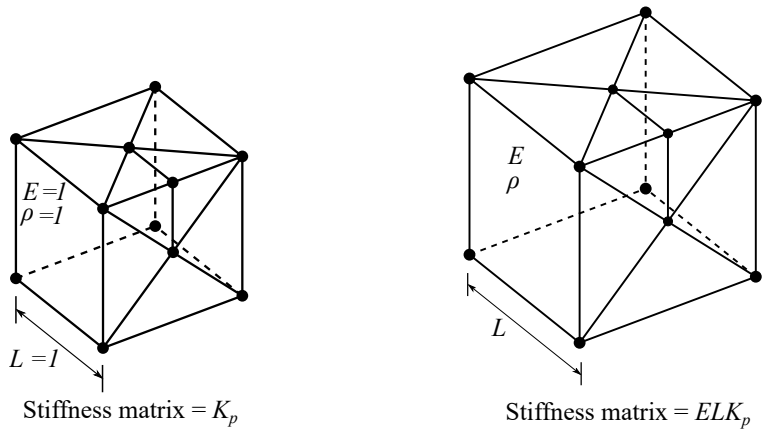
lyse using conventional FEM and require additional treatments explained previously.

1.5 Proposed approach

In this work, the scaled boundary finite element method (SBFEM) is utilised to overcome the challenges identified in the last section. The SBFEM is a semi-analytical method developed in the 1990s [13, 14, 72]. The fundamental theory of the scaled boundary finite element and its application was presented by Song [1].

Pre-processing

The hierarchical grid-based meshing comprising quadtree/octree cells is a promising technique for automating the mesh-generation process. Several researchers have utilised this technique in the SBFEM framework [73–76], and reported significant reduction in the time and effort spent in manual intervention typically required to produce a high-quality mesh. The quadtree/octree cells can be directly modelled as SBFEM-based polytope elements, which inherently eliminates the hanging node issue encountered in



(a) Master element (b) Scaled element

Figure 1.7: Scaling of an octree element

the conventional FEM [76, 77]. The other advantages of using this meshing technique include the local mesh refinement capabilities, the efficient data structure and the ability to capture complex geometric shapes in both the CAD and image-based analyses (Ref: Fig 1.6). For these reasons, the hierarchical grid-based mesh has been utilised in the present work for computational modelling of structures.

When the hierarchical grid-based mesh containing quadtree/octree cells are used in a balanced form (that is, the level of division between two neighbouring cells is limited to one; also called the 2:1 rule), they have a limited number of unique cell patterns [74, 75]. The element matrices of the cell patterns (master-element matrices) can be pre-computed and stored as a library. These matrices have low-memory requirements and are independent of the type and size of the problem domain. The matrices of any element (of a given pattern) in the problem domain can be calculated using a simple scalar multiplication with the master-element matrices, as indicated in Fig. 1.7. Since the storage of individual element matrices is avoided, the octree-patterns significantly reduce memory demands during the simulation of large-scale linear structural problems in three dimensions. Though some additional transformations are needed considering the orientation of the elements, their effect on overall computational resource require-

ment is relatively small.

The concept of exploiting mesh patterns can be extended to cohesive/contact interface elements for solving impact-induced fracture problems. Here, the interface elements with limited patterns are modelled that are compatible with the faces of the octree cells. The cohesive-stiffness matrix for interface elements of a given pattern can then be obtained by simple scalar multiplication of the master-element matrix with a weight variable considering element area and damage.

Analysis

Most of the iterative/explicit solvers used for FE based structural analyses can be decomposed into a sequence of matrix-vector products and inner products of vectors. These matrix operations constitute the most memory intensive and computationally demanding steps. Though the existing EBE approach is commonly utilised, there are several associated drawbacks, as explained in Section 1.4.3. In the present work, a pattern-by-pattern (PBP) approach is proposed, where the octree-cell and interface-element patterns are exploited to recast the majority of the work as pattern-wise dense matrix computations. The use of patterns leads to a much better cache utilisation and aids the associated memory-bandwidth limited operations. Therefore, the matrix computations can be carried out with a much higher operational (or arithmetic) intensity [78, 79] resulting in significant performance gains of the solvers, especially on modern computer architectures [80].

The PBP approach also supports large-scale parallelism for performing matrix operations. In this work, parallelisation is carried out using the mesh-partitioning strategy. The problem domain is divided into a set of smaller partitions, and each partition is assigned to the individual workers (or computer processes). The individual workers run a single computer program and carry out simultaneous local computations over the assigned partitions. The data at the interface of neighbouring partitions are synchronised

through a message-passing algorithm. The PBP approach can help achieve high scalability and efficiency for performing large-scale simulations in a distributed computing environment; therefore, its capabilities are explored in detail in this thesis.

1.6 Aim, objectives, and scope

This thesis aims to develop a high-performance computational framework for efficiently solving large-scale structural problems, especially impact-induced fracture, exploiting octree mesh patterns. The primary objectives include the development of:

- **Octree-based modelling**: The limited number of octree-patterns are exploited in the balanced octree-mesh for pre-computation of stiffness and mass matrices with very little memory needs. The octree-patterns are then employed for carrying out efficient matrix operations during the analysis using the pattern-by-pattern (PBP) approach. The pre-processing is bolstered by automatic octree-based mesh generation, local mesh refinement process and SBFEM-based shape functions to directly handle the octree-cells. These features enable efficient modelling of structures with complex geometries.

- **Parallel computational framework**: A massively parallel computational framework is developed to solve large-scale structural problems. The simulations are performed in a distributed computing environment using the mesh-partitioning strategy. The overall computational framework is implemented in Python using the message passing interface library MPI4Py [81, 82].

- **Linear-structural solvers**: The octree-based parallel formulation is first devised for linear-structural problems. An explicit-solver is developed for linear elastodynamic problems, followed by the development of implicit-solvers for linear elastodynamic and elastostatic problems. The efficiency, scalability, and robustness of

the developed solvers are examined and applied for solving various engineering problems.

- **Cohesive-fracture solvers**: A two-dimensional formulation is initially developed for cohesive/contact interface modelling. Its application is shown for compressive-fracture analysis in concrete involving debonding and matrix cracking. The formulation of the interface elements is then implemented with the octree-based parallel explicit solvers for performing large-scale cohesive-fracture simulation with impact loading in three-dimensions. The practical applicability of the developed solver is explored in detail.

The parallel simulations have been carried out on Gadi − a supercomputing facility provided by National Computing Infrastructure (NCI), Canberra. A typical compute node on Gadi has two sockets, each equipped with a 24-core Intel Xeon Scalable 'Cascade Lake' processor and 35.75 MB L3 Cache. The compute nodes are connected through a high-speed network with a data transfer rate of up to 200 Gbps.

Scope of the work

This thesis mainly focuses on the computational modelling and analysis of large-scale structural problems in a parallel computing environment. An octree-based formulation is developed, which exploits the repeating patterns in the mesh to reduce the memory footprint and improve the efficiency of the solvers.

There are several computer architectural concepts that can be considered to enhance the computational performance of a simulation [83], such as pipelining, instruction-level parallelism, process-mapping, cache policies, etc. However, a detailed study of these concepts is beyond the scope of the present thesis.

1.7 Organisation of the thesis

The rest of the thesis chapters are organised as follows.

- In Chapter 2, a literature review is provided within the scope of the thesis contents.
- In Chapter 3, the scaled boundary finite element formulation is derived for elastostatics and elastodynamics.
- In Chapter 4, a massively parallel explicit solver is developed for solving elastodynamic problems using SBFEM exploiting the octree meshes.
- In Chapter 5, the octree-based formulation is extended for developing a parallel implicit solver for elastostatic and elastodynamic analyses.
- In Chapter 6, a cohesive/contact interface formulation is developed for solving compressive-fracture problems in two dimensions.
- In Chapter 7, the cohesive/contact interface formulation is extended for solving impact-induced fracture/delamination problems in three dimensions utilising the octree-based parallel explicit solvers.
- In Chapter 8, the conclusions are discussed, and recommendations for future work are provided.

1.8 List of publications

Journal publications

- J. Zhang, A. Ankit, H. Gravenkamp, S. Eisentrager, C. Song, A massively parallel explicit solver for elasto-dynamic problems exploiting octree meshes, Computer Methods in Applied Mechanics and Engineering 380 (2021) 113811.
- A. Ankit, A sideface traction approach for cohesive and frictional crack growth problems using SBFEM, Computer Methods in Applied Mechanics and Engineering 386 (2021) 114076.
- A. Saputra, A. Ankit, J. Zhang, S. Eisentrager, C. Song, A parallel implicit solver for elasto-dynamic problems utilising octrees. (to be submitted)
- A. Ankit, J. Zhang, S. Eisentrager, C. Song, An octree-based massively parallel explicit solver for impact-fracture problems. (to be submitted)

1.9 Notable citations

- NCI Australia, 24 May 2021, Creating a super-efficient mesh solver for high-performance computing, <https://nci.org.au/research/research-highlights/creating-a-super-efficient-mesh-solver-high-performance-computing>.

Chapter 2

Literature review

In this chapter, a review of the existing literature relevant to this thesis is presented. In Section 2.1, the commonly used meshing and numerical techniques for structural analysis are discussed with a detailed review of the scaled boundary finite element method (SBFEM). In Section 2.2, the literature on time-domain solution schemes for structural dynamic problems is presented, where direct integration techniques such as explicit and implicit schemes are discussed. In Section 2.3, the development of fracture models is discussed. The computational techniques to implement cohesive crack propagation, cohesive laws and contact conditions are also reviewed. The high-performance computing systems and the associated numerical implementations are discussed in Section 2.4. Finally, the literature review is summarised in Section 2.5.

2.1 Computational modelling of structures

As introduced in Chapter 1, the finite element method (FEM) is the most widely used numerical technique for solving complex engineering problems. An essential aspect of this technique is the spatial discretisation of the problem domain to form a mesh. The generation of high-quality meshes is critical in obtaining an accurate solution and maintaining the fidelity of the underlying physics. In this section, we review the existing

meshing techniques in the literature. Later, we also review the commonly used numerical techniques, especially the scaled boundary finite element method (SBFEM), which is a promising technique for high-performance computing applications.

2.1.1 Mesh generation techniques

The spatial discretisation process typically involves dividing the problem domain into small-sized elements to form a mesh. The meshes can be primarily classified as structured or unstructured [59, 84, 85].

In structured meshes, each grid point is obtained as the intersection of curvilinear coordinate curves in 2D or surfaces in 3D. These meshes are simple to implement, efficient to use and require much lesser memory than an unstructured mesh of the same size. However, it is very difficult to discretise a complicated geometric domain. Further, the overall memory requirements in many problems can be much higher as mesh size transition is difficult to achieve using the structured mesh.

The unstructured meshes provide much higher flexibility in fitting complicated domains. They also provide a faster mesh size transition, which often balances the memory overhead caused due to the requirement of an explicit definition of the neighbour connectivity and coordinate information. Many times, a hybrid approach is utilised where a complicated domain is decomposed into simple patches of structured grids. However, hybrid methods are usually not fully automatic and require user guidance in the decomposition step.

Several meshing techniques have been developed for the efficient generation of structured and unstructured meshes. These can be broadly classified as geometry-first or mesh-first [86], which are briefly explained as follows.

Geometry-first approach

The meshing techniques based on the geometry-first approach utilise the CAD boundary representation to build the nodes and elements of a mesh to fill up the geometric domain. The geometry-first methods are normally sensitive to boundary topology, and a high-quality mesh is obtained at boundaries.

Mapping and sweeping are the two most commonly used meshing algorithms in this category. These are simple to implement and have minimal storage requirements. In the mapping technique [26, 27], a parent structured mesh is mapped conforming to the geometry of the model. On the other hand, the sweeping technique allows generating a 3D solid mesh from 2D surface meshes along a sweeping path [28–30]. Both the mapping and sweeping techniques generate high-quality mesh for simple geometries. However, complex geometries can lead to severe mesh distortion. This issue can be handled by decomposing the model into smaller patches, which requires additional human intervention.

For the generation of unstructured meshes containing tetrahedral elements (or triangle elements in 2D problems), the Delaunay criterion [31, 32] is most popularly used. This algorithm provides a criterion to connect a set of existing points in the space. As only tetrahedral elements can be generated, a high-mesh density is required to improve the accuracy of the solution.

The advancing front [33, 34] is another popular algorithm, where the elements are built progressively inward while maintaining an active front. This algorithm is robust for tetrahedral (or triangle) elements; however, the closure of the front is not always guaranteed when hexahedral (or quadrilateral) elements are employed.

Mesh-first approach

In the mesh-first approach, a space-filling grid or mesh of nodes and elements is first constructed. Then, local modifications to the mesh are employed to capture features of

the geometry and topology of the given model.

A commonly used method in this category is the grid-based method. Here, a background grid, usually made of high-quality elements such as quadrilaterals in 2D and hexahedrons in 3D, is employed. Though the background grid can be directly used to model the problem domain to perform the analysis [38], hierarchical grids containing quadtrees in 2D and octrees in 3D [35–37] are more popular as they allow implementation of local mesh refinement to model the problem domain and have an efficient data structure suitable for parallel processing. Further, this mesh structure can capture the complex geometrical features of different scales efficiently and allow fast mesh size transition in both CAD-based and image-based modelling. However, to capture the shape of the domain, the background grid is usually trimmed at the boundary, often reducing the quality of the elements in that region [87, 88]. Furthermore, additional treatment such as triangulation/tetrahedralization is also required in the conventional FEM following the trimming process as the shape function for arbitrary polytope elements is not available [39].

The grid-based meshes can be efficiently handled by numerical techniques that provide shape functions for arbitrary polytope elements, e.g., scaled boundary finite element method [1]. Further, these meshes typically contain many geometrically similar elements that can be exploited for significantly reducing the memory requirements and performing efficient simulations of large-scale structural problems. These features of the grid-based meshes will be explored in the later chapters of this thesis.

2.1.2 Numerical techniques

In this section, we review the commonly used numerical techniques in structural analyses.

Finite element method

The finite element method (FEM) is a popular numerical method widely used in the field of engineering. Literature on the development of the method can be found in [10, 11, 89, 90]. FEM has been applied to solve a wide variety of problems, including solid and fracture mechanics, elastodynamic analysis, heat transfer and fluid dynamics, electromagnetic analysis, etc. [91–94]. Due to its wide applicability, FEM implementation (serial and parallel) can be commonly found in numerous commercial codes such as Abaqus, NISA, ADINA, Ansys, etc.

A major disadvantage of using FEM is the difficulty in handling grid-based meshes, discussed in the previous section. The technique suffers from the issue of hanging/ incompatible nodes when the hierarchical grid-based meshes containing octree-cells are utilised in analysis. To alleviate this issue, additional treatments such as triangulation/tetrahedralisation of the mesh are usually required [65, 66]. Another technique employs algebraic constraints to impose compatibility at the hanging nodes [67] at the cost of additional computational effort proportional to the number of hanging nodes.

Extended finite element method

The extended finite element method (XFEM) was initially proposed by Belytschko and Black [12] to solve crack problems that avoided re-meshing issues. In this method, the approximate solution space in the FEM was enriched with discontinuous functions. Later, Dolbow et al. [95] further improved the XFEM by incorporating an asymptotic enrichment for crack-tip elements and a Heaviside function enrichment for other cut-elements. Their approach avoided excessive mesh refinement around the crack-tip and re-meshing during crack-propagation.

Subsequently, a generalised crack model was introduced by Belytschko et al. [96] to model arbitrary discontinuities, including intersecting and branching. This technique can be straightforwardly implemented with grid-based meshing, where arbitrary

geometric shapes can be modelled as discontinuities in the domain. Often, the mesh is used in the background to capture the complex geometrical shapes, increasing the overall computational effort due to large number of nodes and elements. To efficiently capture the geometry, the hierarchical grid-based meshes can be employed; however, it creates the issue of hanging/incompatible nodes in the mesh, which can be difficult to handle directly.

Iso-geometric analysis

The concept of iso-geometric analysis (IGA) was introduced by Hughes et al. [15] with the aim of integrating the computer-aided-design (CAD) with the FEM. In this method, the geometry is discretised into "patches", where the Non-Uniform Rational B-splines (NURBS) are utilised to obtain their shape functions. NURBS are used in most CAD software applications for geometric modelling; therefore, a common geometrical data-set can be used for both the design and analysis process. IGA has been applied to various problems, including structural analysis [97], wave propagation [98], contact analysis [99], fluid-structure interaction [100], etc.

To handle the problems with crack-tip singularity and arbitrary discontinuities in the mesh, methods combining XFEM based enrichment and IGA (called XIGA) have been proposed by some authors [101–103], which can be combined with grid-meshing techniques. However, XIGA suffers from the major drawbacks of sub-optimal convergence rates and ill-conditioning of system matrices. A stable generalised iso-geometric analysis (SGIGA) was recently proposed [104] to overcome this issue.

Boundary element method

The boundary element method (BEM) was introduced in the 1970s [105–107]. In BEM, only the boundary of the problem domain is discretised, which is much simpler than domain discretisation performed in the FEM. The method reduces the spatial dimension

of the problem by one, which is done by transforming the integral equations using approaches such as the divergence theorem, Stoke's theorem, and Green's identities [108]. Therefore, it involves much lesser DOF in analyses as compared to FEM. BEM has been shown to be particularly useful to solve problems involving unbounded domain [109] and fracture analysis [110–113].

However, the boundary element based solution relies on apriori availability of the fundamental solutions or Green's functions to satisfy the governing PDEs; e.g., the fundamental solution for LEFM problems is obtained from Kelvin's solution [114]. Further, the BEM based system-matrix is typically fully populated and unsymmetric, which is computationally more costly to solve than the symmetric and banded matrix encountered in FEM.

Scaled boundary finite element method

The scaled boundary finite element method (SBFEM) is a semi-analytical method developed by Song and Wolf in the 1990s [13, 14, 72]. It combines the advantages of the BEM and the FEM, alleviating some of their drawbacks. Like FEM, it allows the problem domain to be divided into various finite-elements (called the s-element in the SBFEM framework). For each element, the symmetric stiffness and mass matrices can be computed and later assembled to form the global equation. At the same time, similar to BEM, the numerical discretisation is only needed to be performed on the boundary of each element, reducing the problem dimension by one. However, no fundamental solution is required to solve the governing PDEs, unlike BEM. The only requirement is a scaling centre, which must be placed in the element from where the element's boundary is directly visible.

The semi-analytical formulation of the SBFEM has been utilised for efficiently handling unbounded domain problems [115–117]. In the case of elastic problems with cracks (bounded problems), it allows direct extraction of stress intensity factors when the

scaling centre is kept at the crack-tip [118, 119]. The SBFEM formulation has been employed for solving a wide variety of problems, including elastostatics and elastodynamics [120, 121], wave propagation [116, 117], acoustics [122], linear and non-linear fracture mechanics [119, 123–125], frictional-contact [126–128], etc. The fundamental theory of the SBFEM and its application is presented by Song [1] along with a computer program written in MATLAB.

To improve the efficiency of mesh generation, hierarchical grid-based meshes containing quadtree/octree-elements have been utilised by some authors in the SBFEM framework [73–75]. The quadtree/octree cells can be directly modelled as scaled-boundary based polytope elements, which automatically eliminates the issue of hanging nodes encountered in the FEM and XFEM. Using this approach, the whole meshing process can be automated, which drastically reduces the time and effort usually required due to manual intervention to produce an acceptable mesh [76]. The quadtree/octree based mesh generation technique has been utilised in several works in the framework of SBFEM, including macroscale and mesoscale cohesive cracking in concrete [4, 129], image-based analyses [74, 75], non-local damage problems [130], thermal conduction problems [131], etc.

When the quadtree/octree meshes are used in a balanced form, they have a limited number of cell patterns. For a given pattern, the element matrices (stiffness and mass matrices) of different sized cells are simply scalar multiples. This feature can be exploited in solving linear structural problems in 3D by pre-computing such matrices and later, quickly extracting (when required), which increases the efficiency of the approach [74, 75]. Although the octree-based formulation in the framework of SBFEM has been extensively studied and applied to a wide area of engineering, its applicability in modelling large scale structures and high-performance computing remains to be established.

2.2 Time domain solutions in structural analysis

The structures subjected to dynamic/impact loading can be analysed numerically by semi-discretising the governing equations of motions in space using one of the computational modelling techniques reviewed in the previous section. It reduces the problem to a system of second-order ordinary differential equations (ODEs) in time given by

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{f}^{\text{int}}(\mathbf{u}) = \mathbf{f}^{\text{ext}}, \tag{2.1}$$

with the initial conditions as

$$\mathbf{u}(0) = \mathbf{u}_0,\ \dot{\mathbf{u}}(0) = \mathbf{v}_0. \tag{2.2}$$

Here, $\mathbf{M}$ is the discrete mass matrix, $\mathbf{C}$ is the viscous damping matrix, $\mathbf{f}^{\text{int}}$ is the internal force vector due to element stresses, and $\mathbf{f}^{\text{ext}}$ is the external load vector . $\mathbf{u}$, $\dot{\mathbf{u}}$ and $\ddot{\mathbf{u}}$ are the nodal displacement, velocity and acceleration vectors. For linear structural dynamics problems, $\mathbf{f}^{\text{int}}(\mathbf{u}) = \mathbf{K}\mathbf{u}$, where $\mathbf{K}$ is the stiffness matrix.

The *direct integration method* is popularly employed to solve such ODEs for both the linear and non-linear structural dynamics problems. It is based on the idea that the dynamic equilibrium equation given by Eq. (2.1) is satisfied at discrete time steps t_i. The variation in displacements, velocities and accelerations is assumed to occur within each time interval Δt_i. Several forms of this assumption have been proposed, giving rise to various direct integration schemes, each having different accuracy, stability, and computational cost. These schemes can be broadly categorised into *explicit* and *implicit* techniques.

As a general guideline, the wave propagation problems are more suitably solved using explicit time-integration techniques, while the inertial problems are efficiently solved using implicit integration techniques. However, several factors, such as the mesh-topology, computer hardware in use, etc., influence the relative performance of both the approaches. Here, we briefly review the two techniques.

2.2.1 Explicit methods

The *explicit methods* [11] are commonly used for solving wave propagation problems, where the intermediate and high-frequency modes tend to be more dominant in the structural response. These methods allow displacements at a given time step to be obtained by considering the equilibrium conditions and material and geometric properties at the previous time step. This feature is particularly useful in non-linear dynamic problems as the internal iterations within the given time step are not required to achieve the equilibrium.

Time-integration schemes

Several forms of explicit time-integration schemes have been proposed. Among the second-order explicit methods, the *central difference method* (CDM) has the highest accuracy and stability limit [132].

The CDM is popularly used for solving large-scale structural dynamic problems. However, due to the small time step size requirement, the scheme is only effectively utilised when each time step solution is performed very efficiently. Thus, the method is primarily used with a lumped mass matrix formulation that allows the system of equations given by Eq. (2.1) to be solved without factorising any matrix. Another advantage of using the central difference method is that the internal force vector can be obtained using the element-level computation or the element-by-element (EBE) technique [68, 69]. This feature avoids the assembly of the global stiffness matrix, which significantly reduces the storage requirements for large-scale problems. Most of the steps in CDM can be parallelised to a large extent using the mesh partitioning strategy, which allows very large-sized structural systems to be solved efficiently in a parallel computing environment [57, 133].

The CDM has a very small critical time step size requirement, and therefore, it is mostly suitable for simulations of short duration. Further, it is non-dissipative, due

to which the solutions can contain significant spurious oscillations. A few authors [134–137] have proposed schemes to improve the quality of the numerical solutions by controlling numerical dissipation in the high-frequency ranges. The explicit time integration methods with higher-order accuracy have also been developed, which allow larger time steps. The classical fourth-order Runge-Kutta method is popularly used as the one-step higher-order method [21, 138]. The other higher-order methods have been proposed in [139–141]. However, most of the higher-order methods are computationally expensive and have larger memory requirements for storing additional higher-derivative vectors, e.g., the time derivative of acceleration.

2.2.2 Implicit methods

The *implicit methods* [142] are more suitable for solving inertial or vibration problems, where the response is controlled by low-frequency modes. In these methods, the solution at a given time step are obtained using the displacement, velocity and acceleration at the same time step. This requires the system of equations to be solved simultaneously. Most of the implicit algorithms are unconditionally stable (for linear problems); therefore, they permit a large time step size that can be determined without stability consideration. However, the computational cost per time step is high, and the storage requirements tend to increase dramatically with the increase in size of the finite element mesh.

Time-integration schemes

The commonly used implicit time integration schemes for practical analyses are second-order accurate [21, 143]. These algorithms have controllable algorithmic dissipation, which is often utilised to eliminate the spurious high-frequency modes in the numerical solutions and to enhance the stability of the non-linear analyses.

The conventional implicit time integration schemes are the Newmark family of meth-

ods [144], Wilson-θ method [145] and Houbolt method [146]. The Newmark family of methods is one of the most popular algorithms used in structural dynamics problems. The method is second-order accurate (no-dissipation), provides a one-step solution algorithm, and has no special starting procedures requirement to advance the solution. The method exactly satisfies the energy conservation for undamped systems and is highly convergent for non-linear structural dynamics problems. The amount of dissipation can be increased; however, the accuracy of the solution is reduced to first order. Further, the dissipative properties of this method are not very effective since the lower modes are too strongly affected.

The finite-difference approximations used in the Newmark method have been utilised for the development of improved methods such as the HHT-α method [147], the WBZ-α method [148], and the generalised-α method [149], which are second-order accurate. In these methods, a parameter α is introduced to linearly vary at least one of the force members (inertial, viscous damping, and internal stress) within the time interval, which provides a controlled dissipation without loosing accuracy.

More recently, composite implicit time integration procedures have been proposed [150, 151] for solving non-linear problems. A few higher-order implicit algorithms have also been developed, which permit a much larger time step size and are very effective in solving highly non-linear problems [152–154]. However, they require more computational effort per time step.

Linear solvers

The computational cost of the implicit methods in structural dynamics problems is mainly dominated by the requirement of solving the linearised equilibrium equations with a sparse system matrix at each time step. Therefore, the choice of an appropriate linear solver strongly affects the feasibility of the given analysis. There are two classes of methods commonly employed: Direct sparse and iterative methods.

The *direct sparse methods* [155] can be used to find an exact solution of linear equations with a sparse system matrix in a fixed number of operations. These methods are robust, reliable and can solve a wide area of structural problems. Several efforts have been made towards the development of direct sparse solvers for shared and distributed memory architectures [156–160]; however, the CPU and memory requirements of these solvers increase too rapidly with problem size, especially in three dimensions.

The *iterative methods* such as those based on the generation of Krylov subspaces have gained much popularity in the finite element community [161–164]. The iterative methods provide better scalability for parallel execution and have lesser memory requirements for large-sized problems. These methods are highly suitable for distributed as well as shared memory machines. The iterative methods start with an initial guess and converge towards the exact solution in several iterations. However, when the system matrix is ill-conditioned, iterative schemes suffer from convergence issues. An appropriate choice of preconditioners is usually required to improve the convergence properties. An extensive review of the preconditioners for Krylov subspace based methods was recently covered by Pearson and Pestana [165]. Another issue with the iterative methods is the lack of robustness. A particular iterative solver can effectively solve only a narrow class of problems associated with a specific type of system matrix and may suffer from convergence issues for other classes.

Most of the finite element-based structural problems are associated with system matrices that are symmetric and positive-definite. This class of problems can be effectively solved using the conjugate gradient (CG) algorithm [166, 167]. The CG method converges to the exact solution in at-most n iterations, where n is the order of the system matrix. When utilised with preconditioners, the method is called the preconditioned conjugate gradient (PCG) method which has a better rate of convergence. In other problems such as those involving fluid flow simulation, the finite element discretisation of the non-linear Navier-Stokes equations leads to systems of equations that are

non-symmetric and indefinite [11, 168]. Several iterative schemes have been proposed to handle them [163, 169–173].

2.2.3 The EBE framework

Most of the explicit or the implicit (iterative) schemes require global matrix operations in their algorithm. However, in large-scale structural problems with several million (or billion) elements, the calculation of global system matrices can dramatically increase the memory and computational requirements.

The element-by-element or EBE method was proposed by Hughes et al. in 1983 [68, 69, 174, 175] to effectively handle this issue. In the EBE method, the matrix operations are performed at an element level, which avoids the assembly of global matrices and the associated memory requirements. Additionally, the EBE approach also supports the parallel execution of the solution scheme using the mesh partitioning strategy (discussed in Section 2.4.3).

Though the EBE method avoids the assembly of global matrices, the overall memory consumption for large structural problems can still be very high due to the storage requirements of individual element matrices. The memory footprint can be reduced by recomputing the element-stiffness matrices at each step within the solver [70]; however, it induces additional computational expenses. Alternatively, the congruency of the elements having the same geometrical and material properties [53, 71] can be exploited to solve the memory problem without increasing the computational burden. As the congruent elements have identical element stiffness matrices, only the distinct element matrices need to be computed and stored. However, the efficacy of such an approach is only problem-dependent as the unstructured meshes typically used in FE techniques may contain a large number of incongruent elements.

Pre-conditioners

In the PCG method, a preconditioner is employed to facilitate the rapid convergence of the iterative solution procedure. To implement the PCG solver in the EBE framework, preconditioning needs to be carried out at the element-level. However, most preconditioners supporting shared or distributed memory system based parallelism have a global architecture; thus, they require sparse matrix representation of the assembled system. Only a few preconditioners, such as diagonal (or Jacobi) [175] or Hughes-Winget (HW) [176], allow EBE implementation. The diagonal preconditioner is simplest to implement, computationally inexpensive, and supports efficient parallel implementation; however, it has a poor performance in improving the convergence of the PCG solver. HW preconditioner provides better preconditioning for some classes of finite elements (e.g., with nearly cubical aspect ratios) and has excellent performance on platforms with vector processors [177, 178]. However, utilising the HW preconditioner is computationally expensive, and parallel implementation requires careful synchronisation at the shared nodal degrees of freedom.

2.3 Fracture analysis

The study of fracture behaviour in structures is indispensable in many engineering applications. The propagation of cracks in a continuum domain is associated with the generation of new surfaces. It tends to reduce the stiffness or load-bearing capacity of the material and may lead to failure. Depending on the material type, several models have been developed to predict the fracture properties.

In this section, we first discuss the classical fracture theory. This is followed by a review of cohesive and contact modelling in cracks. Next, we review the fracture analysis under compressive and impact loading, which has several engineering applications.

2.3.1 Classical fracture theory

The classical theory – the energy based or stress intensity factor based – was originally developed for elastic bodies with sharp cracks and formed the basis of linear elastic fracture mechanics (LEFM).

In 1921, Griffith [179] formulated the energy-based solution of cracks in elastic materials. He calculated the energy release rate criterion for the crack propagation and the critical loading condition for an unstable crack extension. In 1947, Irwin [180] developed the near-crack-tip solution in terms of stress intensity factor (SIF). The SIF describes the state of stress in the crack-tip region and depends on specimen geometry, crack dimensions, and loading conditions. He used the SIF-based approach to derive the criterion for crack-propagation and stability conditions. The two crack formulations are related by: $G_c = K_c/E'$. Here, G_c is the critical energy release rate, K_c is the critical SIF, and $E' = E$ for plane stress and $E' = E/(1-\nu^2)$ for plane strain conditions, where E is the Young's modulus, and ν is the Poisson's ratio.

The classical fracture model is based on the theory of linear-elasticity that predicts an infinite value of stresses at the crack-tip, even for small deformations. This contradicts the behaviour of real materials, where the presence of a non-linear softening zone (fracture process zone) and hardening zone (plastic zone) in front of the crack-tip limits the stresses to finite values [181]. Different materials show different fracture behaviour depending on the size of the process zones. The brittle materials (e.g., glass, PMMA, brittle ceramics, etc.) have very small softening and hardening zones with respect to the size of the structure. In contrast, the ductile materials (e.g., ductile metals) have a small softening zone but a much larger hardening zone. The quasi-brittle materials (e.g., concrete, rock, composites, etc.) have a relatively large softening zone compared to the hardening zone. Here, the strength of the structure depends on not only the strength of the material but also the size of the structure (size-effect).

The LEFM can reasonably predict the fracture behaviour in brittle materials but

fails in other cases. Many theories have been proposed to address this inconsistency and capture the fracture behaviour in real materials. One of them is the cohesive zone model (CZM), originally presented by Dugdale [48] and Barenblatt [49].

2.3.2 Cohesive zone models

In Barenblatt's model, the process zone ahead of the physical crack-tip is represented as a cohesive zone. In this zone, the separation (δ) of the material is resisted by the interatomic cohesive stresses σ_{coh}, which is assumed to follow a cohesive traction function shown in Fig. 2.1. Further, the following three assumptions are made [182]:

1. The cohesive zone is small compared with all other dimensions and concentrated in a small region near the crack-tip (equivalent to the small process zone assumption in LEFM).

2. The cohesive traction function is independent of the loading, geometry and crack dimensions.

3. The stress singularity at the crack-tip vanishes; thus, the stresses remain finite in the material domain. Mathematically, it can be expressed as $K_I + K_I^{coh} = 0$, where K_I and K_I^{coh} are the SIFs due to external loading and cohesive tractions, respectively. Physically, the opposite crack faces join each other smoothly at the end of the process zone, as observed in Fig. 2.1.

Before Barenblatt, Dugdale [48] worked on a similar idea to study the localised plastic deformation occurring ahead of a crack-tip. He used a thin sheet of ductile material and applied the tensile stress in a direction perpendicular to the direction of the crack line. The plastic zone ahead of the crack-tip was modelled similar to the cohesive zone, and the cohesive stresses were given by the yield stress of an ideal plastic material.

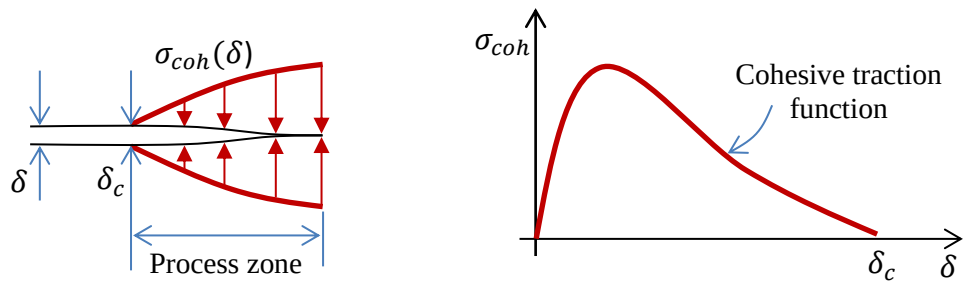


Figure 2.1: Barenblatt's model

Barenblatt's formulation of the cohesive zone model can be extended to capture the realistic fracture behaviour of many materials such as concrete, fibre composites, metals, etc. Depending on the material and the fracture process behaviour, different approximations for the cohesive-traction-separation law could be used [182]. The formulation can also be utilised in the simulation of interfacial delamination in composites. Here, the onset of the softening process and the delamination propagation can be predicted without previous knowledge of the crack location and propagation direction. Further, the cohesive laws could be defined in vector form to consider all modes of separation (I, II and III) occurring along the interface.

Within the framework of finite element based methods, the cohesive zone models are very well suited for the numerical treatment of fracture processes. These models have been extensively used by researchers to study fracture in brittle and quasi-brittle materials as they provide flexibility in capturing complex crack geometries, loading conditions, and material models [3, 18, 50, 51, 77, 123, 129, 183–189]. Further, they have been employed to simulate a wide range of dynamic fracture problems [43, 185, 190–192]. The cohesive tractions can be implemented using the discrete representation of crack interface within the mesh using one of the following approaches:

1. As the traction boundary condition along the interface.

2. By inserting non-linear springs at the interface nodes

3. By inserting the cohesive interface elements (CIEs) along the interfacial surface.

Among these techniques, the CIEs are most commonly employed for modelling cohesive cracks. Next, we briefly discuss the computational approaches to implement the cohesive crack propagation and the cohesive laws in the framework of finite element based methods.

Cohesive crack propagation

Two main strategies can be found in the literature for modelling cohesive crack-propagation: *Intra-element* and *inter-element crack modelling*.

For *intra-element crack modelling*, the crack is extended within the elements, and the cohesive zone is placed along the new crack surfaces as the crack propagates [3, 18, 123, 162, 187, 188, 193–195]. In case of fracture within the bulk material (e.g. matrix cracking in quasi-brittle material like concrete), where the crack path is not known in advance, special criteria are needed to implement the crack-initiation/propagation condition and direction. To accurately impose these criteria, fine meshes are typically required near the crack-tip region. Further, re-meshing is needed to accommodate the advancing crack. The re-meshing problem can be avoided using the X-FEM approach [196] that utilises nodal enrichment and partition of unity concept. However, as the crack path is unknown apriori, a globally refined mesh is needed for the accurate computation of crack paths. Overall, the intra-element approach is suitable for modelling a few major cracks, while capturing distributed cracks and branching behaviour can be challenging.

The intra-element approach has also been utilised to model embedded discontinuities within the mesh at an element level [197–199]. A similar approach called the *strong discontinuity approach* (SDA) has been used [19, 200] for analysing cohesive-crack propagation in quasi-brittle materials. In these works, the use of embedded discontinuities avoids re-meshing as the crack propagates or shear bands are formed;

however, the method suffers from stability issues.

In *inter-element cracks modelling*, the cohesive zone is placed between pairs of neighbouring elements along the element-edges/surfaces in the mesh [185, 186, 201–203]. In this case, the crack growth, branching, or fragmentation is a natural outcome of the fracture analysis in bulk materials. However, very fine meshes are needed to accurately model the crack path due to the limits imposed by the simple geometries of the standard finite elements, e.g., triangles and quadrilaterals, that can induce mesh dependencies. For the problems where the crack path is known, e.g., decohesion/debonding along the material interface, the inter-element approach is conveniently utilised [183, 184, 204]. The practical applications include delamination analysis of laminated composites and sandwich panels, masonry fracture, etc. [50, 51, 205–209]

Cohesive laws

The commonly used cohesive law can be categorised as *intrinsic* and *extrinsic* approaches that differ in capturing the initial response of the cohesive surface.

In the *intrinsic approach*, the cohesive surface is assumed to have an initial elastic response prior to the onset of fracture. In this approach, the cohesive elements are initially placed within the finite element mesh, and the onset of damage is automatically controlled by the cohesive law. This approach is widely used as it is relatively simple to implement. However, due to the finite value of initial stiffness, the cohesive elements contribute to the deformation of the mesh even before the onset of damage which induces artificial compliance in quasi-static simulations of cohesive fracture [210]. Further, the initial stiffness influences the dynamic fracture process and causes convergence issues (except where crack-path is known).

The *extrinsic approach* assumes a rigid response of cohesive surface prior to the onset of fracture. Here, the cohesive elements are only introduced in the finite element mesh based on an external traction-based criterion. However, the approach suffers from time

discontinuity immediately following cohesive element insertion. A detailed analysis of the two classes of cohesive laws has been performed by Kubair and Geubelle [211].

Despite the capability of the cohesive element approach in handling a wide range of fracture mechanics problems, it is associated with many numerical issues and physical limitations such as mesh-dependencies of the crack propagation paths and energy release, scalability for 3D parallel implementations, stability of time integration algorithms, etc. The details are discussed in [56, 185, 212–216]. In inter-element crack modelling, the issue of mesh dependency (caused by the inability of the cohesive element approach to spatially resolve the fracture process zone) requires very highly refined meshes for accurate computation, especially in three dimensions. When the extrinsic approach is utilised, an efficient dynamic insertion of the interface elements and the management of the associated arbitrary changes in the topological mesh are required. For these reasons, it often becomes necessary to carry out large-scale parallel simulations to accurately solve these problems in a reasonable time frame [54, 202, 217].

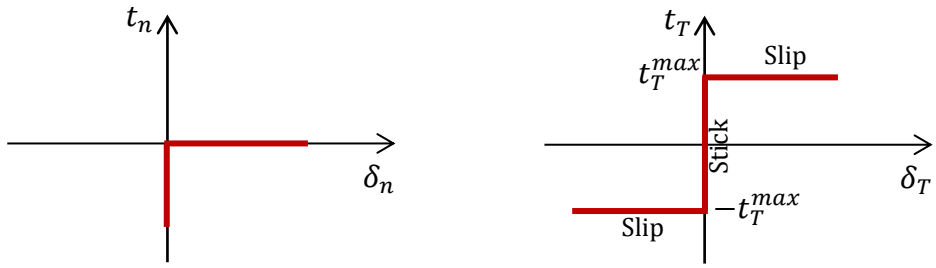
2.3.3 Contact modelling

During the crack-closure, the contact conditions need to be satisfied over the crack-surfaces. In this section, we give an overview of the general contact formulation that is also applicable for the crack-closure problems.

When two bodies come in contact with each other, the contact condition needs to be satisfied over the contacting surface to avoid penetration and consider the effect of friction. These contact conditions can be formulated in the following way.

If there is a gap between the contacting surfaces ($\delta_n > 0$), then the contact pressure is zero ($t_n = 0$). Once the contact is established ($\delta_n = 0$), the contacting surfaces exert normal pressure ($t_n > 0$) on each other. These conditions can be combined as

$$\delta_n \geq 0, \quad t_n \leq 0, \quad t_n \delta_n = 0 \ . \tag{2.3}$$



(a) Normal contact (b) Coulomb friction

Figure 2.2: Contact constraints

Eq. 2.3 is commonly referred to as the Hertz-Signorini-Moreau condition [218] for friction-less contact. The governing constraints form the classical Kuhn-Tucker conditions in mathematical optimisation problems. The normal contact constraint creates a non-smooth formulation, as shown in Fig. 2.2a.

The relative sliding of the contacting surfaces exerts an equal and opposite frictional force on each other. This is due to the local and micro-mechanical phenomenon occurring at the contact surface. An overview related to numerical modelling of friction has been covered by Oden and Martins [219]. Among these models, the Coulomb friction law is the most widely used for practical purposes. It is given by

$$t_T \leq \mu |t_n| + c \ . \tag{2.4}$$

Here, t_T is the tangential contact traction. μ is the frictional coefficient and c is the adhesion, which depend on the material of the contact-interface. Coulomb friction law states that the maximum value of tangential contact traction t_T^{max} is a linear function of the magnitude of the normal contact traction. The stick and slip conditions are related to the Coulomb friction model, as shown in Fig. 2.2b.

Computational methods

There are several well-established computational methods for enforcing contact- conditions [25], such as the Lagrangian multipliers, penalty, augmented-Lagrangian, etc. In the Lagrange multipliers method, the contact condition is exactly satisfied by introducing additional variables into the system [220, 221]. However, the system matrix contains zero diagonal terms, which may cause numerical issues. The penalty method approximates the constraint problem by the use of penalty parameters. This technique is robust and straightforward to implement with cohesive zone models; however, the non-penetration condition is only approximately satisfied [222, 223]. Further, the optimum value of the penalty parameter is not well defined.

The implementation of contact conditions requires contact surface discretisation, which is usually achieved using node-to-node (NTN), node-to-segment (NTS) and segment-to-segment (STS) schemes. The NTN scheme is the earliest technique and typically involves matching the nodal discretisation at the interface of two contacting bodies. The method is only applicable where the relative sliding displacements of the two contact boundaries are sufficiently small [25, 224, 225]. For problems involving larger contact deformations, the NTS or node-to-segment scheme is more appropriate [24, 226, 227]. The STS or segment-to-segment scheme is another discretisation approach where the contact constraints are satisfied through interpolation along the contact segment defined at the interface of the contacting bodies [228–231].

2.3.4 Compressive fracture analysis

The study of cracks under compressive stress fields has many practical applications. One of them is the compression analysis of concrete, briefly reviewed in this section.

Concrete is a quasi-brittle material with a complex heterogeneous structure. It is made of hardened cement paste, aggregates, and voids. Reliable prediction of the material behaviour of concrete is an important research topic in structural engineering.

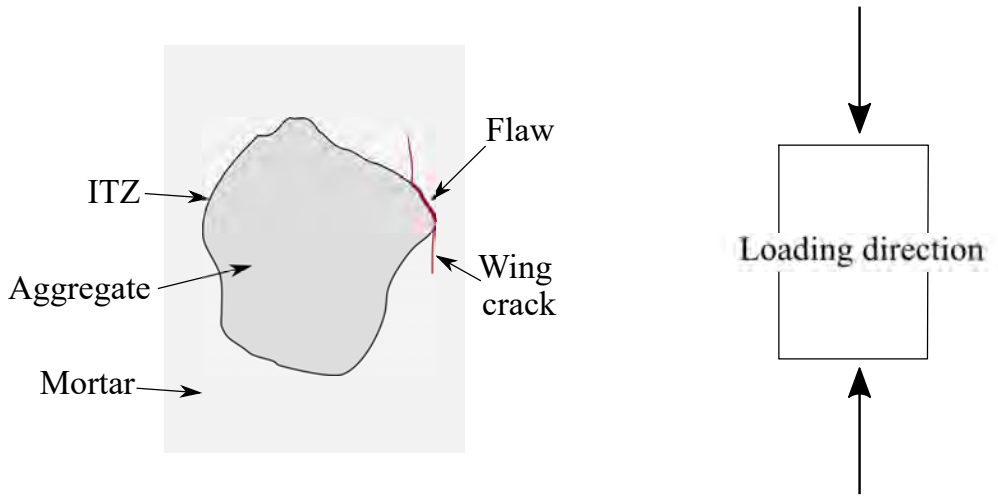


Figure 2.3: Initiation of wing cracks from interfacial flaw under compressive load

For this purpose, various numerical models have been developed by researchers that can capture the material heterogeneity and behaviour of concrete at different length scales [3, 18, 129, 232–241].

In a concrete specimen, the residual stresses and micro-cracks are usually present in the unloaded state. When subjected to compressive loading, the specimen shows a complicated deformation process involving progressive failure due to the initiation, propagation, and coalescence of microcracks. It is commonly accepted that the cracks frequently initiate from pre-existing interfacial flaws or micro-defects at the ITZ (the weak interface between the large aggregates and the mortar) and then propagate into the mortar matrix (Ref. Fig. 2.3) [8, 242–245]. In other cases, cracks can get arrested at the interface edge by a tougher cement paste. The activation and growth of cracks from pre-existing flaws under the compressive stress field are highly influenced by the bond strength and frictional properties of the ITZ, which ultimately affects the non-linear macroscopic response of the concrete. Under the applied external loading, the material is progressively weakened due to the diminishing number of interfacial bonds in the ITZ and coalescence of micro-cracks, which results in softening behaviour of the material.

To capture the material heterogeneity in concrete, several multiscale computational

models have been reported in the literature, including the lattice models [246–250], discrete element models [251–254], continuum models [129, 235, 237, 238], etc. Among these, the continuum models are typically used in finite element techniques. Here, the crack related discontinuities in the structure are modelled using two approaches: Implicit approach using damage models and explicit approach using interface elements.

In the *damage models* [235, 255, 256], the crack and other discontinuities in the structure are modelled within the continuum elements. To model the failure process, a damage variable is introduced that reduces the stiffness of the continuum elements. However, very small-sized elements are required to accurately model the ITZ that has very low thickness. The *interface element approach* [129, 236–238, 257] can explicitly model the fracture process and ITZ following non-linear (cohesive) traction-separation and friction laws. Usually, the interface elements are inserted between the continuum elements, due to which the fracture takes place only along the element interfaces. This method also requires very fine mesh to reduce mesh dependencies and accurately capture the interaction between constituents of the heterogeneous material.

2.3.5 Impact-induced fracture analysis

The impact-induced fracture analysis is critical in many fields concerning automotive, aerospace, structural and defence. The simulation of impact phenomena in composites and other heterogeneous materials like concrete has two components:

1. Simulation of contact, which describes the instantaneous interaction between the impacting bodies, local elastic/plastic deformation at the contact zone, dissipation of energy due to friction, etc.

2. Simulation of the vibration and stress wave propagation in the target structure that can cause failure in various modes, including matrix crushing and cracking, delamination produced by interlaminar stresses, fibre buckling and failure, etc.

In this section, we briefly review the existing literature for modelling contact-impact problem and fracture analysis in composites and other heterogeneous materials under impact loading.

Contact-impact modelling

The general formulation of contact impact problems in a multi-body system requires consideration contact detection mechanisms [258–262] and the non-smooth response of the contacting bodies [263]. Here, we briefly review the latter phenomenon.

During the impact event, the normal velocity component at the contact interface is discontinuous in time. In addition, the tangential velocity components along the interface are also discontinuous due to stick-slip behaviour encountered in Coulomb friction models. These characteristics cause significant difficulties in the convergence of the implicit schemes especially those based on Newton-based solvers since discontinuities are introduced in the Jacobian. Therefore, the benefit of using large time steps in the analysis (as allowed in the implicit methods due to their unconditional stability) cannot be effectively used in capturing the discontinuous response at the contact interface.

The explicit time integration schemes are more suitable for the contact-impact problems for the following reasons [263]:

1. The smaller time steps (though a stability requirement) reduce the ill-effects of the discontinuous response.

2. Each time step is computationally inexpensive as compared to implicit methods

3. It avoids the use of linearisation or a Newton solver; therefore, the associated ill-effects of discontinuities are also avoided.

The most commonly used methods to enforce the contact conditions in contact-impact problems include the Lagrange multiplier method, the penalty method, etc. The Lagrange multiplier method [227, 264], which enforces hard-contact on the contact interface,

induces artificial oscillations at the interface when the time step is reduced for a fixed mesh-size. Several approaches have been adopted to reduce the effect of artificial oscillations, including the mass redistribution techniques [265, 266], introducing artificial damping [267], etc.

A commonly used approach is to approximately satisfy the contact condition using the penalty formulation [263, 268, 269]. However, using a stiff penalty formulation may not be sufficient to avoid artificial oscillations, as it leads to an instantaneous change of velocity at the contact surface. On the other hand, a softer penalty induces error in the solution by increasing the penetration at the contact surface, though it reduces the artificial oscillations. Using non-linear penalty regularisation for impact simulations ensures a smooth transition from non-contact to contact state [270, 271] and alleviates the oscillations even at a high value of penalty stiffness. However, to obtain an accurate response, a high mesh density near the contact surface is generally required. With the aid of local mesh refinement and fast mesh-size transition near the contact interface, the problem can be effectively solved without much increase in computational effort.

Fracture modelling

The impact-induced response of a target structure depends on many factors [40–42]. First, it depends on the impact conditions such as low and high-velocity of the projectile. A high-velocity impact response is controlled by wave propagation and is independent of boundary conditions. On the other hand, a low-velocity impact response is highly influenced by boundary conditions. In addition, the impact response also depends on the structural configuration. E.g., in the case of sandwich panels, it depends not only on the mechanical properties of the sheet plate and core but also on the adhesive capacity of the skin-core interface. The typical damage modes that need to be assessed include delamination of the sandwich face-sheet and core crushing and cracking.

Within the framework of finite-element based methods, the cohesive interface-element

modelling has been extensively [43–47] used to capture the failure modes involving matrix cracking and delamination under impact loading. The development of cohesive element modelling has been explained in detail in Section 2.3.2. This method can also model the contact interaction during the fracture process using the penalty-stiffness based approach.

The explicit time-integration scheme is widely used by researchers for performing impact-induced fracture analysis [272–274]. This is due to the benefits associated with the scheme, such as no requirement of factorising a matrix, easy implementation using the element-by-element (EBE) technique that avoids the assembly of the global stiffness matrix, and support for large scale parallel simulation. Further, during crack-closure, the explicit time-integration scheme can efficiently handle dynamic contact conditions, as explained previously.

2.4 High performance computing

Many engineering applications stated in the previous sections require extensive three-dimensional modelling and sufficient resolution to accurately capture complex geometrical shapes and structural response. This requires handling large models with millions (or billions) of DOFs. Therefore, the development of a high-performance computational framework is necessary so that the simulations of such large-scale models can be performed in a feasible time.

High-performance computing (HPC) is carried out on a class of computers called supercomputers. In modern supercomputers, high-speed computation is primarily achieved through parallel processing, where several operations take place simultaneously on multiple processors. The supercomputers can be classified based on different memory architectures, which are reviewed in this section. Then, the basic functionalities of the MPI-based parallel programming model – commonly used to achieve large-scale paral-

lelism in distributed/hybrid memory architectures, are presented. Finally, we discuss an important computational approach – the domain decomposition (DD) method, which works on a "divide-and-conquer" strategy for obtaining parallel solutions on HPC platforms.

2.4.1 Memory architecture

The supercomputers are based on various architectural models such as shared, distributed, or hybrid memory systems. More recently, GPU accelerators are also being incorporated into the hybrid systems. Here, we briefly discuss these architectures and associated programming languages.

Shared-memory systems

In shared-memory systems, the memory can be accessed by all the processors as global address space leading to high-speed data sharing among the processors. However, these systems have poor scalability in terms of the maximum number of processors used.

OpenMP is a popular programming language used with shared-memory systems. It achieves parallelism through multi-threading, where several threads run concurrently on different cores/processors. OpenMP consists of compiler directives, library routines, and environment variables that allow easier transition from the serial/sequential computers. It has been successfully used to parallelise code in finite element analyses [275–277], molecular dynamics [278, 279], fluid dynamics [280, 281], and many other computational fields.

Distributed/Hybrid-memory systems

In distributed-memory systems, processors operate independently on their local memory within compute nodes connected over a high-speed communication network. These systems are highly scalable with respect to the number of processors and memory.

However, data communication between processors is complicated, and the performance of parallel programs often depends on the interconnect speed. Many modern HPC facilities adopt hybrid architecture, consisting of a cluster of stand-alone shared-memory systems (or compute nodes) connected over a high-speed network.

Most applications running on distributed/hybrid systems use a Message Passing Interface (MPI) based parallel programming model [282, 283]. MPI is a standardised and portable message-passing system that defines a set of library routines for writing parallel programs in C, C++, and FORTRAN. It has been widely utilised for the implementation of explicit as well as implicit schemes [57, 164, 175, 284, 285]. The basic communication primitives of MPI are discussed in Section 2.4.2.

In general, it is possible to use MPI efficiently on shared-memory systems and program them as distributed-memory systems. Recently, hybrid programming models have gained popularity for simulation on hybrid (distributed+shared) memory systems [286–290]. One common approach is to parallelise the original (serial) code for distributed-memory architectures using mesh partitioning strategy and MPI. Then, OpenMP directives are added to the code to use multiple threads (e.g. through loop-level implementation) within each MPI process running on the shared-memory system. However, modifying the existing code for hybrid programming models is usually cumbersome. Further, the hybrid parallel code may not always perform better than those based on a pure MPI program.

GPU based computing

In recent years, utilising the power of graphics processing units for general-purpose computations (GP-GPUs) has gained much popularity. The introduction of programming languages, such as CUDA [291], OpenCL [292], and OpenACC [293], has allowed researchers to exploit the parallel computing power of a CUDA-enabled graphics card. There are several instances where low-cost GPU based parallel computing has replaced

the traditional parallel computing methods based on multi-core systems [294].

In the field of FE analysis, the GPUs have been applied for the parallel computation of mesh generation [295], stiffness matrix assembly [296, 297], matrix-free methods [298], etc. The parallel processing architectures of GPUs have been exploited to perform the explicit and implicit FE analysis for the linear and non-linear dynamic problems such as building models analysis, elastic shell problems, etc. [299–302]. Further, the simulations of non-linear contact-impact problems, including sheet metal forming simulations and crash simulations [296, 303, 304], have been performed exploiting the fine-grained parallel computing power of GPUs.

Although GPUs can be shown to run algorithms anywhere from 10 to 100 or more times faster than CPUs, there are a few associated disadvantages. Programming for the GPU differs significantly from traditional CPUs. Incorporating GPU acceleration into pre-existing codes to achieve speed-ups is hardware-dependent and more complicated than moving from one CPU family to another. A major drawback is the slow data transfer rate between CPU and GPU devices, which can increase the analysis time significantly. Further, there are issues related to added expense in power consumption, heat production, etc., when the GPU hardware is incorporated into systems.

2.4.2 Message passing interface (MPI)

The MPI based parallel programming model is popularly used for performing large-scale simulations in distributed/hybrid-memory systems. This section presents a few functionalities and subprograms available in MPI (and relevant to this thesis). The details of all subprograms available in MPI are described in [282, 283, 305].

In MPI for Python, the base class of communicators (group of MPI processes) is defined as Comm. The size of the communicator (number of MPI processes) and MPI process rank can be obtained with methods Get_size and Get_rank, respectively. The basic communication primitives that we utilise are point-to-point and collective

communications.

Point-to-point communication

Point-to-point communication is used to send data or messages between two different MPI processes (one-to-one). One process performs a send operation while the other process performs a matching read. In this operation, MPI ensures that messages arrive at the receiving side intact and without errors. When send and receive operations do not match, deadlock can occur, which halts the entire program. This situation is undesirable, and care must be taken to avoid it. Point-to-point communications can be categorised as blocking and non-blocking.

In *blocking point-to-point communications*, the send/receive operations are performed in blocking mode – the control is not returned until the send/receive operations are finished. The communicator or Comm object provides several functions to perform these operations such as Send, Recv and Sendrecv. Both the general Python objects and raw memory buffers can be communicated.

The efficiency of the program can be significantly increased by overlapping the communication and computation. This is possible by utilising *non-blocking point-to-point communication*. The Comm object functions such as Isend and Irecv initiate a send and receive operation respectively and return a Request instance to uniquely identify the started operation. The completion of these operations can be managed using the Test, Wait, and Cancel methods of the Request class. The Request objects and associated memory buffers involve a low-level coordination. Therefore, care must be taken to avoid accessing these objects at the Python level while they are still involved in non-blocking message-passing operations as it may corrupt the communication process.

The non-blocking point-to-point communication can be used for synchronising the data at the interfacial nodes when performing matrix operations in the explicit and implicit time-integration schemes, as explained in Chapter 4.

Collective communication

In cases where communication is required among a group of MPI processes (a collective operation), the point-to-point communications are often tedious to use and less efficient. To handle this issue, the Comm object provides general collective communication functions such as Barrier, Bcast, Gather, Allgather, Scatter etc. which can perform collective operations of message passing among processes, namely one-to-all, all-to-one and all-to-all. These methods can communicate either general Python objects or raw memory buffers. There are functions supporting vector variants that can communicate the varying amount of data (only raw memory buffers supported) at each process, such as Gatherv, Allgatherv, Scatterv etc. Further, Comm object also provides functions supporting global reduction operations include Reduce, Allreduce, Scan, etc., where all the predefined (i.e., SUM, PROD, MAX, etc.) and user-defined reduction operations can be performed.

The collective communications such as global reductions have been utilised in the implementation of implicit time-integration schemes and iterative linear solvers such as PCG, which is presented in Chapter 5.

2.4.3 Domain decomposition methods

The domain decomposition (DD) method was originally proposed by Schwarz [306] in 1869 for solving partial differential equations. The idea follows the "divide-and-conquer" strategy, where a large domain is split into relatively smaller domains (called subdomains) that are solved individually, and then combined to provide a global solution.

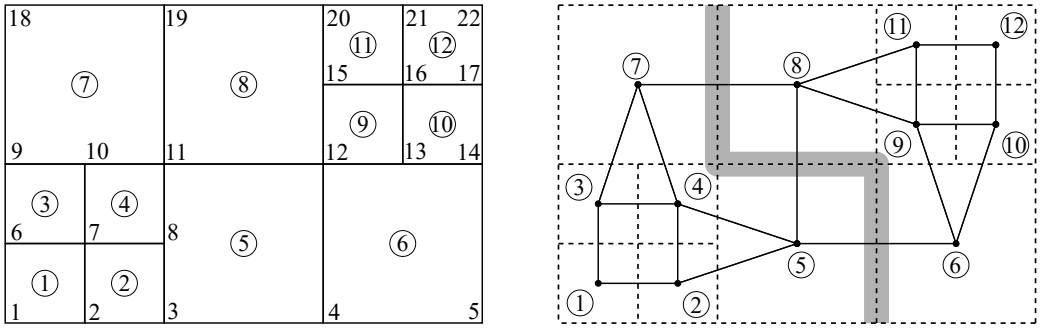
Since its inception, the DD methods have evolved into separate class of computational algorithm and have been utilised in a variety of applications including parallel implementation of direct solvers [307–309], iterative solvers and pre-conditioners [163, 310, 311], performing geometric operations [312–315] such as local mesh refinement, simplifying the mesh (using multiblock or patching grids), and moving compon-

ents such as sliding meshes, etc. Several forms of the DD formulations have been utilised by researchers as explained below.

In *Schur-substructuring methods* [316–320], the internal degrees of freedom are first eliminated independently on each processor (e.g. using direct solver). Then, the reduced system of equations having only interface degrees of freedom is solved using an appropriate iterative solver. The *Schwartz methods* [321–324] follows an iterative approach. The separate solutions are first obtained within the subdomains and at the subdomain interfaces. This is followed by iteratively balancing the forces acting across the subdomain interfaces, and obtaining the solutions (within the subdomain and at the interfaces) until the convergence is achieved.

The *subdomain-by-subdomain (SBS) method* was presented by Papadrakakis et al. [325–327] for iterative solvers. Here, the subdomains are first obtained using a mesh partitioning algorithm that performs multi-element-group partitioning of the entire domain. On each subdomain, the matrix operations and the pre-conditioning are locally performed. The data at the mesh-partition interface is synchronised using the inter-processor communication. In the SBS approach, the serial and parallel codes are very similar; therefore, the convergence is not affected by the parallelisation.

The SBS-like formulation has been utilised by many researchers for performing parallel simulations in a variety of problems [175, 328, 329] in finite element framework. To reduce the memory requirements, the element-by-element technique (Ref: Section 2.2.3) is commonly employed. A majority of the CPU time is spent on performing simultaneous local matrix operations on each mesh-partition while a very little time is spent in exchanging the data with the neighbouring partitions. This feature makes it extremely suitable for performing large-scale simulations in distributed/hybrid-memory systems using MPI based parallel programming model. However, the efficiency of the parallel implementation significantly depends on the quality of mesh-partitions. The following section elaborates on the mesh partitioning and communication strategies commonly



(a) A sample quadtree mesh

(b) Dual graph and its optimal partition: Node-cut represented by the shaded line

Figure 2.4: A quadtree mesh and its dual graph

followed in the framework of finite element based techniques.

Mesh-partitioning

The mesh-partitioning in the finite element framework involves splitting the finite element mesh into smaller parts. Two types of mesh partitioning strategies are primarily employed: node-cut and element-cut strategies [57]. The node-cut strategy is usually adopted as it avoids the duplication of elements, and thus, has a better performance than the element-cut strategy in general.

The quality of the partition is mainly controlled by two factors: *Load imbalance* and *data communication*. The load imbalancing happens when the individual workers are assigned mesh partitions with uneven computational loads (that is, an unequal number of elements). This leads to the workers with the lower computational loads waiting for the workers with higher computational loads to finish the tasks and thus, spending idle time. As the majority of the CPU time of an individual worker is governed by the number of elements assigned to it, the load imbalance directly influences the efficiency of the solver. Another factor that affects the efficiency is the data communication among the workers. The amount of data that needs to be exchanged among the neighbouring partitions is proportional to the number of nodes at the interfaces of the partitions and

thus, they should be kept minimum.

Many existing codes on mesh-partitioning work on the principle of graph partitioning where the finite element based mesh is first converted into a graph, and then divided into smaller parts. The mesh can be either converted into a dual graph where each element becomes a graph vertex or a nodal graph where each node becomes a graph vertex.

For the node-cut strategy, a dual graph is generated from the finite element mesh. Here, each element is represented by a vertex v in the graph, while each interface shared by two elements is represented by an edge e connecting two vertices. We denote the set of vertices as V and the set of edges as E. For illustration, a quadtree mesh in 2D and its dual graph $G = \{V, E\}$ are shown in Fig. 2.4. An optimal partitioning is obtained by cutting through nodes $\{\mathrm{n}_4, \mathrm{n}_{12}, \mathrm{n}_{11}, \mathrm{n}_{19}\}$ which divides the mesh into two parts with six elements each, keeping the number of interfacial nodes minimum.

In general, it is difficult to find the optimal partition, which is known as an NP-hard problem (non-deterministic polynomial-time hardness [330]). However, there are a few mesh partition packages available that can provide a reasonably good approximation of the optimal partition, such as MESHPART [331], METIS [332], JOSTLE [333], etc. The comparison between various mesh-partitioning methods has been explained in [334].

Among these packages, METIS is a popular choice among the researcher for obtaining high-quality mesh partitions. METIS utilises a multi-level k-way partitioning scheme and a graph representation of the element connectivity to minimise the domain boundaries and achieve uniform distribution of computational loads over each partition. It can assign non-uniform weights to each elements to accommodate different computational cost associated with different element types. It can also assign different weights to each mesh-partition, which is useful for running simulations in heterogeneous processing environment.

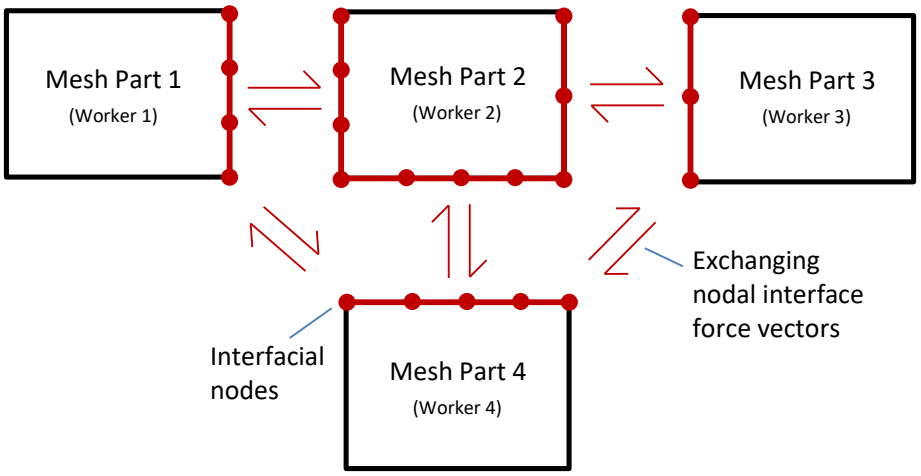


Figure 2.5: Direct communication of nodal interface force vectors between the workers

Data communication

Parallel processing using the mesh-partitioning strategy involves assigning the mesh-partitions to individual worker (or process) that carry out local computations and exchange information with the other workers as necessary. The most commonly used strategy for the information-exchange is the *direct communication* between the workers which share the common interfacial nodes as shown in Fig. 2.5. Such communication strategies have been reported by [335, 336]. The message size, which contains the interfacial nodal data, is generally smaller and therefore, is more suitable for machines with low overhead latency per message. For computer networks with large latencies, much larger messages are required to reduce the latency effect. For these networks, communication patterns like a *star-shaped configuration* of workers synchronised by a master processor are more appropriate [133].

2.5 Summary

The present chapter extensively reviews a wide range of topics pertaining to the development of high-performance computational framework for performing large-scale structural simulations concerning the impact-induced fracture analysis. It is summarised as follows.

- First, the computational modelling of structures is discussed with reference to the commonly used mesh generation and numerical techniques.

  Among the various mesh generation techniques, the hierarchical grid-based method is a promising technique that can be exploited for automating the high-quality mesh-generation process and speeding up computational simulations. However, as these meshes are based on quadtree/octree data structures, they are difficult to handle with the conventional numerical techniques such as FEM or XFEM. It is because of the issues related to hanging nodes between the adjacent cells creating compatibility problems and the unavailability of the shape functions for arbitrary polytope elements to handle the trimmed cells at the boundary.

  The hierarchical grid-based meshes can be efficiently handled by numerical techniques that provide shape functions for arbitrary polytope element. The scaled boundary finite element method (SBFEM) has such capabilities and therefore, is chosen in the present work. The SBFEM formulation can also eliminate the issue of hanging node in these meshes without any additional treatment.

  When the hierarchical grid-meshes containing quadtree/octree cells are used in a balanced form, they have a limited number of cell patterns. This feature can be exploited for solving large-scale linear structural problems in 3D by pre-computing the element matrices of these patterns and reducing the memory requirements associated with element matrices of individual elements. Although, the octree-based formulations in the framework of SBFEM has been extensively studied in the existing literature and applied to many engineering problems, its

applicability in modelling of large scale structures and high-performance computing remains to be established. Therefore, the same is explored in the present work using the octree-based pattern-by-pattern (PBP) approach.

- Next, the numerical solution procedure for structures subjected to dynamic/impact loading is reviewed. The direct-integration methods are commonly utilised and can be broadly categorised into two classes, explicit and implicit time-integration schemes.

  The explicit schemes are more suitable for solving wave propagation analysis. The most popularly used scheme is the central difference method (CDM). The CDM has high-accuracy, allows element level calculation using the element-by-element (EBE) technique and supports parallelisation.

  The implicit integration techniques are commonly used for solving inertial problems. Several schemes such as the HHT-α method, the WBZ-α method, etc., exist which are second-order accurate and provide controlled dissipation. The computational cost of the implicit methods is mainly dominated by the requirement of solving the linearised equilibrium equations at each time step, which can be effectively solved using iterative solvers such as the preconditioned conjugate gradient (PCG) algorithm. For large-scale problems, the memory requirements of matrix operations in PCG solvers increases significantly. It can be handled using the element-by-element (EBE) technique; however, there are several issues associated with the efficient implementation of this approach, as discussed in Section 2.2.3.

  In the present work, the efficacy of the octree-based pattern-by-pattern (PBP) approach is explored in the implementation of the explicit solvers such as CDM for solving wave-propagation problems. The effectiveness of the PBP approach is further explored for iterative/implicit solvers based on PCG/HHT-α methods for solving large-scale structural dynamics problems.

- Next, the computational modelling associated with impact-induced fracture analyses is explored. Such analyses are critical in many fields concerning automotive, aerospace, structural and defence.

  The simulation of high-velocity impact phenomena in composites and other heterogeneous materials like concrete has two components: Simulation of instantaneous contact interaction between the impacting bodies, and simulation of the stress wave propagation in the target structure leading to its failure in various modes.

  Among the various fracture models available, the cohesive interface-element modelling can effectively capture the failure modes involving matrix cracking and interface delamination under impact loading. This method can also model the contact interaction during the fracture process using the penalty-stiffness based approach. However, to accurately capture the non-linear interactions, a very fine mesh is required along the crack-path.

  In the present work, the hierarchical grid-based mesh containing octree-cells is combined with cohesive interface formulation to efficiently model the fracture process in composite structures. The applicability of the proposed PBP approach is explored for performing efficient matrix operations pertaining to cohesive and contact interactions. Further, the explicit time-integration scheme is employed as it is more suitable than the implicit methods to capture the impact-induced wave-propagation and the non-linear interactions associated with the fracture.

- Finally, various aspects related to high-performance computing are discussed: The memory architecture of the supercomputing facilities, MPI based parallel programming model, and computational strategies for parallel processing based on domain-decomposition (DD) methods.

  Several DD formulations exists in the literature. Among them, the subdomain-by-subdomain or the SBS-like approach has been utilised by many researchers for

performing parallel simulations. In this technique, the problem domain is divided into smaller domains (or subdomains) using a mesh partitioning algorithm and assigned to individual processors. On each subdomain, the matrix-vector operations are locally performed and the data at the subdomain interface is synchronised using the inter-processor communication.

The SBS-like approach is very suitable for performing large-scale simulations in distributed/hybrid-memory systems using MPI based parallel programming model and is utilised in the present work. As the quality of mesh-partitions significantly affects the efficiency of the parallel implementation, the existing graph-partitioning packages, such as METIS and MESHPART, have been utilised for obtaining high-quality mesh partitions. Further, the direct communication strategy is adopted in the current work.

Chapter 3

Scaled boundary finite element method

The scaled boundary finite element method (SBFEM) is a semi-analytical method that was developed by Song and Wolf [13] in the field of computational mechanics. The fundamentals of SBFEM is covered in detail by Song [1]. This chapter is organised as follows.

In Section 3.1, the formulation for two dimensional elastostatics and elastodynamics in scaled boundary coordinates is presented. In Section 3.2, the formulation is extended to three dimensions. General summary is given in Section 3.3.

3.1 Formulation of SBFEM in two dimensions

In this section, the basic formulations of the SBFEM for 2D elastostatics and elastodynamics are derived using the principle of virtual work.

3.1.1 Geometric modelling in scaled boundary coordinates

In the framework of the SBFEM, a problem domain is subdivided into small-sized elements or s-elements. Each element is discretised by line elements along its boundary, as shown in Fig. 3.1. There is no restriction on the number of nodes and edges in each element. The only requirement is to have a scaling centre from which the whole

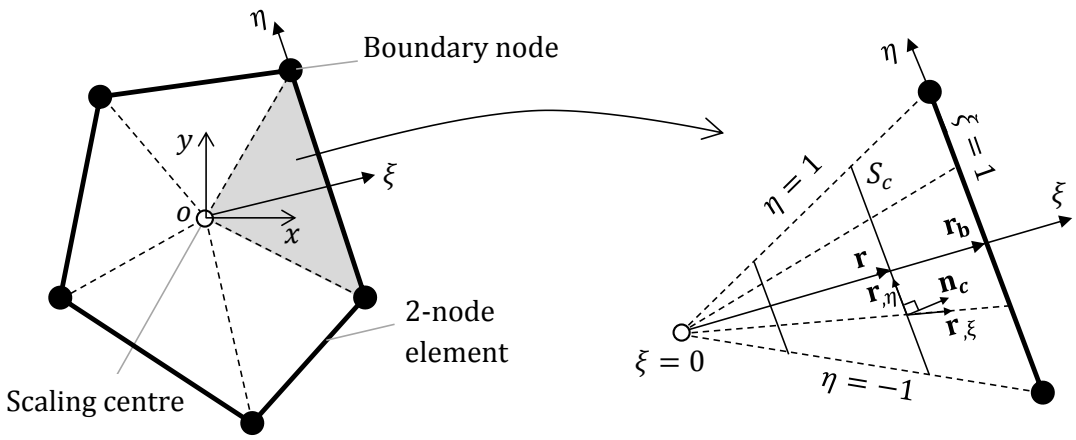


Figure 3.1: An s-element in 2D

boundary of the element is directly visible. The SBFEM formulation uses a normalised local coordinate system called the scaled boundary coordinate system, with ξ and η as the radial and circumferential coordinates respectively.

A sector of the domain is generated by scaling the line element from the boundary at $\xi = 1$ to the scaling centre at $\xi = 0$, as shown by the grey area in Fig. 3.1. Without the loss of generality, we can define a position vector $\mathbf{r}$ within the sector with respect to the origin at the scaling centre as

$$\mathbf{r} = x\mathbf{i} + y\mathbf{j}, \tag{3.1}$$

$$\mathbf{r} = \xi\mathbf{r}_b = \xi x_b\mathbf{i} + \xi y_b\mathbf{j}, \tag{3.2}$$

where, $\mathbf{i}$ and $\mathbf{j}$ are the unit vectors of the Cartesian coordinate system, and x_b and y_b are the coordinates along the line element on the boundary . The coordinates (x, y) of the position vector $\mathbf{r}$ is given by

$$x(\xi, \eta) = \xi x_b = \xi\mathbf{N}(\eta)\mathbf{x}, \tag{3.3}$$

$$y(\xi, \eta) = \xi y_b = \xi\mathbf{N}(\eta)\mathbf{y}, \tag{3.4}$$

where, $\mathbf{x}$ and $\mathbf{y}$ are the coordinates of nodes of the 1D line element on the boundary and $\mathbf{N}(\eta)$ is the shape function of the line element. For a two node line element, $\mathbf{N}(\eta)$

is expressed as

$$\mathbf{N}(\eta) = [N_1(\eta), N_2(\eta)] = \left[\frac{1}{2}(1-\eta), \frac{1}{2}(1+\eta)\right] \tag{3.5}$$

Using the standard isoparametric mapping technique, the local scaled boundary coordinates (ξ, η) can be transformed to the Cartesian coordinates (x, y) as

$$\left\{ \begin{array}{c} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \end{array} \right\} = \mathbf{J}(\xi, \eta) \left\{ \begin{array}{c} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{array} \right\}, \tag{3.6}$$

where, $\mathbf{J}(\xi, \eta)$ is the Jacobian matrix defined as

$$\mathbf{J}(\xi, \eta) = \begin{bmatrix} x_{,\xi} & y_{,\xi} \\ x_{,\eta} & y_{,\eta} \end{bmatrix}. \tag{3.7}$$

The partial derivatives of the Jacobian matrix can be obtained as

$$x_{,\xi} = x_b = \mathbf{N}(\eta)\mathbf{x}, \tag{3.8a}$$

$$x_{,\eta} = \xi x_{b,\eta} = \xi \mathbf{N}(\eta)_{,\eta}\mathbf{x}, \tag{3.8b}$$

$$y_{,\xi} = y_b = \mathbf{N}(\eta)\mathbf{y}, \tag{3.8c}$$

$$y_{,\eta} = \xi y_{b,\eta} = \xi \mathbf{N}(\eta)_{,\eta}\mathbf{y}. \tag{3.8d}$$

Then, Eq. (3.7) can be rewritten as

$$\mathbf{J}(\xi, \eta) = \mathrm{diag}(1, \xi)\mathbf{J}_b, \tag{3.9}$$

where, $\mathbf{J}_b$ is the Jacobian matrix at the boundary of the s-element ($\xi = 1$) and given as

$$\mathbf{J}_b = \begin{bmatrix} x_b & y_b \\ x_{b,\eta} & y_{b,\eta} \end{bmatrix}. \tag{3.10}$$

From Eqs. (3.6) and (3.9), the inverse of the coordinate transformation is obtained as

$$\left\{ \begin{array}{c} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{array} \right\} = \mathbf{J}_b^{-1} \left\{ \begin{array}{c} \frac{\partial}{\partial \xi} \\ \frac{1}{\xi}\frac{\partial}{\partial \eta} \end{array} \right\}, \tag{3.11}$$

where, $\mathbf{J}_b^{-1}$is the inverse of the Jacobian matrix at the boundary and given as

$$\mathbf{J}_b^{-1} = \frac{1}{|\mathbf{J}_b|} \begin{bmatrix} y_{b,\eta} & -y_b \\ -x_{b,\eta} & x_b \end{bmatrix}. \tag{3.12}$$

Substituting Eq. (3.12) into Eq. (3.11), the following coordinate transformation operator is obtained.

$$\begin{Bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{Bmatrix} = \frac{1}{|\mathbf{J}_b|} \begin{Bmatrix} y_{b,\eta} \\ -x_{b,\eta} \end{Bmatrix} \frac{\partial}{\partial \xi} + \frac{1}{|\mathbf{J}_b|} \begin{Bmatrix} -y_b \\ x_b \end{Bmatrix} \frac{1}{\xi} \frac{\partial}{\partial \eta}. \tag{3.13}$$

The derivatives of the position vector $\mathbf{r}$ defined in Eq. (3.2) with respect to η and ξ are given by

$$\mathbf{r}_{,\xi} = x_b \mathbf{i} + y_b \mathbf{j}, \tag{3.14a}$$

$$\mathbf{r}_{,\eta} = \xi x_{b,\eta} \mathbf{i} + \xi y_{b,\eta} \mathbf{j}. \tag{3.14b}$$

We express the infinitesimal length $\mathrm{d}S_c$ on the surface at a constant radial coordinate ξ using the tangential vector $\mathbf{r}_{,\eta}$ as

$$\mathrm{d}S_c = |\mathbf{r}_{,\eta}| \, \mathrm{d}\eta = \xi \, |\mathbf{r}_{b,\eta}| \, \mathrm{d}\eta. \tag{3.15}$$

The infinitesimal change in the volume $\mathrm{d}V$ of the s-element, for infinitesimal change $(d\xi, \mathrm{d}\eta)$ in the scaled boundary coordinates, can be obtained as the magnitude of the cross product of the infinitesimal increments in the tangential vectors

$$\mathrm{d}V = |(\mathbf{r}_{,\xi} \mathrm{d}\xi) \times (\mathbf{r}_{,\eta} \mathrm{d}\eta)| = \xi \, |\mathbf{J}_b| \, \mathrm{d}\xi \mathrm{d}\eta. \tag{3.16}$$

3.1.2 Formulation of linear elasticity in scaled boundary coordinates

In a two dimensional Cartesian coordinate system, the displacement field $\tilde{\mathbf{u}} = \tilde{\mathbf{u}}(x, y)$ can be written as

$$\tilde{\mathbf{u}} = [u_x, u_y]^{\mathrm{T}}. \tag{3.17}$$

For a linear elastic material, the strain field $\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}(x, y)$ is calculated as

$$\boldsymbol{\varepsilon} = [\varepsilon_x, \varepsilon_y, \gamma_{xy}]^{\mathrm{T}} = \mathbf{L}\tilde{\mathbf{u}}, \tag{3.18}$$

where, $\mathbf{L}$ is the differential operator and given by

$$\mathbf{L} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix}. \tag{3.19}$$

The stress field $\boldsymbol{\sigma} = \boldsymbol{\sigma}(x, y)$ is related to the strain field $\boldsymbol{\varepsilon}$ using a constitutive relationship given by

$$\boldsymbol{\sigma} = [\sigma_x, \sigma_y, \tau_{xy}]^{\mathrm{T}} = \mathbf{D}\boldsymbol{\varepsilon}, \tag{3.20}$$

where, $\mathbf{D}$ is the elasticity matrix. Under equilibrium conditions with zero body forces, the governing differential equation for two dimensional elastostatics is given as,

$$\mathbf{L}^{\mathrm{T}}\boldsymbol{\sigma} = 0. \tag{3.21}$$

Using Eq. (3.13), Eq. (3.19) can be expressed in the scaled boundary coordinate system as

$$\mathbf{L} = \mathbf{b}_1(\eta)\frac{\partial}{\partial \xi} + \frac{1}{\xi}\mathbf{b}_2(\eta)\frac{\partial}{\partial \eta}, \tag{3.22}$$

where, $\mathbf{b}_1(\eta)$ and $\mathbf{b}_2(\eta)$ are defined as

$$\mathbf{b}_1(\eta) = \frac{1}{|\mathbf{J}_b|}\begin{bmatrix} y_{b,\eta} & 0 \\ 0 & -x_{b,\eta} \\ -x_{b,\eta} & y_{b,\eta} \end{bmatrix}, \tag{3.23a}$$

$$\mathbf{b}_2(\eta) = \frac{1}{|\mathbf{J}_b|}\begin{bmatrix} -y_b & 0 \\ 0 & x_b \\ x_b & -y_b \end{bmatrix}. \tag{3.23b}$$

Here, the matrices $\mathbf{b}_1(\eta)$ and $\mathbf{b}_2(\eta)$ only depend on the geometry of the element and are independent of the radial coordinate ξ.

We define a nodal displacement function $\mathbf{u}(\xi)$ in the s-element along the lines passing

through the scaling centre and nodes on the boundary. Following the standard isoparametric formulation of the FEM, displacement field within the element is approximately defined as

$$\tilde{\mathbf{u}}(\xi,\eta) = \mathbf{N}_u(\eta)\mathbf{u}(\xi), \tag{3.24}$$

where, the interpolation function $\mathbf{N}_u(\eta)$ can be given as

$$\mathbf{N}_u(\eta) = [N_1(\eta)\mathbf{I},\ N_2(\eta)\mathbf{I}, \ldots, N_n(\eta)\mathbf{I}]. \tag{3.25}$$

Here, $\mathbf{I}$ is the 2×2 identity matrix and n the number of nodes of the line element at the boundary of the s-element.

From Eqs. (3.18), (3.22) and (3.24), the strain field $\boldsymbol{\varepsilon}(\xi,\eta)$ can be related to the displacement field by

$$\boldsymbol{\varepsilon}(\xi,\eta) = \mathbf{B}_1(\eta)\mathbf{u}(\xi)_{,\xi} + \frac{1}{\xi}\mathbf{B}_2(\eta)\mathbf{u}(\xi), \tag{3.26}$$

where, $\mathbf{B}_1(\eta)$ and $\mathbf{B}_2(\eta)$ are defined as

$$\mathbf{B}_1(\eta) = \mathbf{b}_1(\eta)\mathbf{N}_u(\eta), \tag{3.27a}$$

$$\mathbf{B}_2(\eta) = \mathbf{b}_2(\eta)\mathbf{N}_u(\eta)_{,\eta}. \tag{3.27b}$$

Using Eqs. (3.20) and (3.26), the stress field $\boldsymbol{\sigma}(\xi,\eta)$ can be obtained as

$$\boldsymbol{\sigma}(\xi,\eta) = \mathbf{D}(\mathbf{B}_1(\eta)\mathbf{u}(\xi)_{,\xi} + \frac{1}{\xi}\mathbf{B}_2(\eta)\mathbf{u}(\xi)). \tag{3.28}$$

We now derive the expression for nodal force function $\mathbf{q}(\xi)$. The traction $\mathbf{t}$ at the surface of the s-element with a constant ξ can be expressed as [1],

$$\mathbf{t} = \frac{1}{|\mathbf{r}_{b,\eta}|}\begin{bmatrix} y_{b,\eta} & 0 & -x_{b,\eta} \\ 0 & -x_{b,\eta} & y_{b,\eta} \end{bmatrix}\boldsymbol{\sigma}. \tag{3.29}$$

where $|\mathbf{r}_{b,\eta}|$ is the magnitude of the tangential vector $\mathbf{r}_{,\eta}$ obtained from Eq. (3.14b) at the boundary ($\xi = 1$) and is given by

$$|\mathbf{r}_{b,\eta}| = \sqrt{(x_{b,\eta})^2 + (y_{b,\eta})^2}. \tag{3.30}$$

Using Eq. (3.23a), we express Eq. (3.29) as

$$\mathbf{t} = \frac{|\mathbf{J}_b|}{|\mathbf{r}_{b,\eta}|}\mathbf{b}_1^{\mathrm{T}}(\eta)\boldsymbol{\sigma}. \tag{3.31}$$

Along the radial direction in the s-element, a nodal force functions $\mathbf{q}(\xi)$ is introduced which is equal to the resultant of the surface traction $\mathbf{t}$. From energy equivalence, the following equation is obtained:

$$\mathbf{u}^{\mathrm{T}}(\xi)\mathbf{q}(\xi) = \sum_e \int_{S_c} \tilde{\mathbf{u}}^{\mathrm{T}}(\xi,\eta)\mathbf{t}\mathrm{d}S_c. \tag{3.32}$$

Here, the integration is done for each sector corresponding to the line element e in the s-element. Substituting the displacement field from Eq. (3.24) into Eq. (3.32), the nodal force function is obtained as

$$\mathbf{q}(\xi) = \sum_e \int_{S_c} \mathbf{N}_u^{\mathrm{T}}(\eta)\mathbf{t}\mathrm{d}S_c. \tag{3.33}$$

Substituting Eqs. (3.15) and (3.31) into Eq. (3.33), we obtain

$$\mathbf{q}(\xi) = \sum_e \int_{-1}^{+1} \mathbf{N}_u^{\mathrm{T}}(\eta)\mathbf{b}_1^{\mathrm{T}}(\eta)\boldsymbol{\sigma}\xi\,|\mathbf{J}_b|\,\mathrm{d}\eta. \tag{3.34}$$

Putting Eq. (3.27a) into Eq. (3.34), the nodal force function can be expressed as

$$\mathbf{q}(\xi) = \sum_e \xi \int_{-1}^{+1} \mathbf{B}_1^{\mathrm{T}}(\eta)\boldsymbol{\sigma}\,|\mathbf{J}_b|\,\mathrm{d}\eta. \tag{3.35}$$

3.1.3 Derivation of the SBFE formulation for elastostatics

In this section, we derive the scaled boundary finite element (SBFE) formulation using the virtual work principle as shown by Deeks and Wolf [120]. From Eq. (3.24), a virtual displacement field $\delta\tilde{\mathbf{u}}(\xi,\eta)$ can be expressed as

$$\delta\tilde{\mathbf{u}}(\xi,\eta) = \mathbf{N}_u(\eta)\delta\mathbf{u}(\xi). \tag{3.36}$$

Similarly, from Eq. (3.26), a virtual strain field $\delta\boldsymbol{\varepsilon}(\xi,\eta)$ can be obtained as

$$\delta\boldsymbol{\varepsilon}(\xi,\eta) = \mathbf{B}_1(\eta)\delta\mathbf{u}(\xi)_{,\xi} + \frac{1}{\xi}\mathbf{B}_2(\eta)\delta\mathbf{u}(\xi). \tag{3.37}$$

Applying the principle of virtual work in elastostatics, we obtain

$$\delta U_\varepsilon - \delta W = 0, \tag{3.38}$$

where, δU_ε denotes the virtual increment in strain energy and δW is the virtual work done by the external force. These are given by

$$\delta U_\varepsilon = \int_V \delta \boldsymbol{\varepsilon}^{\mathrm{T}} \boldsymbol{\sigma} \mathrm{d}V, \tag{3.39}$$

$$\delta W = \delta \mathbf{u}_b^{\mathrm{T}} \mathbf{f}_b, \tag{3.40}$$

where,

$$\delta \mathbf{u}_b = \delta \mathbf{u}(\xi = 1). \tag{3.41}$$

Here, $\delta \mathbf{u}_b$ represents the virtual displacements at the boundary and $\mathbf{f}_b$ denotes the external force vector. Substituting Eqs. (3.16) and (3.37) into Eq. (3.39), we obtain

$$\begin{aligned} \delta U_\varepsilon = & \underbrace{\sum_e \int_0^1 \delta \mathbf{u}^{\mathrm{T}}(\xi)_{,\xi} \xi \int_{-1}^{+1} \mathbf{B}_1^{\mathrm{T}}(\eta) \boldsymbol{\sigma} \left|\mathbf{J}_b\right| \mathrm{d}\xi \mathrm{d}\eta}_{\delta U_{\varepsilon 1}} \\ & + \underbrace{\sum_e \int_0^1 \delta \mathbf{u}^{\mathrm{T}}(\xi) \int_{-1}^{+1} \mathbf{B}_2^{\mathrm{T}}(\eta) \boldsymbol{\sigma} \left|\mathbf{J}_b\right| \mathrm{d}\xi \mathrm{d}\eta}_{\delta U_{\varepsilon 2}}, \end{aligned} \tag{3.42}$$

where, the integration is performed for each sector corresponding to the line element e in the s-element. Substituting Eq. (3.35) in $\delta U_{\varepsilon 1}$, we get

$$\delta U_{\varepsilon 1} = \int_0^1 \delta \mathbf{u}^{\mathrm{T}}(\xi)_{,\xi} \mathbf{q}(\xi) \mathrm{d}\xi. \tag{3.43}$$

Applying integration by parts with respect to ξ results in

$$\delta U_{\varepsilon 1} = \delta \mathbf{u}^{\mathrm{T}}(\xi) \mathbf{q}(\xi) \mid_{\xi=1} - \int_0^1 \delta \mathbf{u}^{\mathrm{T}}(\xi) \mathbf{q}(\xi)_{,\xi} \mathrm{d}\xi. \tag{3.44}$$

From Eq. (3.41), $\delta U_{\varepsilon 1}$ can be rewritten as

$$\delta U_{\varepsilon 1} = \delta \mathbf{u}_b^{\mathrm{T}} \mathbf{q}(\xi = 1) - \int_0^1 \delta \mathbf{u}^{\mathrm{T}}(\xi) \mathbf{q}(\xi)_{,\xi} \mathrm{d}\xi. \tag{3.45}$$

Substituting Eq. (3.45) back into Eq. (3.42), we get

$$\delta U_\varepsilon = \delta \mathbf{u}_b^{\mathrm{T}} \mathbf{q}(\xi = 1) - \int_0^1 \delta \mathbf{u}^{\mathrm{T}}(\xi) \left(\mathbf{q}(\xi)_{,\xi} - \sum_e \int_{-1}^{+1} \mathbf{B}_2^{\mathrm{T}}(\eta) \boldsymbol{\sigma} \left| \mathbf{J}_b \right| \mathrm{d}\eta \right) \mathrm{d}\xi. \tag{3.46}$$

Putting Eq. (3.46) and Eq. (3.40) into Eq. (3.38), we obtain the following equation

$$\delta \mathbf{u}_b^{\mathrm{T}} (\mathbf{q}(\xi = 1) - \mathbf{f}_b) - \int_0^1 \delta \mathbf{u}^{\mathrm{T}}(\xi) \left(\mathbf{q}(\xi)_{,\xi} - \sum_e \int_{-1}^{+1} \mathbf{B}_2^{\mathrm{T}}(\eta) \boldsymbol{\sigma} \left| \mathbf{J}_b \right| \mathrm{d}\eta \right) \mathrm{d}\xi = 0. \tag{3.47}$$

Considering that Eq. (3.47) holds true for any arbitrary selection of virtual displacement within the s-element with $0 \leq \xi \leq 1$ and on the boundary with $\xi = 1$, we get the following equations of elastostatics in scaled boundary coordinates:

$$\mathbf{f}_b = \mathbf{q}(\xi = 1), \tag{3.48}$$

$$\mathbf{q}(\xi)_{,\xi} = \sum_e \int_{-1}^{+1} \mathbf{B}_2^{\mathrm{T}}(\eta) \boldsymbol{\sigma} \left| \mathbf{J}_b \right| \mathrm{d}\eta. \tag{3.49}$$

Substituting Eq. (3.28) into Eqs. (3.35) and (3.49), we obtain

$$\mathbf{q}(\xi) = \sum_e \xi \int_{-1}^{+1} \mathbf{B}_1^{\mathrm{T}}(\eta) \mathbf{D} \left(\mathbf{B}_1(\eta) \mathbf{u}(\xi)_{,\xi} + \frac{1}{\xi} \mathbf{B}_2(\eta) \mathbf{u}(\xi) \right) \left| \mathbf{J}_b \right| \mathrm{d}\eta, \tag{3.50}$$

and

$$\mathbf{q}(\xi)_{,\xi} = \sum_e \int_{-1}^{+1} \mathbf{B}_2^{\mathrm{T}}(\eta) \mathbf{D} \left(\mathbf{B}_1(\eta) \mathbf{u}(\xi)_{,\xi} + \frac{1}{\xi} \mathbf{B}_2(\eta) \mathbf{u}(\xi) \right) \left| \mathbf{J}_b \right| \mathrm{d}\eta, \tag{3.51}$$

respectively. The following coefficient matrices $\mathbf{E}_0$, $\mathbf{E}_1$ and $\mathbf{E}_2$ are introduced in order to simplify the expression in Eqs. (3.50) and (3.51),

$$\mathbf{E}_0 = \sum_e \int_{-1}^{+1} \mathbf{B}_1^{\mathrm{T}}(\eta) \mathbf{D} \mathbf{B}_1(\eta) \left| \mathbf{J}_b \right| \mathrm{d}\eta, \tag{3.52a}$$

$$\mathbf{E}_1 = \sum_e \int_{-1}^{+1} \mathbf{B}_2^{\mathrm{T}}(\eta) \mathbf{D} \mathbf{B}_1(\eta) \left| \mathbf{J}_b \right| \mathrm{d}\eta, \tag{3.52b}$$

$$\mathbf{E}_2 = \sum_e \int_{-1}^{+1} \mathbf{B}_2^{\mathrm{T}}(\eta) \mathbf{D} \mathbf{B}_2(\eta) \left| \mathbf{J}_b \right| \mathrm{d}\eta. \tag{3.52c}$$

Here, the summation is performed in the standard finite element way for each sector corresponding to the line element e in the s-element. The simplified Eqs. (3.50) and

(3.51) can now be expressed as

$$\mathbf{q}(\xi) = \mathbf{E}_0 \xi \mathbf{u}(\xi)_{,\xi} + \mathbf{E}_1^{\mathrm{T}} \mathbf{u}(\xi), \tag{3.53}$$

$$\xi \mathbf{q}(\xi)_{,\xi} = \mathbf{E}_1 \xi \mathbf{u}(\xi)_{,\xi} + \mathbf{E}_2 \mathbf{u}(\xi). \tag{3.54}$$

From Eq. (3.53) and Eq. (3.54), we obtain the following scaled boundary finite element equation in displacements

$$\mathbf{E}_0 \xi^2 \mathbf{u}(\xi)_{,\xi\xi} + (\mathbf{E}_0 + \mathbf{E}_1^{\mathrm{T}} - \mathbf{E}_1) \xi \mathbf{u}(\xi)_{,\xi} - \mathbf{E}_2 \mathbf{u}(\xi) = 0. \tag{3.55}$$

Substituting Eq. (3.53) into Eq. (3.48), the boundary nodal force vector $\mathbf{f}_b$ can be expressed as

$$\mathbf{f}_b = \mathbf{E}_0 \xi \mathbf{u}_{b,\xi} + \mathbf{E}_1^{\mathrm{T}} \mathbf{u}_b, \tag{3.56}$$

where, $\mathbf{u}_b = \mathbf{u}(\xi = 1)$ is the boundary nodal displacement vector.

