Multiple Grid Multiple Time-Scale (MGMT) Simulations in Linear Structural Dynamics

by Tejas Ruparel

B.E. in Mechanical Engineering, August 2006, University of Pune
M.S. in Mechanical Engineering, May 2010, Worcester Polytechnic Institute

A Dissertation submitted to

The Faculty of
School of Engineering and Applied Science
of The George Washington University
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

January 31, 2015

Dissertation directed by

Azim Eskandarian
Professor of Engineering and Applied Science

James Lee
Professor of Mechanical and Aerospace Engineering

The School of Engineering and Applied Science of The George Washington University certifies that Tejas Ruparel has passed the Final Examination for the degree of Doctor of Philosophy as of December 15, 2014. This is the final and approved form of the dissertation.

Multiple Grid Multiple Time-Scale (MGMT) Simulations in Linear Structural Dynamics

Tejas Ruparel

Dissertation Research Committee:

Azim Eskandarian, Professor of Engineering and Applied Science, Dissertation Co-Director

James Lee, Professor of Mechanical and Aerospace Engineering, Dissertation Co-Director

Majid Manzari, Professor of Civil and Environmental Engineering, Committee Member

Tianshu Li, Assistant Professor of Civil and Environmental Engineering, Committee Member

Kunik Lee, General Research Engineer (Chief Safety Scientist), Operation Research & Development, Turner-Fairbank Highway Research Center, FHWA/USDOT, Committee Member

Dedication

I dedicate this dissertation to my loving parents for their unwavering support, encouragement and constant love over the years.

Acknowledgements

Over the past four years I have received support and encouragement from a great number of individuals to whom I am very thankful and I gratefully acknowledge.

First and foremost, I would like to express my sincere gratitude to Prof. Azim Eskandarian, who has been my research advisor and supervisor since the beginning of my study. Prof. Eskandarian has constantly encouraged and motivated me to face and overcome challenges that this work presented along the way. He has helped me shape my career and grow professionally, and I am immensely grateful for the knowledge and skills I gained from him.

I would also like to express my gratitude to Prof. James D. Lee for his research supervision, academic support and constant motivation. He has always reinforced my belief in a deterministic Universe by pointing out how everything is a consequence of design and not just a mere coincidence. I am also thankful to him for teaching me mechanics.

My keen appreciation goes to Prof. Majid Manzari for teaching me the Finite Element Method, he is an incredible teacher. He has helped me fortify my understanding in mechanics and has always inspired me to pursue challenging problems with patience and critical thinking.

I would also like to thank the committee members for their time and invaluable suggestions to improve the dissertation.

Special thanks to Mohamed ElGhoraiby for sharing his knowledge and experience in structural mechanics, Finite Element Analysis, FORTRAN and GiD. He would always engage me in technical discussions and provide me with insightful feedback on a variety of topics. He has been a great colleague and an even better friend.

I would also like to thank Mr. Enrique Escolano from International Center for Numerical Methods in Engineering (CIMNE) for his continual and prompt support on GiD, Prof. Alain Combescure from LaMCoS INSA de Lyon and Prof. Arun Prakash from Purdue University for their valuable suggestions.

Many thanks also go to the CEE department and ISO staff, especially Doreen Walters-Brooks and Joyce Randolph (aka J), for their assistance with the logistics over the years and for making my experience at GWU a pleasant one.

I cannot thank my family enough. Without their care and nurturing, I would not be where I am today. I thank my parents, Meena and Harshad Ruparel, and my brother Siddharth Ruparel for their unconditional love and support. I owe much of my success to them. A very special thanks to my girlfriend Melanie Lynn Carlson, who loved and supported me during the final, critical months of my dissertation, and made me feel like anything was possible.

Last but not the least; I would like to thank Dr. Kunik Lee from the U.S. Department of Transportation and Federal Highway Administration for his continual support. This dissertation is part of a research project 'Nano Material Modeling and Simulation by New Multiple Length/Time Scale Theories and Algorithms' (Grant No: DTFH61-10-H-00005).