3.1.4 Solution of the SBFE equation

In this section, the second order homogeneous ordinary differential equation (ODE) obtained in Eq. (3.55) is solved using the method of eigenvalue decomposition. One of the solution procedures for solving the second order ODEs is to convert it into a system of first order differential equations. Rearranging Eq. (3.53), we obtain

$$\xi \mathbf{u}(\xi)_{,\xi} = \mathbf{E}_0^{-1} \mathbf{q}(\xi) - \mathbf{E}_0^{-1} \mathbf{E}_1^{\mathrm{T}} \mathbf{u}(\xi). \tag{3.57}$$

Substituting Eq. (3.57) into Eq. (3.55) results in

$$\xi \mathbf{q}(\xi)_{,\xi} - \mathbf{E}_1 \mathbf{E}_0^{-1} \mathbf{q}(\xi) + \mathbf{E}_1 \mathbf{E}_0^{-1} \mathbf{E}_1^{\mathrm{T}} \mathbf{u}(\xi) + \mathbf{E}_2 \mathbf{u}(\xi) = 0. \tag{3.58}$$

Let, n denote the total number of degrees of freedom at the boundary of the s-element. A vector $\mathbf{X}(\xi)$ with n nodal displacement functions and n nodal force functions is introduced and is given by

$$\mathbf{X}(\xi) = \left\{ \begin{array}{c} \mathbf{u}(\xi) \\ \mathbf{q}(\xi) \end{array} \right\}. \tag{3.59}$$

Then, Eqs. (3.57) and (3.58) can be reformulated as a system of $2n$ equations given by

$$\xi \mathbf{X}(\xi)_{,\xi} = \mathbf{Z}_p \mathbf{X}(\xi), \tag{3.60}$$

where, $\mathbf{Z}_p$ is a Hamiltonian matrix expressed as

$$\mathbf{Z}_p = \begin{bmatrix} -\mathbf{E}_0^{-1}\mathbf{E}_1^{\mathrm{T}} & \mathbf{E}_0^{-1} \\ \mathbf{E}_2 - \mathbf{E}_1\mathbf{E}_0^{-1}\mathbf{E}_1^{\mathrm{T}} & \mathbf{E}_1\mathbf{E}_0^{-1} \end{bmatrix}. \tag{3.61}$$

A well known procedure to solve Eq. (3.60) is to assume that the solution is in the power function form given by

$$\mathbf{X}(\xi) = \xi^{\lambda}\boldsymbol{\phi}. \tag{3.62}$$

Substituting Eq. (3.62) into Eq. (3.60), we obtain an eigenvalue problem given by

$$\mathbf{Z}_p\boldsymbol{\phi} = \lambda\boldsymbol{\phi}, \tag{3.63}$$

with λ and $\boldsymbol{\phi}$ being the eigenvalue and corresponding eigenvector of the Hamiltonian matrix $\mathbf{Z}_p$. Considering all the eigenvalues and eigenvectors in matrix form, Eq. (3.63) can be written as

$$\mathbf{Z}_p\boldsymbol{\Phi} = \boldsymbol{\Phi}\mathrm{diag}(\lambda), \tag{3.64}$$

where, the eigenvector matrix $\boldsymbol{\Phi}$ is given by

$$\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_{2n}]. \tag{3.65}$$

The corresponding eigenvalues in diagonal form is written as

$$\mathrm{diag}(\lambda) = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{2n}). \tag{3.66}$$

The general solution of the variable $\mathbf{X}(\xi)$ can be expressed as a linear combination of $2n$ independent solutions and is given by

$$\mathbf{X}(\xi) = \sum_{i=1}^{2n} c_i \xi^{\lambda_i}\boldsymbol{\phi}_i \tag{3.67}$$

where, c_i denote the integration constant. Each eigenvector $\boldsymbol{\phi}_i$ can be divided into two sub-vectors of the size n and is written as

$$\boldsymbol{\phi}_i = \left\{ \begin{array}{c} \boldsymbol{\phi}_i^u \\ \\ \boldsymbol{\phi}_i^q \end{array} \right\}. \tag{3.68}$$

Therefore, the solutions of the nodal displacement and force functions can be obtained as

$$\mathbf{u}(\xi) = \sum_{i=1}^{2n} c_i \xi^{\lambda_i} \boldsymbol{\phi}_i^u, \tag{3.69a}$$

$$\mathbf{q}(\xi) = \sum_{i=1}^{2n} c_i \xi^{\lambda_i} \boldsymbol{\phi}_i^q. \tag{3.69b}$$

For a bounded s-element, when the eigenvalues are sorted in descending order of their real parts, only the first n eigenvalues tend to produces finite displacements. Therefore, imposing this condition, the solutions are rewritten as

$$\mathbf{u}(\xi) = \sum_{i=1}^{n} c_i \xi^{\lambda_i} \boldsymbol{\phi}_i^u, \tag{3.70a}$$

$$\mathbf{q}(\xi) = \sum_{i=1}^{n} c_i \xi^{\lambda_i} \boldsymbol{\phi}_i^q. \tag{3.70b}$$

The solutions can be expressed in matrix form as

$$\mathbf{u}(\xi) = \boldsymbol{\Phi}_b^u \xi^{\langle \lambda_b \rangle} \mathbf{c}, \tag{3.71a}$$

$$\mathbf{q}(\xi) = \boldsymbol{\Phi}_b^q \xi^{\langle \lambda_b \rangle} \mathbf{c}, \tag{3.71b}$$

where, $\boldsymbol{\Phi}_b^u$ and $\boldsymbol{\Phi}_b^q$ are sub-matrices of the eigenvector matrix corresponding to the first n eigenvalues and given as

$$\left[\begin{array}{c} \boldsymbol{\Phi}_b^u \\ \\ \boldsymbol{\Phi}_b^q \end{array} \right] = \left[\begin{array}{cccc} \boldsymbol{\phi}_1^u & \boldsymbol{\phi}_2^u & .... & \boldsymbol{\phi}_n^u \\ \\ \boldsymbol{\phi}_1^q & \boldsymbol{\phi}_2^q & .... & \boldsymbol{\phi}_n^q \end{array} \right]. \tag{3.72}$$

$\langle \lambda_b \rangle$ is a diagonal matrix containing the eigenvalues, given by

$$\langle \lambda_b \rangle = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n). \tag{3.73}$$

$\mathbf{c}$ is a vector containing the integration constants and expressed as

$$\mathbf{c} = [c_1, c_2, \ldots, c_n]^{\mathrm{T}}. \tag{3.74}$$

The integration constant vector $\mathbf{c}$ can be obtained by setting $\xi = 1$ in Eq. (3.71a) as

$$\mathbf{c} = [\mathbf{\Phi}_b^u]^{-1} \mathbf{u}_b, \tag{3.75}$$

where, $\mathbf{u}_b = \mathbf{u}(\xi = 1)$. Substituting the integration constant vector $\mathbf{c}$ from Eq. (3.75) into Eq. (3.71b) at the boundary of the s-element ($\xi = 1$), we obtain

$$\mathbf{q}(\xi = 1) = \mathbf{\Phi}_b^q [\mathbf{\Phi}_b^u]^{-1} \mathbf{u}_b. \tag{3.76}$$

The boundary nodal forces are expressed as $\mathbf{f}_b = \mathbf{q}(\xi = 1)$. Therefore, the nodal force-displacement relationship can be re-written as

$$\mathbf{f}_b = \mathbf{K}^e \mathbf{u}_b, \tag{3.77}$$

where, $\mathbf{K}^e$ is the element stiffness matrix and given by

$$\mathbf{K}^e = \mathbf{\Phi}_b^q [\mathbf{\Phi}_b^u]^{-1}. \tag{3.78}$$

The global stiffness matrix of the problem domain can be assembled from the element stiffness matrices in the conventional finite element way. The global boundary displacement vector can then be found from the equilibrium equation after imposing the displacement and external loading conditions. Next, for each s-element, the integration constant vector $\mathbf{c}$ can be obtained from Eq. (3.75). Following this, the displacement field in each sector of the s-element is obtained from Eqs. (3.24) and (3.70a) as

$$\tilde{\mathbf{u}}(\xi, \eta) = \mathbf{N}_u(\eta)\mathbf{u}(\xi) = \mathbf{N}_u(\eta) \sum_{i=1}^{n} c_i \xi^{\lambda_i} \boldsymbol{\phi}_i^u. \tag{3.79}$$

Using Eqs. (3.24) and (3.71a), the displacement field can also be expressed as

$$\tilde{\mathbf{u}}(\xi, \eta) = \mathbf{N}_u(\eta)\mathbf{u}(\xi) = \mathbf{N}_u(\eta)\mathbf{\Phi}_b^u \xi^{\langle \lambda_b \rangle} [\mathbf{\Phi}_b^u]^{-1} \mathbf{u}_b = \mathbf{N}_v(\xi, \eta)\mathbf{u}_b, \tag{3.80}$$

where, $\mathbf{N}_v(\xi,\eta)$ is the shape function of the s-element. The stress field can be calculated by substituting $\mathbf{u}(\xi)$ into Eq. (3.28) as

$$\boldsymbol{\sigma}(\xi,\eta) = \mathbf{D}\sum_{i=1}^{n} c_i \xi^{\lambda_i - 1}(\lambda_i \mathbf{B}_1(\eta) + \mathbf{B}_2(\eta))\boldsymbol{\phi}_i^u. \tag{3.81}$$

3.1.5 Derivation of SBFE formulation for elastodynamics

We can obtain the equations of motion in elastodynamics by including the virtual work δU_ρ due to inertial force into the principle of virtual work define by the Eq. (3.38). δU_ρ is given as,

$$\delta U_\rho = \int_V \delta\tilde{\mathbf{u}}^T \rho \tilde{\mathbf{a}} \mathrm{d}V. \tag{3.82}$$

Here, ρ is the mass density. The acceleration field $\tilde{\mathbf{a}}(\xi,\eta)$ can be defined using the same shape function as the displacement field in Eq. (3.80) as

$$\tilde{\mathbf{a}}(\xi,\eta) = \mathbf{N}_v(\xi,\eta)\ddot{\mathbf{u}}_b. \tag{3.83}$$

Substituting the displacement and acceleration fields from Eqs. (3.80) and (3.83) into Eq. (3.82), we obtain

$$\begin{aligned}\delta U_\rho &= \delta\mathbf{u}_b^T \int_V \mathbf{N}_v^T(\xi,\eta)\rho\mathbf{N}_v(\xi,\eta)\mathrm{d}V\ddot{\mathbf{u}}_b \\ &= \delta\mathbf{u}_b^T \mathbf{M}^e \ddot{\mathbf{u}}_b,\end{aligned} \tag{3.84}$$

where, $\mathbf{M}^e$ is the element mass matrix. The expression for $\mathbf{M}^e$ can be simplified by substituting the shape function $\mathbf{N}_v(\xi,\eta)$ and differential volume $\mathrm{d}V$ from Eqs. (3.80) and (3.16) respectively into Eq. (3.84). Then,

$$\mathbf{M}^e = \left[\boldsymbol{\Phi}_b^u\right]^{-T} \int_0^1 \xi^{\langle\lambda_b\rangle} \left[\boldsymbol{\Phi}_b^u\right]^T \mathbf{M}_0 \left[\boldsymbol{\Phi}_b^u\right] \xi^{\langle\lambda_b\rangle} \xi \mathrm{d}\xi \left[\boldsymbol{\Phi}_b^u\right]^{-1}, \tag{3.85}$$

where, the coefficient matrix $\mathbf{M}_0$ is given by

$$\mathbf{M}_0 = \sum_e \int_{-1}^{+1}\int_{-1}^{+1} \mathbf{N}_u^T(\eta)\rho\mathbf{N}_u(\eta)\left|J_b\right| \mathrm{d}\eta. \tag{3.86}$$

Here, the integration is done for each sector corresponding to the line element e in the s-element. The expression for the mass matrix can be rewritten as

$$\mathbf{M}^e = [\mathbf{\Phi}_b^u]^{-T} \mathbf{m} [\mathbf{\Phi}_b^u]^{-1}, \tag{3.87}$$

where,

$$\begin{aligned} \mathbf{m} &= \int_0^1 \xi^{\langle \lambda_b \rangle} [\mathbf{\Phi}_b^u]^T \mathbf{M}_0 [\mathbf{\Phi}_b^u] \xi^{\langle \lambda_b \rangle} \xi \mathrm{d}\xi \\ &= \int_0^1 \xi^{\langle \lambda_b \rangle} \mathbf{m}_0 \xi^{\langle \lambda_b \rangle} \xi \mathrm{d}\xi. \end{aligned} \tag{3.88}$$

Each entry of the matrix $\mathbf{m}$ can be integrated analytically as

$$\begin{aligned} m_{ij} &= \int_0^1 \xi^{\lambda_{bi}} m_{0ij} \xi^{\lambda_{bj}} \xi \mathrm{d}\xi \\ &= \frac{m_{0ij}}{\lambda_{bi} + \lambda_{bj} + 2}. \end{aligned} \tag{3.89}$$

Augmenting Eq. (3.38) to include inertial term, we obtain

$$\delta U_\rho + \delta U_\varepsilon - \delta W = 0, \tag{3.90}$$

Substituting Eqs. (3.40), (3.46) and (3.84) into Eq. (3.90) yields

$$\delta \mathbf{u}_b^T (\mathbf{M}^e \ddot{\mathbf{u}}_b + \mathbf{q}(\xi = 1) - \mathbf{f}_b) - \int_0^1 \delta \mathbf{u}^T(\xi) \left(\mathbf{q}(\xi)_{,\xi} - \sum_e \int_{-1}^{+1} \mathbf{B}_2^T(\eta) \boldsymbol{\sigma} \left| \mathbf{J}_b \right| \mathrm{d}\eta \right) \mathrm{d}\xi = 0, \tag{3.91}$$

Considering the arbitrariness of the virtual displacement at the boundary as well as within the s-element, we obtain

$$\mathbf{f}_b = \mathbf{M}^e \ddot{\mathbf{u}}_b + \mathbf{q}(\xi = 1), \tag{3.92}$$

$$\mathbf{q}(\xi)_{,\xi} = \sum_e \int_{-1}^{+1} \mathbf{B}_2^T(\eta) \boldsymbol{\sigma} \left| \mathbf{J}_b \right| \mathrm{d}\eta. \tag{3.93}$$

From Eq. (3.93), the scaled boundary equation given in Eq. (3.55) can be formulated whose solution is derived in Section 3.1.4. From Eqs. (3.76), (3.78), and Eq. (3.92), we obtain the following equation of motion in elastodynamics in two dimensions:

$$\mathbf{M}^e \ddot{\mathbf{u}}_b + \mathbf{K}^e \mathbf{u}_b = \mathbf{f}_b. \tag{3.94}$$

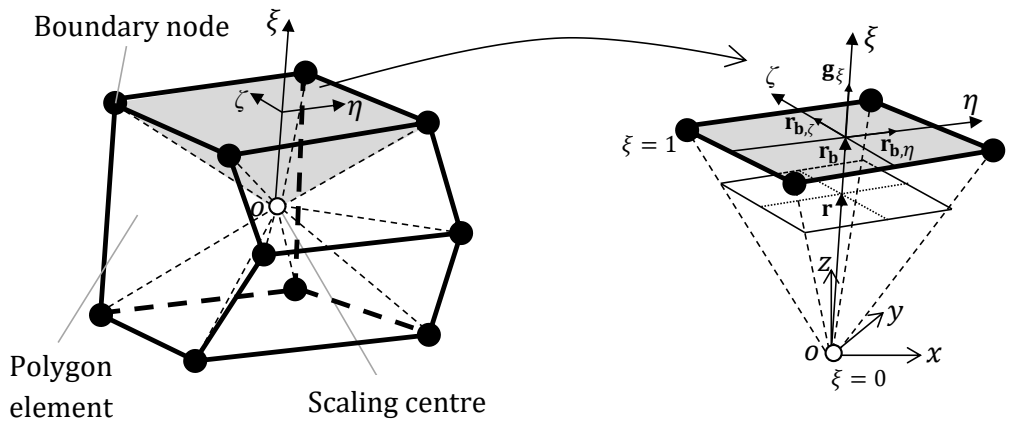


Figure 3.2: An s-element in 3D

3.2 Formulation of SBFEM in three dimensions

In this section, we briefly derive the formulations of the SBFEM for 3D elastostatics and elastodynamics.

3.2.1 Geometric modelling in scaled boundary coordinates

In the framework of the SBFEM, a problem domain in three dimensions can be subdivided into arbitrary shaped polyhedral elements or s-elements. There is no restriction on the number of nodes, edges and faces in each element. Each element only needs to satisfy the star convexity criterion, that is, any point on its boundary is directly visible from a point (the scaling centre) inside it. The boundary of the element is discretised into two dimensional surface elements. We introduce the three-dimensional scaled boundary coordinates (ξ, η, ζ), as shown in Fig. 3.2. ξ is a radial coordinate varying from $\xi = 0$ at the scaling centre O to $\xi = 1$ at the boundary. η and ζ are the circumferential coordinates defined for each surface element. The origin of the local Cartesian coordinate system is chosen at the scaling centre without the loss of generality.

A solid sector of the domain is generated by scaling the surface element from the boundary at $\xi = 1$ to the scaling centre at $\xi = 0$, as shown by the grey volume in

Fig. 3.2. A position vector $\mathbf{r}$ within the solid sector can be given by,

$$\mathbf{r} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k}, \tag{3.95}$$

$$\mathbf{r} = \xi\mathbf{r}_b = \xi x_b\mathbf{i} + \xi y_b\mathbf{j} + \xi z_b\mathbf{k}, \tag{3.96}$$

where, $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are unit vectors of the Cartesian coordinate system, and x_b, y_b, and z_b are the coordinates of a point on the surface element at the boundary. The coordinates (x, y, z) of the position vector $\mathbf{r}$ can be obtained as

$$x(\xi, \eta, \zeta) = \xi x_b = \xi\mathbf{N}(\eta, \zeta)\mathbf{x}, \tag{3.97}$$

$$y(\xi, \eta, \zeta) = \xi y_b = \xi\mathbf{N}(\eta, \zeta)\mathbf{y}, \tag{3.98}$$

$$z(\xi, \eta, \zeta) = \xi z_b = \xi\mathbf{N}(\eta, \zeta)\mathbf{z}, \tag{3.99}$$

where, $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ are the coordinates of nodes of the 2D surface element on the boundary, and $\mathbf{N}(\eta, \zeta)$ is the shape function of the surface element. The surface element can be chosen as any arbitrary shaped polygon. In the present work, only linear triangular and quadrilateral surface elements are considered (Fig. 3.3) whose shape functions can be obtained from isoparametric formulations in FEM. For an isoparametric linear triangular surface element, $\mathbf{N}(\eta, \zeta)$ is expressed as

$$\mathbf{N}(\eta, \zeta) = [N_1(\eta, \zeta), N_2(\eta, \zeta), N_3(\eta, \zeta)], \tag{3.100}$$

where,

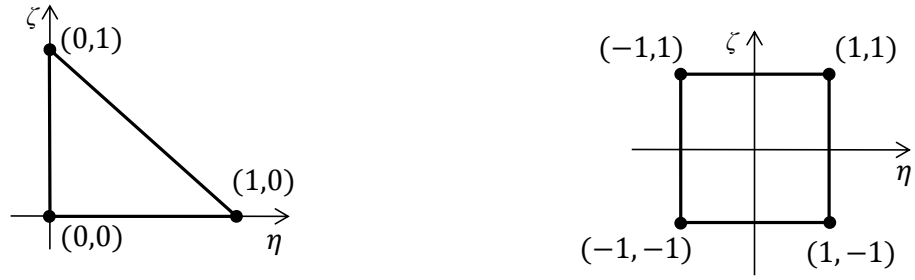
$$N_1(\eta, \zeta) = 1 - \eta - \zeta, \tag{3.101a}$$

$$N_2(\eta, \zeta) = \eta, \tag{3.101b}$$

$$N_3(\eta, \zeta) = \zeta. \tag{3.101c}$$

For an isoparametric linear quadrilateral element, $\mathbf{N}(\eta)$ is given by

$$\mathbf{N}(\eta, \zeta) = [N_1(\eta, \zeta), N_2(\eta, \zeta), N_3(\eta, \zeta), N_4(\eta, \zeta)], \tag{3.102}$$



(a) Isoparametric linear triangular element

(b) Isoparametric linear quadrilateral element

Figure 3.3: Isoparametric elements

where,

$$N_1(\eta,\zeta) = \frac{1}{4}(1-\eta)(1-\zeta), \tag{3.103a}$$

$$N_2(\eta,\zeta) = \frac{1}{4}(1+\eta)(1-\zeta), \tag{3.103b}$$

$$N_3(\eta,\zeta) = \frac{1}{4}(1-\eta)(1+\zeta), \tag{3.103c}$$

$$N_4(\eta,\zeta) = \frac{1}{4}(1+\eta)(1+\zeta). \tag{3.103d}$$

Similar to the 2D transformation procedure, the 3D scaled boundary coordinates (ξ,η,ζ) can be transformed into the 3D Cartesian coordinates (x,y,z) using the standard isoparametric mapping technique. The transformation equation is given by

$$\left\{\begin{array}{c}\frac{\partial}{\partial\xi}\\ \frac{\partial}{\partial\eta}\\ \frac{\partial}{\partial\zeta}\end{array}\right\} = \mathbf{J}(\xi,\eta,\zeta)\left\{\begin{array}{c}\frac{\partial}{\partial x}\\ \frac{\partial}{\partial y}\\ \frac{\partial}{\partial z}\end{array}\right\}, \tag{3.104}$$

where, the Jacobian matrix $\mathbf{J}(\xi,\eta,\zeta)$ is expressed as

$$\mathbf{J}(\xi,\eta,\zeta) = \begin{bmatrix} x_{,\xi} & y_{,\xi} & z_{,\xi} \\ x_{,\eta} & y_{,\eta} & z_{,\eta} \\ x_{,\zeta} & y_{,\zeta} & z_{,\zeta} \end{bmatrix}. \tag{3.105}$$

The items in the Jacobian matrix are related to the nodal boundary coordinates and

are obtained as

$$x_{,\xi} = x_b = \mathbf{N}(\eta,\zeta)\mathbf{x}, \tag{3.106a}$$

$$x_{,\eta} = \xi x_{b,\eta} = \xi \mathbf{N}(\eta,\zeta)_{,\eta}\mathbf{x}, \tag{3.106b}$$

$$x_{,\zeta} = \xi x_{b,\zeta} = \xi \mathbf{N}(\eta,\zeta)_{,\zeta}\mathbf{x}, \tag{3.106c}$$

$$y_{,\xi} = y_b = \mathbf{N}(\eta,\zeta)\mathbf{y}, \tag{3.106d}$$

$$y_{,\eta} = \xi y_{b,\eta} = \xi \mathbf{N}(\eta,\zeta)_{,\eta}\mathbf{y}, \tag{3.106e}$$

$$y_{,\zeta} = \xi y_{b,\zeta} = \xi \mathbf{N}(\eta,\zeta)_{,\zeta}\mathbf{y}, \tag{3.106f}$$

$$z_{,\xi} = z_b = \mathbf{N}(\eta,\zeta)\mathbf{z}, \tag{3.106g}$$

$$z_{,\eta} = \xi z_{b,\eta} = \xi \mathbf{N}(\eta,\zeta)_{,\eta}\mathbf{z}, \tag{3.106h}$$

$$z_{,\zeta} = \xi z_{b,\zeta} = \xi \mathbf{N}(\eta,\zeta)_{,\zeta}\mathbf{z}. \tag{3.106i}$$

The Eq. (3.104) can be rewritten as

$$\mathbf{J}(\xi,\eta,\zeta) = \mathrm{diag}(1,\xi,\xi)\mathbf{J}_b, \tag{3.107}$$

where, $\mathbf{J}_b$ is the Jacobian matrix at the boundary of the s-element ($\xi = 1$) and given as

$$\mathbf{J}_b = \begin{bmatrix} x_b & y_b & z_b \\ x_{b,\eta} & y_{b,\eta} & z_{b,\eta} \\ x_{b,\zeta} & y_{b,\zeta} & z_{b,\zeta} \end{bmatrix}. \tag{3.108}$$

From Eqs. (3.104) and (3.107), the inverse relationship is obtained as

$$\begin{Bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{Bmatrix} = \mathbf{J}_b^{-1} \begin{Bmatrix} \frac{\partial}{\partial \xi} \\ \frac{1}{\xi}\frac{\partial}{\partial \eta} \\ \frac{1}{\xi}\frac{\partial}{\partial \zeta} \end{Bmatrix}, \tag{3.109}$$

where, $\mathbf{J}_b^{-1}$ is the inverse of the Jacobian matrix at the boundary and given by

$$\mathbf{J}_b^{-1} = \frac{1}{|\mathbf{J}_b|} \begin{bmatrix} \mathbf{A} & \mathbf{B} & \mathbf{C} \end{bmatrix}, \tag{3.110}$$

where,

$$\mathbf{A} = \left\{ \begin{array}{c} y_{b,\eta} z_{b,\zeta} - z_{b,\eta} y_{b,\zeta} \\ z_{b,\eta} x_{b,\zeta} - x_{b,\eta} z_{b,\zeta} \\ x_{b,\eta} y_{b,\zeta} - y_{b,\eta} x_{b,\zeta} \end{array} \right\}, \tag{3.111a}$$

$$\mathbf{B} = \left\{ \begin{array}{c} z_b y_{b,\zeta} - y_b z_{b,\zeta} \\ x_b z_{b,\zeta} - z_b x_{b,\zeta} \\ y_b x_{b,\zeta} - x_b y_{b,\zeta} \end{array} \right\}, \tag{3.111b}$$

$$\mathbf{C} = \left\{ \begin{array}{c} y_b z_{b,\eta} - z_b y_{b,\eta} \\ z_b x_{b,\eta} - x_b z_{b,\eta} \\ x_b y_{b,\eta} - y_b x_{b,\eta} \end{array} \right\}. \tag{3.111c}$$

Substituting Eq. (3.110) into Eq. (3.109) leads to

$$\left\{ \begin{array}{c} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{array} \right\} = \frac{\mathbf{A}}{|\mathbf{J}_b|} \frac{\partial}{\partial \xi} + \frac{1}{\xi} \left(\frac{\mathbf{B}}{|\mathbf{J}_b|} \frac{\partial}{\partial \eta} + \frac{\mathbf{C}}{|\mathbf{J}_b|} \frac{\partial}{\partial \zeta} \right) . \tag{3.112}$$

The partial derivatives of the position vector $\mathbf{r}$ defined in Eq. (3.96) with respect to the scaled boundary coordinates (ξ, η, ζ) are expressed as

$$\mathbf{r}_{,\xi} = \mathbf{r}_b = x_b \mathbf{i} + y_b \mathbf{j} + z_b \mathbf{k}, \tag{3.113a}$$

$$\mathbf{r}_{,\eta} = \xi \mathbf{r}_{b,\eta} = \xi (x_{b,\eta} \mathbf{i} + y_{b,\eta} \mathbf{j} + z_{b,\eta} \mathbf{k}), \tag{3.113b}$$

$$\mathbf{r}_{,\zeta} = \xi \mathbf{r}_{b,\zeta} = \xi (x_{b,\zeta} \mathbf{i} + y_{b,\zeta} \mathbf{j} + z_{b,\zeta} \mathbf{k}), \tag{3.113c}$$

where, $\mathbf{r}_{b,\eta}$ and $\mathbf{r}_{b,\zeta}$ are the tangential vectors to the surface element. Thus, the outward normal vector in the perpendicular direction of the surface element is obtained as

$$\mathbf{g}_\xi = \mathbf{r}_{b,\eta} \times \mathbf{r}_{b,\zeta}. \tag{3.114}$$

The outward normal vectors to the sidefaces with constant η or ζ are given by

$$\mathbf{g}_\eta = \mathbf{r}_{b,\zeta} \times \mathbf{r}_{,\xi}, \tag{3.115a}$$

$$\mathbf{g}_\zeta = \mathbf{r}_{,\xi} \times \mathbf{r}_{b,\eta}. \tag{3.115b}$$

The infinitesimal surface area $\mathrm{d}S_\xi$ at given value of ξ can be calculated as

$$\mathrm{d}S_\xi = |\mathbf{r}_{,\eta} \times \mathbf{r}_{,\zeta}|\,\mathrm{d}\eta\mathrm{d}\zeta = |\xi\mathbf{r}_{b,\eta} \times \xi\mathbf{r}_{b,\zeta}|\,\mathrm{d}\eta\mathrm{d}\zeta = \xi^2|\mathbf{g}_\xi|\mathrm{d}\eta\mathrm{d}\zeta. \tag{3.116}$$

Similarly, the infinitesimal surface area $\mathrm{d}S_\eta$ and $\mathrm{d}S_\zeta$ on the sidefaces are calculated as

$$\mathrm{d}S_\eta = |\mathbf{r}_{,\zeta} \times \mathbf{r}_{,\xi}|\,\mathrm{d}\zeta\mathrm{d}\xi = |\xi\mathbf{r}_{b,\zeta} \times \mathbf{r}_b|\,\mathrm{d}\zeta\mathrm{d}\xi = \xi|\mathbf{g}_\eta|\mathrm{d}\zeta\mathrm{d}\xi, \tag{3.117a}$$

$$\mathrm{d}S_\zeta = |\mathbf{r}_{,\xi} \times \mathbf{r}_{,\eta}|\,\mathrm{d}\xi\mathrm{d}\eta = |\mathbf{r}_b \times \xi\mathbf{r}_{b,\eta}|\,\mathrm{d}\xi\mathrm{d}\eta = \xi|\mathbf{g}_\zeta|\mathrm{d}\xi\mathrm{d}\eta. \tag{3.117b}$$

An infinitesimal volume change $\mathrm{d}V$ of the s-element for infinitesimal change $(d\xi, d\eta, d\zeta)$ in the scaled boundary coordinates can be calculated as

$$\mathrm{d}V = \mathbf{r}_{,\xi} \cdot (\mathbf{r}_{,\eta} \times \mathbf{r}_{,\zeta})\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta = \xi^2\,|\mathbf{J}_b|\,\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\zeta. \tag{3.118}$$

3.2.2 Formulation of linear elasticity in scaled boundary coordinates

In a three-dimensional Cartesian coordinate system, the displacement field $\tilde{\mathbf{u}} = \tilde{\mathbf{u}}(x, y, z)$ of a point can be written as

$$\tilde{\mathbf{u}} = [u_x, u_y, u_z]^{\mathrm{T}}. \tag{3.119}$$

The strain field $\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}(x, y, z)$ can be defined in terms of the displacement field as

$$\boldsymbol{\varepsilon} = [\varepsilon_x, \varepsilon_y, \varepsilon_z, \gamma_{yz}, \gamma_{zx}, \gamma_{xy}]^{\mathrm{T}} = \mathbf{L}\tilde{\mathbf{u}}, \tag{3.120}$$

where, $\mathbf{L}$ is the differential operator given by

$$\mathbf{L} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \end{bmatrix}. \tag{3.121}$$

For a linear elastic material, the constitutive relationship can be expressed as

$$\boldsymbol{\sigma} = [\sigma_x, \sigma_y, \sigma_z, \tau_{yz}, \tau_{zx}, \tau_{xy}]^{\mathrm{T}} = \mathbf{D}\boldsymbol{\varepsilon}, \tag{3.122}$$

where, $\mathbf{D}$ is a 6×6 elasticity matrix and $\boldsymbol{\sigma} = \boldsymbol{\sigma}(x, y, z)$ is the field stress. Under zero body forces, the governing equilibrium equation of elasticity is obtained as

$$\mathbf{L}^{\mathrm{T}}\boldsymbol{\sigma} = 0. \tag{3.123}$$

Using Eq. (3.112), we express Eq. (3.121) in the scaled boundary coordinates as

$$\mathbf{L} = \mathbf{b}_1(\eta, \zeta)\frac{\partial}{\partial \xi} + \frac{1}{\xi}\left(\mathbf{b}_2(\eta, \zeta)\frac{\partial}{\partial \eta} + \mathbf{b}_3(\eta, \zeta)\frac{\partial}{\partial \zeta}\right), \tag{3.124}$$

where, $\mathbf{b}_1(\eta, \zeta)$, $\mathbf{b}_2(\eta, \zeta)$ and $\mathbf{b}_3(\eta, \zeta)$ are defined by

$$\mathbf{b}_1(\eta, \zeta) = \frac{1}{|\mathbf{J_b}|}\begin{bmatrix} y_{b,\eta}z_{b,\zeta} - z_{b,\eta}y_{b,\zeta} & 0 & 0 \\ 0 & z_{b,\eta}x_{b,\zeta} - x_{b,\eta}z_{b,\zeta} & 0 \\ 0 & 0 & x_{b,\eta}y_{b,\zeta} - y_{b,\eta}x_{b,\zeta} \\ 0 & x_{b,\eta}y_{b,\zeta} - y_{b,\eta}x_{b,\zeta} & z_{b,\eta}x_{b,\zeta} - x_{b,\eta}z_{b,\zeta} \\ x_{b,\eta}y_{b,\zeta} - y_{b,\eta}x_{b,\zeta} & 0 & y_{b,\eta}z_{b,\zeta} - z_{b,\eta}y_{b,\zeta} \\ z_{b,\eta}x_{b,\zeta} - x_{b,\eta}z_{b,\zeta} & y_{b,\eta}z_{b,\zeta} - z_{b,\eta}y_{b,\zeta} & 0 \end{bmatrix}, \tag{3.125a}$$

$$\mathbf{b}_2(\eta, \zeta) = \frac{1}{|\mathbf{J_b}|}\begin{bmatrix} z_b y_{b,\zeta} - y_b z_{b,\zeta} & 0 & 0 \\ 0 & x_b z_{b,\zeta} - z_b x_{b,\zeta} & 0 \\ 0 & 0 & y_b x_{b,\zeta} - x_b y_{b,\zeta} \\ 0 & y_b x_{b,\zeta} - x_b y_{b,\zeta} & x_b z_{b,\zeta} - z_b x_{b,\zeta} \\ y_b x_{b,\zeta} - x_b y_{b,\zeta} & 0 & z_b y_{b,\zeta} - y_b z_{b,\zeta} \\ x_b z_{b,\zeta} - z_b x_{b,\zeta} & z_b y_{b,\zeta} - y_b z_{b,\zeta} & 0 \end{bmatrix}, \tag{3.125b}$$

$$\mathbf{b}_3(\eta, \zeta) = \frac{1}{|\mathbf{J}_b|}\begin{bmatrix} y_b z_{b,\eta} - z_b y_{b,\eta} & 0 & 0 \\ 0 & z_b x_{b,\eta} - x_b z_{b,\eta} & 0 \\ 0 & 0 & x_b y_{b,\eta} - y_b x_{b,\eta} \\ 0 & x_b y_{b,\eta} - y_b x_{b,\eta} & z_b x_{b,\eta} - x_b z_{b,\eta} \\ x_b y_{b,\eta} - y_b x_{b,\eta} & 0 & y_b z_{b,\eta} - z_b y_{b,\eta} \\ z_b x_{b,\eta} - x_b z_{b,\eta} & y_b z_{b,\eta} - z_b y_{b,\eta} & 0 \end{bmatrix}. \tag{3.125c}$$

The field displacement $\tilde{\mathbf{u}}(\xi, \eta, \zeta)$ inside the s-element can be given as

$$\tilde{\mathbf{u}}(\xi, \eta, \zeta) = \mathbf{N}_u(\eta, \zeta)\mathbf{u}(\xi), \tag{3.126}$$

where, $\mathbf{u}(\xi)$ represent the displacement along the radial lines passing through the scaling centre and boundary nodes of the s-element. Here, the interpolation shape function $\mathbf{N}_u(\eta, \zeta)$ is expressed as

$$\mathbf{N}_u(\eta, \zeta) = [N_1(\eta, \zeta)\mathbf{I},\ N_2(\eta, \zeta)\mathbf{I}, ..., N_n(\eta, \zeta)\mathbf{I}], \tag{3.127}$$

where, $\mathbf{I}$ is a 3×3 identity matrix and n is the number of nodes of a surface element. Using Eqs. (3.120) and (3.124), the strain field in scaled boundary coordinates can be obtained as

$$\boldsymbol{\varepsilon}(\xi, \eta, \zeta) = \mathbf{B}_1(\eta, \zeta)\mathbf{u}(\xi)_{,\xi} + \frac{1}{\xi}\mathbf{B}_2(\eta, \zeta)\mathbf{u}(\xi), \tag{3.128}$$

where, $\mathbf{B}_1(\eta, \zeta)$ and $\mathbf{B}_2(\eta, \zeta)$ are expressed as

$$\mathbf{B}_1(\eta, \zeta) = \mathbf{b}_1(\eta, \zeta)\mathbf{N}_u(\eta, \zeta), \tag{3.129a}$$

$$\mathbf{B}_2(\eta, \zeta) = \mathbf{b}_2(\eta, \zeta)\mathbf{N}_u(\eta, \zeta)_{,\eta} + \mathbf{b}_3(\eta, \zeta)\mathbf{N}_{\mathrm{u}}(\eta, \zeta)_{,\zeta}. \tag{3.129b}$$

The stress field defined in Eq. (3.122) can be obtained in scaled boundary coordinates using Eq. (3.128) as

$$\boldsymbol{\sigma}(\xi, \eta, \zeta) = \mathbf{D}(\mathbf{B}_1(\eta, \zeta)\mathbf{u}(\xi)_{,\xi} + \frac{1}{\xi}\mathbf{B}_2(\eta, \zeta)\mathbf{u}(\xi)). \tag{3.130}$$

The tractions on the surfaces [1] with constant ξ, η and ζ can be expressed as

$$\mathbf{t}_\xi = \frac{|\mathbf{J}_b|}{|\mathbf{g}_\xi|}\mathbf{b}_1^{\mathrm{T}}(\eta, \zeta)\boldsymbol{\sigma}, \tag{3.131a}$$

$$\mathbf{t}_\eta = \frac{|\mathbf{J}_b|}{|\mathbf{g}_\eta|}\mathbf{b}_2^{\mathrm{T}}(\eta, \zeta)\boldsymbol{\sigma}, \tag{3.131b}$$

$$\mathbf{t}_\zeta = \frac{|\mathbf{J}_b|}{|\mathbf{g}_\zeta|}\mathbf{b}_3^{\mathrm{T}}(\eta, \zeta)\boldsymbol{\sigma}. \tag{3.131c}$$

We introduce an internal nodal force functions $\mathbf{q}(\xi)$ along the radial coordinate such that it is equal to the resultant of the surface traction $\mathbf{t}_\xi$ at the surface with a constant

ξ. From energy equivalence, we have

$$\mathbf{u}^{\mathrm{T}}(\xi)\mathbf{q}(\xi) = \sum_{e} \int_{S_\xi} \tilde{\mathbf{u}}^{\mathrm{T}}(\xi,\eta,\zeta)\mathbf{t}_\xi \mathrm{dS}_\xi. \tag{3.132}$$

Here, the integration is performed for each solid sector corresponding to the surface element e in the s-element. Substituting Eqs. (3.116), (3.126), (3.129a) and (3.131a) into Eq. (3.132), we obtain

$$\mathbf{q}(\xi) = \sum_{e} \xi^2 \int_{-1}^{+1} \int_{-1}^{+1} \mathbf{B}_1^{\mathrm{T}}(\eta,\zeta)\boldsymbol{\sigma}\, |\mathbf{J}_b|\, \mathrm{d}\eta \mathrm{d}\zeta. \tag{3.133}$$

3.2.3 Derivation of the SBFE formulation for elastostatics

In this section, we derive the scaled boundary finite element formulation using the principle of virtual work. Using Eq. (3.126), we can obtain a virtual displacement field as

$$\delta\tilde{\mathbf{u}}(\xi,\eta,\zeta) = \mathbf{N}_u(\eta,\zeta)\delta\mathbf{u}(\xi). \tag{3.134}$$

Similarly, from Eq. (3.128), a virtual strain field $\delta\boldsymbol{\varepsilon}(\xi,\eta)$ can be obtained as

$$\delta\boldsymbol{\varepsilon}(\xi,\eta,\zeta) = \mathbf{B}_1(\eta,\zeta)\delta\mathbf{u}(\xi)_{,\xi} + \frac{1}{\xi}\mathbf{B}_2(\eta,\zeta)\delta\mathbf{u}(\xi), \tag{3.135}$$

Using the principle of virtual work in elastostatics, one can obtain

$$\delta U_\varepsilon - \delta W = 0, \tag{3.136}$$

where, δU_ε is the virtual increment in strain energy and δW is the virtual work done by the external force. These are given as

$$\delta U_\varepsilon = \int_V \delta\boldsymbol{\varepsilon}^T\boldsymbol{\sigma}\mathrm{d}V, \tag{3.137}$$

$$\delta W = \delta\mathbf{u}_b^T\mathbf{f}_b, \tag{3.138}$$

where,

$$\delta\mathbf{u}_b = \delta\mathbf{u}(\xi = 1). \tag{3.139}$$

Here, $\delta\mathbf{u}_b$ denotes the virtual displacement at the boundary ($\xi = 1$) and $\mathbf{f}_b$ represents the boundary nodal forces. Substituting Eqs. (3.118) and (3.135) into Eq. (3.137), we obtain

$$\delta U_\varepsilon = \underbrace{\sum_e \int_0^1 \delta\mathbf{u}^T(\xi)_{,\xi}\xi^2 \int_{-1}^{+1}\int_{-1}^{+1} \mathbf{B}_1^T(\eta,\zeta)\boldsymbol{\sigma}\,|\mathbf{J}_b|\,\mathrm{d}\eta d\zeta \mathrm{d}\xi}_{\delta U_{\varepsilon 1}} + \underbrace{\sum_e \int_0^1 \delta\mathbf{u}^T(\xi)\xi \int_{-1}^{+1}\int_{-1}^{+1} \mathbf{B}_2^T(\eta,\zeta)\boldsymbol{\sigma}\,|\mathbf{J}_b|\,\mathrm{d}\eta d\zeta \mathrm{d}\xi}_{\delta U_{\varepsilon 2}}. \tag{3.140}$$

The integration is done for each solid sector corresponding to the surface element e in the s-element. Substituting Eq. (3.133) in $\delta U_{\varepsilon 1}$, we get

$$\delta U_{\varepsilon 1} = \int_0^1 \delta\mathbf{u}^T(\xi)_{,\xi}\mathbf{q}(\xi)\mathrm{d}\xi. \tag{3.141}$$

Applying integration by parts with respect to ξ leads to

$$\delta U_{\varepsilon 1} = \delta\mathbf{u}^T(\xi)\mathbf{q}(\xi)\,|_{\xi=1} - \int_0^1 \delta\mathbf{u}^T(\xi)\mathbf{q}(\xi)_{,\xi}\mathrm{d}\xi. \tag{3.142}$$

From Eq. (3.139), $\delta U_{\varepsilon 1}$ can be expressed as

$$\delta U_{\varepsilon 1} = \delta\mathbf{u}_b^T\mathbf{q}(\xi = 1) - \int_0^1 \delta\mathbf{u}^T(\xi)\mathbf{q}(\xi)_{,\xi}\mathrm{d}\xi. \tag{3.143}$$

Putting Eq. (3.143) back into Eq. (3.140), we get

$$\delta U_\varepsilon = \delta\mathbf{u}_b^T\mathbf{q}(\xi = 1) - \int_0^1 \delta\mathbf{u}^T(\xi)\left(\mathbf{q}(\xi)_{,\xi} - \sum_e \xi \int_{-1}^{+1}\int_{-1}^{+1} \mathbf{B}_2^T(\eta,\zeta)\boldsymbol{\sigma}\,|\mathbf{J}_b|\,\mathrm{d}\eta d\zeta\right)\mathrm{d}\xi. \tag{3.144}$$

Substituting Eqs. (3.144) and (3.138) into Eq. (3.136), we obtain the following equation

$$\delta\mathbf{u}_b^T(\mathbf{q}(\xi = 1) - \mathbf{f}_b) - \int_0^1 \delta\mathbf{u}^T(\xi)\left(\mathbf{q}(\xi)_{,\xi} - \sum_e \xi \int_{-1}^{+1}\int_{-1}^{+1} \mathbf{B}_2^T(\eta,\zeta)\boldsymbol{\sigma}\,|\mathbf{J}_b|\,\mathrm{d}\eta d\zeta\right)\mathrm{d}\xi = 0. \tag{3.145}$$

For any arbitrary selection of virtual displacement within the s-element with $0 \leq \xi \leq 1$ and on the boundary with $\xi = 1$, the following condition must be satisfied:

$$\mathbf{f}_b = \mathbf{q}(\xi = 1), \tag{3.146}$$

$$\mathbf{q}(\xi)_{,\xi} = \sum_e \xi \int_{-1}^{+1} \int_{-1}^{+1} \mathbf{B}_2^{\mathrm{T}}(\eta, \zeta) \boldsymbol{\sigma} \left|\mathbf{J}_b\right| \mathrm{d}\eta d\zeta. \tag{3.147}$$

Substituting Eq. (3.130) into Eqs. (3.133) and (3.147), we obtain

$$\mathbf{q}(\xi) = \sum_e \xi^2 \int_{-1}^{+1} \int_{-1}^{+1} \mathbf{B}_1^{\mathrm{T}}(\eta, \zeta) \mathbf{D} \left(\mathbf{B}_1(\eta, \zeta) \mathbf{u}(\xi)_{,\xi} + \frac{1}{\xi} \mathbf{B}_2(\eta, \zeta) \mathbf{u}(\xi) \right) \left|\mathbf{J}_b\right| \mathrm{d}\eta \mathrm{d}\zeta, \tag{3.148}$$

and

$$\mathbf{q}(\xi)_{,\xi} = \sum_e \xi \int_{-1}^{+1} \int_{-1}^{+1} \mathbf{B}_2^{\mathrm{T}}(\eta, \zeta) \mathbf{D} \left(\mathbf{B}_1(\eta, \zeta) \mathbf{u}(\xi)_{,\xi} + \frac{1}{\xi} \mathbf{B}_2(\eta, \zeta) \mathbf{u}(\xi) \right) \left|\mathbf{J}_b\right| \mathrm{d}\eta \mathrm{d}\zeta, \tag{3.149}$$

respectively. We introduce the following coefficient matrices $\mathbf{E}_0$, $\mathbf{E}_1$ and $\mathbf{E}_2$ in order to simplify the expression in Eqs. (3.148) and (3.149),

$$\mathbf{E}_0 = \sum_e \int_{-1}^{+1} \int_{-1}^{+1} \mathbf{B}_1^{\mathrm{T}}(\eta, \zeta) \mathbf{D} \mathbf{B}_1(\eta, \zeta) \left|\mathbf{J}_b\right| \mathrm{d}\eta \mathrm{d}\zeta, \tag{3.150a}$$

$$\mathbf{E}_1 = \sum_e \int_{-1}^{+1} \int_{-1}^{+1} \mathbf{B}_2^{\mathrm{T}}(\eta, \zeta) \mathbf{D} \mathbf{B}_1(\eta, \zeta) \left|\mathbf{J}_b\right| \mathrm{d}\eta \mathrm{d}\zeta, \tag{3.150b}$$

$$\mathbf{E}_2 = \sum_e \int_{-1}^{+1} \int_{-1}^{+1} \mathbf{B}_2^{\mathrm{T}}(\eta, \zeta) \mathbf{D} \mathbf{B}_2(\eta, \zeta) \left|\mathbf{J}_b\right| \mathrm{d}\eta \mathrm{d}\zeta. \tag{3.150c}$$

The Eqs. (3.148) and (3.149) can be rewritten as

$$\mathbf{q}(\xi) = \mathbf{E}_0 \xi^2 \mathbf{u}(\xi)_{,\xi} + \mathbf{E}_1^{\mathrm{T}} \xi \mathbf{u}(\xi), \tag{3.151}$$

$$\mathbf{q}(\xi)_{,\xi} = \mathbf{E}_1 \xi \mathbf{u}(\xi)_{,\xi} + \mathbf{E}_2 \mathbf{u}(\xi). \tag{3.152}$$

Combining Eq. (3.151) and Eq. (3.152), the following scaled boundary finite element equation is obtained:

$$\mathbf{E}_0 \xi^2 \mathbf{u}(\xi)_{,\xi\xi} + (2\mathbf{E}_0 + \mathbf{E}_1^{\mathrm{T}} - \mathbf{E}_1) \xi \mathbf{u}(\xi)_{,\xi} + (\mathbf{E}_1^{\mathrm{T}} - \mathbf{E}_2) \mathbf{u}(\xi) = 0. \tag{3.153}$$

3.2.4 Solution of the SBFE equation

The solution procedure of a three dimensional problem is very similar to the two dimensional problem explained in Section 3.1.4. Therefore, only the important results are shown here for brevity.

Eq. (3.153) is a second order homogeneous ODE in $\mathbf{u}(\xi)$. A system of first order ODEs can be obtained from Eqs. (3.151) and (3.153) by doubling the number of variables. It is given by

$$\xi \mathbf{X}(\xi)_{,\xi} = \mathbf{Z}_p \mathbf{X}(\xi), \tag{3.154}$$

where, $\mathbf{Z}_p$ is the Hamiltonian coefficient matrix expressed as

$$\mathbf{Z}_p = \begin{bmatrix} -\mathbf{E}_0^{-1}\mathbf{E}_1^{\mathrm{T}} + 0.5\mathbf{I}, & \mathbf{E}_0^{-1} \\ \mathbf{E}_2 - \mathbf{E}_1\mathbf{E}_0^{-1}\mathbf{E}_1^{\mathrm{T}}, & \mathbf{E}_1\mathbf{E}_0^{-1} - 0.5\mathbf{I} \end{bmatrix}, \tag{3.155}$$

and the vector $\mathbf{X}$ is given as

$$\mathbf{X}(\xi) = \begin{Bmatrix} \xi^{0.5}\mathbf{u}(\xi) \\ \xi^{-0.5}\mathbf{q}(\xi) \end{Bmatrix}. \tag{3.156}$$

Following the eigenvalue decomposition technique, the solution of Eq. (3.154) for a bounded s-element is obtained as

$$\mathbf{u}(\xi) = \sum_{i=1}^{n} c_i \xi^{\lambda_i} \boldsymbol{\phi}_i^u = \boldsymbol{\Phi}_b^u \xi^{\langle \lambda_b \rangle} \mathbf{c}, \tag{3.157a}$$

$$\mathbf{q}(\xi) = \sum_{i=1}^{n} c_i \xi^{\lambda_i} \boldsymbol{\phi}_i^q = \boldsymbol{\Phi}_b^q \xi^{\langle \lambda_b \rangle} \mathbf{c}, \tag{3.157b}$$

where, $\boldsymbol{\Phi}_b^u$ and $\boldsymbol{\Phi}_b^q$ are sub-matrices of the eigenvector matrix corresponding to the first n eigenvalues. $\langle \lambda_b \rangle$ is a diagonal matrix containing the eigenvalues, given as

$$\langle \lambda_b \rangle = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n), \tag{3.158}$$

and $\mathbf{c}$ is a vector containing the integration constants, given by

$$\mathbf{c} = [c_1, c_2, \ldots, c_n]^{\mathrm{T}}. \tag{3.159}$$

Eliminating the integration constant from Eqs. (3.157a) and (3.157b), we obtain

$$\mathbf{q}(\xi) = \mathbf{\Phi}_b^q \left[\mathbf{\Phi}_b^u\right]^{-1} \mathbf{u}(\xi). \tag{3.160}$$

The nodal force and displacement relationship on the boundary of the s-element can be obtained as

$$\mathbf{q}(\xi = 1) = \mathbf{\Phi}_b^q \left[\mathbf{\Phi}_b^u\right]^{-1} \mathbf{u}(\xi = 1). \tag{3.161}$$

Here, $\mathbf{f}_b = \mathbf{q}(\xi = 1)$ and $\mathbf{u}_b = \mathbf{u}(\xi = 1)$ represent the boundary nodal force and displacement, respectively. It leads to

$$\mathbf{f}_b = \mathbf{K}^e \mathbf{u}_b, \tag{3.162}$$

where,

$$\mathbf{K}^e = \mathbf{\Phi}_b^q \left[\mathbf{\Phi}_b^u\right]^{-1}. \tag{3.163}$$

Here, $\mathbf{K}^e$ is the stiffness matrix of the element. It can be later assembled into a global stiffness matrix in a similar manner as the conventional finite element procedure. Once, the nodal boundary displacements are obtained from the global equilibrium equations, the displacement and stress field within the given sector in the s-element can be obtained as

$$\tilde{\mathbf{u}}(\xi, \eta, \zeta) = \mathbf{N}_u(\eta, \zeta)\mathbf{u}(\xi) = \mathbf{N}_u(\eta, \zeta) \sum_{i=1}^{n} c_i \xi^{\lambda_i} \boldsymbol{\phi}_i^u, \tag{3.164}$$

$$\boldsymbol{\sigma}(\xi, \eta, \zeta) = \mathbf{D} \sum_{i=1}^{n} c_i \xi^{\lambda_i - 1} (\lambda_i \mathbf{B}_1(\eta, \zeta) + \mathbf{B}_2(\eta, \zeta)) \boldsymbol{\phi}_i^u. \tag{3.165}$$

The displacement field can also be expressed in terms of boundary nodal displacement $\mathbf{u}_b$ as

$$\tilde{\mathbf{u}}(\xi, \eta, \zeta) = \mathbf{N}_u(\eta, \zeta) \mathbf{\Phi}_b^u \xi^{\langle \lambda_b \rangle} \left[\mathbf{\Phi}_b^u\right]^{-1} \mathbf{u}_b = \mathbf{N}_v(\xi, \eta, \zeta)\mathbf{u}_b, \tag{3.166}$$

where, $\mathbf{N}_v(\xi, \eta, \zeta)$ is the shape function of the given solid sector in the s-element.

3.2.5 Derivation of SBFE formulation for elastodynamics

The equations of motion in elastodynamics can be obtained by augmenting the Eq. (3.136) with the virtual work δU_ρ due to inertial force, and is given as,

$$\delta U_\rho = \int_V \delta \tilde{\mathbf{u}}^{\mathrm{T}} \rho \tilde{\mathbf{a}} \mathrm{d}V. \tag{3.167}$$

The acceleration field $\tilde{\mathbf{a}}(\xi, \eta, \zeta)$ can be obtained using the same shape function as the displacement field in Eq. (3.166) as

$$\tilde{\mathbf{a}}(\xi, \eta, \zeta) = \mathbf{N}_v(\xi, \eta, \zeta) \ddot{\mathbf{u}}_b. \tag{3.168}$$

Substituting Eqs. (3.166) and (3.168) into Eq. 3.167, we get

$$\begin{aligned} \delta U_\rho &= \delta \mathbf{u}_b^T \int_V \mathbf{N}_v^T(\xi, \eta, \zeta) \rho \mathbf{N}_v(\xi, \eta, \zeta) \mathrm{d}V \ddot{\mathbf{u}}_b \\ &= \delta \mathbf{u}_b^T \mathbf{M}^e \ddot{\mathbf{u}}_b, \end{aligned} \tag{3.169}$$

where, $\mathbf{M}^e$ is the element mass matrix. Substituting the shape function $\mathbf{N}_v(\xi, \eta, \zeta)$ from Eq. (3.166) and $\mathrm{d}V$ from Eq. (3.118) into the mass matrix $\mathbf{M}^e$ and simplifying the expression, we get

$$\mathbf{M}^e = [\mathbf{\Phi}_b^u]^{-T} \int_0^1 \xi^{\langle \lambda_b \rangle} [\mathbf{\Phi}_b^u]^T \mathbf{M}_0 [\mathbf{\Phi}_b^u] \xi^{\langle \lambda_b \rangle} \xi^2 \mathrm{d}\xi \, [\mathbf{\Phi}_b^u]^{-1}, \tag{3.170}$$

where, the coefficient matrix $\mathbf{M}_0$ is given by

$$\mathbf{M}_0 = \sum_e \int_{-1}^{+1} \int_{-1}^{+1} \mathbf{N}_u^T(\eta, \zeta) \rho \mathbf{N}_u(\eta, \zeta) \, |J_b| \, \mathrm{d}\eta \mathrm{d}\zeta. \tag{3.171}$$

Here, the integration is done for each solid sector corresponding to the surface element e in the s-element. The expression in Eq. (3.170) can be rewritten as

$$\mathbf{M}^e = [\mathbf{\Phi}_b^u]^{-T} \mathbf{m} \, [\mathbf{\Phi}_b^u]^{-1}, \tag{3.172}$$

where,

$$\begin{aligned} \mathbf{m} &= \int_0^1 \xi^{\langle \lambda_b \rangle} [\mathbf{\Phi}_b^u]^T \mathbf{M}_0 [\mathbf{\Phi}_b^u] \xi^{\langle \lambda_b \rangle} \xi^2 \mathrm{d}\xi \\ &= \int_0^1 \xi^{\langle \lambda_b \rangle} \mathbf{m}_0 \xi^{\langle \lambda_b \rangle} \xi^2 \mathrm{d}\xi. \end{aligned} \tag{3.173}$$

Each entry of the matrix $\mathbf{m}$ can be integrated analytically as

$$\begin{aligned} m_{ij} &= \int_0^1 \xi^{\lambda_{bi}} m_{0ij} \xi^{\lambda_{bj}} \xi^2 \mathrm{d}\xi \\ &= \frac{m_{0ij}}{\lambda_{bi} + \lambda_{bj} + 3}. \end{aligned} \tag{3.174}$$

Augmenting Eq. (3.136) to include the inertial term δU_ρ, we obtain

$$\delta U_\rho + \delta U_\varepsilon - \delta W = 0. \tag{3.175}$$

Substituting Eqs. (3.138), (3.144) and (3.169) into Eq. (3.175) yields

$$\delta \mathbf{u}_b^T (\mathbf{M}^e \ddot{\mathbf{u}}_b + \mathbf{q}(\xi = 1) - \mathbf{f}_b) - \int_0^1 \delta \mathbf{u}^{\mathrm{T}}(\xi) \left(\mathbf{q}(\xi)_{,\xi} - \sum_e \xi \int_{-1}^{+1} \int_{-1}^{+1} \mathbf{B}_2^{\mathrm{T}}(\eta, \zeta) \boldsymbol{\sigma} \left| \mathbf{J}_b \right| \mathrm{d}\eta d\zeta \right) \mathrm{d}\xi = 0. \tag{3.176}$$

Invoking the arbitrariness of the virtual displacement at the boundary and within the s-element, we obtain

$$\mathbf{f}_b = \mathbf{M}^e \ddot{\mathbf{u}}_b + \mathbf{q}(\xi = 1), \tag{3.177}$$

$$\mathbf{q}(\xi)_{,\xi} = \sum_e \xi \int_{-1}^{+1} \int_{-1}^{+1} \mathbf{B}_2^{\mathrm{T}}(\eta, \zeta) \boldsymbol{\sigma} \left| \mathbf{J}_b \right| \mathrm{d}\eta d\zeta. \tag{3.178}$$

Eq. (3.178) can be formulated into the scaled boundary equation given in Eq. (3.153), whose solution is derived in Section 3.2.4. Substituting Eqs. (3.161) and (3.163) into Eq. (3.177) leads to the following equation of motion in elastodynamics in three dimensions:

$$\mathbf{M}^e \ddot{\mathbf{u}}_b + \mathbf{K}^e \mathbf{u}_b = \mathbf{f}_b. \tag{3.179}$$

3.3 Summary

In this chapter, the formulations for two and three dimensional elastostatics and elastodynamics were presented and derived using the principle of virtual work in the framework of SBFEM.

In the two dimensional formulation, each element is discretised by 2-node line elements along its boundary. There is no restriction on the number of nodes and edges in

each element. The only requirement is to have a scaling centre from which the whole boundary of the element is directly visible.

For three dimensional problems, the problem domain can be subdivided into arbitrary shaped polyhedral elements. Each element can have arbitrary number of nodes, edges and faces and only needs to satisfy the star convexity criterion.

In the following chapters, it is shown that the shape functions of polyhedrons can be utilised in handling the octree meshes directly in the framework of the SBFEM, without the need for special treatments of hanging nodes (as required by the conventional FEM).

Chapter 4

A parallel explicit solver for elastodynamic problems utilising octree mesh

In this chapter[1], a massively parallel implementation of an explicit time integration scheme is formulated for elastodynamic problems exploiting the hierarchical grid-based mesh containing octree cells. The chapter is organised as follows.

In Section 4.1, the explicit-dynamics is introduced in the context of SBFEM. In Section 4.2, a brief description of octree-based mesh and unique cell patterns constructed in the framework of SBFEM is provided. Section 4.3 introduces the central difference method (CDM) – a commonly used explicit time integration scheme in elastodynamics. The implementation of CDM using octree mesh, mass lumping strategies, and implementation of octree cell patterns are discussed. In Section 4.4, the parallel implementation of the explicit solver, including mesh-partitioning and data communication strategies, is discussed. In Section 4.5, the implementation of the explicit solver is veri-

[1]The contents of this chapter were published in [337]. I acknowledge the contribution of Dr Junqi Zhang for generating the octree mesh pattern data, and other authors for their valuable suggestions.

fied through modal and transient analyses of benchmark problems. Then, the parallel performance of the solver is demonstrated utilising three numerical examples of varying complexity of the geometric models and the number of degrees of freedom (DOFs). General remarks and summary are given in Section 4.6.

4.1 Introduction

The elastodynamic problems can be classified as wave propagation and inertial problems. Many engineering applications involving impact loading on structures, shock blasts, etc., fall into the category of wave propagation problems [338, 339]. In these problems, the structural response is dominated by the intermediate and high-frequency modes throughout the time-span of interest. Structural health monitoring (SHM) [340, 341] is an important application of wave-propagation in an elastic medium, where the structures are subjected to ultrasonic guided waves in order to detect possible damage. Here, very high-frequency signals are employed to detect flaws on smaller scales. The simulation of such problems generally requires a very fine discretisation spatially and temporally so as to capture the structural response due to high-frequency contents in the loading.

As introduced in Section 2.2.1, the explicit time integration schemes are particularly suitable for solving the wave propagation problems [11, 272]. These methods allow displacements at a given time step to be obtained from the displacements, velocities, and accelerations at the previous time step. The computational cost per time step and storage requirement in explicit methods are usually much lesser than the implicit time integration schemes when the mass lumping strategy is employed. However, the explicit methods are only conditionally stable and have a very small time step size restriction to ensure numerical stability. In most cases, the accuracy requirements in the explicit method are far exceeded by its stability requirements. Due to this, the effectiveness of

the approach is generally limited to wave propagation problems of very short duration. For the dynamics problems of moderate duration, e.g. seismic response of structures, the explicit methods may not be effectively utilised.

In the framework of the scaled boundary finite element method (SBFEM) for dynamic analyses, the implicit time integration schemes have been employed extensively [72, 121, 342–345]. In many of these publications, the higher-order shape functions of the elements have been exploited, which require complex procedures and additional non-physical degrees of freedom. In the context of explicit time integration schemes, Gravenkamp et al. [346] recently developed a lumped mass formulation for simple two-dimensional problems.

To accurately capture the structural dynamic behaviour, many engineering applications require three-dimensional modelling of the problem domain and possibly a very large number of degrees of freedom. As the SBFEM provides more flexibility in the meshing paradigm, its features can be utilised for automatic mesh generation of complex geometries and the development of robust and efficient parallel explicit solvers.

In the present chapter, the implementation of a massively parallel explicit solver is formulated exploiting the features of balanced octree meshes in the framework of the scaled boundary finite element method (SBFEM). Such meshes can be directly utilised using the SBFE formulation without the need for special treatments of hanging nodes. In a balanced octree mesh, a recursive partition of the three-dimensional space into smaller octants is performed to approximate the geometry such that the adjacent elements can only differ by one octree level. The advantage of such meshes is that only a limited number of octree patterns are present; therefore, the element stiffness and mass matrices can be pre-computed and have much smaller memory requirements as compared to the individual (un-grouped) element matrices required for the entire structure.

To perform the computationally most demanding step – the nodal force vector cal-

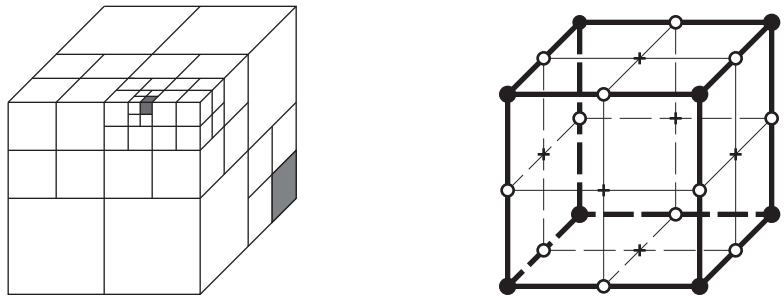
culation – a pattern-by-pattern (PBP) approach is proposed. Here, the octree patterns are exploited to group all the elements of the same shape (or nodal pattern). It allows performing the dense matrix operations associated with the nodal force vector calculation, boosting the performance of the solver.

The PBP approach supports large-scale parallelisation, which is carried out using the mesh-partitioning strategy. Here, the domain-mesh is divided into several partitions and assigned to individual workers (or computer processes). Each worker assembles the nodal force vector within an individual partition, and the data exchange among individual partitions is only needed at the interfaces of the neighbouring partitions. The nodal displacement vector for the next time step is computed locally on each partition by extending the diagonal (lumped) mass matrix formulation developed by Gravenkamp et al. [346] to three dimensions. Through several numerical examples, it is shown that the developed explicit solver can efficiently solve very large structural dynamics problems and allows massively parallel computation scaling to thousands of workers.

4.2 Octree-based mesh

The hierarchical grid-based mesh containing octree cells was briefly introduced in Chapters 1 and 2. In this section, we discuss the categorisation of the octree cells into unique patterns.

In octree-based meshes, a uniform background mesh is first generated, followed by a recursive octree decomposition of the problem domain until some specified stopping criteria are met [74]. This technique provides fast mesh size transition; however, it creates several hanging nodes in the process, which are difficult to handle using conventional finite element methods. In the SBFEM framework, the difficulty associated with the hanging nodes can be easily overcome by using the shape functions of polyhedrons. An



(a) A balanced octree mesh

(b) A balanced octree cell indicating all possible corner (•), mid-edge (○) and center (+) nodes

Figure 4.1: Example of balanced octree mesh and octree cell patterns [1]

example of octree mesh is shown in Fig. 4.1a, in which each octree cell is modelled as a scaled boundary finite element.

4.2.1 Balanced octree mesh

An octree mesh is balanced when the length ratio between the two adjacent cells is 2 (known as the 2 : 1 rule). A balanced octree cell following the 2 : 1 rule is shown in Fig. 4.1b, where all the possible locations of the nodes are indicated. Each face of the octree cell can have a total of $2^5 = 32$ patterns (ideally). To allow smooth mesh size transition, the face discretisation of the octree cells can be restricted to 6 unique node arrangements considering rotations. These arrangements depending on the number of mid-edge nodes n_{h} and their locations are shown in Fig. 4.2. It is noticed that only the rectangular and isosceles triangles are used for the face discretisation. The shape functions, the location of integration points on surface elements, and the order of quadrature can be chosen following the guidelines of standard isoparametric finite elements [1, 347]. It is explained in the following sections that using restricted face discretisation of the octree cells allows creating unique octree cell patterns, which leads to efficient utilisation of computational resources during the analysis.

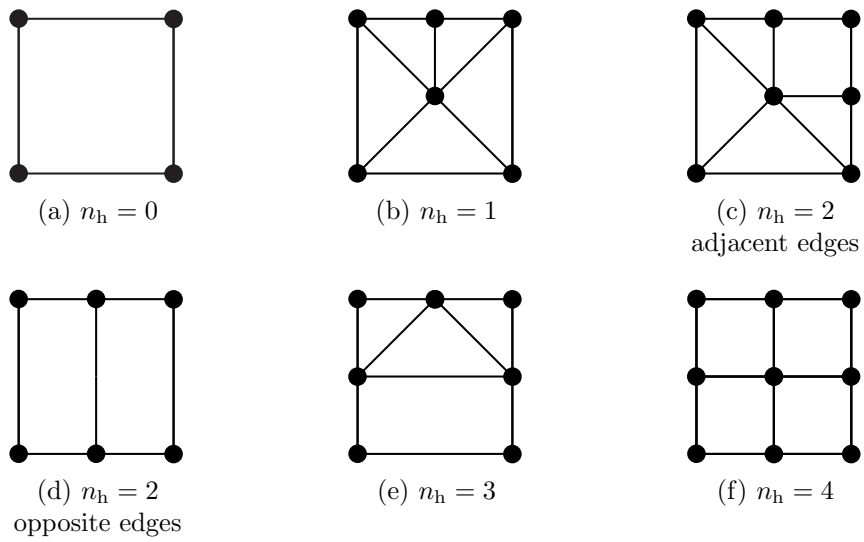


Figure 4.2: Face discretisation patterns for an octree cell. The number of hanging nodes on the edges of each face is indicated by n_h.

4.2.2 Unique octree cell patterns

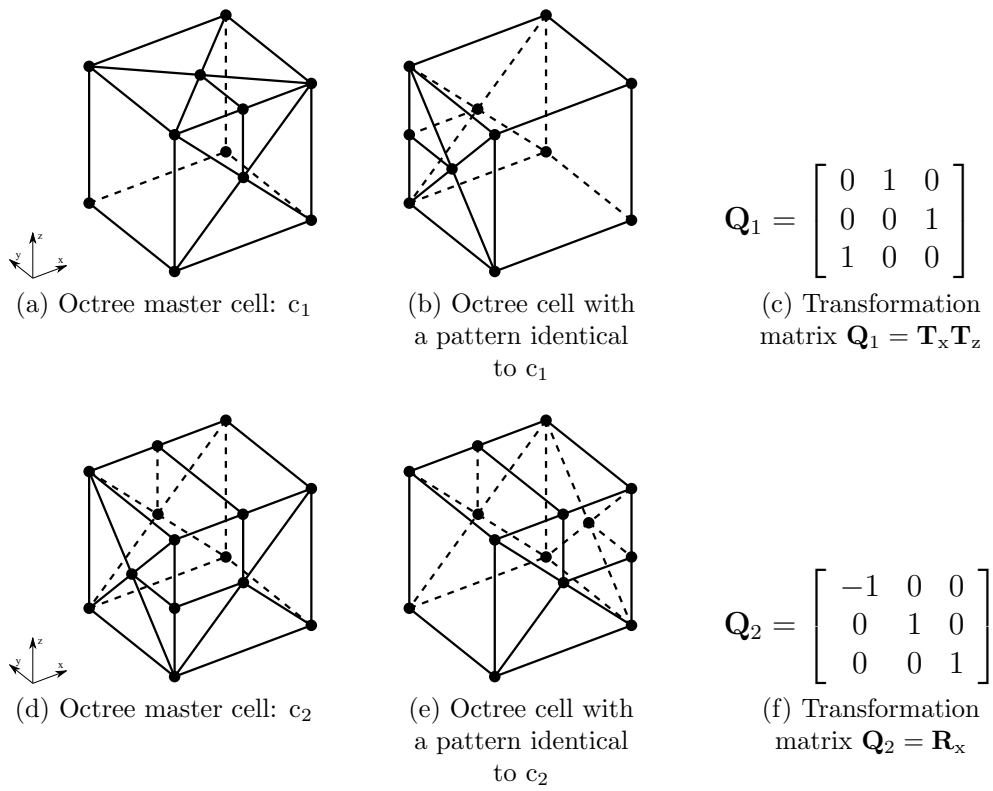
As explained by Song [1], there are total of $2^{12} = 4096$ patterns possible in a balanced octree cell. These 4096 patterns can be further reduced to 144 unique cases considering rotation and mirroring operations. The 90 degree rotation matrices about the x, y and z axes are

$$\mathbf{T}_\mathrm{x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}, \quad \mathbf{T}_\mathrm{y} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}, \quad \mathbf{T}_\mathrm{z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.1}$$

and the mirror matrices are

$$\mathbf{R}_\mathrm{x} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{R}_\mathrm{y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{R}_\mathrm{z} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}. \tag{4.2}$$

The transformation matrices can be combined to form 48 unique transformations of an octree cell. These 48 transformations can also be obtained by filling 0 and ± 1 into a 3×3 empty matrix such that there is exactly one non-zero value (± 1) in each row and



(a) Octree master cell: c_1 (b) Octree cell with a pattern identical to c_1 (c) Transformation matrix $\mathbf{Q}_1 = \mathbf{T}_x\mathbf{T}_z$

(d) Octree master cell: c_2 (e) Octree cell with a pattern identical to c_2 (f) Transformation matrix $\mathbf{Q}_2 = \mathbf{R}_x$

Figure 4.3: Transformation of octree cells

column. Then, the number of transformations can be obtained as $3 \times 2 \times 1 \times 2^3 = 48$. Using one of these 48 transformations, each of the 4096 patterns can be transformed into one of the 144 unique patterns.

Some sample transformations are illustrated in Fig. 4.3. Two octree cells with an identical pattern are considered, as shown in Fig. 4.3a-b. It can be noticed that the octree cell in Fig. 4.3b can be transformed into the corresponding master cell in Fig. 4.3a by rotating 90 degrees about the x-axis, and then rotating 90 degrees about the z-axis following the right-hand rule. The corresponding transformation matrix can be given as $\mathbf{Q}_1 = \mathbf{T}_x\mathbf{T}_z$. Another example is shown in Fig. 4.3d-e, where the octree cell shown in Fig. 4.3e can be transformed into the one shown in Fig. 4.3d by mirroring about the y-z plane using the transformation matrix given by $\mathbf{Q}_2 = \mathbf{R}_x$.

4.3 Explicit time integration

This section discusses the formulation of an explicit time integration scheme, followed by SBFEM-specific mass lumping technique and implementation of octree cell patterns. This lays the foundation for the parallel implementation of the explicit solver discussed in the next section.

The governing equilibrium equation of a structural system can be given by

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{f}^{\text{int}} = \mathbf{f}^{\text{ext}}, \tag{4.3}$$

where, $\mathbf{M}$ is the discrete mass matrix, $\mathbf{C}$ is the viscous damping matrix, $\mathbf{f}^{\text{int}}$ is the internal force vector due to element stresses and $\mathbf{f}^{\text{ext}}$ is the external load vector. $\mathbf{u}$, $\dot{\mathbf{u}}$, and $\ddot{\mathbf{u}}$ are the nodal displacement, velocity, and acceleration vectors, respectively. For a linear structural system, $\mathbf{f}^{\text{int}} = \mathbf{K}\mathbf{u}$, where $\mathbf{K}$ is the stiffness matrix.

The explicit time integration technique is used for the temporal integration of the governing equilibrium equation given in Eq. (4.3). In this technique, the displacement vector determined at the end of the load step depends on the mechanical loads or the material properties calculated at the previous load step. This avoids the need for iterative calculations within the load-step.

4.3.1 Central difference method

The most widely used explicit time integration scheme in structural dynamics problems is the central difference method (CDM), as introduced in Section 2.2.1. The scheme can be obtained using the finite difference equations that relate the accelerations, velocities and displacements. At the j-th time step, the velocity $\dot{\mathbf{u}}_j$ and acceleration $\ddot{\mathbf{u}}_j$ can be obtained from the central difference formulas given by

$$\dot{\mathbf{u}}_j = \frac{1}{2\Delta t}(\mathbf{u}_{j+1} - \mathbf{u}_{j-1}), \tag{4.4a}$$

$$\ddot{\mathbf{u}}_j = \frac{1}{\Delta t^2}(\mathbf{u}_{j+1} - 2\mathbf{u}_j + \mathbf{u}_{j-1}), \tag{4.4b}$$

where, Δt denotes the time step size. Substituting Eqs. (4.4) into Eq. (4.3), we obtain

$$\frac{1}{\Delta t^2}\mathbf{M}(\mathbf{u}_{j+1} - 2\mathbf{u}_j + \mathbf{u}_{j-1}) + \frac{1}{2\Delta t}\mathbf{C}(\mathbf{u}_{j+1} - \mathbf{u}_{j-1}) + \mathbf{f}_j^{\text{int}} = \mathbf{f}_j^{\text{ext}}. \tag{4.5}$$

Solving for the displacement at the next time step $\mathbf{U}_{j+1}$, we get

$$\left(\frac{1}{\Delta t^2}\mathbf{M} + \frac{1}{2\Delta t}\mathbf{C}\right)\mathbf{u}_{j+1} = \frac{2}{\Delta t^2}\mathbf{M}\mathbf{u}_j - \left(\frac{1}{\Delta t^2}\mathbf{M} - \frac{1}{2\Delta t}\mathbf{C}\right)\mathbf{u}_{j-1} + \mathbf{f}_j^{\text{ext}} - \mathbf{f}_j^{\text{int}}. \tag{4.6}$$

If mass-proportional damping is considered, then the damping matrix is expressed as

$$\mathbf{C} = \alpha\mathbf{M}, \tag{4.7}$$

where, α is a constant. Eq. (4.6) can now be written as

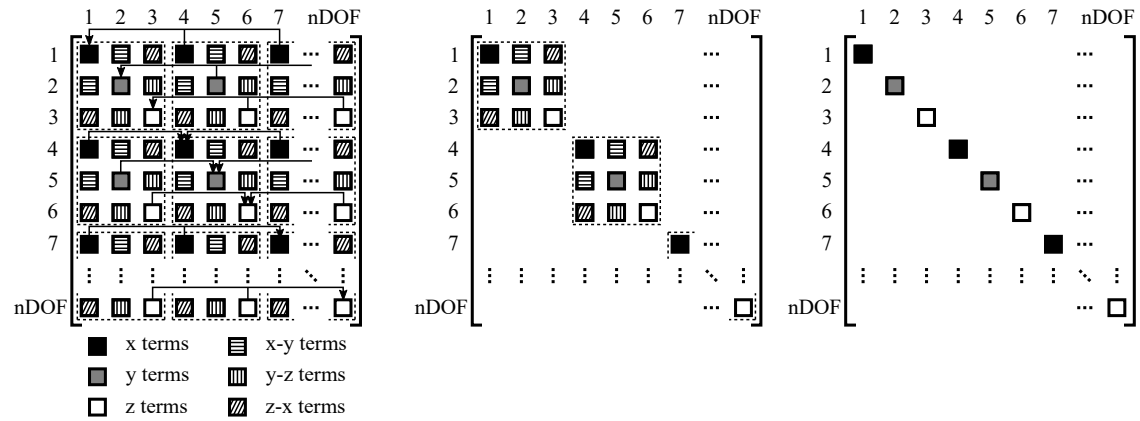
$$\mathbf{u}_{j+1} = \frac{2\mathbf{u}_j - (1 - 0.5\alpha\Delta t)\,\mathbf{u}_{j-1} + \Delta t^2\mathbf{M}^{-1}\left(\mathbf{f}_j^{\text{ext}} - \mathbf{f}_j^{\text{int}}\right)}{1 + 0.5\alpha\Delta t}. \tag{4.8}$$

It is observed from Eq. (4.8) that the displacement at the next step can be obtained from the displacement of the current and previous time steps. For this reason, the CDM is categorised as explicit integration technique. The initial condition at $j = 0$ can be incorporated by defining a displacement $\mathbf{u}_{-1}$ at fictitious point $j = -1$, given as

$$\mathbf{u}_{-1} = \mathbf{u}_0 - \Delta t\dot{\mathbf{u}}_0 + \frac{\Delta t^2}{2}\ddot{\mathbf{u}}_0. \tag{4.9}$$

Using the CDM, very large structural systems can be solved efficiently. However, the method can only be effectively utilised if a diagonal mass matrix $\mathbf{M}$ obtained by mass lumping is incorporated in Eq. (4.8). This makes the inversion of the mass matrix trivial. The mass lumping procedure in the SBFEM framework is explained in Section 4.3.2. Also, the calculation of the internal force vector $\mathbf{f}^{\text{int}}(\mathbf{u}_j)$ requires matrix-vector multiplication, which increasingly affects the computational performance of the CDM as the size of the structural system increases. This issue can be efficiently handled in the SBFEM framework by exploiting the octree cell patterns, as explained in Section 4.3.3.

One major drawback of the CDM (and other explicit methods in general) is the conditional stability. Therefore, the numerical scheme is stable only if the time step size Δt is smaller than a critical time step size Δt_{cr}. The critical time step size can be defined



(a) Fully populated mass matrix $\mathbf{M}$ (b) Block-diagonal mass matrix (c) Diagonal mass matrix $\mathbf{M}_{\text{diag}}$

Figure 4.4: Lumping of the mass matrix

as the least traversal time required for an acoustic wave to traverse any element within the problem domain. The exact value of Δt_{cr} involves the calculation of the maximum natural frequency of the whole structure, which can be computationally expensive. In the present implementation, the stability limit is estimated by the maximum frequency on element level, as explained later in Section 4.3.3.

4.3.2 Mass lumping in SBFE framework

Mass lumping strategies are often employed to diagonalise the mass matrix, so that the CDM is efficiently implemented. It can be inferred from Eq. (4.8) that if the mass matrix $\mathbf{M}$ is available in a diagonal form, the matrix inversion becomes trivial. Therefore, the need for solving the system of linear equations simultaneously is avoided.

Several mass lumping schemes have been reported in Refs. [348, 349] in the context of the finite element methods (FEM) and spectral element methods (SEM). As compared to FEM and SEM, the x, y, and z displacement components inside the scaled boundary finite elements are coupled and the displacement interpolations are obtained semi-analytically. Therefore, only those components of the mass matrix that correspond to DOFs in the same coordinate direction are summed up in the present mass lumping

scheme, as illustrated in Fig. 4.4a. The procedure is repeated for all the components corresponding to x, y, and z directions. This creates a block-diagonal matrix, shown in Fig. 4.4b, from which a diagonal mass matrix $\mathbf{M}_{\text{diag}}$ can be extracted (Fig. 4.4c). The off-diagonal terms which sum up to zero are ignored while the total mass of the structural system is conserved. Further details and verification of the present mass lumping technique can be found in [346] for two dimensional problems.

Once the diagonal mass matrix $\mathbf{M}_{\text{diag}}$ is available, the expression for the displacement vector in Eq. (4.8) can be re-written as

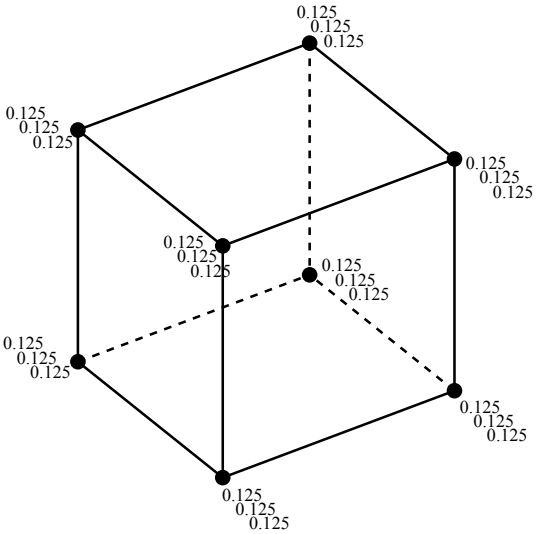
$$\mathbf{u}_{j+1} = \frac{2\mathbf{u}_j - (1 - 0.5\alpha\Delta t)\,\mathbf{u}_{j-1} + \Delta t^2 \mathbf{M}_{\text{diag}}^{-1}\left(\mathbf{f}_j^{\text{ext}} - \mathbf{f}_j^{\text{int}}\right)}{1 + 0.5\alpha\Delta t}. \tag{4.10}$$

The lumped mass (as ratios to the total mass of the octree cells) of a few octree patterns mentioned in Section 4.2.2 are shown in Fig. 4.5 and Fig. 4.6 for Poisson's ratios 0 and 0.3 respectively. At each node, the values are presented for x, y, and z directions.

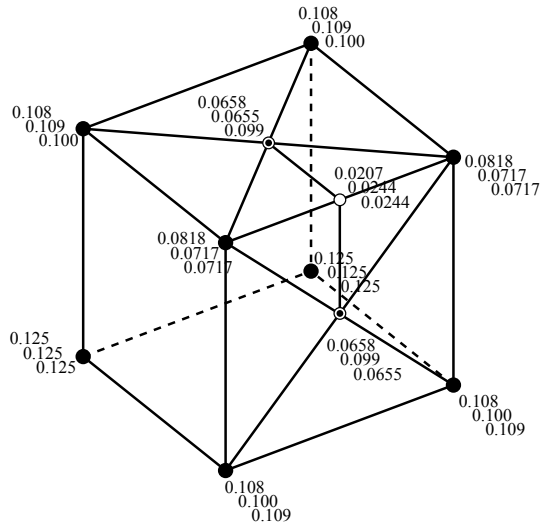
4.3.3 The octree-based pattern-by-pattern (PBP) approach

The calculation of internal force vector $\mathbf{f}_j^{\text{int}}$ at each time step requires multiplication of the global stiffness matrix $\mathbf{K}$ and displacement vector $\mathbf{u}_j$. As the size of the structural system increases, the direct multiplication methods tend to be excessively computationally demanding due to the requirement of assembly and storage of a large stiffness matrix. Several techniques such as Sparse-Matrix-Vector (SpMV) product [350, 351] and Element-by-Element (EBE) approach [352, 353] have been proposed to handle this issue. In this section, a pattern-by-pattern (PBP) approach is proposed, where the octree cell patterns are exploited in order to improve computational performance, reduce memory requirements, and provide scalability to the solver.

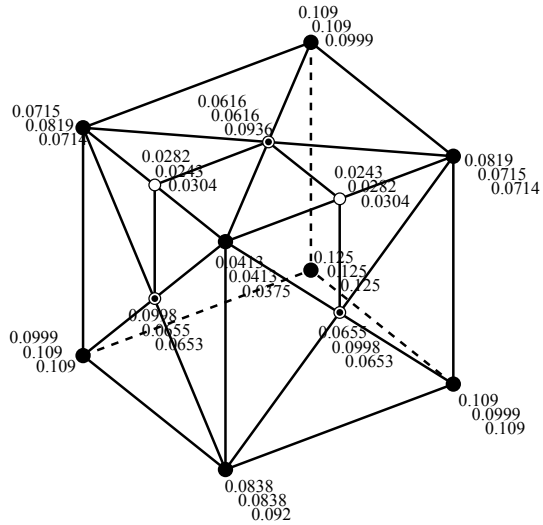
For each unique octree cell pattern p, we can define a master cubic cell (or element) of unit edge length ($L = 1$) and material properties (Young's modulus $E = 1$ and mass



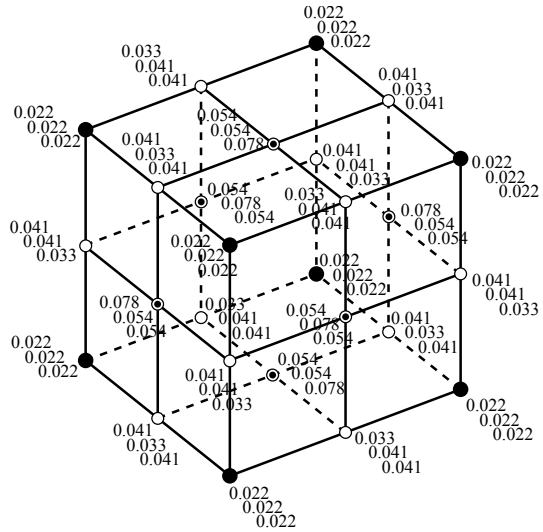
(a) Element with 8 nodes



(b) Element with 11 nodes

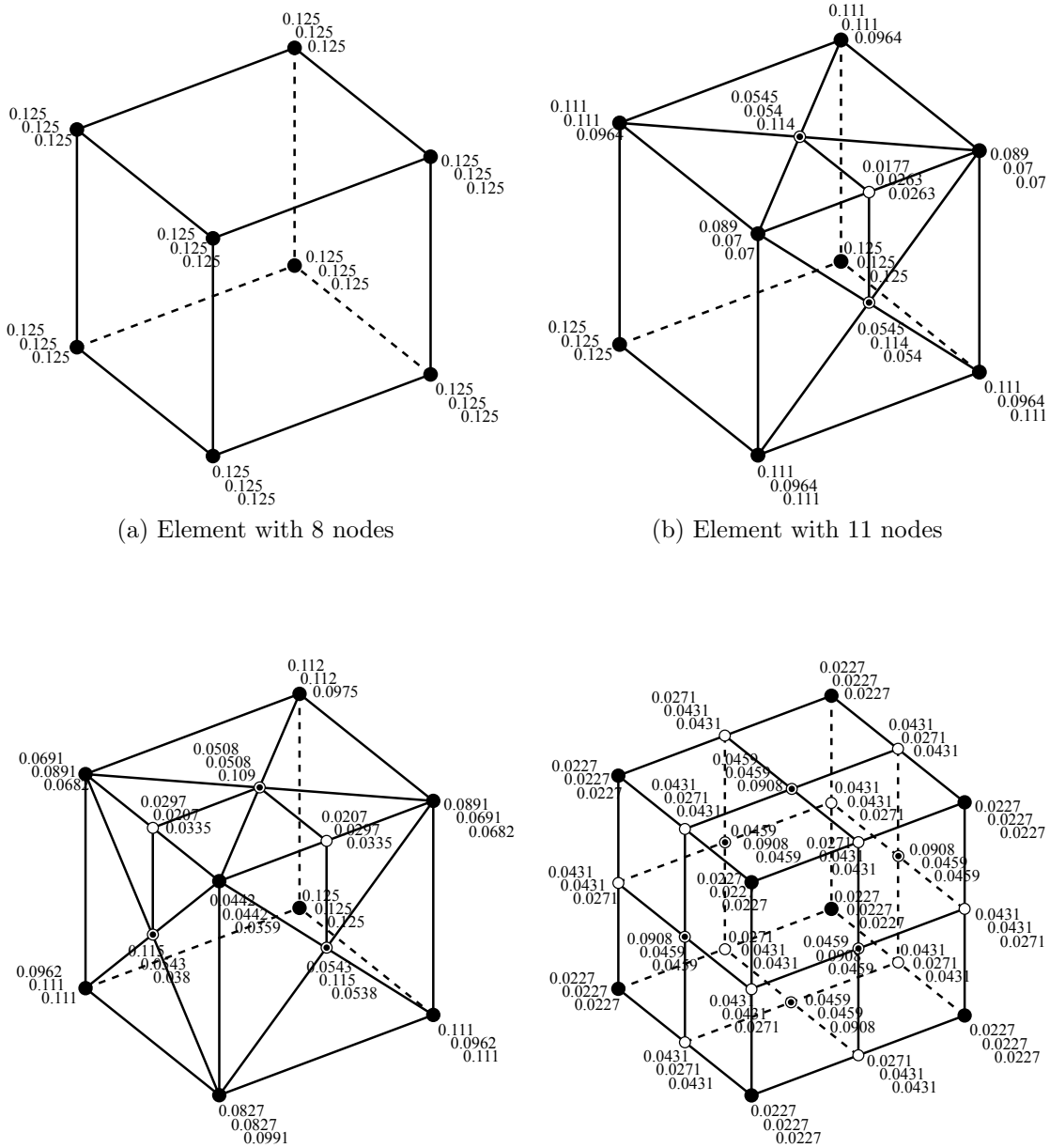


(c) Element with 13 nodes



(d) Element with 26 nodes

Figure 4.5: Lumped mass with Poisson's ratio $\nu = 0$



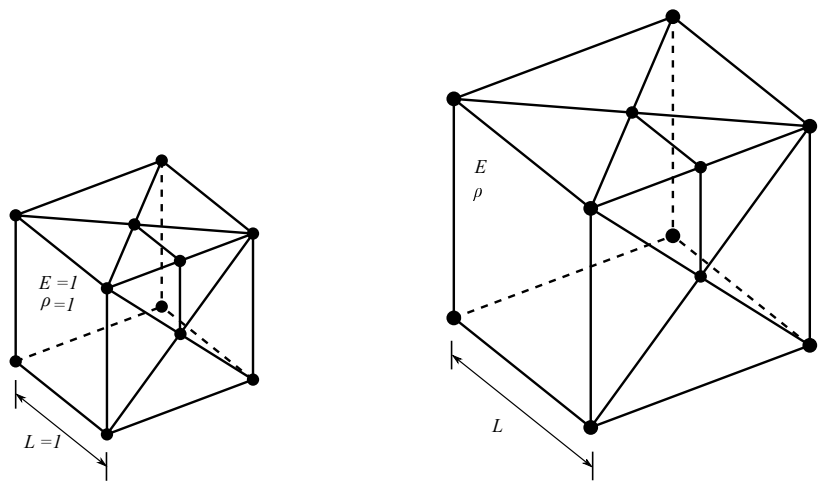
(a) Element with 8 nodes

(b) Element with 11 nodes

(c) Element with 13 nodes

(d) Element with 26 nodes

Figure 4.6: Lumped mass with Poisson's ratio $\nu = 0.3$



(a) Master element (b) Scaled element

Figure 4.7: Scaling of an octree element

density $\rho = 1$) for a given Poisson's ratio, as illustrated in Fig. 4.7a. The stiffness and mass matrices of each master element can be pre-computed and stored with little memory requirement. Depending on the edge size and material properties of a given element e of the same pattern (Ref Fig. 4.7b), its stiffness and mass matrices can be obtained by scaling the corresponding matrices of the master element by a suitable factor. This procedure is explained as follows.

The stiffness matrix of the given element is proportional to its Young's modulus E and edge length L. On the other hand, the mass matrix of the element is proportional to the mass density ρ and the cube of edge length L^3. Therefore, the mass $\mathbf{M}^e$ and stiffness $\mathbf{K}^e$ matrices of the given element can be derived easily from a master element of the same pattern and Poisson's ratio as

$$\mathbf{M}^e = \rho L^3 \mathbf{M}^p = r^{ep} \mathbf{M}^p, \tag{4.11a}$$

$$\mathbf{K}^e = EL\mathbf{K}^p = s^{ep} \mathbf{K}^p, \tag{4.11b}$$

where, $\mathbf{M}^p$ and $\mathbf{K}^p$ are mass and stiffness matrices respectively of the master element. In the case where materials with different Poisson's ratios are present, the master element matrices have to be computed for each material separately.

Using the EBE approach, the internal force vector at the j-th time step $\mathbf{f}_j^{\text{int}}$ can be calculated without the assembly of the global stiffness matrix as

$$\mathbf{f}_j^{\text{int}} = \sum_e \mathbf{K}^e \mathbf{u}_j^e, \tag{4.12}$$

where, $\sum$ represents the assembly process as per the element connectivity. $\mathbf{K}^e$ and $\mathbf{u}_j^e$ are the stiffness matrix and the nodal displacement vector, respectively, for the eth element. Augmenting the EBE technique, a pattern-by-pattern (PBP) approach is formulated, where the nodal force vectors can be calculated as

$$\mathbf{f}_j^{\text{int}} = \sum_{p=1}^{N_P} \mathbf{F}_j^p = \sum_{p=1}^{N_P} \mathbf{K}^p \left[\begin{array}{cccc} s^{1p}\mathbf{u}_j^{1p}, & s^{2p}\mathbf{u}_j^{2p} & \cdots & s^{N_E p}\mathbf{u}_j^{N_E p} \end{array} \right]. \tag{4.13}$$

Here, N_P is the number of unique patterns in the octree mesh and N_E is the number of elements of pattern p. For a given element e of pattern p, $\mathbf{K}^p$ is the stiffness matrix of the master element, $\mathbf{u}_j^{ep}$ is the nodal displacement vector and s^{ep} is the factor proportional to the edge lengths and Young's modulus, as given in Eq (4.11).

In order to utilise the unique octree patterns with $N_P = 144$, a transformation is carried out using the transformation matrix $\mathbf{Q}$ (see: Section 4.2.2) on nodal displacement vector $\mathbf{u}_j^{ep}$ prior to calculating the internal force matrix $\mathbf{F}_j^p$. Then, for each pattern, the internal force matrix $\mathbf{F}_j^p$ is obtained using matrix-multiplication operation given in Eq. (4.13), which is then transformed back using the inverse of transformation matrix $\mathbf{Q}^{-1}$. The obtained internal force matrix $\mathbf{F}_j^p$ is assembled into the internal force vector $\mathbf{f}_j^{\text{int}}$ using the bincount operation in Python (available in Numpy library). The bincount operation performs a weighted assembly to obtain the internal force vector $\mathbf{f}_j^{\text{int}}$, where it counts the number of occurrences of each DOF in the element connectivity array and assigns each entry of the internal force matrix $\mathbf{F}_j^p$ as corresponding weights.

It is noticed that majority of the operations is recast to carry out pattern-level dense matrix computations, given in Eq. (4.13). This results in a much better cache utilisation, and gives a significant performance boost to the explicit solver by aiding the relevant memory-bandwidth bound operations.

The octree patterns can also be exploited to efficiently obtain the critical time step Δt_{cr} of the problem domain. From the definition, Δt_{cr} is given by

$$\Delta t_{\text{cr}} = \frac{2}{\omega_{\text{max}}}, \tag{4.14}$$

where, ω_{max} is the natural frequency of the most critical element [347, 354]. Considering Eq. (4.11), the critical time step size Δt^e_{cr} of an element e can be calculated from the known time step of the master element Δt^p_{cr} as

$$\Delta t^e_{\text{cr}} = \sqrt{\frac{\rho}{E}} L \Delta t^p_{\text{cr}}. \tag{4.15}$$

Therefore, the critical time step size Δt_{cr} of the problem domain can be estimated based on the eigen-frequency of the most critical element as

$$\Delta t_{\text{cr}} = \min(\Delta t^e_{\text{cr}}). \tag{4.16}$$

4.4 Parallel explicit solver for elasto-dynamics

In this section, we discuss the parallel implementation of the explicit solver using mesh partitioning strategy. Further, we also discuss the CPU time and efficiency measurements performed during the analysis.

4.4.1 Mesh partitioning strategy

The finite-element based explicit solvers are the simple logical structure that can be exploited for large-scale parallelisation. In the present work, parallelisation based on the mesh-partitioning strategy (or subdomain-by-subdomain like formulation) – introduced in Section 2.4.3 – is implemented. Here, the problem domain is divided into a set of smaller partitions and each partition is assigned to the individual workers (or computer processes). The parallelisation is achieved by the individual workers running a single computer program and carrying out the simultaneous local computations over the assigned partitions. The data at the interface of neighbouring partitions are synchronised

through explicit message passing.

The Message Passing Interface or MPI-based parallel programming model (Ref: Section 2.4.2) is utilised for implementing the mesh-partitioning strategy. The computer program follows the SPMD (single-program, multiple-data) approach, where different workers execute the same command on different data sets (or mesh-partitions) simultaneously. The message-passing algorithm (incorporated using the MPI library) is particularly suitable for the SPMD paradigm and allows efficient data communication among the workers in distributed memory systems.

The present thesis makes use of the MPI library using the Python package MPI4Py [81, 82, 305]. The MPI4Py is implemented on top of the MPI-1/2/3 specification and exposes an API that grounds on the standard MPI-2 C++ bindings. The MPI implementation is available from the open-source library Open-MPI 4.0. The MPI4Py supports direct (point-to-point and collective) communication of any object (e.g. Numpy arrays) that implements the Python buffer interface and allows access in the C side to a contiguous memory buffer (e.g. address and length) containing the relevant data. This feature enables the communication of multidimensional numeric arrays (e.g. interfacial nodal force vectors) with negligible overhead.

All the parallel simulations in this thesis have been carried out on Gadi − a supercomputing facility provided by NCI, Canberra. Gadi's network interconnect provides high throughput and little network latency that makes the direct communication (Ref: Fig. 2.5) among the workers more suitable; therefore, the same is utilised in the current work.

4.4.2 The explicit solver

The in-house explicit solver is written in Python 3.7 and implements the MPI library for information exchange among the workers. The solver starts with creating a set of MPI processes (a set of computer processes or workers) that are bind to the cores/socket

of the computer hardware. For optimal computational performance, keeping the same number of MPI processes as the number of available cores is recommended. Also, the number of mesh-partitions is kept the same as the number of MPI processes. With this configuration, each MPI process carries out computation on each mesh partition using exactly one core.

The pseudo-code of the parallel explicit solver is presented in Algorithm 4.1. In the SPMD paradigm, the same code is executed by all MPI processes; however, each MPI process only loads the partitioned mesh-data file assigned to it. It is noted that all the loaded variables are critical for analysis and remain constant during the analysis. A barrier is used so that all MPI processes start the analysis at the same time after initialisation. At each time step j, the internal force vector $\mathbf{f}_j^{\text{int}}$ is calculated locally in each partition using the octree-based PBP approach, explained in Section 4.3.3. The local internal force vector is divided into two parts – one corresponding to the nodes inside each partition (internal DOFs) and those shared with other neighbouring partitions (interfacial DOFs). The values of the latter part are exchanged among the neighbouring partitions through direct communication, as illustrated in Fig. 2.5. Then, the internal force vector $\mathbf{f}_j^{\text{int}}$ is updated by the contributions of the neighbouring partitions. As the size of the interface is relatively small compared to the total size of the partition, this operation can be done in a relatively short time. Once the internal force vector is calculated, the local portion of displacement of the next step $\mathbf{u}_{j+1}$ is calculated using the central difference equation given in Eq. (4.10). Then, the Dirichlet boundary conditions are applied, and the notations of the displacement variables are adjusted before moving to the next time step.

4.4.3 CPU time measurements and scaling

In this section, we discuss the CPU time measurements and the speed-up and efficiency calculations performed for parallel computing.

Algorithm 4.1 Pseudo-code for the parallel explicit solver

1: Start MPI process; Create MPI communicator object: Comm

2: Load partitioned mesh-data file

3: Use barrier: Comm.barrier()

4: **for** $j = 1 : N_{Tmax}$ **do**

5: Initialise internal force vector: $\mathbf{f}_j^{\text{int}} = \{\mathbf{0}\}$

6: **for** $p = 1 : N_P$ **do**

7: Calculate $\mathbf{F}_j^p$ of pth pattern:

$$\mathbf{F}_j^p = \mathbf{K}^p \left[\; s^{1p}\mathbf{u}_j^{1p}, \quad s^{2p}\mathbf{u}_j^{2p} \quad \cdots \quad s^{N_E p}\mathbf{u}_j^{N_E p} \; \right]$$

8: **end for**

9: Assemble the $\mathbf{f}_j^{\text{int}}$ using bincount summation: $\mathbf{f}_j^{\text{int}} = \sum_{p=1}^{N_P} \mathbf{F}_j^p$

10: Calculate force vectors $\mathbf{f}_{j,\text{intfc}}^{\text{int}}$ at the mesh partition interfaces

11: **for** $m = 1 : N_{\text{nbr}}$ **do**

12: Send $\mathbf{f}_{mj,\text{intfc}}^{\text{int}}$ to m^{th} neighbouring partition through non-blocking communication: Comm.Isend

13: **end for**

14: **for** $m = 1 : N_{\text{nbr}}$ **do**

15: Receive $\mathbf{f}_{mj,\text{intfc}}^{\text{int}}$ from m^{th} neighbouring partition through blocking communication: Comm.Recv

16: **end for**

17: Update the internal force vectors based on received data: $\mathbf{f}_j^{\text{int}}$

18: Calculate external force $\mathbf{f}_j^{\text{ext}}$ at the current time step

19: Calculate next step displacement $\mathbf{u}_{j+1}$ using Eq. (4.10)

20: Apply Dirichlet boundary condition given by $\boldsymbol{\Gamma}_{\text{D}}$

21: $\mathbf{u}_{j-1} = \mathbf{u}_j$; $\mathbf{u}_j = \mathbf{u}_{j+1}$

22: **end for**

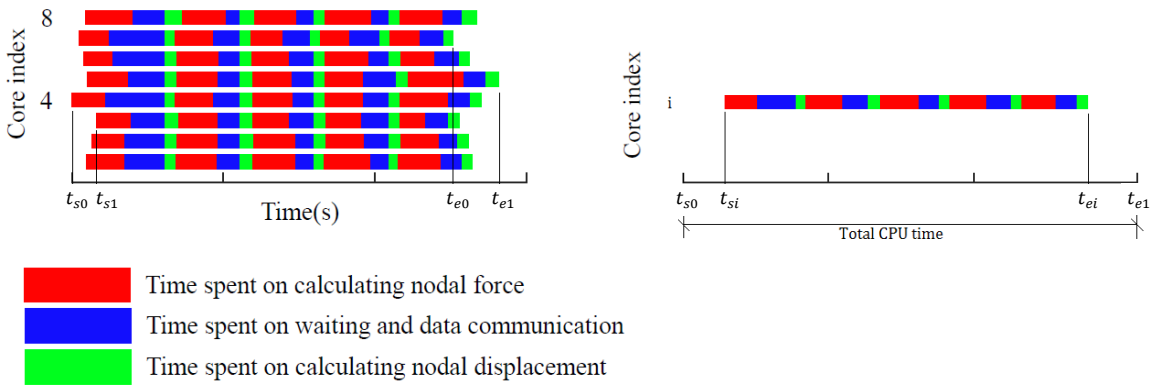


Figure 4.8: A sample of time distribution of parallel processing

The generation of mesh data and preparation of the partitioned mesh data (which also involves I/O operations) are considered as an initialisation stage. This stage is not included in the CPU time measurement since it is independent of the number of time steps involved in the analysis. The speed-up and efficiency calculations are only based on the CPU time measurements recorded during the calculation of nodal force and displacement vectors and the synchronisation among the workers (Steps 4 to 22 of Algorithm 4.1).

CPU time measurements

A sample time stream of a parallel computation using 8 workers (or 8 MPI processes working on 8 cores) is shown in Fig. 4.8. The red bar represents the time spent by each worker to calculate the nodal force vector locally. The mesh partitioning may not be ideal; thus, the workload distributed to each worker can be slightly different, leading to load imbalance. Therefore, it is observed that the time spent on the nodal force calculation is slightly different for each worker. The blue bar indicates the time spent by each worker waiting for the neighbouring workers for the exchange of the nodal forces. The green bar represents the time required to calculate the displacement vector

at each time step. For the CPU time measurements, we define the following terms.

t_{s0} : Earliest start time (when the first worker starts computing) = Actual start time of the analysis

t_{s1} : Latest start time (when all the workers have started working)

t_{e0} : Earliest end time (when the first worker finished computing)

t_{e1} : Latest end time (when all the workers have finished working) = Actual end time of the analysis.

The initialisation time, which involves loading partitioned mesh data by individual workers, is usually different for different workers due to I/O operations. The start time of the analysis is considered when the first worker starts the computation process after the initialisation. The finish time of the analysis is recorded when the slowest worker completes the computation process. This ensures that the total analysis time is the same for all the workers. To reduce the time gap among the workers at the starting point of the analysis (after initialisation), the MPI command Comm.barrier() is utilised, as given in Step 3 of Algorithm 4.1. For the ith worker, we define

t_{si} : Start time of the worker

t_{ei} : End time of the worker

$T_{C,i}$: Sum of calculation time over all the time steps = sum of nodal force (red) and displacement (green) calculation time

$T_{W,i}$: Sum of Communication + Waiting (blue) time over all the time steps

$T_{wt,si} = t_{si} – t_{s0}$: Waiting time at the start of the analysis

$T_{wt,ei} = t_{e1} – t_{ei}$: Waiting time at the end of the analysis

$T_{wt,i} = T_{wt,si} + T_{wt,ei}$

As the total analysis is same for all the workers, it is reasonable to calculate the average calculation time and average communication+waiting time to indicate the per-

formance of the explicit solver. Thus, we have

$$T_C^m = \frac{1}{N}\sum_{i=1}^{N} T_{C,i}, \tag{4.17}$$

$$T_W^m = \frac{1}{N}\sum_{i=1}^{N} (T_{W,i} + T_{wt,i}), \tag{4.18}$$

where, T_C^m is the mean calculation time and T_W^m is the mean communication+wait time. For each worker, the total CPU (or analysis) time remains the same. Therefore,

$$T_T = t_{e1}–t_{s0} = T_{C,i} + T_{W,i} + T_{wt,i} = T_C^m + T_W^m. \tag{4.19}$$

Finally, we define % CPU time components as

$$\%T_C = \frac{T_C^m}{T_T} \times 100, \tag{4.20}$$

$$\%T_W = \frac{T_W^m}{T_T} \times 100. \tag{4.21}$$

Scaling calculation

There are two scaling strategies that are commonly used to define the speed-ups and efficiencies in a distributed computing environment – *Strong* and *Weak scaling*, explained as follows.

1. *Strong scaling* (or *fixed model size scaling*) is based on the concept of an execution workload that does not change with respect to the improvement of the computational resources (that is, with an increase in the number of cores and memory size) [355]. The given problem is distributed among the workers as smaller chunks and the speed-up (S_s) is given by the ratio of the CPU-time of single-core (T_1) and multiple cores (T_{n_c}), that is

$$S_s(n_c) = \frac{T_1}{T_{n_c}}, \tag{4.22}$$

where, n_c is the number of cores (also denoted as N_{cores}). The efficiency calculated from the strong scaling (E_s) is given as

$$E_s(n_c) = \frac{S_s(n_c)}{n_c} = \frac{T_1}{n_c T_{n_c}}. \tag{4.23}$$

2. *Weak scaling* (or *fixed partition size scaling*) proposes that the size of problems be scaled to fully exploit the computing power that becomes available as the resources improve. Therefore, if better equipment is available (with larger number of cores and memory size), then larger problems can be solved within the same time [356, 357]. Gustafson defined the speed-up as the ratio of CPU-speed of multiple-core (v_{n_c}) and single-core (v_1), where the CPU-speed (v_{n_c}) is given as total volume of computations (N_{n_c}) performed per unit time by n_c cores. If the volume of computation is kept proportional to the number of cores, then $N_{n_c} = n_c N_1$. Thus, the speed-up due to weak scaling (S_w) is given as

$$S_w(n_c) = \frac{v_{n_c}}{v_1} = \frac{N_{n_c} T_1}{N_1 T_{n_c}} = \frac{n_c T_1}{T_{n_c}}. \tag{4.24}$$

The efficiency calculated from the weak scaling (E_w) is given as

$$E_w(n_c) = \frac{S_w(n_c)}{n_c} = \frac{T_1}{T_{n_c}}. \tag{4.25}$$

Here, we observe that if the volume of computation kept constant with respect to number of cores (that is, $N_{n_c} = N_1$), then the expressions for speed-up and efficiency reduce to strong scaling as obtained in Eqs. (4.22) and (4.23).

In the present work, the speed-ups and the efficiencies are calculated using the total CPU time T_T of the analysis. In the case where the CPU time of single-core T_1 is not available, the speed-up and efficiencies can be calculated relative to the lowest number of core $n_1 (< n_c)$ for which the CPU time is available. Therefore, for the strong scaling,

$$S_s^{rel}(n_c) = \frac{n_1 T_{n_1}}{T_{n_c}}, \tag{4.26}$$

$$E_s^{rel}(n_c) = \frac{S_s^{rel}(n_c)}{n_c} = \frac{n_1 T_{n_1}}{n_c T_{n_c}}. \tag{4.27}$$

For the weak scaling, the relative expressions are:

$$S_w^{rel}(n_c) = \frac{n_c T_{n_1}}{T_{n_c}}, \tag{4.28}$$

$$E_w^{rel}(n_c) = \frac{S_w^{rel}(n_c)}{n_c} = \frac{T_{n_1}}{T_{n_c}}. \tag{4.29}$$

4.5 Numerical examples

A few numerical examples are presented in this section to verify and demonstrate the efficacy of the proposed octree-based parallel explicit solver.

First, the time-dependent excitations used in the numerical examples are defined. Then, the implementation of the octree-based explicit solver is verified using two benchmark examples for modal and transient analyses.

Finally, the applicability of the solver is illustrated for real-world applications using three numerical examples. These include: (1) The dynamic response of a multi-storey building under blast-loading. (2) The modelling of wave propagation for a moderately long duration in a mountainous region. (3) The simulation of guided waves in a sandwich panel – useful in structural health monitoring. These examples demonstrate a robust, efficient, and massively parallel implementation of the solver for simulating very large-sized problems in a distributed computing environment.

4.5.1 Time-dependent excitations

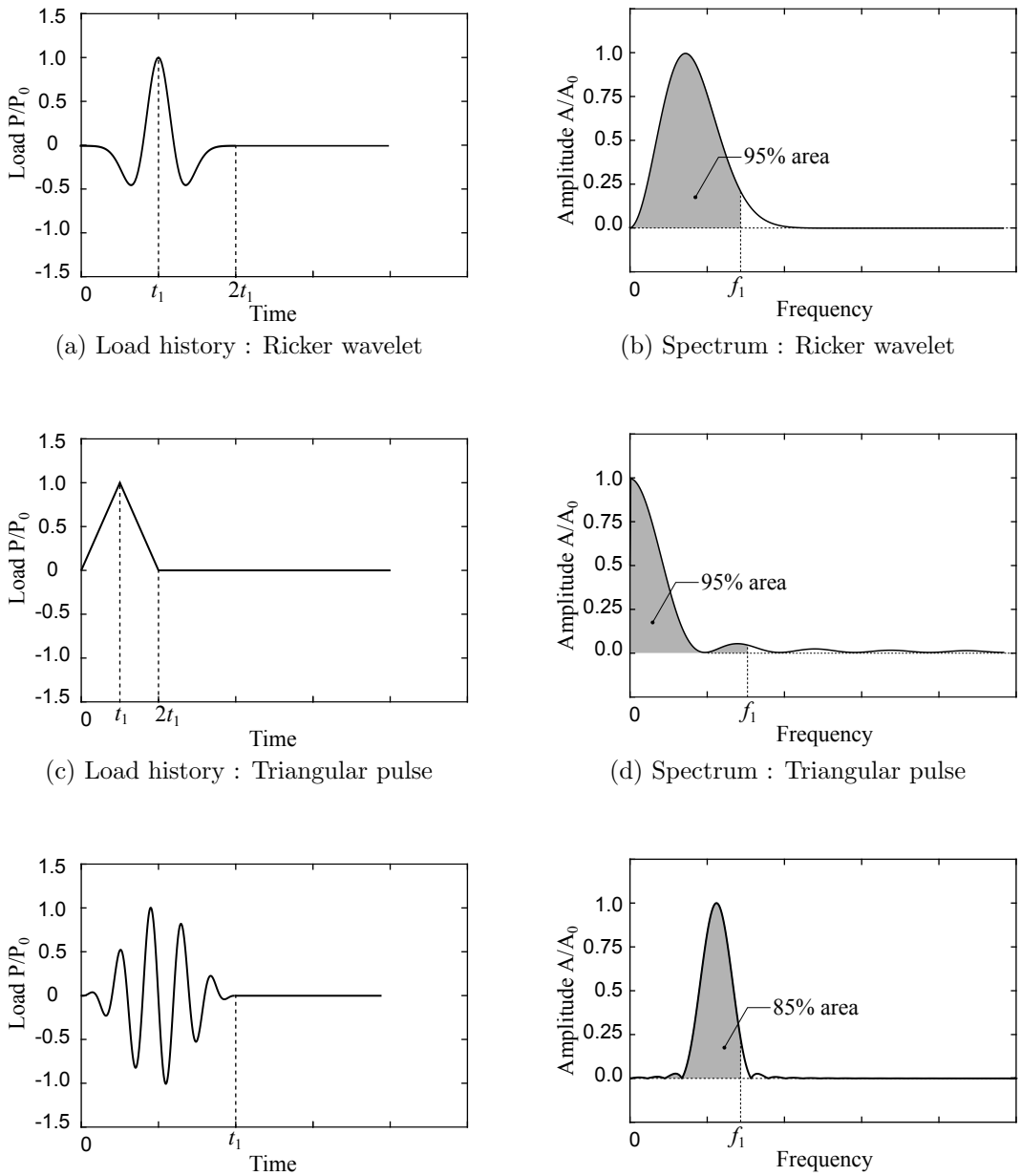
We consider three time-dependent excitations that make use of different transient amplitude functions, as shown in Fig. 4.9. These include: a Ricker wavelet (Fig. 4.9a), a triangular pulse (Fig. 4.9c), and a sine-burst signal (Fig. 4.9e). The analytical form of the Ricker wavelet in the time domain is

$$P(t) = \left(1 - \left(\frac{t - t_1}{t_1/5}\right)^2\right) \exp\left(-\frac{1}{2}\left(\frac{t - t_1}{t_1/5}\right)^2\right) P_0, \tag{4.30}$$

where, P_0 is the maximum load. The triangular impact load is expressed as

$$P(t) = \begin{cases} \left(1 - \left|\frac{t}{t_1} - 1\right|\right) P_0 & \text{if } 0 \leq t \leq 2t_1 \\ 0 & \text{if } t > 2t_1 \end{cases}, \tag{4.31}$$

while the sine-burst (sine excitation modulated by a Hann window) is expressed as



(a) Load history : Ricker wavelet

(b) Spectrum : Ricker wavelet

(c) Load history : Triangular pulse

(d) Spectrum : Triangular pulse

(e) Load history : Sine-burst signal

(f) Spectrum : Sine-burst signal

Figure 4.9: Time-dependent excitations

$$P(t) = \begin{cases} \sin\left(\frac{2\pi n t}{t_1}\right) \sin^2\left(\frac{\pi t}{t_1}\right) P_0 & \text{if } 0 \leq t \leq t_1 \\ 0 & \text{if } t > t_1 \end{cases}, \tag{4.32}$$

where, n is the number of cycles. The central frequency f_m of the sine-burst is calculated as $f_\mathrm{m} = n/t_1$.

The frequency spectrum of the given excitation functions can be obtained using the Fourier transform. For e.g., the frequency spectrum of the Ricker wavelet (depicted in Fig. 4.9b) is expressed as

$$A(f) = \left(\frac{2}{\sqrt{\pi}}\right)\left(\frac{f}{f_m}\right)^2 \exp\left(-\left(\frac{f}{f_m}\right)^2\right), \tag{4.33}$$

where, the central frequency f_m is given as $f_\mathrm{m} = 5/(\sqrt{2}\pi t_1)$.

We define an upper limit in the frequency spectrum f_1 (also called the highest frequency of interest or the critical frequency) such that the area under the frequency spectrum curve in the region $0 \leq f \leq f_1$ is equal to 95% of its total area. This region of the frequency spectrum accounts for most of the energy content of the signal. For a Ricker wavelet, f_1 is approximately $2.2/t_1$, while in the case of triangular impact f_1 is $1.8/t_1$. For the sine-burst excitation, f_1 is identified corresponding to 85% of the area, which is $5.8/t_1$ when $n = 5$. The resolution of the spatial discretisation of the problem domain can be estimated from the critical frequency f_1 and material properties as explained below.

In an infinite elastic media, the propagating waves are categorized into two types: dilatational/primary and shear/secondary waves. The dilatational and shear wave velocities (v_P and v_S) for an isotropic, homogeneous, linear-elastic medium can be given as

$$v_\mathrm{P} = \sqrt{\frac{E(1-\nu)}{\rho(1+\nu)(1-2\nu)}}, \tag{4.34a}$$

$$v_\mathrm{S} = \sqrt{\frac{E}{2\rho(1+\nu)}}, \tag{4.34b}$$

where, E is the Young's modulus, ν is the Poisson's ratio, and ρ is the mass density of

the material. The minimum value of dilatational and shear wavelengths (L_{P} and L_{S}), associated with the highest frequency of interest in the analysis, can be given by

$$\lambda_{\mathrm{P}} = \frac{v_{\mathrm{P}}}{f_1}, \tag{4.35a}$$

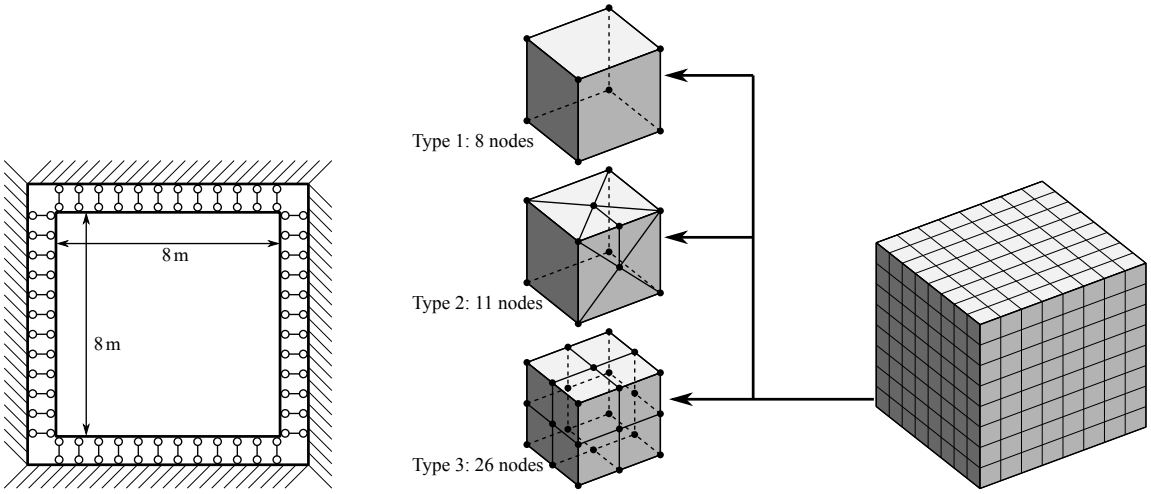
$$\lambda_{\mathrm{S}} = \frac{v_{\mathrm{S}}}{f_1}. \tag{4.35b}$$

In structural wave propagation problems, it is often suggested to limit the finite element size with at-least 10 (linear) finite elements per wavelength (λ_{P} or λ_{S}). However, this may result in deviations of more than 1% compared to a reference solution. Therefore, some authors [358, 359] suggest roughly 30 nodes per wavelength when using linear shape functions. In addition to spatial sampling, appropriate temporal sampling is also necessary to accurately capture the input excitation wave in discrete form. A sufficient condition for the sampling rate of a signal can be given by the Shannon-Nyquist theorem, which states that at least two sampling points are required for one time-period of the signal associated with the highest frequency of interest. In general, it is recommended to use at-least 10 sampling points per time-period to appropriately capture the input excitation wave. Nevertheless, the stability requirements for time step size are often much more critical in explicit time-integration schemes, which automatically satisfy the minimum temporal-sampling requirements.

4.5.2 Modal analysis

We first perform modal analysis of a cube to verify the mass lumping formulation of octrees. A detailed investigation regarding the use of mass lumping in the framework of 2D SBFEM is covered by Gravenkamp et al. [346].

We consider a cube of length $L = 8\,\mathrm{m}$ and having a linear elastic material with Young's modulus $E = 1000\,\mathrm{Pa}$, Poisson's ratio of $\nu = 0.3$ and mass density $\rho = 1\,{}^{\mathrm{kg}}/_{\mathrm{m}^3}$. The surface of the cube is constrained against the motion in the perpendicular direction to the surface. The geometry and boundary conditions in a 2D view are illustrated in



(a) Geometry and boundary conditions

(b) Three types of meshes

Figure 4.10: Geometry and mesh of the cube

Fig. 4.10a. Three types of meshes based on different octree patterns are considered, as shown in Fig. 4.10b. The eigenfrequencies based on these meshes are calculated for both kinds of mass matrices: consistent mass matrix (CMM) and lumped mass matrix (LMM). For comparison, the FEM results obtained using commercial software ABAQUS with type 1 (Hexahedral) mesh are also calculated. A convergence study is done where the L_2 error norm (e) of the first 100 eigenfrequencies for all mesh types is compared. The error e is calculated as

$$e = \sqrt{\frac{\sum_{i=1}^{100} \left(\omega_i^{\text{num}} - \omega_i^{\text{ref}}\right)^2}{\sum_{i=1}^{100} \left(\omega_i^{\text{ref}}\right)^2}}, \tag{4.36}$$

where, ω_i^{num} is the ith eigenfrequency of the cube obtained using numerical simulation. ω_i^{ref} is the analytical solution given by

$$\omega^{\text{ref}} = \sqrt{\frac{E\pi^2}{\rho\,(1-\nu^2)\,L^2}(l^2+m^2+n^2)\,\gamma}, \tag{4.37}$$

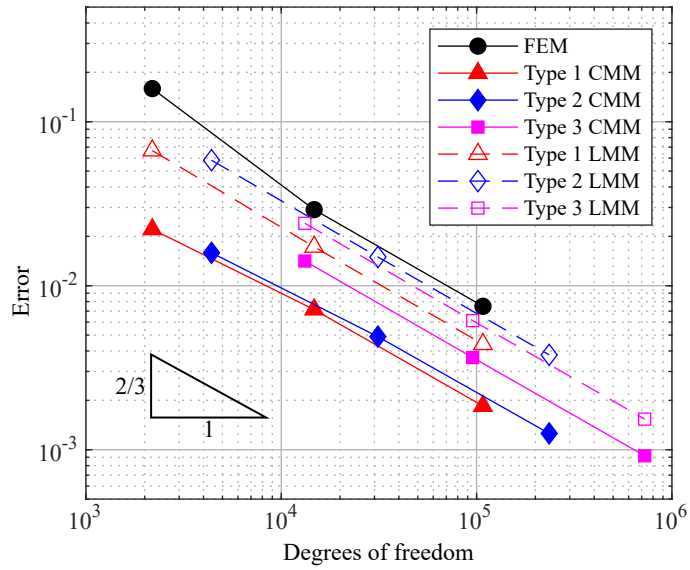


Figure 4.11: Relative error of first 100 eigenfrequencies

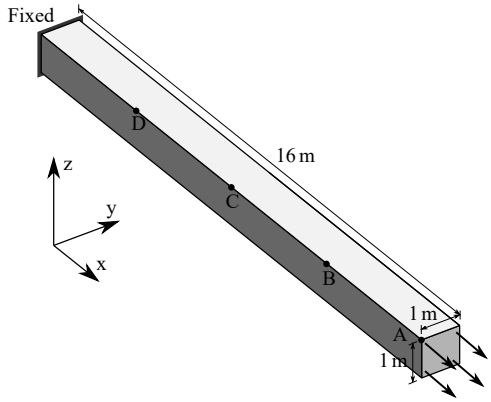
where l, m and n are non-negative integers, and the parameter γ is defined as

$$\gamma = \begin{cases} 1 & \text{if two of } \{l, m, n\} \text{ are } 0 \\ \left\{1, \frac{1-\nu}{2}\right\} & \text{if one of } \{l, m, n\} \text{ is } 0 \\ \left\{1, \frac{1-\nu}{2}, \frac{1-\nu}{2}\right\} & \text{if none of } \{l, m, n\} \text{ is } 0 \end{cases}, \tag{4.38}$$

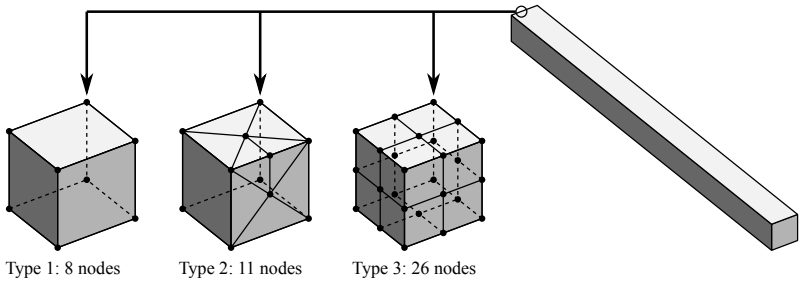
The results are shown in Fig. 4.11, where it is observed that the CMM based approach yields slightly more accurate results than the LMM for the same number of DOFs. Nevertheless, an optimal convergence rate (2/3) is achieved for both the LMM and CMM based approaches. Further, the influence of different octree pattern types on the slope of the curves is negligible.

4.5.3 Transient analysis

In this example, we perform a transient analysis in a cantilever beam to verify the solver implementation. The dimensions of the beam are $16\,\mathrm{m} \times 1\,\mathrm{m} \times 1\,\mathrm{m}$. The material is assumed linear-elastic with Young's modulus $E = 10\,\mathrm{kPa}$, Poisson's ratio $\nu = 0$, and mass density $\rho = 1\,{}^{\mathrm{kg}}/_{\mathrm{m}^3}$. As depicted in Fig. 4.12a, the beam is fixed on one end, and



(a) Geometry and boundary conditions



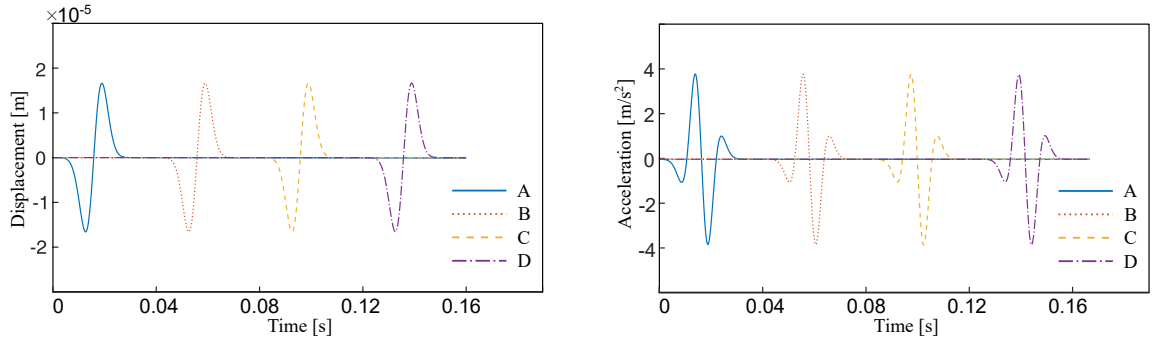
(b) Three types of meshes

Figure 4.12: Geometry and mesh of the beam

a uniform pressure load is applied on the other end. The applied pressure load is a Ricker wavelet shown in Fig. 4.9a with the parameters $t_1 = 15\,\text{ms}$ and $P_0 = 1\,\text{Pa}$. From Section 4.5.1, the maximum frequency of interest f_1 and the associated dilatational wavelength is calculated as 150 Hz (approx.) and 0.67 m respectively.

Similar to the modal analysis in the last section, we consider three types of meshes that are generated using different octree patterns, as shown in Fig. 4.12b. In the coarsest group of meshes, the element size is set to 0.0625 m; thus, there are at-least 10 elements per wavelength of the applied pressure load. Further, at-least 10 sampling points per time-period is used to appropriately capture the input excitation wave.

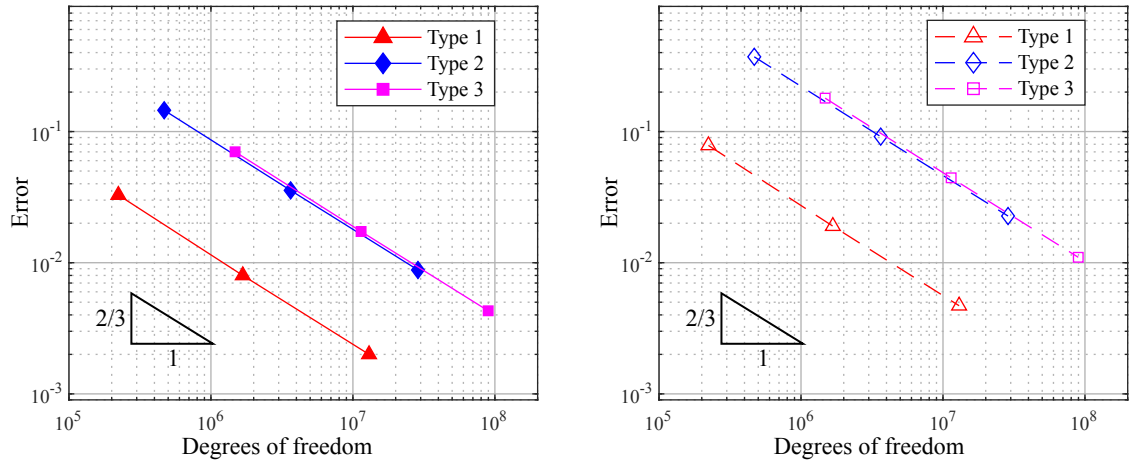
Fig. 4.13 shows the displacement and acceleration history of the four selected points (depicted in Fig. 4.12a). The reference solution for the displacement response can be



(a) Displacement history of four selected points

(b) Acceleration history of four selected points

Figure 4.13: Time histories of four selected points on the beam



(a) Convergence curve of displacement

(b) Convergence curve of acceleration

Figure 4.14: Convergence curves of different types of meshes

found in [346]. The L_2 error norm of the selected points for all time steps are calculated, and the convergence curves are presented in Fig. 4.14 for the given mesh-types. It is observed that the optimal convergence rates are obtained for both the displacement and acceleration responses using the present approach, which confirms the correctness of the solver implementation.

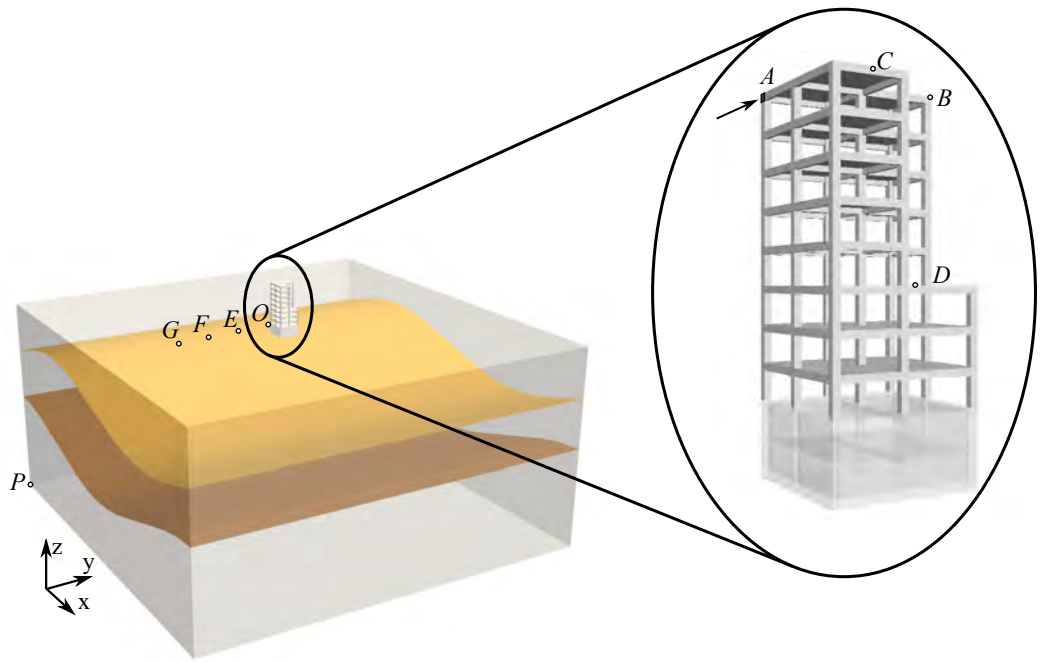


Figure 4.15: Geometry of the building and ground

4.5.4 Multi-storey building and ground

To further verify the proposed technique and evaluate its computational performance, we analyse the response of a multi-storey building and ground under blast-loading. The building is modelled using columns, beams, slabs, and a foundation with a basement, as shown in Fig. 4.15. The origin of the coordinate system, denoted as O, is kept near the base of the building. The dimensions are illustrated in Fig. 4.16. The material of the building is assumed as concrete with linear-elastic characteristics and the following properties: Young's modulus $E = 17\,\text{GPa}$, Poisson's ratio $\nu = 0.2$, and mass density $\rho = 2400\,{}^{\text{kg}}/_{\text{m}^3}$. The dilatational and the shear wave speeds for this material are calculated as $2805\,{}^{\text{m}}/_{\text{s}}$ and $1718\,{}^{\text{m}}/_{\text{s}}$, respectively.

The dimension of the surrounding ground under consideration (Ref: Fig. 4.15) is $400\,\text{m} \times 400\,\text{m} \times 200\,\text{m}$. The ground is assumed to be made of three layers, where the

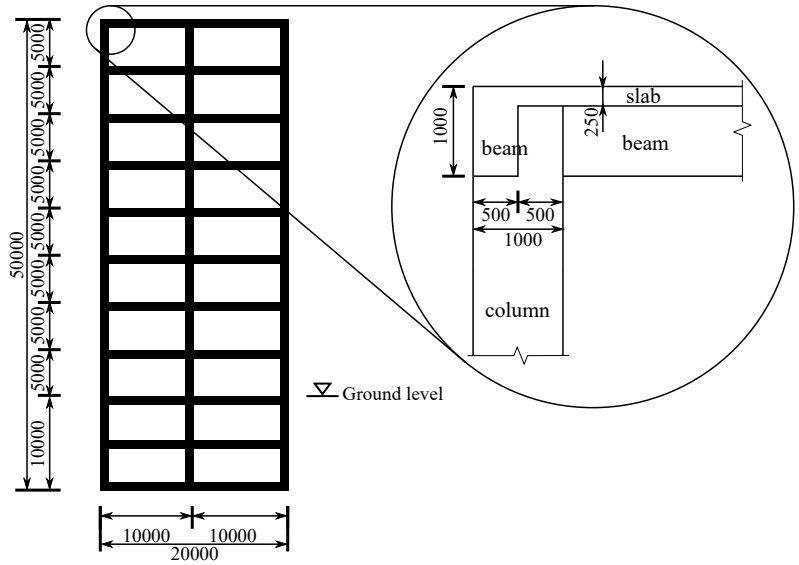


Figure 4.16: Dimension of the building (Unit: mm)

shape of two interfaces between the individual layers is defined as (unit: m)

$$z_{12} = -60 + 20 \times \sin\left(\frac{x + 183}{200}\pi\right), \tag{4.39a}$$

$$z_{23} = -130 + 20 \times \cos\left(\frac{x + 183}{200}\pi\right). \tag{4.39b}$$

The ground material is also assumed to have the linear-elastic behaviour with increasing Young's moduli from the top to the bottom layers as 0.5 GPa, 0.8 GPa, and 1 GPa. The Poisson's ratio ν and mass density ρ are assumed to be constant for all the layers with $\nu = 0.2$ and $\rho = 2400\,{}^{\mathrm{kg}}/_{\mathrm{m}^3}$. The corresponding dilatational wave velocities in the three layers are calculated as $481\,{}^{\mathrm{m}}/_{\mathrm{s}}$, $609\,{}^{\mathrm{m}}/_{\mathrm{s}}$, and $680\,{}^{\mathrm{m}}/_{\mathrm{s}}$, while the shear wave velocities are calculated as $295\,{}^{\mathrm{m}}/_{\mathrm{s}}$, $373\,{}^{\mathrm{m}}/_{\mathrm{s}}$, and $417\,{}^{\mathrm{m}}/_{\mathrm{s}}$.

A blast load with the positive phase idealized as a triangular-shaped pressure profile is shown in Fig. 4.9c (the negative phase ignored for simplicity). It is applied over an area of $1\,\mathrm{m}^2$ near the top side-face of the building ($2.5\,\mathrm{m} \leq x \leq 3.5\,\mathrm{m}$, $y = 2.5\,\mathrm{m}$ and $39\,\mathrm{m} \leq z \leq 40\,\mathrm{m}$), indicated by region A in Fig. 4.15 with a peak pressure $P_0 = 10^7\,\mathrm{Pa}$. The half duration of the blast load is kept as $t_1 = 250\,\mathrm{ms}$, from which the maximum frequency of interest is calculated as 8 Hz. The total duration of the analysis is 1 s, and

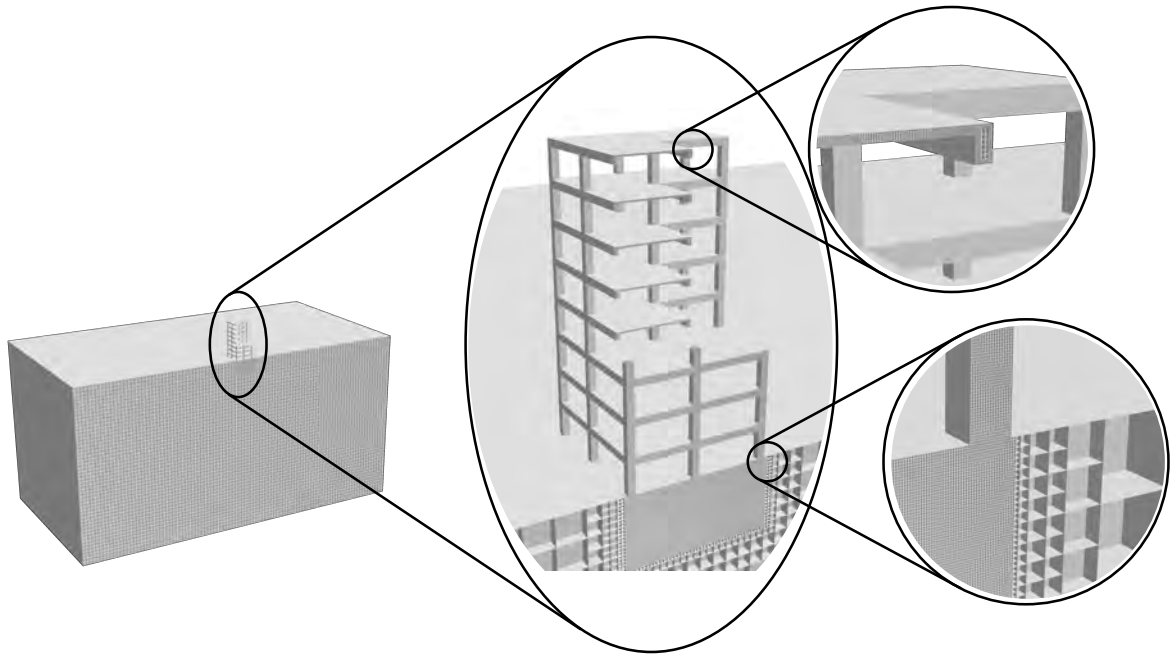
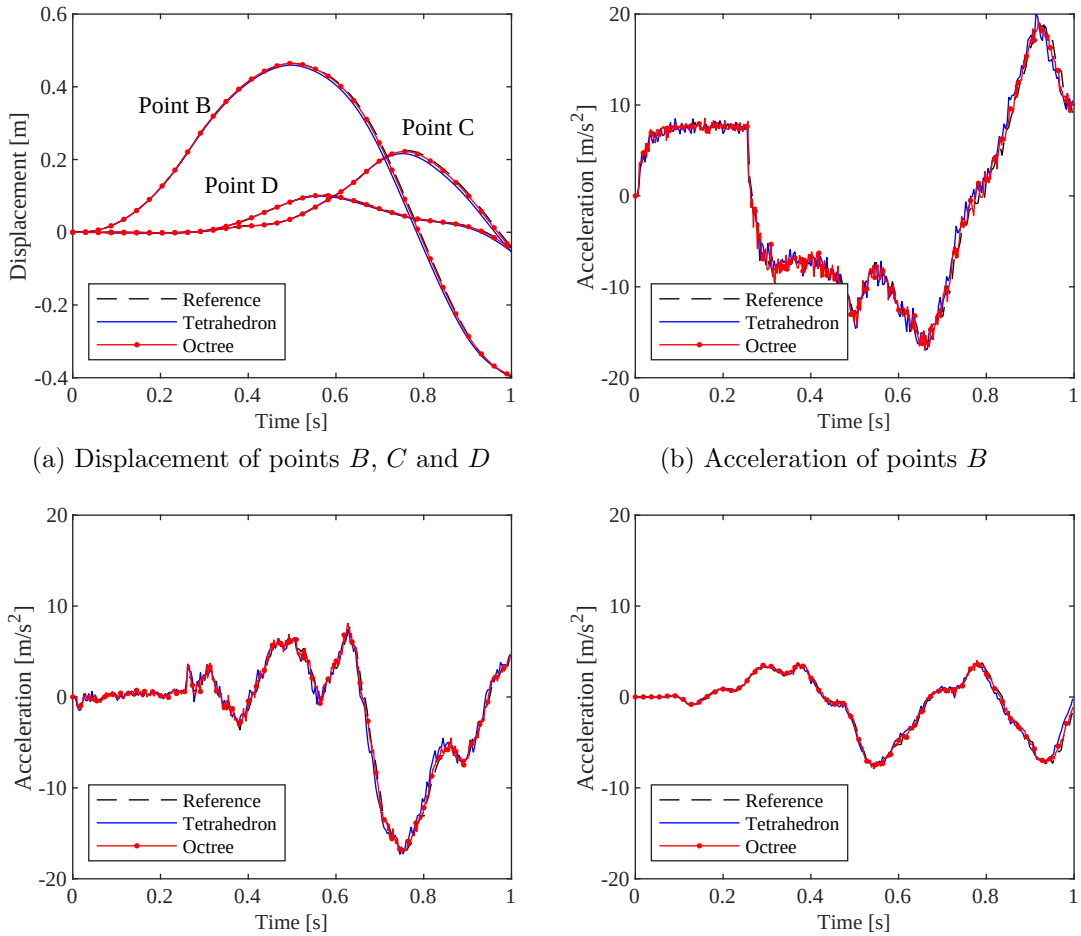


Figure 4.17: Octree mesh of the building and ground

the time step size is $\Delta t = 0.018\,\text{ms}$ estimated using Eq. (4.14). Thus, the total number of time steps is 55,556. The damping coefficient α in Eq. (4.7) is kept as 0.1.

From a convergence study, we obtained a reference solution of the problem by performing analysis on the commercial finite element package ABAQUS using hexahedral elements (C3D8). Only the building including its foundation with basement was modelled, and the displacements perpendicular to the side faces below the ground level and the bottom of the basement were constrained. From the results of the convergence study, it was inferred that an acceptable accuracy is obtained using the hexahedral mesh with an element size of 0.0625 m.

Generating a non-uniform hexahedral mesh for performing an efficient wave propagation simulation in a complete model including both the building and the ground requires considerable human interventions. Therefore, we use the non-structured tetrahedral meshes in ABAQUS to compare with the present technique that makes use of fully automatically generated octree meshes (shown in Fig. 4.17). Using the reference solution, the tetrahedral and octree mesh sizes were identified that yielded a similar level



(a) Displacement of points B, C and D

(b) Acceleration of points B

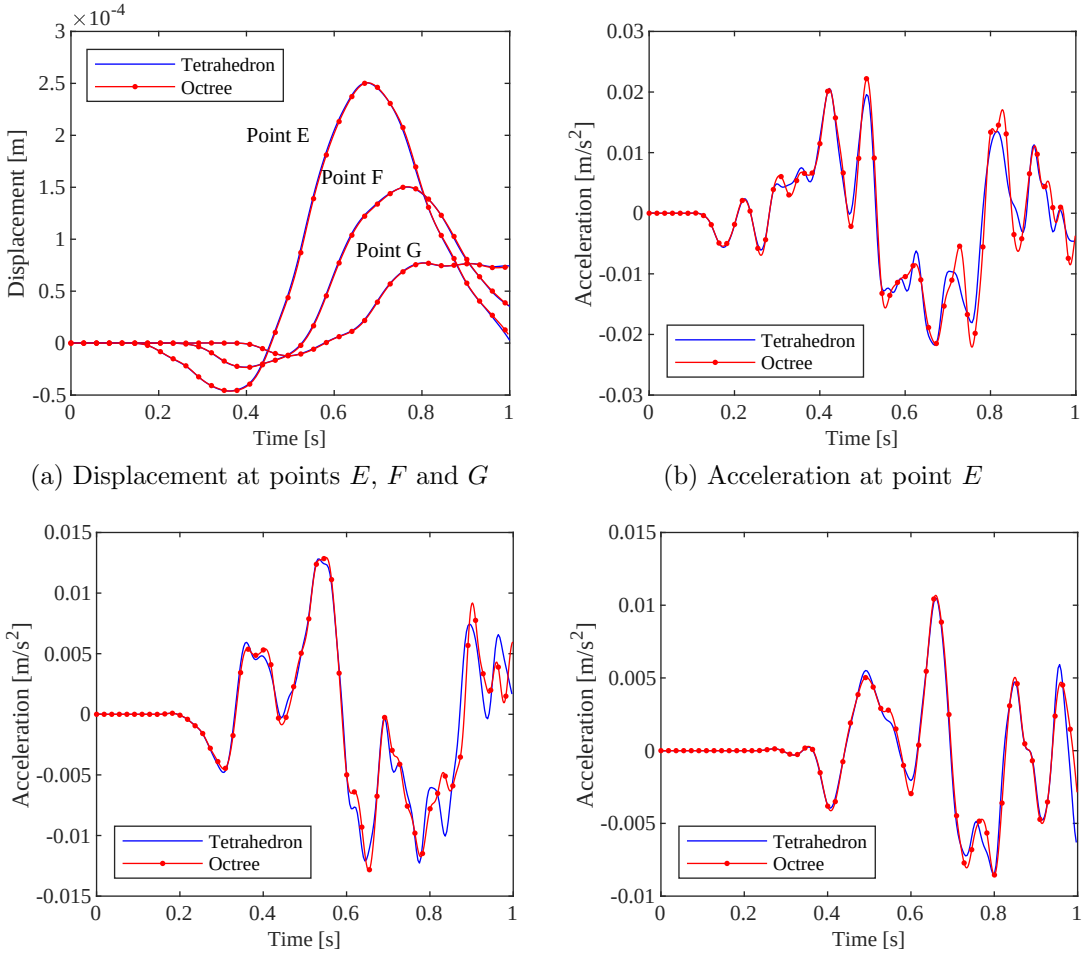
(c) Acceleration of points C

(d) Acceleration of points D

Figure 4.18: The displacement and acceleration history obtained at the selected points in the building using the octree mesh (SBFEM) and ABAQUS (FEM)

of accuracy. For the building model, the identified minimum and maximum element sizes in the octree mesh are 0.0625 m and 0.25 m, respectively, while for the tetrahedral mesh, the corresponding sizes are 0.05 m and 0.25 m, respectively. For the ground model, both the tetrahedral and octree meshes are generated with a maximum element size of 4 m. Overall, there are 7,339,939 elements and 30,482,412 DOFs in the octree mesh (Fig. 4.17), while there are 38,847,472 elements and 26,192,571 DOFs in the tetrahedral mesh of the whole model (building and ground).

To compare the results, a few reference points are chosen in the model as indicated in Fig. 4.15. In the building, the reference points are: $B(2.5\,\text{m}, 22.5\,\text{m}, 40\,\text{m})$



(a) Displacement at points E, F and G

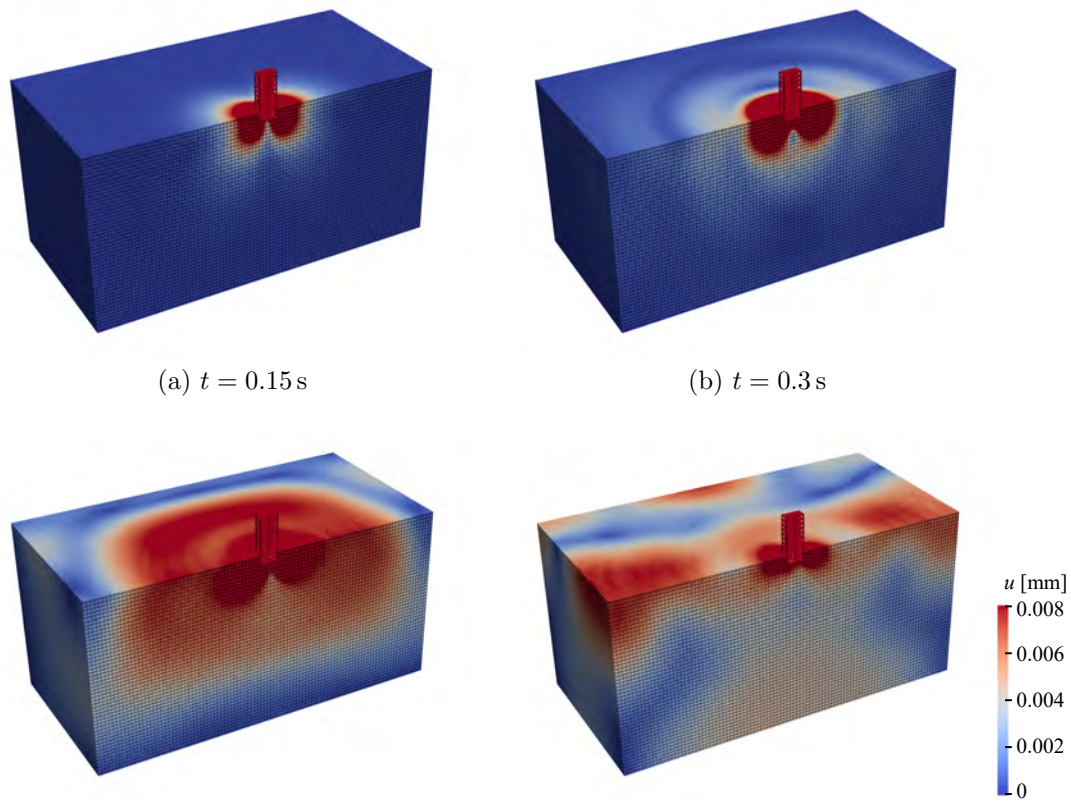
(b) Acceleration at point E

(c) Acceleration at point F

(d) Acceleration at point G

Figure 4.19: Displacement and acceleration history obtained at the selected points on the ground surface obtained using the octree mesh (SBFEM) and ABAQUS (FEM)

at the corner of the building, $C(22.5\,\mathrm{m}, 7.5\,\mathrm{m}, 40\,\mathrm{m})$ at the middle of a beam, and $D(17\,\mathrm{m}, 17\,\mathrm{m}, 15\,\mathrm{m})$ at the centre of a slab. On the ground, three reference points are chosen: $E(0\,\mathrm{m}, -30\,\mathrm{m}, 0\,\mathrm{m})$, $F(0\,\mathrm{m}, -60\,\mathrm{m}, 0\,\mathrm{m})$, and $G(0\,\mathrm{m}, -90\,\mathrm{m}, 0\,\mathrm{m})$ which are located at a certain distance from the building for comparison purposes. The displacement and acceleration response histories of the building and the ground obtained using both the meshes are shown in Fig. 4.18 and Fig. 4.19 respectively. It is observed that the results obtained with our in-house software utilising the octree mesh show a very good agreement with the reference solution as well as those obtained using tetrahedral mesh



(a) $t = 0.15\,\mathrm{s}$ (b) $t = 0.3\,\mathrm{s}$

(c) $t = 0.45\,\mathrm{s}$ (d) $t = 0.6\,\mathrm{s}$

Figure 4.20: Contour of displacement magnitude in the building and ground model at various time instances

in ABAQUS. The process of wave propagation in the ground is illustrated in Fig. 4.20 at four selected time instances.

Since the CPU time per time step remains fairly constant during the analysis, the CPU time required to complete the first 1000 steps is selected for measuring the computational performance. This allows the evaluation of the speedup of parallel computing, starting from a relatively smaller number of cores. The computational time taken by ABAQUS and the octree-based in-house code is listed in Table 4.1. To limit the wall-clock time required by the ABAQUS analyses, we opt for using four or more computer cores. It is observed that the computational performance of our in-house code (using octree-mesh) is significantly faster than the commercial software ABAQUS (using

Table 4.1: Computational time of the in-house code and ABAQUS for the building and ground model

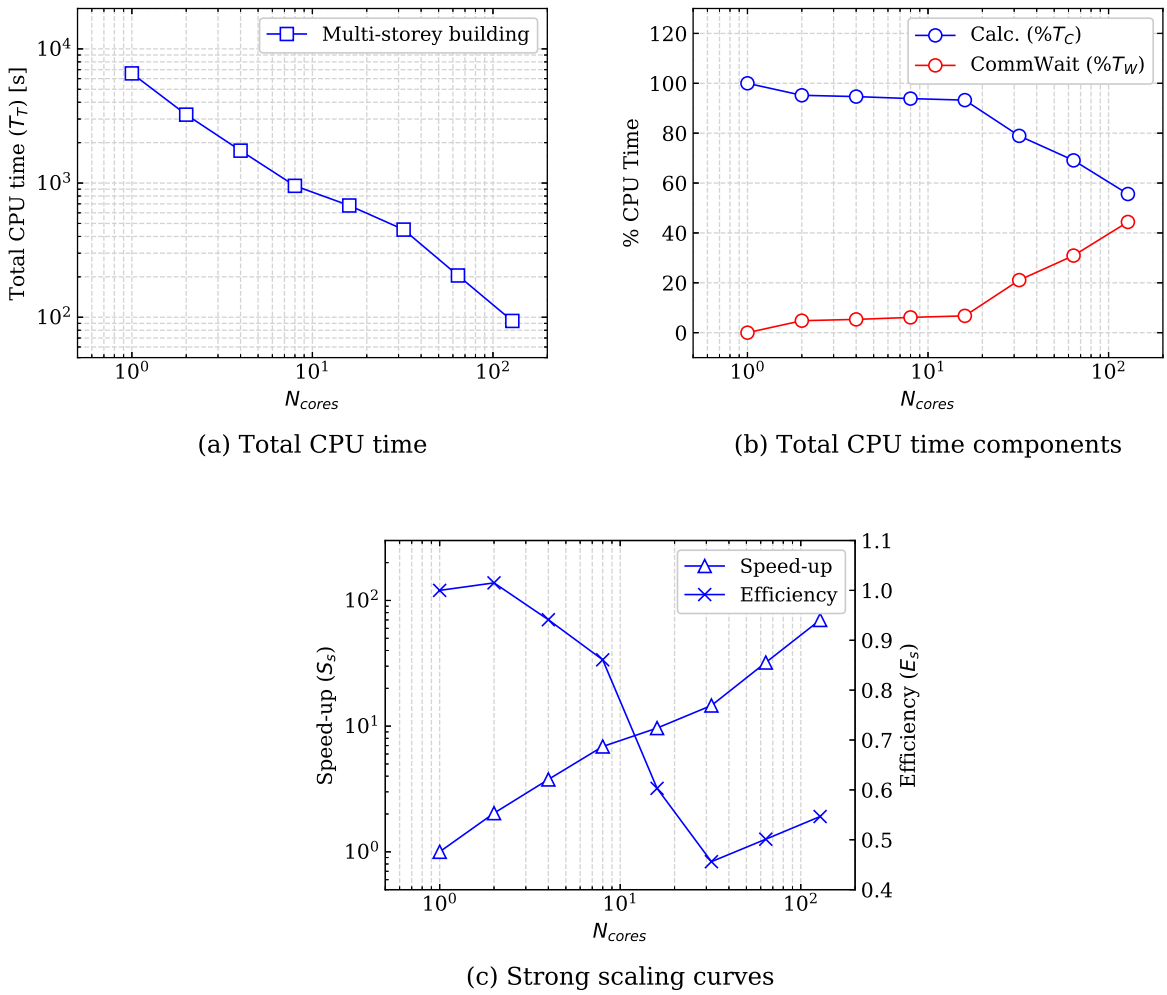
N_{cores}	ABAQUS	In-house code						$\frac{T_{T(\text{ABAQUS})}}{T_{T(\text{In-house})}}$
	T_T	T_C^m	T_W^m	T_T	S_s	E_s	M_T	
1	-	6,565.47	0.01	6,566.12	1.00	1.00	11.04	-
2	-	3,078.62	155.46	3,234.37	2.03	1.02	11.29	-
4	4,550	1,651.08	92.94	1,744.20	3.76	0.94	11.32	2.61
8	2,264	894.91	58.47	953.49	6.89	0.86	12.09	2.37
16	1,407	634.49	45.90	680.48	9.65	0.60	12.68	2.07
32	1,154	355.03	94.84	449.92	14.59	0.46	13.61	2.56
64	534	141.43	63.28	204.73	32.07	0.50	17.38	2.61
128	334	52.18	41.66	93.85	69.97	0.55	26.58	3.56

T_C^m: Mean calculation time [s], T_T: Total CPU time [s], M_T: Total memory [GB]
T_W^m: Mean communication+waiting time [s] S_s, E_s: Speedup and efficiency (Strong scaling)

tetrahedral mesh) when the results are obtained with similar accuracy. The proposed explicit solver outperforms ABAQUS by a factor of at least two. Further, the memory requirement for the analysis using the in-house code is fairly small due to the use of unique-octree patterns. The exact memory requirement for explicit analysis considering only critical variables in Abaqus could not be extracted and hence, not shown.

The CPU time obtained using the in-house code is further analysed using Fig. 4.21. From Fig. 4.21b, it is observed that the Communication+Wait component of the CPU time ($\%T_W$) increases steadily till 16 cores and then, increases sharply for the higher number of cores. Several factors influence the Communication+Wait time, which includes load-imbalancing of the mesh-partitioning, non-uniform memory bandwidth distribution per core, etc. The study of such factors is beyond the scope of the present work.

The strong scaling study of the proposed method is shown in Fig. 4.21c. It is observed that the efficiency of the solver becomes minimum near 16 and 32 cores. This behaviour can be attributed to limited memory bandwidth available on a single



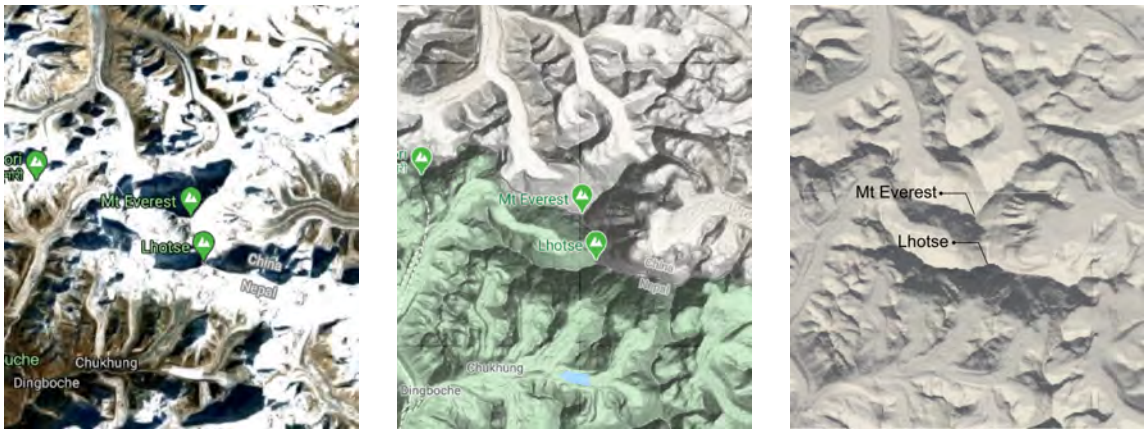
(a) Total CPU time

(b) Total CPU time components

(c) Strong scaling curves

Figure 4.21: Computational time study of the in-house code for the building and ground model

compute-node ($N_{cores} \leq 48$), as explained by Chang et al. [83]. On multiple compute-nodes ($N_{cores} > 48$), the efficiency tends to increase steadily with the increasing number of cores. This is mainly caused by the cache-effect – explained later in Section 4.5.6. It is worth mentioning that for a given model size, the use of a higher number of compute-nodes in a distributed computing environment may lead to under-utilisation of the available computational resources. Therefore, it is often suggested to perform a weak scaling study [356, 357], where the size of the model increases with the usage of more computational resources. Such a study has been carried out in Section 4.5.6 for a



(a) Satellite image (b) Terrain (c) STL model

Figure 4.22: Geometry of the mountainous region

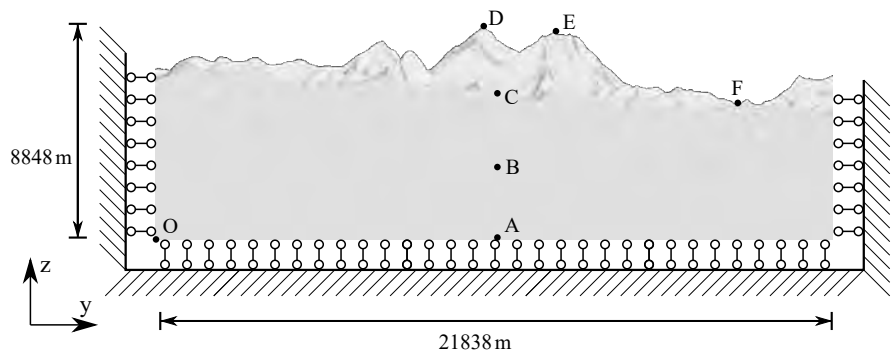
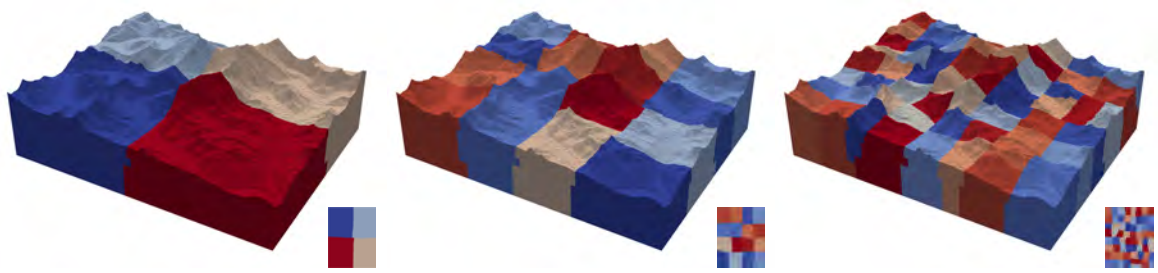


Figure 4.23: Boundary condition and dimensions of the mountainous region

different problem.

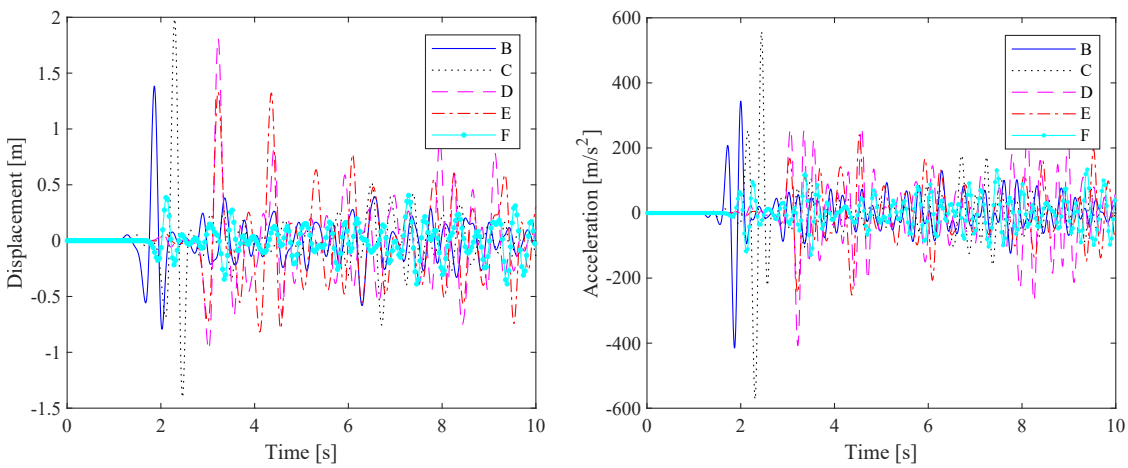
4.5.5 Mountainous region

In this example, we investigate the wave propagation in a mountainous region for a moderately-long duration to demonstrate the practical applicability of the proposed methodology for the seismic-like events. The location of the region is shown in a satellite image from Google Maps in Fig. 4.22a, while the terrain information (Fig. 4.22b) and the STL model (Fig. 4.22c) are generated using the online service Terrain2STL at `http://jthatch.com/Terrain2STL/` [360]. For illustration purposes, the model size of 21.8 km $\times$ 18.0 km area is considered. A homogeneous and linear-elastic material



(a) four parts (b) 16 parts (c) 64 parts

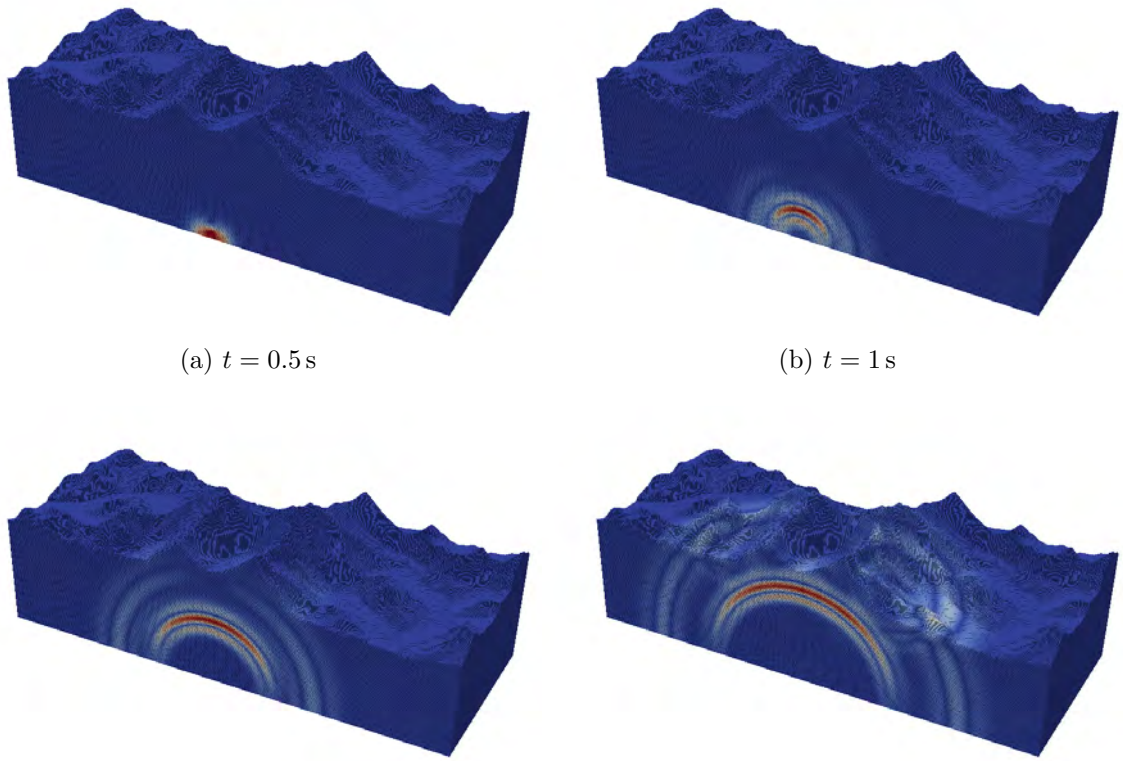
Figure 4.24: Mesh partition of the mountainous region



(a) Displacements (b) Accelerations

Figure 4.25: Displacements and accelerations in y direction at five selected points in the mountainous region model

is assumed with a Young's modulus $E = 55\,\text{GPa}$, Poisson's ratio $\nu = 0.2$, and mass density $\rho = 2400\,\text{kg/m}^3$ (Density of a typical rock is in the range 2200-2800 kg/m^3). For this material, the dilatational and shear wave speeds are calculated as 5046 m/s and 3090 m/s, respectively. The boundary conditions of the model are shown in Fig. 4.23, where the displacements perpendicular to the side and the bottom surfaces are constrained. Excitation load in the form of a Ricker wavelet (Fig. 4.9a) with $t_1 = 0.5\,\text{s}$ and $P_0 = 10^{15}\,\text{N}$ is applied at the centre of the bottom of the model (point A) in the y-direction. The maximum frequency of interest is calculated as 5 Hz. The critical wavelength is calculated from the shear wave speed and given as 618 m.

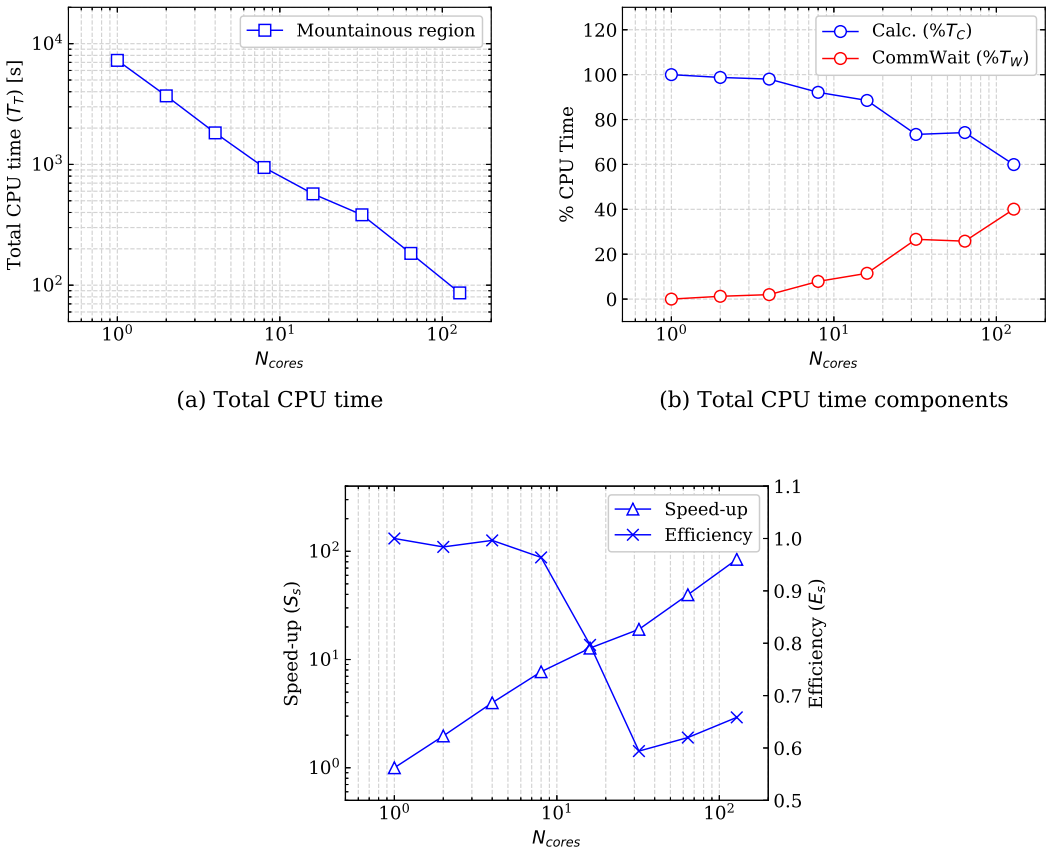


(a) $t = 0.5\,\mathrm{s}$ (b) $t = 1\,\mathrm{s}$

(c) $t = 1.5\,\mathrm{s}$ (d) $t = 2\,\mathrm{s}$

Figure 4.26: Contour of displacement magnitude of the mountain region model at four selected time instances t

Based on the critical wavelength, an octree mesh is generated with 7,608,933 elements and 32,052,075 DOFs having a maximum element size of 68.125 m and a minimum element size of 8.516 m. The mesh is partitioned using *meshpart* developed by [331] and depicted in Fig. 4.24 for various values of mesh-part counts. To illustrate the time-dependent displacement and acceleration, the following reference points are selected as shown in Fig. 4.23: $B(9000\,\mathrm{m}, 10000\,\mathrm{m}, 4000\,\mathrm{m})$, $C(7848\,\mathrm{m}, 11090\,\mathrm{m}, 5460\,\mathrm{m})$, $D(11431\,\mathrm{m}, 11772\,\mathrm{m}, 7848\,\mathrm{m})$, $E(11772\,\mathrm{m}, 9042\,\mathrm{m}, 7336\,\mathrm{m})$ and $F(9895\,\mathrm{m}, 4095\,\mathrm{m}, 4265\,\mathrm{m})$. The origin of the coordinate system is point O. The time step calculated using the eigenanalysis is given as $\Delta t = 1.38\,\mathrm{ms}$. The response is analysed for a duration of 10 s using 7246 time steps.



(a) Total CPU time

(b) Total CPU time components

(c) Strong scaling curves

Figure 4.27: Computational time study of the in-house code for the mountainous region

The displacement and acceleration in y-direction of the reference points are plotted against time in Fig. 4.25. Further, the contour of displacement magnitude at the four time instances are presented in Fig. 4.26, illustrating the wave propagation process.

The computational performance is evaluated for the first 1000 steps for increasing numbers of cores as shown in Fig. 4.27 and Table 4.2. The presented results are similar to those reported for the multi-storey building and ground example in Section 4.5.4. From Fig. 4.27b, it is observed that %Comm+Wait time increases steadily from 0 to $\sim 40\%$ over the range of 128 cores. From the strong scaling curves shown in Fig. 4.27c, it is observed that the efficiency of the solver becomes minimum near 32 cores (on single compute node, $N_{cores} \leq 48$), while the efficiency tends to increase steadily with further

Table 4.2: Computational time of the in-house code for the mountainous region

N_{cores}	T_C^m	T_W^m	T_T	S_s	E_s	M_T
1	7,273.36	0.01	7,273.94	1.00	1.00	11.97
2	3,650.19	46.03	3,696.50	1.97	0.99	12.26
4	1,788.73	36.26	1,825.15	3.99	1.00	12.47
8	868.96	74.28	943.34	7.71	0.96	12.79
16	504.91	65.58	570.55	12.7	0.79	13.18
32	280.71	101.89	382.62	19.0	0.59	14.28
64	136.02	47.31	183.34	39.7	0.62	19.69
128	51.72	34.58	86.30	84.3	0.66	26.49

T_C^m: Mean calculation time [s], T_T: Total CPU time[s], M_T: Total memory [GB]
T_W^m: Mean communication+wait time [s], S_s, E_s: Speedup and efficiency (Strong scaling)

increase in number of cores (on multiple compute nodes, $N_{cores} > 48$). This scaling behaviour is consistent with those obtained in the previous example.

4.5.6 Sandwich panel

In this example, we perform the simulation of guided wave propagation in a sandwich panel, which has applications in structural health monitoring. A detailed discussion on guided waves can be found in Refs. [361–365].

We consider a sandwich panel with two cover sheets and a foam-core. The digital image is obtained using X-ray CT scans [2] and shown in Fig. 4.28. The outer dimension of the panel is 288 mm × 288 mm × 57.6 mm and the thickness of the cover sheet is 4.8 mm. The material of the foam-core is assumed as aluminium with the following properties: Young's modulus $E_{\text{a}} = 70\,\text{GPa}$, Poisson's ratio $\nu_{\text{a}} = 0.3$, and mass density $\rho_{\text{a}} = 2700\,\text{kg/m}^3$. The cover-sheet material is assumed as steel with the following properties: $E_{\text{s}} = 210\,\text{GPa}$, $\nu_{\text{s}} = 0.3$, and $\rho_{\text{s}} = 8050\,\text{kg/m}^3$. In order to test the present approach

[2]The CT data of the aluminium foam-core has been provided by Prof. A. Düster and Dr. Meysam Joulaian [365, 366] from Hamburg University of Technology.

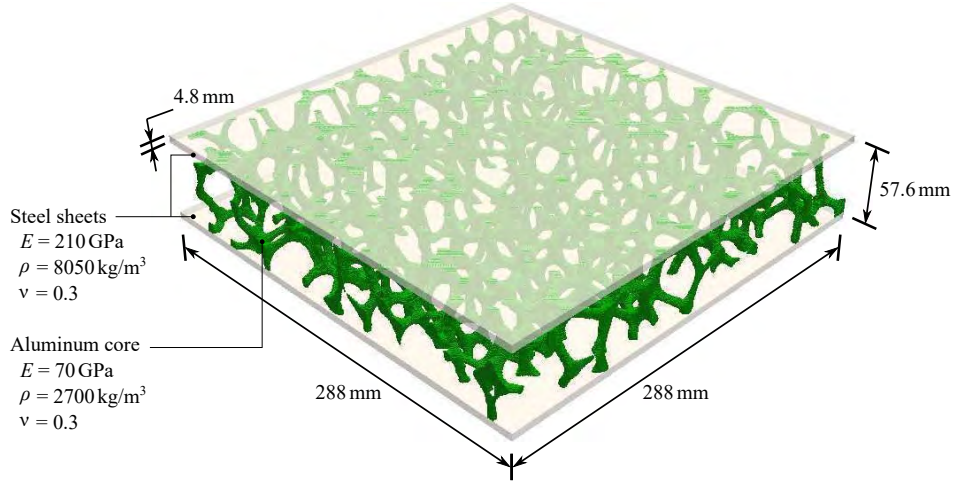
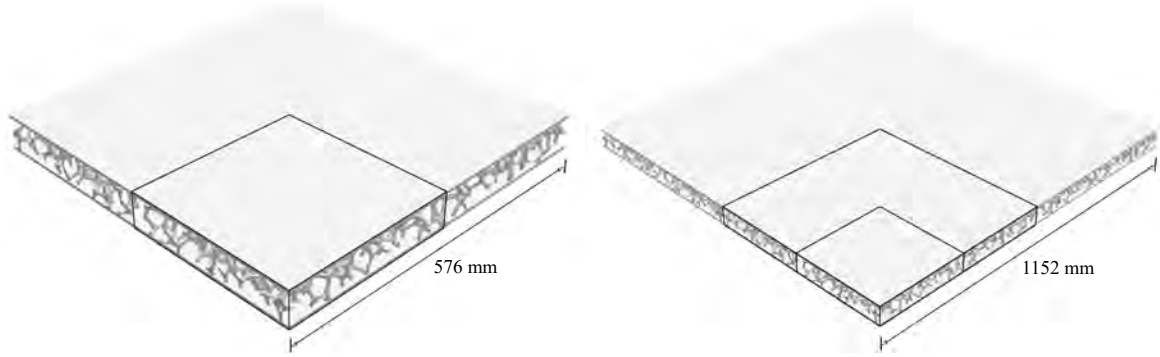


Figure 4.28: Geometry of the sandwich panel



(a) 2×2 panel (b) 4×4 panel

Figure 4.29: Mirroring operation of the sandwich panel

for massively parallel computation of large-scale wave propagation problems and for studying its weak scaling behaviour, we mirror the given panel twice along its section, as illustrated in Fig. 4.29. The first mirroring operation creates a 2×2 panel shown in Fig. 4.29a, where the outer edge length is 576 mm. The second mirroring operation creates a 4×4 panel shown in Fig. 4.29b with an outer edge length of 1152 mm.