Abstract

Multiple Grid Multiple Time-Scale (MGMT)
Simulations in Linear Structural Dynamics

The work presented in this dissertation describes a general algorithm and its Finite Element (FE) implementation for performing concurrent multiple sub-domain simulations in linear structural dynamics. Using this approach one can solve problems in which the domain under analysis can be selectively discretized spatially and temporally, hence allowing the user to obtain a desired level of accuracy in critical regions whilst improving computational efficiency globally. The mathematical background for this approach is largely derived from the fundamental principles of Domain Decomposition Methods (DDM) and Lagrange Multipliers, used to obtain coupled equations of motion for distinct regions of a continuous domain. These methods when combined together systematically yield constraint forces that not only ensure conservation of energy, but also enforce continuity of field quantities across sub-domain interfaces. Multiple Grid (MG) coupling between conforming or non-conforming sub-domains is achieved in the form of linear multi-point constraints that are modeled using Mortar Finite Element Method (M-FEM); whereas coupled Multiple Time-scale (MT) equations are derived for the classical Newmark integration scheme and its constituent algorithms. A rigorous proof of stability is provided using Energy Method and necessary conditions for enforcing energy balance are discussed in reference with field variables that are selected to enforce sub-domain interface continuity. Fully discretized equations of motion for component sub-domains, augmented with an interface continuity condition are then solved using block elimination method and Crout factorization. A step-by-step solution approach, utilizing recursive black box sub-routines, is modeled in order to allow efficient implementation within existing finite element frameworks.

Proposed MGMT Method and corresponding solution algorithm is systematically implemented by using the finite element approach and programming in FORTRAN 90. Resulting in-house code – FEAPI (Finite Element Analysis Programming Interface) is capable of solving linear structural dynamics problems that are modeled using independently discretized sub-domains. Auxiliary sub-routines for defining pre simulation parameters and for viewing global/component sub-domain results are built into FEAPI and work in conjugation with GiD; a universal, adaptive and user-friendly pre and post-processor. Overall stability, numerical accuracy and computational efficiency of MGMT Method is evaluated and verified using a series of benchmark examples. Verification matrices take into consideration performance evaluation factors such as energy balance (at global and component-sub-domain levels), interface continuity, evolution/distribution of kinematic quantities and propagation of structural waves across connecting sub-domains. Assessment of computational efficiency is derived by comparing the size of respective FE problems (nodes, elements, number of equations, skyline storage requirements) and the required computation times (CPU solution time). Discussed examples highlight the greatest advantage of MGMT Method; which is significant gain in simulation speedups (at the cost of reasonably small errors).

Table of Contents

Dedication iv

Acknowledgements v

Abstract vii

Table of Contents ix

List of Figures xiv

List of Tables xxiii

List of Acronyms xxviii

Chapter 1: Introduction 1

1.1 Problem Statement and Motivation 1

1.2 Goals and Objectives 4

1.3 Contributions of the Dissertation 5

1.4 Dissertation Organization 6

Chapter 2: Review of Literature and Theoretical Foundation 9

2.1 Governing Equations in Linear Structural Dynamics 9

2.1.1 Kinematics 9

2.1.2 Balance Laws and Equilibrium 13

2.1.3 Constitutive Model 16

2.2 Finite Element Implementation and Spatial Discretization 18

2.3 Temporal Discretization and Direct Integration Methods 25

2.3.1 Explicit Methods 25

2.3.2 Implicit Methods 28

2.3.3 Generalized-α Method 30
2.4 Domain Decomposition Methods (DDM) 35
2.4.1 Node-cut Partitioning 37
2.4.2 Element-cut Partitioning 38
2.5 Coupling Sub-Domains in Space and Time 41
2.5.1 Mortar Finite Element Method (M-FEM) 41
2.5.2 Finite Element Tearing and Interconnecting (FETI) 46
2.5.3 Mixed Methods and Subcycling 48
2.5.4 GC Method 52
2.5.5 PH Method 55
Chapter 3: MGMT Formulation 63
3.1 Approach and Methodology 63
3.2 Domain Decomposition for Structural Dynamics 65
3.3 Multiple Grid Coupling 69
3.3.1 Coupling Conforming Grids 70
3.3.2 Coupling Non-Conforming Grids 71
3.4 Multiple Time-Scale Coupling 75
3.4.1 Newmark Time Integration 75
3.4.2 Interface Condensation 78
3.5 Stability Analysis Using Energy Method 86
3.6 Solution Algorithm and its Finite Element Implementation 93
3.6.1 Step 1: Solve Av = b 95
3.6.2 Step 2: Compute δ = d – cv 98