The top view of the panel is shown in Fig. 4.30. The origin of the Cartesian coordinate system is placed at the bottom-left corner. The displacement DOFs of the panel-sections at $x = 0$ and $y = 0$ are fixed in the orthogonal direction of the sections. The wave excitation, which is usually achieved by using the piezoelectric shear trans-

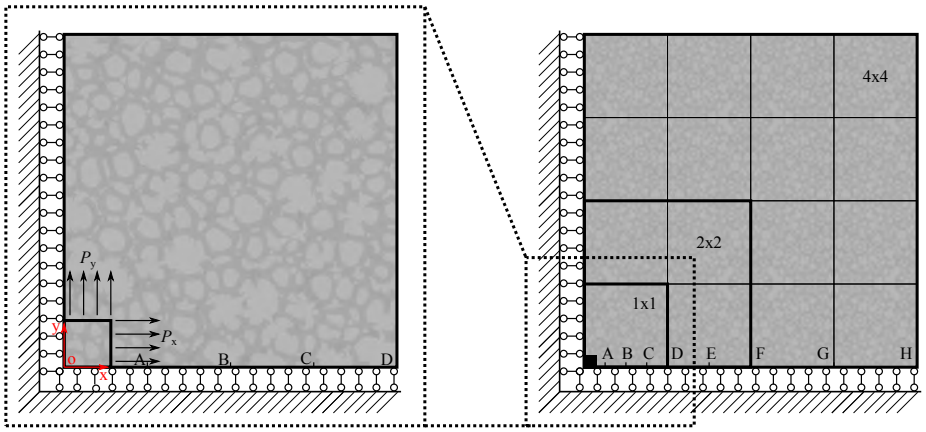


Figure 4.30: Boundary condition of the sandwich panel with an enlarged view of the loading area (left)

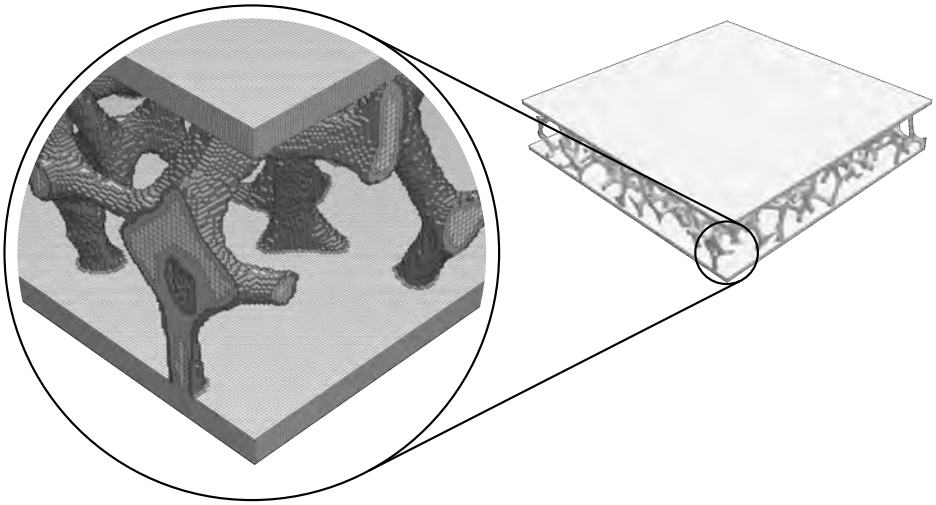
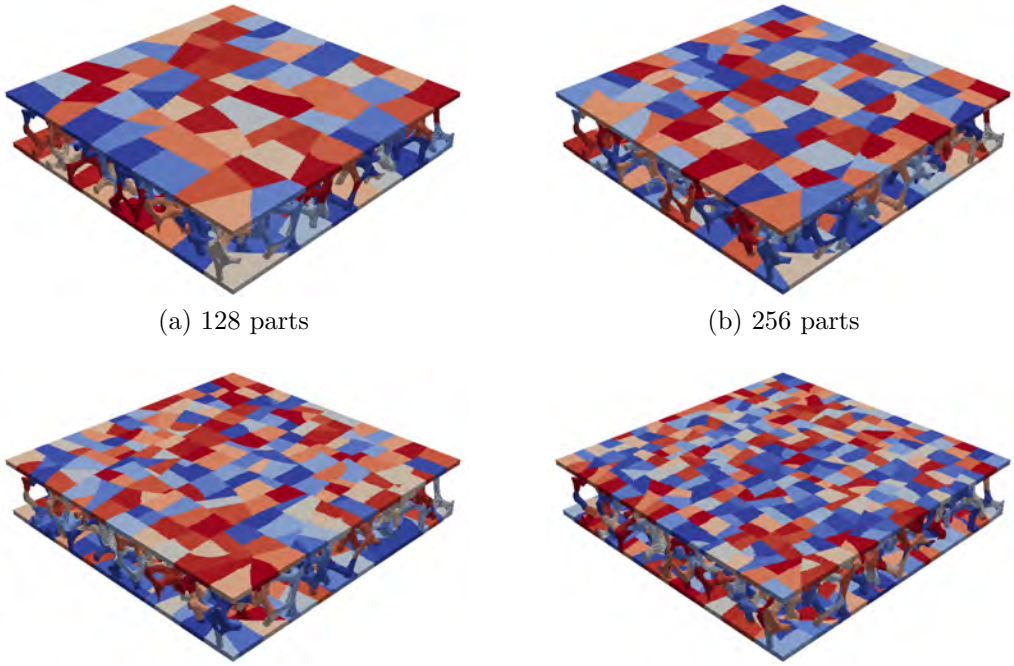


Figure 4.31: Mesh of the sandwich panel

ducers attached to the plate, is applied as uniformly distributed line loads (P_x and P_y) along the two line segments on the top-plate. These two line segments describe the outer contour of the transducers. The fundamental idea for this excitation is based on the pin-force model, which is often adopted in the simulation of guided waves in the context of structural health monitoring [363, 367, 368]. It avoids the adverse effects associated with the concentrated force and thus, the excitation is achieved more realistically without the need of including multi-physics elements with piezoelectric capabilities. P_x is applied in x-direction on the line segment defined by ($x = 19.2\,\mathrm{mm}$,



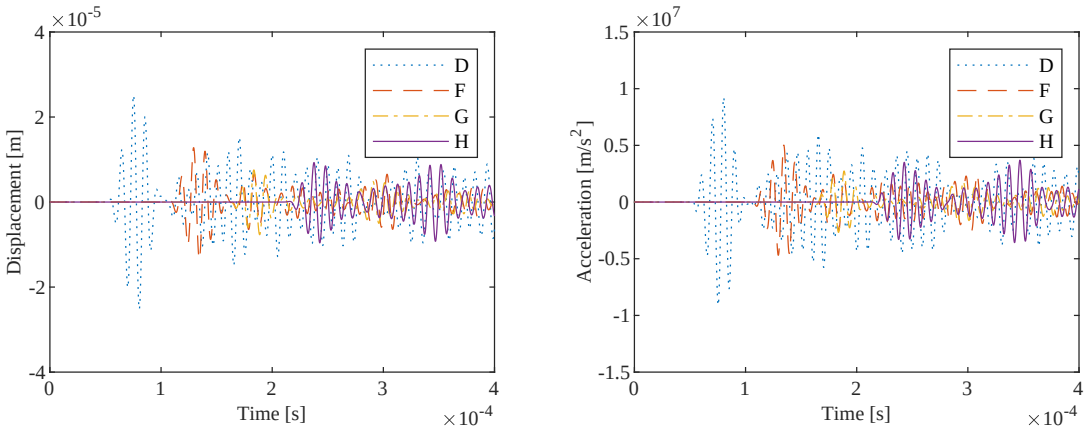
(a) 128 parts (b) 256 parts

(c) 512 parts (d) 1024 parts

Figure 4.32: Mesh partition of the 1×1 sandwich panel

$0\,\text{mm} \leq y \leq 19.2\,\text{mm}$ and $z = 57.6\,\text{mm}$), while P_y is applied in y-direction on the line segment given by ($0\,\text{mm} \leq x \leq 19.2\,\text{mm}$, $y = 19.2\,\text{mm}$ and $z = 57.6\,\text{mm}$). Both of the distributed loads vary over time following the sine-burst signal depicted in Fig. 4.9e with $t_1 = 0.050\,\text{ms}$, $n = 5$, and $P_0 = 2.1 \times 10^4\,\text{N/mm}$. From Section 4.5.1, the central frequency of the sine-burst excitation is calculated as 100 kHz, and the maximum frequency of interest as 116 kHz.

For aluminium, the dilatational and shear wave speeds are calculated as 6198 m/s, and 3122 m/s, respectively. For steel, the corresponding wave speeds are 5926 m/s and 3168 m/s, respectively. Therefore, the shear wavelengths at the maximum frequency of interest in the aluminium and steel parts are 26.9 mm and 27.3 mm, respectively. To accurately solve the wave-propagation problem, a fine discretisation of the sandwich panel geometry is created using octree mesh with a maximum element size of 0.96 mm and a minimum element size of 0.24 mm. The octree mesh generated for the 1×1 panel

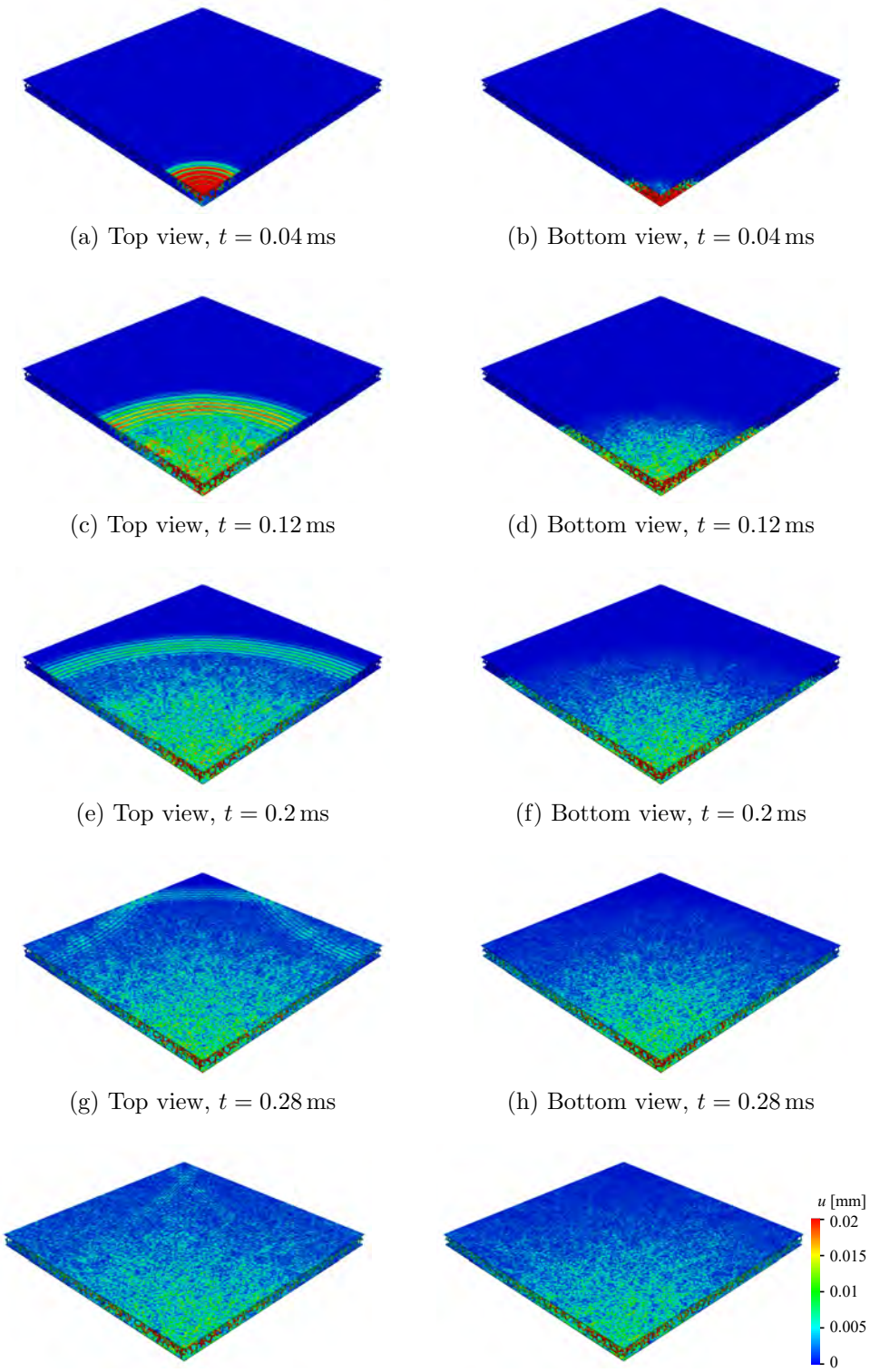


(a) Displacement (b) Acceleration

Figure 4.33: Response history of selected points of the sandwich panel

is shown in Fig. 4.31, which consists of 13,520,194 elements and 66,645,588 DOFs. This is significantly lesser than a uniform voxel-based mesh that would contain 89,862,518 elements and 288,049,833 DOFs to achieve a similar level of discretisation. The octree mesh for the 2×2 and 4×4 panels are also generated. The 2×2 panel mesh contains 54,080,776 elements and 266,295,885 DOFs, while the 4×4 panel mesh has 216,323,104 elements and 1,064,602,902 DOFs. The mesh partitioning is done using the *meshpart* package [331]. The mesh-partitioning of 1×1 panel for the selected number of partitions is shown in Fig. 4.32.

The time step for this example is calculated as $\Delta t = 2.98 \times 10^{-5}$ ms according to Eq. (4.14). We first perform a full wave propagation simulation for 4×4 panel with a total duration of 0.4 ms. The displacement and acceleration along the $x-$direction of four selected points shown in Fig. 4.30 (coordinates are $D(288.0, 0.0, 57.6)$, $F(576.0, 0.0, 57.6)$, $G(864.0, 0.0, 57.6)$, and $H(1152.0, 0.0, 57.6)$, unit: mm) are plotted in Fig. 4.33. Further, the magnitude of the displacement response at the five time instances are presented in Fig. 4.34, illustrating the wave propagation process in the top and bottom plates. It can be observed that the propagating waves in the top plate interact with the foam-core; hence, the wave energy gradually leaks from the top plate into the bottom plate.



(a) Top view, $t = 0.04\,\mathrm{ms}$

(b) Bottom view, $t = 0.04\,\mathrm{ms}$

(c) Top view, $t = 0.12\,\mathrm{ms}$

(d) Bottom view, $t = 0.12\,\mathrm{ms}$

(e) Top view, $t = 0.2\,\mathrm{ms}$

(f) Bottom view, $t = 0.2\,\mathrm{ms}$

(g) Top view, $t = 0.28\,\mathrm{ms}$

(h) Bottom view, $t = 0.28\,\mathrm{ms}$

(i) Top view, $t = 0.36\,\mathrm{ms}$

(j) Bottom view, $t = 0.36\,\mathrm{ms}$

Figure 4.34: Contour of displacement magnitude at various time instances t

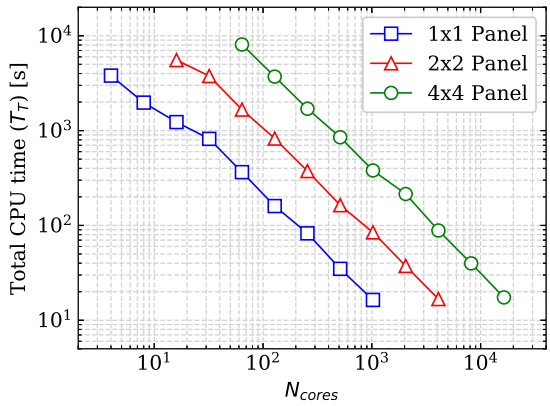
Table 4.3: Computational time of the in-house code for the sandwich panel

N_{cores}	1×1 panel			2×2 panel			4×4 panel		
	T_C^m	T_{W}^m	T_{T}	T_C^m	T_{W}^m	T_{T}	T_C^m	T_{W}^m	T_{T}
4	3,790.8	12.54	3,803.7	-	-	-	-	-	-
8	1,937.8	36.19	1,974.4	-	-	-	-	-	-
16	1,175.1	55.04	1,230.4	5,317.2	232.9	5,550.7	-	-	-
32	709.8	108.8	818.73	3,253.8	514.9	3,769.1	-	-	-
64	283.48	81.35	364.87	1,320.6	347.4	1,668.2	6,316.7	1,780.3	8,097.6
128	119.05	41.04	160.11	620.23	204.4	824.66	2,699.8	1,010.4	3,710.5
256	44.49	38.07	82.57	258.58	115.9	374.49	1,184.5	516.15	1,700.8
512	21.61	13.25	34.87	120.45	43.02	163.48	553.94	297.26	851.26
1,024	9.92	6.46	16.37	47.45	37.05	84.63	264.50	115.35	379.87
2,048	-	-	-	19.60	17.82	37.42	103.23	111.36	214.60
4,096	-	-	-	9.81	6.92	16.73	42.76	45.70	88.46
8,192	-	-	-	-	-	-	19.58	20.20	39.78
16,384	-	-	-	-	-	-	9.81	7.60	17.41

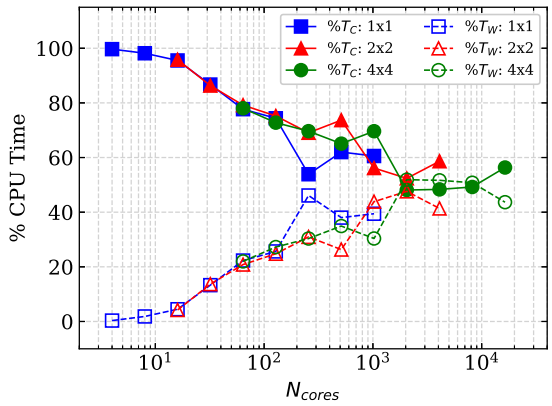
T_C^m: Mean calculation time [s] T_W^m: Mean comm.+wait time [s] T_T: Total CPU time [s]

The computational time of the wave propagation analysis in sandwich panels is shown in Fig. 4.35. For convenience, the results are also presented in Tables 4.3– 4.5. The scaling study is performed on Gadi supercomputer at NCI Canberra. A typical compute node on Gadi has two sockets, each equipped with a 24-core Intel Xeon Scalable 'Cascade Lake' processor and 35.75 MB L3 Cache.

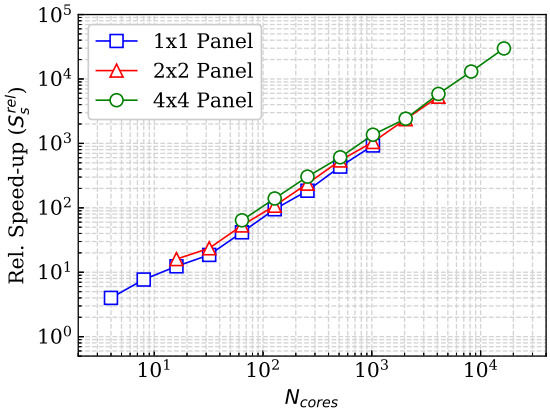
The CPU time is obtained for first 1000 time steps of the analysis. From Fig. 4.35a, it is observed that a consistent scaling is obtained for all the panel sizes over a large range of number of CPU cores. As shown in Fig. 4.35b, the %Communication+Wait time ($\%T_W$) increases steadily from 0.3% to 50% over a range of ∼16.4K cores, which demonstrates the robustness of the massively parallel implementation of the explicit solver using the octree-based approach. In order to show the parallel efficiency of the solver over the given range of N_{cores}, we perform two scaling studies as explained in Section 4.4.3: *The strong and the weak scaling.*



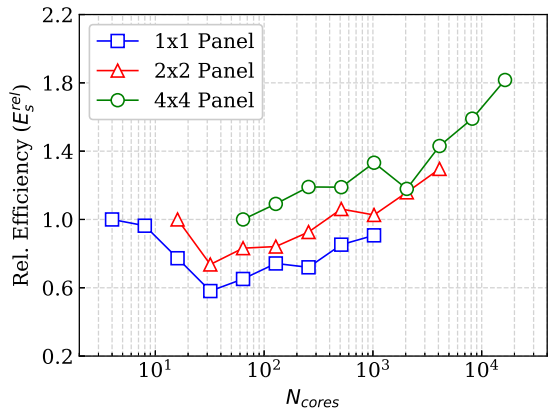
(a) Total CPU time



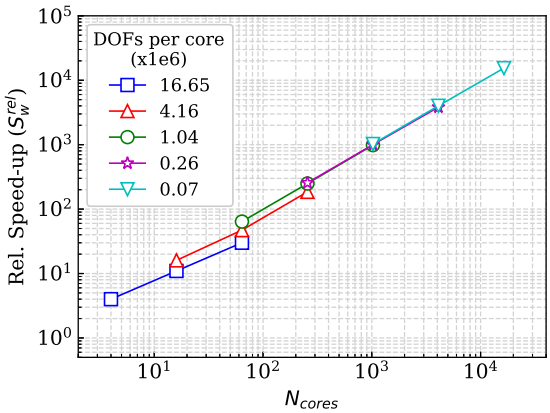
(b) Total CPU time components



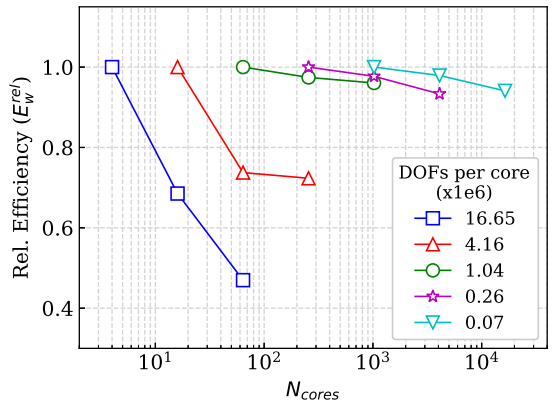
(c) Strong scaling: Rel. Speed-up



(d) Strong scaling: Rel. Efficiency



(e) Weak scaling: Rel. Speed-up



(f) Weak scaling: Rel. Efficiency

Figure 4.35: Computational time study of the in-house code for the sandwich panels

Table 4.4: Strong scaling results of the sandwich panel

N_{cores}	1×1 panel		2×2 panel		4×4 panel	
	S_s^{rel}	E_s^{rel}	S_s^{rel}	E_s^{rel}	S_s^{rel}	E_s^{rel}
4	4.00	1.00	-	-	-	-
8	7.71	0.96	-	-	-	-
16	12.37	0.77	16.00	1.00	-	-
32	18.58	0.58	23.56	0.74	-	-
64	41.70	0.65	53.24	0.83	64.00	1.00
128	95.03	0.74	107.69	0.84	139.67	1.09
256	184.27	0.72	237.15	0.93	304.71	1.19
512	436.36	0.85	543.25	1.06	608.80	1.19
1,024	929.21	0.91	1,049.42	1.02	1,364.27	1.33
2,048	-	-	2,373.29	1.16	2,414.93	1.18
4,096	-	-	5,309.59	1.30	5,858.25	1.43
8,192	-	-	-	-	13,028.57	1.59
16,384	-	-	-	-	29,775.04	1.82

S_s^{rel}: Relative speed-up (Strong scaling) E_s^{rel}: Relative efficiency (Strong scaling)

Table 4.5: Weak scaling results of the sandwich panel

DOFs per core ($\times 10^6$)	Panel	N_{cores}	T_{T} [s]	S_w^{rel}	E_w^{rel}
16.65	1×1	4	3,803.67	4.00	1.00
	2×2	16	5,550.67	10.96	0.69
	4×4	64	8,097.55	30.06	0.47
4.16	1×1	16	1,230.24	16.00	1.00
	2×2	64	1,668.21	47.20	0.74
	4×4	256	1,700.79	185.17	0.72
1.04	1×1	64	364.87	64.00	1.00
	2×2	256	374.49	249.42	0.97
	4×4	1,024	379.87	983.57	0.96
0.26	1×1	256	82.57	256.00	1.00
	2×2	1,024	84.63	999.07	0.98
	4×4	4,096	88.46	3,823.01	0.93
0.07	1×1	1,024	16.37	1,024.00	1.00
	2×2	4,096	16.73	4,009.64	0.98
	4×4	16,384	17.41	15,413.05	0.94

S_w^{rel}: Relative speed-up (Weak scaling) E_w^{rel}: Relative efficiency (Weak scaling)

The *strong scaling* or *fixed model-size scaling* results are shown in Fig. 4.35c-d. It is observed that for $N_{cores} \leq 48$, the efficiency tends to decrease due to memory-bandwidth limitation on a single compute node (that has 2×24 cores), which is consistent with the previous examples. This behaviour is reflected in the calculation time T_C^m of Table 4.3 that scales poorly on a single compute node. Another reason for the reduction in efficiency is the increase in communication and waiting time T_W^m.

For $N_{cores} > 48$ (multiple compute nodes), the efficiency tends to increase and in some cases, the values are even higher than 1. This phenomenon is called the super-linear speed-up, which is mostly caused due to the *cache-effect* as explained here: To improve the computational efficiency of CPUs, high-speed *cache memories* are installed in close proximity to the processors/cores. These memory are much smaller in size, provide lower latencies, and have higher data retrieval rate than *random access memory* or RAM. During the strong scaling studies, as the mesh is partitioned into smaller chunks and distributed to more number of compute nodes, a larger portion of the model data can fit into the cache memory. Therefore, the data requested by a core can be fetched more frequently from the cache memory, which tends to increase the efficiency of the analysis.

In a distributed computing environment involving thousands of cores across multiple compute nodes, the *weak scaling* or *fixed partition-size scaling* allows better utilisation of computational resources for an increasing number of cores. In such scaling, as the problem size per core (or per compute node) remains constant, the cache memory is proportionally utilised as compared to the model data size for increasing number of compute nodes; thus, the super-linear speed-up behaviours are effectively avoided.

The weak scaling results are shown in Fig. 4.35e-f for various values of DOFs per core, where it is observed that the efficiency tends to decrease with an increase in number of cores, when $N_{cores} \leq 48$. One possible reason for this is the limited availability of the memory-bandwidth on single compute node that tends to reduce the performance.

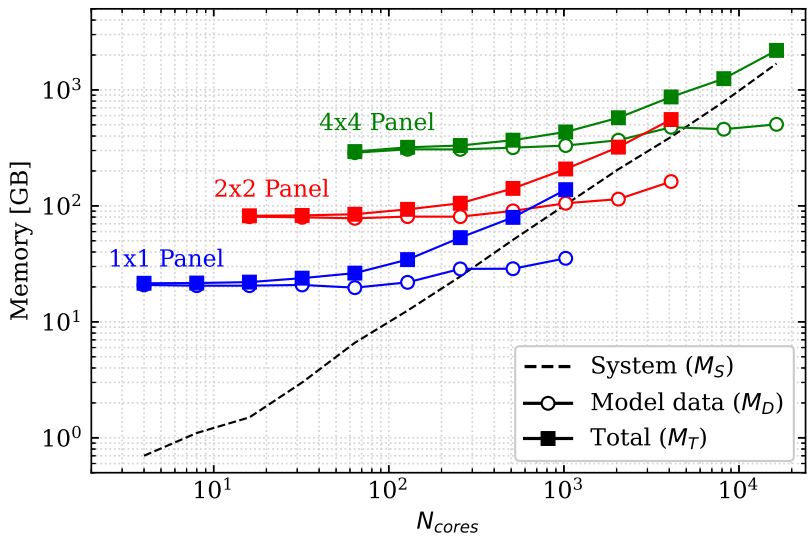


Figure 4.36: Memory usage of the in-house code for the explicit analysis of the sandwich panels

It is also noticed that for $N_{cores} > 48$, the efficiency does not decrease much with an increasing number of cores. This can be attributed to the proportional usage of the available cache memory with respect to the model data size, as explained before.

The memory usage in carrying out explicit dynamic analysis of the sandwich panels is shown in Fig. 4.36 and tabulated in Table 4.6. The total memory usage M_T represents the maximum memory usage attained at any point while carrying out the analysis. It has two components: The system memory usage M_S (which includes Python library, Open-MPI-4.0.1 and other system files) and the model data memory usage M_D (which includes critical data required to carry out analysis, such as mesh data, force/displacement vectors and other temporary variables). The system memory usage M_S can be computed by running a computer program, where all the libraries are loaded without running the analysis. Then, the model data memory usage can be computed as $M_D = M_T - M_S$.

It is observed that the system memory usage is almost proportional to the number of cores utilised during the analysis. This is because in an MPI environment, each

Table 4.6: Memory usage of the in-house code for the explicit analysis of the sandwich panels

N_{cores}	1×1 panel			2×2 panel			4×4 panel		
	M_S	M_D	M_T	M_S	M_D	M_{T}	M_S	M_D	M_T
4	0.7	20.8	21.5	-	-	-	-	-	-
8	1.1	20.5	21.6	-	-	-	-	-	-
16	1.5	20.5	22.0	1.5	80.6	82.1	-	-	-
32	3.0	20.8	23.8	3.0	79.7	82.7	-	-	-
64	6.6	19.7	26.3	6.6	78.1	84.7	6.6	287.3	293.9
128	12.5	21.9	34.4	12.5	80.8	93.3	12.5	307.3	319.8
256	24.6	28.6	53.2	24.6	80.7	105.3	24.6	306.8	331.4
512	50.9	28.7	79.6	50.9	90.5	141.4	50.9	317.5	368.4
1,024	102.6	35.3	137.9	102.6	105.3	207.9	102.6	331.1	433.7
2,048	-	-	-	206.8	114.3	321.1	206.8	368.2	575.0
4,096	-	-	-	393.1	162.0	555.1	393.1	476.2	869.3
8,192	-	-	-	-	-	-	791.9	458.1	1,250.0
16,384	-	-	-	-	-	-	1,685.5	504.5	2,190.0

M_S: System memory [GB], M_D: Model data memory [GB], M_T: Total memory [GB]

process (running on individual cores) is assigned a separate memory space. The system files are copied into each memory space so that the computations can be carried out locally in each process. On the other hand, the model data memory usage increases only marginally with increase in the number of cores, which can be attributed to the requirement of copying the octree-pattern data to each process (which consumes little memory), addition of more DOFs due to mesh partitioning, etc.

4.6 Summary

In this chapter, a parallel explicit solver based on the CDM is developed, exploiting the inherent advantages of the octree meshes. The solver is developed in the framework of SBFEM, which allows efficient handling of octree meshes without any special

treatment of hanging nodes. A balanced octree mesh is utilised with a limited number of cell patterns. It allows the pre-computation of stiffness and mass matrices with much smaller memory requirements. A pattern-by-pattern (PBP) approach is proposed to compute the nodal force vector, which allows recasting the work into dense matrix computation leading to significant performance gains. The advantages of an explicit time integrator are leveraged by extending the special mass lumping technique [346] to three-dimensional problems. It enables the nodal displacements to be computed without the need for simultaneously solving a system of linear equations. The parallelisation is carried out using the mesh-partitioning strategy, where the mesh is divided into several partitions and assigned to individual workers. Each worker performs the local computations in each mesh part, exchanging the interfacial nodal forces with the neighbour partitions using the message passing technique implemented using the MPI library.

The implementation of the explicit solver is verified using the benchmark examples, where optimal convergence rates are obtained for the eigenfrequencies in the modal analysis and the displacement and acceleration responses in the transient analysis. The computational performance and scalability of the solver are demonstrated utilising three numerical examples of varying complexity: the multi-storey building and ground, the mountainous region, and the sandwich panel. A Ricker wavelet, a triangular pulse load, and a sine-burst signal are considered in these examples to excite the structures and to illustrate realistic and high-resolution engineering applications. Using the multi-storey building and ground example, the in-house code is shown to achieve an improved efficiency using octree mesh, when compared to the commercial software ABAQUS utilising tetrahedral mesh producing similar accuracy. Simulation of wave propagation in the mountainous region showed the applicability of the solver in modelling the seismic-like events of moderately long duration. Finally, using the sandwich panel model, an efficient and robust implementation of the solver is demonstrated by carrying out massively

parallel simulations of the very large-scale problem of approx. one billion DOFs and utilising $\sim$16.4K computing cores.

Chapter 5

An octree-based parallel implicit solver for elastostatic and elastodynamic analyses

In the previous chapter, a massively parallel implementation of the explicit time integration scheme for elastodynamic problems was demonstrated using the octree-based approach. However, in many engineering applications, it is often more advantageous to use the solvers based on implicit-time integration schemes. The aim of this chapter is to examine the efficacy of the octree-based formulation for the parallel implementation of implicit solvers for elastostatic and elastodynamic problems. The chapter is organised as follows.

Section 5.1 introduces the inertial problems and implicit time integration schemes. This is followed by the development of the parallel implicit-elastodynamics solver based on HHT-α time integration in Section 5.2. In Section 5.3, iterative solvers such as the preconditioned conjugate gradient (PCG) are discussed to solve a system of linear equations in elastostatic and elastodynamic analyses. Then, a formulation is developed to implement the parallel PCG solver utilising the octree meshes for the analysis of

large-scale structures. A few numerical examples are presented with static and dynamic loading in Section 5.4 to show the efficacy of the developed solvers in solving various problems of practical interest. Finally, general remarks and a summary are given in Section 5.5.

5.1 Introduction

There are two general classes of elastodynamic problems – wave propagation and inertial problems. In wave propagation problems, the structural response is dominated by the intermediate and high-frequency modes throughout the time span of interest (covered in detail in Chapter 4). On the other hand, the response in inertial problems is mainly governed by a relatively small number of low-frequency modes. Typical applications of the latter category include the seismic response of structures such as buildings, dams, and bridges [369, 370], seismic pseudo-dynamic testing [371], vibration isolation of buildings [372], etc. Effective simulations of these problems require specific algorithmic properties, such as excellent numerical dissipation, desirable energy conservation, and efficient and accurate computations of long-duration analyses.

The implicit integration schemes [11, 373] are usually employed to solve inertial problems as they have the aforementioned algorithmic properties in general, and thus, show satisfactory performance. In these methods, the displacements at a given time step are obtained using the velocities and accelerations at the same time step, which requires a system of equations to be solved simultaneously. Therefore, the computational cost per time step is much higher than the explicit schemes, and storage requirements tend to increase dramatically with the increasing size of the finite element mesh. Nonetheless, most of the implicit algorithms are unconditionally stable (especially for linear analysis). Therefore, they permit large a time step size that is governed by accuracy only (without stability consideration), which makes the implicit schemes very suitable for performing

long-duration analyses.

A wide variety of implicit time integration schemes have been developed by researchers, as reviewed in Section 2.2.2. In many engineering applications, such as those involving seismic response, the requirement of controlled numerical dissipation and high accuracy in the calculation [374] is critical. The numerical dissipations are usually designed to filter out the spurious and non-physical high-frequency modes induced by spatial discretisation. However, the addition of high-frequency dissipation should neither reduce the accuracy nor induce damping in the low-frequency modes. These requirements are reasonably satisfied by α-schemes [147–149], and thus, they are widely used in dynamic analyses of structures.

The implicit time integration schemes have been employed extensively in the framework of the scaled boundary finite element method (SBFEM) for dynamic analyses [72, 121, 342–345]. However, unlike the conventional finite element formulation, where dynamic problems can be solved straightforwardly by constructing a mass matrix based on the same interpolation functions as the static stiffness matrix, employing the SBFEM for solving elastodynamic problems is more complicated. In many existing works, higher-order shape functions of the elements have been employed, which require complex procedures and additional non-physical degrees of freedom. Further, the efficacy of such formulations for performing large-scale dynamic analyses has not yet been demonstrated.

In the present chapter, the octree-based formulation (developed in Chapter 4) is extended for the implementation of a massively parallel implicit (HHT-α) solver in the framework of the scaled boundary finite element method (SBFEM). The balanced octree mesh with a limited number of octree patterns is used for domain discretisation, which allows the pre-computation of the element stiffness and mass matrices. These matrices require much smaller memory than the individual (un-grouped) element matrices needed for the entire structure. The computationally most demanding step – the solution of

a system of linear equations – is solved using an iterative solver (PCG) utilising the pattern-by-pattern (PBP) approach. Here, the octree-patterns are exploited to group all the elements of the same shape (nodal pattern), which allows performing efficient matrix operations associated with the iterative solver. In addition, the developed PCG solver is also applied for efficiently solving elastostatic problems, as shown later in this chapter. The parallelisation is carried out using the mesh-partitioning strategy introduced in Section 2.4.3. Here, each worker performs local computation within an individual partition, and the data between individual partitions are synchronised using the message passing algorithm. A few numerical examples are presented to demonstrate the efficacy and robustness of the developed solver in solving very large-sized structural dynamics problems using thousands of workers.

5.2 Implicit elastodynamics

The governing equilibrium equation in a structural dynamics problem is given as

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{f}^{\text{int}} = \mathbf{f}^{\text{ext}}, \tag{5.1}$$

where, $\mathbf{M}$ is the discrete mass matrix, $\mathbf{C}$ is the viscous damping matrix, $\mathbf{f}^{\text{int}}$ is the internal force vector due to element stresses, and $\mathbf{f}^{\text{ext}}$ is the external load vector. $\mathbf{u}$, $\dot{\mathbf{u}}$, and $\ddot{\mathbf{u}}$ are the nodal displacement, velocity, and acceleration vectors, respectively. For linear structural problems, $\mathbf{f}^{\text{int}} = \mathbf{K}\mathbf{u}$, where $\mathbf{K}$ is the global stiffness matrix.

In the present chapter, the HHT-α method [147] is used for the implicit time integration of the governing equilibrium equation. This method is second-order accurate and provides controlled dissipation of higher frequency modes.

5.2.1 Implicit time integration using HHT-α method

According to Hilber et al. [147], the equation of motion in Eq. (5.1) can be rewritten at the $(j+1)^{th}$ time step as

$$\mathbf{M}\ddot{\mathbf{u}}_{j+1} + \mathbf{C}\dot{\mathbf{u}}_{j+1} + (1+\alpha)\mathbf{K}\mathbf{u}_{j+1} - \alpha\mathbf{K}\mathbf{u}_j = \mathbf{f}_{j+1}^{\text{ext}}. \tag{5.2}$$

Here, the parameter α is utilised to control the numerical damping properties of the method. Introducing parameters γ and β and time step size $\triangle t$, the nodal displacements and velocities at the time step $(j+1)$ can be written as

$$\mathbf{u}_{j+1} = \mathbf{u}_j^* + (\triangle t)^2 \beta \ddot{\mathbf{u}}_{j+1}, \tag{5.3}$$

$$\dot{\mathbf{u}}_{j+1} = \dot{\mathbf{u}}_j^* + \triangle t \gamma \ddot{\mathbf{u}}_{j+1}, \tag{5.4}$$

where,

$$\mathbf{u}_j^* = \mathbf{u}_j + \triangle t \dot{\mathbf{u}}_j + (\triangle t)^2 \left(\frac{1}{2} - \beta\right) \ddot{\mathbf{u}}_j, \tag{5.5}$$

$$\dot{\mathbf{u}}_j^* = \dot{\mathbf{u}}_j + \triangle t \left(1-\gamma\right) \ddot{\mathbf{u}}_j. \tag{5.6}$$

Substituting Eqs. (5.3) and (5.4) into Eq. (5.2) yields

$$\mathbf{M}\ddot{\mathbf{u}}_{j+1} + \mathbf{C}\left[\dot{\mathbf{u}}_j^* + \triangle t\gamma\ddot{\mathbf{u}}_{j+1}\right] + (1-\alpha)\mathbf{K}\left[\mathbf{u}_j^* + (\triangle t)^2 \beta\ddot{\mathbf{u}}_{j+1}\right] + \alpha\mathbf{K}\mathbf{u}_j = \mathbf{f}_{j+1}, \tag{5.7}$$

that can be rearranged as

$$\left[\mathbf{M} + \triangle t\gamma\mathbf{C} + (1-\alpha)\left(\triangle t\right)^2 \beta\mathbf{K}\right]\ddot{\mathbf{u}}_{j+1} = \mathbf{f}_{j+1} - \mathbf{K}\left[(1-\alpha)\mathbf{u}_j^* + \alpha\mathbf{u}_j\right] - \mathbf{C}\dot{\mathbf{u}}_j^*. \tag{5.8}$$

Alternatively, Eq. (5.8) can be expressed as

$$\mathbf{M}_c\ddot{\mathbf{u}}_{j+1} = \mathbf{f}_c, \tag{5.9}$$

with,

$$\mathbf{M}_c = \mathbf{M} + \triangle t\gamma\mathbf{C} + (1-\alpha)\left(\triangle t\right)^2 \beta\mathbf{K}, \tag{5.10a}$$

$$\mathbf{f}_c = \mathbf{f}_{j+1} - \mathbf{C}\dot{\mathbf{u}}_j^* - \mathbf{K}\left[(1-\alpha)\mathbf{u}_j^* + \alpha\mathbf{u}_j\right]. \tag{5.10b}$$

Numerical damping is commonly introduced by setting $\alpha = -0.05$, which will be utilised for all examples of dynamics in this paper. The other parameters can be computed as

$$\beta = \frac{1}{4}(1-\alpha)^2, \tag{5.11a}$$

$$\gamma = \frac{1}{2} - \alpha. \tag{5.11b}$$

Note that when $\alpha = 0$, the standard Newmark-beta algorithm for constant average acceleration method is obtained with $\beta = \frac{1}{4}$ and $\gamma = \frac{1}{2}$.

At the beginning of the simulation, the system is assumed to be at rest; thus, $\mathbf{u}_0 = \mathbf{0}$, $\dot{\mathbf{u}}_0 = \mathbf{0}$, $\ddot{\mathbf{u}}_0 = \mathbf{M}_c^{-1}\mathbf{f}_0$. In this work, the damping matrix is neglected ($\mathbf{C} = 0$). Most of the computational effort is spent in solving the system of linear equations given by Eq. (5.9)) to obtain the acceleration of the next time step $\ddot{\mathbf{u}}_{j+1}$. For a very large-sized model, iterative procedures such as the preconditioned conjugate gradient (PCG) method become necessary to reduce memory requirements and CPU time, as described in Section 5.3.1. Afterwards, the nodal displacements and nodal velocities can be solved using Eqs. (5.3) and (5.4), respectively.

5.2.2 Parallel implementation

In the present work, the parallel HHT-α method is implemented utilising the mesh-partitioning strategy built on MPI-based parallel programming model (Ref: Sections 2.4.3 and 4.4.1). Here, the problem domain is divided into a set of smaller partitions, where each partition is assigned to the individual workers. The parallelisation is achieved by the individual workers running a single computer program and carrying out the simultaneous local computations over the assigned partitions (SPMD paradigm). The data among the neighbouring partitions are synchronised through explicit message passing.

The pseudo code for the implementation of the parallel HHT-α method is given in Algorithm 5.1. The solver starts with creating a set of MPI processes that are bind to the cores/socket of the computer hardware. For optimal computational performance,

Algorithm 5.1 Pseudo code for parallel HHT-α algorithm

1: Start MPI process; Create MPI communicator object: Comm
2: Load partitioned mesh-data file
3: Use barrier: Comm.barrier()
4: Initialise variables: $\mathbf{u}_0 = \mathbf{0}$, $\dot{\mathbf{u}}_0 = \mathbf{0}$
5: Solve for $\ddot{\mathbf{u}}_0$: $\mathbf{M}_c\ddot{\mathbf{u}}_0 = \mathbf{f}_0$ ▷ using parallel PCG solver: Algorithm 5.3
6: **for** $j = 0 : N_{Tmax}$ **do**
7: Compute $\mathbf{f}_c$ ▷ using Eq. (5.10b)
8: Solve for $\ddot{\mathbf{u}}_{j+1}$: $\mathbf{M}_c\ddot{\mathbf{u}}_{j+1} = \mathbf{f}_c$ ▷ using parallel PCG solver: Algorithm 5.3
9: Calculate $\mathbf{u}_{j+1}$ and $\dot{\mathbf{u}}_{j+1}$ ▷ using Eqs. (5.3) and (5.4), respectively
10: **end for**

each MPI process carries out computation on each mesh partition using exactly one core. In the SPMD paradigm, the same code is executed by all MPI processes, where each MPI process only loads the partitioned mesh-data file assigned to it. It is noted that all the loaded variables are critical for analysis and remain constant during the analysis. A barrier is used in Step 3 of Algorithm 5.1 so that all MPI processes start the analysis at the same time after initialisation.

In each mesh partition, all the vectors such as the $\mathbf{f}_c$, $\mathbf{u}_{j+1}$, $\dot{\mathbf{u}}_{j+1}$ and $\ddot{\mathbf{u}}_{j+1}$ are locally computed. The matrix product operation required in the computation of $\mathbf{f}_c$, given in Step 7 of Algorithm 5.1, is carried out using the octree-based pattern-by-pattern (PBP) approach. The acceleration vector $\ddot{\mathbf{u}}_{j+1}$, given in Steps 5 and 8, is computed using the octree-based parallel PCG solver explained in Section 5.3.3. Once the acceleration vector is calculated, displacement and velocity vectors are calculated using Eqs. (5.3) and (5.4), respectively.

5.3 Linear solvers

A system of linear equations can be given by

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{5.12}$$

where, $\mathbf{A}$ is a square matrix, $\mathbf{x}$ is a vector of unknowns variable to be solved, and $\mathbf{b}$ is a known vector.

The solution procedures for solving linear equations is reviewed detail in Section 2.2.2. For a small system of equations, the direct sparse method is commonly used. The advantage is that an exact solution can be obtained in a fixed number of operations. However, for large-scale systems (e.g., with more than a million degrees of freedom), the direct approach is overly expensive. The CPU and memory requirements of these solvers increase too rapidly with the increasing problem size due to the requirement of numerical factorisation [375, 376]. Therefore, iterative solution procedures yielding approximate solutions are often preferred, which considerably reduce the overall memory requirements. In this work preconditioned conjugate gradient (PCG) [164, 175, 327] based iterative solver is chosen for solving large-scale structural problems, as explained in the next section.

5.3.1 Preconditioned conjugate gradient (PCG) solver

The method of conjugate gradient (CG) is an iterative technique for obtaining the solution of a system of linear equations given by Eq 5.12. This method is typically used for solving equations having sparse, symmetric and positive-definite system matrix $\mathbf{A}$, frequently found in finite element based structural analyses of elastostatic, elastodynamic, and many non-linear problems.

For linear-elastostatics problems (Ref: Section 3.2.4), the governing equilibrium equation is given as

$$\mathbf{K}\mathbf{u} = \mathbf{f}^{\text{ext}}, \tag{5.13}$$

where, $\mathbf{K}$ is the global stiffness matrix, $\mathbf{u}$ is the global displacement vector, and $\mathbf{f}^{\text{ext}}$ is the global external force vector. Comparing with Eq (5.12), we get $\mathbf{A} = \mathbf{K}$, $\mathbf{x} = \mathbf{u}$, and $\mathbf{b} = \mathbf{f}^{\text{ext}}$.

For implicit-elastodynamics problems, the governing equation is expressed in Eq. (5.9) and given by

$$\mathbf{M}_c \ddot{\mathbf{u}}_{j+1} = \mathbf{f}_c, \tag{5.14}$$

where, $\mathbf{M}_c$ is the equivalent mass matrix and $\mathbf{f}_c$ is the equivalent force vector obtained from Eq. (5.10). $\ddot{\mathbf{u}}_{j+1}$ is the acceleration vector at $(j+1)$-th time step. Comparing with Eq (5.12), we get $\mathbf{A} = \mathbf{M}_c$, $\mathbf{x} = \ddot{\mathbf{u}}_{j+1}$ and $\mathbf{b} = \mathbf{f}_c$.

To improve the convergence of the iterative solver, the system of linear equations given by Eq (5.12) can be preconditioned using a suitable matrix $\mathbf{P}$. Then, Eq (5.12) can be solved indirectly by solving

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{x} = \mathbf{P}^{-1}\mathbf{b}. \tag{5.15}$$

The preconditioning typically causes the system matrix to loose its symmetry and definiteness. Nevertheless, through careful transformations and substitution operations, a preconditioned conjugate gradient (PCG) method can be developed to handle this issue [377], as given in Algorithm 5.2.

In the present work, a diagonal (or Jacobi) preconditioner is adopted, where $\mathbf{P} = \text{diag}(\mathbf{A})$. This approach is easy to implement, supports EBE-like framework (Ref: Section 2.2.3), and allows straightforward parallelisation of the algorithm.

The PCG algorithm requires two input parameters: the desired L2-norm of the residual vector ε and the maximum number of iterations allowed $n_{\max}$. If necessary, an initial guess value $\mathbf{x}_0$ can be prescribed, otherwise a zero vector can be inputted. During the iteration step n, the solution vector $\mathbf{x}_n$ is updated at Step 4 of Algorithm 5.2, until the convergence criteria is met or maximum number of iterations is achieved.

For a large-scale problem, the computational cost and the memory demands of the

Algorithm 5.2 Preconditioned conjugate gradient (PCG) method to solve a system of linear equation with a diagonal preconditioner

% Initialise the variables at the beginning

1: Set: $n = 1$, $\mathbf{x}_0$, $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\hat{\mathbf{r}}_0 = \mathbf{P}^{-1}\mathbf{r}_0$, $\mathbf{p}_0 = \hat{\mathbf{r}}_0$

2: **while** $\varepsilon < \| \mathbf{r}_{n-1} \|_{\mathrm{L2}}$ and $n < n_{\max}$ **do**

% Compute the step length $\tilde{\alpha}_n$

3: $\tilde{\alpha}_n = \frac{\rho_{n-1}}{\kappa_{n-1}} = \frac{\mathbf{r}_{n-1}^{\mathrm{T}}\hat{\mathbf{r}}_{n-1}}{\mathbf{p}_{n-1}^{\mathrm{T}}\mathbf{A}\mathbf{p}_{n-1}}$ ▷ with $\hat{\mathbf{r}}_{n-1} = \mathbf{P}^{-1}\mathbf{r}_{n-1}$

% Update the solution $\mathbf{x}_n$

4: $\mathbf{x}_n = \mathbf{x}_{n-1} + \tilde{\alpha}_n \mathbf{p}_{n-1}$

% Update the residual $\mathbf{r}_n$

5: $\mathbf{r}_n = \mathbf{r}_{n-1} - \tilde{\alpha}_n \mathbf{A}\mathbf{p}_{n-1}$

% Get the gradient correction factor $\tilde{\beta}_n$

6: $\tilde{\beta}_n = \frac{\rho_n}{\rho_{n-1}} = \frac{\mathbf{r}_n^{\mathrm{T}}\hat{\mathbf{r}}_n}{\mathbf{r}_{n-1}^{\mathrm{T}}\hat{\mathbf{r}}_{n-1}}$ ▷ with $\hat{\mathbf{r}}_{n-1} = \mathbf{P}^{-1}\mathbf{r}_{n-1}$and $\hat{\mathbf{r}}_n = \mathbf{P}^{-1}\mathbf{r}_n$

% Find a new search direction $\mathbf{p}_n$

7: $\mathbf{p}_n = \hat{\mathbf{r}}_n + \tilde{\beta}_n \mathbf{p}_{n-1}$ ▷ with $\hat{\mathbf{r}}_n = \mathbf{P}^{-1}\mathbf{r}_n$

% Next iteration step

8: $n = n + 1$

9: **end while**

PCG algorithm increases significantly in step 3 or 5, where the matrix-vector product operation $\mathbf{q}_{n-1} = \mathbf{A}\mathbf{p}_{n-1}$ is required to be performed once per iteration step. A commonly used approach to handle this issue is the element-by-element (EBE) technique (explained in Section 2.2.3). It avoids the requirement of constructing the global system matrix $\mathbf{A}$ and supports large scale parallelisation. Using this approach, matrix-vector product can be calculated as

$$\mathbf{q}_{n-1} = \sum_e \mathbf{A}^e \mathbf{p}_{n-1}^e, \tag{5.16}$$

where, $\sum$ represents the assembly process as per the element connectivity, and $\mathbf{A}^e$ and $\mathbf{p}_{n-1}^e$ are the system matrix and the search direction vector of the eth element at the $(n-1)^{th}$ iteration step. An alternative approach is to construct a sparse form of the

system matrix $\mathbf{A}$ to compute the matrix-vector product, which reduces the operation count compared to the EBE technique; however, it increases the memory requirements depending on the details of the implementation.

In the present work, the octree-based pattern-by-pattern (PBP) approach (developed in Chapter 4) is extended to perform efficient matrix-vector product operations associated with the PCG algorithm. This approach reduces the memory footprint of the solver due to the usage of a limited number of octree patterns. Further, it allows massive parallelisation using thousands of cores on a distributed computing architecture.

5.3.2 Octree-based matrix operations

Utilising the pattern-by-pattern or PBP approach, the computation of matrix-vector product $\mathbf{q}_{n-1} = \mathbf{A}\mathbf{p}_{n-1}$ required in step 3 or 5 of Algorithm 5.2 can be expressed as

$$\mathbf{q}_{n-1} = \sum_{p=1}^{N_P} \mathbf{Q}_{n-1}^{p} = \sum_{p=1}^{N_P} \mathbf{A}^{p} \left[\begin{array}{cccc} w^{1p}\mathbf{p}_{n-1}^{1p}, & w^{2p}\mathbf{p}_{n-1}^{2p} & \cdots & w^{N_E p}\mathbf{p}_{n-1}^{N_E p} \end{array} \right]. \tag{5.17}$$

Here, N_P is the number of unique patterns in the octree mesh and N_E is the number of elements of pattern p. For the eth element of the given pattern p, $\mathbf{A}_p$ is the system matrix of the master element and $\mathbf{p}_{n-1}^{ep}$ is the nodal search direction vector. w^{ep} is the factor proportional to the edge lengths, Young's modulus, or density of the element, depending on the type of system matrix, as explained below.

For each unique octree cell pattern, we define a master cubic cell (or element) of unit edge length ($L = 1$) and material properties (Young's modulus $E = 1$ and mass density $\rho = 1$) for a given value of Poisson's ratio ν (Ref: Section 4.3.3). The stiffness and mass matrices of each master element can be pre-computed with a small memory requirement. Depending on the edge size and material properties of an element e of a given pattern p, its stiffness $\mathbf{K}^e$ and mass matrices $\mathbf{M}^e$ can be obtained by scaling the

corresponding master element matrices by a suitable factor as

$$\mathbf{M}^e = \rho L^3 \mathbf{M}^p = r^{ep}\mathbf{M}^p, \tag{5.18a}$$

$$\mathbf{K}^e = EL\mathbf{K}^p = s^{ep}\mathbf{K}^p. \tag{5.18b}$$

Here, $\mathbf{M}^p$ and $\mathbf{K}^p$ are the mass and stiffness matrices, respectively, of the master element. In the case where materials with different Poisson's ratios are present, the master element matrices have to be computed for each material separately.

For linear elastostatic problems, $\mathbf{A}^p = \mathbf{K}^p$, where $\mathbf{K}^p$ is the stiffness matrix of the master element with the pattern p. Therefore,

$$\mathbf{q}_{n-1} = \sum_{p=1}^{N_P} \mathbf{Q}^p_{n-1} = \sum_{p=1}^{N_P} \mathbf{K}^p \left[\begin{array}{cccc} s^{1p}\mathbf{p}^{1p}_{n-1}, & s^{2p}\mathbf{p}^{2p}_{n-1} & \cdots & s^{N_Ep}\mathbf{p}^{N_Ep}_{n-1} \end{array} \right]. \tag{5.19}$$

For implicit-elastodynamics problems (referring to Eqs. (5.9) and (5.10) and neglecting damping matrix),

$$\begin{aligned} \mathbf{q}_{n-1} = \sum_{p=1}^{N_P} \mathbf{Q}^p_{n-1} = \sum_{p=1}^{N_P} \Bigg(& \mathbf{M}^p \left[\begin{array}{cccc} r^{1p}\mathbf{p}^{1p}_{n-1}, & r^{2p}\mathbf{p}^{2p}_{n-1} & \cdots & r^{N_Ep}\mathbf{p}^{N_Ep}_{n-1} \end{array} \right] \ldots \\ & + (1-\alpha)(\triangle t)^2 \beta \mathbf{K}^p \left[\begin{array}{cccc} s^{1p}\mathbf{p}^{1p}_{n-1}, & s^{2p}\mathbf{p}^{2p}_{n-1} & \cdots & s^{N_Ep}\mathbf{p}^{N_Ep}_{n-1} \end{array} \right] \Bigg) \end{aligned} \tag{5.20}$$

In order to utilise the unique octree patterns with $N_P = 144$, transformation operations followed by the bincount-based assembly are required to be carried out. These operations are essentially the same as those followed in case of the explicit solver, explained in Section 4.3.3. It can be noticed that the cell patterns are exploited such that a majority of the work is recast to carry out dense matrix computations. This provides a significant performance gain to the PCG solver due to better cache utilisation, aiding the memory bandwidth limited matrix operations.

5.3.3 Parallel implementation

In the present work, a parallel version of the PCG algorithm is developed using the mesh-partitioning strategy and MPI based parallel programming model. The problem domain is divided into a set of smaller partitions, and each partition is assigned to the

individual workers. The parallelisation is achieved by the individual workers running a single computer program and carrying out the simultaneous local computations over the assigned partitions (SPMD paradigm). The data among the neighbouring partitions are synchronised through the message passing algorithm. A review of the MPI programming model, mesh-partitioning, and communication strategies are covered in Section 2.4.2 and 2.4.3. In addition, the CPU time measurements and scaling calculations for parallel processing are discussed in Section 4.4.3.

The pseudo-code of the parallel Preconditioned conjugate gradient (PCG) method with a diagonal preconditioner is presented in Algorithm 5.3. The same code is executed by all MPI processes; however, each MPI process only loads the partitioned mesh-data file assigned to it. A barrier is used so that all MPI processes start the analysis at the same time after initialisation.

In each mesh partition, the variables such as the matrix-vector product $\mathbf{q}_{n-1}$, residual $\mathbf{r}_n$, search direction $\mathbf{p}_n$, and solution $\mathbf{x}_n$ are locally computed. The matrix-vector product $\mathbf{q}_{n-1}$ is obtained using the octree-based PBP approach, explained in Section 5.3.2. The product $\mathbf{q}_{n-1}$ has two components: one corresponding to the nodes inside each partition (internal DOFs) and those shared with neighbouring partitions (interfacial DOFs). The values of the latter part are exchanged with the neighbouring partitions, following which the product $\mathbf{q}_{n-1}$ is updated. The information exchange is implemented using the *direct communication* approach, as illustrated in Fig. 2.5. The message containing the product $\mathbf{q}_{n-1}$ at the interfacial nodes is generally small; therefore, it has low overhead latency per message.

For the parallel computation of the global scalars, such as step-length $\tilde{\alpha}_n$ and gradient correction factor $\tilde{\beta}_n$, the global reduction operation is carried out using the collective communication function *allreduce*. The scalars κ_{n-1} and ρ_n are first locally obtained on each partition using the weighted vector-vector product: $\kappa_{n-1} = \mathbf{p}^T(\mathbf{q}_{n-1} \circ \mathbf{w})$ and $\rho_n = \mathbf{r}_n^T(\hat{\mathbf{r}}_n \circ \mathbf{w})$, where $\mathbf{w}$ is the weight vector and $\circ$ denotes the entry-wise multi-

Algorithm 5.3 Pseudo-code for the parallel preconditioned conjugate gradient (PCG) method with a diagonal preconditioner

1: Start MPI process; Create MPI communicator object: Comm

2: Load partitioned mesh-data file

3: Use barrier: Comm.barrier()

4: Initialise variables: $n = 1$, $\rho_0^G = 1$, $\mathbf{x}_0$, $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$, $\hat{\mathbf{r}}_0 = \mathbf{P}^{-1}\mathbf{r}_0$, $\mathbf{p}_0 = \hat{\mathbf{r}}_0$

5: **while** $\varepsilon < \| \mathbf{r}_{n-1}^G \|_{\mathrm{L2}}$ and $n < n_{\max}$ **do**

% Compute matrix-vector product $\mathbf{q}_{n-1}$

6: Calculate $\mathbf{q}_{n-1}$ locally: $\mathbf{q}_{n-1} = \sum_{p=1}^{N_P} \mathbf{Q}_{n-1}^p$ ▷ Using Eq. 5.17

7: Calculate vector $\mathbf{q}_{n-1}^{\mathrm{intfc}}$ at the each mesh partition interface

8: **for** $m = 1 : N_{\mathrm{nbr}}$ **do**

9: Send $\mathbf{q}_{m,n-1}^{\mathrm{intfc}}$ to m^{th} neighbouring partition (Comm.Isend)

10: **end for**

11: **for** $m = 1 : N_{\mathrm{nbr}}$ **do**

12: Receive $\mathbf{q}_{m,n-1}^{\mathrm{intfc}}$ from m^{th} neighbouring partition (Comm.Recv)

13: **end for**

14: Update the vector $\mathbf{q}_{n-1}$ based on received data

% Compute κ_{n-1}^G

15: $\kappa_{n-1} = \mathbf{p}_{n-1}^{\mathrm{T}}(\mathbf{q}_{n-1} \circ \mathbf{w})$, $\kappa_{n-1}^G = \sum \kappa_{n-1}$ ▷ $\sum$ represents global reduction

% Get the step length $\tilde{\alpha}_n$

16: $\tilde{\alpha}_n = \rho_{n-1}^G / \kappa_{n-1}^G$

% Update the solution $\mathbf{x}_n$ and residual $\mathbf{r}_n$

17: $\mathbf{x}_n = \mathbf{x}_{n-1} + \tilde{\alpha}_n \mathbf{p}_{n-1}$, $\mathbf{r}_n = \mathbf{r}_{n-1} - \tilde{\alpha}_n \mathbf{A}\mathbf{p}_{n-1}$

% Compute ρ_n^G

18: $\rho_n = \mathbf{r}_n^{\mathrm{T}}(\hat{\mathbf{r}}_n \circ \mathbf{w})$, $\rho_n^G = \sum \rho_n$ ▷ with $\hat{\mathbf{r}}_n = \mathbf{P}^{-1}\mathbf{r}_n$

% Get the gradient correction factor $\tilde{\beta}_n$

19: $\tilde{\beta}_n = \rho_n^G / \rho_{n-1}^G$

% Find a new search direction $\mathbf{p}_n$

20: $\mathbf{p}_n = \hat{\mathbf{r}}_n + \tilde{\beta}_n \mathbf{p}_{n-1}$

21: Set the next iteration step: $n = n + 1$

22: **end while**

plication. Then, global reduction operations are carried out for computation of global scalars κ_{n-1}^G and ρ_n^G. The weight vector $\mathbf{w}$ is pre-computed for each partition such that the repeated summation of values at the interfacial nodes during the global reduction operation is avoided. The *allreduce* function ensures that the global variables κ_{n-1}^G and ρ_n^G are scattered back to all the workers. The step-length $\tilde{\alpha}_n$ and gradient correction factor $\tilde{\beta}_n$ can now be computed as: $\tilde{\alpha}_n = \rho_{n-1}^G/\kappa_{n-1}^G$ and $\tilde{\beta}_n = \rho_n^G/\rho_{n-1}^G$. The calculation of global scalars at each iteration step takes place efficiently, since only the locally computed scalars (ρ_n and κ_{n-1}) participate in the collective communication during the global reduction operations.

5.4 Numerical examples

First, two elastostatic problems are presented and solved using the octree-based parallel PCG solver. These include (1) A prismatic beam under pure bending that verifies the implementation of the code using a convergence study. (2) A concrete block under uniaxial static load that shows the capability of employing the parallel solver to a relatively large number of DOF and complex multiscale (or mesoscale) models.

Next, two numerical examples with dynamic loading are presented, which are solved using the HHT-α time integration method. These are: (1) The model of a building surrounded by a trench that shows the efficacy of the developed solver for large-sized structural dynamic problems. (2) The example of a castle on a mountain that demonstrates the practical applicability of the developed formulation for performing the multiscale dynamic simulation. Here, the octree mesh is utilised to efficiently model the geometries at varying scales, including the castle, a fault line and the mountain in a single problem domain.

For the dynamic examples, the minimum number of linear elements that satisfactorily represents a wavelength with acceptable accuracy is typically chosen between 10 and

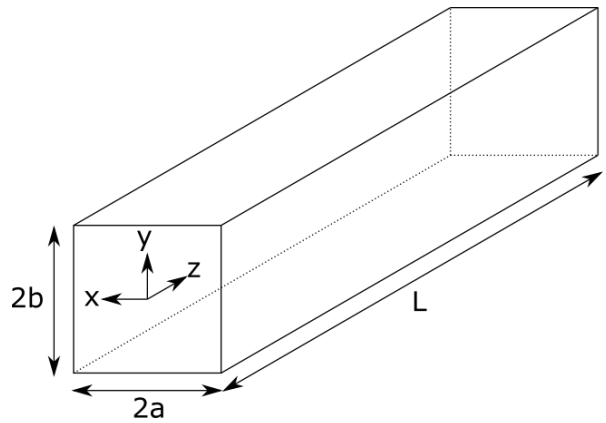


Figure 5.1: Prismatic beam geometry

Table 5.1: List of meshes employed in the prismatic beam example

Mesh ID	Element size	No. of Elements	No. of DOFs
Mesh 1	$a/4$	4,096	15,795
Mesh 2	$a/8$	32,768	111,843
Mesh 3	$a/16$	262,144	839,619
Mesh 4	$a/32$	2,097,152	6,502,275

30 [358, 378]. In this study, 20 linear elements are assumed to be satisfactory. The time step size Δt of the analysis is obtained in the following way: The Fast Fourier Transform (FFT) of the excitation signal is performed from which the maximum frequency of interest $f_{\max}^{\text{FFT}}$ is obtained. An 85% area cutoff below the frequency distribution (the frequency that covers 85% of the cumulative area under the frequency curve) is utilised. The recommended time step size Δt that should be used for the analysis is computed as

$$\Delta t = \frac{1}{10 f_{\max}^{\text{FFT}}} \tag{5.21}$$

5.4.1 Prismatic beam under pure bending (Convergence study)

A prismatic beam with a rectangular cross section is considered as shown in Fig 5.1. It has the following dimensions: $a = 0.5$ m, $b = 0.5$ m, and $L = 8$ m. The origin of the Cartesian coordinates is placed at the centre of the beam's cross-section as shown.

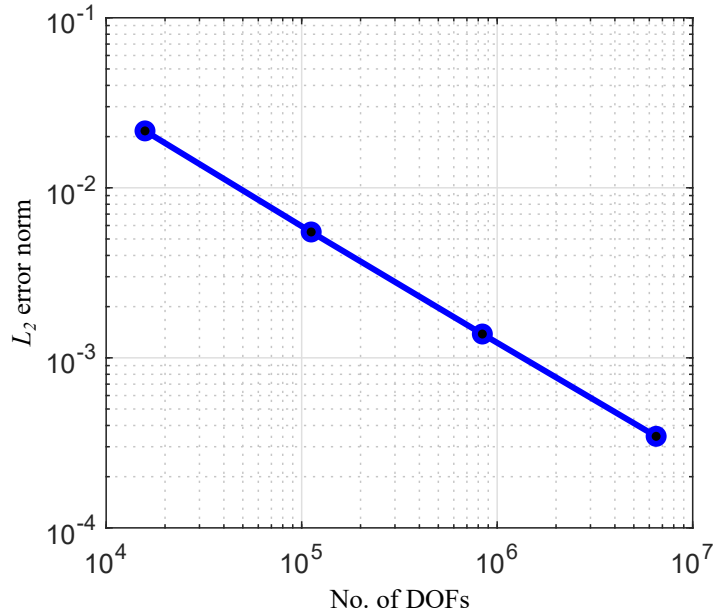


Figure 5.2: Convergence study of beam in pure bending

A linear-elastic material is assumed with Young's modulus $E = 1$ Pa and Poisson's ratio $\nu = 0.2$. For the pure bending case, two equal but opposite moments with a magnitude $M = 1$ N-m are applied at the two ends of the beam. This results in a uniform curvature of the beam $R = \frac{EI_x}{M}$, where $I_x = \frac{4ab^3}{3}$ is the second moment of area of the beam's cross-section about the x-axis. The analytical displacement solutions of this problem can be expressed as

$$u_x = \frac{\nu x y}{R}, \tag{5.22a}$$

$$u_y = \frac{1}{2R}\left(z^2 - \nu\left(x^2 - y^2\right)\right), \tag{5.22b}$$

$$u_z = -\frac{yz}{R}. \tag{5.22c}$$

The only non-zero analytical stress solution in this problem is given as

$$\sigma_z = -\frac{Ey}{R} \tag{5.23}$$

For this analysis, the analytical displacement boundary conditions are applied at the beam end at $z = 0$ according to Eq. (5.22). The analytical stress field given in Eq. (5.23) is applied on the other end ($z = L$). A static solution is sought with a tolerance of 1E-6

Figure 5.3: The geometry of a 3D concrete obtained from X-ray CT scans

using the developed PCG solver with the Jacobi preconditioner.

The geometry is meshed with uniform hexahedral elements with varying size, as given in Table 5.1. A convergence study is performed using the meshes, and the relative L_2 error norm of the displacement results is plotted against the number of degrees of freedom, as shown in Fig. 5.2. The rate of convergence of the displacement results is 0.687, which is near to the theoretical rate of convergence of 2/3.

5.4.2 Concrete under uniaxial static loading

A 3D concrete image with a size of $256 \times 256 \times 256$ voxels is considered, as shown in Fig. 5.3. The image is a part of the X-ray CT scans of a concrete block analysed by Huang et al. [379]. The black material represents the aggregates and the grey material represents the mortar. In the image, each voxel represents 0.08 mm $\times$ 0.08 mm $\times$ 0.08 mm. The elastic material properties of the aggregates are $E = 70$ GPa and $\nu = 0.2$, while the elastic material properties for the mortar are $E = 25$ GPa and $\nu = 0.2$.

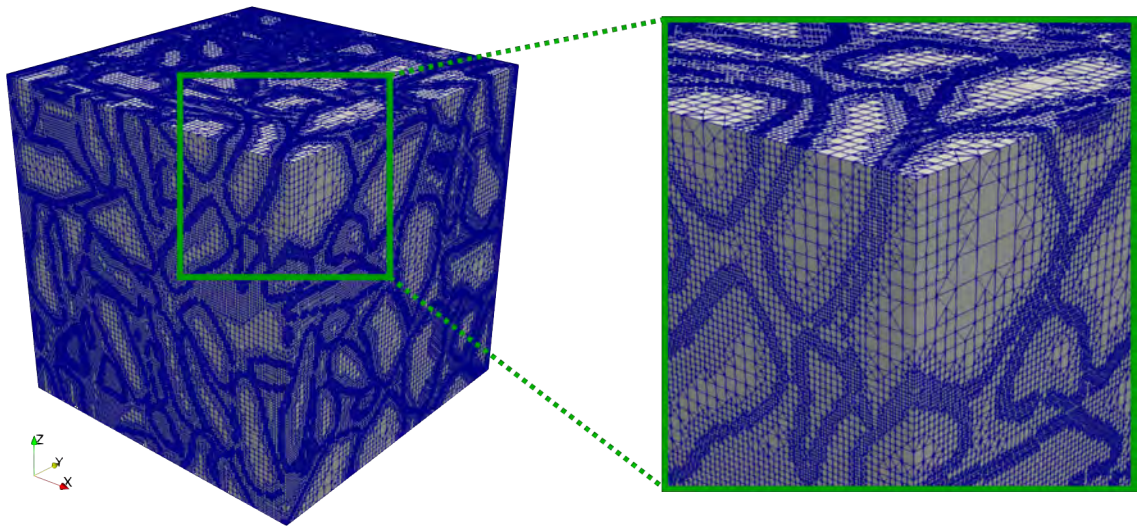


Figure 5.4: The octree mesh for concrete

The octree decomposition, where the maximum cell size is 8 voxels and the minimum cell size is 1 voxel, results in $3,048,578$ cells. The total number of cells generated is only around 18.2% of the total number of voxels in the image. The mesh has $5,854,499$ nodes, which means a total of $17,563,497$ DOFs. The octree mesh is shown in Fig. 5.4.

Stress analysis is conducted by fixing the base of the concrete sample and applying a uniform pressure load of 1 MPa at the top. Fixing the base reduces the effective number of unknowns (or DOFs) to be solved to 17,498,283. For this example, the PCG method is employed with Jacobi preconditioning and a residual tolerance of 1E-8. The relative L_2 norm of the relative residual achieved after $2,376$ iterations is 9.9E-9. The displacement magnitude of the concrete specimen is plotted in Fig. 5.5, which is in agreement with the applied loading and boundary conditions.

The computational time statistics for the strong scaling study of this example are given in Table 5.2 and plotted in the Fig. 5.6. The CPU time measurements and scaling calculations are explained in Section 4.4.3. From Fig. 5.6, it is observed that overall a good scaling behaviour is achieved using the octree-based parallel PCG solver. The

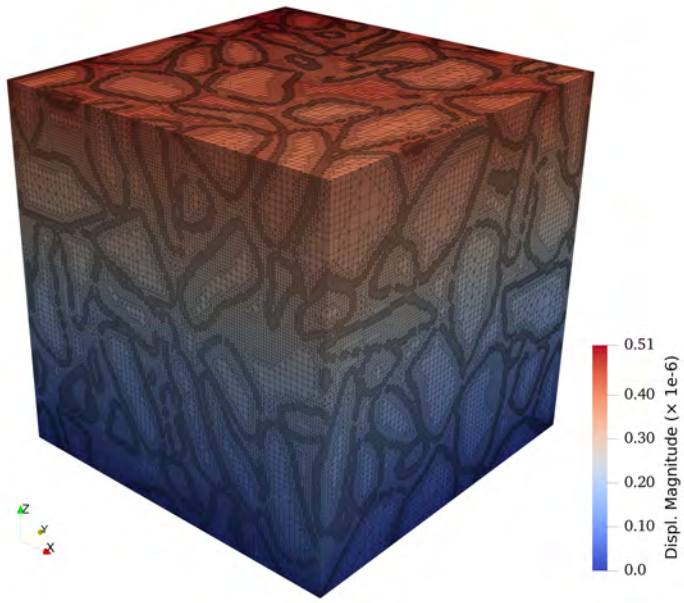


Figure 5.5: The octree mesh for concrete under uniaxial static loading with the displacement contour

Table 5.2: Strong scaling study for concrete under uniaxial static loading

N_{cores}	T_C^m	T_W^m	T_T	S_s^{rel}	E_s^{rel}	M_T
6	1,506.3	82.8	1,589.1	6.0	1.00	9.5
12	762.4	36.2	798.6	11.9	0.99	11.3
24	400.7	39.6	440.3	21.7	0.90	16.1
48	295.7	31.7	327.4	29.1	0.61	23.7
96	131.4	16.8	148.2	64.3	0.67	26.7
192	58.4	11.3	69.7	136.8	0.71	45.1

T_C^m: Mean calculation time [s], T_T: Total CPU time [s], M_T: Total Memory [GB]
T_W^m: Mean comm.+wait time [s], S_s^{rel}, E_s^{rel}: Relative speedup and efficiency (Strong scaling)

speed-up and efficiency curves follow similar trends as obtained for the explicit solver in Chapter 4. Within a single compute-node ($N_{cores} \leq 48$), a drop in the efficiency is observed as the number of cores increases. This behaviour can be mainly attributed to the memory-bandwidth limitation within the compute-nodes [83], which reduces the computational performance of the matrix operations. It is observed that the same is reflected in the calculation time (T_C^m) that scales poorly between $N_{cores} = 24$ and

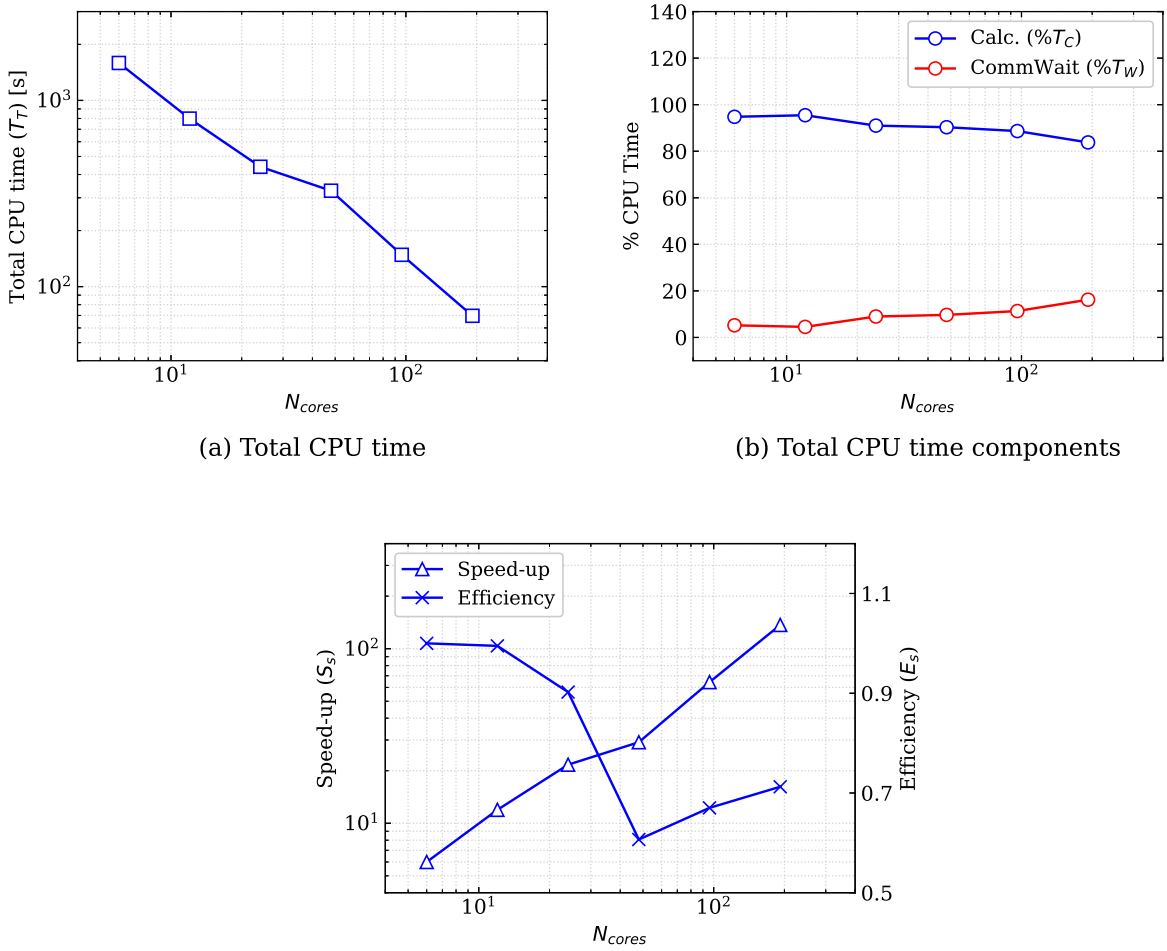


Figure 5.6: Computational time study (strong scaling) of the parallel solver for concrete under uniaxial static loading

48. Another reason is the increase in the percentage of communication and waiting time, though its effect on the efficiency is relatively small. On multiple compute-nodes ($N_{cores} > 48$), the efficiency increases steadily due to the *cache-effect* (explained in Section 4.5.6)..

5.4.3 A building surrounded by trench

The ground-borne vibrations arising from construction works, blasting activities, expanding road and rail networks, etc., pose a great challenge to engineers for building

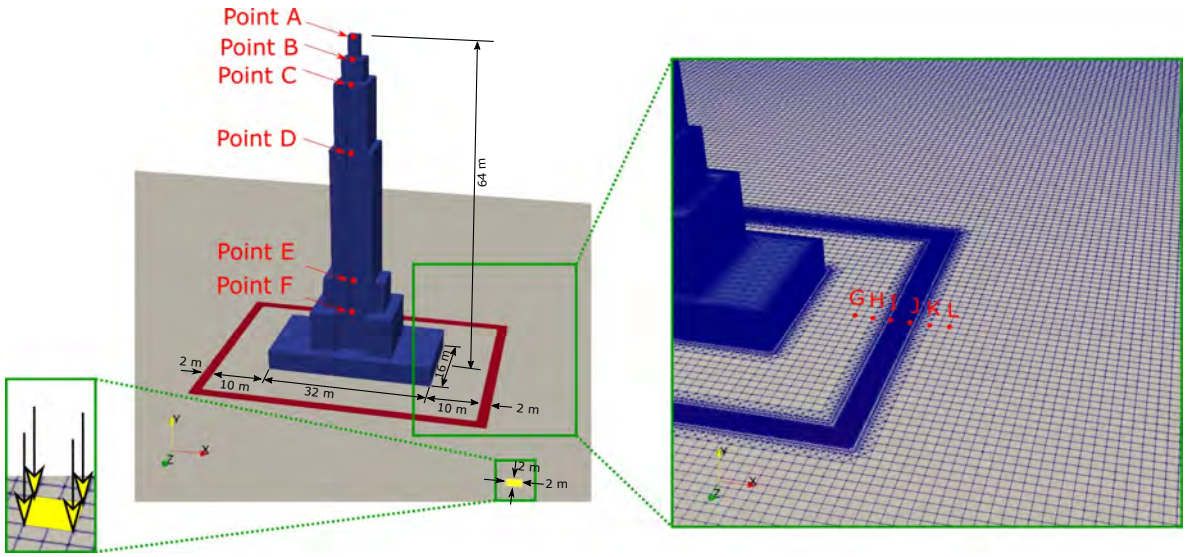


Figure 5.7: Geometry of for the building surrounded by trench

serviceable structures, especially for residential purposes. To deal with this problem, the wave-barriers constructed using open or filled trenches can provide a simple and cost-effective way of curbing the motion of travelling waves and achieving vibration isolation [372, 380]. The open trench acts like a finite discontinuity in the ground, and therefore, avoids the energy transmission across it. The infill trench causes the redistribution of wave energy in the form of reflected and transmitted waves. In addition, the trenches can be used for active or passive isolation. In active isolation, the trenches are built enclosing the source of vibration, while for passive isolation, they are built near the objects to be shielded.

A building model, as shown in Fig. 5.7, is considered. It is surrounded by a 2 m wide and 10 m deep infill trench, built 10 m away from the outer perimeter of the building for providing passive isolation from the incoming ground wave excitations. The dimension of the ground model is $400\,\mathrm{m} \times 200\,\mathrm{m} \times 400\,\mathrm{m}$. In the figure, blue represents the concrete building, grey is the ground (or soil), and red is the filled trench.

For this example, the elastic material properties are considered, as listed in Table 5.3. These include the Young's modulus E, Poisson's ratio ν and density ρ. The dilatational

Tab

Ma
Concrete
Soil outsid
Soil inside

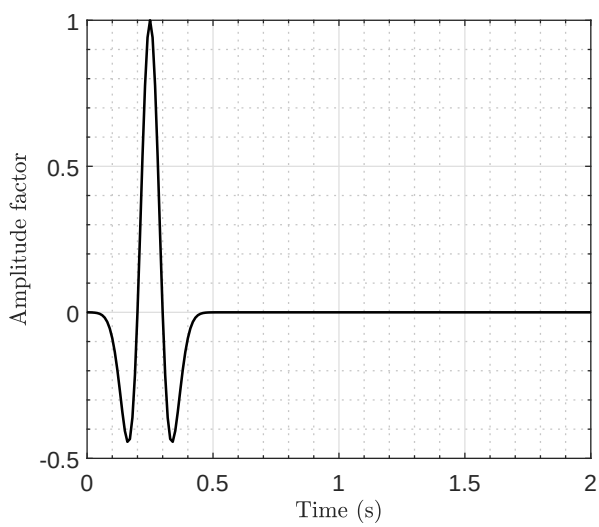
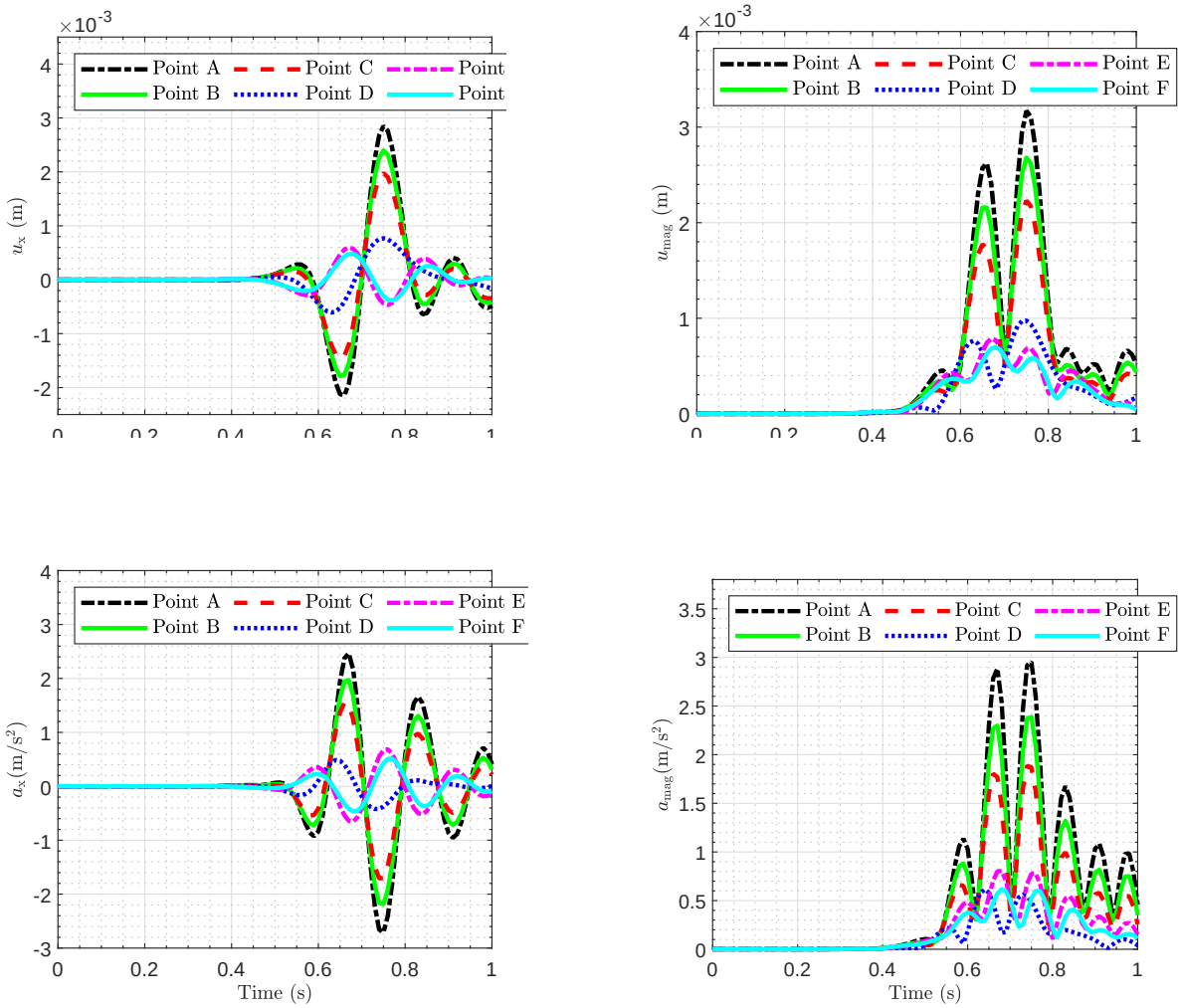


Figure 5.9: Ricker wavelet load for the building on a large soil block example



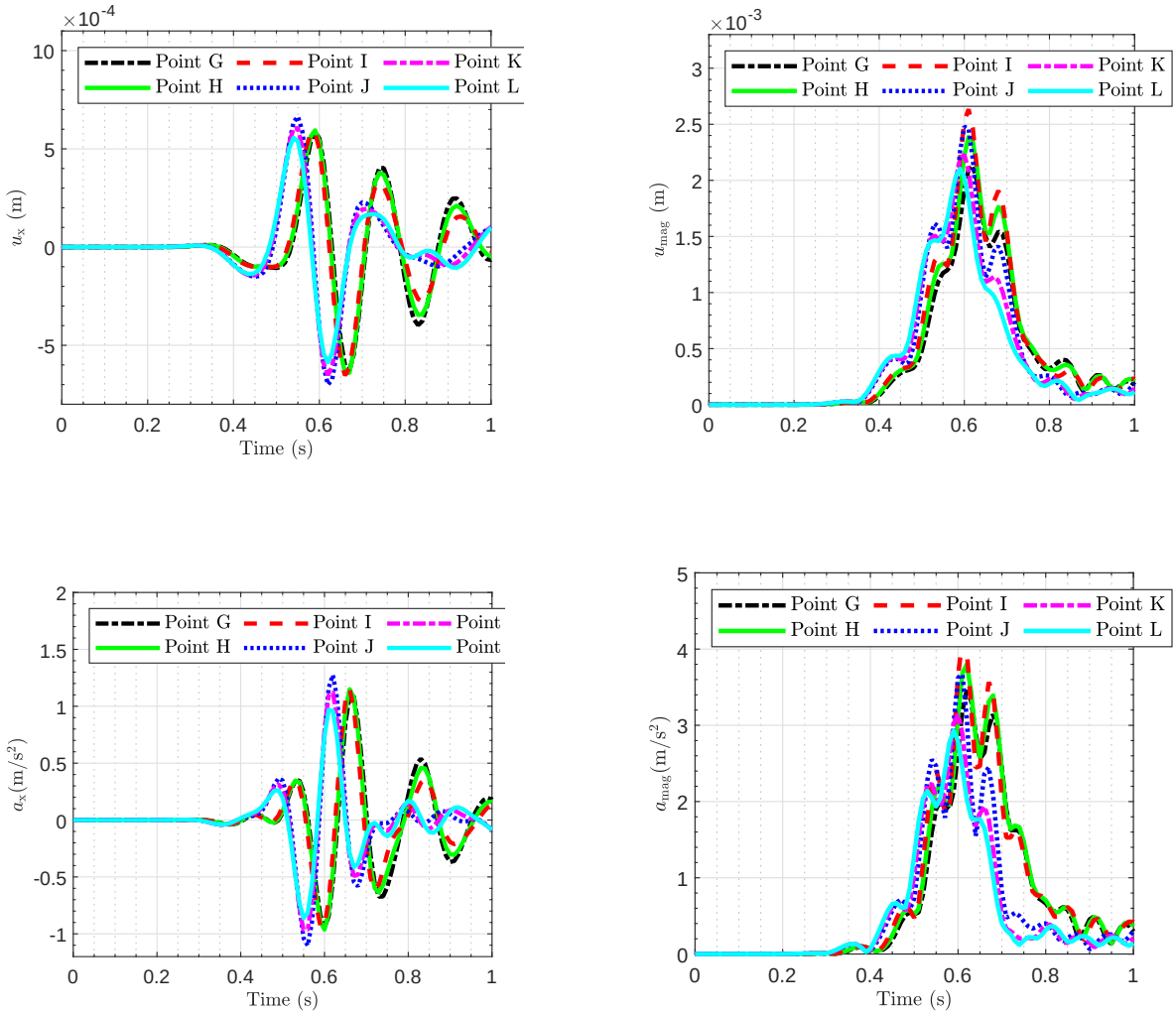
(c) a_{x} (d) a_{mag}

Figure 5.10: Results at points A to F for the building surrounded by trench

wave velocity v_{p}, shear wave velocity v_{s} and the recommended maximum element size s_{max} for each material based on 20 linear elements per wavelength guideline is also presented.

The model is discretised using the octree mesh with a maximum element size of 1 m (8^3 voxels) and a minimum element size of 0.125 m (1^3 voxel). It contains $34,398,360$ elements and $35,316,434$ nodes ($105,949,302$ DOF). The mesh is refined around the boundary of the building and the trench area which is shown in Fig. 5.8.

A pressure load is applied on a $2\,\mathrm{m} \times 2\,\mathrm{m}$ area at 50 m away in x-direction and



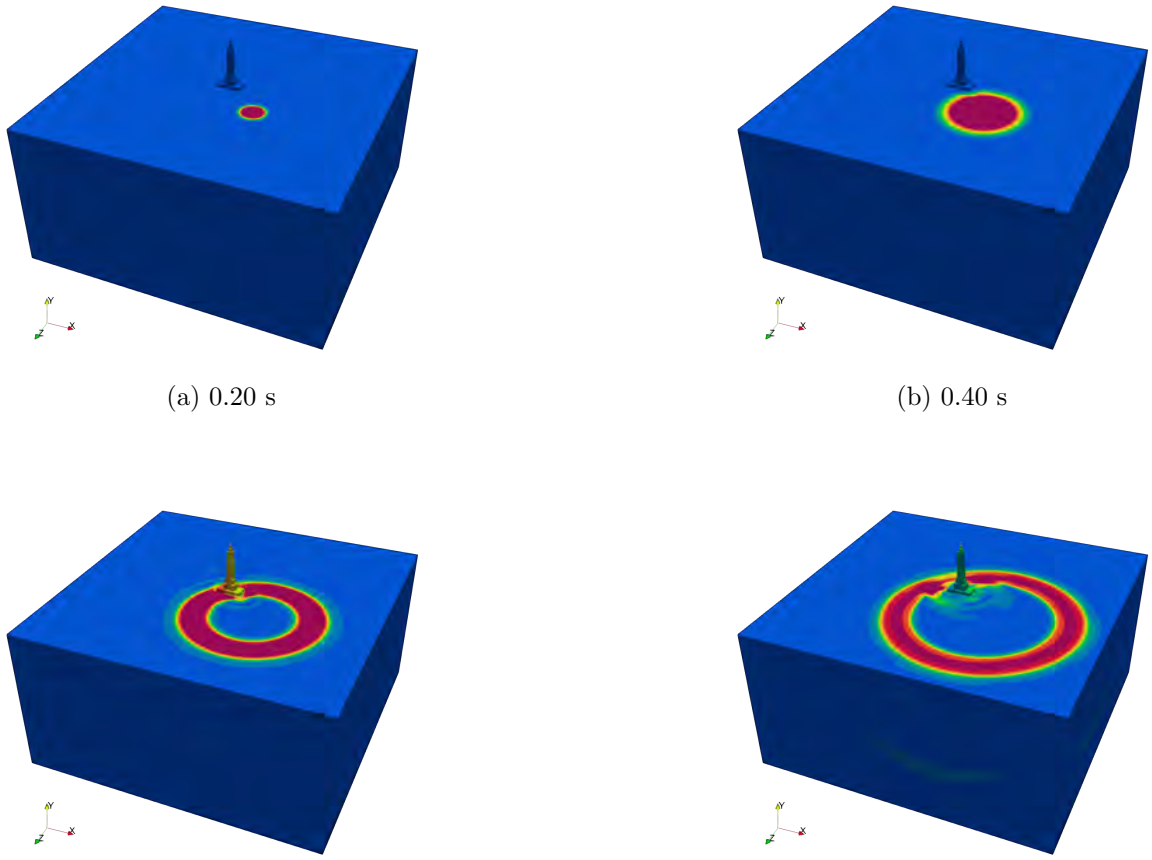
(c) a_{x} (d) a_{mag}

Figure 5.11: Results at points G to L for the building surrounded by trench

50 m away in z-direction from the centre of the base of the building (as depicted in Fig. 5.7). A Ricker wavelet pressure distribution shown in Fig. 5.9 is employed, which can be expressed as

$$\sigma\left(t\right)=\left(1-\left(\frac{t-t_c}{0.2t_c}\right)^2\right)\exp\left(-\frac{1}{2}\left(\frac{t-t_c}{0.2t_c}\right)^2\right)\sigma_0. \tag{5.24}$$

The load has a magnitude of $\sigma_0 = 10\,\mathrm{MPa}$ and $t_c = 0.25\,\mathrm{s}$ according to Eq. (5.24). The base of the ground is fixed and its lateral faces are constrained in the perpendicular direction to the surface.



(a) 0.20 s

(b) 0.40 s

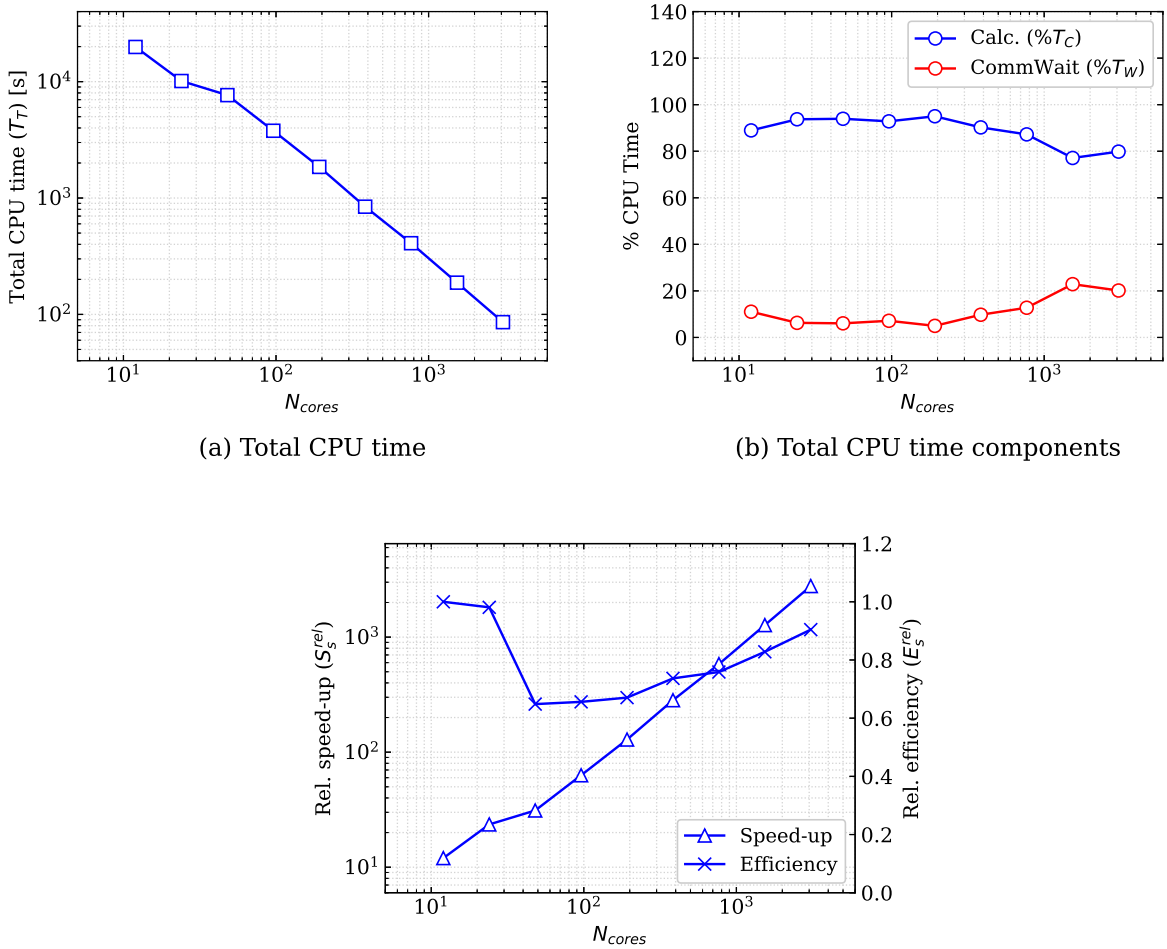
(c) 0.70 s

(d) 0.90 s

Figure 5.12: Displacement contour plots u_{mag} for the building surrounded by trench at the given time stamps

The maximum frequency of interest of the Ricker wavelet is calculated as $f_{\text{max}} = 8.8$ Hz. For the implicit HHT-α time integration, the recommended time step size (based on 10 time steps for each wave period) is $\Delta t = 1.1 \times 10^{-2}$ s. Therefore a time step size of $\Delta t = 0.01$ s is chosen here. The total simulation spans 100 time steps with a total time of 1 s. To solve the resultant system of linear algebraic equations, the PCG method is utilised with Jacobi preconditioning and an error tolerance of 1E-7. The total number of iteration across all the time steps is $88,309$ averaging at 883 per time step.

Several observation points are chosen for plotting of results, as marked in Fig. 5.7.



(a) Total CPU time

(b) Total CPU time components

(c) Strong scaling curves

Figure 5.13: Computational time study (strong scaling) of the parallel solver for the building surrounded by trench

Point A to point F are located at different elevations on the building, while Point G through point L are located on a straight line across the trench. The coordinates are given as: A$(0, 64, 1)$, B$(0, 60, 2)$, C$(0, 56, 3)$, D$(0, 44, 3)$, E$(0, 18, 3)$, F$(0, 12, 4)$, G$(22, 0, 0)$, H$(24, 0, 0)$, I$(26, 0, 0)$, J$(28, 0, 0)$, K$(30, 0, 0)$, and L$(32, 0, 0)$ (unit: m).

The displacement and acceleration results for points A to F are plotted in the Fig. 5.10 and for points G to L are given in the Fig. 5.11. Here, u_x and a_x represent the displacement and acceleration along the x-direction, and u_{mag} and a_{mag} represent their total magnitude. Near the trench, Point L is the closest to the excitation source while

Table 5.4: Strong scaling study for the building surrounded by trench

N_{cores}	T_C^m	T_W^m	T_T	S_s^{rel}	E_s^{rel}
12	17,667.9	2,195.2	19,863.1	12 .0	1.00
24	9,489.5	634.9	10,124.4	23.5	0.98
48	7,195.4	463.9	7,659.3	31.1	0.65
96	3,513.8	270.8	3,784.6	63.0	0.66
192	1,759.1	92.1	1,851.2	128.8	0.67
384	760.3	82.2	842.5	282.9	0.74
768	356.9	52.2	409.1	582.6	0.76
1536	144.5	42.9	187.4	1,271.9	0.83
3072	68.5	17.3	85.8	2,778.1	0.90

T_C^m: Mean calculation time [s], T_T: Total CPU time [s], T_W^m: Mean comm.+wait time [s], S_s^{rel}, E_s^{rel}: Relative speedup and efficiency (Strong scaling)

Point G is furthest away. This corresponds to the result, where approximately around 0.2 s, Point L is excited first while Point G is excited last.

The displacement plots are given at selected time snapshots in Fig. 5.12. It can be seen that initially, the excited wave is radiating uniformly in all the direction around the source area. As it passes through the trench, an apparent diffraction can be observed clearly.

The computational time statistics for the strong scaling study are given in Table 5.4 and plotted in the Fig. 5.13. It is observed that the speed-up and efficiency curves are nearly consistent with those obtained with the PCG solver. A drop in the efficiency is observed for higher number of cores within a single compute-node (e.g., at $N_{cores} = 48$), which can be attributed to the memory-bandwidth limitation on each compute-node [83]. On the other hand, for $N_{cores} > 48$, the efficiency increases steadily due to the *cache-effect* (as explained in Section 4.5.6).

The memory usage in the implicit dynamic analysis of the building surrounded by trench is shown in Fig. 5.14 and tabulated in Table 5.5. As explained in the previous chapter, the total memory usage M_T has two components: The system memory usage

Table 5.5: Memory usage in the analysis of the building surrounded by trench

N_{cores}	M_S	M_D	M_T
12	3.8	79.4	83.2
24	7.7	75.0	82.7
48	15.0	81.2	96.2
96	18.1	81.3	99.4
192	36.3	82.2	118.5
384	71.3	90.5	161.8
768	142.8	93.6	236.4
1536	295.5	91.8	387.3
3072	595.3	86.1	681.4

M_S: System memory [GB], M_D: Model data memory [GB]
M_T: Total memory [GB]

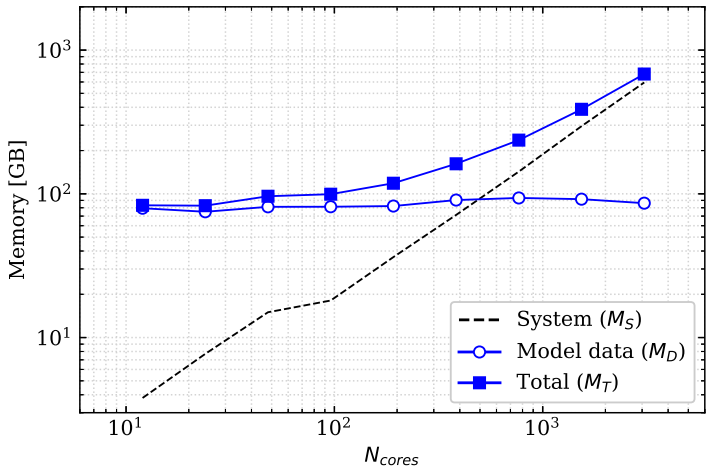


Figure 5.14: Memory usage in the analysis of the building surrounded by trench

M_S and the model data memory usage M_D.

The system memory usage is somewhat different from those obtained in Section 4.5.6 as a different version of Open-MPI (4.0.5) has been utilised from this chapter onwards. Nonetheless, the system memory usage is nearly proportional to the number of cores, which agrees well with those obtained in previous chapter. In addition, it is observed that the model data memory usage increases marginally with increasing number of cores. This can be attributed to the addition of interfacial DOFs due to mesh partitioning,

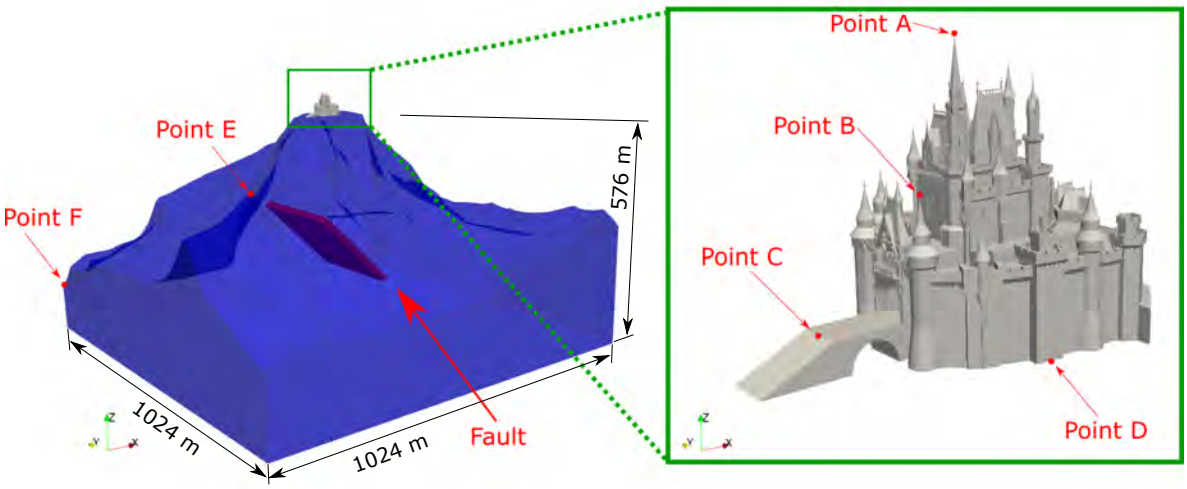


Figure 5.15: Geometry for the castle on a mountain example

the requirement of copying the octree-pattern data to each process, etc.

5.4.4 Seismic response of a castle on a mountain

The seismic response of a structure significantly depends on the spectral and peak accelerations features of the ground motion. In turn, the ground motion at a particular site could be remarkably affected due to the varying geological conditions along the seismic wave propagation path (e.g., presence of a fault) and the local topographical conditions near the structure. For performing high-fidelity structural dynamics simulation, modelling the site-specific ground motion is critical. However, efficiently performing the combined simulation – the large-scale seismic-wave propagation and fine-scale dynamic response of the structure in one model – can be very challenging [381]. This is typically due to the varying element size requirements to simulate these two phenomena accurately at very different scales. In this example, the capability of the octree-based approach is examined in modelling such multi-scale problem.

A castle with a maximum height of 64 m is modelled on a mountain. The mountain spans a region of $1024\,\mathrm{m} \times 1024\,\mathrm{m} \times 576\,\mathrm{m}$. Within the mountain, there exists a blocky

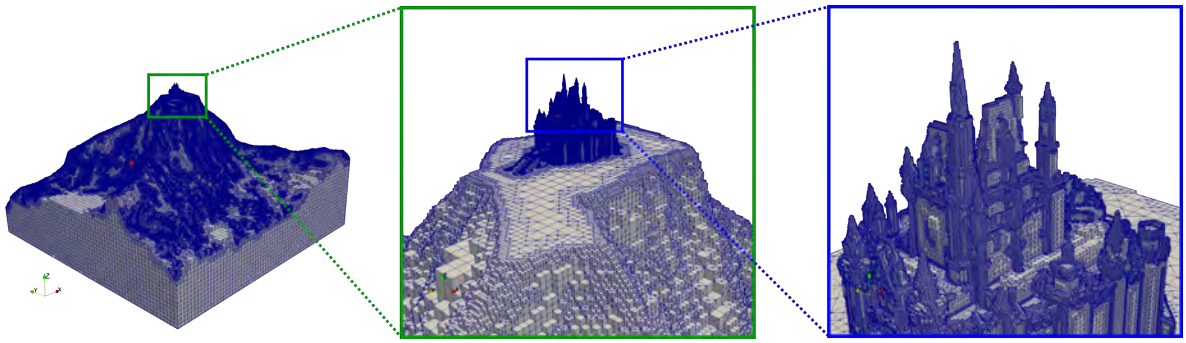


Figure 5.16: Overall mesh of the castle on a mountain example

Table 5.6: Material properties for the castle on a mountain example

Material	E (GPa)	ν	$\rho\,(\mathrm{kg/m^3})$	v_p (m/s)	v_s (m/s)	s_max (m)
Limestone for castle	30	0.3	2,700	3,867	2,067	21.1
Rock in the mountain	60	0.3	2,400	5,801	3,101	31.7
Weak fault layer	3	0.3	2,400	1,297	693	7.1

regional weak fault with a size of $256\,\mathrm{m} \times 256\,\mathrm{m} \times 16\,\mathrm{m}$ oriented 45° degree in the $xz-$plane. The geometry is shown in Fig. 5.15. Grey colour represents the castle, blue colour represents the mountain's rocky soil and red colour indicates the fault. The mountain part of the geometry is made slightly transparent to reveal the fault contained in it. The Cartesian-coordinate's origin is placed at the bottom centre of the model. Based on this origin, the centroid of the fault is located at $(0, 0, 256)$ m. The elastic material properties for this example are listed in Table 5.6.

The octree mesh is utilised to discretise the entire problem domain. The maximum element size is 16 m and the minimum element size of 0.0625 m. It contains $638,707$ elements and $1,353,006$ nodes ($4,059,018$ DOF). The mesh is refined around the mountain's boundary that has a high curvature. Finer refinement is applied around the castle features to better capture the geometry of the castle. This can be observed in the mesh shown in Fig. 5.16. Additionally, the model is sliced to to reveal the inner mesh, as observed in the Fig. 5.17. Here, only the elements in the positive $y-$axis are shown. It can be seen that the area near the fault is refined to better represent the fault geometry.

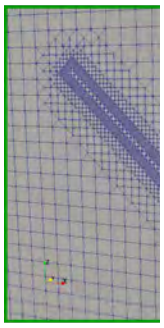


Figure 5.1

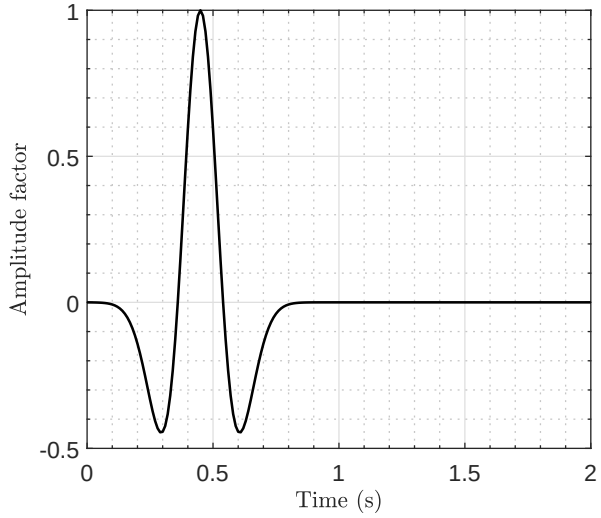
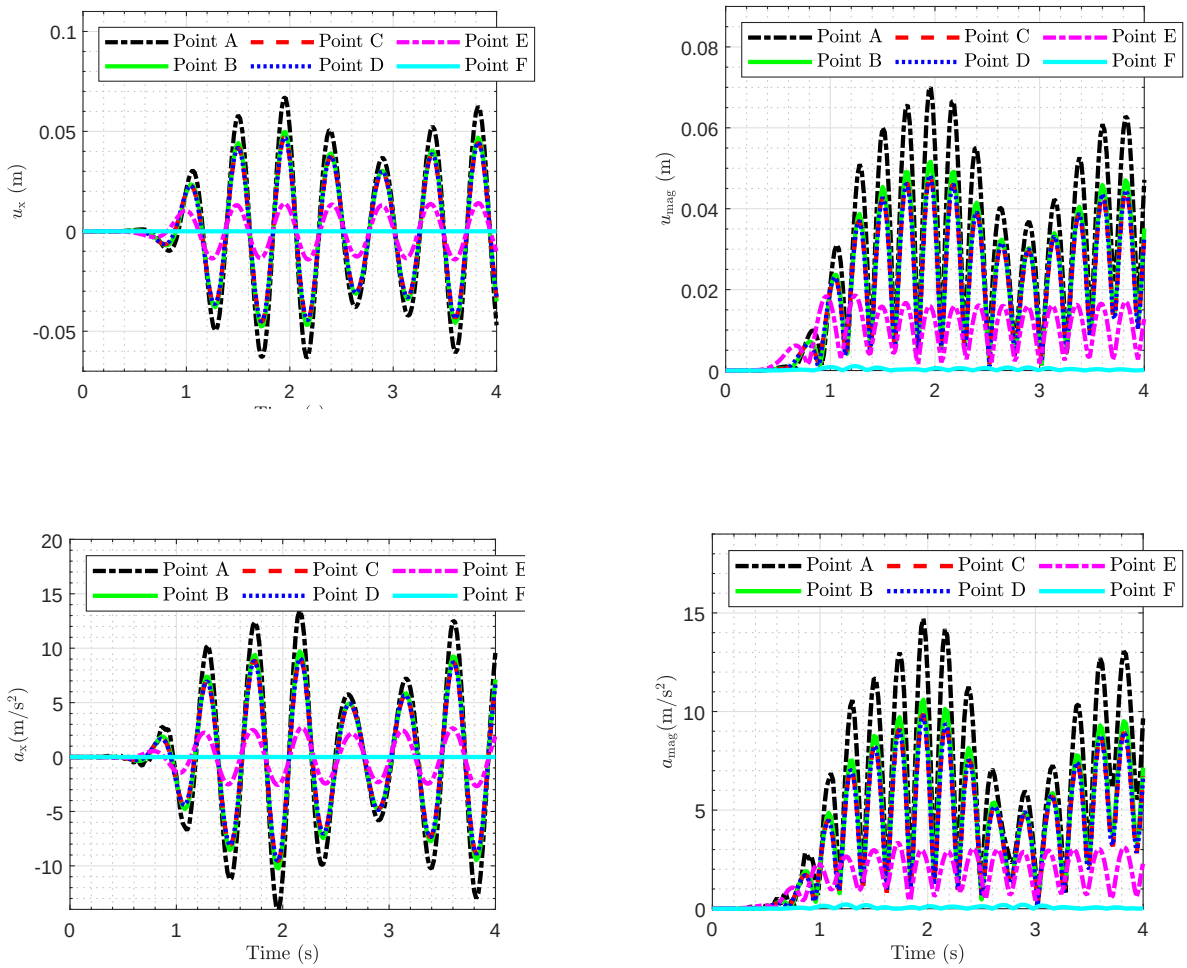


Figure 5.18: Ricker wavelet load for the castle on a mountain example



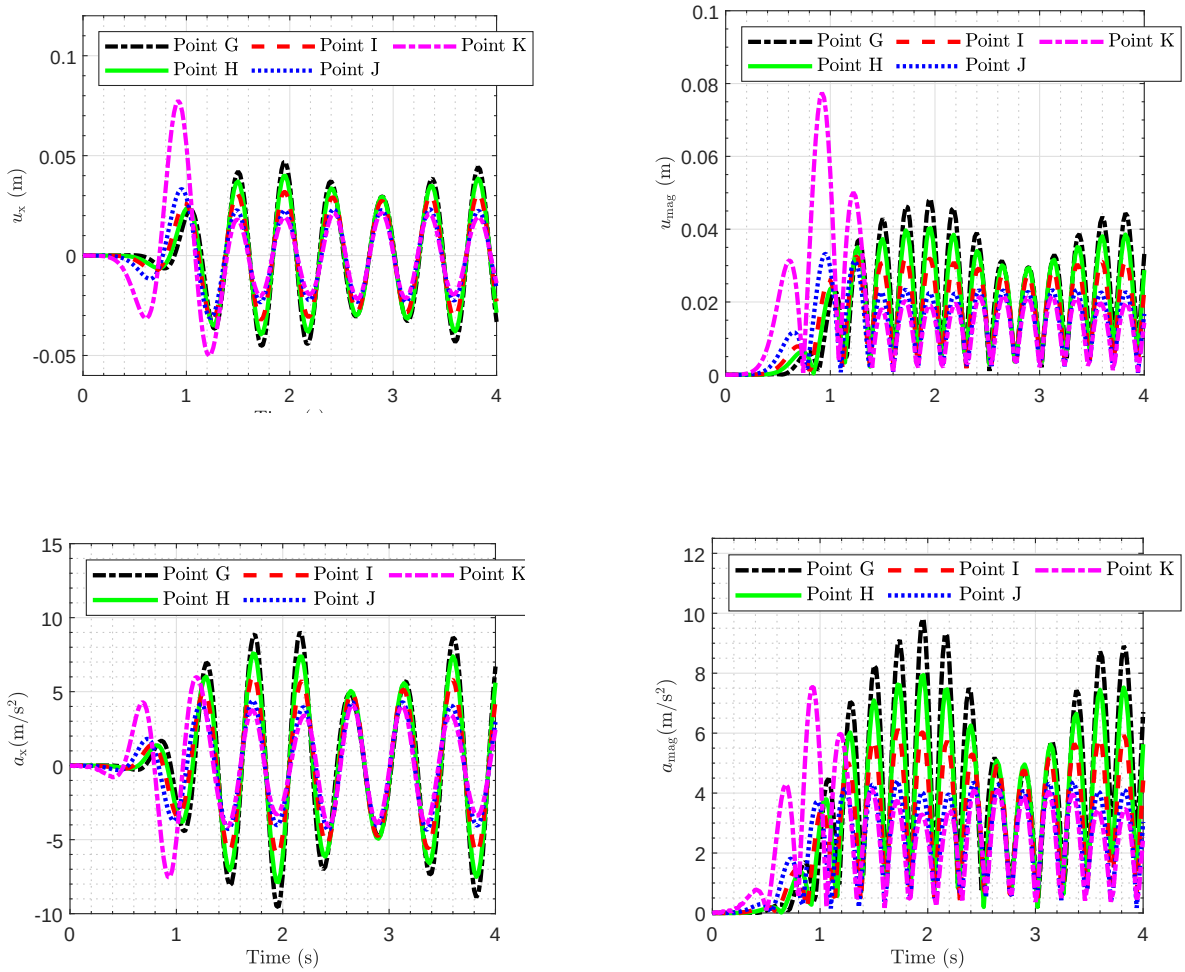
(c) a_x (d) a_{mag}

Figure 5.19: Results at points A to F for the castle on a mountain example

Further, it is worth mentioning that the entire mesh is automatically generated from a STL model and without much human intervention.

A Ricker wavelet pressure load (as shown in Fig. 5.18) in the $x-$direction is applied with a magnitude of $\sigma_0 = 1\,\text{GPa}$ and $t_c = 0.45\,\text{s}$ (according to Eq. 5.24) on a $16\,\text{m} \times 16\,\text{m}$ area at the middle of the base cross-section, as shown in Fig. 5.17. The base and the lateral faces of the soil are constrained in the perpendicular direction to the surfaces.

The maximum frequency of interest of the Ricker wavelet is calculated as $f_{\max} = 4.9$ Hz. For the implicit HHT-α time integration, the recommended time step size is calculated as $\Delta t = 2 \times 10^{-2}$ s, which is chosen here. The total simulation spans 200



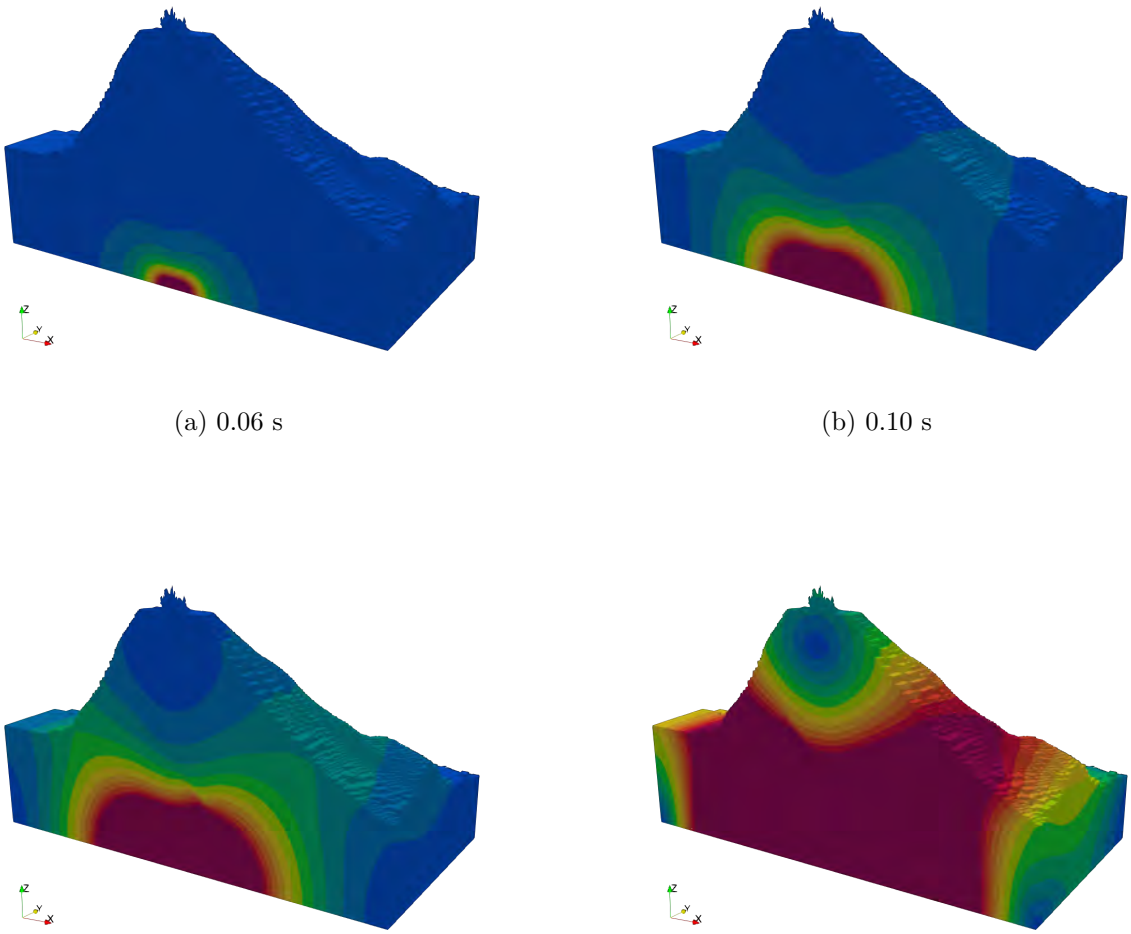
(c) a_x (d) a_{mag}

Figure 5.20: Results at points G to K for the castle on a mountain example

time steps with a total time of 4 s. An error tolerance of 1E-7 is chosen for the PCG method employed in the solver. The total number of iteration across the 200 time steps is $305,924$, averaging at $1,530$ iteration per time step.

The observation points in the mesh are set up to compare the displacement, velocity and acceleration results. Point A to point D (as shown in Fig. 5.15) are located within the castle's boundary, point E and F (as shown in Fig. 5.15) are located on the mountain's boundary and the Point G through point K (as shown in Fig. 5.17) are within the mountainous region along a straight line with different elevations.

The coordinates of these points are given by: A$(-1.5, -7,638)$, B$(-12.5, -9,609)$,



(a) 0.06 s

(b) 0.10 s

(c) 0.12 s

(d) 0.16 s

Figure 5.21: Displacement contour plots u_{mag} for the castle on a mountain example at time snapshot

C$(-30, 0, 584)$, D$(8, -24, 576)$, E$(-332, -188, 452)$, F$(-512, 512, 118)$, G$(0, 0, 576)$, H$(0, 0, 448)$, I$(0, 0, 320)$, J$(0, 0, 192)$, and K $(0, 0, 64)$ (unit: m).

The results for points A to F are plotted in the Fig. 5.19. Here, u_x and a_x represent the displacement and acceleration along the x-direction, and u_{mag} and a_{mag} represent their total magnitude. It can be seen that the magnitude of displacements are generally higher for the points on the castle as it has slender features and lower stiffness. The $x-$direction results at Point F is zero as it is located at the mountain's boundary that

are fixed in $x-$ and $y-$directions.

The results for the points G to K are plotted in the Fig. 5.20. The point K is excited first as the source is located at the base, while point G at the base of the castle is excited last. After some time, it can be seen that the points at the higher elevation displaced more as the mountain narrows towards the peak where the castle is sitting on.

The displacement plots for half of the mesh (similar to the one shown in Fig. 5.17) are provided at selected time snapshots in Fig. 5.21. As the wave travels from the source to the rest of the region, it can be seen that the wave is diffracted as it arrives at the vicinity of the fault. This is depicted by the contour plots, especially at around the time 0.12 s in Fig. 5.21c.

5.5 Summary

In the present chapter, the octree-based approach is extended to develop a massively parallel implicit-solver HHT-α for solving large-scale elastodynamic problems. The solution of the system of linear equations, which constitutes the dominant portion of the CPU time of the implicit solvers, is obtained using the PCG solver. A limited number of octree patterns are utilised to discretise the problem domain, which reduces the overall memory requirement. The work associated with the PCG solver is performed efficiently using the octree-based pattern-by-pattern (PBP) approach that allows carrying out dense matrix computations leading to much better cache utilisation. The parallelisation is carried out using the mesh-partitioning strategy. The problem domain is divided into a set of smaller partitions, and the workers carry out the simultaneous local computations over the assigned partitions. A message-passing algorithm is implemented using the MPI4Py library to synchronise the data among the neighbouring mesh partitions.

The implementation of the developed solvers is first verified using the static loading

example of a prismatic beam under pure-bending, where optimal convergence rates are obtained. The computational performance and scalability of the PCG solver are shown by carrying out a parallel simulation of a concrete block under uniaxial static loading.

Two numerical examples with dynamic loading are presented, which are solved using HHT-α time integration method. The example of a building surrounded by a trench is solved, which shows the efficient scaling of the parallel HHT-α time integration for the large-scale structural dynamic problem. Then, a castle on a mountain example is presented to demonstrate the ability of the octree mesh in performing multiscale dynamic analysis. Here, the ratio of maximum to minimum cell size was 256, with the maximum element size 16 m for the mountain and the minimum element size 0.0625 m for the castle. The large-scale seismic wave simulation in the mountain (considering the effect of the fault) and small-scale dynamic analysis of the structure (considering the complex shape of the structure) were performed simultaneously.

Chapter 6

Compressive fracture analysis in two dimensions

This chapter[1] develops a cohesive/contact interface formulation for performing 2D compressive fracture analysis in the framework of the SBFEM. The organisation of the chapter is as follows.

Section 6.1 introduces to the compressive fracture modelling based on the cohesive and contact interface formulations. In Section 6.2, a sideface traction formulation is developed for the SBFEM based cracked elements. In Section 6.3, the implementation of the cohesive and contact formulations are shown using the interface-elements and sideface tractions. Section 6.4 presents the numerical solution scheme using the local-arc length procedure. Section 6.5 shows the complete solution procedure for solving the compressive fracture analysis. In Section 6.6, a few numerical results are presented to verify the developed formulation and solve the engineering problems. The solver implementation procedure and results are summarised in Section 6.7.

[1]The contents of this chapter were published in [382]. I acknowledge the valuable suggestions received from Prof Chongmin Song, Dr Sascha Eisenträger, and Dr Eantat Ooi.

6.1 Introduction

The study of crack and fracture under compression has attracted much attention from researchers in the past decades. It has a wide range of applications, including the response of concrete under compression, the delamination of composite panels, the geomechanical study of the rocks with fractures and joints, etc. These problems are often associated with non-linear structural behaviour due to interfacial debonding (e.g. at the aggregate-matrix interface in concrete), crack initiation and propagation into the matrix, presence of cohesive tractions over the crack surface (e.g. due to aggregate/fibre interlocking in concrete and composites) and the frictional-contact tractions over the crack surfaces due to crack closure.

Several models exist to capture the crack induced non-linearity [3, 18, 185, 186, 196, 241, 383–386]. A popularly used technique is cohesive crack modelling (CCM), where the cracks are modelled by introducing the discontinuities within the mesh (discrete crack representation), and the crack-surface tractions are captured by inserting the cohesive interface elements (CIEs) within the discontinuities. This approach also enables to capture the behaviour of cracks under compressive stress fields (or the crack closure) because the information related to crack surface geometry is readily available, and thus, the contact-constraints are satisfied with greater accuracy. A detailed review of cohesive crack and contact modelling and their numerical implementations is covered in Sections 2.3.2 - 2.3.4.

In the recent years, the scaled boundary finite element method (SBFEM) [13, 14] has gained much popularity among the researchers because of its ability to handle the complexities associated with the cracks, crack propagation and contact analyses [4, 73, 77, 123, 125, 188, 194, 344, 345]. The SBFEM formulation enables direct extraction of stress intensity factors (SIFs) from the formulation, a simple mesh mapping procedure, and the ability to handle hanging nodes that greatly simplifies the remeshing problem during crack propagation. The method can be easily coupled with other

numerical techniques such as FEM. This salient feature was utilised by [188] in modelling a fully automatic cohesive crack growth procedure in quasi-brittle materials. To simulate cohesive tractions in the fracture process zone (FPZ), zero-thickness interface elements [240, 386] were inserted along the crack. A shadow domain procedure was utilised, which required mesh-mapping to transfer displacements on to the cracked element for SIF calculation at the end of each load step. Since its inception, the shadow-domain procedure has been popularly utilised in several works [4, 123, 344] to model cohesive crack growth in the framework of SBFEM.

For the problems related to the contact analysis, SBFEM based formulations can efficiently handle the mismatching mesh surfaces using the node-insertion at the contact-interfaces [126, 128]. However, literature related to the application of SBFEM on compressive cracks is rare. Recently, a crack-contact formulation was developed using a segment-to-segment approach for the cracked element [127]. The concept was further extended to solve wing crack propagation; however, the application was limited to brittle solids only [387].

The present work develops a cohesive interface modelling formulation for solving compressive fracture problems in the framework of SBFEM. The crack surface tractions are implemented by inserting the 4-noded zero-thickness interface elements [240, 386] along the crack. In order to handle the stresses near the crack-tip, the crack surface tractions are linearised and applied on the sidefaces of the cracked elements as traction boundary conditions. The non-linear analysis is carried out using a local arc length procedure, where an updated normal plane constraint equation [388] is adopted. A novel scheme is proposed, which utilises SIFs at each iteration within the load step to update the linearised sideface-tractions applied on cracked elements. The iteration is carried out until the equilibrium position is achieved and the crack-tip singularities vanish, allowing the stress fields at the crack-tips to be accurately captured without crack-tip mesh refinement. The present method eliminates the mesh-mapping requirements for

SIF calculation at the end of each load step, unlike the shadow domain procedure. In addition, the stresses in the crack-tip region are directly extracted from the SBFEM formulation considering the effect of crack-surface tractions. Therefore, the stress-based criterion can be efficiently used for calculating the crack propagation condition and direction. The efficacy of the method is shown through practical applications involving interfacial debonding and matrix cracking in Brazilian split tests and wing crack propagation in concrete.

6.2 The SBFE formulation for a cracked s-element in 2D

The SBFEM for two-dimensional elastostatics is discussed in Section 3.1. Here, we derive the stress intensity factors (SIF) calculation in a cracked element (also called the cracked s-element). We also develop the formulation for sideface tractions in the cracked element to model the cohesive and contact tractions near the crack-tip region.

6.2.1 The stress intensity factor calculation

In the framework of SBFEM, the crack-tip region can be modelled using a cracked element, as shown in Fig. 6.1. Here, no excessive discretisation or presence of any singular elements are required in the vicinity of the crack-tip. In a cracked element, the scaling centre is placed at the crack-tip, which allows accurate modelling of the square root singularity of the crack-tip and enables direct extraction of the stress intensity factors (SIFs) from the scaled boundary formulation. Apart from the scaling centre requirement, a cracked element can be treated identically to the other elements without crack. The sidefaces connecting the boundary nodes to the crack-tip are not discretised as they are defined through scaling in the radial direction.

In order to extract the stress intensity factors from a cracked element, we rewrite

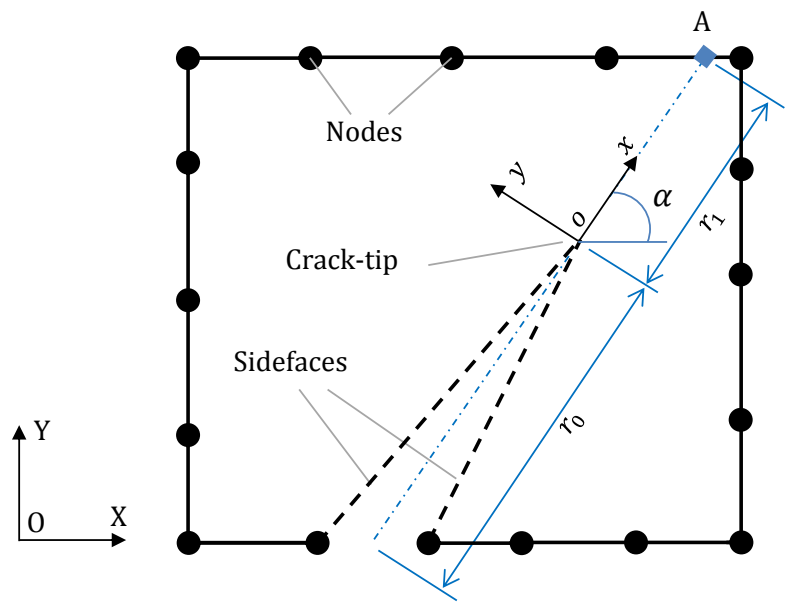


Figure 6.1: A cracked s-element in 2D

Eq. (3.81) as

$$\boldsymbol{\sigma}(\xi,\eta) = \mathbf{D}\sum_{i=1}^{n} c_i \xi^{\lambda_i - 1}\boldsymbol{\Psi}_i(\eta), \tag{6.1}$$

where, $\boldsymbol{\Psi}_i(\eta)$ is the stress mode that depends on the circumferential coordinate η. It is expressed as

$$\boldsymbol{\Psi}_i(\eta) = \left\{ \begin{array}{c} \psi_{xx}(\eta) \\ \psi_{yy}(\eta) \\ \psi_{xy}(\eta) \end{array} \right\} = \mathbf{D}(\lambda_i \mathbf{B}_1(\eta) + \mathbf{B}_2(\eta))\boldsymbol{\phi}_i^u. \tag{6.2}$$

The crack surfaces are inclined at angle α to the horizontal and have local axes xoy with the crack-tip at point o, as shown in Fig. 6.1. It was shown by Song and Wolf [118] that when the scaling centre is placed at the crack-tip, the two SIFs can be directly obtained from the stress solution in Eq. (6.1) and are expressed as

$$\left\{ \begin{array}{c} K_I \\ K_{II} \end{array} \right\} = \sqrt{2\pi r_1} \sum_{i=I,II} c_i \left\{ \begin{array}{c} \psi_{yy}(\eta = \eta_A) \\ \psi_{xy}(\eta = \eta_A) \end{array} \right\}, \tag{6.3}$$

where, r_1 is the distance between the crack tip and point A on the boundary of the

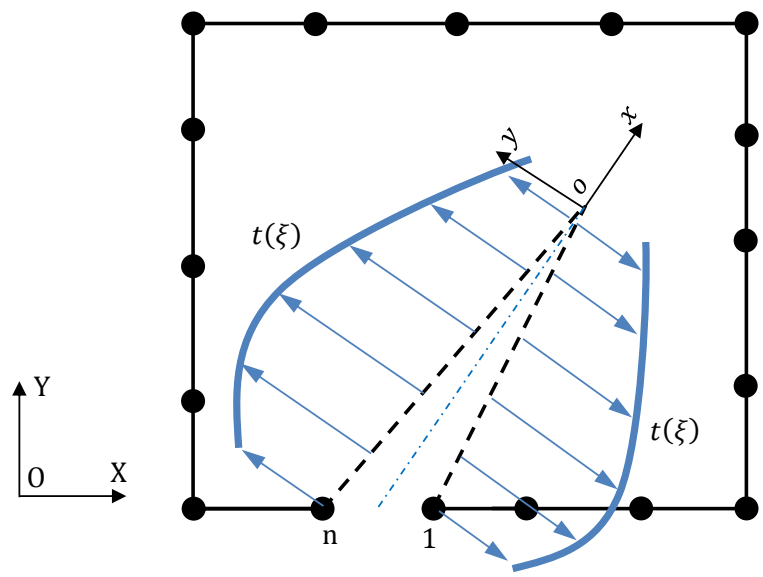


Figure 6.2: A cracked s-element in 2D with sideface tractions

cracked element, and $i = I, II$ represents the two modes with $\lambda_i = 0.5$ in Eq. (6.1) that corresponds to the singular values of stresses when $\xi \to 0$.

6.2.2 The sideface traction formulation

We now consider a cracked element with equal and opposite traction $t(\xi)$ acting on the sidefaces, as shown in Fig. 6.2. The sideface traction $t(\xi)$ has two components: $t_s(\xi)$ and $t_n(\xi)$ in the shear $(o - x)$ and normal $(o - y)$ directions respectively. These can be approximated as a summation of power functions of radial coordinate ξ as

$$t_s(\xi) = \sum_{k=0}^{N_t} \hat{f}_{sk} \xi^k, \tag{6.4a}$$

$$t_n(\xi) = \sum_{k=0}^{N_t} \hat{f}_{nk} \xi^k, \tag{6.4b}$$

where, $\hat{f}_{sk}$ and $\hat{f}_{nk}$ are the coefficients of power function in shear and normal directions of the sidefaces respectively and N_t is the number of terms in the power function summation. The equivalent nodal sideface force function $\mathbf{f_t}(\xi)$ for the s-element in

the global coordinate system is given as

$$\mathbf{f_t}(\xi) = (t_s(\xi)\mathbf{r}_s + t_n(\xi)\mathbf{r}_n)A = \sum_{k=0}^{N_t} \xi^k(\hat{f}_{sk}\mathbf{r}_s + \hat{f}_{nk}\mathbf{r}_n)A = \sum_{k=0}^{N_t} \xi^k \mathbf{f}_k. \tag{6.5}$$

Here, A is the surface area of the sideface given by $A = t_h r_0$, where r_0 is the length of the sideface (Ref: Fig. 6.1) and t_h is the thickness. The sideface traction rotation vectors (which depend on crack angle α) are given by

$$\mathbf{r}_s = \begin{bmatrix} -\cos(\alpha) & -\sin(\alpha) & 0 & .... & 0 & \cos(\alpha) & \sin(\alpha) \end{bmatrix}^T, \tag{6.6}$$

$$\mathbf{r}_n = \begin{bmatrix} \sin(\alpha) & -\cos(\alpha) & 0 & .... & 0 & -\sin(\alpha) & \cos(\alpha) \end{bmatrix}^T. \tag{6.7}$$

The additional virtual work done by the tractions acting on the sidefaces of the s-element is given by

$$\delta U_s = \int_0^1 \delta \mathbf{u}^{\mathrm{T}}(\xi)\mathbf{f_t}(\xi)\mathrm{d}\xi, \tag{6.8}$$

Augmenting Eq. (3.38) with the virtual work by sideface tractions, we get

$$\delta U_\varepsilon + \delta U_s - \delta W = 0. \tag{6.9}$$

As derived by Deeks and Wolf [120], the scaled boundary finite element equation considering the sideface tractions can be obtained from Eq. (6.9) as

$$\mathbf{E}_0\xi^2\mathbf{u}(\xi)_{,\xi\xi} + (\mathbf{E}_0 + \mathbf{E}_1^{\mathrm{T}} - \mathbf{E}_1)\xi\mathbf{u}(\xi)_{,\xi} - \mathbf{E}_2\mathbf{u}(\xi) + \xi\mathbf{f_t}(\xi) = 0. \tag{6.10}$$

A general solution to this non-homogeneous ODE can be given as a linear combination of the solution of the homogeneous ODE Eq. (3.55) and a particular solution of the same form as $\xi\mathbf{f_t}(\xi)$. Similar to the solution of homogeneous ODE given in Eq. (3.69a), the particular solution can be given as modes of deformation $\boldsymbol{\phi}_k$, which satisfies the internal equilibrium with the sideface tractions. Thus, if the nodal sideface force function $\mathbf{f_t}(\xi)$ is expressed as summation of power function $\xi^k\mathbf{f}_k$ (as given in Eq.(6.5)), then the particular solution can be represented as the nodal sideface displacement function $\mathbf{u_t}(\xi)$ and given

by

$$\mathbf{u_t}(\xi) = \sum_{k=0}^{N_t} \xi^{k+1} \boldsymbol{\phi}_k. \tag{6.11}$$

Substituting Eqs. (6.5) and (6.11) into Eq. (6.10), and rearranging the terms, we obtain the modal sideface displacement solution $\mathbf{\Phi_t}$ as

$$\mathbf{\Phi_t} = \sum_{k=0}^{N_t} \boldsymbol{\phi}_k = \sum_{k=0}^{N_t} - \left[(k+1)^2 \mathbf{E}_0 + (k+1)(\mathbf{E}_1^T - \mathbf{E}_1) - \mathbf{E}_2\right]^{-1} \mathbf{f}_k. \tag{6.12}$$

The corresponding boundary nodal force vector in equilibrium with these displacements is obtained by substituting Eq. (6.11) into Eq. (3.53) at $\xi = 1$ as

$$\mathbf{q_t} = \sum_{k=0}^{N_t} \left((k+1)\mathbf{E}_0 + \mathbf{E}_1^{\mathrm{T}}\right) \boldsymbol{\phi}_k. \tag{6.13}$$

Then, the displacement solution considering the sideface loads can be expressed as

$$\mathbf{u}(\xi) = \sum_{i=1}^{n} c_i \xi^{\lambda_i} \boldsymbol{\phi}_i^u + \sum_{k=0}^{N_t} \xi^{k+1} \boldsymbol{\phi}_k. \tag{6.14}$$

The net boundary nodal displacement $\mathbf{u}_b$ is obtained by evaluating Eq. (6.14) at $\xi = 1$ as

$$\mathbf{u}_b = \mathbf{\Phi}_b^u \mathbf{c} + \mathbf{\Phi_t}. \tag{6.15}$$

The net equivalent nodal boundary force vector in equilibrium with the net displacement field is given by

$$\mathbf{f}_b = \mathbf{\Phi}_b^q \mathbf{c} + \mathbf{q_t}. \tag{6.16}$$

Substituting the value of $\mathbf{c}$ from Eq. (6.15) into Eq. (6.16), the following equilibrium equation is achieved

$$\mathbf{K}^e \mathbf{u}_b = \mathbf{f}_b - \mathbf{q_t} + \mathbf{K}^e \mathbf{\Phi_t} = \mathbf{f}_b + \mathbf{f_t}, \tag{6.17}$$

where,

$$\mathbf{f_t} = -\mathbf{q_t} + \mathbf{K}^e \mathbf{\Phi_t}. \tag{6.18}$$

Here, $\mathbf{f_t}$ is the equivalent nodal boundary force vector due to sideface tractions $t(\xi)$. Once the global displacement vector is obtained from global equilibrium equation, the

integration constant $\mathbf{c}$ is obtained as

$$\mathbf{c} = \left[\mathbf{\Phi}_b^u\right]^{-1} \left(\mathbf{u}_b - \mathbf{\Phi}_\mathbf{t}\right), \tag{6.19}$$

The displacement field in each s-element is given as

$$\tilde{\mathbf{u}}(\xi,\eta) = \mathbf{N}_u(\eta)\mathbf{u}(\xi) = \mathbf{N}_u(\eta)\left[\sum_{i=1}^{n} c_i \xi^{\lambda_i} \boldsymbol{\phi}_i^u + \sum_{k=0}^{N_t} \xi^{k+1} \boldsymbol{\phi}_k\right]. \tag{6.20}$$

The stress field in the s-element is obtained by substituting Eq. (6.20) into Eq. (3.28) and expressed as

$$\boldsymbol{\sigma}(\xi,\eta) = \mathbf{D}\sum_{i=1}^{n} c_i \xi^{\lambda_i - 1}\left(\lambda_i \mathbf{B}_1(\eta) + \mathbf{B}_2(\eta)\right)\boldsymbol{\phi}_i^u + \mathbf{D}\sum_{k=0}^{N_t} \xi^k \left((k+1)\mathbf{B}_1(\eta) + \mathbf{B}_2(\eta)\right)\boldsymbol{\phi}_k. \tag{6.21}$$

Compared with Eqs. (3.79) and (3.81), the Eqs. (6.20) and (6.21) have an additional term on RHS, which considers the effect of sideface-tractions. It is observed that as $\xi \to 0$, only the stress modes with $\lambda_i = 0.5$ contribute to the stress intensity factors (SIFs) in Eq. (6.21) while the rest of terms tend to zero. Therefore, the expression for SIFs remains the same as Eq. (6.3). Here, the effects of sideface-tractions on the SIFs are embedded in the integration constants $\mathbf{c}$ given in Eq. (6.19).

6.3 Cohesive and frictional-contact formulations

In this section, we formulate the cohesive and frictional-contact tractions occurring at the crack-surfaces. In quasi-brittle materials (e.g. concrete) and composites, the presence of microcracks and aggregate/fibre interlocking induces stress transfer and energy dissipation across the crack surfaces. This induces highly localised non-linear material behaviour in such material over the crack surfaces which can be effectively captured by cohesive crack models (CCM) utilising a softening traction separation relation. On the other hand, the frictional contact conditions over the crack surface can be effectively satisfied using the computational methods such as penalty-stiffness and Lagrange multipliers methods [25, 221, 223]. In the present work, the penalty-stiffness method is

chosen as it is more robust and can be straightforwardly integrated with the cohesive crack models as explained below.

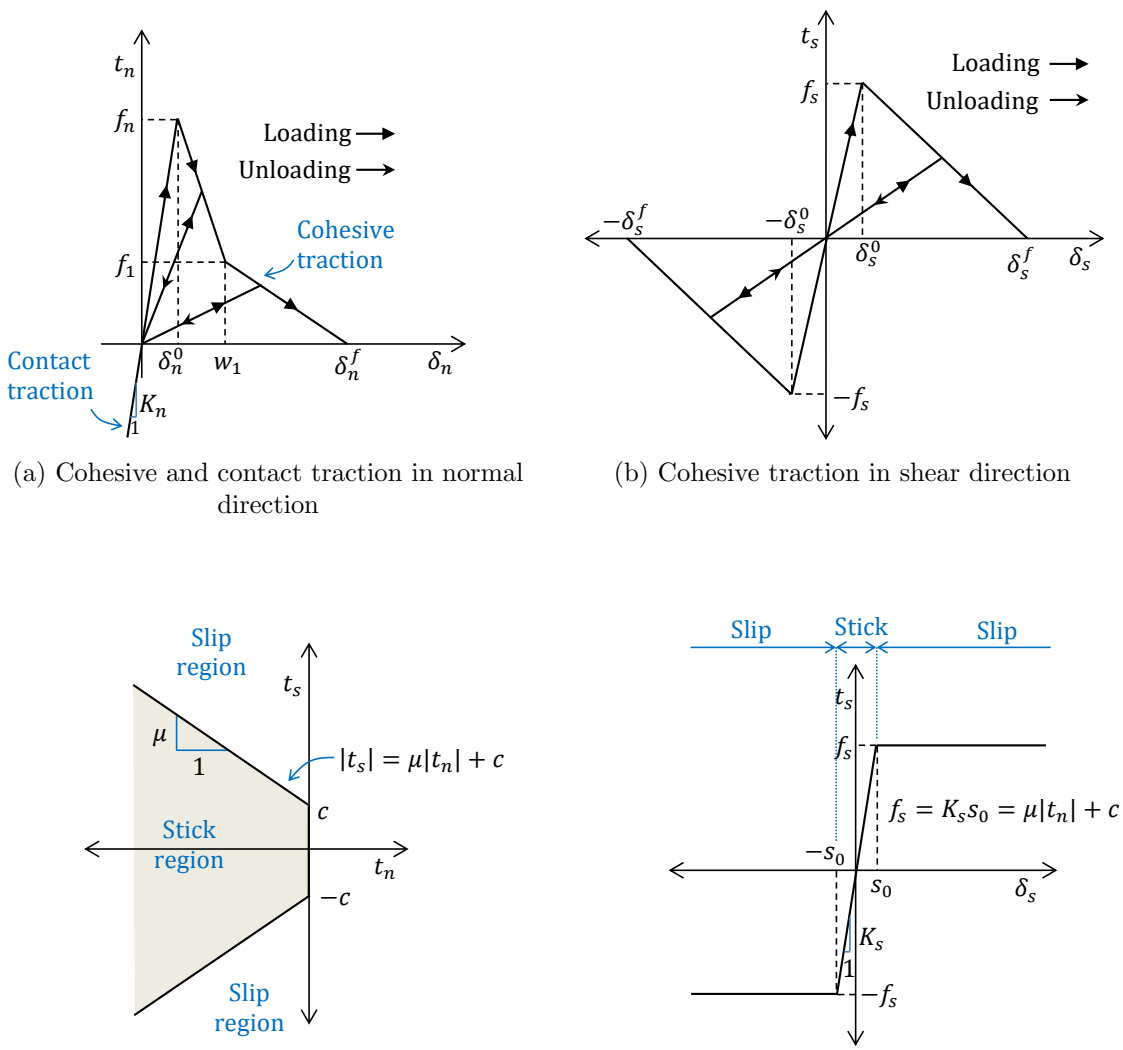
We define a traction-separation relation along the crack surface, given by

$$\left\{ \begin{array}{c} t_s \\ t_n \end{array} \right\} = \left\{ \begin{array}{c} c_s \\ c_n \end{array} \right\} + \left[\begin{array}{cc} k_{ss} & k_{sn} \\ k_{ns} & k_{nn} \end{array} \right] \left\{ \begin{array}{c} \delta_s \\ \delta_n \end{array} \right\} = \tilde{\mathbf{c}} + \tilde{\mathbf{K}}\boldsymbol{\delta}. \tag{6.22}$$

Here, δ_s and δ_n are the relative (in-plane) sliding and opening displacements of the crack faces. During the crack sliding or opening ($\delta_n > 0$), t_s and t_n represent the shear (tangential) and normal components of the cohesive traction acting along the crack surface. $\tilde{\mathbf{K}}$ is the constitutive matrix, where k_{ss} and k_{nn} are secant stiffness of the traction separation relation in sliding and opening modes respectively. k_{ns} and k_{sn} are the stiffness due to the coupling between sliding and opening modes (ignored in the present study; Ref: [4, 123, 188]). $\tilde{\mathbf{c}}$ is the adhesion vector, where c_s and c_n are the components in sliding and opening modes. An example of the resulting cohesive traction separation relation is shown in Fig. 6.3a-b. It can be noticed that the cohesive tractions have a softening behaviour with the peak values as f_s (shear strength) and f_n (tensile strength) in the shear and normal directions respectively. An irreversible loading behaviour is considered.

During the crack-closure ($\delta_n \leq 0$), t_s and t_n in Eq. (6.22) represent the shear and normal components of contact traction acting along the crack surface. In this case, $\tilde{\mathbf{K}}$ represents the penalty stiffness matrix, where $k_{ss} = K_s$ and $k_{nn} = K_n$ are tangential and normal penalty stiffness with sufficiently high values so that the contact conditions are reasonably satisfied.

The resulting contact traction in normal direction is shown in Fig. 6.3a. The friction, which creates traction contact condition on crack surfaces when $\delta_n \leq 0$, is assumed to follow the Coulomb friction model and is implemented using the penalty stiffness method as (Ref: Fig. 6.3c-d),



(a) Cohesive and contact traction in normal direction

(b) Cohesive traction in shear direction

(c) Coulomb friction model

(d) Frictional-contact traction in shear direction when $\delta_n \leq 0$

Figure 6.3: Traction-separation relations along the crack-surface

$$
\left.\begin{array}{c} k_{nn} = K_n \gg 0,\ k_{ss} = K_s \gg 0, \\ k_{ns} = k_{sn} = c_n = c_s = 0 \end{array}\right\} \quad \text{if} \quad K_s|\delta_s| < \mu|t_n| + c \quad (\text{Stick} - \text{condition})
$$

$$
\left.\begin{array}{c} k_{nn} = K_n \gg 0,\ k_{sn} = \text{sign}(\Delta\delta_s) \times \mu K_n, \\ k_{ss} = k_{ns} = 0,\ c_n = 0,\ c_s = \text{sign}(\Delta\delta_s) \times c \end{array}\right\} \quad \text{if} \quad K_s|\delta_s| \geq \mu|t_n| + c \quad (\text{Slip} - \text{condition}) \tag{6.23}
$$

Here, c and μ are surface adhesion and friction coefficient respectively, which are Coulomb friction parameters of the material surface. $\Delta\delta_s$ is the relative slip of the crack surface w.r.t. the last load step.

Implementation of the cohesive and contact traction acting on the crack surface is shown in Fig. 6.4. The interface between the crack surfaces (not containing the crack-tip) is modelled using 4-noded zero-thickness interface elements developed by [240, 386]. In such elements, the displacements within the element are linear while the normal and tangential tractions can be non-linear. The following expression for the local stiffness matrix $\bar{\mathbf{K}}_f^e$ of the interface element can be derived using the principle of virtual work

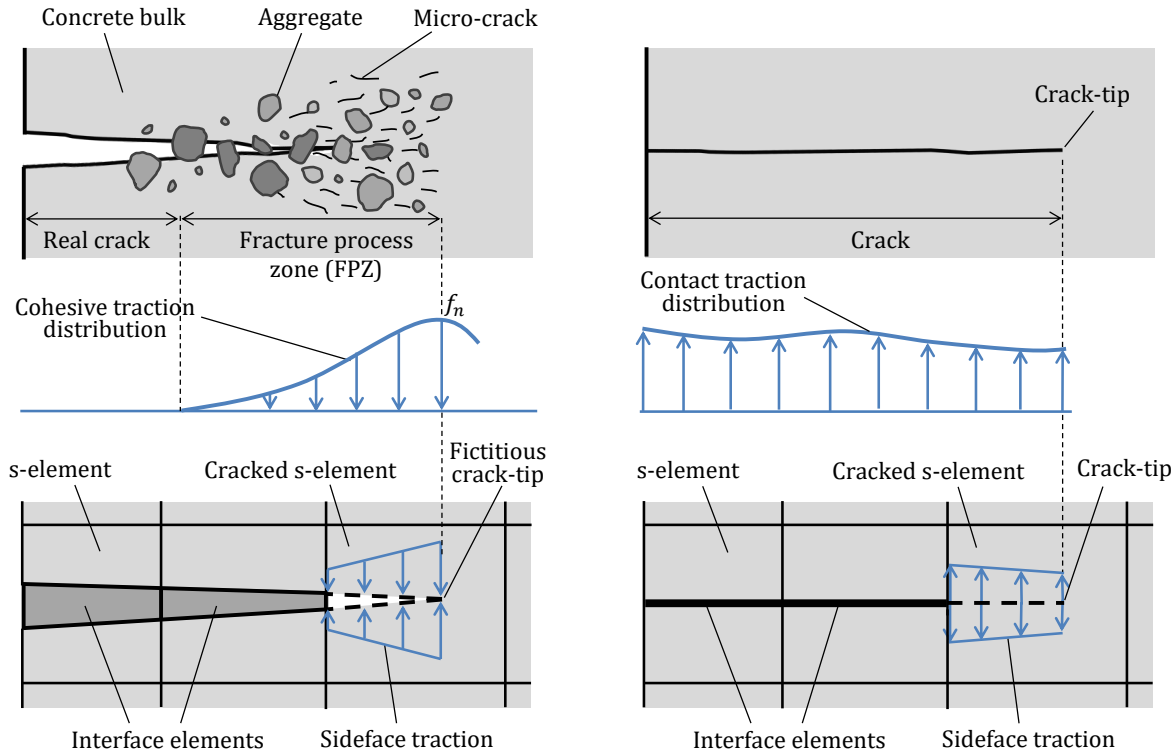
$$
\bar{\mathbf{K}}_f^e = \frac{A^e}{2} \sum_{i=1}^{N_g} H_i \mathbf{N}_{fi}^T \tilde{\mathbf{K}}_i \mathbf{N}_{fi}. \tag{6.24}
$$

Here, A^e (= element length $\times$ thickness) is the crack surface area modelled by the interface element and N_g is the number of Gaussian integration points used in the element. At the ith Gaussian integration point, H_i is the 1-D Gaussian weighing factor, $\tilde{\mathbf{K}}_i$ is material/penalty stiffness matrix given in Eq. 6.22 and $\mathbf{N}_{fi}$ is the shape function matrix given by

$$
\mathbf{N}_{fi} = \begin{bmatrix} -N_I & 0 & -N_{II} & 0 & N_{II} & 0 & N_I & 0 \\ 0 & -N_I & 0 & -N_{II} & 0 & N_{II} & 0 & N_I \end{bmatrix}, \tag{6.25}
$$

where,

$$
\begin{array}{c} N_I = 0.5(1 - \eta_i) \\ N_{II} = 0.5(1 + \eta_i) \end{array}. \tag{6.26}
$$



(a) Cohesive traction

(b) Contact traction

Figure 6.4: Schematic representation of crack surface tractions implemented using interface elements (away from the crack-tip) and sideface tractions (near the crack-tip)

Here, η_i is the 1-D natural coordinate of the ith Gaussian point. In the global coordinate system, the stiffness matrix of the interface element can be obtained as

$$\mathbf{K}_f^e = \mathbf{T}^T \bar{\mathbf{K}}_f^e \mathbf{T}, \tag{6.27}$$

where, $\mathbf{T}$ is the standard coordinate transformation matrix. The adhesion component of the traction in Eq. (6.22) is implemented as distributed load acting on the crack surface. The local load vector $\bar{\mathbf{c}}_f^e$ corresponding to adhesion is given by

$$\bar{\mathbf{c}}_f^e = \frac{A^e}{2} \sum_{i=1}^{N_g} H_i \mathbf{N}_{fi}^T \tilde{\mathbf{c}}_i, \tag{6.28}$$

where, $\tilde{\mathbf{c}}_i$ is the adhesion vector given in Eq. 6.22 at the ith Gaussian point. In the global coordinate system,

$$\mathbf{c}_f^e = \mathbf{T}^T \bar{\mathbf{c}}_f^e. \tag{6.29}$$

We now derive the expression for the traction distribution near the crack-tip. As explained in Section 6.2, the crack surfaces near the crack-tip are modelled as the sidefaces of the cracked elements in the SBFEM framework. Since the sidefaces cannot be discretised, the interface elements cannot be inserted in this region. Therefore, the surface tractions are directly applied on the sidefaces of the cracked element. As given in Eq. (6.4), the sideface tractions are expressed as the summation of power function of radial coordinate ξ. If the sideface tractions are assumed linearised ($N_t = 1$), then Eq. (6.4) reduces to

$$t_s(\xi) = \hat{f}_{s0} + \xi \hat{f}_{s1}, \tag{6.30a}$$

$$t_n(\xi) = \hat{f}_{n0} + \xi \hat{f}_{n1}. \tag{6.30b}$$

Putting the values of radial coordinate as $\xi = 0$ and $\xi = 1$,

$$t_{s0} = t_s(0) = \hat{f}_{s0}, t_{s1} = t_s(1) = \hat{f}_{s0} + \hat{f}_{s1}, \tag{6.31a}$$

$$t_{n0} = t_n(0) = \hat{f}_{n0}, t_{n1} = t_n(1) = \hat{f}_{n0} + \hat{f}_{n1}. \tag{6.31b}$$

Implementation of linearised tractions on the sidefaces of the cracked element is shown in detail in Fig. 6.5. The values of t_{s0}, t_{s1}, t_{n0} and t_{n1} are obtained at each iteration step using the sideface traction update scheme, as explained in the next section.

6.4 Numerical solution scheme and convergence criteria

The arc-length controlled algorithms are commonly utilised for solving the non-linear structural problems. They are suitable for capturing highly localised non-linear material behaviour, e.g. the FPZ in quasi-brittle materials such as concrete. These solvers can trace the equilibrium paths characterised with strong snap-back and snap-through that often occur in mixed-mode cohesive crack propagation. The arc-length controlled methods can be categorised into local and global ones [18, 187]. The equations in

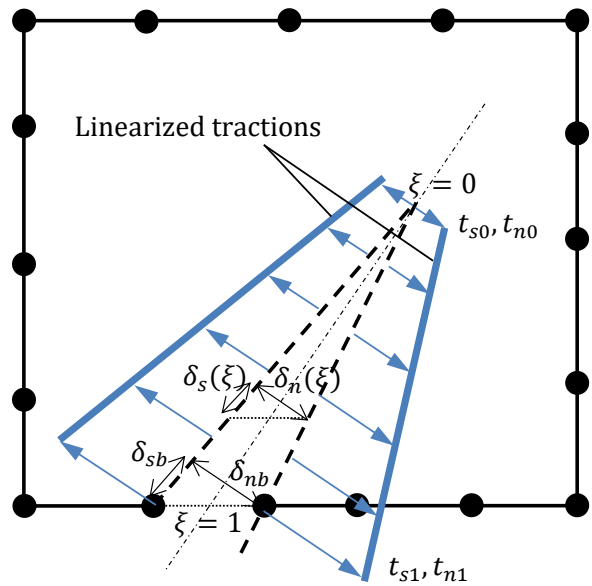


Figure 6.5: A cracked element showing linearized traction applied on the sidefaces

the local formulation include only the selected degrees of freedom which dominantly contribute to the material non-linearity while the global methods include all the degrees of freedom. It was demonstrated by [187] that the local arc-length controlled methods are superior to global ones in terms of numerical robustness and efficiency. Therefore, they are used in the present work where Duan's local updated normal plane constraint equation [388] is adopted.

As explained in Section 6.3, the crack surface tractions away from the crack-tip region are implemented using zero-thickness interface elements. The procedure for implementing interface elements with local arc-length controlled method is discussed in details in [4, 18, 123, 187, 188]. Near the crack-tip, the tractions are applied on the sidefaces of the cracked elements. A numerical solution scheme for implementing sideface tractions is not yet available in the existing literature and therefore, it is proposed in the present work as described below.

At the mth iteration within a load step, let $\mathbf{u}^m$ represent the nodal displacement vector. From $\mathbf{u}^m$, the stress intensity factors (K_I^m, K_{II}^m) can be obtained as explained

in Section 6.2. We define $T^m = \left\{ \begin{array}{cccc} t_{s0}^m & t_{n0}^m & t_{s1}^m & t_{n1}^m \end{array} \right\}$ as a set of values of linearised sideface tractions shown in Fig. 6.5. T^m is obtained using the following equations:

$$\left.\begin{aligned} t_{s0}^m &= t_{s0}^{m-1} + \alpha_0 K_{II}^m \\ t_{n0}^m &= t_{n0}^{m-1} + \alpha_0 K_I^m \end{aligned}\right\}, \tag{6.32a}$$

$$\left.\begin{aligned} t_{s1}^m &= (1-\alpha_1) t_{s1}^{m-1} + \alpha_1 t_{sb}^m \\ t_{n1}^m &= (1-\alpha_1) t_{n1}^{m-1} + \alpha_1 k_{nn} \delta_{nb}^m \end{aligned}\right\}, \tag{6.32b}$$

where,

$$t_{sb}^m = \begin{cases} k_{ss}\delta_{sb}^m & \text{(Crack opening or stick contact condition)} \\ \text{sign}(\Delta\delta_{sb}) \times (c + \mu K_n |\delta_{nb}^m|) & \text{(Slip contact condition)} \end{cases} \tag{6.33}$$

Eqs. (6.32a-b) represent the sideface traction update scheme. Here, α_0 and α_1 are the proportionality parameters with which the crack-tip tractions (t_{s0}, t_{n0}) and crack-mouth tractions (t_{s1}, t_{n1}) are updated respectively at each iteration. δ_{nb}^m and δ_{sb}^m are the opening and sliding displacements at the crack-mouth of cracked element (Ref: Fig. 6.5). $\Delta\delta_{sb}^m$ is the relative sliding of the crack-mouth w.r.t. the last load step. At the first iteration ($m = 1$) of each load step, T^0 is taken as the converged values of the linearised sideface tractions in the last load step.

From the obtained value of T^m, the sideface tractions $t_s^m(\xi)$ and $t_n^m(\xi)$ can be obtained as power (or linear) function of radial coordinate using Eqs. (6.30)-(6.31). Then, as explained in Section 6.2.2, the equivalent nodal boundary force vector $\mathbf{f}_\mathbf{t}^m$ for the sideface traction is obtained from Eq. (6.18) and assembled into the global internal force vector $\mathbf{f}_{\text{int}}^m$. The residual vector $\mathbf{r}^m$ is calculated using: $\mathbf{r}^m = \mathbf{f}_{\text{int}}^m - \mathbf{f}_{\text{ext}}^m$, where $\mathbf{f}_{\text{ext}}^m$ is the global external force vector. The iteration at each load step of the local-arc length algorithm is carried out till the SIFs (K_I^m, K_{II}^m) and the norm of residual vector $|\mathbf{r}^m|$ converge to a specified tolerance ($\varepsilon \approx 0$). This ensures that at each load step, the crack-tip singularities vanish when the equilibrium position is reached. The scheme can

be extended to various numerical techniques such as FEM, XFEM etc. to obtain the surface tractions near the crack-tip. In the SBFEM, the scheme is efficiently implemented as the SIFs are directly extracted from the formulation (Ref: Section 6.2.2) and therefore, the additional computational effort due to SIF calculation at each iteration step is negligible.

6.5 Solution procedure

Fig. 6.6 shows the flow chart of the solution procedure for solving the compressive fracture/debonding problems. The user inputs the geometry, material properties, loading and boundary conditions to the pre-processing module. This module discretises the geometry into elements based on the hybrid-polygon-quadtree meshing technique [4, 73]which provides several advantages such as the reduction in computational time required for stiffness matrices of the master cells, handling of hanging nodes between adjacent cells, easier mesh operations such as refining or splitting cells near domain boundary, and localised re-meshing in crack-propagation problems. The generated mesh is passed to the non-linear solver module, where the structural response is computed for each load step as explained in the last section. After each load step, a check is made if the maximum number of load steps is achieved. If not, then the crack-propagation condition is checked. In the present work, stress-based maximum principal stress criterion is used as crack-propagation condition. This implies that the crack propagates when $\max(P_1, P_2) \geq f_n$, where P_1 and P_2 are the principal stresses calculated at the crack-tip and f_n is the tensile strength of the material. If the crack-propagation condition is satisfied, its direction is calculated. The crack initiates/propagates in the maximum tensile principal stress plane calculated at the crack-tip. The crack-tip stress is obtained using the average stress method [389, 390] that smoothens the stresses in the crack-tip region order to accurately determine the principal stress direction. The

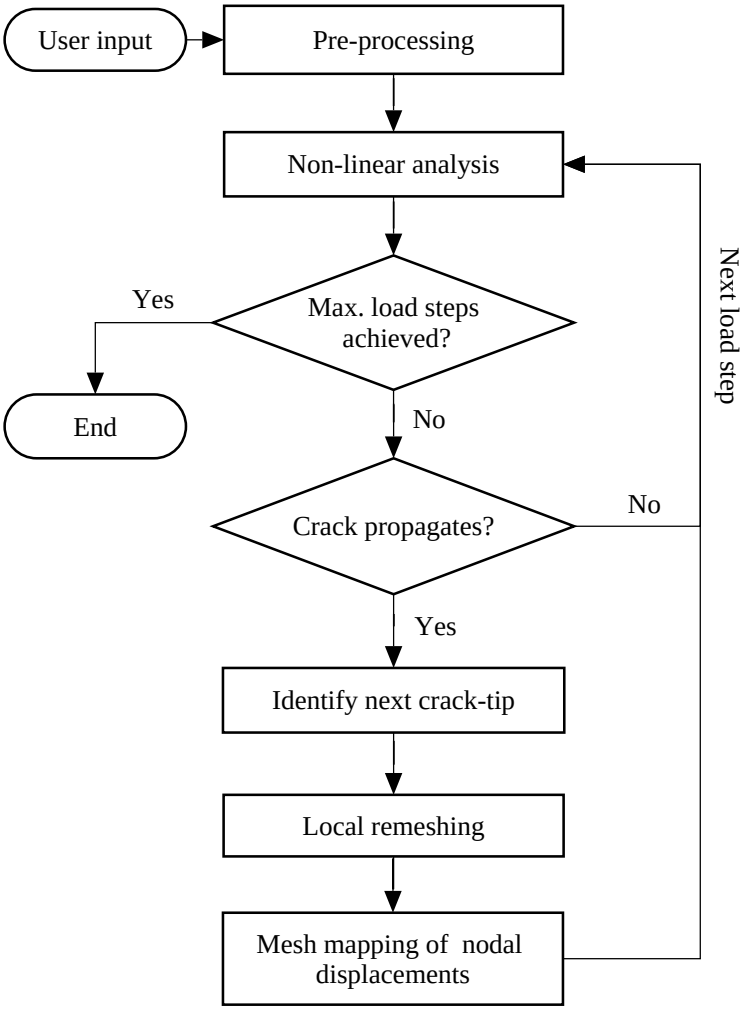


Figure 6.6: Flow chart of the solution procedure

stresses in the crack-tip region are extracted directly using Eq. (6.21) at the radius of $0.1L_e$, where L_e is the width of the cracked element. Due to the semi-analytical expression in Eq. (6.21), the computational effort in stress extraction around crack-tip is minimal. However, in the case where the crack-tip singularity exists (e.g. at the first crack propagation step in the SENS beam analysed in Section 6.6.2), the direction is calculated from the LEFM/SIF based maximum principal stress criterion [4, 123, 188]. Additionally, a constant crack-increment length is used in the present work for the crack-propagation simulation.

After the coordinates of the next crack-tip are found, a local re-meshing is done to create new crack surfaces between the current and the next crack-tips. Then, a mesh mapping is carried out to transfer the nodal displacements to a few newly generated nodes. This is followed by non-linear analysis for the next load step. In the SBFEM, the mesh mapping procedure is very simple [4, 123, 188] as the displacement or stress at any point within the element can be directly calculated from the formulations. Further, a very few new nodes are generated due to the local re-meshing. This makes the mesh mapping process very efficient and avoids any additional step to perform a non-linear analysis to reach an equilibrium position at the same nodal displacements after the mesh mapping is done.

6.6 Numerical examples

In this section, we present a few numerical examples to verify and show the practical applicability of the developed formulation.

First, the contact and cohesive interface formulations are validated using the benchmark examples. Next, two examples of practical interest are solved – The Brazilian split test is simulated for size-effect analysis. Then, the compressive fracture analysis in concrete is performed to capture interfacial debonding and matrix cracking.

In all the simulations, the value of proportionality parameters in sideface traction update scheme is: $1.0 \leq \alpha_0 \leq 8.0$ and $\alpha_1 = 0.2$. The value of penalty parameters in frictional-contact simulation and initial stiffness in cohesive crack simulation is kept as γE, where E is the Young's modulus of the bulk material and $100 \leq \gamma \leq 4000$ depending on the mesh element size.

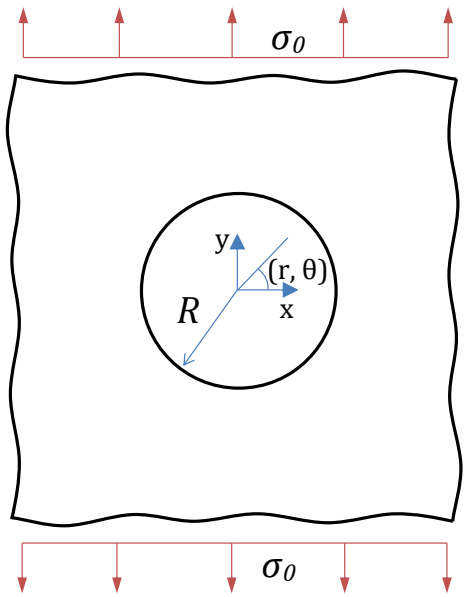


Figure 6.7: Infinite plate with circular hole

6.6.1 Crack with frictional-contact

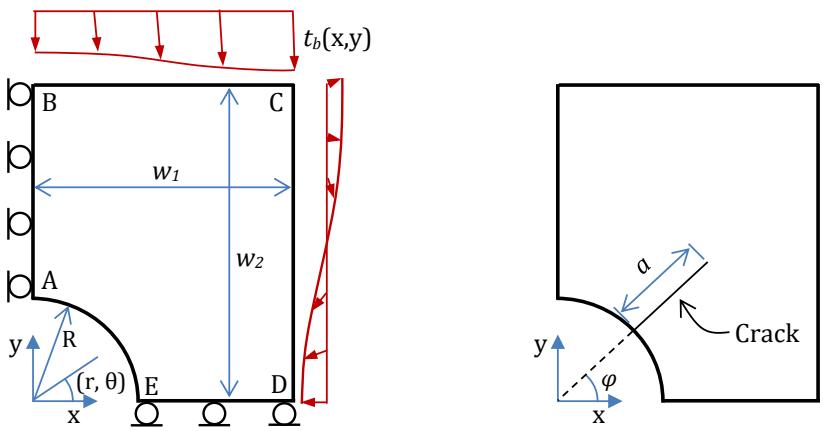
We consider an infinitely extending plate with a circular hole of radius R as shown in Fig. 6.7. Under far field uniaxial stress σ_0, the circular hole acts as a stress raiser and creates a non-uniform stress field around itself. An exact solution for this problem is available [2] in polar coordinate system (r, θ). With the origin kept at the centre of the hole and σ_0 acting along $\theta = 90°$, the solution is given by

$$\sigma_{rr}(r,\theta) = \frac{\sigma_0}{2}\left(1 - \frac{R^2}{r^2}\right) - \frac{\sigma_0}{2}\left(1 - \frac{4R^2}{r^2} + \frac{3R^4}{r^4}\right)\cos(2\theta), \tag{6.34a}$$

$$\sigma_{\theta\theta}(r,\theta) = \frac{\sigma_0}{2}\left(1 + \frac{R^2}{r^2}\right) + \frac{\sigma_0}{2}\left(1 + \frac{3R^4}{r^4}\right)\cos(2\theta), \tag{6.34b}$$

$$\tau_{r\theta}(r,\theta) = \frac{\sigma_0}{2}\left(1 + \frac{2R^2}{r^2} - \frac{3R^4}{r^4}\right)\sin(2\theta). \tag{6.34c}$$

In the present work, the infinitely extending plate is modelled as a finite-sized plate with appropriate boundary conditions as shown in Fig. 6.8a: AB and ED have displacement constraints in the orthogonal direction to the plate sections, BC and CD



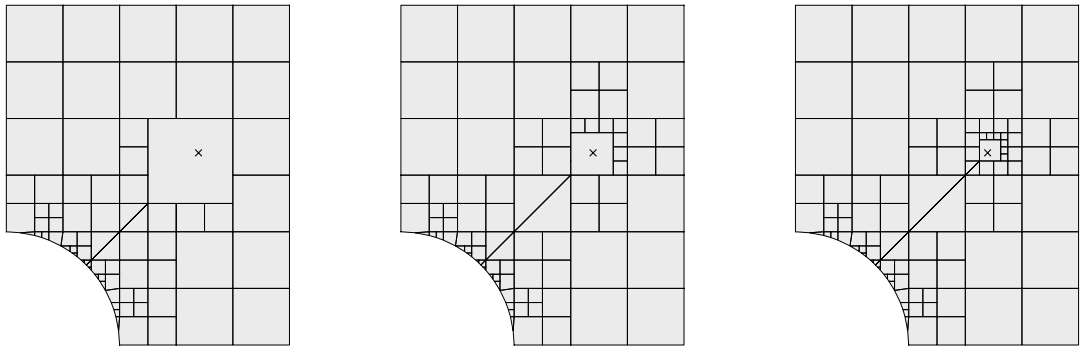
(a) Geometry and boundary conditions

(b) Partially-cracked

Figure 6.8: Finite-sized plate model

have traction constraints (t_b) which can be obtained from Eq. (6.34) using stress transformations, and arc AE is unconstrained. The applied far-field uniaxial stress is $\sigma_0 = -1$ kPa (compressive). The geometric properties of the plate are taken as: $w_1 = 2.5\,\text{m}$, $w_2 = 3\,\text{m}$ and $R = 1.0\,\text{m}$. To model the frictional-contact, we introduce a crack of length $a = 1.4\,\text{m}$ emerging from the hole at an inclination φ as shown in Fig. 6.8b. The friction coefficient is kept sufficiently high so that the contact is in stick-condition.

We first consider the crack segment with an inclination $\varphi = 45°$. We study the effect of the cracked element width (or size) on the accuracy of the solution obtained using the present formulation. Three meshes: Mesh-A, B and C with the cracked element widths 0.75 m, 0.375 m, and 0.1875 m respectively are considered as shown in Fig. 6.9. The base element width is kept constant at 0.5 m. The crack-tip is marked with '×'. Additional nodes are inserted along BC and CD to apply the traction constraints with a greater accuracy. The variation of contact tractions along the crack-segment is shown in Fig. 6.10. It can be observed that the results obtained for all the meshes are in good agreement with those obtained from analytical solution given in Eq. (6.34). The sideface traction approach correctly estimates the contact traction in the crack-tip re-



(a) Mesh-A (b) Mesh-B (c) Mesh-C

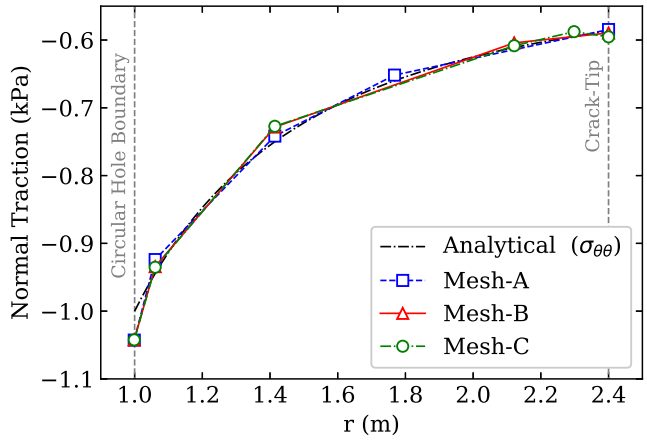
Figure 6.9: Mesh for the partially-cracked plate model

gion, even when the size of the cracked element is large (Mesh-A). Slight deviation from the analytical solution is observed near $r = 1.0$, which is attributed to the polygonal discretisation of the circular-hole boundary.

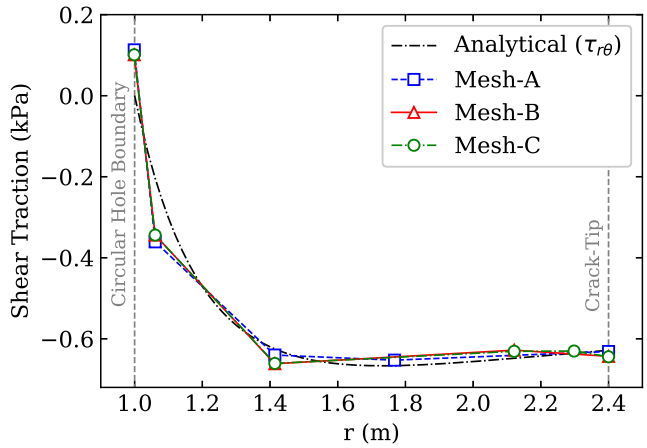
A convergence analysis is carried out to study the effect of crack-tip proportionality parameter α_0 of the sideface traction update scheme given in Eqs. (6.32)a-b used in the contact analysis. The value of crack-mouth proportionality parameter (α_1) is kept constant at 0.2. Convergence of following variables are studied: Stress intensity factors (K_I and K_{II}) and normalised difference in contact traction (ε) given by

$$\varepsilon = \sqrt{\frac{\sum_j \left(t_{n,j} - t_{n,j}^{ref}\right)^2 + \sum_j \left(t_{s,j} - t_{s,j}^{ref}\right)^2}{\sum_j \left(t_{n,j}^{ref}\right)^2 + \sum_j \left(t_{s,j}^{ref}\right)^2}}. \tag{6.35}$$

Here, $t_{n,j}$ and $t_{s,j}$ are the contact tractions in normal and shear directions calculated at the jth contact node along the crack surface. $t_{n,j}^{ref}$ and $t_{s,j}^{ref}$ are the reference values which are obtained from the linear-elastic solution of an un-cracked plate (or reference plate) under compressive load as shown in Fig. 6.8a. The convergence results of the contact analysis of Mesh-A are shown in Fig. 6.11. It can be observed from Fig. 6.11c that an almost linear convergence is achieved. From Fig. 6.11a-b, it is noticed that the SIFs tend to zero with increase in number of iterations. Further, the rate of convergence is low at a smaller value of α_0. This is because at each iteration, the crack-tip contact



(a) Normal traction distribution



(b) Shear traction distribution

Figure 6.10: Variation of contact tractions along the crack surface for $\varphi = 45°$ (Analytical: [2])

traction is updated with a smaller fractions of SIFs (K_I and K_{II}) as given in Eqs. (6.32)a-b. As the value of α_0 increases, the rate of convergence increases. However, a very high value of α_0 can introduce a sudden increase in the value of crack-tip contact traction at the first iteration step. This leads to oscillatory convergence behaviour of SIFs that reduces the rate of convergence or may lead to divergence.

Fig. 6.12a shows the contour plots of the field variables obtained from linear elastic analysis of the reference (un-cracked) plate. Fig. 6.12b-d show the contour plots obtained from frictional contact analysis of Mesh-A with crack inclination angles $\varphi = 30°$,

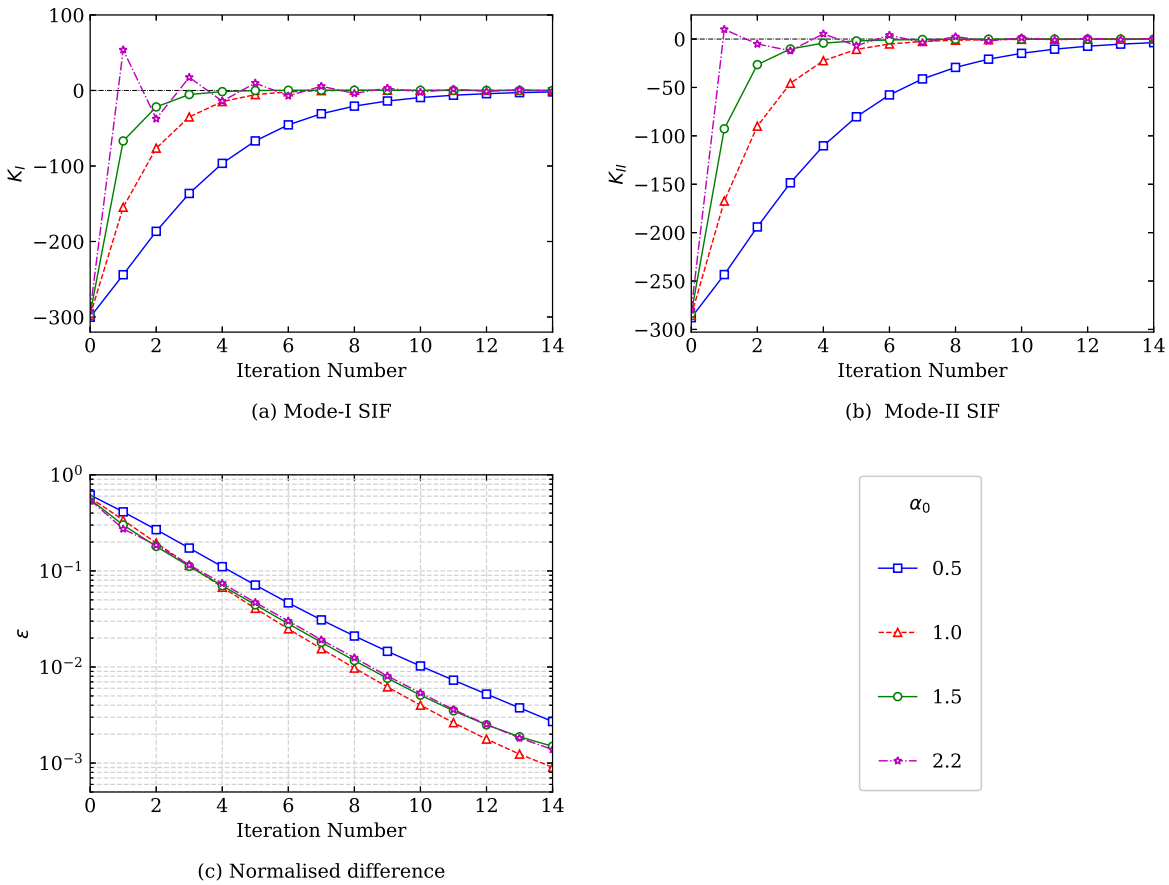
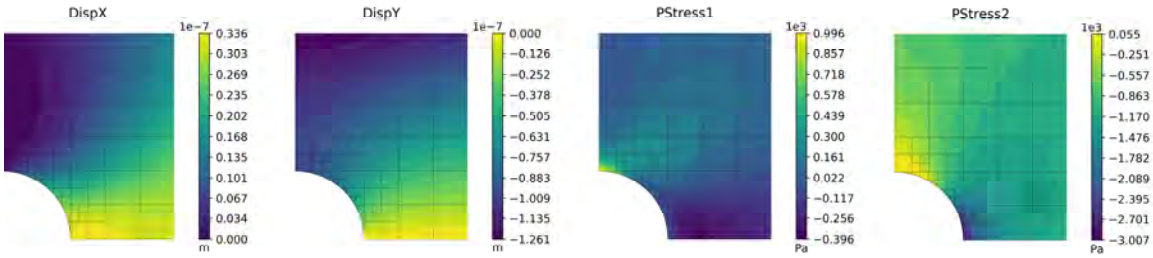
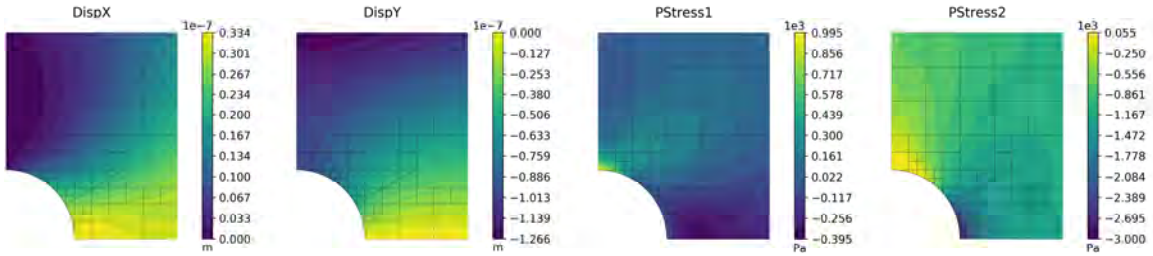


Figure 6.11: Convergence results of contact analysis of Mesh-A for $\varphi = 45°$

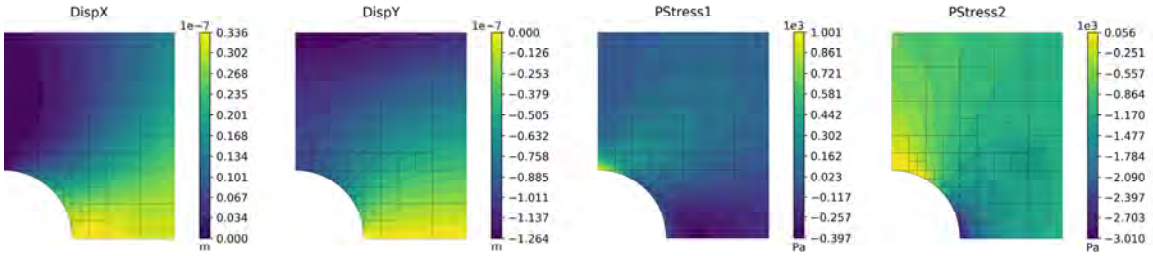
45°, and 60°. The field variable shown are: Displacements along x-axis and y-axis (DispX and DispY), and principal stresses (PStress1 and PStress2). It is observed that the solutions obtained from the contact analysis are in excellent agreement with the solutions obtained from the linear elastic analysis of the reference (un-cracked) plate. As the contact tractions are correctly estimated along the crack surfaces using the developed approach, accurate values of displacement and stress fields are obtained. The crack-tip singularities vanish completely, which represents the physically correct state of the stresses in the crack-tip region when the crack is in stick condition.