3.6.3 Step 3: Solve Aw = f 98

3.6.4 Step 4: Compute $y = \delta^{-1}(g - cw)$ 100

3.6.5 Step 5: Compute x = w – vy........100

Chapter 4: Programming the MGMT Method........103

4.1 Finite Element Analysis Programming Interface (FEAPI) 103

4.1.1 Program Structure and Component Interfaces 106

4.1.2 Program Installation 117

4.1.3 Driver Programs 119

Chapter 5: Numerical Analysis and Verification........121

5.1 Benchmark Case Descriptions 122

5.1.1 Case 1 – Uniform grid uniform time-step (UGUT) reference cases 122

5.1.2 Case 2 – Multiple Grid Multiple Time-Step I-I Coupling (MGMT1)........124

5.1.3 Case 3 – Multiple Grid Multiple Time-Step I-E Coupling (MGMT2)........125

5.1.4 Case 4 – Multiple Time-Step I-I Coupling (MTC) 126

5.1.5 Case 5 – Multiple Grid Coupling with Implicit Time Integration (MGC)....127

5.2 UGUT Convergence........128

5.3 Stability Analysis........134

5.3.1 Global Energy Balance 135

5.3.2 Augmented Interface Energy 138

5.3.3 Interface Continuity........145

5.4 Evaluation of Numerical Accuracy........150

5.4.1 Example 1: Transverse Vibrations........150

5.4.2 Example 2: Longitudinal Vibrations 155

5.5 Evaluation of Computational Efficiency 169

5.6 Numerical Analysis and Verification Summary 170

Chapter 6: MGMT Example Problems and Results 174

6.1 Example 1: Stress Resolution in Critical Regions 176

6.1.1 Example 1.1: Plate with a Hole 176

6.1.2 Example 1.2: 3 Point Bending Test 185

6.2 Example 2: Analysis of Heterogeneous Structural Systems 191

6.3 Example 3: Steel Girder Subjected to Impulse Loading 203

6.4 Example 4: Curved Frame under Point Loading 217

6.5 Example 5: Bridge Analysis 226

Chapter 7: Conclusions and Future Directions 238

7.1 Conclusions 238

7.2 Future Directions 241

Bibliography 243

Appendix A: FEAPI Input File 249

A.1 Example Input File 249

A.2 Input File Data Blocks 250

Appendix B: FEAPI Output Files 257

B.1 Post Mesh File 257

B.2 Post Result Files 257

B.3 Comma Separated Value File 257

B.4 Simulation Summary File 257

Appendix C: Pre/Post Example 260

C.1 Program 1: Uniform Grid Uniform Time-scale Simulations 260

C.1.1 Pre-processing (Input File Creation)..260

C.1.2 Solver 264

C.1.3 Post-processing (Result Visualization) 264

C.2 Program 2: Multiple Grid Multiple Time-scale Simulations................................266

C.2.1 Pre-processing (Input File Creation)..266

C.2.2 Solver 273

C.2.3 Post-processing (Result Visualization) 274

Appendix D: Library Routines 276

D.1 FEAPI..276

D.1.1 Modules 276

D.1.2 Library – FEA..284

D.1.3 Library – MATH 307

D.1.4 Library – Program 310

D.2 FEAPI–GiD 326

D.2.1 Preprocessor...326

D.2.2 Postprocessor...329

D.2.3 GiDPost..334

D.3 MGMT 337

D.3.1 Modules 337

D.3.2 Library 338

List of Figures

Figure 1-1: Proposed multiscale approach 3

Figure 2-1: Motion and deformation of a point in continuum (Chen et al. 2000) 9

Figure 2-2: (a) Surface traction and (b) Internal body force 14

Figure 2-3: Discretization in space using Finite Elements 19

Figure 2-4: Static mesh refinement using transition elements 23

Figure 2-5: Adaptive mesh refinement (h-refinement) (a) Initial discretization: Mesh 1 (b) Mesh 2 and (c) Mesh 3 (Zienkiewicz et al. 2005) 23