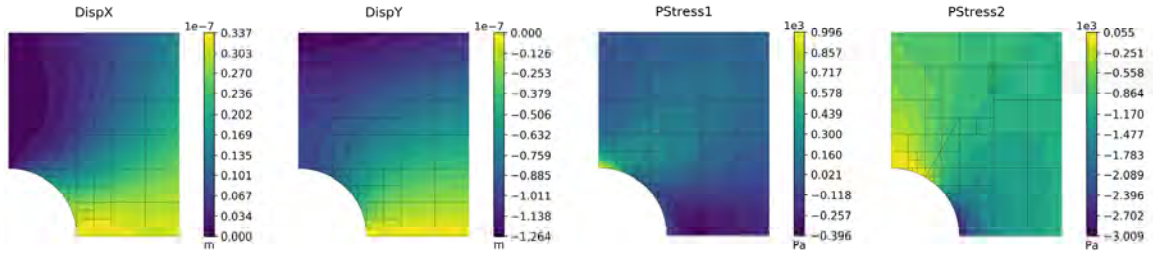
(a) Reference (un-cracked) plate



(b) $\varphi = 30°$



(c) $\varphi = 45°$



(d) $\varphi = 60°$

Figure 6.12: Field variable contour plots for Mesh-A

6.6.2 Cohesive crack propagation

We now validate the developed formulation using the benchmark example of the single edge notched shear beam (SENS), which was experimentally analysed by [5] for mixed mode crack propagation. The geometry and boundary conditions of the beam are

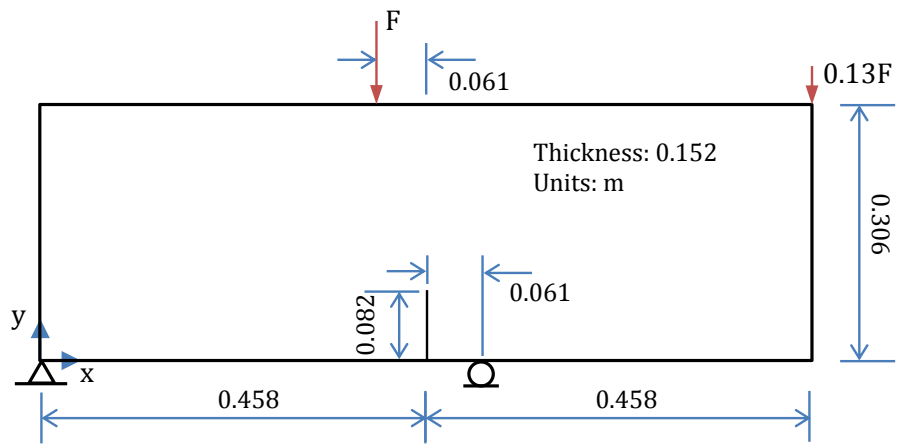
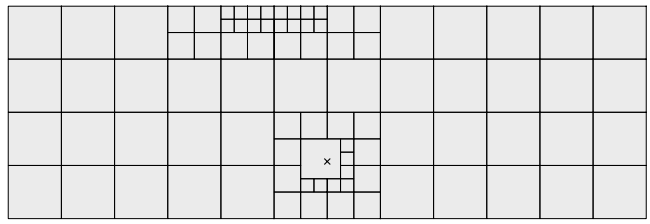


Figure 6.13: SENS model under shear loading

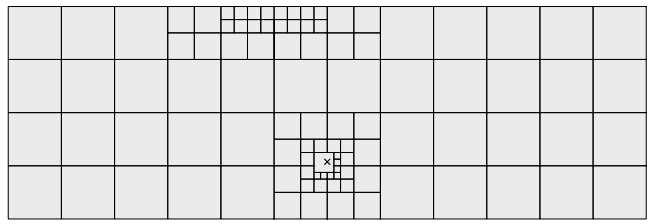
shown in Fig. 6.13. The material properties available from [5] are: Young's modulus $E = 24.8\,\text{GPa}$ and the Poisson's ratio $\nu = 0.18$. The mode-I fracture energy (G_I^c) and the tensile strength (f_n) were not reported. A wide range of their values have been used by different authors [3, 4, 18, 188, 386]. In the present work, Petersson's bilinear softening law for opening mode is utilised with $G_I^c = 125\,\text{N/m}$ and $f_n = 4.0\,\text{MPa}$ [3]. Mode-II fracture energy G_{II}^c and shear strength f_s (with linear softening) are assumed to be present in small amount (5-10% of G_I^c and f_n respectively) so as to balance Mode-II singularity at the crack-tip.

We verify the accuracy of the current implementation using three meshes: Mesh-A, B and C with different mesh-densities over the crack paths. The initial meshes are shown in Fig. 6.14. The base-mesh size (or maximum element size) is kept same in all the meshes as 76.3 mm. The minimum element size (along the crack-curve) for the three meshes are: 19.1 mm, 9.54 mm, and 4.77 mm respectively. Further, the corresponding cracked element widths are 57.2 mm, 28.6 mm, and 14.3 mm respectively. The crack increment lengths are kept in the range 12 mm to 44 mm.

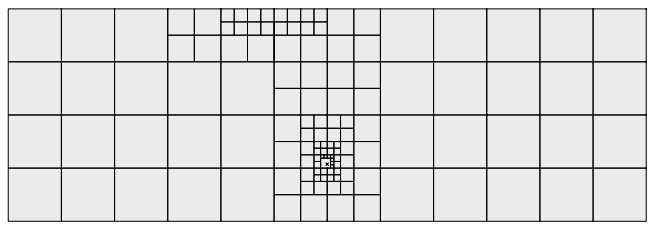
Fig. 6.15 shows the load-CMSD (crack mouth sliding displacement) curves and crack



(a) Mesh-A



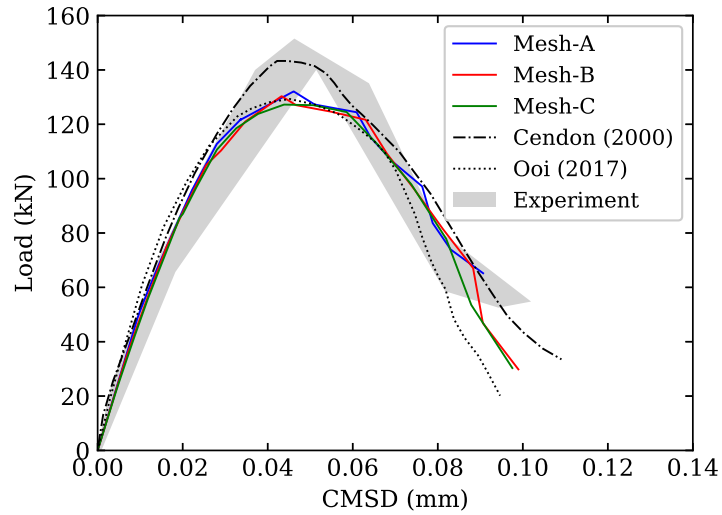
(b) Mesh-B



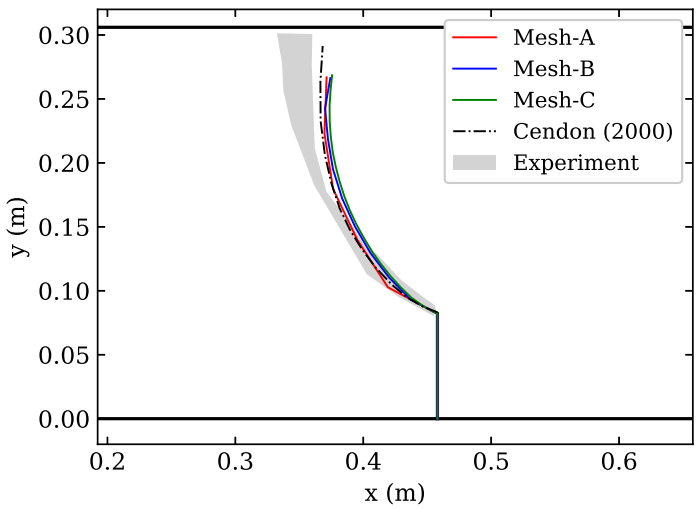
(c) Mesh-C

Figure 6.14: Initial meshes for the SENS model

paths obtained from the mixed-mode cohesive crack propagation simulation of the SENS beam. It can be observed that the present results are well within the experimental range obtained by [5] and agree well with the existing numerical results [3, 4]. The predicted maximum load is in the range 130 kN - 132 kN. These results confirm the correct implementation of the present formulation. Further, it is also inferred that sideface traction approach can calculate the crack-tip stresses and thus, the stress-based crack-propagation condition and direction with sufficient accuracy even when relatively coarser crack-tip mesh is used. Fig. 6.16 show the various stages of crack propagation of SENS under shear load for Mesh-A and crack increment length $\Delta a = 44\,\text{mm}$. Fig. 6.16a shows the development of cohesive traction in the FPZ along the crack path during the pre-peak load stage. In Fig. 6.16b, the near-peak load stage is shown where a load



(a) Load-CMSD curve



(b) Crack path

Figure 6.15: Results for cohesive crack propagation in SENS (Ref - Cendon (2000): [3], Ooi (2017): [4], Experiment: [5])

of 124.4 kN is applied with CMSD = 0.0609 mm, followed by post-peak stages in Fig. 6.16c-d.

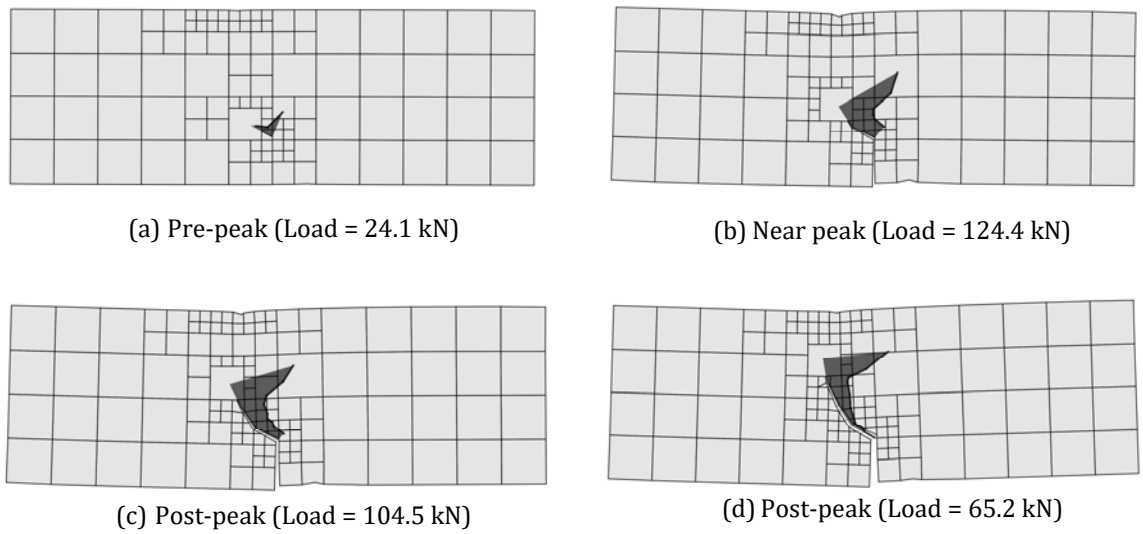
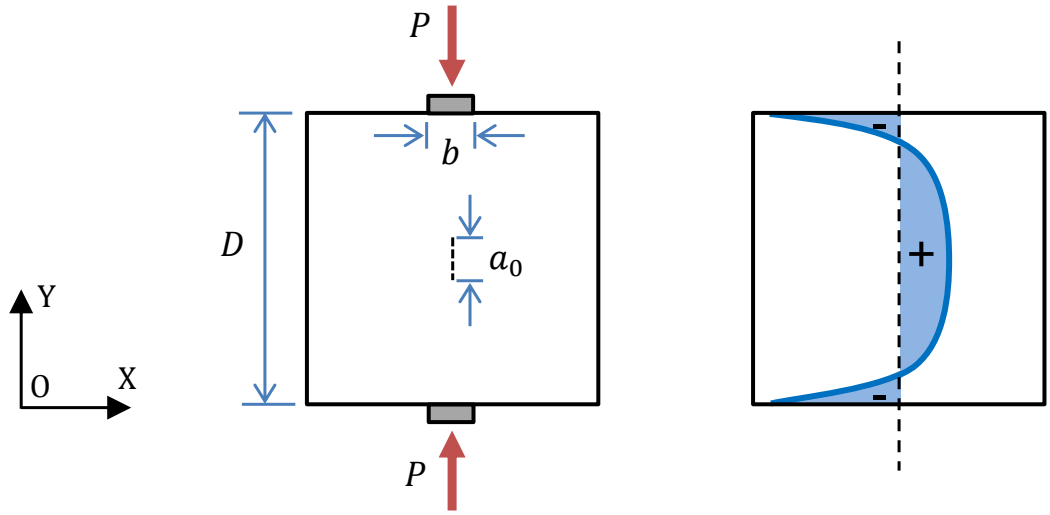


Figure 6.16: Cohesive traction distribution along the crack in the deformed Mesh-A of SENS. (Deformation Scale: 100)

6.6.3 Brazilian split test

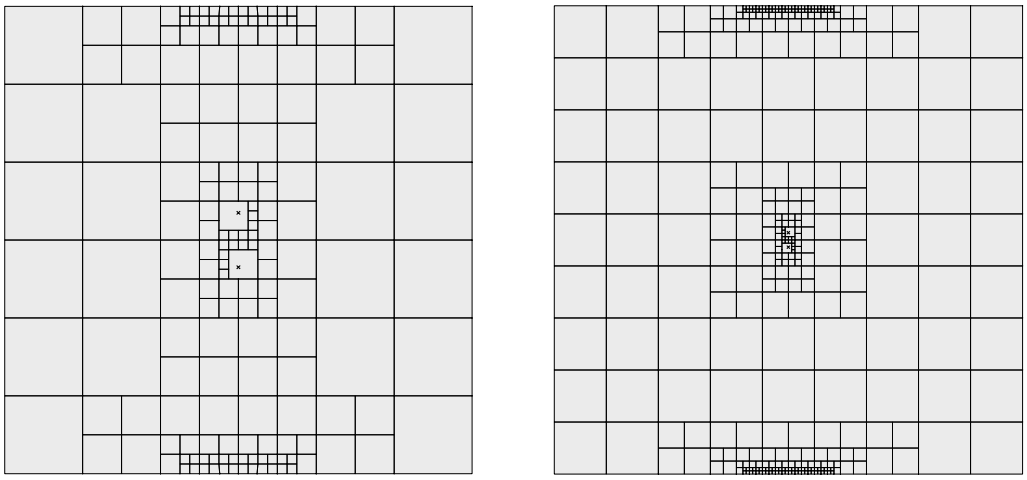
Brazilian split tests are commonly used to evaluate the tensile strength of concrete. In these tests, a compressive point load P is applied along the two opposite faces of a specimen, as shown in Fig. 6.17a schematically. A prismatic specimen with a square (or cylindrical) section [6, 7, 391] is generally used with side length D and thickness t. In order to distribute the stresses generated at the loading points, a bearing strip of width b is used, which prevents local crushing and multiple cracking near the loading points. For a linear-elastic material and a relatively narrow bearing strip, a nearly uniform tensile stress state, as shown in Fig. 6.17b, is produced at the centre vertical plane of the specimen [2, 6]. Therefore, the expected rupture mode is the split of the specimen into two halves across the plane of loading. However, the rupture mechanism in the splitting test is not simple, and a few failure mechanisms can be identified during the test [391]. Nevertheless, in most cases, the maximum load is governed by the primary rupture mode that is marked by the growth of a single crack in the mid-vertical plane of the specimen from the centre towards the top/bottom edges.



(a) Schematic diagram

(b) Stress distribution σ_{xx} along the centre vertical plane

Figure 6.17: Specimen for the Brazilian split test



(a) $D/l_{ch1} = 0.9$

(b) $D/l_{ch1} = 2.7$

Figure 6.18: Initial meshes for the Brazilian test specimen

The relative width of the bearing strip is defined as $\beta = b/D$. For smaller values of $\beta\,(\leq 0.16)$, the maximum tensile stress f at the centre of the prismatic specimen is approximately given as [6]:

$$f = \frac{2P}{\pi D t} \tag{6.36}$$

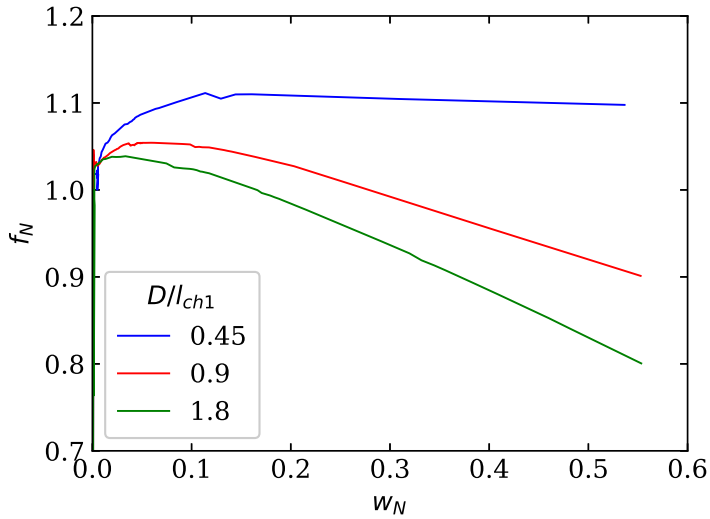
For a peak (ultimate) load P_u applied on the specimen, one can obtain the splitting tensile strength f^{st} as:

$$f^{st} = \frac{2P_u}{\pi D t} \tag{6.37}$$

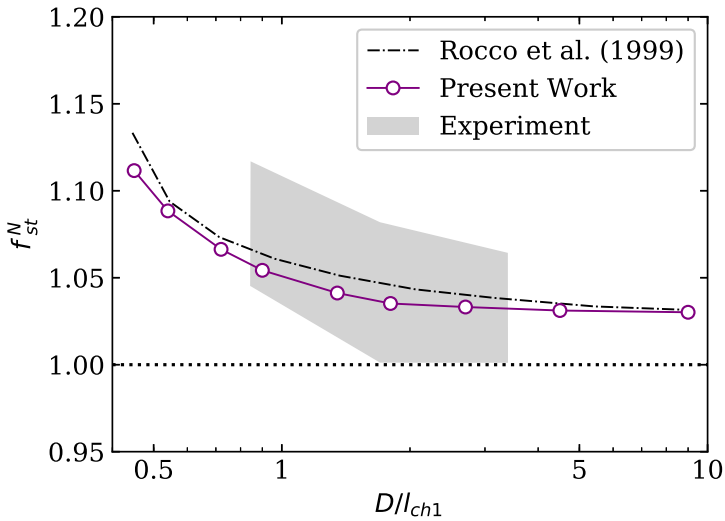
It was shown by [6] that Eq. 6.37 slightly overestimates the splitting tensile strength of brittle-elastic materials. In quasi-brittle material like concrete, this is further intensified due to the size-effect.

In this example, we perform the size effect study of concrete in the Brazilian split test for various sizes of the specimen. The relative width of the bearing strip is kept constant as $\beta = 0.08$. The failure is simulated using cohesive crack propagation, where only the primary rupture mode is considered. The bulk material is assumed linear elastic with Young's modulus E and the Poisson's ratio ν. Since the specimen is un-notched, the maximum load is only governed by the slope of the initial portion of the softening curve [6]. Therefore, a linear softening curve is utilised with mode-I fracture energy G_I^c and tensile strength f_n. The critical crack-opening width is given by $w_1 = 2G_I^c/f_n$. Further, we define the relative-size of the specimen as D/l_{ch1}, where l_{ch1} is the dominant material size given by $l_{ch1} = Ew_1/2f_n$ [6].

We consider the specimen models of increasing relative-sizes D/l_{ch1} in the range $0.45 - 9.0$ which are discretised using the quadtree mesh. The corresponding relative element-sizes (L_e/l_{ch1}) for the base-mesh (that is, the maximum element size) is kept in the range $0.075 - 0.9$. Further, the relative element-size along the cohesive-crack (which has the minimum element size) is kept sufficiently small and in the range $0.01 - 0.028$ to accurately capture the cohesive tractions in all the specimen considered. The quadtree



(a) Normalised stress-displacement curves



(b) Size-effect curve

Figure 6.19: Results of the Brazilian test simulation (Ref - Rocco et al. (1999): [6], Experiment: [7])

mesh for the relative specimen size $D/l_{ch1} = 0.9$ and 2.7 is shown in Fig. 6.18 for comparison. Since the maximum tensile stress occurs at the centre of the specimen in a vertical plane (primary rupture mode assumed), the crack initiation is modelled by placing an undamaged cohesive-crack of a small length (a_0) at the beginning of the simulation as shown in these meshes. It is noticed that in specimen with larger value

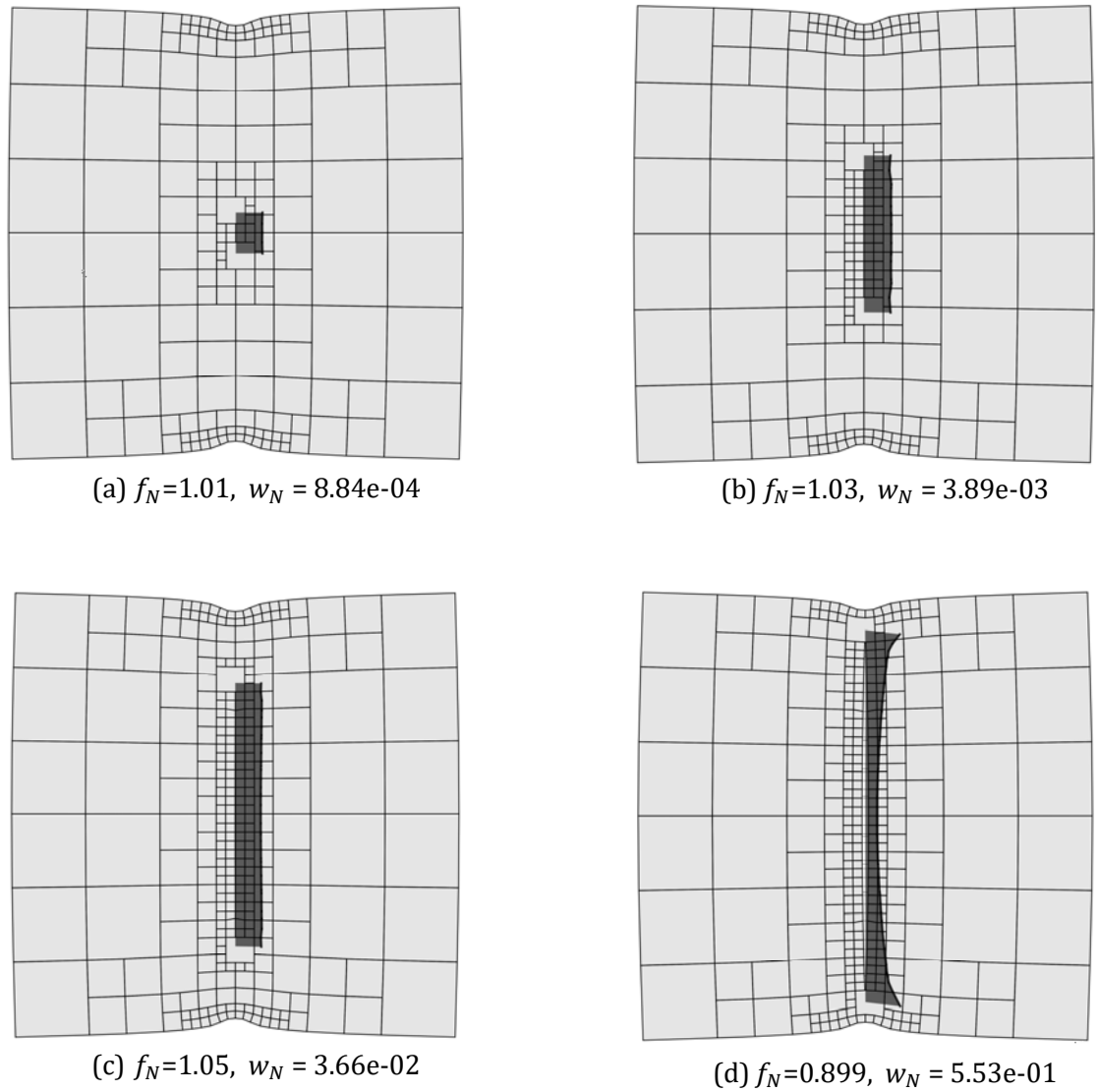


Figure 6.20: Cohesive traction distribution along the crack in the Brazilian test specimen of size $D/l_{ch1} = 0.9$ (Deformation Scale: 100)

of D/l_{ch1}, larger number of elements are used, especially along the crack path.

The normalised stress-displacement curves obtained from the simulation for a few values of D/l_{ch1} are shown in Fig. 6.19a. Here, f_N is the maximum normalised tensile stress at the centre of the specimen given by $f_N = f/f_n$ and w_N is the normalised displacement given by $w_N = w/w_1$, where w is the crack opening displacement measured at the centre of the specimen. The size effect curve is shown in Fig. 6.19b, where the

normalised splitting tensile strength f_N^{st} (given by $f_N^{st} = f^{st}/f_n$) is obtained from the peak value of maximum normalised tensile stress curve f_N (shown in Fig. 6.19a) for various specimen sizes D/l_{ch1}. It is observed that the present results are in good agreement with those reported in [6, 7]. The splitting tensile strength is strongly dependent on the size of the specimen. For the lower relative size of the specimen (D/l_{ch1}), the splitting tensile strength is over-predicted by a relatively large margin. As the relative size of the specimen increases, the splitting tensile strength approaches towards the tensile strength of the specimen. As suggested by [6, 7], the size effect of the specimen also depends on the relative width of the bearing strips, though it is not covered in the present study. The cohesive traction distribution at various stages of crack-propagation for specimen size $D/l_{ch1} = 0.9$ is shown in Fig. 6.20. It is observed that for the selected specimen size, the traction distribution remains almost uniformly distributed until the crack-tips reach near the specimen edges.

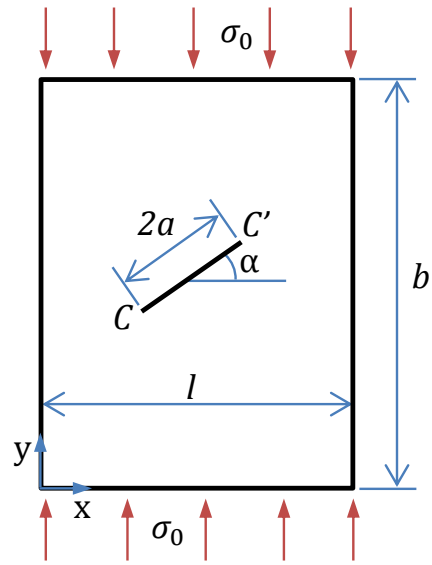
6.6.4 Compressive fracture analysis in concrete

We consider a mortar specimen with a slender inclusion made of limestone (aggregate interface) at its centre as shown in Fig. 6.21a. This specimen was experimentally analysed by [8] to study the phenomena of interfacial cracking (at the limestone-mortar interface) and subsequent matrix/mortar cracking under uniaxial compressive load in vertical direction. Three distinct stages of cracking were observed as the compressive load was increased: First, debonding occurs at the interface of limestone and mortar. Second, wing crack initiates at the crack-tips of the interfacial crack. Third, the wing crack propagates into the matrix in the direction of the applied load.

To analyse this problem numerically, we assume the specimen as a two-dimensional model under plane stress condition as shown in Fig. 6.21b. The limestone inclusion is modelled as an initial crack interface CC' inclined at an angle α to the horizontal. The geometric and material properties available from [8] are following: $l = 0.15\,\mathrm{m}$,

Figure removed due to copyright restriction (available in master copy)

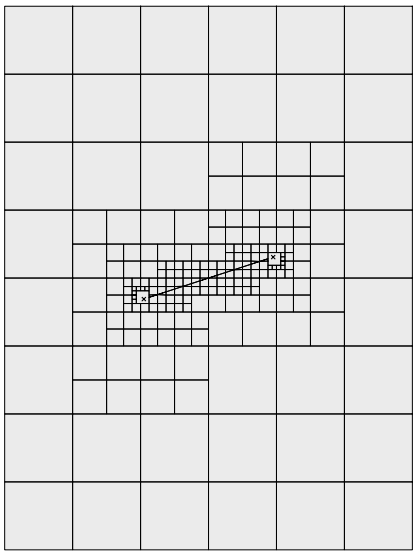
(a) Mortar specimen [8]



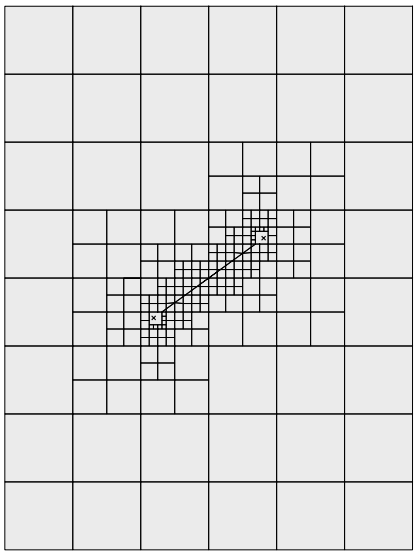
(b) Specimen modeled in 2D

Figure 6.21: Geometry and loading of the specimen

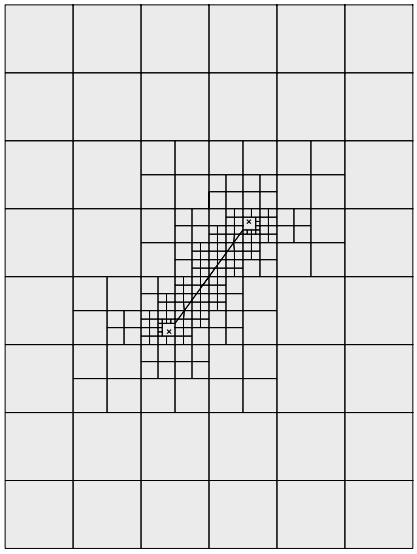
$b = 0.2\,\text{m}$, $a = 0.025\,\text{m}$ and thickness is $0.076\,\text{m}$; bulk material of the specimen is considered linear-elastic with Young's modulus $E = 27\text{GPa}$ and Poisson's ratio $\nu = 0.21$. Under compressive load, the initial crack interface undergoes debonding and slip, which is captured using Mohr-Coulomb shear failure criteria given by $t_s \geq c + \mu t_n$, where t_s and t_n are the frictional/shear and normal stresses respectively developed on the interfacial crack surface, c is the surface adhesion and μ is the friction coefficient. The experimentally obtained values of the friction parameters are: $c = 0.73\,\text{MPa}$ and $\mu = 0.25$. No information on cohesive fracture parameters was provided. Therefore, a linear traction separation law is assumed (for mortar cracking in opening mode) with $G_I^c = 60\,\text{N/m}$ and $f_n = 4\,\text{MPa}$. Mode-II fracture energy G_{II}^c and shear strength f_s (with linear softening) are assumed to be present in small amount (5-10% of G_I^c and f_n respectively). Four cases of interfacial crack inclination angle α are considered: 18°, 36°, 54° and 72°. The initial meshes are shown in Fig. 6.22. The crack increment lengths



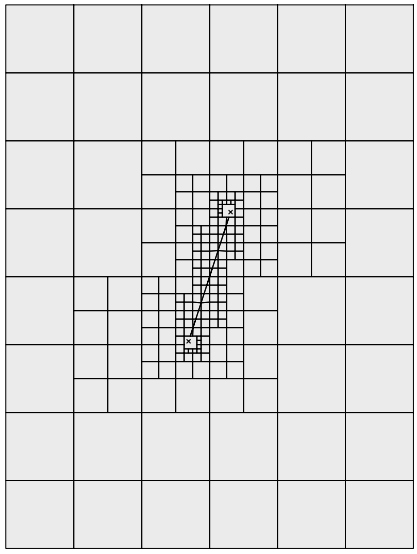
(a) $\alpha = 18°$



(b) $\alpha = 36°$



(c) $\alpha = 54°$



(d) $\alpha = 72°$

Figure 6.22: Initial meshes for various values of interfacial crack inclination angle α

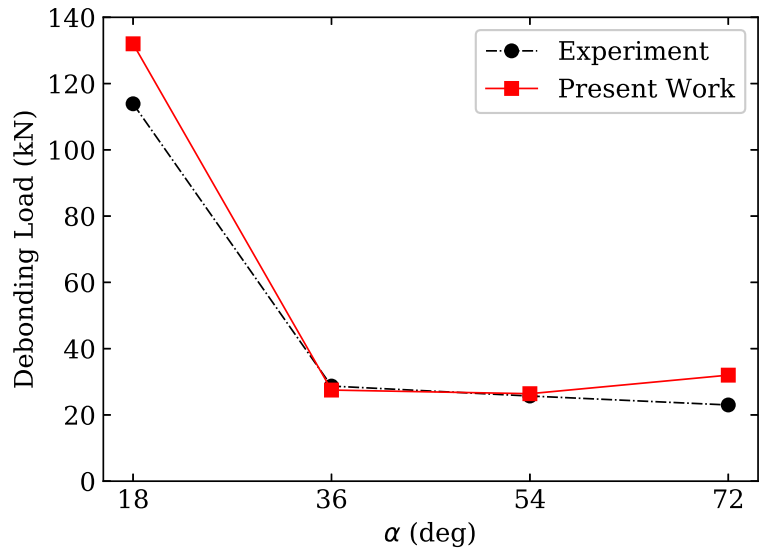


Figure 6.23: Debonding load (P_D) (Experiment: [8])

are kept in the range of $2.0 - 5.0\,\text{mm}$.

We now define some terminologies used in the present analysis. P is the applied compressive load given as: $P = A\sigma_0$, where σ_0 is the applied compressive stress and A is the section area of the specimen. P_D and P_I are the applied compressive loads measured at the debonding of the initial crack interface and at the wing crack initiation respectively. P_N is the normalised load given by, $P_N = P/P_D$ and $P_{I,N}$ is the normalised crack initiation load given by, $P_{I,N} = P_I/P_D$, where P_D is the debonding load obtained in the present work (Ref: Fig. 6.23). Δs is the crack mouth sliding displacement (CMSD) measured along the interfacial crack surface (CC') at the initial crack-tip (C or C'). [8] measured the sliding (δ_s) and opening (δ_n) of the crack surface using clip gages in the crack initiation (or kink) direction as shown in Fig. 6.21a. Therefore, CMSD can be obtained from the clip gauge measurements using $\Delta_s = \sqrt{\delta_s^2 + \delta_n^2}$.

We now analyse all the three stages of compressive cracking:

Debonding stage: The debonding at the initial crack interface is based on Mohr-Coulomb shear failure criterion as suggested by [8]. Fig. 6.23 shows the debonding load (P_D) obtained in the present work for various values of initial crack interface angle (α).

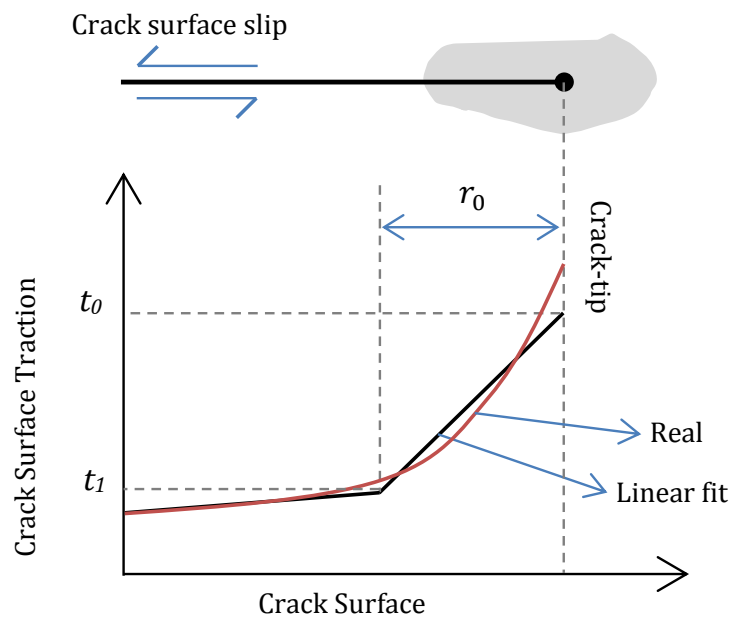


Figure 6.24: Crack surface traction distribution in the crack-tip region after debonding

The results are in good agreement in general with those obtained experimentally by [8]. Some deviations are observed at $\alpha = 18°$ and $72°$ that are attributed to the use of constant value of friction parameters in the present analysis.

Crack initiation stage: When the debonding occurs, the frictional contact condition at the initial crack interface changes from stick to slip. Near the crack-tip, this creates a small region of high stress gradient as shown in Fig. 6.24. The stress distribution in this region is captured using the sideface traction approach. The crack surface traction (t_s, t_n) distribution in the crack-tip region is idealised as linear-fit as shown and applied on the sidefaces of the cracked element. The value of this linearised sideface traction changes from $t_1 = (t_{s1}, t_{n1})$ at the boundary of the cracked element to $t_0 = (t_{s0}, t_{n0})$ at the crack-tip over length r_0, where r_0 is equal to the length of the sidefaces.

At each load step, the crack-tip is checked for wing-crack initiation using the strength

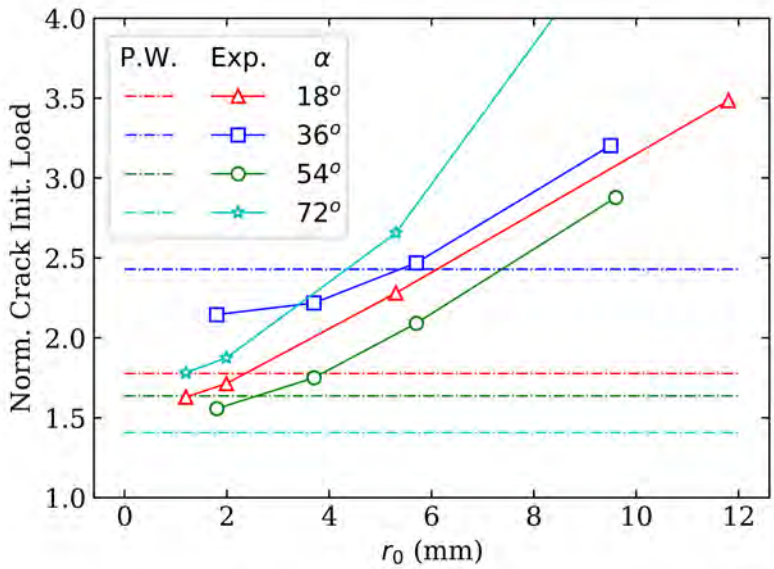
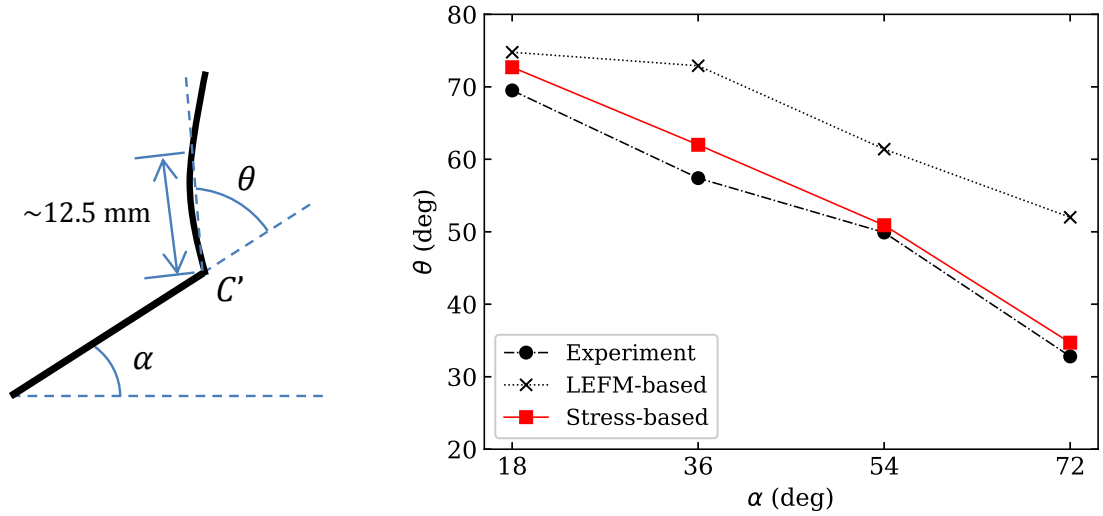


Figure 6.25: Variation of normalised crack initiation load with r_0 and α (P.W. = Present Work, Exp. = Experiment [8])

based maximum principal stress criterion. A mesh convergence analysis is carried out to study the variation of estimated crack-initiation load with respect to r_0. In hybrid-polygon-quadtree mesh, r_0 is roughly proportional to the size of the cracked element. Therefore, a lower value of r_0 creates higher mesh density near the crack-tip. Fig. 6.25 shows the variation of normalised crack initiation load ($P_{I,N}$) with r_0. Here, the normalised crack initiation load is shown so that all curves can be fitted in a single diagram. It can be observed that for each initial crack interface angle (α), as the value of r_0 decreases, the estimated normalised crack initiation load tends to converge to a value near to that obtained from experiments. In order to improve the rate of convergence, a higher order fitting of stresses in the crack-tip region can be utilised to efficiently capture the high stress gradients in this region.

Fig. 6.26a shows the definition of wing crack initiation angle (θ) used by [8] in their experiments. The wing crack initiation angle is measured at ~ 12.5 mm from the initial crack-tip (C') relative to initial crack interface angle (α). Fig. 6.26b shows the variation of θ with increasing value of α. In the present work, θ is obtained using stress-



(a) Definition of the wing crack initiation angle θ

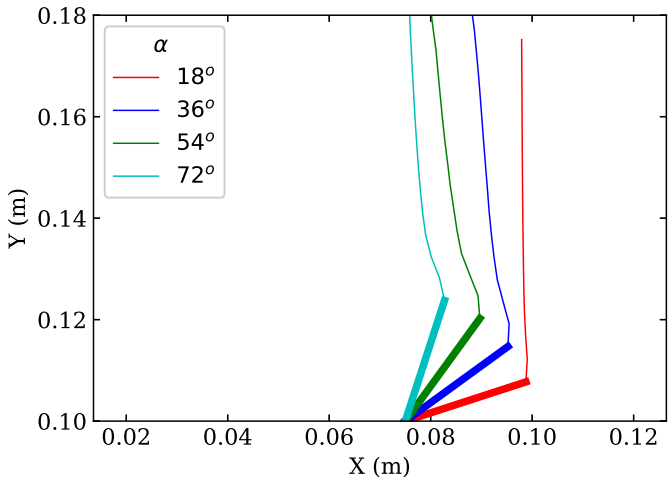
(b) Variation of θ with inital interface crack angle α

Figure 6.26: Wing crack initiation angle θ (Experiment: [8])

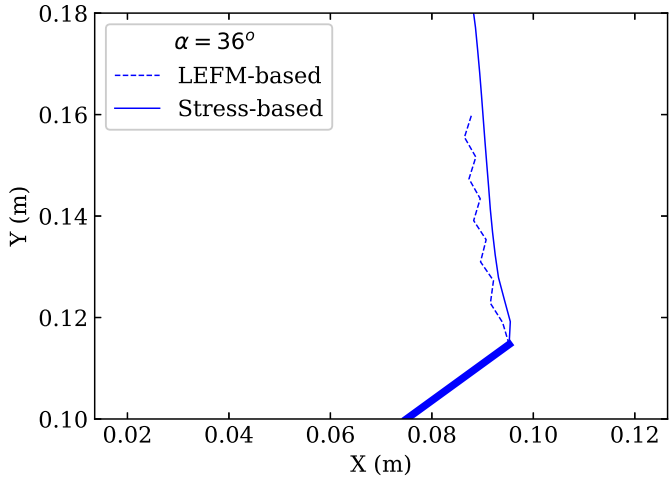
based maximum principal stress criterion (as explained in Section 6.5), which shows a good agreement with those obtained from the experiment and LEFM based maximum principal stress criterion.

Crack propagation stage: After the wing crack initiates, it propagates into the matrix in the direction of the applied uniaxial compressive load. Fig. 6.27a shows the predicted wing crack paths for increasing value of initial crack interface angle (α). The direction of wing crack propagation in the matrix is determined using the stress-based maximum principal stress criterion as explained in Section 6.5. The crack propagation length is kept at 4.5 mm. Fig. 6.27b shows the wing crack path obtained using stress based is smooth and without any zig-zag oscillations unlike those obtained from LEFM based maximum principal stress criterion [387, 392]. This avoids any convergence issues associated with the cohesive tractions applied along the wing crack paths.

Fig. 6.28a shows the applied compressive load (P) vs CMSD curve for $\alpha = 54°$. It can be observed that the result obtained in the present work shows similar trend as those obtained experimentally by [8]. When the applied compressive load is increased,



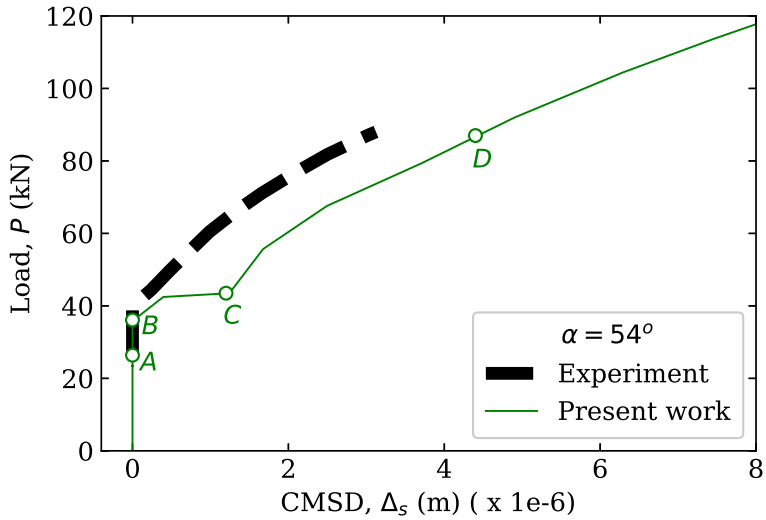
(a) Predicted wing crack paths using stress based criterion for various values of α



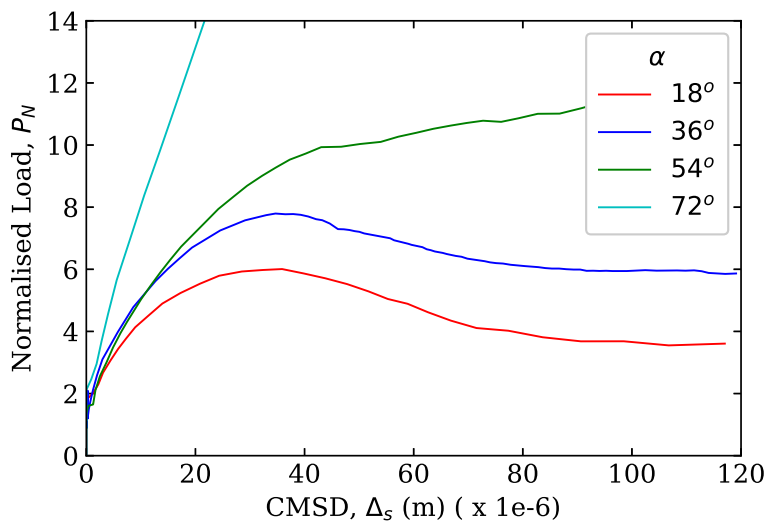
(b) Comparison of the crack paths predicted by stress based vs LEFM based criteria for $\alpha = 36°$

Figure 6.27: Wing crack paths

the debonding at initial crack interface takes place at point A. As the load is further increased, the wing-crack initiation takes place at point B and a jump in CMSD is observed along the path BC. Along CD, the wing crack starts propagating into the matrix. It is observed that that slope of CD is similar to the experimentally observed value. In Fig. 6.28b, normalised load (P_N) vs CMSD curve is shown for various values



(a) Load vs CMSD curve for $\alpha = 54^o$

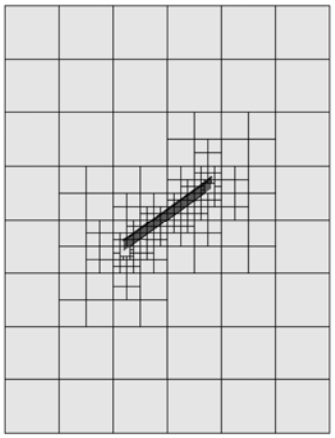


(b) Normalised load vs CMSD curves

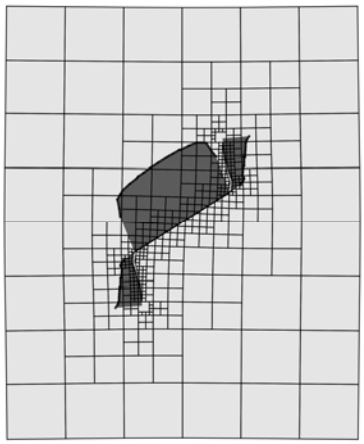
Figure 6.28: Load-CMSD curves for the specimen (Experiment: [8])

of α. For $\alpha = 18°$ and $36°$, as the CMSD increases, a softening behaviour of the load curve is observed, followed by a relatively flat region due to the residual frictional forces on the initial crack interface.

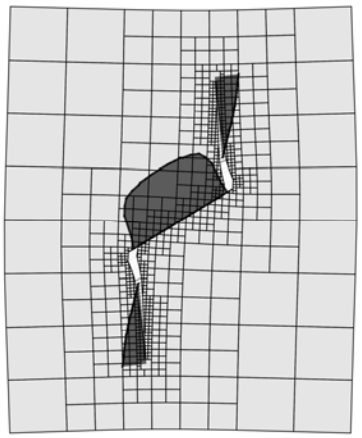
Fig. 6.29 shows the various stages of wing crack propagation in the specimen model with $\alpha = 36°$. It is observed that in the pre-debonding stage, the stick condition exists along the initial crack interface where the direction of contact traction is parallel to the



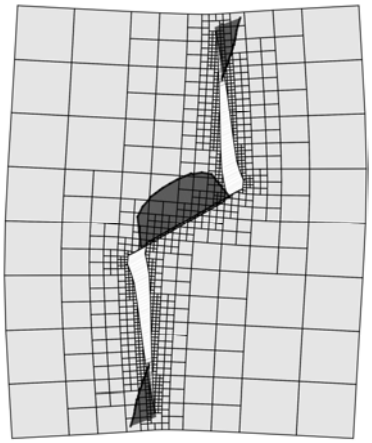
(a) P = 22.9 kN, Δ_s = 0.00 × 10^{-6} m (pre-debonding)



(b) P = 199.4 kN, Δ_s = 24.5 × 10^{-6} m (pre-peak)



(c) P = 192.5 kN, Δ_s = 55.7 × 10^{-6} m (post-peak)



(d) P = 164.2 kN, Δ_s = 102.5 × 10^{-6} m (post-peak)

Figure 6.29: Various stages of wing crack propagation for specimen model for $\alpha = 36°$ (deformation scale = 100)

applied compressive load as given in Fig. 6.29a. The pre-peak and post-peak stages are shown in Fig. 6.29b-d after the crack-propagates into the matrix. The distribution of the contact traction magnitude along the initial crack interface CC' with increasing CMSD (Δ_s) is shown in Fig. 6.30. The length of CC' is $2a = 0.05$ m. It is observed that at lower values of CMSD, the contact traction is almost uniformly distributed over the CC'. As the CMSD increases, the magnitude of the contact traction increases near the centre of CC' and decreases towards the initial crack-tips (C or C'). Thus, the

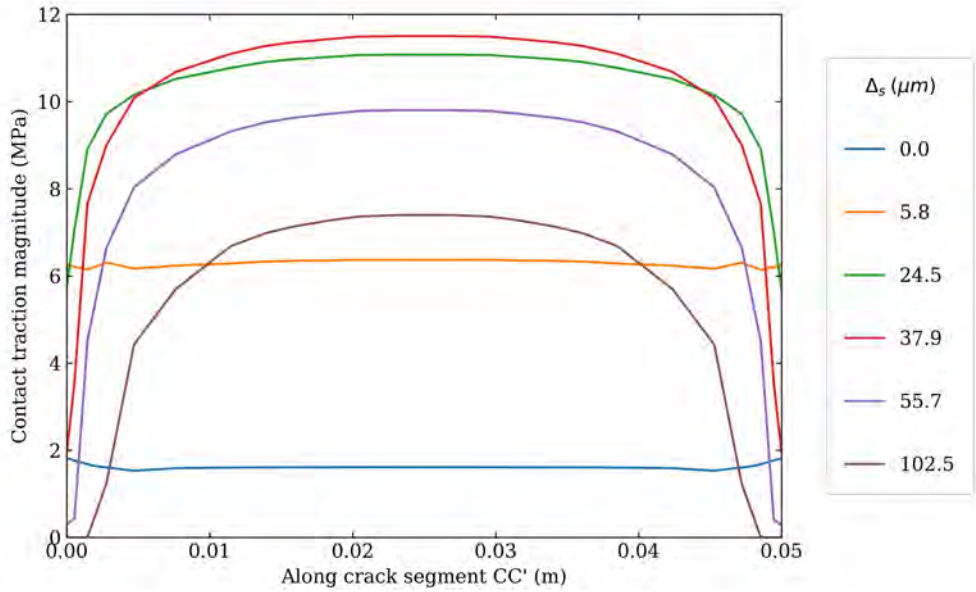


Figure 6.30: Variation of magnitude of contact traction along the initial crack interface with increasing CMSD (Δ_s) for $\alpha = 36°$

distribution becomes more rounded with the peak near the centre of CC'. With further increase in CMSD, the contact traction near the initial crack-tips vanishes. The peak value of the contact traction decreases as well.

6.7 Summary

In the present chapter, a cohesive/contact interface formulation is developed for solving the compressive fracture problems in the framework of SBFEM in two dimensions. The crack surface tractions are modelled by inserting 4-noded zero-thickness interface elements along the crack segments. The contact condition is implemented using penalty-stiffness method as it allows robust and seamless integration with the cohesive crack models. To handle the stress field near the crack-tip, a sideface-traction approach is developed, where linearised tractions are applied on the sidefaces of the cracked elements. These linearised sideface tractions are estimated using a novel sideface traction update scheme. The scheme allows the crack-tip singularities to vanish when the equilibrium

position is reached at each load step.

From the mesh-sensitivity analysis of the plate model with a crack, it is showed that the developed formulation accurately estimates stress and displacement fields during the crack closure, especially in the crack-tip region with large-sized cracked elements. For solving cohesive crack propagation problems, the crack propagation condition and directions are estimated using the stress-based criterion with minimal computational effort. It is because the stresses in the crack-tip region can be directly extracted from the formulation considering the effect of cohesive and frictional-contact tractions. Using benchmark example of mixed-mode cohesive crack propagation, it is shown that the load-displacement curves and the crack paths are captured with sufficient accuracy even for a coarse crack-tip mesh. The application of the proposed method is shown through practical examples. Simulation of the Brazilian split test showed that the developed formulation could effectively capture the size-effect in concrete. Further, the method is utilised to simulate compressive fracture analysis in concrete involving interfacial debonding and matrix-cracking phenomena. A linear-fit of the contact tractions on the sidefaces of the cracked element predicts initiation load and angle of the wing cracks with sufficient accuracy.

Chapter 7

An octree-based parallel explicit solver for impact-induced fracture analysis

In this chapter, the cohesive/contact interface and the octree-based explicit formulations (developed previously) are combined to carry out a massively parallel impact fracture analysis in three dimensions. This chapter is organised as follows.

Section 7.1 gives a general introduction of impact-fracture problems. In Section 7.2, the formulation of cohesive interfaces for mixed-mode loading and contact conditions is briefly described. In Section 7.3, the interface element formulation is developed, where the element matrices are derived and unique patterns are described. The implementation of the explicit solver and internal force vector calculation using interface element patterns is discussed in Section 7.4. The numerical examples for contact-impact and cohesive-fracture under impact loading are presented in Section 7.5. A summary of this chapter is discussed in Section 7.6.

7.1 Introduction

The impact-induced fracture analysis, as introduced in Chapter 1, is critical in many engineering applications, where the structures are exposed to impact by foreign objects.

Examples include composite laminates used in aircraft structures subjected to collision with runway debris, the sandwich panels under the impact of dropped or flying objects, etc. These composite materials have relatively weak behaviour under localised impact. A common energy absorption mechanism (due to impact loading) is the creation of a large fracture area, especially at the weaker interfaces, e.g., delamination of the facesheet-core interface in sandwich panels [393–395]. Such phenomena can trigger additional damage mechanisms and may cause a significant loss in structural integrity and load-carrying capacity.

The impact events can be categorised as low-velocity and high-velocity impacts [40–42]. Under the low-velocity impact loading, the time of contact between the projectile and target is much longer; therefore, the transmitted kinetic energy is accommodated well beyond the point of impact. In the case of the high-velocity impact loading, a highly localised response in the target is induced by the projectile, which results in energy dissipating over a much smaller region.

The simulation of high-velocity impact-induced fracture phenomena in composites is a fairly complicated process. It requires modelling instantaneous contact between the impacting bodies, impact-induced wave propagation, and delamination at the weak interfaces. Though core/matrix crushing and fracturing are also common, they are not modelled in the present work. Within the framework of finite-element based methods, cohesive interface element modelling has been extensively used for capturing fracture and delamination under impact loading [43–47]. These elements are based on the cohesive zone approach, originally proposed by Dugdale and Barenblatt [48, 49]. A traction-separation relation is utilised, which allows modelling the onset of delamination using the strength-based criteria and capturing the propagation using the fracture energy-based criteria. The main advantage of the cohesive elements is the ability to predict both the initiation and propagation of delamination without previous knowledge of the crack location and propagation direction [50, 51]. Further, these interface elements

can also model the contact condition using the penalty-stiffness based approach. A detailed review of the computational modelling of the processes associated with the impact-induced fracture analysis is provided in Section 2.3.

In general, impact-induced cohesive fracture/delamination simulations require a fine mesh discretisation to capture the non-linear behaviour arising from instantaneous contact-condition and cohesive-crack propagation. Additionally, capturing the realistic response of structures with complicated shapes typically requires three-dimensional modelling, which can lead to a large number of unknowns (or DOFs) to solve. Developing an efficient local mesh refinement procedure and parallel solver can therefore help obtain an accurate solution in a feasible time.

In the present work, the octree-based parallel explicit solver developed in Chapter 4 is extended for performing the impact-induced cohesive-fracture simulation. The octree mesh structure is exploited to carry out efficient local-mesh refinement near the interfaces as well as capture complex geometrical shapes. Further, similar to the octree-based formulation, unique interface element patterns are identified, which are compatible with the faces of octree patterns. The master-element matrices of each pattern are precomputed and consume only a small amount of memory. The element matrix of the individual interface elements can be obtained using simple scalar multiplication with the corresponding master-element matrix. The internal force vector is computed from the contributions of elastic strain in octree-cells and cohesive-separations in the interface elements using element group-wise operation. This approach reduces the memory footprint of the explicit solver, enables efficient calculation of the nodal force vectors and supports a massively parallel computation with thousands of cores in a distributed computing environment.

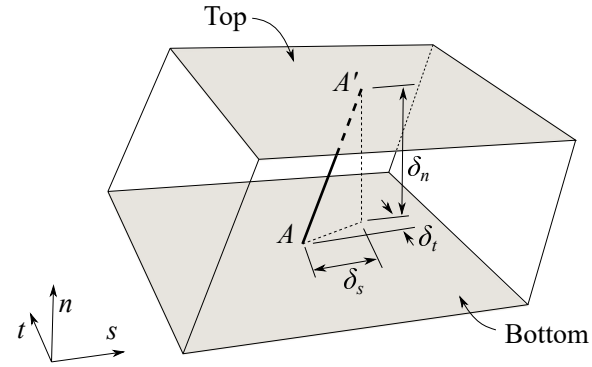


Figure 7.1: Separation of top and bottom surfaces of the cohesive interface

7.2 Cohesive interface formulation

The cohesive interface is modelled using the two surfaces (top and bottom), as shown in Fig. 7.1. Each point on one of these surfaces (say A) has a corresponding point on the other surface (A'). The relative displacement or separation δ between the pair of points is transformed into the local coordinate system defined by (s, t, n). Thus, the separation of points can be expressed as an opening mode (I) along $n-$direction and a shear mode (II or III) in $s-t$ plane. To simulate the decohesion process, a traction-separation constitutive relation is employed to implement the cohesive tractions as a function of the separation of each pair of points at the interface. Depending on the loading conditions, the decohesion can take place in either single-mode or mixed-mode. The decohesion formulation has been described in detail by Camanho and Carlos [50, 51], and is briefly explained here for convenience.

7.2.1 Single-mode loading

A bi-linear traction-separation constitutive relation is shown in Fig 7.2. A high value of penalty (or initial) stiffness is used (K_n, K_s or K_t) which holds the top and bottom faces of the interface element together in the linear elastic range. In pure mode loading, the onset of damage happens when interfacial normal (t_n) or shear tractions (t_s or t_t) attain

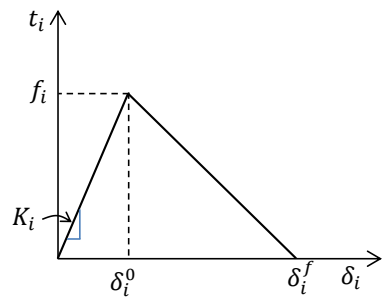


Figure 7.2: Bi-linear traction-separation constitutive law in pure modes, $i \in (s, t, n)$

their respective allowable tensile (f_n) or shear strengths (f_s or f_t). The components of the onset separation δ^0 in the three directions are given by

$$\delta_n^0 = \frac{f_n}{K_n},\ \delta_s^0 = \frac{f_s}{K_s},\ \delta_t^0 = \frac{f_t}{K_t}. \tag{7.1}$$

The fracture toughness G^c in pure modes can be obtained as the area under the traction-separation curve shown in Fig 7.2 and are given by

$$G_n^c = \int_0^{\delta_n^f} t_n d\delta_n = \frac{1}{2} K_n \delta_n^0 \delta_n^f, \tag{7.2}$$

$$G_s^c = \int_0^{\delta_s^f} t_s d\delta_s = \frac{1}{2} K_s \delta_s^0 \delta_s^f, \tag{7.3}$$

$$G_t^c = \int_0^{\delta_t^f} t_t d\delta_t = \frac{1}{2} K_t \delta_t^0 \delta_t^f. \tag{7.4}$$

The cohesive formulation under single-mode loading conditions can be obtained as the special case of the mixed-mode loading explained in the next section.

7.2.2 Mixed-mode loading

In this section, we derive the formulation for decohesion under mixed-mode loading. A coupled damage cohesive-zone model [50, 51, 396, 397] is adopted, which requires the criteria for both the onset of softening and full decohesion. The evolution of the cohesive interface under mixed-mode loading conditions is controlled by a single damage variable.

Definitions

The mixed-mode relative displacement or separation δ_M can be expressed as

$$\delta_M = \sqrt{\langle \delta_n \rangle^2 + \delta_T^2}, \tag{7.5}$$

where, $\delta_T = \sqrt{\delta_s^2 + \delta_t^2}$ is the shear/tangential component and δ_n is the normal component of the mixed-mode separation. For $\delta_n > 0$, a displacement based mode-mixity ratio β can be defined as

$$\beta = \frac{\delta_T}{\delta_n}. \tag{7.6}$$

Solving Eqs. (7.5) and (7.6), we obtain

$$\delta_n = \frac{\delta_M}{\sqrt{1+\beta^2}}, \qquad \delta_T = \frac{\beta \delta_M}{\sqrt{1+\beta^2}}. \tag{7.7}$$

It can be noticed that for pure mode-I loading, $\beta = 0$ and for pure mode-II (or III) loading, $\beta \to \infty$.

Onset of mixed-mode softening

In the case of pure mode loading, the onset of softening (or damage) at the interface can simply be captured by comparing the traction components with their respective allowables. However, under mixed-mode loading conditions, the onset of damage and the softening behaviour can occur much before any of the traction components involved reach their respective allowables. Therefore, the effect of the interaction of the traction components on the onset of delamination or softening is captured using a mixed-mode criterion such as the quadratic-failure criterion [50, 51, 398] given by

$$\left(\frac{\langle t_n \rangle}{f_n}\right)^2 + \left(\frac{t_s}{f_s}\right)^2 + \left(\frac{t_t}{f_t}\right)^2 = 1, \tag{7.8}$$

where, t_s and t_t are the tangential components and t_n is the normal component of the cohesive tractions at the interface. The Macaulay operator $\langle\rangle$ implies that negative normal component of the tractions do not contribute to the mixed-mode effects. f_s, f_t and f_n are the allowable stresses in the corresponding directions.

We define the penalty parameters for the three components of the cohesive tractions as K_s, K_t and K_n. In the present work, the tangential interface properties are assumed isotropic, that is, $K_s = K_t = K_T$ and $f_s = f_t = f_T$. Therefore, the pure-mode separations at the onset of softening are obtained as

$$\delta_n^0 = \frac{f_n}{K_n},\ \delta_s^0 = \delta_t^0 = \delta_T^0 = \frac{f_T}{K_T}. \tag{7.9}$$

Using Eqs. (7.5), (7.6), (7.8) and (7.9), the mixed-mode separation at the onset of softening δ_M^0 is obtained as

$$\delta_M^0 = \begin{cases} \delta_n^0 \delta_T^0 \sqrt{\frac{(1+\beta^2)}{\left(\delta_T^0\right)^2 + (\beta\delta_n^0)^2}} & \text{if } \delta_n > 0 \\ \delta_T^0 & \text{otherwise} \end{cases}. \tag{7.10}$$

In case of pure mode-I loading, $\delta_M^0 = \delta_n^0$ as $\beta = 0$. For pure shear-mode loading (mode-II or III), $\delta_M^0 = \delta_T^0$ as $\beta \to \infty$ (or $\delta_n = 0$). Similar to Eq. (7.7), the components of δ_M^0 are obtained as

$$\delta_{nM}^0 = \frac{\delta_M^0}{\sqrt{1+\beta^2}}, \qquad \delta_{TM}^0 = \frac{\beta\delta_M^0}{\sqrt{1+\beta^2}}. \tag{7.11}$$

Mixed-mode decohesion

The most commonly used criteria to predict the decohesion/delamination propagation under mixed-mode loading is the power law criterion [50, 51, 399, 400], which is described in terms of energy release rates (G_n, G_s and G_t) and the corresponding fracture toughness (G_n^c, G_s^c and G_t^c). It can be expressed as

$$\left(\frac{G_n}{G_n^c}\right)^a + \left(\frac{G_s}{G_s^c}\right)^a + \left(\frac{G_t}{G_t^c}\right)^a = 1 \tag{7.12}$$

Here, a is fitting parameters and is chosen as $a = 1$. As the shear interface properties are assumed isotropic, $G_s^c = G_t^c = G_T^c$. Further, the concept of energy release rates with shear loading, $G_T = G_s + G_t$, is adopted [50, 401]. Under these assumptions, Eq. (7.12) can be re-written as

$$\frac{G_n}{G_n^c} + \frac{G_T}{G_T^c} = 1 \tag{7.13}$$

The energy release rates for a bi-linear traction-separation law under mixed-mode conditions can be given as

$$G_n = \int_0^{\delta_{nM}^f} t_n d\delta_n = \frac{1}{2} K_n \delta_{nM}^0 \delta_{nM}^f \tag{7.14a}$$

$$G_s = \int_0^{\delta_{sM}^f} t_s d\delta_s = \frac{1}{2} K_T \delta_{sM}^0 \delta_{sM}^f \tag{7.14b}$$

$$G_t = \int_0^{\delta_{tM}^f} t_t d\delta_t = \frac{1}{2} K_T \delta_{tM}^0 \delta_{tM}^f \tag{7.14c}$$

where δ_{nM}^f, δ_{sM}^f and δ_{tM}^f are the components of the separation δ_M^f under mixed-mode loading corresponding to total decohesion. From Eqs. (7.2.2), (7.13), and (7.14), δ_M^f can be obtained as

$$\delta_M^f = \begin{cases} \frac{2(1+\beta^2)}{\delta_M^0} \left(\frac{K_n}{G_n^c} + \frac{\beta^2 K_T}{G_T^c} \right)^{-1}, & \text{if } \delta_n > 0 \\ \delta_T^f & \text{otherwise} \end{cases} \tag{7.15}$$

Damage evolution

The damage evolution is described using a single scalar damage variable D that is defined as

$$D = \frac{\delta_M^f \left(\delta_M - \delta_M^0 \right)}{\delta_M \left(\delta_M^f - \delta_M^0 \right)} \tag{7.16}$$

In order to capture irreversible loading/unloading at time step j, the maximum value of damage variable is stored as

$$D = \max_{j \leq J} (D(j)) \tag{7.17}$$

The interfacial stresses can then be obtained as

$$t_i = \begin{cases} K_i \delta_i & \text{if } \delta_M \leq \delta_M^0 \\ (1-D) K_i \delta_i & \text{if } \delta_M^0 < \delta_M \leq \delta_M^f \\ 0 & \text{if } \delta_M > \delta_M^f \end{cases}, \tag{7.18}$$

where, $i \in (s, t, n)$.

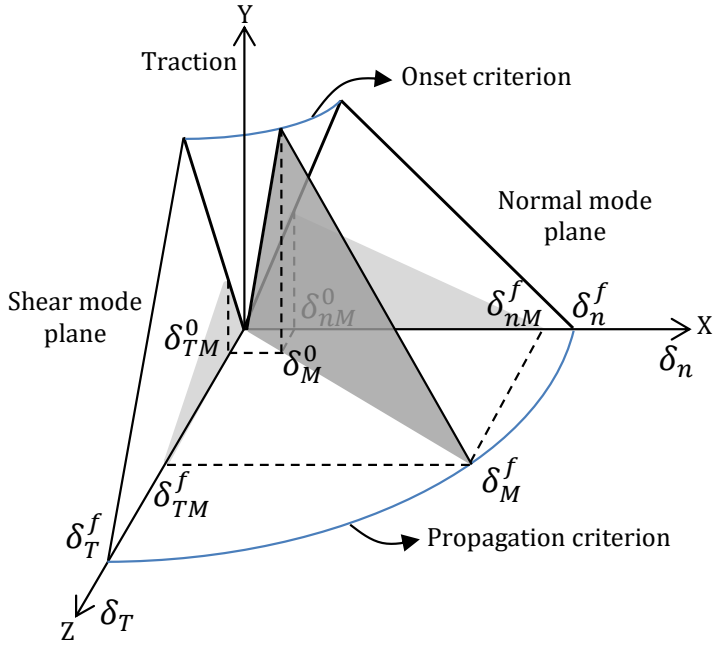


Figure 7.3: Bi-linear traction-separation constitutive law in mixed mode

The mixed-mode softening law is illustrated in the three-dimensional map, shown in Fig 7.3. The pure normal mode is represented by the XY plane, and the pure shear mode is represented by the YZ plane. Any point on the XZ plane represents a mixed-mode separation. The mixed-mode softening law is instantaneous in nature, that is, its value varies with time as the damage evolves. Further, it also varies at each point along the interface depending on the damage level. An instantaneous state of the interface during the mixed-mode loading is marked by the grey regions in the XY and YZ planes. Due to the mixed-mode effect, the peak allowable stresses at the onset of damage reduces with respect to the corresponding single-mode values. Further, the value of the separation at complete decohesion of the interface is also reduced.

7.2.3 Contact formulation

As the separation of top and bottom interfaces becomes zero, the contact condition needs to be satisfied. In this section, we briefly explain the general contact formulation.

We consider the motion of two deformable bodies Ω_1 and Ω_2 and select $\Gamma_1 \in \Omega_1$ and $\Gamma_2 \in \Omega_2$ as prospective contact points over a given time interval. For any point $X_1 \in \Gamma_1$ and $X_2 \in \Gamma_2$, the normal contact constraints enforced are given by

$$\delta_n \geq 0, \quad t_n \leq 0, \quad t_n \delta_n = 0 \ . \tag{7.19}$$

Here, δ_n is the normal gap and t_n is the normal contact traction between the contacting bodies. The governing constraints form the classical Kuhn-Tucker conditions in mathematical optimisation problem. In physical terms, it implies that when the contact force is nonzero, the gap should be zero and vice-versa.

The contact constraints can be effectively satisfied using the computational methods such as penalty-stiffness and Lagrange multipliers methods [25]. In the present work, the penalty-stiffness method is chosen as it is robust and easy to implement. The normal contact traction t_n is computed as

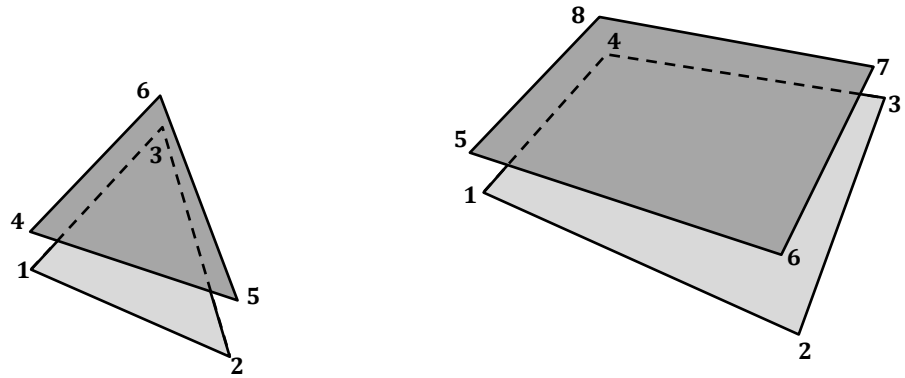
$$t_n = \begin{cases} K_n \delta_n, & \text{if } \delta_n \leq 0 \\ 0, & \text{otherwise} \end{cases} \ . \tag{7.20}$$

Here, K_n is normal penalty parameter and the contact traction t_n is proportional to the depth of penetration $|\delta_n|$. The benefit of this method is that it does not add any additional variables to the final equation of motion and the system matrix is positive definite. However, the contact condition is only approximately satisfied since a small amount of penetration is introduced at the contact surface.

The tangential contact behaviour is captured using the Coulomb friction model. It is expressed as

$$\mathbf{t}_T \leq (\mu |t_n| + c) \frac{\dot{\boldsymbol{\delta}}_T}{\dot{\delta}_T}, \tag{7.21}$$

where, $\dot{\boldsymbol{\delta}}_T$ is the slip-rate vector, $\dot{\delta}_T \left(= ||\dot{\boldsymbol{\delta}}_T|| \right)$ is the slip-rate magnitude, $\mathbf{t}_T$ is the tangential contact traction vector in the direction of slip $(\dot{\boldsymbol{\delta}}_T / \dot{\delta}_T)$, μ is the coefficient of friction and c is adhesion parameter. The stick-slip conditions are modelled by



(a) Six-node element (b) Eight-node element

Figure 7.4: Linear interface elements

penalising the tangential traction and are given by

$$\mathbf{t}_T = \begin{cases} K_T \delta_T \frac{\dot{\boldsymbol{\delta}}_T}{\dot{\delta}_T}, & \text{if } K_T \delta_T < \mu |t_n| + c \, (\text{Stick} - \text{condition}) \\ (\mu |t_n| + c) \frac{\dot{\boldsymbol{\delta}}_T}{\dot{\delta}_T}, & \text{if } K_T \delta_T \geq \mu |t_n| + c \, (\text{Slip} - \text{condition}) \end{cases}, \tag{7.22}$$

where, K_T is tangential penalty parameter and δ_T is the slip magnitude.

7.3 Interface elements

In this work, the contact/cohesive surface is discretised using the interface elements. Each interface element is modelled using a pair of 2D surface elements. The six and eight node linear interface elements are shown in Fig. 7.4, which are constructed using a pair of triangular and quadrilateral elements, respectively.

7.3.1 Derivation of stiffness matrix

In this section, we derive the expression for the stiffness matrix of interface elements using the principle of virtual work.

The displacement field $\tilde{\mathbf{u}}(s,t)$ of the each surface element (top or bottom) with m

nodes can be obtained as

$$\tilde{\mathbf{u}}(s,t) = \begin{Bmatrix} u(s,t) \\ v(s,t) \\ w(s,t) \end{Bmatrix} = \begin{bmatrix} N_1 & 0 & 0 & N_2 & ... & 0 & 0 \\ 0 & N_1 & 0 & 0 & ... & N_m & 0 \\ 0 & 0 & N_1 & 0 & ... & 0 & N_m \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ w_1 \\ u_2 \\ ... \\ v_m \\ w_m \end{Bmatrix} \tag{7.23}$$

$$= \begin{bmatrix} N_1\mathbf{I} & N_2\mathbf{I} & N_3\mathbf{I} & ... & N_m\mathbf{I} \end{bmatrix} \hat{\mathbf{u}}_0$$

$$= \tilde{\mathbf{N}}(s,t)\hat{\mathbf{u}}_0$$

Here, $N_i(s,t)\ \forall\ i \in (1,2\ldots m)$ are the shape functions of the 2D surface element ($m = 3$ for triangular element and $m = 4$ for quadrilateral element) constructed as in the finite element methods. This ensures compatibility of the interface elements with the surface of the octree-based polyhedral elements [402]. $\hat{\mathbf{u}}_0$ is the nodal displacement vector of the surface element. Let $\tilde{\mathbf{x}}^{\text{init}}(s,t) = \tilde{\mathbf{N}}(s,t)\hat{\mathbf{x}}_0$ represent the initial position of any given point on the surface element. The separation vector field $\boldsymbol{\delta}(s,t) = \begin{bmatrix} \delta_s & \delta_t & \delta_n \end{bmatrix}^T$ between a pair of surface elements (top and bottom) can be given by

$$\boldsymbol{\delta}(s,t) = \left(\tilde{\mathbf{x}}^{\text{init}}(s,t) + \tilde{\mathbf{u}}(s,t)\right)_{\text{top}} - \left(\tilde{\mathbf{x}}^{\text{init}}(s,t) + \tilde{\mathbf{u}}(s,t)\right)_{\text{bottom}}$$

$$= \begin{bmatrix} -\tilde{\mathbf{N}}(s,t) & \tilde{\mathbf{N}}(s,t) \end{bmatrix} \begin{Bmatrix} \left(\hat{\mathbf{x}}_0^{\text{init}} + \hat{\mathbf{u}}_0\right)_{\text{bottom}} \\ \left(\hat{\mathbf{x}}_0^{\text{init}} + \hat{\mathbf{u}}_0\right)_{\text{top}} \end{Bmatrix}, \tag{7.24}$$

$$= \mathbf{N}_f(s,t)\left(\hat{\mathbf{x}}^{\text{init}} + \hat{\mathbf{u}}\right)$$

$$= \mathbf{N}_f(s,t)\hat{\mathbf{d}}$$

where,

$$\hat{\mathbf{d}} = \hat{\mathbf{x}}^{\text{init}} + \hat{\mathbf{u}}. \tag{7.25}$$

Here, $\hat{\mathbf{d}}$ is the nodal displacement vector of the interface element. We define a traction-

separation relation between the surface elements, given by

$$\mathbf{t}(s,t) = \begin{Bmatrix} t_s(s,t) \\ t_t(s,t) \\ t_n(s,t) \end{Bmatrix} = \begin{Bmatrix} c_s(s,t) \\ c_t(s,t) \\ c_n(s,t) \end{Bmatrix} + \begin{bmatrix} k_{ss} & k_{st} & k_{sn} \\ k_{ts} & k_{tt} & k_{tn} \\ k_{ns} & k_{nt} & k_{nn} \end{bmatrix} \begin{Bmatrix} \delta_s(s,t) \\ \delta_t(s,t) \\ \delta_n(s,t) \end{Bmatrix} \tag{7.26}$$
$$= \tilde{\mathbf{c}}(s,t) + \tilde{\mathbf{K}}(s,t)\boldsymbol{\delta}(s,t)$$

Here, $\tilde{\mathbf{K}}$ is the interface constitutive matrix and $\tilde{\mathbf{c}}$ is the surface adhesion vector. The value of interface constitutive matrix $\tilde{\mathbf{K}}$ is obtained from Eqs. (7.20) and (7.22) to implement the contact condition, and Eq. (7.18) to simulate the decohesion. From Eqs. (7.24) and (7.26),

$$\mathbf{t} = \tilde{\mathbf{c}} + \tilde{\mathbf{K}}\mathbf{N}_f\hat{\mathbf{d}} \tag{7.27}$$

Here, the field variables (s,t) are not shown for brevity. The stiffness matrix of the interface element can be derived using the principle of virtual work as explained here. From the definition of virtual work principle,

$$\Delta W^{\text{ext}} = \Delta W^{\text{int}} \tag{7.28}$$

where, ΔW^{ext} and ΔW^{int} are the virtual work due to external and internal forces given by

$$\Delta W^{\text{ext}} = \Delta\hat{\mathbf{u}}^T\hat{\mathbf{f}}^{\text{ext}} \tag{7.29}$$

$$\Delta W^{\text{int}} = \iint_A \Delta\boldsymbol{\delta}^T\mathbf{t}dA = \iint_A \Delta\hat{\mathbf{d}}^T\mathbf{N}_f^T\left(\tilde{\mathbf{c}} + \tilde{\mathbf{K}}\mathbf{N}_f\hat{\mathbf{d}}\right)dA \tag{7.30}$$

From Eq. (7.25) , $\Delta\hat{\mathbf{d}} = \Delta\hat{\mathbf{u}}$. Thus, Eq. (7.30) can re-written as

$$\Delta W^{\text{int}} = \Delta\hat{\mathbf{u}}^T\iint_A \mathbf{N}_f^T\tilde{\mathbf{c}}dA + \Delta\hat{\mathbf{u}}^T\iint_A \mathbf{N}_f^T\tilde{\mathbf{K}}\mathbf{N}_f\hat{\mathbf{d}}dA \tag{7.31}$$

From Eqs. (7.29) and (7.31)

$$\hat{\mathbf{f}}^{\text{ext}} = \iint_A \mathbf{N}_f^T\tilde{\mathbf{c}}dA + \iint_A \mathbf{N}_f^T\tilde{\mathbf{K}}\mathbf{N}_f dA\hat{\mathbf{d}} = \mathbf{c}_f^e + \mathbf{K}_f^e\hat{\mathbf{d}} \tag{7.32}$$

Here, $\mathbf{c}_f^e$ is the adhesion vector (assumed zero in this work) and $\mathbf{K}_f^e$ is the stiffness

matrix of the interface element. The infinitesimal surface area dA can be defined as

$$dA = |\mathbf{J}|\, dsdt \tag{7.33}$$

where, $\mathbf{J}$ is the Jacobian matrix.

The integration of a given function $\mathbf{g}$ can be evaluated numerically over the interface element area A^e using the Gauss quadrature [11, 16, 403]. For a 6-noded linear interface element, it is given by

$$\iint_A \mathbf{g} dA = \frac{1}{2} \sum_{i=1}^{N_g} H_i \mathbf{g}_i \left|\mathbf{J}_i\right|, \tag{7.34}$$

For an 8-noded interface element, the integration is numerically computed as

$$\iint_A \mathbf{g} dA = \sum_{i=1}^{N_g} \sum_{k=1}^{N_g} H_{ik} \mathbf{g}_{ik} \left|\mathbf{J}_{ik}\right| \tag{7.35}$$

Here, N_g is the number of Gaussian integration points used within the interface element and H is the Gaussian weight for integration. For a triangular element, $|\mathbf{J}_i| = 2A^e$ and for a rectangular element, $|\mathbf{J}_{ik}| = A^e/4$.

We express the numerical integration of the stiffness matrix in a generalised notation as

$$\mathbf{K}_f^e = A^e \sum_{i=1}^{N_g} h_i \left(\mathbf{N}_f^T \tilde{\mathbf{K}} \mathbf{N}_f\right)_i = A^e \sum_{i=1}^{N_g} \check{\mathbf{K}}_i, \tag{7.36}$$

where, h_i is the relevant Gaussian weight and $\check{\mathbf{K}}_i$ is the stiffness matrix computed at the ith Gaussian point.

7.3.2 Unique interface element patterns

The problem domain is discretised using the balanced octree mesh. For smooth mesh size transition, the face discretisation of the octree cells is restricted to 6 unique node arrangements considering rotations, as used previously in Chapter 4. These are shown again in Fig. 7.5 for clarity. These arrangements depend on the number of mid-edge nodes n_{h} and their locations. It can be observed that only two unique element pat-

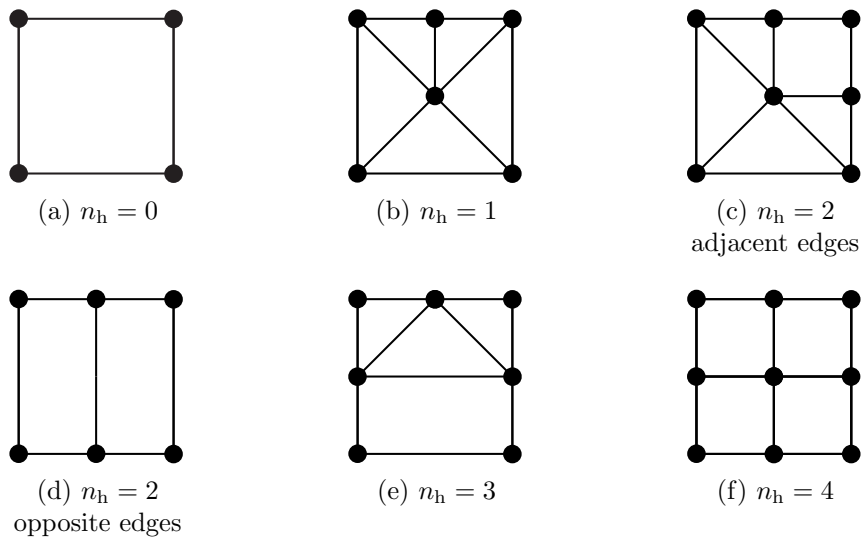


Figure 7.5: Face discretisation patterns for an octree cell with the number of hanging nodes on the edges of each face indicated by n_h.

terns are required for face discretisation: the rectangular and isosceles right-triangle elements. Therefore, the interface elements have only two unique patterns (called the master interface elements), which are compatible with the surface of the octree cells. The element matrices of each master interface element can be pre-computed and stored with little memory requirements. Then, the element matrix of all the interface elements in the octree mesh can be calculated using simple scalar multiplication with the corresponding master interface element matrix. This approach reduces the memory footprint of the solver and enables an efficient calculation of nodal force vectors, as explained in Section 7.4.

7.4 Explicit solver for cohesive-fracture simulation

In this chapter, the explicit time integration scheme is utilised to solve the cohesive fracture problem under impact loading. The governing equilibrium equation of the

structural system can be given as

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{f}^{\text{int}} = \mathbf{f}^{\text{ext}}, \tag{7.37}$$

where, $\mathbf{M}$ is the discrete mass matrix, $\mathbf{C}$ is the viscous damping matrix, $\mathbf{f}^{\text{int}}$ is the internal force vector due to element stresses, and $\mathbf{f}^{\text{ext}}$ is the external load vector. $\mathbf{u}$, $\dot{\mathbf{u}}$, and $\ddot{\mathbf{u}}$ are the nodal displacement, velocity, and acceleration vectors, respectively.

The central difference method (CDM) is chosen for the explicit time integration of Eq. (7.37). The details of CDM are covered in Chapter 4. At the $(j+1)$-th time step, the undamped system of equation can be written as

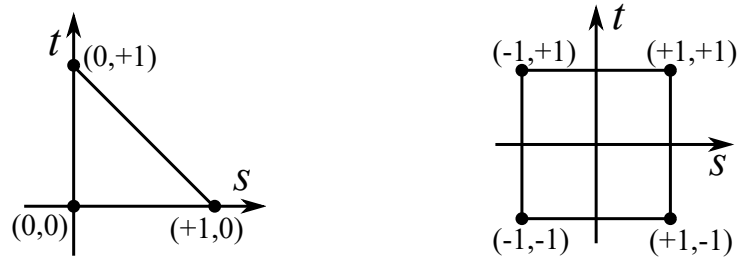
$$\mathbf{u}_{j+1} = 2\mathbf{u}_j - \mathbf{u}_{j-1} + \Delta t^2 \mathbf{M}^{-1}\left(\mathbf{f}_j^{\text{ext}} - \mathbf{f}_j^{\text{int}}\right). \tag{7.38}$$

It is observed from Eq. (7.38) that the displacement at the $(j+1)$-th time step can be obtained from the displacement and force vectors computed in the previous time steps. This avoids the requirement of the iterative solution procedure and therefore, the complicated non-linear problems can be handled without any convergence issues. A diagonal mass matrix ($\mathbf{M}$=$\mathbf{M}_{\text{diag}}$) obtained by the mass lumping technique explained in Section 4.3.2 is utilised, making the inversion of the mass matrix trivial. The internal force vector $\mathbf{f}_j^{\text{int}}$ can be divided into two parts: The elastic and the cohesive. The elastic force vector is computed using the octree-based PBP approach for efficient matrix multiplication, as explained in Section 4.3.3. For the calculation of the cohesive force vector, the unique interface element patterns are utilised, as explained in the next section.

7.4.1 Implementing interface element patterns

For each unique interface element pattern p, we can define a master interface element, as illustrated in Fig. (7.6). Each master element has predefined geometric and interfacial properties.

To consider the directional variation in cohesive stiffness and contact condition



(a) A triangular element

(b) A rectangular element

Figure 7.6: Master interface element patterns

within each interface element, the master interface element stiffness matrix $\mathbf{K}_f^p$ is divided into three components, denoted as $\mathbf{K}_{f,l}^p$ for $l \in (s, t, n)$. From Eq. (7.32), these can be computed as

$$\mathbf{K}_{f,l}^p = \iint_A \mathbf{N}_f^T \tilde{\mathbf{K}}_l \mathbf{N}_f dA \tag{7.39}$$

Here, $\tilde{\mathbf{K}}_l$ is the interface constitutive matrix for $l \in (s, t, n)$, expressed as

$$\tilde{\mathbf{K}}_s = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tilde{\mathbf{K}}_t = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tilde{\mathbf{K}}_n = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.40}$$

Additionally, to consider the spatial variation in cohesive stiffness and contact condition within the interface element, the components of the stiffness matrix defined in Eq.(7.39) can be numerically computed at the ith Gaussian point as

$$\check{\mathbf{K}}_{li}^p = h_i \left(\mathbf{N}_f^T \tilde{\mathbf{K}}_l \mathbf{N}_f \right)_i, \tag{7.41}$$

where, h_i is the Gaussian weight, as given in the generalised notation of the Gaussian integration in Eq. (7.36). The various components of the stiffness matrix ($\check{\mathbf{K}}_{li}^p$) can be pre-computed and stored with little memory requirement. For an interface element e with given size and damage properties at time step j, the stiffness matrix $\check{\mathbf{K}}_{lij}^e$ can be obtained by scaling the corresponding master interface element matrix $\check{\mathbf{K}}_{li}^p$ by a suitable factor, as explained below.

The stiffness matrix of a given interface element $\check{\mathbf{K}}^e_{lij}$ is proportional to its surface area A^e and damaged stiffness K^D_{lij}. It can be derived from the master element stiffness matrix $\check{\mathbf{K}}^p_{li}$ as

$$\check{\mathbf{K}}^e_{lij} = A^e K^D_{lij} \mathbf{K}^{\rho l}_i = w^{e\rho l}_{ij} \check{\mathbf{K}}^p_{li}. \tag{7.42}$$

Here, the tangential component of the damaged stiffness K^D_{lij} for $l \in (s,t)$ is given as

$$K^D_{lij} = (1 - D_{ij}) K_l, \tag{7.43}$$

and the normal component (for $l = n$) is given by

$$K^D_{lij} = \begin{cases} (1 - D_{ij}) K_l, & \text{if } \delta_{nij} \geq 0 \\ K_l, & \text{otherwise} \end{cases}, \tag{7.44}$$

where, K_l for $l \in (s,t,n)$ is the penalty stiffness introduced in Section 7.2.1, D_{ij} is the mixed-mode scalar damage variable formulated in Section 7.2.2 and δ_{nij} is the opening gap in the interface element.

Applying the pattern-by-pattern (PBP) approach to the interface elements, the nodal cohesive force vectors can be calculated as

$$\mathbf{f}^{\text{int}}_j = \sum_{p=1}^{N_P} \mathbf{F}^p_j = \sum_{p=1}^{N_P} \sum_{i=1}^{N_g} \sum_{l}^{s,t,n} \check{\mathbf{K}}^p_{li} \left[\begin{array}{cccc} w^{1p}_{lij} \mathbf{u}^{1p}_j, & w^{2p}_{lij} \mathbf{u}^{2p}_j & \cdots & w^{N_E p}_{lij} \mathbf{u}^{N_E p}_j \end{array} \right]. \tag{7.45}$$

Here, $N_P\,(= 2)$ is the number of unique patterns of the interface element, N_g is the number of Gaussian points within the interface element and N_E is the number of elements of pattern p. For a given element e of pattern p, w^{ep}_{lij} is the stiffness weight obtained in Eq (7.42) and $\mathbf{u}^{ep}_j$ is the nodal displacement vector at at time step j.

Similar to the procedure followed in Chapter 4, the internal force matrix $\mathbf{F}^p_j$ is computed using dot product operation and assembled into the internal force vector $\mathbf{f}^{\text{int}}_j$ using the bincount operation. A much better cache utilisation can be achieved as most of the work is recast to perform pattern-level dense matrix computations. This improves the operational intensity of the matrix operations, providing significant performance gains to the explicit solver, especially on modern supercomputer architectures.

7.4.2 Parallel implementation

The parallel implementation of the octree-based explicit solver is covered in detail in Chapter 4 for solving elastodynamic problems. In this chapter, it is extended for obtaining the parallel solution of the cohesive fracture problems. The parallelisation is achieved using the mesh-partitioning strategy in a distributed computing environment. The problem domain mesh, containing both the octree and interface elements, is divided into a set of smaller partitions and assigned to the individual workers. The workers run a single computer program and carry out simultaneous local computations over the assigned partitions. The internal force vector is updated with the contributions from both the elastic and interfacial (cohesive/contact) force vectors. At each time step, the damage/contact property of the interfacial elements is calculated at all the Gaussian points, and the stiffness weight vector is updated, as explained in the previous section. Next, the data among the neighbouring partitions are synchronised through the message passing algorithm, following which the displacement vector at the current time step is locally computed. The procedure for measuring the CPU time and scaling properties of the parallel computation is covered in Section 4.4.3.

7.5 Numerical examples

In this section, a few numerical examples are presented to verify and demonstrate the efficacy of octree-based implementation for performing impact-induced fracture analysis.

First, the contact-impact problem is simulated to show the advantage of local-mesh refinement using octree meshes. Then, cohesive fracture formulation is verified using the cantilever beam pair example. Finally, to demonstrate the massively parallel implementation of the explicit solver, an impact-induced delamination process is simulated for a large-scale sandwich panel.

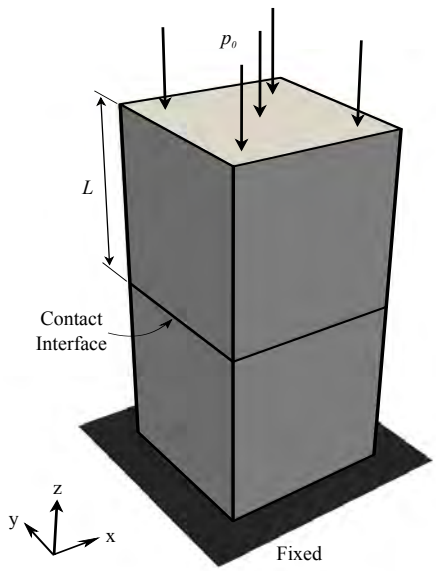


Figure 7.7: Cubes under normal static contact

7.5.1 Cubes under normal contact

First, we verify the normal contact modelling using a pair of cubes. A normal static contact problem is studied, followed by the normal contact-impact analysis.

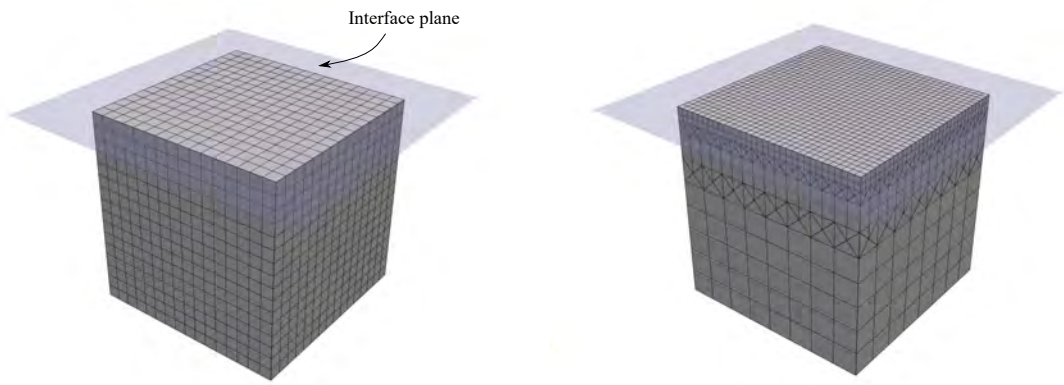
Static contact

We consider two cubes of side length L and in contact with each other, as shown in Fig. 7.7. The top surface of the upper cube is subjected to static compressive pressure. The bottom surface of the lower cube is constrained against displacement. The material properties are: Young's modulus $E = 1\,\text{GPa}$ and Poisson's ratio $\nu = 0$. The analytical displacement solution (u, v, w) is given by

$$u = v = 0,\ w = \frac{p_0 z}{E} \tag{7.46}$$

Each cube is discretised using hexahedral and octree meshes, as shown in Fig. 7.8. In the octree meshes, the contact surface is refined locally, where the mesh size transition is done upto two octree-levels.

The contact is modelled using the penalty stiffness approach. Choosing an appro-



(a) A hexahedral mesh (b) An octree mesh

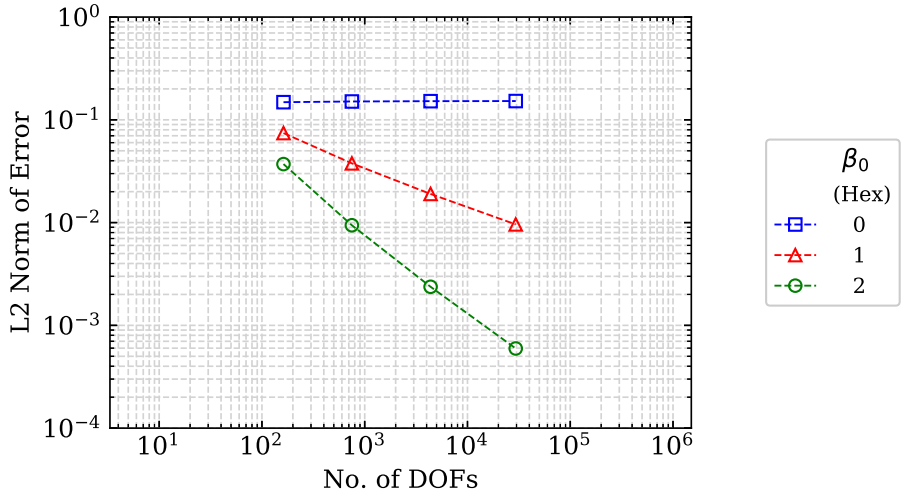
Figure 7.8: Meshes for the cube with interface plane

priate value of the penalty stiffness can be challenging. In general, it is a function of the mesh-size as well as the material of the contacting bodies [25, 269]. Therefore, we can define the normal penalty stiffness K_n as

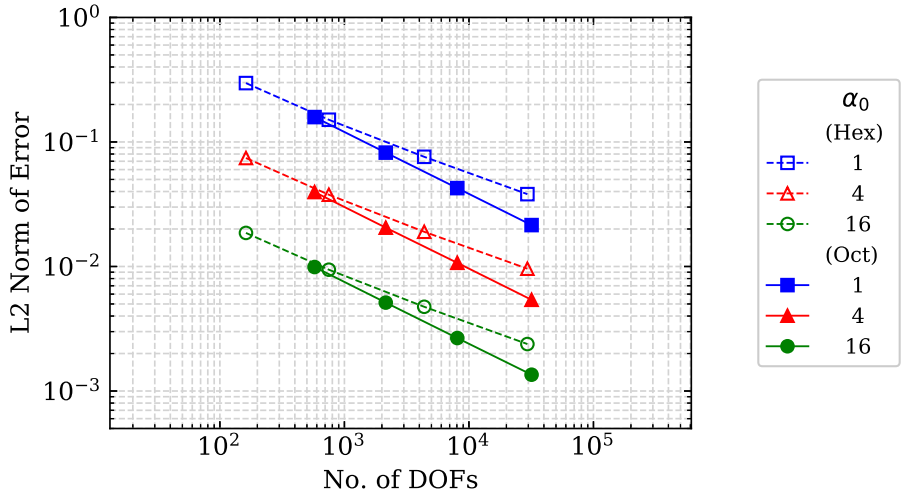
$$K_n = \frac{\alpha_0 E}{L_e^{\beta_0}}. \tag{7.47}$$

Here, L_e is the length of interface element. α_0 and β_0 are the penalty control parameters.

The static-contact problem is solved using the Newton-based solver. A convergence study is done to analyse the effect of mesh-type, mesh-size and penalty parameters on the accuracy of the displacement solution. Fig. 7.9a shows the effect of parameter β_0 on the accuracy of the solution obtained for hexahedral mesh and $\alpha_0 = 4$. At $\beta_0 = 0$, it is observed that the accuracy of the solution does not improve with finer discretisation as the penalty stiffness K_n becomes independent of the interface element length L_e. As the value of β_0 increases, more accurate results are obtained for a given mesh-size. The effect of parameter α_0 is shown in Fig. 7.9b for hexahedral and octree meshes with $\beta_0 = 1$. It is noticed that accuracy of the solution increases as the value of α_0 increases. Further, the octree mesh provides a better rate of convergence as compared to the hexahedral mesh.



(a) Increasing β_0 with α_0=4.0 for hexahedral mesh



(b) Comparision of hexahedral and occtree meshes for increasing α_0 with β_0=1.0

Figure 7.9: Convergence study of the cubes under frictionless static-contact

Contact-impact

We consider a normal frictionless-contact-impact problem, as shown in Fig. 7.10. It consists of two cubes of side length $L = 1\,\mathrm{m}$, having an initial gap $2\delta_n = 2g = 2.0e-4\,\mathrm{m}$ and moving towards each other with equal speed $V_o = 1\,\mathrm{m/s}$. The material is assumed linear-elastic with the following properties: Young's modulus $E = 1\,\mathrm{GPa}$, Poisson's ratio $\nu = 0$ and density $\rho = 10\,\mathrm{kg/m^3}$. The cubes are initially undeformed and the problem is

Table 7.1: Mesh properties for the cube

Mesh name	C_L	No. of Elements	No. of DOFs	L_e^{min} [m]	L_e^{max} [m]
Hex4	–	8,448	29,478	0.0625	0.0625
Hex5	–	66,560	215,622	0.0313	0.0313
Hex6	–	528,384	1,647,750	0.0156	0.0156
Oct65	1	112,640	379,608	0.0156	0.0313
Oct64	2	69,632	246,864	0.0156	0.0625

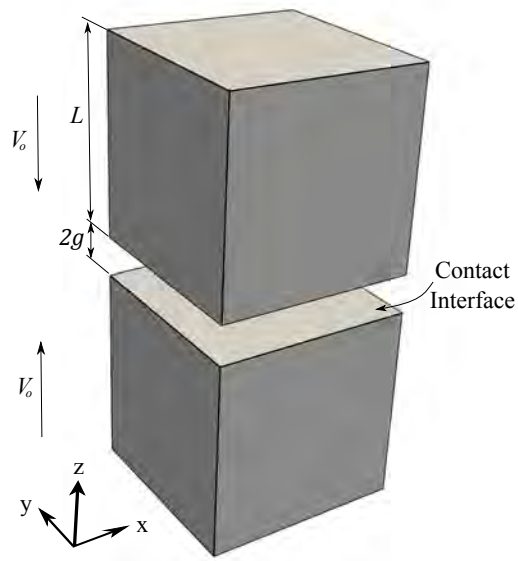


Figure 7.10: Two impacting cubes

symmetric about the point at which the two cubes face impact. The analytical solution for this problem is available from Ref. [9].

The mesh-properties of the simulation model (including two cubes and interface elements) are summarised in Table 7.1. The mesh names are defined in the following way: A hexahedral mesh with element length $L_e = 2^{-i}$ is named as Hex$i$ (i = 4, 5, 6). An octree mesh with the minimum and maximum element lengths as $L_e^{min} = 2^{-i}$ and $L_e^{max} = 2^{-j}$, respectively, is named as Octij (i, j = 4, 5, 6). Here, the contact interfaces are locally refined, where L_e^{min} is equal to the length of the interface elements.

The normal frictionless contact-impact problem is solved using the penalty-stiffness based explicit dynamics formulation. Time integration is done using the central differ-

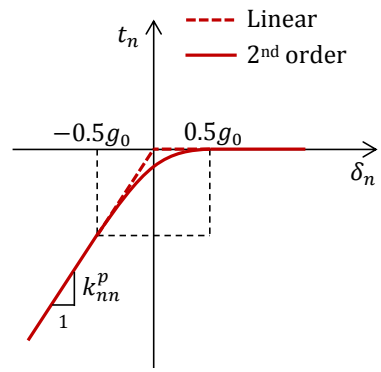


Figure 7.11: Linear vs 2nd order penalty regularisation

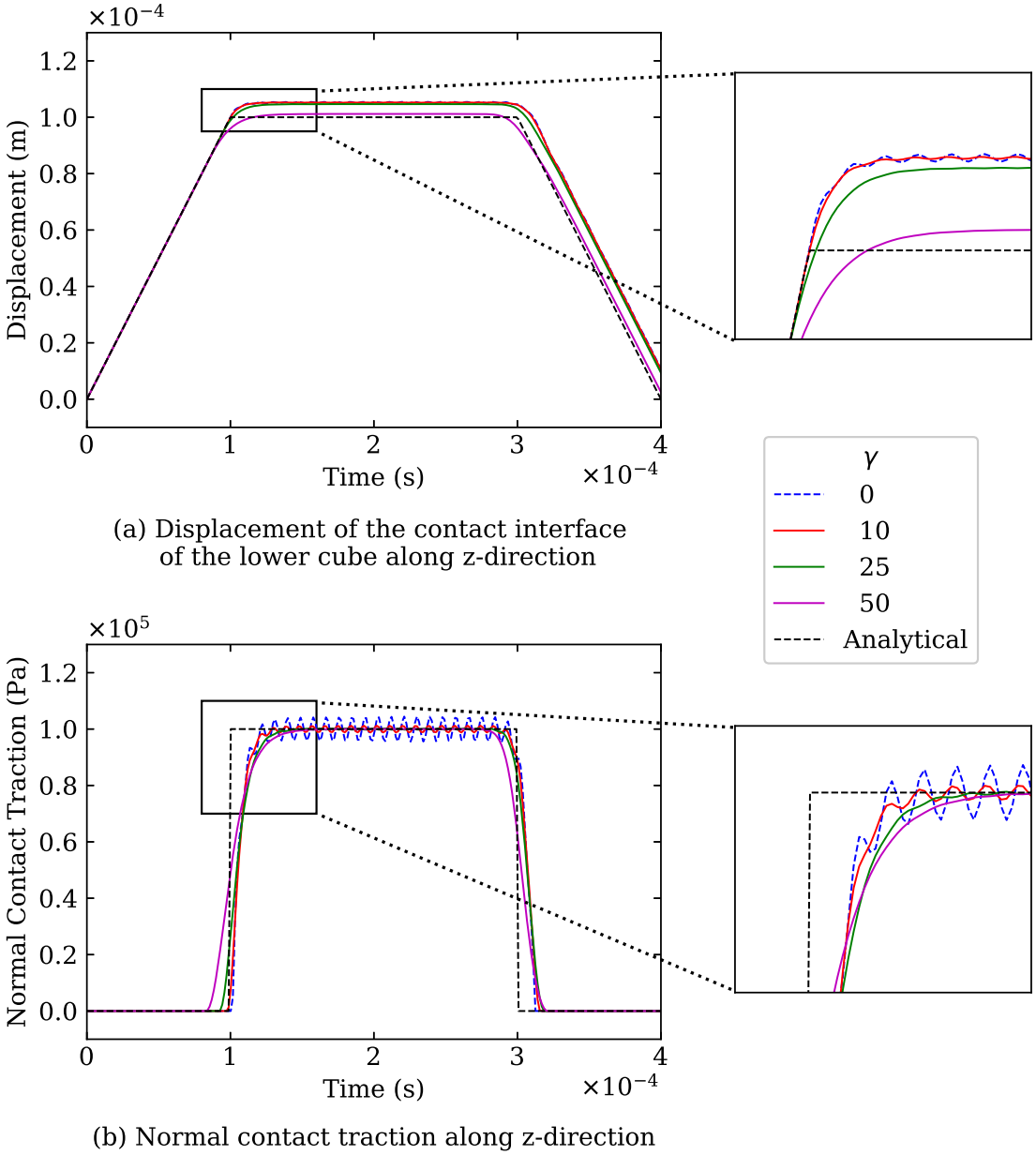
ence scheme, where the time step is chosen as $\Delta t = 0.5\Delta t_{cr}$. Here, Δt_{cr} is the critical time step of the dynamic equilibrium problem without the penalty term.

In quasi-static contact problems, where the relative velocity of the contacting surfaces is negligible, the accuracy of the solution can be improved by simply increasing the values of the penalty parameters, as shown in previous sections. However, in the case of contact-impact problems, the instantaneous change of velocity of the contact interface can induce artificial oscillations in the contacting bodies for the higher value of penalty parameters. A non-linear penalty regularisation [270, 271] has been shown to effectively avoid this issue; therefore, the same is used in the present work. A polynomial function of order q can be defined, which provides C^{q-1} smooth transition between non-contact and contact state [271]. Here, a 2nd order polynomial function is adopted. The expression for normal contact traction t_n is defined by

$$t_n = \begin{cases} K_n \delta_n & \text{if } \delta_n \leq -0.5g_0 \\ -\frac{K_n}{2g_0}\left(\delta_n - 0.5g_0\right)^2 & \text{if } -0.5g_0 < \delta_n \leq 0.5g_0 \\ 0 & \text{if } \delta_n > 0.5g_0 \end{cases} . \tag{7.48}$$

Here, K_n is the normal penalty stiffness given by Eq. (7.47), δ_n is the normal gap between contact interfaces, and g_0 is the reference penetration given as [271]

$$g_0 = \gamma \Delta t_c V_o \tag{7.49}$$



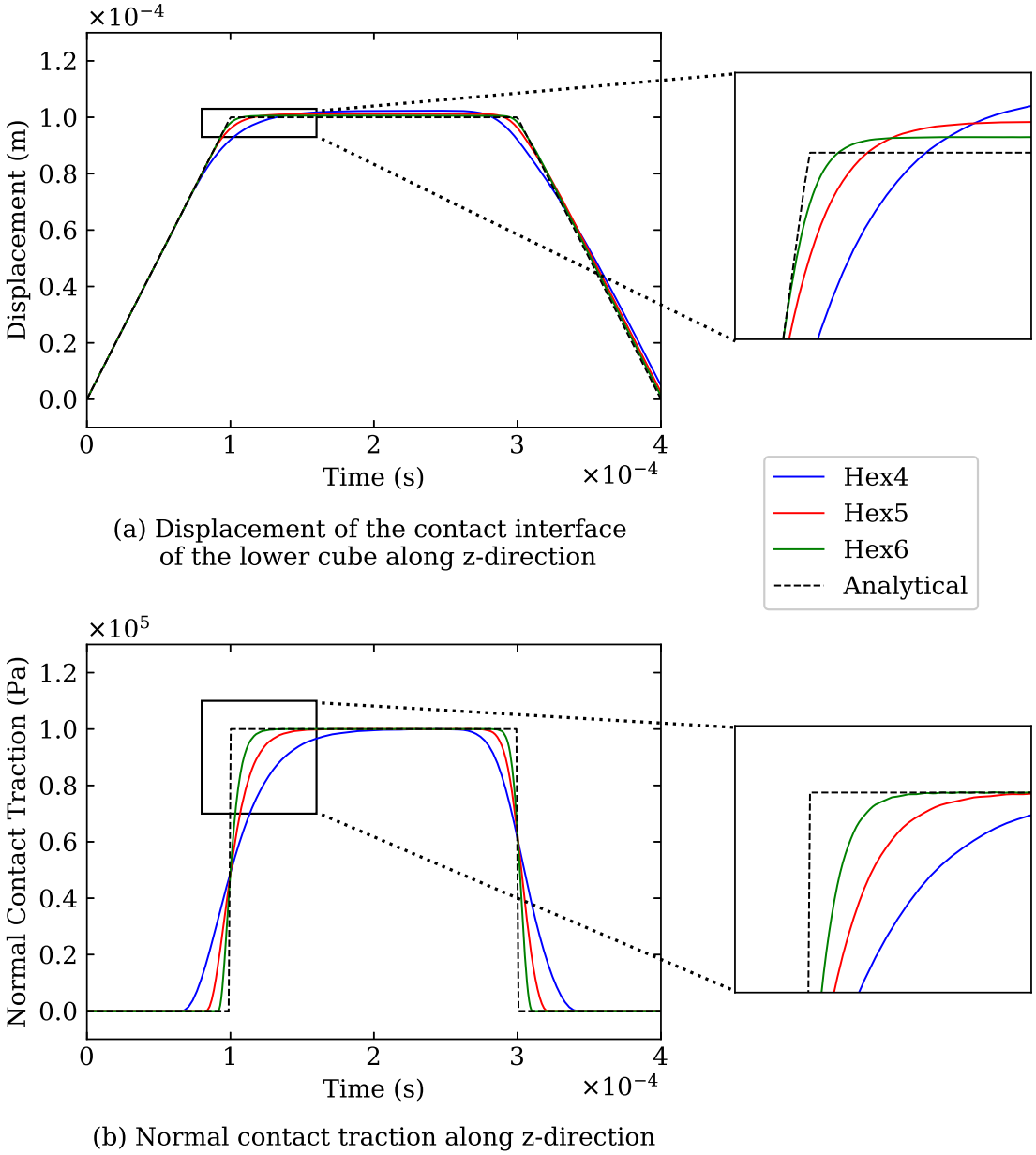
(a) Displacement of the contact interface of the lower cube along z-direction

(b) Normal contact traction along z-direction

Figure 7.12: Results of penalty regularisation of Hex5 mesh (Analytical: [9])

where, γ is a user-defined regularisation parameter. Fig. 7.11 shows the variation of normal contact traction t_n with the normal gap δ_n, obtained for linear and 2nd order penalty regularisation.

For the normal contact-impact simulation of cubes, the following values of penalty

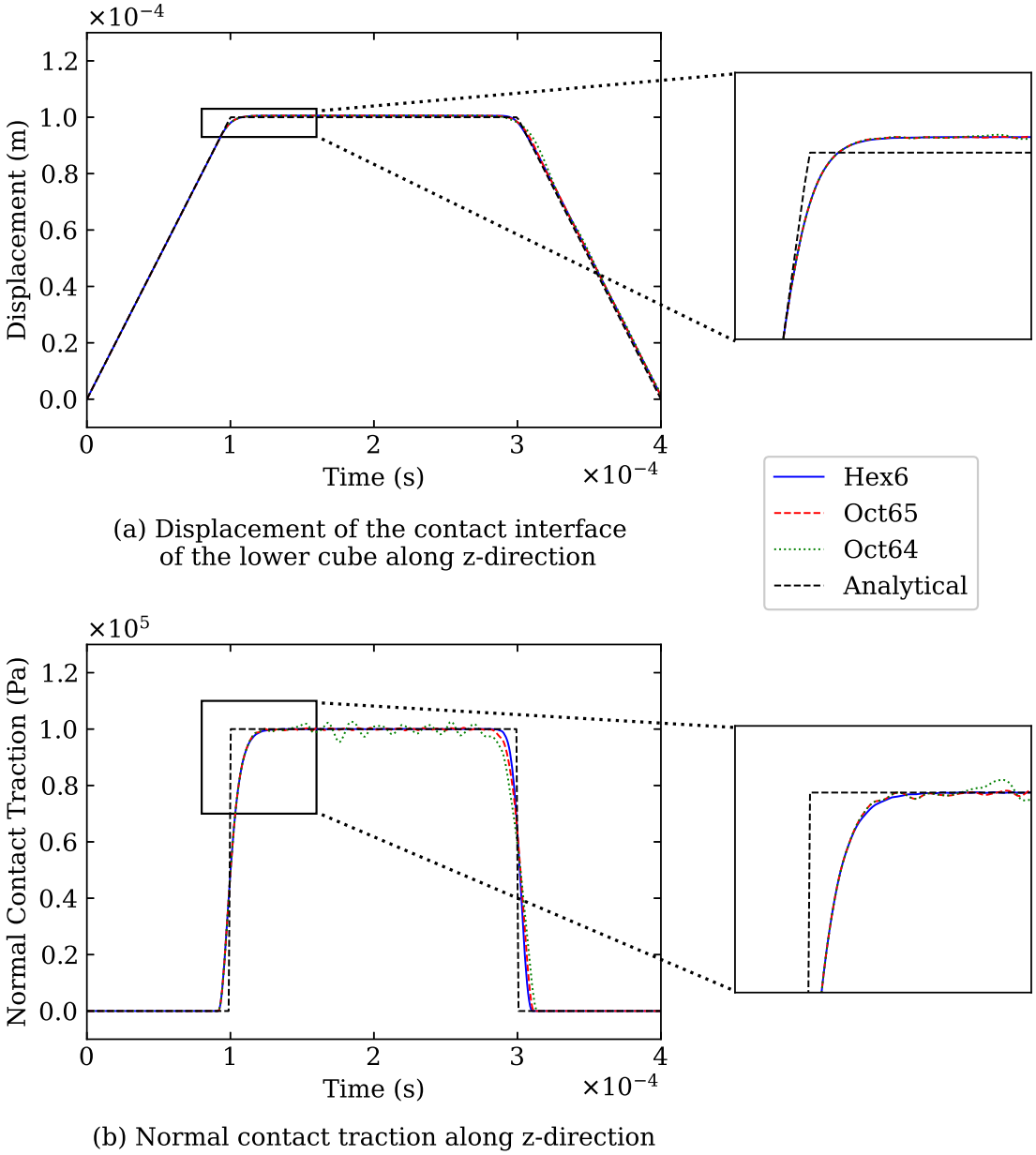


(a) Displacement of the contact interface of the lower cube along z-direction

(b) Normal contact traction along z-direction

Figure 7.13: Convergence study of hexahedral mesh (Analytical: [9])

parameters are chosen: $\alpha_0 = 0.3$ and $\beta_0 = 1.0$. A convergence study is performed to examine the effect of regularisation parameter γ on the accuracy of the solution. Fig. 7.12 shows the results for Hex5 mesh. It is observed that the structural response obtained using the linear regularisation ($\gamma = 0$) is associated with large oscillations.



(a) Displacement of the contact interface of the lower cube along z-direction

(b) Normal contact traction along z-direction

Figure 7.14: Comparision of results obtained with hexahedral and octree meshes (Analytical: [9])

This is due to instantaneous change in the velocity of contact interface during impact. As the value of 2nd order regularisation parameter γ is increased, the oscillations in the structural response is effectively reduced. Based on this study, the value of γ is chosen

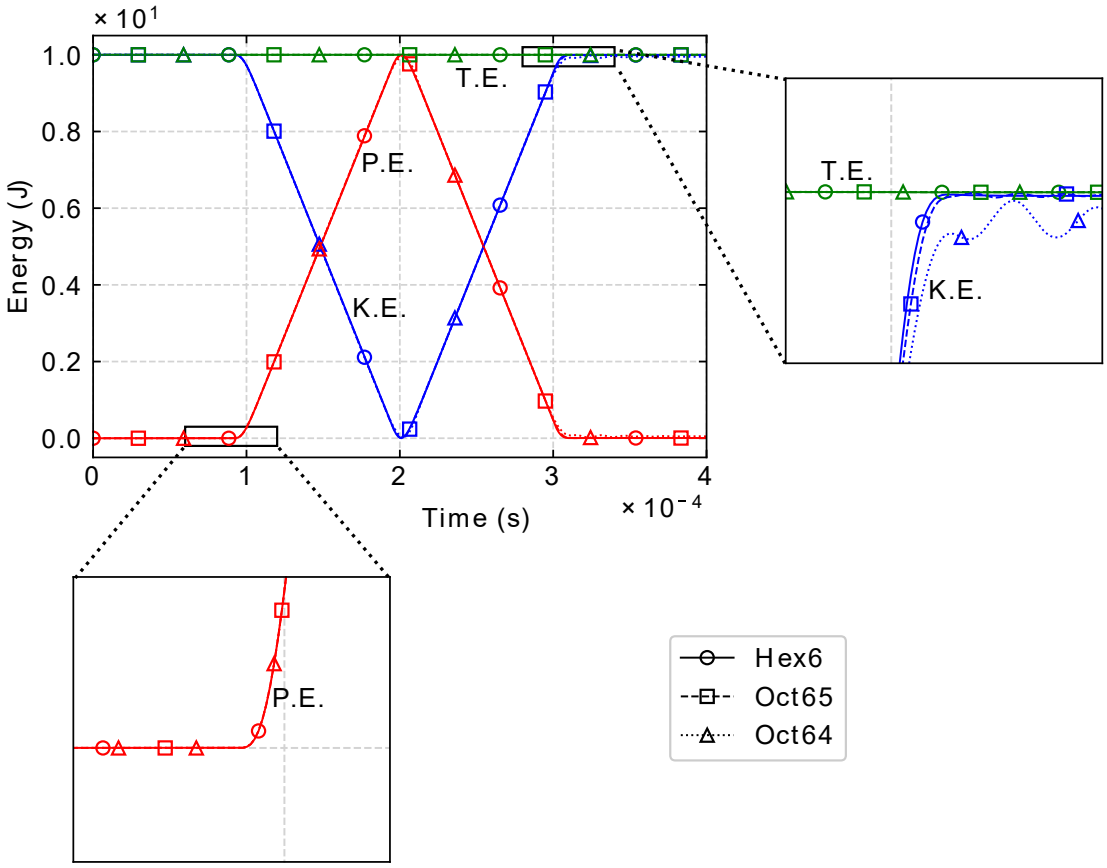


Figure 7.15: Variation in energy with time
(K.E. = Kinetic Energy, P.E. = Potential Energy, T.E. = Total Energy (P.E.+K.E.)

as 50 for performing rest of the simulation in this section.

Fig. 7.13 shows the convergence study of hexahedral mesh using 2nd order regularisation. It is observed that as the mesh-density is increased, the results tend to converge towards the analytical solution. Fig. 7.14 compares the results obtained from the octree meshes with those obtained from the hexahedral mesh. It is observed that the octree mesh with 1-level coarsening (Oct65) produces nearly identical results as the corresponding hexahedral mesh (Hex6) even-though the number of DOFs in the octree mesh is significantly less (Ref: Table 7.1). However, the octree mesh with 2-level coarsening (Oct64) is associated with a relatively larger oscillations.

Fig. 7.15 shows the variation in energy of the cubes with time. It is observed that the total energy (T.E.) is constant throughout the simulation, which implies that energy is conserved in the given framework. For the given problem, the kinetic energy (K.E.) at the beginning of the simulation amounts to T.E. = K.E. $= 2 \times 0.5\rho L^3 V_o^2 = 10\,\mathrm{J}$, which agrees with the value of the total energy obtained during the simulation. It is also observed that the octree mesh with 1-level coarsening (Oct65) produces almost similar results as the hexahedral mesh (Hex6). On the other hand, some oscillations (though small) are observed in Oct64.

7.5.2 Cantilever beam pair

In this section, we verify the present implementation by studying the frictional and cohesive interaction between a pair of cantilever beams.

Frictional contact with slip

We consider a pair of cantilever beams in contact with each other, as shown in Fig. 7.16. The upper beam is subjected to uniformly distributed pressure p_0 in the transverse direction. The material properties are: Young's modulus $E = 1\,\mathrm{GPa}$, density $\rho = 10\,\mathrm{kg/m^3}$, and Poisson's ratio $\nu = 0$. The dimensions of each beam are: $L = 2\,\mathrm{m}$, $B = 0.25\,\mathrm{m}$ and $H = 0.5\,\mathrm{m}$. The origin of the coordinate system is kept at point-O. The interaction at the contact interface is defined using Coulomb friction model with friction coefficient $\mu = 0.25$ and adhesion $c = 0$. Small-sliding is assumed.

The mesh-properties of the model are summarised in Table 7.2. A few meshes (Hex4 and Oct43) for the beam model are shown in Fig. 7.17. It is noticed that the octree mesh is locally refined near the interface plane.

The frictional-contact is modelled using the penalty stiffness approach. The shear penalty stiffnesses K_s and K_t are kept equal to K_n. The penalty parameters are chosen as $\alpha_0 = \beta_0 = 1$. The problem is solved using local-arc length controlled algorithm

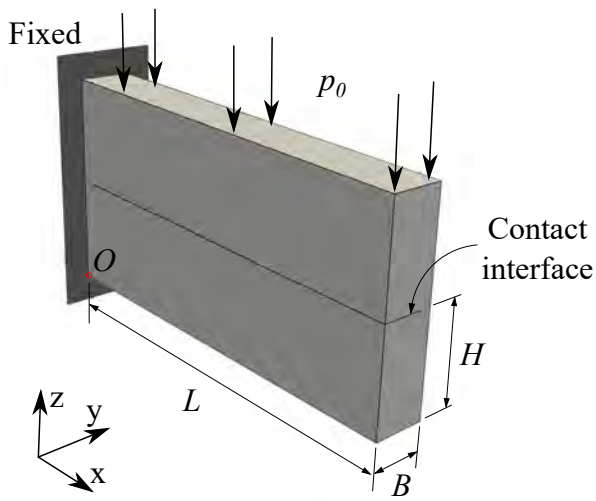
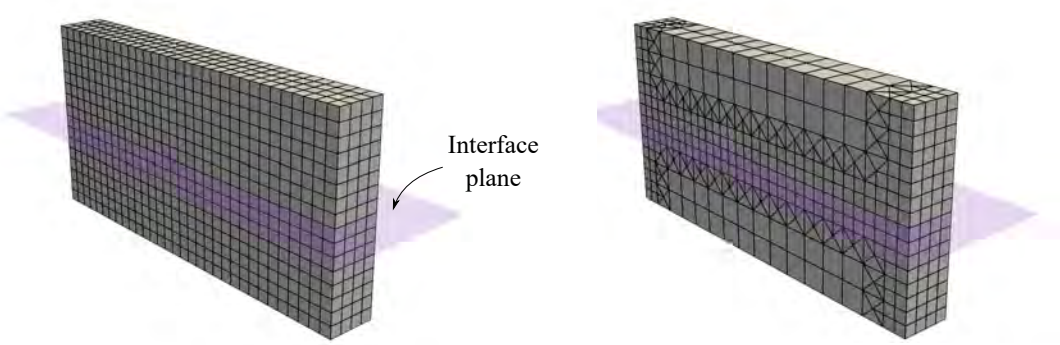


Figure 7.16: A pair of cantilever beams with contact interface



(a) Hex4 (b) Oct43

Figure 7.17: Meshes for the cantilever beam pair

Table 7.2: Mesh properties of the cantilever beam pair

Mesh name	No. of Elements	No. of DOFs	L_e^{min}[m]	L_e^{max}[m]
Hex4	2,176	8,910	0.0625	0.0625
Hex3	288	1,530	0.125	0.125
Oct43	1,000	5,304	0.0625	0.125
Oct42	916	5,082	0.0625	0.25

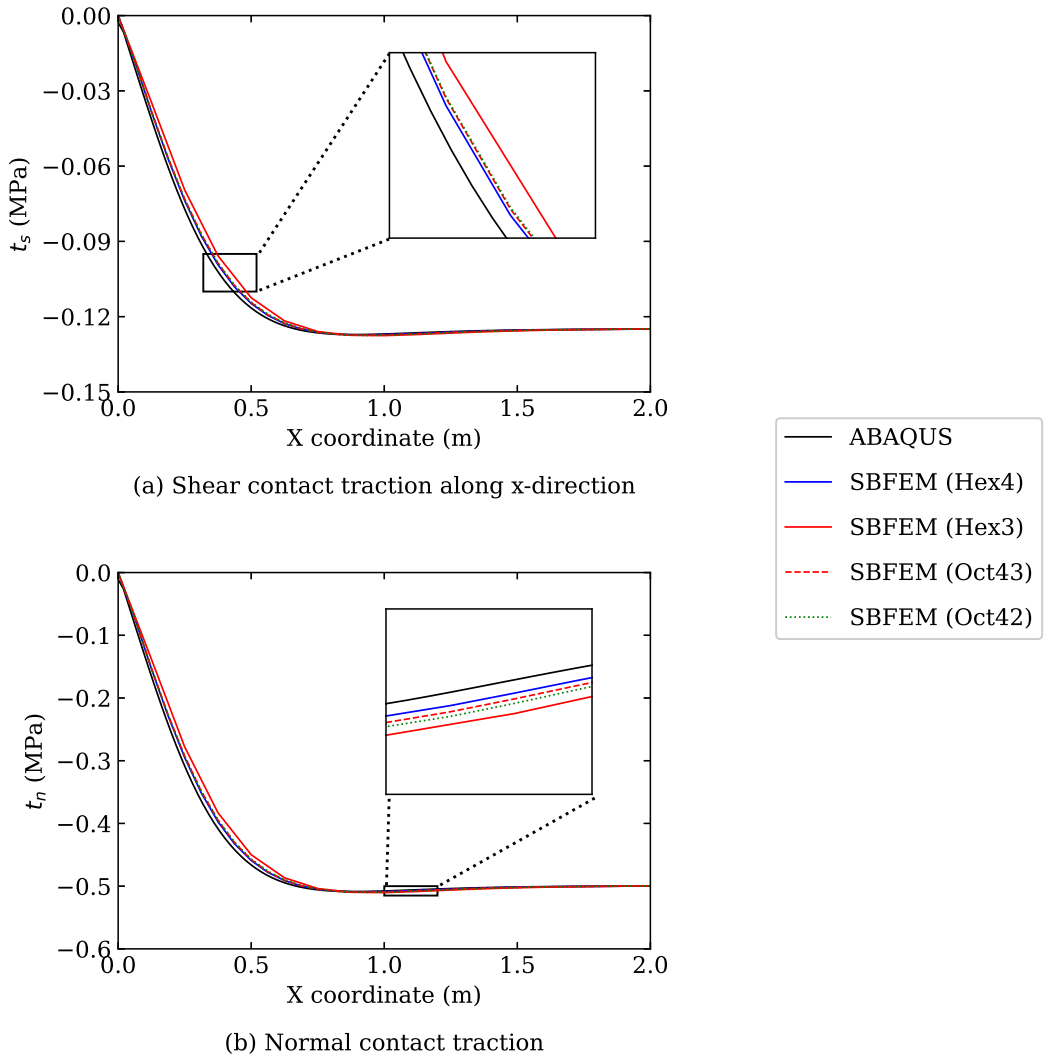


Figure 7.18: Convergence study of the contact tractions

(introduced in Chapter 6).

The convergence of the normal and shear tractions obtained along the contact surface at the mid-plane ($y = 0.125\,\mathrm{m}$) for various mesh sizes are shown in Fig 7.18. The reference solution is obtained using Abaqus with fine hexahedral mesh having element length $L_e = 0.02\,\mathrm{m}$. It is observed that the results obtained using the octree-based local mesh refinement (e.g. Oct43 and Oct42) are of similar accuracy as the globally refined mesh (Hex4).

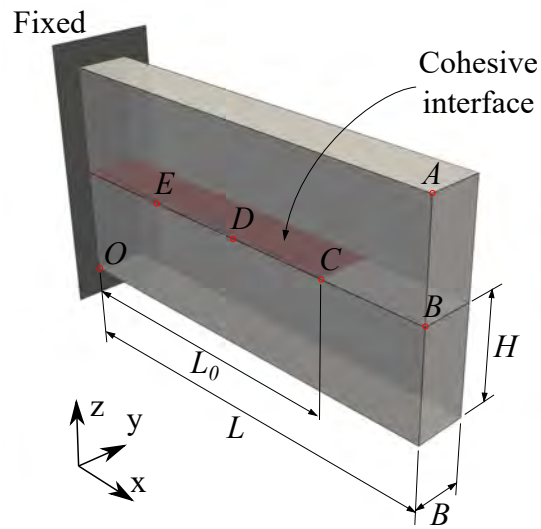


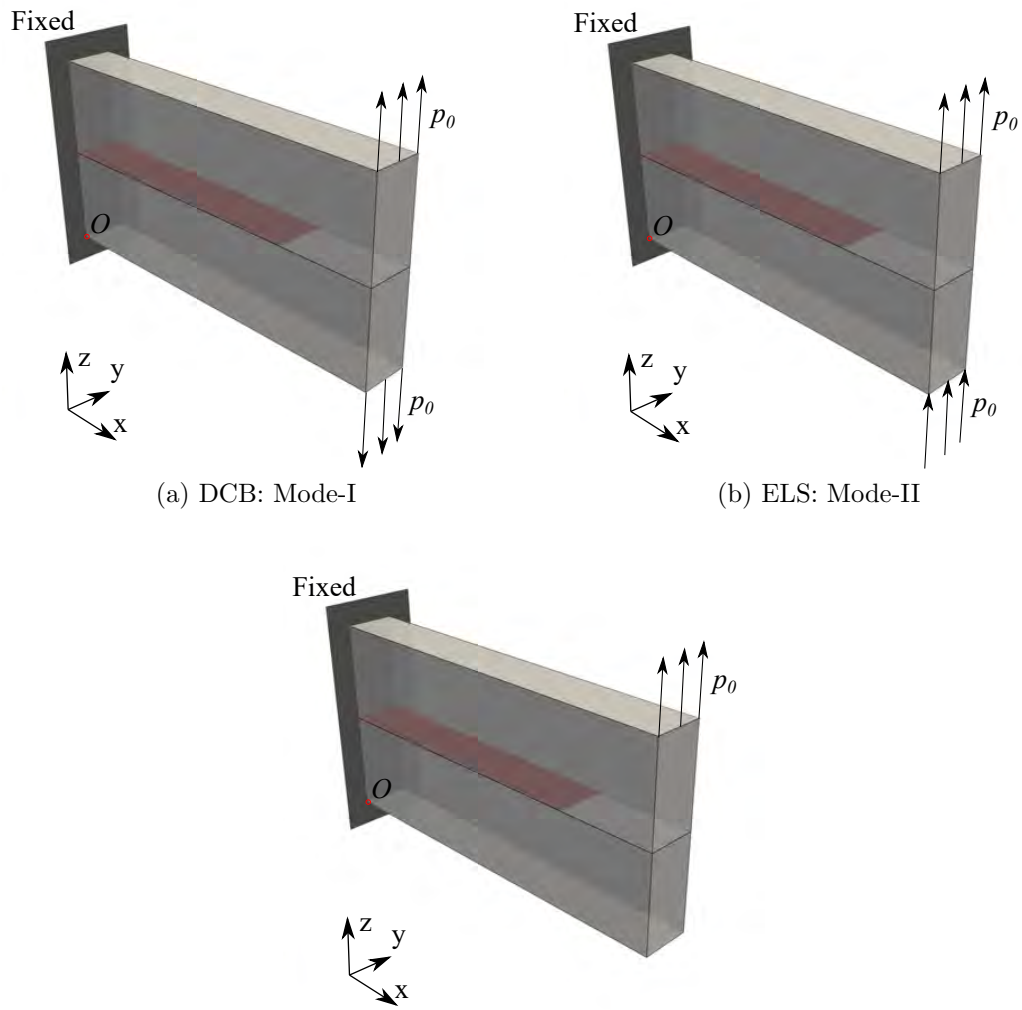
Figure 7.19: A pair of cantilever beams with cohesive interface

Quasi-static decohesion

In this section, we verify the implementation of decohesion/delamination process for various loading conditions. We consider a pair of cantilever beams, which are initially in contact with each other, as shown in Fig. 7.19. The geometric and bulk material properties of the model are the same as used in the previous example. A cohesive interface of length $L_0 = 1.5\,\mathrm{m}$ is introduced at the interface of the two beams. The friction is assumed to be absent.

The delamination process is simulated for the mode-I, mode-II and mixed-mode (I and II) loading conditions. These are represented by a double cantilever beam (DCB) model, an end-loaded split (ELS) model and a mixed-mode end-loaded split (MMELS) model respectively, as illustrated in Fig. 7.20. The cohesive properties of the interface is based on linear traction separation law with $f_n = 10\,\mathrm{kPa}$, $f_s = f_t = 20\,\mathrm{kPa}$, $G_n^c = 1\,\mathrm{N/m}$, $G_s^c = G_t^c = 2\,\mathrm{N/m}$, $K_n = 1.0\,\mathrm{GPa/m}$ and $K_s = K_t = 0.5\,\mathrm{GPa/m}$. The mesh-properties of the model are same as given in Table 7.2. The equilibrium solution is obtained using local-arc length controlled algorithm.

We perform a convergence study for the pure- and mixed-mode delamination using



(a) DCB: Mode-I

(b) ELS: Mode-II

(c) MMELS: Mixed mode I and II

Figure 7.20: Loading conditions

both the hexahedral and octree meshes, as shown in Fig. 7.21. Here, the force applied on the edge of the beam ($F_0 = p_0 B$) is plotted against the displacement measured at point-A along the z-axis at each time step. The reference solution is obtained using Abaqus with a hexahedral mesh having the element length $L_e = 0.05\,\mathrm{m}$. It is observed that the octree-mesh is able to achieve convergence using local mesh refinement near the contact interface.

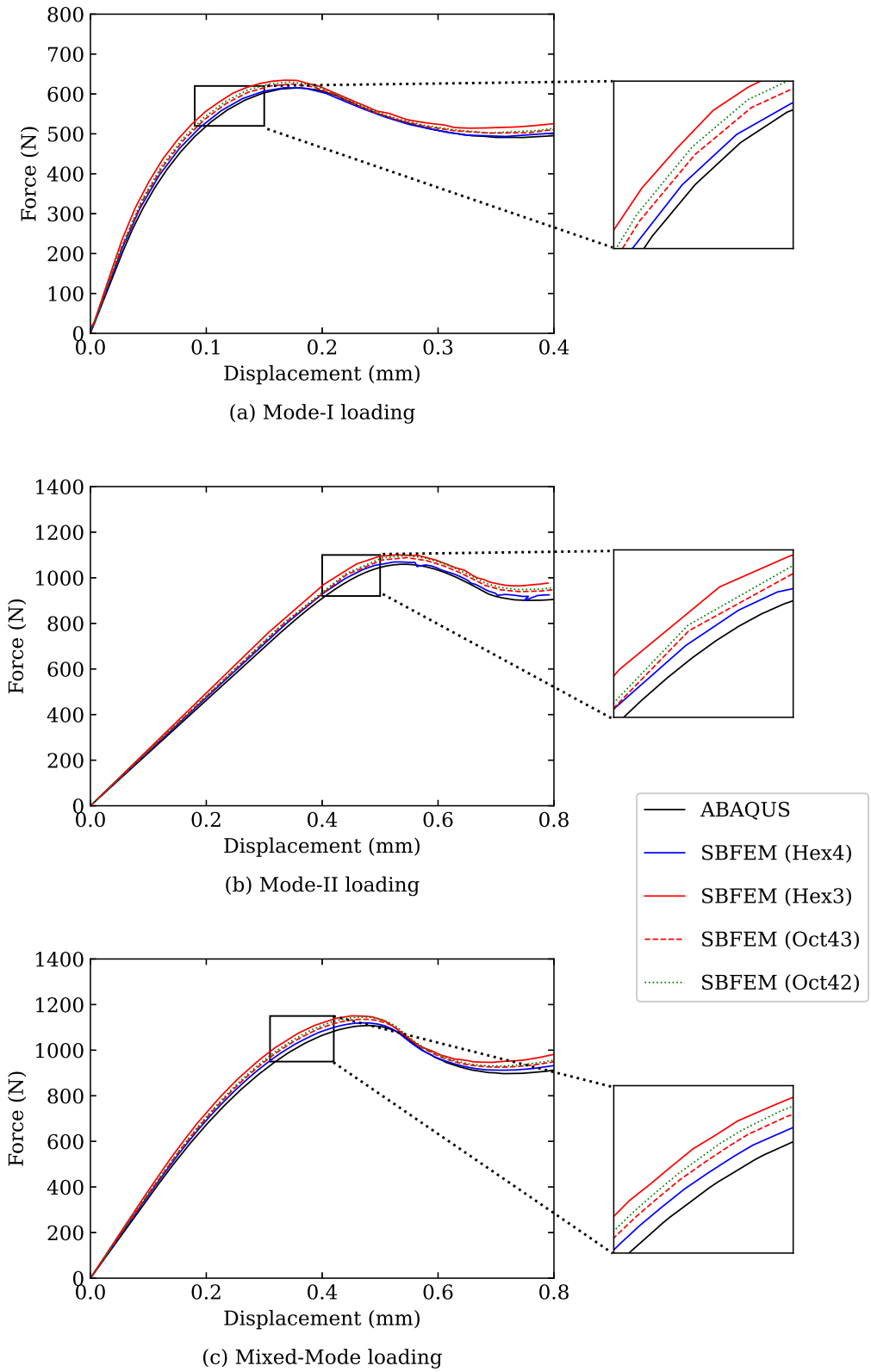
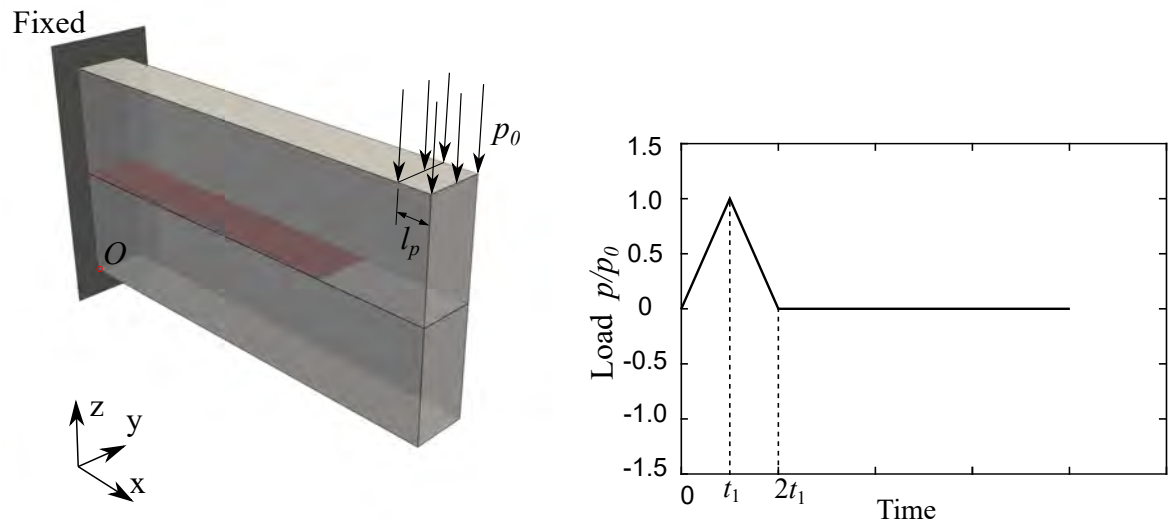


Figure 7.21: Convergence study of the decohesion for various loading conditions



(a) Load location (b) Load history: Triangular

Figure 7.22: Impact loading condition

Dynamic decohesion

We now verify the decohesion process under dynamic/impact loading for the cantilever beam pair. The upper beam is subjected to an impact loading as shown in Fig. 7.22. The impact loading is modelled as a pressure p_0 uniformly distributed over the length $l_p = 0.125\,\mathrm{m}$ and having a triangular load history with $t_1 = 25\,\mu\mathrm{s}$. The cohesive properties of the interface is based on linear traction separation law with $f_n = f_s = f_t = 0.5\,\mathrm{MPa}$ and $G_n^c = G_s^c = G_t^c = 50\,\mathrm{N/m}$. The penalty parameters are taken as $0.7 \leq \alpha_0,\ \beta_0 \leq 1$. The analysis is performed using the octree meshes (Oct65 and Oct64) and compared with the reference solution obtained in Abaqus software using hexahedral mesh (Hex6). The mesh properties are summarised in Table 7.3. The solution is obtained using an explicit time integration scheme based on the CDM. Time step is chosen as $\Delta t = 0.5\Delta t_{cr}$, where Δt_{cr} is the critical time step of the dynamic equilibrium problem without the penalty term.

The results of the dynamic delamination simulation of the cantilever beam pair are shown in Fig. 7.23. Two loading cases are considered: $p_0 = 2.5$ MPa and 5.0 MPa. The

Table 7.3: Mesh properties of the cantilever beam pair

Mesh name	No. of Elements	No. of DOFs	L_e^{min}[m]	L_e^{max}[m]
Hex6	133,120	434,214	0.01563	0.01563
Oct65	32,768	121,506	0.01563	0.03125
Oct64	14,848	64,272	0.01563	0.06250

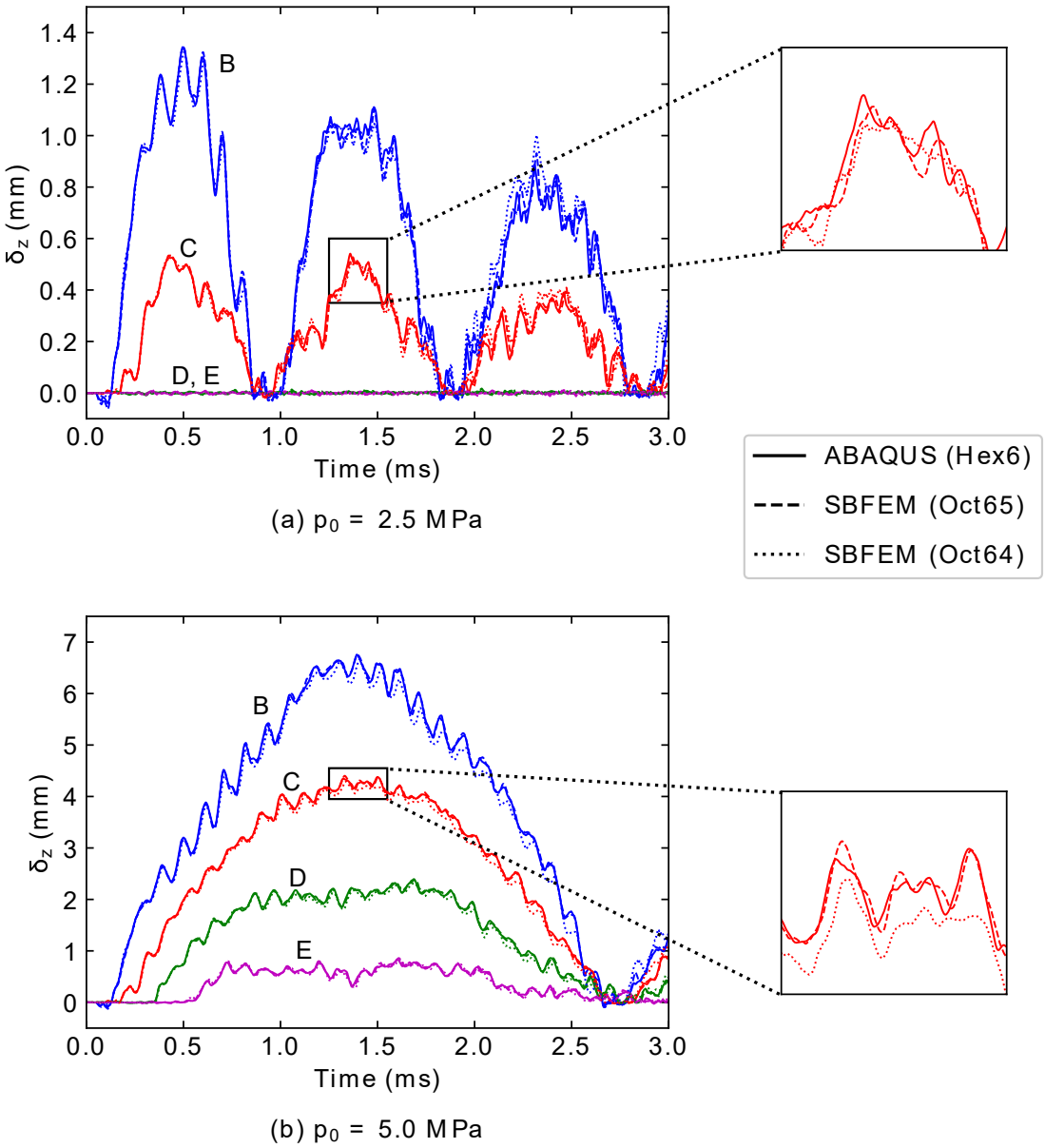
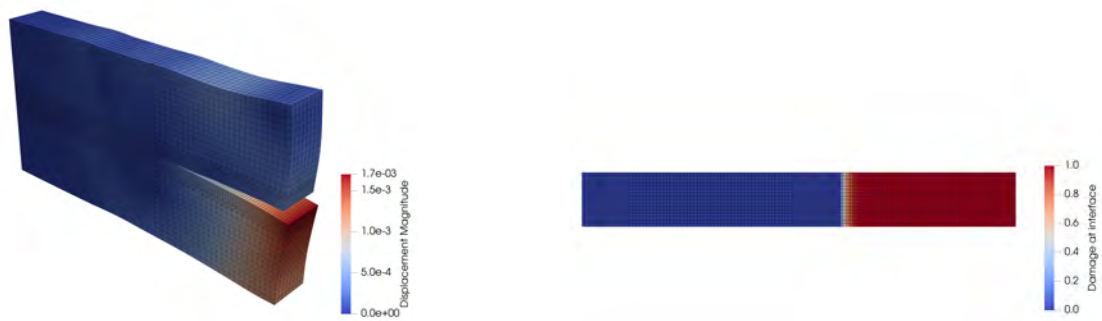
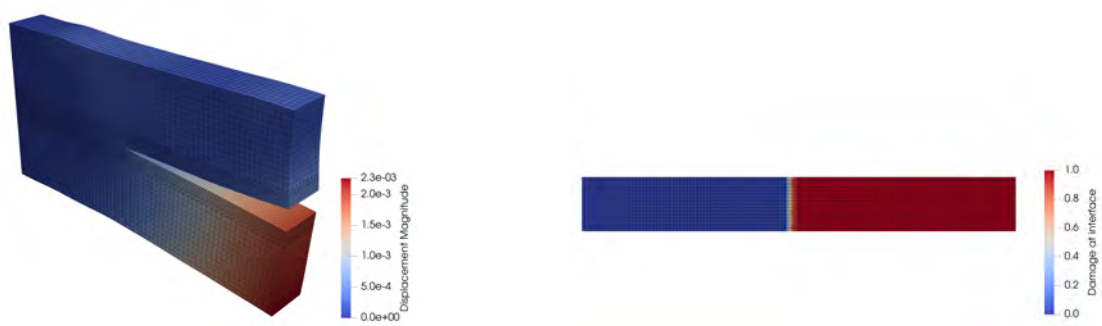


Figure 7.23: Results of dynamic decohesion process for various loading conditions

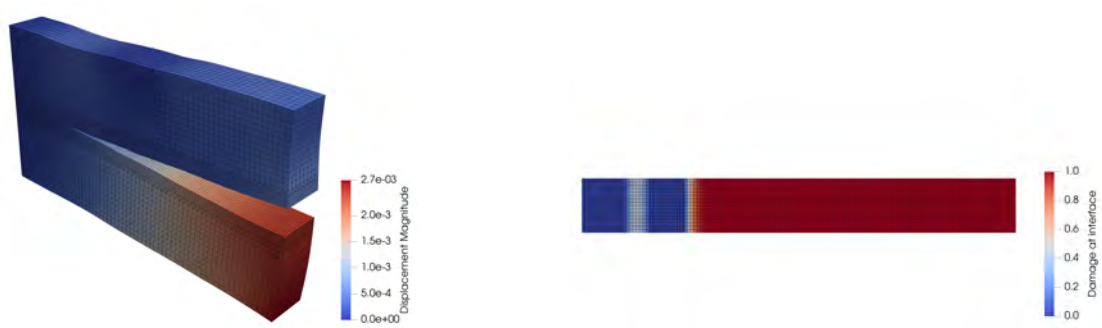
(a) $t = 0.074\,\mathrm{ms}$



(b) $t = 0.221\,\mathrm{ms}$



(c) $t = 0.369\,\mathrm{ms}$



(d) $t = 0.516\,\mathrm{ms}$

Figure 7.24: Displacement (deformation scale: 50) and interface damage value obtained for $p_0 = 5.0$ MPa at various time steps using Oct65 mesh

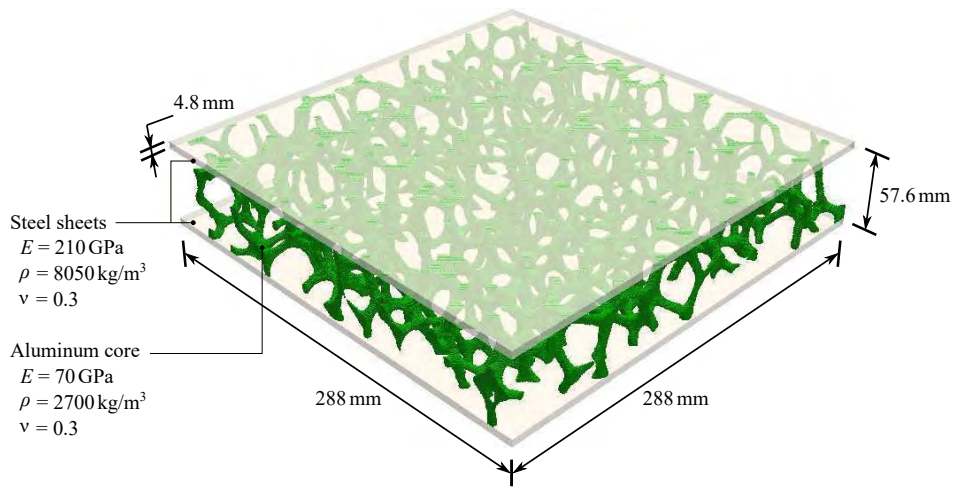


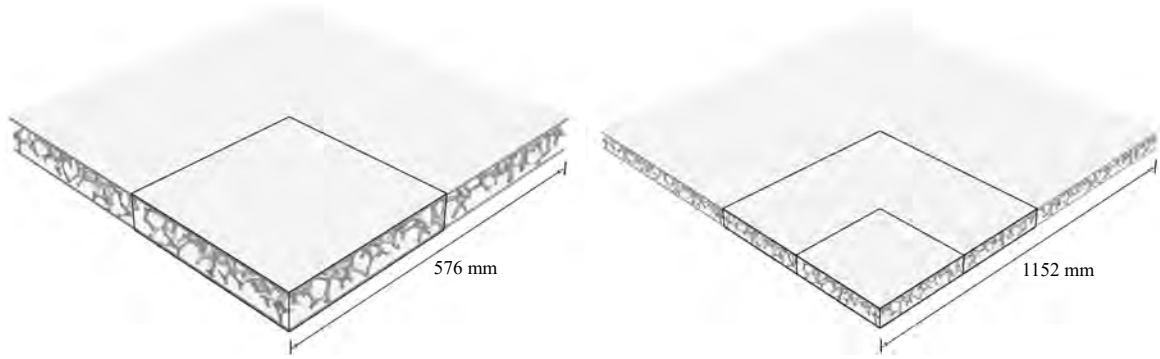
Figure 7.25: Geometry of the 1×1 sandwich panel

opening gap (δ_z) is measured at four points B, C, D and E, which are equally placed along the interface (Ref: Fig. 7.19). It is observed that at $p_0 = 2.5$ MPa, the impact load is not sufficient to cause the full delamination of the cantilever beam pair, as seen from the negligible opening gaps at points D and E. As the impact load is increased to $p_0 = 5.0$ MPa, the cantilever beam pair is fully delaminated after the impact. Also, it is observed that the results obtained using the octree-mesh shows good agreement with those obtained using hexahedral mesh, eventhough the number of DOFs in the octree meshes is much lesser. The displacement field at a few time steps obtained for $p_0 = 5.0$ MPa using Oct65 mesh is shown in Fig. 7.24.

7.5.3 Impact delamination of sandwich panel

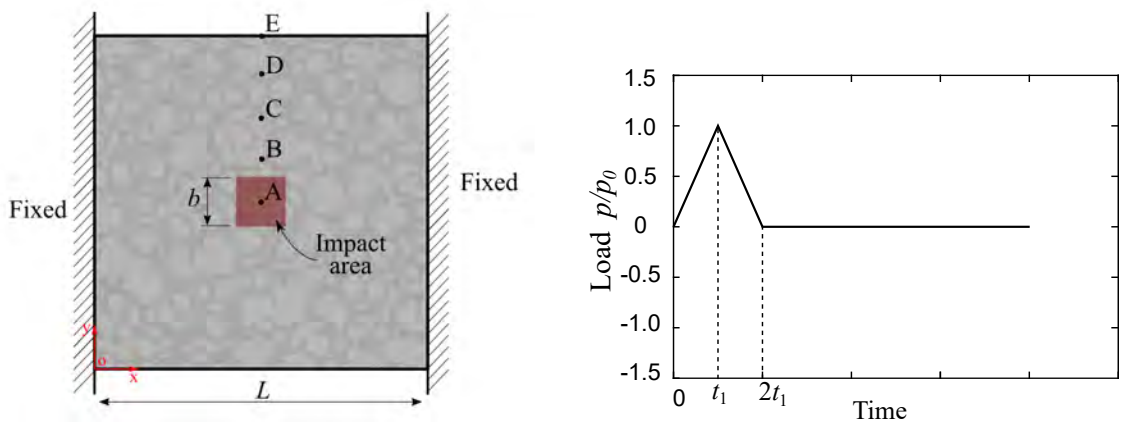
In this example, we perform the simulation of impact-induced delamination in a sandwich panel.

We consider a sandwich panel with two cover sheets and a foam-core (as previously used in Chapter 4). The digital image is obtained using X-ray CT scans and shown in Fig. 7.25. The outer dimension of the panel is 288 mm $\times$ 288 mm $\times$ 57.6 mm (denoted as 1×1 panel) and the thickness of the cover sheet is 4.8 mm. The material of the



(a) 2×2 panel (b) 4×4 panel

Figure 7.26: Mirroring operation of the sandwich panel



(a) Boundary condition and impact area (Top-view) (b) Load history: Triangular impulse

Figure 7.27: Boundary and loading conditions of the sandwich panel

foam-core is assumed as aluminium with the following properties: Young's modulus $E_\mathrm{a} = 70\,\mathrm{GPa}$, Poisson's ratio $\nu_\mathrm{a} = 0.3$, and mass density $\rho_\mathrm{a} = 2700\,\mathrm{kg/m^3}$. The cover-sheet material is assumed as steel with the following properties: $E_\mathrm{s} = 210\,\mathrm{GPa}$, $\nu_\mathrm{s} = 0.3$ and $\rho_\mathrm{s} = 8050\,\mathrm{kg/m^3}$. The foam core is assumed to be bonded to cover sheet using the a thin layer of adhesive AV138. Further, the cohesive interface of the adhesive bond is assumed to follow a linear traction-separation law with the following parameters [404]: $f_n = 40\,\mathrm{MPa}$, $f_s = f_t = 30\,\mathrm{MPa}$, $G_n^c = 300\,\mathrm{N/m}$ and $G_s^c = G_t^c = 600\,\mathrm{N/m}$. The value of initial/penalty stiffness is kept at $K_n = K_s = K_t = 45000\,\mathrm{GPa/m}$. Number of Gaussian

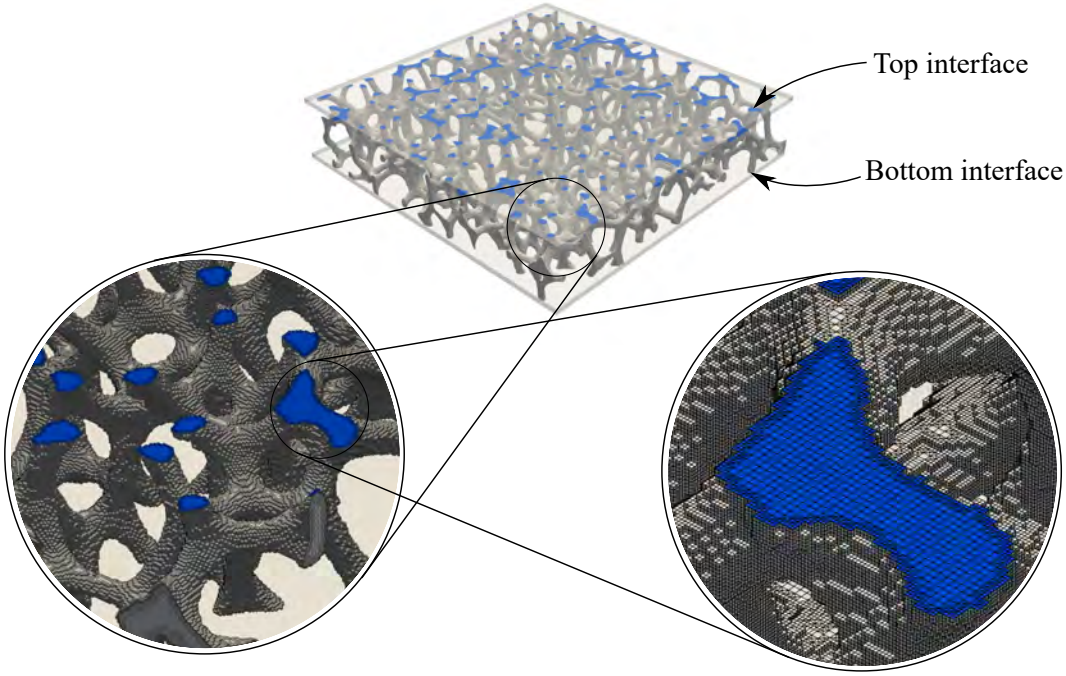


Figure 7.28: Mesh of the 1×1 sandwich panel showing interfaces

Table 7.4: Mesh properties of the sandwich panels

Model name	No. of Elements	No. of DOFs
1×1	13,682,061	67,104,564
2×2	54,728,244	268,129,533
4×4	218,912,976	1,071,934,908

points in each interface element is kept as $N_g = 1$. The frictional interaction at the interface is ignored. In order to test the developed octree-based approach for massively parallel computation of impact delamination problem and for studying its weak scaling behaviour, we mirror the given panel twice along its section, as illustrated in Fig. 7.26. The first mirroring operation creates a 2×2 panel shown in Fig. 7.26a, where the outer edge length is 576 mm. The second mirroring operation creates a 4×4 panel shown in Fig. 7.26b with an outer edge length of 1152 mm.

The top view of the panel (for all the three sizes considered) is shown in Fig. 7.27a. The origin of the Cartesian coordinate system is placed at the bottom-left corner.

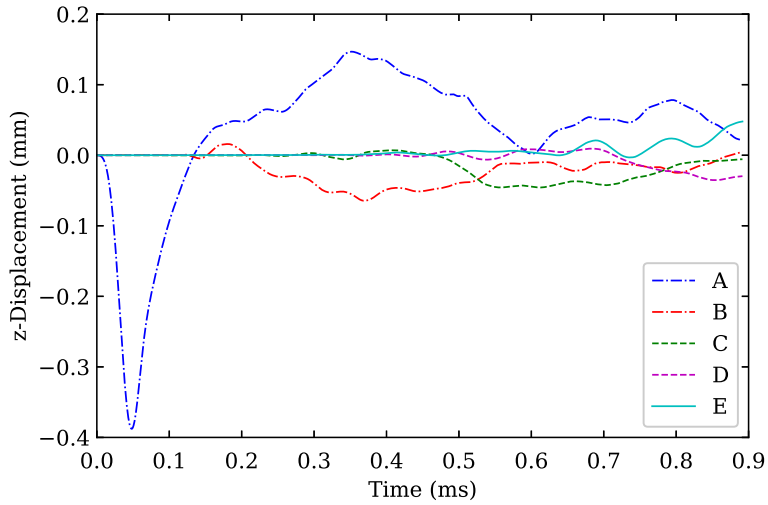


Figure 7.29: Response history of selected points in 4×4 sandwich panel

The panel-sections at $x = 0$ and $x = L$ are fixed against displacement. A uniformly distributed impact load is applied over a small square area of side length $b = 48\,\text{mm}$ at the centre of the top cover sheet. The impact load history is assumed as a triangular shaped impulse, as shown in Fig. 7.27b with $t_1 = 25\,\mu\text{s}$ and amplitude $p_0 = 25\,\text{MPa}$.

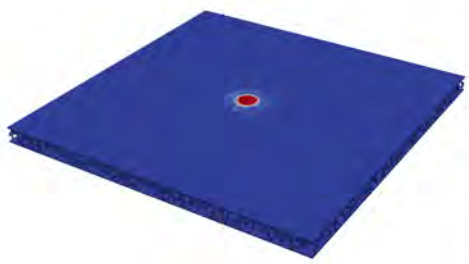
To accurately capture the wave propagation in the panel and the delamination of the cover sheets, a fine discretisation of the sandwich panel geometry is created using octree mesh with a maximum element size of 0.96 mm and a minimum element size of 0.24 mm. The octree mesh generated for the 1×1 panel is shown in Fig. 7.28. To capture the decohsion process under impact loading, the zero-thickness cohesive interface elements (shown in blue color) are inserted at the interface of the foam-core and the cover-sheets (top and bottom). The mesh properties of the three sandwich panels are shown in Table 7.4. The mesh partitioning is done using the *METIS* package [332, 405].

For the explicit analysis, the time step size is kept as $\Delta t = 1.486 \times 10^{-5}\,\text{ms}$. First, we perform an impact delamination simulation for 4×4 panel with a total duration of 0.89 ms (60000 time steps). The displacement along the $z-$direction of the five selected points, shown in Fig. 7.27a (coordinates are $A(576, 576, 57.6)$, $B(576, 720, 57.6)$, $C(576, 864, 57.6)$, $D(576, 1008, 57.6)$ and $E(576, 1152, 57.6)$, unit: mm), are plotted in

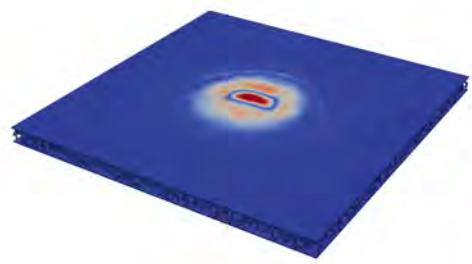
Fig. 7.29. The magnitude of the displacement response at the five time instances are presented in Fig. 7.30, illustrating the wave propagation process after the impact. Further, the delamination at the interface of foam-core and two cover sheets (both top and bottom) due to the impacting load is shown in Figs. 7.31-7.32 for the selected time steps. It is observed that the delamination (or damage) of the interface is prominent around the area of impact, and is observed for both the top and bottom interfaces. However, away from the impacting region, the interfaces remain intact.

The computational time study of the impact-delamination analysis in sandwich panels is shown in Fig. 7.33. These results are also presented in Tables 7.5, 7.6, and 7.7 for convenience. The CPU time is obtained for first 1000 time steps of the analysis. A consistent scaling is obtained for all the panel sizes over a large range of number of CPU cores, as observed from Fig. 7.33a. Further, the %Communication+Wait time ($\%T_W$) increases steadily from 3.5% to 50% over a range of ∼12.3k cores, as shown in Fig. 7.33b. This demonstrates the robustness of the parallel implementation of the octree-based explicit non-linear solver over a large number of cores.

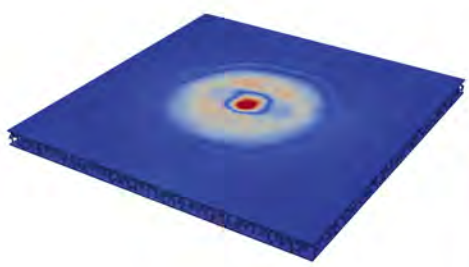
We now perform two scaling studies: *The strong and the weak scaling*, in order to show the parallel efficiency of the solver over the given range of N_{cores}. The *strong scaling* or *fixed model-size scaling* results are shown in Fig. 7.33c-d. It is observed that for $N_{cores} \leq 48$ (single compute-node), the efficiency tends to decrease due to memory-bandwidth limitation on a single compute-node. For $N_{cores} > 48$ (multiple compute-nodes), the efficiency increases due to the *cache-effect* (explained in Section 4.5.6). The weak scaling results are shown in Fig. 7.33e-f for various values of DOFs per core. It is observed that the efficiency decreases with an increase in number of cores when $N_{cores} \leq 48$. However, for $N_{cores} > 48$, the reduction in the efficiency is marginal. Overall, the present curves show similar trends as obtained previously for the explicit elasto-dynamics analysis of sandwich panels in Section 4.5.6.



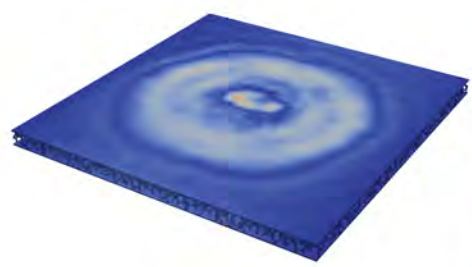
(a) $t = 0.059\,\mathrm{ms}$



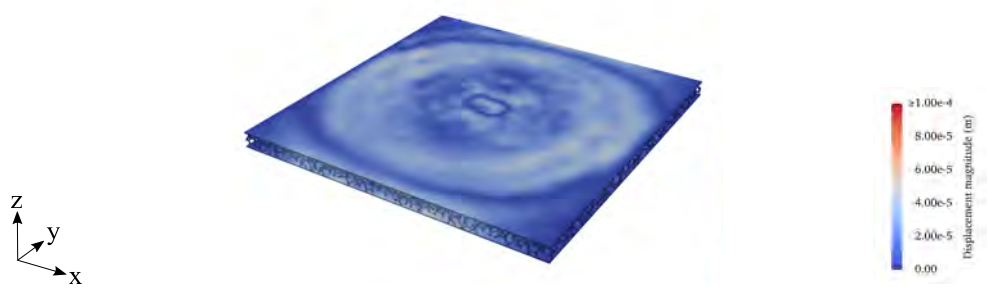
(b) $t = 0.297\,\mathrm{ms}$



(c) $t = 0.416\,\mathrm{ms}$

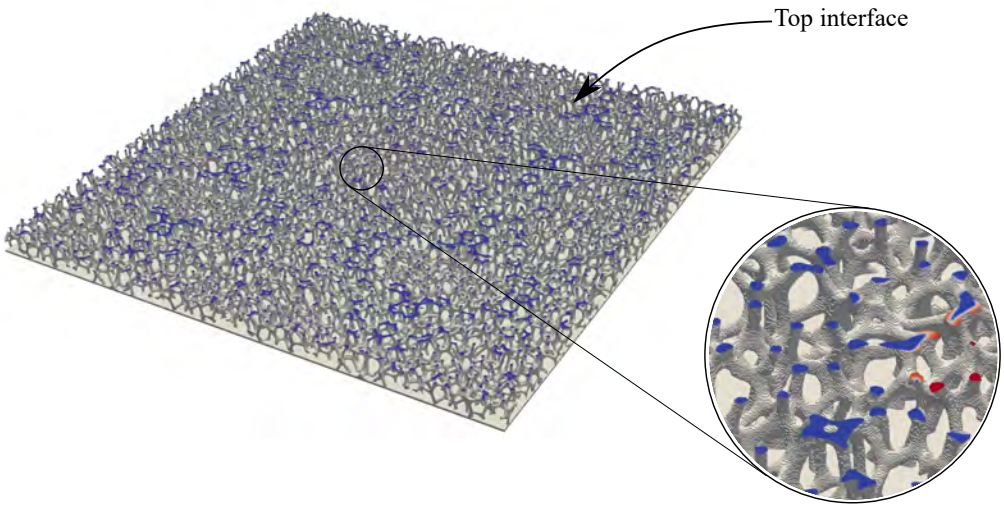


(d) $t = 0.654\,\mathrm{ms}$

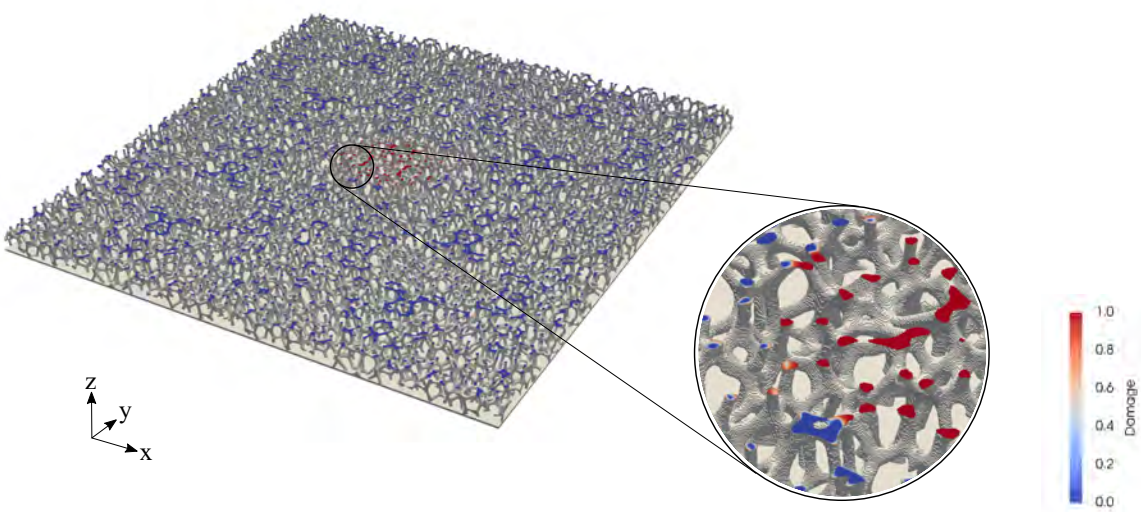


(e) $t = 0.892\,\mathrm{ms}$

Figure 7.30: Displacement contours obtained at selected time steps for 4×4 sandwich panel

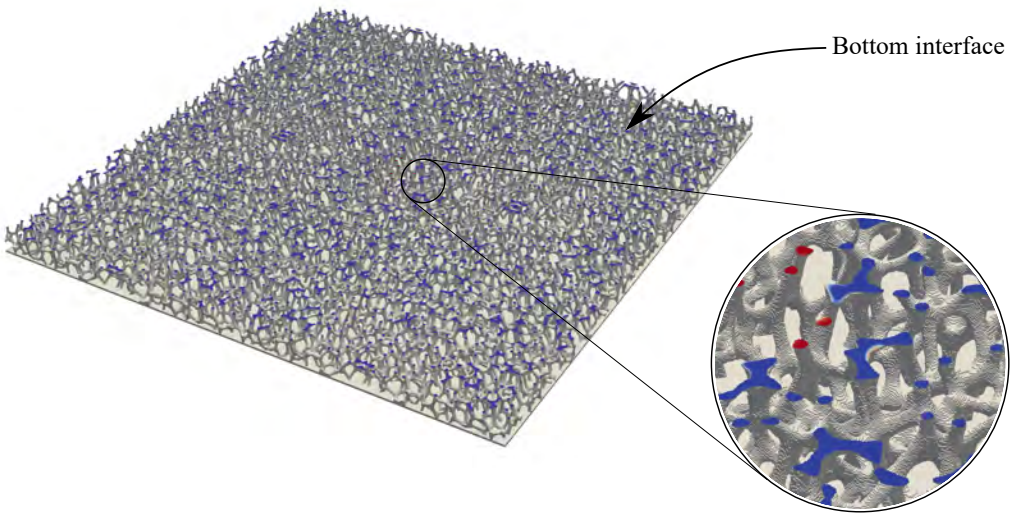


(a) $t = 0.059\,\mathrm{ms}$

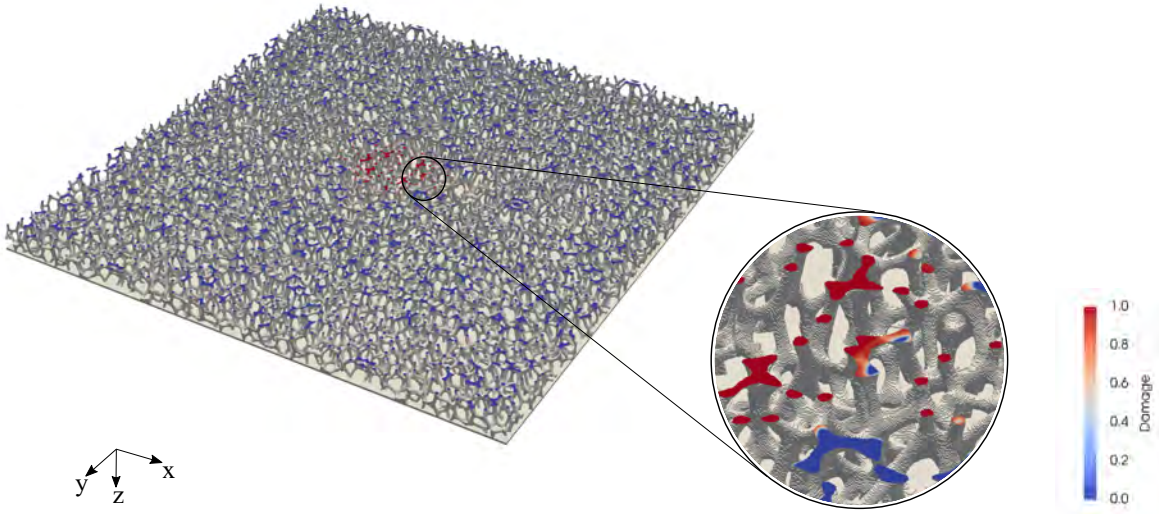


(b) $t = 0.535\,\mathrm{ms}$

Figure 7.31: Damage contours at the interface of foam-core and top cover sheet shown at selected time steps for 4×4 sandwich panel

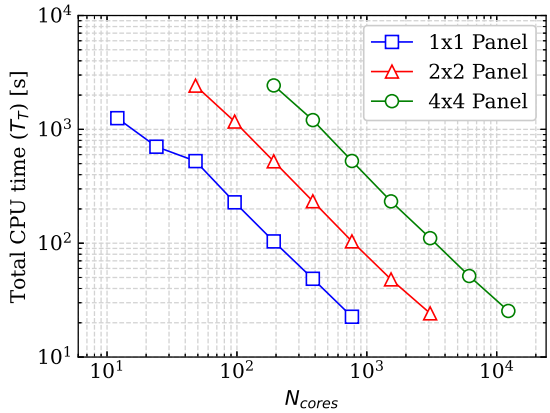


(a) $t = 0.059\,\mathrm{ms}$

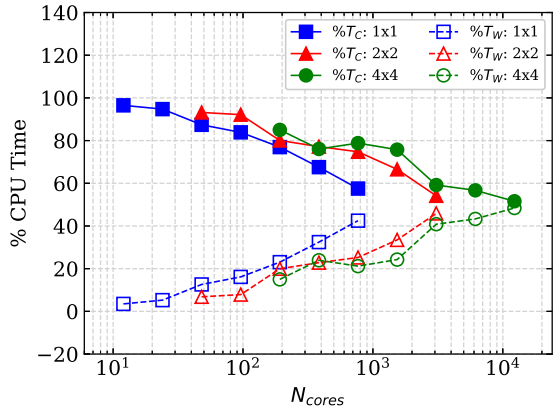


(b) $t = 0.535\,\mathrm{ms}$

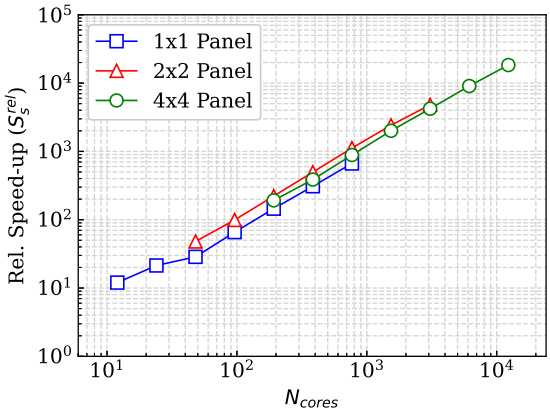
Figure 7.32: Damage contours at the interface of foam-core and bottom cover sheet shown at the selected time steps for 4×4 sandwich panel



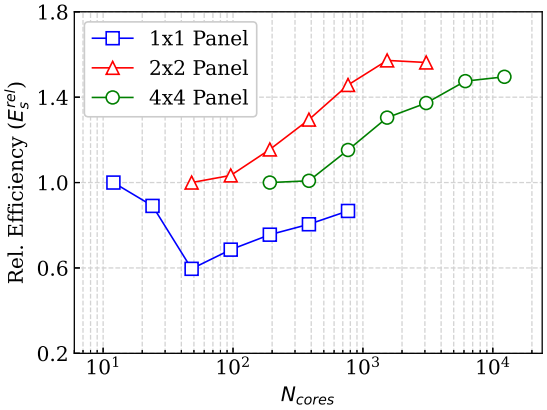
(a) Total CPU time



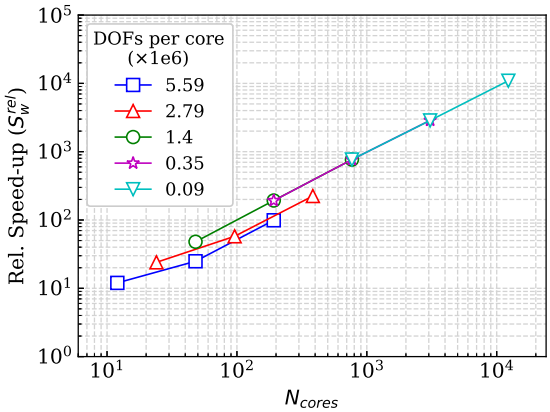
(b) Total CPU time components



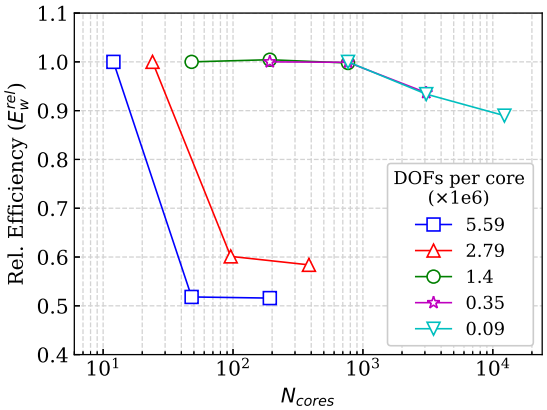
(c) Strong scaling: Rel. Speed-up



(d) Strong scaling: Rel. Efficiency



(e) Weak scaling: Rel. Speed-up



(f) Weak scaling: Rel. Efficiency

Figure 7.33: Computational time study for cohesive delamination of the sandwich panel

Table 7.5: Computational time for the cohesive delamination of the sandwich panel

N_{cores}	1×1 panel			2×2 panel			4×4 panel		
	T_C^m	T_{W}^m	T_{T}	T_C^m	T_{W}^m	T_{T}	T_C^m	T_{W}^m	T_{T}
12	1,210.2	43.8	1,254.0	-	-	-	-	-	-
24	667.0	37.0	704.0	-	-	-	-	-	-
48	459.8	66.3	526.1	2,255.0	165.0	2,420.0	-	-	-
96	191.7	36.9	228.6	1078.9	91.9	1170.8	-	-	-
192	79.8	23.9	103.7	419.3	104.5	523.6	2,066.8	364.2	2,431.0
384	32.9	15.8	48.7	180.3	53.3	233.6	917.1	288.5	1205.6
768	13.0	9.6	22.6	77.6	26.2	103.8	415.6	111.7	527.3
1536	-	-	-	32.0	16.1	48.1	176.5	56.5	233.0
3072	-	-	-	13.1	11.1	24.2	65.5	45.2	110.7
6144	-	-	-	-	-	-	29.2	22.3	51.5
12288	-	-	-	-	-	-	13.1	12.3	25.4

T_C^m: Mean calculation time [s], T_W^m: Mean comm.+waiting time [s], T_{T}: Total CPU time [s]

Table 7.6: Strong scaling results of the sandwich panel

N_{cores}	1×1 panel		2×2 panel		4×4 panel	
	S_s^{rel}	E_s^{rel}	S_s^{rel}	E_s^{rel}	S_s^{rel}	E_s^{rel}
12	12.0	1.00	-	-	-	-
24	21.4	0.89	-	-	-	-
48	28.6	0.60	48.0	1.00	-	-
96	65.8	0.69	99.2	1.03	-	-
192	145.1	0.76	221.8	1.16	192.0	1.00
384	309.0	0.80	497.3	1.29	387.2	1.01
768	665.8	0.87	1,119.1	1.46	885.2	1.15
1536	-	-	2,414.9	1.57	2,003.2	1.30
3072	-	-	4,800.0	1.56	4,216.4	1.37
6144	-	-	-	-	9,063.2	1.48
12288	-	-	-	-	18,376.1	1.50

S_s^{rel}: Relative speed-up (strong scaling), E_s^{rel}: Relative efficiency (strong scaling)

Table 7.7: Weak scaling results of the sandwich panel

DOFs per core ($\times 10^6$)	Panel	N_{cores}	T_{T} [s]	S_w^{rel}	E_w^{rel}
5.59	1×1	12	1,254.0	12.0	1.00
	2×2	48	2,420.0	24.9	0.52
	4×4	192	2,431.0	99.0	0.52
2.79	1×1	24	704.0	24.0	1.00
	2×2	96	1,170.8	57.7	0.60
	4×4	384	1,205.6	224.2	0.58
1.4	1×1	48	526.1	48.0	1.00
	2×2	192	523.8	192.8	1.00
	4×4	768	527.3	766.3	1.00
0.35	1×1	192	103.7	192.0	1.00
	2×2	768	103.8	767.3	1.00
	4×4	3,072	110.7	2,877.8	0.94
0.09	1×1	768	22.6	768.0	1.00
	2×2	3,072	24.2	2,868.9	0.93
	4×4	12,288	25.4	10,933.4	0.89

S_w^{rel}: Relative speed-up (weak scaling), E_w^{rel}: Relative efficiency (weak scaling)

The memory usage in performing explicit delamination analysis of the sandwich panels is shown in Fig. 7.34 and tabulated in Table 7.8. The total memory usage M_T has two components: The system memory usage M_S and the model data memory usage M_D. These variables have been introduced in Section 4.5.6.

The system memory usage is almost proportional to the number of cores utilised during the analysis. This is due to the requirement of a separate memory space for each MPI process, where the system files are copied to allow local computations.

On the other hand, the model data memory usage increases only marginally when the number of cores is increased. This behaviour is similar to those obtained in the explicit elastodynamic analysis of the sandwich panels performed in Section 4.5.6. It can be attributed to the requirement of copying the octree-pattern data to each process, increase in interfacial DOFs with more mesh partitions, etc.

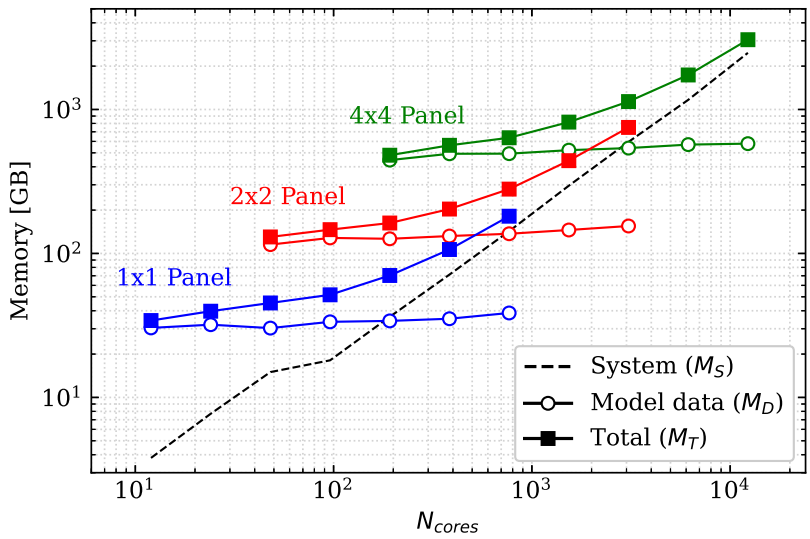


Figure 7.34: Memory usage in the cohesive delamination analysis of the sandwich panel

Table 7.8: Memory usage in the cohesive delamination analysis of the sandwich panel

N_{cores}	1×1 panel			2×2 panel			4×4 panel		
	M_S	M_D	M_T	M_S	M_D	M_{T}	M_S	M_D	M_T
12	3.8	30.4	34.2	-	-	-	-	-	-
24	7.7	32.0	39.7	-	-	-	-	-	-
48	15.0	30.3	45.3	15.0	115.2	130.2	-	-	-
96	18.1	33.5	51.6	18.1	128.2	146.3	-	-	-
192	36.3	34.0	70.3	36.3	126.4	162.7	36.3	444.9	481.2
384	71.3	35.2	106.5	71.3	132.1	203.4	71.3	492.4	563.7
768	142.8	38.6	181.4	142.8	137.0	279.8	142.8	493.3	636.1
1536	-	-	-	295.5	145.5	441.0	295.5	521.6	817.1
3072	-	-	-	595.3	155.1	750.4	595.3	539.3	1,134.6
6144	-	-	-	-	-	-	1,166.1	570.1	1,736.2
12288	-	-	-	-	-	-	2,473.2	578.9	3,052.1

M_S: System memory [GB], M_D: Model data memory [GB], M_T: Total memory [GB]

7.6 Summary

In this chapter, the octree-based formulation is extended for high-performance computing of cohesive-fracture analysis under impact loading. Two separate processes are analysed in the impact-induced fracture problems: (1) The instantaneous contact-condition at the contact surface of the impacting bodies and (2) impact-induced wave propagation resulting in delamination at the material interfaces or other weak spots in the impacting bodies. The contact-impact simulation is performed using the non-linear regularised penalty-based approach, which reduces the artificial oscillations at the contact surface of the impacting bodies. For the simulation of delamination at the material interface, the delamination initiation is captured using the quadratic stress-based criteria, while the delamination propagation is captured using the fracture energy-based criteria. A cohesive interface element formulation is utilised, which can model both the onset of delamination and its propagation without any prior knowledge of the crack location and propagation direction. The interface element formulation is also used to model the impact problem using the penalty-based approach. The explicit-time integration technique is employed to obtain the solution as the method can stably capture a highly non-linear physical process involving contact-impact, wave propagation and cohesive delamination occurring over a very short duration.

The formulation for octree-based parallel explicit-solver, developed in Chapter 4, is extended to accommodate the interface element patterns to model the cohesive-fracture process. To satisfy the compatibility of the interface elements with the faces of the octree cells, only two unique interface element patterns are required: the rectangular and isosceles right-triangle elements. The master-element matrices of each pattern can be pre-computed and stored with minimal memory requirements. The element matrix of all the interface elements in the mesh can be calculated using scalar (weight) multiplication with the corresponding master-element matrix. An efficient calculation of nodal force vectors is achieved using the pattern-by-pattern (PBP) approach that

allows dense matrix computation leading to better cache utilisation.

A few numerical examples are presented to establish the efficacy of the developed formulation. The examples of cubes under normal contact-impact and cantilever beam pair validate the implementation of the computer code. Further, these examples also show the advantage of applying local-mesh refinement near the contact/cohesive interface region using the octree mesh. Here, a much lesser number of DOFs is required as compared to hexahedral mesh to achieve a similar level of accuracy in the structural response. Further, to show the applicability of the octree-based approach in solving practical examples, the impact delamination in a sandwich panel is solved. It is observed that the developed solver can efficiently perform the massively parallel computation of impact-induced cohesive delamination in large-scale structural models with approximately one billion DOFs and using more than 12K cores in a distributed computing environment.

Chapter 8

Conclusions and recommendations

8.1 Concluding statements

In this thesis, a high-performance computational framework was developed for solving large-scale structural problems concerning impact-induced fracture analysis utilising octree meshes.

The SBFEM-based element shape functions were employed, which inherently eliminated the hanging node issue associated with octree-cells, unlike the conventional FEM. The use of balanced octree mesh allowed utilising only a limited number of octree-cell patterns for structural modelling and analysis. The master element matrices corresponding to these patterns were pre-computed. The storage of element matrices of individual elements is avoided as they could be obtained from simple scalar multiplication with master element matrices. This significantly reduced the memory requirements, especially for the large models. The developed computational framework was bolstered by automatic octree-based mesh generation and local refinement techniques, which enabled efficient pre-processing of models with complex geometries.

The matrix operations involved in the analysis were performed using the proposed pattern-by-pattern (PBP) approach. Here, the octree-mesh patterns were exploited to

recast the majority of the work into pattern-level dense matrix computations, avoiding global matrix assembly. This approach allowed better cache utilisation and aided the associated matrix operations limited by memory bandwidth, resulting in significant performance gains of the solvers.

The PBP approach could also provide very high scalability in a distributed computing environment allowing large-scale parallel simulations of structural problems. The parallelisation was carried out using the mesh-partitioning strategy in the SPMD paradigm, where the mesh was divided into several partitions and assigned to individual MPI processes working at individual cores. Each worker (or the MPI process) performed the local computations on each mesh part and exchanged the information with the neighbour partitions using the message passing technique.

Utilising the octree-based approach, an explicit solver was developed based on the central difference scheme for solving impact-induced fracture problems. Here, the associated physical processes such as elastic wave propagation, contact-impact and impact-induced cohesive delamination were efficiently simulated. Additionally, the efficacy of the octree-based approach was also demonstrated in the efficient implementation of implicit/iterative solvers (HHT-α and PCG) for simulating elastodynamic and elastostatic problems.

The summary of the developments undertaken in this thesis is as follows.

A parallel explicit-solver for elastodynamic problems using octree meshes

The octree-based approach was first employed in developing a parallel explicit-solver for efficiently obtaining the numerical solution of elastodynamic problems. The central difference method was adopted as it provides very high accuracy and stability limits in explicit time integration. The octree-patterns were utilised in the efficient computation of nodal force vector and the critical time step size. A special mass lumping technique was extended to three-dimensional problems, which allowed solving the system of linear

equations simultaneously for obtaining nodal displacements.

The implementation of the octree-based explicit solver was verified using the benchmark examples. An optimal convergence rate was obtained in the modal and transient analyses. Then, three numerical examples of varying complexity were solved to demonstrate the computational performance and scalability of the developed solver. A Ricker wavelet, a triangular pulse load, and a sine-burst signal were considered in these examples to excite the structures and to illustrate realistic and high-resolution engineering applications. The primary outcomes of these examples are explained as follows.

First, the multi-storey building and ground example was solved, which showed that the in-house code achieved a better efficiency (using octree mesh) when compared to the commercial software ABAQUS (utilising tetrahedral mesh) for a given accuracy. Next, the wave propagation in the mountainous region was simulated, which showed the applicability of the solver in modelling the seismic-like events of moderately long duration. Finally, a guided wave propagation simulation was performed in a large-scale sandwich panel with more than a billion DOFs. The scaling study revealed the robust implementation of the solver as it attained a very high efficiency in carrying out massively parallel simulations utilising more than 16K computing cores in a distributed computing environment.

Octree-based parallel implicit-solver for elasto-dynamics/statics

The application of the octree-based approach was also shown for parallel implementation of implicit solver based on the HHT-α scheme. Compared to the explicit solver, the implicit solver has the advantage of unconditional stability and allows larger time steps. Therefore, they enable efficient simulation of long-duration seismic and vibration analysis. The computationally most expensive step of the implicit solver is to solve a system of linear equations. An iterative (PCG) solver was employed that has several advantages over the direct solvers in handling large-sized models. The PBP approach was

utilised for carrying out efficient matrix operations associated with the iterative solver, avoiding the need for global matrix assembly and significantly reducing the associated memory requirements.

The implementation of the developed implicit solver was first verified using the static loading example of a prismatic beam under pure-bending, where an optimal convergence rate was observed. Next, a larger problem of a concrete block under uniaxial static loading was analysed, where high computational performance and scalability of the octree-based PCG solver were established. Then, two numerical examples with dynamic loading were presented and solved using the developed implicit (HHT-α) solver. These are explained as follows.

The example of a building surrounded by a trench showed the practical application of the developed solver. A computational time study was performed for parallel computation of the analysis with $\sim$106M DOF model. An efficient scaling with more than 3K computing cores was obtained.

Next, a castle on a mountain example was presented to demonstrate the ability of the octree mesh in performing efficient multiscale modelling. The large-scale seismic wave simulation in the mountain (considering the effect of a fault) and small-scale dynamic analysis of the structure (considering its complex geometric shape) were performed simultaneously in one model. Such simulation may not be very efficiently solved using the explicit time integration schemes as the small element size in one region generally controls the critical time step size for the entire analysis.

Cohesive/contact interface formulation for simulating compressive-fracture in two dimensions

A formulation for cohesive/contact interfaces was developed for performing compressive fracture analysis in two-dimensional. The cohesive/contact traction along the crack surface was implemented using 4-node zero-thickness interface elements. The contact

condition was satisfied using the penalty-stiffness method. Additionally, a novel sideface traction approach was developed to handle the stress field near the crack-tip region. Here, the crack-tip was modelled using an SBFEM based cracked element. The tractions arising from cohesive/contact interaction were applied as linear sideface tractions on the cracked elements. An iterative scheme was proposed to estimate the sideface tractions, such that the crack-tip singularities vanish when the equilibrium position is reached at each load step.

The proposed method led to an accurate estimate of stress and displacement fields over the crack surface, including the crack-tip region. For solving cohesive crack propagation problems, the proposed technique eliminated the mesh-mapping requirement at the end of each load-step as encountered in the shadow domain procedure [188]. The stresses in the crack-tip region could be directly extracted from the formulation considering the effect of cohesive and frictional-contact tractions. Therefore, the crack propagation condition and directions were estimated using the stress-based criterion with minimal computational effort.

The efficacy of the developed approach was first demonstrated for the crack-contact problems through a mesh-sensitivity analysis of the cracked plate model with a hole. Next, a benchmark example of mixed-mode cohesive crack propagation was solved, where the load-displacement curves and the crack paths were captured with sufficient accuracy, even for a coarse crack-tip mesh. Then, the application of the proposed method was shown through practical examples. Simulation of the Brazilian split test showed that the developed formulation could effectively capture the size-effect in concrete. Next, the compressive fracture analysis of the concrete-aggregate interface was simulated, considering the interfacial-debonding and matrix cracking involving wing-crack initiation/propagation phenomena. A linear fit of the contact tractions on the sidefaces of the cracked element predicted the initiation load and angle of the wing cracks with reasonable accuracy.

Impact-induced fracture simulation employing octrees and interface elements

The cohesive/contact interface formulation was extended and merged with the octree-based approach for performing massively parallel computation of impact-induced fracture problems in three dimensions. Two distinct processes were considered: (1) The instantaneous contact condition at the surface of the impacting bodies and (2) impact-induced wave propagation resulting in delamination at the material interfaces or other weak spots in the impacting bodies.

The contact-impact simulation was performed using the non-linear regularised penalty-based approach. The non-linear regularisation reduced the artificial oscillations at the contact surface of the impacting bodies. The delamination at the material interface was modelled using two processes: (1) The delamination initiation, which was captured using the quadratic stress-based criteria. (2) The delamination propagation, which was incorporated using the fracture energy-based criteria. A formulation for 2D interface elements was developed to model cohesive interaction on the crack surface during the delamination process. The same interface element formulation was applied to capture contact-impact interaction using the penalty-based approach. To stably capture the highly non-linear physical process involving contact-impact, wave propagation and cohesive-delamination occurring over a very short duration, the explicit-time integration scheme was adopted.

To model the surface interaction (cohesive or contact), unique patterns of interface elements were identified (the rectangular and isosceles right-triangle shape) compatible with the faces of the octree cells. The master-element matrices of each pattern were pre-computed and stored with a little memory requirement. The pattern-by-pattern (PBP) approach was utilised to perform dense matrix operations of the interfacial force vectors that allowed better cache utilisation, boosting the performance of the explicit solver.

The implementation of the developed formulation was validated by presenting a few numerical examples. First, the problem of cubes under normal contact-impact was solved. It was observed that the non-linear regularisation of the penalty parameter was effective in reducing the artificial oscillations at the contacting surface. Next, the problem of a cantilever beam pair was modelled for verifying the frictional contact and cohesive delamination implementation. These examples showed the advantage of using the octree mesh to apply local-mesh refinement near the contact/cohesive interface region.

Finally, the real-world application of the developed approach was demonstrated by modelling the impact delamination phenomena in a sandwich panel. It was shown that the solver could efficiently perform the impact-induced cohesive delamination analysis in large-scale structural models with more than a billion DOFs. The scalability of the solver was shown by performing a massively parallel computation in a distributed computing environment using more than 12K cores.

8.2 Recommendations for future work

The octree-based computational framework developed in this thesis can be further utilised in solving a wide variety of engineering problems. The possible extensions of the present work are suggested as follows.

Modelling multiple failure modes in impact-induced fracture analysis

The high-velocity impact fracture in composites laminates, timber-composites, sandwich panels, and other heterogeneous materials like concrete is a fairly complicated process. Due to the heterogeneous nature of these materials, the failure can occur in various modes, including matrix crushing and cracking, delamination produced by interlaminar stresses, fibre buckling and failure, and full penetration with the projectile

perforating through the target.

A more realistic analysis demands incorporating these failure modes into the simulation. However, a very fine mesh could be required to accurately capture these failure modes, demanding large computational resources, especially when cohesive element formulation is used. The developed octree-based parallel formulation could be utilised to efficiently perform the large-scale simulation with a much lower memory footprint.

Adaptive phase-field fracture/damage analyses

The computational modelling of quasi-static and dynamic fracture of brittle and quasi-brittle materials using phase-field fracture/damage models (PFMs) have gained much attention from researchers over the last decade. The PFMs can straightforwardly handle topologically complex fracture phenomena such as initiation, branching and intersecting cracks in both two and three dimensions. However, solving fracture problems using PFMs is computationally expensive as a very fine mesh is needed to capture the high spatial variability of the displacement field.

In the framework of SBFEM, the quadtree-based adaptive mesh refinement have been utilised by researchers to efficiently model the fracture using PFMs in two dimensions [406]. To cope with the high computational demand in large-scale fracture problems in three dimensions, a massively parallel computation can be performed using the octree-based approach to obtain the solution in a feasible time. Further, octree-based adaptive mesh-refinement can be implemented to capture the changing geometry encountered in crack-propagation problems.

Topology optimisation

Topology optimisation is a powerful tool capable of generating highly efficient and economical structural designs across diverse sectors, such as construction, automotive, etc. However, optimising the topology of continuum structures subjected to constraints

is computationally expensive, esp in three dimensions.

In the framework of SBFEM, some authors have utilised an octree-based adaptive approach to reduce the computational burden [407, 408]. Combining the octree mesh adaptivity with unique patterns can provide significant improvement in computational efficiency and allow handling of a much larger problem size for HPC applications.

Extension to GPU based computing

The use of graphics processor unit (GPU) devices for large-scale parallel computing of matrix operations has gained much popularity in recent years. One of the convenient means for the researchers to use GPU based parallel computing is NVIDIA's compute unified device architecture (CUDA) [409].

The structural solver can be decomposed into a sequence of matrix operations, e.g. matrix multiplication, which constitutes the most memory-bandwidth intensive and computationally demanding steps. Since the GPU devices have high memory bandwidth and work in single instruction multiple threads (SIMT) paradigm, they can significantly accelerate the octree-based matrix operations for analysing large-scale models.

Sideface traction approach in three dimensions

In the present work, the sideface traction approach developed for the two-dimensional cohesive crack propagation and crack-closure problems has shown accurate computation of displacement and stress fields in the crack-tip region. This approach can be extended to perform three-dimensional simulation by applying sideface tractions over the cracked polyhedral/octree-type element.

Bibliography

[1] C. Song, The scaled boundary finite element method: Introduction to theory and implementation, John Wiley & Sons, 2018.

[2] S. Timoshenko, J. N. Goodier, Theory of elasticity, Engineering societies monographs, 3d ed., McGraw-Hill, New York,, 1969.

[3] D. Cendon, J. Galvez, M. Elices, J. Planas, Modelling the fracture of concrete under mixed loading, International Journal of Fracture 103 (2000) 293–310.

[4] E. T. Ooi, S. Natarajan, C. Song, E. H. Ooi, Crack propagation modelling in concrete using the scaled boundary finite element method with hybrid polygon–quadtree meshes, International Journal of Fracture 203 (2017) 135–157.

[5] M. Arrea, Mixed-mode crack propagation in mortar and concrete, Dept. of Struct. Engrg. Rep. (1981) 81–13.

[6] C. Rocco, G. V. Guinea, J. Planas, M. Elices, Size effect and boundary conditions in the brazilian test: Theoretical analysis, Materials and Structures 32 (1999) 437.

[7] C. Rocco, G. V. Guinea, J. Planas, M. Elices, Size effect and boundary conditions in the brazilian test: Experimental verification, Materials and Structures 32 (1999) 210.

[8] M. A. Tasdemir, A. K. Maji, S. P. Shah, Crack propagation in concrete under compression, Journal of Engineering Mechanics 116 (1990) 1058–1076.

[9] N. J. Carpenter, R. L. Taylor, M. G. Katona, Lagrange constraints for transient finite element surface contact, International Journal for Numerical Methods in Engineering 32 (1991) 103–128.

[10] J. N. Reddy, Introduction to the finite element method, McGraw-Hill Education, 2019.

[11] K. J. Bathe, Finite element procedures, Upper Saddle River, NJ: Prentice Hall, 2006.

[12] T. Belytschko, T. Black, Elastic crack growth in finite elements with minimal remeshing, International Journal for Numerical Methods in Engineering 45 (1999) 601–620.

[13] C. Song, J. P. Wolf, The scaled boundary finite-element method – a primer: Solution procedures, Computers & Structures 78 (2000) 211–225.

[14] J. P. Wolf, C. Song, The scaled boundary finite-element method – a primer: Derivations, Computers & Structures 78 (2000) 191–210.

[15] T. J. Hughes, J. A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Computer Methods in Applied Mechanics and Engineering 194 (2005) 4135–4195.

[16] O. C. Zienkiewicz, R. L. Taylor, J. Z. Zhu, The finite element method: Its basis and fundamentals, Elsevier, 2005.

[17] S. Chan, I. Tuba, W. Wilson, On the finite element method in linear fracture mechanics, Engineering Fracture Mechanics 2 (1970) 1–17.

[18] Z. Yang, J. Chen, Fully automatic modelling of cohesive discrete crack propagation in concrete beams using local arc-length methods, International Journal of Solids and Structures 41 (2004) 801–826.

[19] J. Gálvez, J. Planas, J. Sancho, E. Reyes, D. Cendón, M. Casati, An embedded cohesive crack model for finite element analysis of quasi-brittle materials, Engineering Fracture Mechanics 109 (2013) 369–386.

[20] D. Y. Hodges, M. Y. Rutkowski, Free-vibration analysis of rotating beams by a variable-order finite-element method, AIAA Journal 19 (1981) 1459–1466.

[21] M. Dokainish, K. Subbaraj, A survey of direct time-integration methods in computational structural dynamics–i. explicit methods, Computers & Structures 32 (1989) 1371–1386.

[22] R. M. McMeeking, J. Rice, Finite-element formulations for problems of large elastic-plastic deformation, International Journal of Solids and Structures 11 (1975) 601–616.

[23] A. J. Abbo, Finite element algorithms for elastoplasticity and consolidation, University of Newcastle Newcastle Upon Tyne, Australia, 1997.

[24] P. Wriggers, Finite element algorithms for contact problems, Archives of Computational Methods in Engineering 2 (1995) 1–49.

[25] P. Wriggers, Computational contact mechanics, Springer, 2006.

[26] W. J. Gordon, C. A. Hall, Construction of curvilinear co-ordinate systems and applications to mesh generation, International Journal for Numerical Methods in Engineering 7 (1973) 461–477.

[27] S.-S. Liu, R. Gadh, Basic LOgical Bulk shapes (BLOBs) for finite element hexahedral mesh generation, in: 5th International Meshing Roundtable, Citeseer, 1996, pp. 291–306.

[28] L. Mingwu, S. E. Benzley, G. Sjaardema, T. Tautges, A multiple source and target sweeping method for generating all hexahedral finite element meshes, in:

Proceedings, 5th International Meshing Roundtable, volume 96, Citeseer, 1996, pp. 217–225.

[29] T. Blacker, The cooper tool, 5th international meshing roundtable, Sandia National Laboratories, Pitsburgh (1996) 13–29.

[30] M. L. Staten, S. A. Canann, S. J. Owen, Bmsweep: locating interior nodes during sweeping, Engineering with Computers 15 (1999) 212–218.

[31] B. Delaunay, et al., Sur la sphere vide, Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk 7 (1934) 1–2.

[32] S. J. Owen, A survey of unstructured mesh generation technology., IMR 239 (1998) 267.

[33] S. Lo, A new mesh generation scheme for arbitrary planar domains, International journal for numerical methods in engineering 21 (1985) 1403–1426.

[34] J. Peraire, K. Morgan, J. Peiró, Unstructured finite element mesh generation and adaptive procedures for cfd, in: AGARD, Applications of Mesh Generation to Complex 3-D Configurations, 1990, pp. 15–34.

[35] M. A. Yerry, M. S. Shephard, A modified quadtree approach to finite element mesh generation, IEEE Computer Graphics and Applications 3 (1983) 39–46.

[36] M. A. Yerry, M. S. Shephard, Automatic three-dimensional mesh generation by the modified-octree technique, International Journal for Numerical Methods in Engineering 20 (1984) 1965–1990.

[37] H. Samet, The quadtree and related hierarchical data structures, ACM Computing Surveys (CSUR) 16 (1984) 187–260.

[38] L. Giovannelli, J. Ródenas, J. Navarro-Jiménez, M. Tur, Direct medical image-based finite element modelling for patient-specific simulation of future implants, Finite Elements in Analysis and Design 136 (2017) 37–57.

[39] I. T. Ghisi, J. J. Camata, A. L. Coutinho, Impact of tetrahedralization on parallel conforming octree mesh generation, International Journal for Numerical Methods in Fluids 75 (2014) 800–814.

[40] W. Cantwell, J. Morton, Comparison of the low and high velocity impact response of cfrp, Composites 20 (1989) 545–551.

[41] M. Richardson, M. Wisheart, Review of low-velocity impact properties of composite materials, Composites Part A: Applied Science and Manufacturing 27 (1996) 1123–1131.

[42] M. Meo, A. Morris, R. Vignjevic, G. Marengo, Numerical simulations of low-velocity impact on an aircraft sandwich panel, Composite Structures 62 (2003) 353–360.

[43] P. H. Geubelle, J. S. Baylor, Impact-induced delamination of composites: A 2D simulation, Composites Part B: Engineering 29 (1998) 589–602.

[44] D. J. Elder, R. S. Thomson, M. Q. Nguyen, M. L. Scott, Review of delamination predictive methods for low speed impact of composite laminates, Composite Structures 66 (2004) 677–683.

[45] F. Aymerich, F. Dore, P. Priolo, Prediction of impact-induced delamination in cross-ply composite laminates using cohesive interface elements, Composites science and technology 68 (2008) 2383–2390.

[46] M. May, Numerical evaluation of cohesive zone models for modeling impact induced delamination in composite materials, Composite Structures 133 (2015) 16–21.

[47] J. Zhang, X. Zhang, Simulating low-velocity impact induced delamination in composites by a quasi-static load model with surface-based cohesive contact, Composite Structures 125 (2015) 51–57.

[48] D. S. Dugdale, Yielding of steel sheets containing slits, Journal of the Mechanics and Physics of Solids 8 (1960) 100–104.

[49] G. I. Barenblatt, The mathematical theory of equilibrium cracks in brittle fracture, in: Advances in applied mechanics, volume 7, Elsevier, 1962, pp. 55–129.

[50] P. P. Camanho, C. G. Dávila, Mixed-mode decohesion finite elements for the simulation of delamination in composite materials, NTRS, 2002.

[51] P. P. Camanho, C. G. Dávila, M. De Moura, Numerical simulation of mixed-mode progressive delamination in composite materials, Journal of Composite Materials 37 (2003) 1415–1438.

[52] P. Arbenz, G. H. van Lenthe, U. Mennel, R. Müller, M. Sala, A scalable multi-level preconditioner for matrix-free μ-finite element analysis of human bone structures, International Journal for Numerical Methods in Engineering 73 (2008) 927–947.

[53] K. Suresh, P. Yadav, Large-scale modal analysis on multi-core architectures, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, volume 45011, Citeseer, 2012, pp. 785–791.

[54] R. Espinha, K. Park, G. H. Paulino, W. Celes, Scalable parallel dynamic fracture simulation using an extrinsic cohesive zone model, Computer Methods in Applied Mechanics and Engineering 266 (2013) 144–161.

[55] Y. Efendiev, T. Y. Hou, Multiscale finite element methods: theory and applications, volume 4, Springer Science & Business Media, 2009.

[56] Z. Zhang, G. H. Paulino, W. Celes, Extrinsic cohesive modelling of dynamic fracture and microbranching instability in brittle materials, International Journal for Numerical Methods in Engineering 72 (2007) 893–923.

[57] P. Krysl, Z. Bittnar, Parallel explicit finite element solid dynamics with domain decomposition and message passing: Dual partitioning scalability, Computers & Structures 79 (2001) 345–360.

[58] J. C. Caendish, D. A. Field, W. H. Frey, An apporach to automatic three-dimensional finite element mesh generation, International Journal for Numerical Methods in Engineering 21 (1985) 329–347.

[59] K. Ho-Le, Finite element mesh generation methods: A review and classification, Computer-Aided Design 20 (1988) 27–38.

[60] M. S. Shephard, Approaches to the automatic generation and control of finite element meshes, Applied Mechanics Reviews 41 (1988) 169–185.

[61] P. George, Automatic mesh generation and finite element computation, Handbook of Numerical Analysis 4 (1996) 69–190.

[62] T. Belytschko, C. Parimi, N. Moës, N. Sukumar, S. Usui, Structured extended finite element methods for solids defined by implicit surfaces, International journal for numerical methods in engineering 56 (2003) 609–635.

[63] J. Keyak, J. Meagher, H. Skinner, C. Mote, Automated three-dimensional finite element modelling of bone: a new method, Journal of biomedical engineering 12 (1990) 389–397.

[64] W. E. Lorensen, H. E. Cline, Marching cubes: A high resolution 3d surface construction algorithm, in: ACM siggraph computer graphics, volume 21, ACM, 1987, pp. 163–169.

[65] D. M. Greaves, A. Borthwick, Hierarchical tree-based finite element mesh generation, International Journal for Numerical Methods in Engineering 45 (1999) 447–471.

[66] A. Tabarraei, N. Sukumar, Adaptive computations on conforming quadtree meshes, Finite Elements in Analysis and Design 41 (2005) 686–702.

[67] J. Bielak, O. Ghattas, E. Kim, Parallel octree-based finite element method for large-scale earthquake ground motion simulation, Computer Modeling in Engineering and Sciences 10 (2005) 99.

[68] T. J. Hughes, I. Levit, J. Winget, Element-by-element implicit algorithms for heat conduction, Journal of Engineering Mechanics 109 (1983) 576–585.

[69] T. J. Hughes, I. Levit, J. Winget, An element-by-element solution algorithm for problems of structural and solid mechanics, Computer Methods in Applied Mechanics and Engineering 36 (1983) 241–254.

[70] I. Kiss, S. Gyimothy, Z. Badics, J. Pavo, Parallel realization of the element-by-element fem technique by cuda, IEEE Transactions on magnetics 48 (2012) 507–510.

[71] P. Yadav, K. Suresh, Large scale finite element analysis via assembly-free deflated conjugate gradient, Journal of Computing and Information Science in Engineering 14 (2014).

[72] C. Song, J. P. Wolf, The scaled boundary finite-element method–alias consistent infinitesimal finite-element cell method–for elastodynamics, Computer Methods in Applied Mechanics and Engineering 147 (1997) 329–355.

[73] E. T. Ooi, H. Man, S. Natarajan, C. Song, Adaptation of quadtree meshes in the scaled boundary finite element method for crack propagation modelling, Engineering Fracture Mechanics 144 (2015) 101–117.

[74] A. Saputra, H. Talebi, D. Tran, C. Birk, C. Song, Automatic image-based stress analysis by the scaled boundary finite element method, International Journal for Numerical Methods in Engineering 109 (2017) 697–738.

[75] A. A. Saputra, S. Duczek, H. Gravenkamp, C. Song, Image-based 3D homogenisation using the scaled boundary finite element method, Computers & Structures 237 (2020) 106263.

[76] Y. Liu, A. A. Saputra, J. Wang, F. Tin-Loi, C. Song, Automatic polyhedral mesh generation and scaled boundary finite element analysis of STL models, Computer Methods in Applied Mechanics and Engineering 313 (2017) 106–132.

[77] E. T. Ooi, C. Song, F. Tin-Loi, Z. Yang, Polygon scaled boundary finite elements for crack propagation modelling, International Journal for Numerical Methods in Engineering 91 (2012) 319–342.

[78] S. Blackford, J. Dongarra, Lapack working note 41, appendix c (1999). URL: `http://www.netlib.org/lapack/lawnspdf/lawn41.pdf`.

[79] A. Abdelfattah, S. Tomov, J. Dongarra, Matrix multiplication on batches of small matrices in half and half-complex precisions, Journal of Parallel and Distributed Computing 145 (2020) 188–201.

[80] K. Ljungkvist, Techniques for finite element methods on modern processors, Ph.D. thesis, Uppsala University, 2015.

[81] L. Dalcín, R. Paz, M. Storti, MPI for Python, Journal of Parallel and Distributed Computing 65 (2005) 1108–1115.

[82] L. Dalcín, R. Paz, M. Storti, J. DElía, MPI for Python: Performance improvements and MPI-2 extensions, Journal of Parallel and Distributed Computing 68 (2008) 655–662.

[83] J. Chang, K. Nakshatrala, M. G. Knepley, L. Johnsson, A performance spectrum for parallel computational frameworks that solve PDEs, Concurrency and Computation: Practice and Experience 30 (2018) e4401.

[84] M. W. Bern, P. E. Plassmann, Mesh Generation., volume 6, 2000.

[85] M. Farrashkhalvat, J. Miles, Basic Structured Grid Generation: With an introduction to unstructured grid generation, Elsevier, 2003.

[86] S. J. Owen, T. R. Shelton, Evaluation of grid-based hex meshes for solid mechanics, Engineering with Computers 31 (2015) 529–543.

[87] S. Karman, V. Betro, Parallel hierarchical unstructured mesh generation with general cutting, in: 46th AIAA aerospace sciences meeting and exhibit, 2008, p. 918.

[88] L. Maréchal, Advances in octree-based all-hexahedral mesh generation: handling sharp features, in: Proceedings of the 18th international meshing roundtable, Springer, 2009, pp. 65–84.

[89] J. T. Oden, Historical comments on finite elements, in: A history of scientific computing, 1990, pp. 152–166.

[90] K. Gupta, J. Meek, A brief history of the beginning of the finite element method, International Journal for Numerical Methods in Engineering 39 (1996) 3761–3774.

[91] H. Liebowitz, E. Moyer Jr, Finite element methods in fracture mechanics, Computers & Structures 31 (1989) 1–9.

[92] T. J. Hughes, G. M. Hulbert, Space-time finite element methods for elastodynamics: Formulations and error estimates, Computer Methods in Applied Mechanics and Engineering 66 (1988) 339–363.

[93] J. N. Reddy, D. K. Gartling, The finite element method in heat transfer and fluid dynamics, CRC press, 2010.

[94] J. M. Jin, The finite element method in electromagnetics, John Wiley & Sons, 2015.

[95] J. Dolbow, N. Moës, T. Belytschko, Discontinuous enrichment in finite elements with a partition of unity method, Finite Elements in Analysis and Design 36 (2000) 235–260.

[96] T. Belytschko, N. Moës, S. Usui, C. Parimi, Arbitrary discontinuities in finite elements, International Journal for Numerical Methods in Engineering 50 (2001) 993–1013.

[97] J. A. Cottrell, A. Reali, Y. Bazilevs, T. J. Hughes, Isogeometric analysis of structural vibrations, Computer Methods in Applied Mechanics and Engineering 195 (2006) 5257–5296.

[98] T. J. Hughes, A. Reali, G. Sangalli, Duality and unified analysis of discrete approximations in structural dynamics and wave propagation: Comparison of p-method finite elements with k-method NURBS, Computer Methods in Applied Mechanics and Engineering 197 (2008) 4104–4124.

[99] I. Temizer, P. Wriggers, T. Hughes, Three-dimensional mortar-based frictional contact treatment in isogeometric analysis with NURBS, Computer Methods in Applied Mechanics and Engineering 209 (2012) 115–128.

[100] Y. Bazilevs, V. M. Calo, Y. Zhang, T. J. Hughes, Isogeometric fluid–structure interaction analysis with applications to arterial blood flow, Computational Mechanics 38 (2006) 310–322.

[101] E. De Luycker, D. J. Benson, T. Belytschko, Y. Bazilevs, M. C. Hsu, X-FEM in isogeometric analysis for linear fracture mechanics, International Journal for Numerical Methods in Engineering 87 (2011) 541–565.

[102] S. S. Ghorashi, N. Valizadeh, S. Mohammadi, Extended isogeometric analysis for simulation of stationary and propagating cracks, International Journal for Numerical Methods in Engineering 89 (2012) 1069–1101.

[103] W. Fang, X. Chen, T. Yu, T. Q. Bui, Effects of arbitrary holes/voids on crack growth using local mesh refinement adaptive xiga, Theoretical and Applied Fracture Mechanics 109 (2020) 102724.

[104] S. D. Rao, S. Raju, Stable generalized iso-geometric analysis (sgiga) for problems with discontinuities and singularities, Computer Methods in Applied Mechanics and Engineering 348 (2019) 535–574.

[105] J. Lachat, J. Watson, Effective numerical treatment of boundary integral equations: A formulation for three-dimensional elastostatics, International Journal for Numerical Methods in Engineering 10 (1976) 991–1005.

[106] C. Brebbia, J. Dominguez, Boundary element methods for potential problems, Applied Mathematical Modelling 1 (1977) 372–378.

[107] J. Dominguez, Stress analysis around anchor plates: A boundary element method application, Ph.D. thesis, PhD Thesis. Universidad de Sevilla, 1977.

[108] A. H. D. Cheng, D. T. Cheng, Heritage and early history of the boundary element method, Engineering Analysis with Boundary Elements 29 (2005) 268–302.

[109] L. Gaul, M. Schanz, A comparative study of three boundary element approaches to calculate the transient response of viscoelastic solids with unbounded domains, Computer Methods in Applied Mechanics and Engineering 179 (1999) 111–123.

[110] T. Cruse, Lateral constraint in a cracked, three dimensional elastic body, International Journal of Fracture Mechanics 6 (1970) 326–328.

[111] T. Cruse, W. Vanburen, Three-dimensional elastic stress analysis of a fracture specimen with an edge crack, International Journal of Fracture Mechanics 7 (1971) 1–15.

[112] G. E. Blandford, A. R. Ingraffea, J. A. Liggett, Two-dimensional stress intensity factor computations using the boundary element method, International Journal for Numerical Methods in Engineering 17 (1981) 387–404.

[113] A. Portela, M. Aliabadi, D. Rooke, The dual boundary element method: Effective implementation for crack problems, International Journal for Numerical Methods in Engineering 33 (1992) 1269–1287.

[114] Z. Q. Yue, H. T. Xiao, Generalized kelvin solution based boundary element method for crack problems in multilayered solids, Engineering Analysis with Boundary Elements 26 (2002) 691–705.

[115] J. P. Wolf, C. Song, Finite-element modelling of unbounded media, Wiley Chichester, 1996.

[116] M. H. Bazyar, C. Song, Transient analysis of wave propagation in non-homogeneous elastic unbounded domains by using the scaled boundary finite-element method, Earthquake Engineering & Structural Dynamics 35 (2006) 1787–1806.

[117] M. H. Bazyar, C. Song, A continued-fraction-based high-order transmitting boundary for wave propagation in unbounded domains of arbitrary geometry, International Journal for Numerical Methods in Engineering 74 (2008) 209–237.

[118] C. Song, J. P. Wolf, Semi-analytical representation of stress singularities as occurring in cracks in anisotropic multi-materials with the scaled boundary finite-element method, Computers & Structures 80 (2002) 183 – 197.

[119] C. Song, Evaluation of power-logarithmic singularities, T-stresses and higher order terms of in-plane singular stress fields at cracks and multi-material corners, Engineering Fracture Mechanics 72 (2005) 1498–1530.

[120] A. J. Deeks, J. P. Wolf, A virtual work derivation of the scaled boundary finite-element method for elastostatics, Computational Mechanics 28 (2002) 489–504.

[121] C. Song, The scaled boundary finite element method in structural dynamics, International Journal for Numerical Methods in Engineering 77 (2009) 1139–1171.

[122] L. Liu, J. Zhang, C. Song, C. Birk, A. A. Saputra, W. Gao, Automatic three-dimensional acoustic-structure interaction analysis using the scaled boundary finite element method, Journal of Computational Physics 395 (2019) 432–460.

[123] E. T. Ooi, C. Song, F. Tin-Loi, Z. J. Yang, Automatic modelling of cohesive crack propagation in concrete using polygon scaled boundary finite elements, Engineering Fracture Mechanics 93 (2012) 13–33.

[124] E. T. Ooi, Z. J. Yang, Modelling dynamic crack propagation using the scaled boundary finite element method, International Journal for Numerical Methods in Engineering 88 (2011) 329–349.

[125] C. Song, E. T. Ooi, S. Natarajan, A review of the scaled boundary finite element method for two-dimensional linear elastic fracture mechanics, Engineering Fracture Mechanics (2017).

[126] W. Xing, C. Song, F. Tin-Loi, A scaled boundary finite element based node-to-node scheme for 2D frictional contact problems, Computer Methods in Applied Mechanics and Engineering 333 (2018) 114–146.

[127] P. Zhang, C. Du, X. Tian, S. Jiang, A scaled boundary finite element method for modelling crack face contact problems, Computer Methods in Applied Mechanics and Engineering 328 (2018) 431–451.

[128] W. Xing, J. Zhang, C. Song, F. Tin-Loi, A node-to-node scheme for three-dimensional contact problems using the scaled boundary finite element method, Computer Methods in Applied Mechanics and Engineering 347 (2019) 928–956.

[129] H. Guo, E. T. Ooi, A. A. Saputra, Z. Yang, S. Natarajan, E. H. Ooi, C. Song, A quadtree-polygon-based scaled boundary finite element method for image-based mesoscale fracture modelling in concrete, Engineering Fracture Mechanics 211 (2019) 420–441.

[130] Z. Zhang, Y. Liu, D. D. Dissanayake, A. A. Saputra, C. Song, Nonlocal damage modelling by the scaled boundary finite element method, Engineering Analysis with Boundary Elements 99 (2019) 29–45.

[131] Y. He, J. Guo, H. Yang, Image-based numerical prediction for effective thermal conductivity of heterogeneous materials: A quadtree based scaled boundary finite element method, International Journal of Heat and Mass Transfer 128 (2019) 335–343.

[132] R. D. Krieg, Unconditional stability in numerical time integration methods, Journal of Applied Mechanics 40 (1973) 417–421.

[133] P. Krysl, T. Belytschko, Object-oriented parallelization of explicit structural dynamics with pvm, Computers & Structures 66 (1998) 259 – 273.

[134] J. Chung, J. M. Lee, A new family of explicit time integration methods for linear and non-linear structural dynamics, International Journal for Numerical Methods in Engineering 37 (1994) 3961–3976.

[135] G. Noh, K. J. Bathe, An explicit time integration scheme for the analysis of wave propagations, Computers & Structures 129 (2013) 178–193.

[136] W. Kim, J. H. Lee, An improved explicit time integration method for linear and nonlinear structural dynamics, Computers & Structures 206 (2018) 42–53.

[137] W. Kim, A simple explicit single step time integration algorithm for structural dynamics, International Journal for Numerical Methods in Engineering 119 (2019) 383–403.

[138] W. Kutta, Beitrag zur naherungsweisen integration totaler differentialgleichungen, Z. Math. Phys. 46 (1901) 435–453.

[139] C. C. Fu, A method for the numerical integration of the equations of motion arising from a finite-element analysis, Journal of Applied Mechanics 37 (1970) 599–605.

[140] I. P. E. Kinnmark, W. G. Gray, Fourth-order accurate one-step integration methods with large imaginary stability limits, Numerical Methods for Partial Differential Equations 2 (1986) 63–70.

[141] W. Kim, Higher-order explicit time integration methods for numerical analyses of structural dynamics, Latin American Journal of Solids and Structures 16 (2019).

[142] K. J. Bathe, G. Noh, Insight into an implicit time integration scheme for structural dynamics, Computers & Structures 98-99 (2012) 1–6.

[143] K. Subbaraj, M. Dokainish, A survey of direct time-integration methods in computational structural dynamics–ii. implicit methods, Computers & structures 32 (1989) 1387–1401.

[144] N. M. Newmark, A method of computation for structural dynamics, Journal of the Engineering Mechanics Division 85 (1959) 67–94.

[145] E. Wilson, I. Farhoomand, K. Bathe, Nonlinear dynamic analysis of complex structures, Earthquake Engineering & Structural Dynamics 1 (1972) 241–252.

[146] J. C. Houbolt, A recurrence matrix solution for the dynamic response of elastic aircraft, Journal of the Aeronautical Sciences 17 (1950) 540–550.

[147] H. M. Hilber, T. J. Hughes, R. L. Taylor, Improved numerical dissipation for time integration algorithms in structural dynamics, Earthquake Engineering & Structural Dynamics 5 (1977) 283–292.

[148] W. Wood, M. Bossak, O. Zienkiewicz, An alpha modification of newmark's method, International Journal for Numerical Methods in Engineering 15 (1980) 1562–1566.

[149] J. Chung, G. Hulbert, A time integration algorithm for structural dynamics with improved numerical dissipation: The generalized-α method, Journal of Applied Mechanics 60 (1993) 271–275.

[150] K. J. Bathe, M. M. I. Baig, On a composite implicit time integration procedure for nonlinear dynamics, Computers & Structures 83 (2005) 2513–2524.

[151] W. Kim, S. Y. Choi, An improved implicit time integration algorithm: The generalized composite time integration algorithm, Computers & Structures 196 (2018) 341–354.

[152] C. Semler, W. Gentleman, M. Paıdoussis, Numerical solutions of second order implicit non-linear ordinary differential equations, Journal of Sound and Vibration 195 (1996) 553–574.

[153] W. Kim, J. Reddy, Effective higher-order time integration algorithms for the analysis of linear structural dynamics, Journal of Applied Mechanics 84 (2017).

[154] W. Kim, J. Reddy, A new family of higher-order time integration algorithms for the analysis of structural dynamics, Journal of Applied Mechanics 84 (2017).

[155] I. S. Duff, A. M. Erisman, J. K. Reid, Direct methods for sparse matrices, Oxford University Press, 2017.

[156] M. T. Heath, P. Raghavan, Performance of a fully parallel sparse solver, The International Journal of Supercomputer Applications and High Performance Computing 11 (1997) 49–64.

[157] X. S. Li, J. Demmel, A scalable sparse direct solver using static pivoting., in: PPSC, Citeseer, 1999.

[158] D. T. Nguyen, Finite element methods: Parallel-sparse statics and eigen-solutions, Springer Science & Business Media, 2006.

[159] G. P. Krawezik, G. Poole, Accelerating the ANSYS direct sparse solver with GPUs, in: Symposium on application accelerators in high performance computing, SAAHPC, 2009.

[160] S. Koric, Q. Lu, E. Guleryuz, Evaluation of massively parallel linear sparse solvers on unstructured finite element meshes, Computers & Structures 141 (2014) 19–25.

[161] M. Papadrakakis, Solving large-scale problems in mechanics: The development and application of computational solution methods, J. Wiley and Sons, 1993.

[162] A. S. Gullerud, X. Gao, R. H. Dodds Jr, R. Haj-Ali, Simulation of ductile crack growth using computational cells: Numerical aspects, Engineering Fracture Mechanics 66 (2000) 65–92.

[163] Y. Saad, Iterative methods for sparse linear systems, SIAM, 2003.

[164] A. R. M. Rao, Mpi-based parallel finite element approaches for implicit nonlinear dynamic analysis employing sparse PCG solvers, Advances in Engineering Software 36 (2005) 181–198.

[165] J. W. Pearson, J. Pestana, Preconditioners for krylov subspace methods: An overview, GAMM-Mitteilungen 43 (2020) e202000015.

[166] M. R. Hestenes, E. Stiefel, et al., Methods of conjugate gradients for solving linear systems, volume 49, NBS Washington, DC, 1952.

[167] G. H. Golub, C. F. Van Loan, Matrix computations, volume 3, JHU press, 2013.

[168] K. J. Bathe, J. Dong, Studies of finite element procedures – the use of adiana-f in fluid flow analyses, Computers & Structures 32 (1989) 499–516.

[169] Y. Saad, Krylov subspace methods for solving large unsymmetric linear systems, Mathematics of Computation 37 (1981) 105–126.

[170] Y. Saad, The lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems, SIAM Journal on Numerical Analysis 19 (1982) 485–506.

[171] Y. Saad, Practical use of some Krylov subspace methods for solving indefinite and nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing 5 (1984) 203–228.

[172] Y. Saad, M. H. Schultz, Conjugate gradient-like algorithms for solving nonsymmetric linear systems, Mathematics of Computation 44 (1985) 417–424.

[173] G. Brussino, V. Sonnad, A comparison of direct and preconditioned iterative techniques for sparse, unsymmetric systems of linear equations, International Journal for Numerical Methods in Engineering 28 (1989) 801–815.

[174] G. F. Carey, B. N. Jiang, Element-by-element linear and nonlinear solution schemes, Communications in Applied Numerical Methods 2 (1986) 145–153.

[175] A. S. Gullerud, R. H. Dodds Jr, MPI-based implementation of a PCG solver using an ebe architecture and preconditioner for implicit, 3-D finite element analysis, Computers & Structures 79 (2001) 553–575.

[176] J. M. Winget, T. J. Hughes, Solution algorithms for nonlinear transient heat conduction analysis employing element-by-element iterative strategies, Computer Methods in Applied Mechanics and Engineering 52 (1985) 711–815.

[177] T. J. Hughes, R. M. Ferencz, J. O. Hallquist, Large-scale vectorized implicit calculations in solid mechanics on a cray x-mp/48 utilizing ebe preconditioned conjugate gradients, Computer Methods in Applied Mechanics and Engineering 61 (1987) 215–248.

[178] J. Erhel, A. Traynard, M. Vidrascu, An element-by-element preconditioned conjugate gradient method implemented on a vector computer, Parallel Computing 17 (1991) 1051–1065.

[179] A. A. Griffith, The phenomena of rupture and flow in solids, Philosophical Transactions of the Royal Society of London Series A 221 (1921) 163–198.

[180] G. R. Irwin, Fracture dynamics, Fracturing of Metals (1947).

[181] Z. P. Bažant, J. Planas, Fracture and size effect in concrete and other quasibrittle materials, Routledge, 2019.

[182] D. Gross, T. Seelig, Fracture mechanics: With an introduction to micromechanics, Springer International Publishing, 2018.

[183] A. Needleman, A continuum model for void nucleation by inclusion debonding, Journal of Applied Mechanics 54 (1987) 525–531.

[184] A. Needleman, An analysis of decohesion along an imperfect interface, International Journal of Fracture 42 (1990) 21–40.

[185] X. P. Xu, A. Needleman, Numerical simulations of fast crack growth in brittle solids, Journal of the Mechanics and Physics of Solids 42 (1994) 1397–1434.

[186] G. T. Camacho, M. Ortiz, Computational modelling of impact damage in brittle materials, International Journal of Solids and Structures 33 (1996) 2899–2938.

[187] Z. J. Yang, D. Proverbs, A comparative study of numerical solutions to non-linear discrete crack modelling of concrete beams involving sharp snap-back, Engineering Fracture Mechanics 71 (2004) 81–105.

[188] Z. J. Yang, A. J. Deeks, Fully-automatic modelling of cohesive crack growth using a finite element–scaled boundary finite element coupled method, Engineering Fracture Mechanics 74 (2007) 2547–2573.

[189] E. Ooi, Z. Yang, Modelling multiple cohesive crack propagation using a finite element–scaled boundary finite element coupled method, Engineering Analysis with Boundary Elements 33 (2009) 915–929.

[190] X. P. Xu, A. Needleman, Numerical simulations of dynamic crack growth along an interface, International Journal of Fracture 74 (1996) 289–324.

[191] A. Needleman, Numerical modeling of crack growth under dynamic loading conditions, Computational Mechanics 19 (1997) 463–469.

[192] G. Camacho, M. Ortiz, Adaptive lagrangian modelling of ballistic penetration of metallic targets, Computer Methods in Applied Mechanics and Engineering 142 (1997) 269–301.

[193] P. Bocca, A. Carpinteri, S. Valente, Mixed mode fracture of concrete, International Journal of Solids and Structures 27 (1991) 1139–1153.

[194] Z. Yang, Fully automatic modelling of mixed-mode crack propagation using scaled boundary finite element method, Engineering Fracture Mechanics 73 (2006) 1711–1731.

[195] A. Khoei, H. Moslemi, M. Sharifi, Three-dimensional cohesive fracture modeling of non-planar crack growth using adaptive FE technique, International Journal of Solids and Structures 49 (2012) 2334–2348.

[196] N. Moes, T. Belytschko, Extended finite element method for cohesive crack growth, Engineering Fracture Mechanics 69 (2002) 813–833.

[197] M. Ortiz, Y. Leroy, A. Needleman, A finite element method for localized failure analysis, Computer Methods in Applied Mechanics and Engineering 61 (1987) 189–214.

[198] T. Belytschko, J. Fish, B. E. Engelmann, A finite element with embedded localization zones, Computer Methods in Applied Mechanics and Engineering 70 (1988) 59–89.

[199] H. R. Lotfi, P. B. Shing, Embedded representation of fracture in concrete with mixed finite elements, International Journal for Numerical Methods in Engineering 38 (1995) 1307–1325.

[200] J. M. Sancho, J. Planas, J. C. Gálvez, E. Reyes, D. A. Cendón, An embedded cohesive crack model for finite element analysis of mixed mode fracture of concrete, Fatigue & Fracture of Engineering Materials & Structures 29 (2006) 1056–1065.

[201] M. G. Tijssens, B. L. Sluys, E. van der Giessen, Numerical simulation of quasi-brittle fracture using damaging cohesive surfaces, European Journal of Mechanics-A/Solids 19 (2000) 761–779.

[202] G. H. Paulino, W. Celes, R. Espinha, Z. J. Zhang, A general topology-based framework for adaptive insertion of cohesive elements in finite element meshes, Engineering with Computers 24 (2008) 59–78.

[203] K. Park, G. H. Paulino, W. Celes, R. Espinha, Adaptive mesh refinement and coarsening for cohesive zone modeling of dynamic fracture, International Journal for Numerical Methods in Engineering 92 (2012) 1–35.

[204] A. Needleman, An analysis of tensile decohesion along an interface, Journal of the Mechanics and Physics of Solids 38 (1990) 289–324.

[205] S. El-Sayed, S. Sridharan, Cohesive layer models for predicting delamination growth and crack kinking in sandwich structures, International Journal of Fracture 117 (2002) 63–84.

[206] C. Yu, M. Ortiz, A. Rosakis, 3D modelling of impact failure in sandwich structures, in: European structural integrity society, volume 32, Elsevier, 2003, pp. 527–537.

[207] S. Li, M. Thouless, A. Waas, J. Schroeder, P. Zavattieri, Use of mode-I cohesive-zone models to describe the fracture of an adhesively-bonded polymer-matrix composite, Composites Science and Technology 65 (2005) 281–293.

[208] Q. Yang, B. Cox, Cohesive models for damage evolution in laminated composites, International Journal of Fracture 133 (2005) 107–137.

[209] X. Lu, M. Ridha, B. Chen, V. Tan, T. Tay, On cohesive element parameters and delamination modelling, Engineering Fracture Mechanics 206 (2019) 278–296.

[210] N. Blal, L. Daridon, Y. Monerie, S. Pagano, Artificial compliance inherent to the intrinsic cohesive zone models: Criteria and application to planar meshes, International Journal of Fracture 178 (2012) 71–83.

[211] D. V. Kubair, P. H. Geubelle, Comparative analysis of extrinsic and intrinsic cohesive models of dynamic fracture, International Journal of Solids and Structures 40 (2003) 3853–3868.

[212] O. Miller, L. Freund, A. Needleman, Energy dissipation in dynamic fracture of brittle materials, Modelling and Simulation in Materials Science and Engineering 7 (1999) 573.

[213] M. L. Falk, A. Needleman, J. R. Rice, A critical evaluation of dynamic fracture simulations using cohesive surfaces, J. Phys. IV 11 (2001) 43–52.

[214] K. D. Papoulia, C.-H. Sam, S. A. Vavasis, Time continuity in cohesive finite element modeling, International Journal for Numerical Methods in Engineering 58 (2003) 679–701.

[215] H. D. Espinosa, P. D. Zavattieri, A grain level model for the study of failure initiation and evolution in polycrystalline brittle materials. Part I: Theory and numerical implementation, Mechanics of Materials 35 (2003) 333–364.

[216] R. de Borst, J. J. Remmers, A. Needleman, Mesh-independent discrete numerical representations of cohesive-zone models, Engineering Fracture Mechanics 73 (2006) 160–177.

[217] A. Mota, J. Knap, M. Ortiz, Fracture and fragmentation of simplicial finite element meshes using graphs, International Journal for Numerical Methods in Engineering 73 (2008) 1547–1570.

[218] H. R. Hertz, Uber die beruhrung fester elastischer korper und uber die harte, Verhandlung des Vereins zur Beforderung des GewerbefleiBes, Berlin (1882) 449.

[219] J. Oden, J. Martins, Models and computational methods for dynamic friction phenomena, Computer Methods in Applied Mechanics and Engineering 52 (1985) 527–634.

[220] A. B. Chaudhary, K. J. Bathe, A solution method for static and dynamic analysis of three-dimensional contact problems with friction, Computers & Structures 24 (1986) 855 – 873.

[221] O. C. Zienkiewicz, R. L. Taylor, The finite element method for solid and structural mechanics, Elsevier, 2005.

[222] D. Peric, D. R. J. Owen, Computational model for 3-D contact problems with friction based on the penalty method, International Journal for Numerical Methods in Engineering 35 (1992) 1289–1309.

[223] N. Kikuchi, J. T. Oden, Contact Problems in Elasticity, Studies in Applied and Numerical Mathematics, Society for Industrial and Applied Mathematics, 1988.

[224] A. Francavilla, O. C. Zienkiewicz, A note on numerical computation of elastic contact problems, International Journal for Numerical Methods in Engineering 9 (1975) 913–924.

[225] T. J. Hughes, R. L. Taylor, J. L. Sackman, A. Curnier, W. Kanoknukulchai, A finite element method for a class of contact-impact problems, Computer Methods in Applied Mechanics and Engineering 8 (1976) 249 – 276.

[226] G. Zavarise, L. De Lorenzis, A modified node-to-segment algorithm passing the contact patch test, International Journal for Numerical Methods in Engineering 79 (2009) 379–416.

[227] T. A. Laursen, Computational contact and impact mechanics: Fundamentals of modeling interfacial phenomena in nonlinear finite element analysis, Springer Science & Business Media, 2013.

[228] G. Zavarise, P. Wriggers, A segment-to-segment contact strategy, Mathematical and Computer Modelling 28 (1998) 497 – 515.

[229] F. Belgacem, P. Hild, P. Laborde, The mortar finite element method for contact problems, Mathematical and Computer Modelling 28 (1998) 263 – 271.

[230] M. A. Puso, T. A. Laursen, A mortar segment-to-segment contact method for large deformation solid mechanics, Computer Methods in Applied Mechanics and Engineering 193 (2004) 601 – 629.

[231] E. Giner, M. Tur, J. E. Tarancón, F. J. Fuenmayor, Crack face contact in X-FEM using a segment-to-segment approach, International Journal for Numerical Methods in Engineering 82 (2010) 1424–1449.

[232] H. Schorn, U. Rode, Numerical simulation of crack propagation from microcracking to fracture, Cement and Concrete Composites 13 (1991) 87–94.

[233] D. J. Stevens, D. Liu, Strain-based constitutive model with mixed evolution rules for concrete, Journal of Engineering Mechanics 118 (1992) 1184–1200.

[234] Z. Wang, A. Kwan, H. Chan, Mesoscopic study of concrete I: Generation of random aggregate structure and finite element mesh, Computers & Structures 70 (1999) 533–544.

[235] P. Wriggers, S. Moftah, Mesoscale models for concrete: Homogenisation and damage behaviour, Finite Elements in Analysis and Design 42 (2006) 623–636.

[236] C. M. López, I. Carol, A. Aguado, Meso-structural study of concrete fracture using interface elements. I: Numerical model and tensile behavior, Materials and Structures 41 (2008) 583–599.

[237] C. M. López, I. Carol, A. Aguado, Meso-structural study of concrete fracture using interface elements. II: Compression, biaxial and brazilian test, Materials and Structures 41 (2008) 601–620.

[238] Y. Huang, Z. Yang, G. Liu, X. Chen, An efficient FE–SBFE coupled method for mesoscale cohesive fracture modelling of concrete, Computational Mechanics 58 (2016) 635–655.

[239] O. Yılmaz, J. F. Molinari, A mesoscale fracture model for concrete, Cement and Concrete Research 97 (2017) 84–94.

[240] W. H. Gerstle, M. Xie, FEM modeling of fictitious crack propagation in concrete, Journal of Engineering Mechanics 118 (1992) 416–434.

[241] J. Ozbolt, H. Reinhardt, Numerical study of mixed-mode fracture in concrete, International Journal of Fracture 118 (2002) 145–162.

[242] R. Vonk, A micromechanical investigation of softening of concrete loaded in compression, Heron (Delft) 38 (1993).

[243] D. Fanella, D. Krajcinovic, A micromechanical model for concrete in compression, Engineering Fracture Mechanics 29 (1988) 49–66.

[244] T. Sadowski, G. Golewski, Effect of aggregate kind and graining on modelling of plain concrete under compression, Computational Materials Science 43 (2008) 119–126.

[245] L.-Y. Zhao, J.-F. Shao, Q. Z. Zhu, Analysis of localized cracking in quasi-brittle materials with a micro-mechanics based friction-damage approach, Journal of the Mechanics and Physics of Solids 119 (2018) 163–187.

[246] E. Schlangen, E. J. Garboczi, Fracture simulations of concrete using lattice models: Computational aspects, Engineering Fracture Mechanics 57 (1997) 319–332.

[247] J. Kozicki, J. Tejchman, Modelling of fracture process in concrete using a novel lattice model, Granular Matter 10 (2008) 377–388.

[248] J. Eliáš, H. Stang, Lattice modeling of aggregate interlocking in concrete, International Journal of Fracture 175 (2012) 1–11.

[249] B. B. Aydin, B. Binici, K. Tuncay, Lattice simulation of concrete compressive behaviour as indirect tension failure, Magazine of Concrete Research (2020) 1–16.

[250] G. Cusatis, Z. P. Bažant, L. Cedolin, Confinement-shear lattice model for concrete damage in tension and compression: I. theory, Journal of Engineering Mechanics 129 (2003) 1439–1448.

[251] V. Tran, F.-V. Donzé, P. Marin, A discrete element model of concrete under high triaxial loading, Cement and Concrete Composites 33 (2011) 936–948.

[252] I. Iturrioz, G. Lacidogna, A. Carpinteri, Experimental analysis and truss-like discrete element model simulation of concrete specimens under uniaxial compression, Engineering Fracture Mechanics 110 (2013) 81–98.

[253] S. Rangari, K. Murali, A. Deb, Effect of meso-structure on strength and size effect in concrete under compression, Engineering Fracture Mechanics 195 (2018) 162–185.

[254] J. Suchorzewski, J. Tejchman, M. Nitka, Discrete element method simulations of fracture in concrete under uniaxial compression based on its real internal structure, International Journal of Damage Mechanics 27 (2018) 578–607.

[255] J. Tejchman, et al., Calculations of fracture process zones on meso-scale in notched concrete beams subjected to three-point bending, European Journal of Mechanics-A/Solids 29 (2010) 746–760.

[256] S. Shahbeyk, M. Hosseini, M. Yaghoobi, Mesoscale finite element prediction of concrete failure, Computational Materials Science 50 (2011) 1973–1990.

[257] E. A. Rodrigues, O. L. Manzoli, L. A. Bitencourt Jr, T. N. Bittencourt, 2D mesoscale model for concrete based on the use of interface element with a high aspect ratio, International Journal of Solids and Structures 94 (2016) 112–124.

[258] C. Hahn, P. Wriggers, An explicit multi-body contact algorithm, in: PAMM: Proceedings in Applied Mathematics and Mechanics, volume 3, Wiley Online Library, 2003, pp. 280–281.

[259] G. Hippmann, An algorithm for compliant contact between complexly shaped bodies, Multibody System Dynamics 12 (2004) 345–362.

[260] S. Ebrahimi, G. Hippmann, P. Eberhard, Extension of the polygonal contact model for flexible multibody systems, International Journal of Applied Mathematics and Mechanics 1 (2005) 33–50.

[261] K. He, S. Dong, Z. Zhou, Multigrid contact detection method, Physical Review E 75 (2007) 036710.

[262] C. Wellmann, C. Lillie, P. Wriggers, A contact detection algorithm for superellipsoids based on the common-normal concept, Engineering Computations (2008).

[263] T. Belytschko, W. K. Liu, B. Moran, K. Elkhodary, Nonlinear finite elements for continua and structures, John Wiley & Sons, 2013.

[264] S. Sitzmann, K. Willner, B. I. Wohlmuth, A dual lagrange method for contact problems with regularized contact conditions, International Journal for Numerical Methods in Engineering 99 (2014) 221–238.

[265] H. B. Khenous, P. Laborde, Y. Renard, Mass redistribution method for finite element contact problems in elastodynamics, European Journal of Mechanics-A/Solids 27 (2008) 918–932.

[266] A. Tkachuk, B. Wohlmuth, M. Bischoff, Hybrid-mixed discretization of elasto-dynamic contact problems using consistent singular mass matrices, International Journal for Numerical Methods in Engineering 94 (2013) 473–493.

[267] A. Idesman, H. Samajder, E. Aulisa, P. Seshaiyer, Benchmark problems for wave propagation in elastic materials, Computational Mechanics 43 (2009) 797–814.

[268] T. Belytschko, M. O. Neal, Contact-impact by the pinball algorithm with penalty and lagrangian methods, International Journal for Numerical Methods in Engineering 31 (1991) 547–572.

[269] I. Huněk, On a penalty formulation for contact-impact problems, Computers & Structures 48 (1993) 193 – 203.

[270] P. Otto, L. De Lorenzis, J. F. Unger, A regularized model for impact in explicit dynamics applied to the split hopkinson pressure bar, Computational Mechanics 58 (2016) 681–695.

[271] P. Otto, L. De Lorenzis, J. F. Unger, Explicit dynamics in impact simulation using a NURBS contact interface, International Journal for Numerical Methods in Engineering 121 (2020) 1248–1267.

[272] M. Q. Nguyen, D. J. Elder, J. Bayandor, R. S. Thomson, M. L. Scott, A review of explicit finite element software for composite impact analysis, Journal of Composite Materials 39 (2005) 375–386.

[273] L. Iannucci, Dynamic delamination modelling using interface elements, Computers & Structures 84 (2006) 1029–1048.

[274] O. Bettinotti, O. Allix, U. Perego, V. Oancea, B. Malherbe, Simulation of delamination under impact using a global–local method in explicit dynamics, Finite Elements in Analysis and Design 125 (2017) 1–13.

[275] C. Farhat, L. Crivelli, A general approach to nonlinear FE computations on shared-memory multiprocessors, Computer Methods in Applied Mechanics and Engineering 72 (1989) 153–171.

[276] P. Jarzebski, K. Wisniewski, R. Taylor, On parallelization of the loop over elements in feap, Computational Mechanics 56 (2015) 77–86.

[277] S. E. Oh, J. W. Hong, Parallelization of a finite element fortran code using OpenMP library, Advances in Engineering Software 104 (2017) 28–37.

[278] R. Couturier, C. Chipot, Parallel molecular dynamics using OpenMP on a shared memory machine, Computer Physics Communications 124 (2000) 49–59.

[279] S. Goedecker, Optimization and parallelization of a force field for silicon using OpenMP, Computer Physics Communications 148 (2002) 124–135.

[280] M. J. Martín, M. Parada, R. Doallo, High performance air pollution simulation using OpenMP, The Journal of Supercomputing 28 (2004) 311–321.

[281] E. Ayguade, M. Gonzalez, X. Martorell, G. Jost, Employing nested OpenMP for the parallelization of multi-zone computational fluid dynamics applications, Journal of Parallel and Distributed Computing 66 (2006) 686–697. IPDPS '04 Special Issue.

[282] P. Pacheco, Parallel programming with MPI, Morgan Kaufmann, 1997.

[283] W. Gropp, S. Huss-Lederman, M. Snir, MPI: The complete reference. The MPI-2 extensions, volume 2, MIT Press, 1998.

[284] B. Patzák, D. Rypl, Z. Bittnar, Parallel explicit finite element dynamics with nonlocal constitutive models, Computers & Structures 79 (2001) 2287–2297.

[285] G. Karniadakis, G. E. Karniadakis, R. M. Kirby II, Parallel scientific computing in C++ and MPI: A seamless approach to parallel algorithms and their implementation, volume 1, Cambridge University Press, 2003.

[286] G. Mahinthakumar, F. Saied, A hybrid MPI-OpenMP implementation of an implicit finite-element code on parallel architectures, The International Journal of High Performance Computing Applications 16 (2002) 371–393.

[287] K. Nakajima, Parallel iterative solvers for finite-element methods using an OpenMP/MPI hybrid programming model on the earth simulator, Parallel Computing 31 (2005) 1048–1065.

[288] X. Guo, M. Lange, G. Gorman, L. Mitchell, M. Weiland, Developing a scalable hybrid MPI/OpenMP unstructured finite element model, Computers & Fluids 110 (2015) 227–234. ParCFD 2013.

[289] A. Klawonn, M. Lanser, O. Rheinbach, H. Stengel, G. Wellein, Hybrid MPI/OpenMP parallelization in feti-dp methods, in: Recent trends in computational engineering-CE2014, Springer, 2015, pp. 67–84.

[290] T. Yamaguchi, Y. Kawase, A. Nagase, S. Ishimura, Performance evaluation of 3-D hybrid parallel finite element method by MPI/OpenMP, Journal of the Japan Society of Applied Electromagnetics and Mechanics 27 (2019) 85–90.

[291] J. Sanders, E. Kandrot, CUDA by example: An introduction to general-purpose GPU programming, Addison-Wesley Professional, 2010.

[292] A. Munshi, B. Gaster, T. G. Mattson, D. Ginsburg, OpenCL programming guide, Pearson Education, 2011.

[293] S. Wienke, P. Springer, C. Terboven, D. an Mey, Openacc–first experiences with real-world applications, in: European conference on parallel processing, Springer, 2012, pp. 859–870.

[294] F. I. Fazanaro, D. C. Soriano, R. Suyama, M. K. Madrid, J. R. de Oliveira, I. B. Muñoz, R. Attux, Numerical characterization of nonlinear dynamical systems using parallel computing: The role of GPUs approach, Communications in Nonlinear Science and Numerical Simulation 37 (2016) 143–162.

[295] D. Zhao, J. Chen, Y. Zheng, Z. Huang, J. Zheng, Fine-grained parallel algorithm for unstructured surface mesh generation, Computers & Structures 154 (2015) 177–191.

[296] Y. Cai, X. Cui, G. Li, W. Liu, A parallel finite element procedure for contact-impact problems using edge-based smooth triangular element and GPU, Computer Physics Communications 225 (2018) 47–58.

[297] A. Karatarakis, P. Metsis, M. Papadrakakis, GPU-acceleration of stiffness matrix calculation and efficient initialization of efg meshless methods, Computer Methods in Applied Mechanics and Engineering 258 (2013) 63–80.

[298] J. M. Frutos, D. H. Perez, Efficient matrix-free GPU implementation of fixed grid finite element analysis, Finite Elements in Analysis and Design 104 (2015) 61–71.

[299] D. Komatitsch, G. Erlebacher, D. Göddeke, D. Michéa, High-order finite-element seismic wave propagation modeling with MPI on a large GPU cluster, Journal of Computational Physics 229 (2010) 7692–7714.

[300] P. Huthwaite, Accelerated finite element elastodynamic simulations using the GPU, Journal of Computational Physics 257 (2014) 687–707.

[301] Y. S. Yang, C. M. Yang, T. J. Hsieh, GPU parallelization of an object-oriented nonlinear dynamic structural analysis platform, Simulation Modelling Practice and Theory 40 (2014) 112–121.

[302] A. Bartezzaghi, M. Cremonesi, N. Parolini, U. Perego, An explicit dynamics GPU structural solver for thin shell finite elements, Computers & Structures 154 (2015) 29–40.

[303] Y. Cai, G. Li, H. Wang, G. Zheng, S. Lin, Development of parallel explicit finite element sheet forming simulation system based on GPU architecture, Advances in Engineering Software 45 (2012) 370–379.

[304] Y. Cai, G. Wang, G. Li, H. Wang, A high performance crashworthiness simulation system based on GPU, Advances in Engineering Software 86 (2015) 29–38.

[305] L. D. Dalcin, R. R. Paz, P. A. Kler, A. Cosimo, Parallel distributed computing using Python, Advances in Water Resources 34 (2011) 1124–1139.

[306] H. A. Schwarz, Ueber einige abbildungsaufgaben. (1869).

[307] J. S. Przemieniecki, Matrix structural analysis of substructures, AIAA Journal 1 (1963) 138–147.

[308] I. Babuška, H. C. Elman, Some aspects of parallel implementation of the finite-element method on message passing architectures, in: Advances in Parallel Computing, volume 1, Elsevier, 1990, pp. 157–187.

[309] G. Nikishkov, A. Makinouchi, G. Yagawa, S. Yoshimura, Performance study of the domain decomposition method with direct equation solver for parallel finite element analysis, Computational mechanics 19 (1996) 84–93.

[310] O. Axelsson, Iterative solution methods, Cambridge university press, 1996.

[311] A. Basermann, B. Reichel, C. Schelthoff, Preconditioned cg methods for sparse matrices on massively parallel machines, Parallel computing 23 (1997) 381–398.

[312] K. Murotani, S.-I. Sugimoto, H. Kawai, S. Yoshimura, Hierarchical domain decomposition with parallel mesh refinement for billions-of-dof scale finite element analyses, International Journal of Computational Methods 11 (2014) 1350061.

[313] P. Kuus, J. Šístek, Coupling parallel adaptive mesh refinement with a nonoverlapping domain decomposition solver, Advances in Engineering Software 110 (2017) 34–54.

[314] Y. Takahashi, K. Fujiwara, T. Iwashita, H. Nakashima, Parallel finite-element analysis of rotating machines based on domain decomposition considering nonconforming mesh connection, IEEE Transactions on Magnetics 52 (2015) 1–4.

[315] A. Korobenko, Y. Bazilevs, K. Takizawa, T. E. Tezduyar, Recent advances in ale-vms and st-vms computational aerodynamic and fsi analysis of wind turbines, Frontiers in Computational Fluid-Structure Interaction and Flow Simulation (2018) 253–336.

[316] C. Farhat, N. Maman, G. W. Brown, Mesh partitioning for implicit computations via iterative domain decomposition: impact and optimization of the subdomain aspect ratio, International Journal for Numerical Methods in Engineering 38 (1995) 989–1000.

[317] C. Farhat, L. Crivelli, F.-X. Roux, A transient feti methodology for large-scale parallel implicit computations in structural mechanics, International Journal for Numerical Methods in Engineering 37 (1994) 1945–1975.

[318] Y. Achdou, Y. Maday, O. Widlund, Iterative substructuring preconditioners for mortar element methods in two dimensions, SIAM journal on numerical analysis 36 (1999) 551–580.

[319] C. R. Dohrmann, A preconditioner for substructuring based on constrained energy minimization, SIAM Journal on Scientific Computing 25 (2003) 246–258.

[320] Y. El Gharbi, A. Parret-Fréaud, C. Bovet, P. Gosselet, Hierarchical substructuring and parallel mesh generation for domain decomposition methods (2020).

[321] A. Matsokin, S. Nepomnyaschikh, On the convergence of the non-overlapping schwartz subdomain alternating method (1989).

[322] S. Nepomnyaschikh, Schwartz alternating method for solving the singular neumann problem (1990).

[323] M. A. Belonosov, S. Solovyev, V. Cheverda, K. Kostov, G. Reshetova, Parallel computations for the simulation of seismic waves on the basis of the additive schwartz method, Numerical methods and programming 13 (2012) 525–535.

[324] T. Yamada, K. Goto, Balancing domain decomposition method on additive schwartz framework for multi-level, Numerical Methods for Flows: FEF 2017 Selected Contributions 132 (2020) 297.

[325] M. Papadrakakis, N. Bitoulas, K. Hatjikonstantinou, An effective superelement-by-superelement solution method for large finite element computations, Computing Systems in Engineering 2 (1991) 535–540.

[326] S. Bitzarakis, M. Papadrakakis, A. Kotsopulos, Parallel solution techniques in computational structural mechanics, Computer methods in applied mechanics and engineering 148 (1997) 75–104.

[327] M. Papadrakakis, S. Bitzarakis, Domain decomposition PCG methods for serial and parallel processing, Advances in Engineering Software 25 (1996) 291–307.

[328] S. Bova, G. Carey, A distributed memory parallel element-by-element scheme for semiconductor device simulation, Computer methods in applied mechanics and engineering 181 (2000) 403–423.

[329] F. Ribeiro, I. Ferreira, Parallel implementation of the finite element method using compressed data structures, Computational Mechanics 41 (2007) 31–48.

[330] D. E. Knuth, Postscript about NP-hard problems, SIGACT News 6 (1974) 15–16.

[331] Y. Li, meshpart, `https://github.com/YingzhouLi/meshpart`, 2018.

[332] G. Karypis, V. Kumar, Multilevelk-way partitioning scheme for irregular graphs, Journal of Parallel and Distributed Computing 48 (1998) 96 – 129.

[333] C. Walshaw, Serial jostle library interface: Version 1.1. 8, Technical Report, Technical report, School of Computing & Mathematical Sciences, University of Greenwich, London, UK, 1999.

[334] Y.-S. Yang, S.-H. Hsieh, Iterative mesh partitioning optimization for parallel nonlinear dynamic finite element analysis with direct substructuring, Computational mechanics 28 (2002) 456–468.

[335] T. Belytschko, E. Plaskacz, H. Y. Chiang, Explicit finite element methods with contact-impact on simd computers, Computing Systems in Engineering 2 (1991) 269 – 276.

[336] T. Belytschko, E. J. Plaskacz, Simd implementation of a non-linear transient shell program with partially structured meshes, International Journal for Numerical Methods in Engineering 33 (1992) 997–1026.

[337] J. Zhang, A. Ankit, H. Gravenkamp, S. Eisenträger, C. Song, A massively parallel explicit solver for elasto-dynamic problems exploiting octree meshes, Computer Methods in Applied Mechanics and Engineering 380 (2021) 113811.

[338] L. Davison, Fundamentals of shock wave propagation in solids, Springer Science & Business Media, 2008.

[339] J. Achenbach, Wave propagation in elastic solids, Elsevier, 2012.

[340] W. Ostachowicz, P. Kudela, M. Krawczuk, A. Zak, Guided waves in structures for SHM: the time-domain spectral element method, John Wiley & Sons, 2011.

[341] S. Gopalakrishnan, M. Ruzzene, S. Hanagud, Application of the finite element method in shm, in: Computational Techniques for Structural Health Monitoring, Springer, 2011, pp. 157–175.

[342] H. Gravenkamp, S. Natarajan, Scaled boundary polygons for linear elastodynamics, Computer Methods in Applied Mechanics and Engineering 333 (2018) 238 – 256.

[343] H. Gravenkamp, A. A. Saputra, C. Song, C. Birk, Efficient wave propagation simulation on quadtree meshes using SBFEM with reduced modal basis, International Journal for Numerical Methods in Engineering 110 (2017) 1119–1141.

[344] E. T. Ooi, Z. J. Yang, Z. Y. Guo, Dynamic cohesive crack propagation modelling using the scaled boundary finite element method, Fatigue & Fracture of Engineering Materials & Structures 35 (2012) 786–800.

[345] E. T. Ooi, S. Natarajan, C. Song, E. H. Ooi, Dynamic fracture simulations using the scaled boundary finite element method on hybrid polygon–quadtree meshes, International Journal of Impact Engineering 90 (2016) 154 – 164.

[346] H. Gravenkamp, C. Song, J. Zhang, On mass lumping and explicit dynamics in the scaled boundary finite element method, Computer Methods in Applied Mechanics and Engineering 370 (2020) 113274.

[347] R. Cook, D. Malkus, M. Plesha, Concepts and applications of finite element analysis, Wiley, 1989.

[348] S. Duczek, H. Gravenkamp, Mass lumping techniques in the spectral element method: On the equivalence of the row-sum, nodal quadrature, and diagonal scaling methods, Computer Methods in Applied Mechanics and Engineering 353 (2019) 516–569.

[349] S. Duczek, H. Gravenkamp, Critical assessment of different mass lumping schemes for higher order serendipity finite elements, Computer Methods in Applied Mechanics and Engineering 350 (2019) 836–897.

[350] S. Filippone, V. Cardellini, D. Barbieri, A. Fanfarillo, Sparse matrix-vector multiplication on gpgpus, ACM Trans. Math. Softw. 43 (2017).

[351] J. Gao, Y. Wang, J. Wang, A novel multi–graphics processing unit parallel optimization framework for the sparse matrix-vector multiplication, Concurrency and Computation: Practice and Experience 29 (2017) e3936. E3936 cpe.3936.

[352] T. J. Hughes, K. S. Pister, R. L. Taylor, Implicit-explicit finite elements in nonlinear transient analysis, Computer Methods in Applied Mechanics and Engineering 17-18 (1979) 159 – 182.

[353] A. Gravouil, T. Elguedj, H. Maigre, An explicit dynamics extended finite element method. part 2: Element-by-element stable-explicit/explicit dynamic scheme, Computer Methods in Applied Mechanics and Engineering 198 (2009) 2318 – 2328.

[354] L. A. de Béjar, K. T. Danielson, Critical time-step estimation for explicit integration of dynamic higher-order finite-element formulations, Journal of Engineering Mechanics 142 (2016) 04016043.

[355] G. M. Amdahl, Validity of the single processor approach to achieving large scale computing capabilities, in: Proceedings of the April 18-20, 1967, spring joint computer conference, 1967, pp. 483–485.

[356] J. L. Gustafson, Fixed time, tiered memory, and superlinear speedup, in: Proceedings of the Fifth Distributed Memory Computing Conference (DMCC5), IEEE Press, 1990, pp. 1255–1260.

[357] J. L. Gustafson, Reevaluating amdahl's law, Communications of the ACM 31 (1988) 532–533.

[358] C. Willberg, S. Duczek, J. Vivar Perez, D. Schmicker, U. Gabbert, Comparison of different higher order finite element schemes for the simulation of Lamb waves, Computer Methods in Applied Mechanics and Engineering 241 (2012) 246–261.

[359] H. Gravenkamp, Numerical Methods for the Simulation of Ultrasonic Guided Waves, volume 116 of *Dissertationsreihe*, BAM-Berlin, 2014.

[360] T. Chamberlin, Terrain2stl, `https://www.jthatch.com/Terrain2STL/`, 2020.

[361] I. A. Viktorov, Rayleigh and Lamb Waves, Ultrasonic Technology, Plenum Press., 1967.

[362] J. L. Rose, Ultrasonic Waves in Solid Media, Cambridge University Press, 1999.

[363] V. Giurgiutiu, Structural Health Monitoring with Piezoelectric Active Wafer Sensors: Fundamentals and Applications, Elsevier, 2008.

[364] S. Gopalakrishnan, M. Ruzzene, S. Hanagud, Computational techniques for structural health monitoring, Springer Science & Business Media, 2011.

[365] F. Mossaiby, M. Joulaian, A. Düster, The spectral cell method for wave propagation in heterogeneous materials simulated on multiple GPUs and CPUs, Computational Mechanics 63 (2019) 805–819.

[366] M. Joulaian, The Hierarchical Finite Cell Method for Problems in Structural Mechanics, VDI Fortschritt-Berichte Reihe 18 Nr. 348, 2017.

[367] J. H. Nieuwenhius, J. Neumann, D. W. Greve, I. J. Oppenheim, Generation and detection of guided waves using PZT wafer transducers, IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control 52 (2005) 2103–2111.

[368] D. W. Greve, J. J. Neumann, J. H. Nieuwenhius, I. J. Oppenheim, N. L. Tyson, Use of Lamb waves to monitor plates: Experiments and simulations, in: Proceedinggs of SPIE 5765, 2005.

[369] M. Ghaemian, A. Ghobarah, Nonlinear seismic response of concrete gravity dams with dam–reservoir interaction, Engineering Structures 21 (1999) 306–315.

[370] A. K. Chopra, Dynamics of structures, Pearson Education India, 2007.

[371] S. A. Mahin, P.-s. B. Shing, Pseudodynamic method for seismic testing, Journal of Structural Engineering 111 (1985) 1482–1503.

[372] T. Bose, D. Choudhury, J. Sprengel, M. Ziegler, Efficiency of open and infill trenches in mitigating ground-borne vibrations, Journal of Geotechnical and Geoenvironmental Engineering 144 (2018) 04018048.

[373] T. Liu, F. Huang, W. Wen, X. He, S. Duan, D. Fang, Further insights of a composite implicit time integration scheme and its performance on linear seismic response analysis, Engineering Structures 241 (2021) 112490.

[374] S. Kontoe, L. Zdravkovic, D. M. Potts, An assessment of time integration schemes for dynamic geotechnical problems, Computers and geotechnics 35 (2008) 253–264.

[375] M. Raju, S. Khaitan, High performance computing using out-of-core sparse direct solvers, International Journal of Mathematical, Physical and Engineering Sciences 3 (2009) 377–383.

[376] P. Sao, R. Vuduc, X. S. Li, A distributed cpu-gpu sparse direct solver, in: European Conference on Parallel Processing, Springer, 2014, pp. 487–498.

[377] J. R. Shewchuk, et al., An introduction to the conjugate gradient method without the agonizing pain, 1994.

[378] H. Gravenkamp, C. Birk, C. Song, Numerical modeling of elastic waveguides coupled to infinite fluid media using exact boundary conditions, Computers & Structures 141 (2014) 36–45.

[379] Y. Huang, Z. Yang, W. Ren, G. Liu, C. Zhang, 3D meso-scale fracture modelling and validation of concrete based on in-situ X-ray computed tomography images using damage plasticity model, International Journal of Solids and Structures 67 (2015) 340–352.

[380] G. Di Mino, M. Giunta, C. M. Di Liberto, Assessing the open trenches in screening railway ground-borne vibrations by means of artificial neural network, Advances in Acoustics and Vibration 2009 (2009).

[381] L. Zhang, J.-T. Wang, Y.-J. Xu, C.-H. He, C.-H. Zhang, A procedure for 3d seismic simulation from rupture to structures by coupling sem and fem, Bulletin of the Seismological Society of America 110 (2020) 1134–1148.

[382] A. Ankit, A sideface traction approach for cohesive and frictional crack growth problems using SBFEM, Computer Methods in Applied Mechanics and Engineering 386 (2021) 114076.

[383] J. G. Rots, R. d. Borst, Analysis of mixed-mode fracture in concrete, Journal of Engineering Mechanics 113 (1987) 1739–1758.

[384] Z. P. Bazant, F. B. Lin, Nonlocal smeared cracking model for concrete fracture, Journal of Structural Engineering 114 (1988) 2493–2510.

[385] E. Yamaguchi, W. Chen, Cracking model for finite element analysis of concrete materials, Journal of Engineering Mechanics 116 (1990) 1242–1260.

[386] M. Xie, W. H. Gerstle, Energy-based cohesive crack propagation modeling, Journal of Engineering Mechanics 121 (1995) 1349–1358.

[387] P. Zhang, C. Du, C. Birk, W. Zhao, A scaled boundary finite element method for modelling wing crack propagation problems, Engineering Fracture Mechanics 216 (2019) 106466.

[388] I. M. May, Y. Duan, A local arc-length procedure for strain softening, Computers & Structures 64 (1997) 297–303.

[389] G. N. Wells, L. J. Sluys, A new method for modelling cohesive cracks using finite elements, International Journal for Numerical Methods in Engineering 50 (2001) 2667–2682.

[390] P. Dumstorff, G. Meschke, Crack propagation criteria in the framework of X-FEM-based structural analyses, International Journal for Numerical and Analytical Methods in Geomechanics 31 (2007) 239–259.

[391] C. Rocco, G. Guinea, J. Planas, M. Elices, Experimental analysis of rupture mechanisms in the brazilian test, AEDIFICATIO Publishers, Fracture Mechanics of Concrete Structures 1 (1998) 121–130.

[392] J. Dolbow, N. Moës, T. Belytschko, An extended finite element method for modeling crack growth with frictional contact, Computer Methods in Applied Mechanics and Engineering 190 (2001) 6825–6846.

[393] P. M. Schubel, J.-J. Luo, I. M. Daniel, Impact and post impact behavior of composite sandwich panels, Composites Part A: applied science and manufacturing 38 (2007) 1051–1057.

[394] D. Feng, F. Aymerich, Damage prediction in composite sandwich panels subjected to low-velocity impact, Composites Part A: Applied Science and Manufacturing 52 (2013) 12–22.

[395] S. Long, X. Yao, H. Wang, X. Zhang, Failure analysis and modeling of foam sandwich laminates under impact loading, Composite Structures 197 (2018) 10–20.

[396] L. De Lorenzis, D. Fernando, J. G. Teng, Coupled mixed-mode cohesive zone modeling of interfacial debonding in simply supported plated beams, International Journal of Solids and Structures 50 (2013) 2477–2494.

[397] S. Pinho, L. Iannucci, P. Robinson, Formulation and implementation of decohesion elements in an explicit finite element code, Composites Part A: Applied science and manufacturing 37 (2006) 778–789.

[398] W. Cui, M. Wisnom, M. Jones, A comparison of failure criteria to predict delamination of unidirectional glass/epoxy specimens waisted through the thickness, Composites 23 (1992) 158–166.

[399] V. K. Goyal, E. R. Johnson, C. G. Davila, Irreversible constitutive law for modeling the delamination process using interfacial surface discontinuities, Composite Structures 65 (2004) 289–305.

[400] J. R. Reeder, 3-D mixed mode delamination fracture criteria-an experimentalist's perspective (2006).

[401] J. Li, J. Sen, Analysis of frame-to-skin joint pull-off tests and prediction of the delamination failure, in: 19th AIAA Applied Aerodynamics Conference, 2001, p. 1484.

[402] J. Zhang, C. Song, A polytree based coupling method for non-matching meshes in 3D, Computer Methods in Applied Mechanics and Engineering 349 (2019) 743–773.

[403] G. Cowper, Gaussian quadrature formulas for triangles, International Journal for Numerical Methods in Engineering 7 (1973) 405–408.

[404] N. Stein, P. Weißgraeber, W. Becker, A model for brittle failure in adhesive lap joints of arbitrary joint configuration, Composite Structures 133 (2015) 707–718.

[405] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, SIAM Journal on scientific Computing 20 (1998) 359–392.

[406] A. Pramod, R. Annabattula, E. Ooi, C. Song, S. Natarajan, et al., Adaptive phase-field modeling of brittle fracture using the scaled boundary finite element method, Computer Methods in Applied Mechanics and Engineering 355 (2019) 284–307.

[407] W. Zhang, Z. Xiao, C. Liu, Y. Mei, S.-k. Youn, X. Guo, A scaled boundary finite element based explicit topology optimization approach for three-dimensional structures, International Journal for Numerical Methods in Engineering 121 (2020) 4878–4900.

[408] A. Egger, A. Saputra, S. P. Triantafyllou, E. Chatzi, Exploring topology optimization on hierarchical meshes by scaled boundary finite element method, in: Proceedings of the International Conference on Computational Methods, volume 6, Scientech Publisher LLC, 2019, pp. 479–492.

[409] W.-M. Hwu, C. Rodrigues, S. Ryoo, J. Stratton, Compute unified device architecture application suitability, Computing in Science & Engineering 11 (2009) 16–26.