Figure 2-6: Pseudo-code for direct time integration using Generalized-α Method acceleration form 34

Figure 2-7: Domain decomposition method 35

Figure 2-8: Node-cut grid partitioning 37

Figure 2-9: Element-cut grid partitioning 38

Figure 2-10: (a) Non-overlapping DD with independent (non-conforming) discretization in component sub-domains (b) Interface (Mortar) elements and corresponding Lagrange Multiplier space 41

Figure 2-11: Direct subdomain connection using global Lagrange Multipliers (Park & Felippa 2000) 43

Figure 2-12: Localized-Multiplier FEM discretization (Park & Felippa 2000) 43

Figure 2-13: Interface definition (Herry et al. 2002) 44

Figure 2-14: Finite element and interconnecting with Lagrange polynomial function 47

Figure 2-15: Comparison between polynomial and mortar Lagrange Multipliers (Lacour & Maday 1997) 48

Figure 2-16: Element time partitioning 49

Figure 2-17: Nodal time partitioning 49

Figure 2-18: Treatment of incompatible meshes using two-grid approach (Gravouil & Combescure 2003) 53

Figure 2-19: (a) Structural domain under consideration (b) Decomposed sub-domains linked through Lagrange Multipliers at the interface (c) sub-domain time-stepping parameters and intermediate time-step counter 56

Figure 2-20: Comparison between (a) GC Method and (b) PH Method 62

Figure 3-1: Hybrid MGMT approach 64

Figure 3-2: Structural domain decomposition (a) Domain under analysis (b) Decomposed sub-domains with inherited boundary conditions and augmented interface reactions 65

Figure 3-3: Domain decomposition and resulting conforming/non-conforming grids 69

Figure 3-4: Interface of Lagrange Multipliers (conforming grids) 70

Figure 3-5: Reference sub-domain and corresponding Boolean projection matrix 71

Figure 3-6: Interface of Lagrange Multipliers (non-conforming grids) and corresponding shape functions 71

Figure 3-7: Reference sub-domains, interface of Lagrange Multipliers and corresponding shape functions 72

Figure 3-8: Interface integration grid 74

Figure 3-9: Pseudo-code for direct time integration using Newmark Method acceleration form 77

Figure 3-10: MGMT time-stepping 79

Figure 3-11: Comparison between (a) PH Method and (b) MGMT Method 85

Figure 3-12: Pseudo-code for computing unit load response matrix for component sub-domains 97

Figure 3-13: Pseudo-code for computing sub-domain response under the action of (applied) external forces and known interface reactions 99

Figure 4-1: Programming the Finite Element Method (Smith et al. 2013) 103

Figure 4-2: PFEM (UGUT) and FEAPI (UGUT/MGMT) program components 104

Figure 4-3: GiD – Universal, adaptive and user-friendly pre- and post-processor 105

Figure 4-4: Finite Element Analysis Programming Interface 106

Figure 4-5: FEAPI code structure 107

Figure 4-6: FEAPI interface sub-components – Library and Modules 107

Figure 4-7: FEAPI interface sub-components – Library > FEA 109

Figure 4-8: FEAPI interface sub-components – FEAPI > Library > MATH 109

Figure 4-9: FEAPI interface sub-components – FEAPI > Library > Program 109

Figure 4-10: MGMT interface sub-components – Library and Modules 111

Figure 4-11: FEAPI-GiD interface sub-components – Preprocessor 112

Figure 4-12: feapi-gid.gid interface sub-components..113

Figure 4-13: Input file creation flowchart..116

Figure 5-1: Forced vibrations analysis of a cantilever beam...121

Figure 5-2: Case 1 (UGUT) domain grids 122

Figure 5-3: Case 2 (MGMT1) sub-domain grids..124

Figure 5-4: Case 2 sub-domain time increments 125

Figure 5-5: Case 3 (MGMT2) sub-domain grids..125

Figure 5-6: Case 4 (MTC) sub-domain grids..126

Figure 5-7: Case 4 (MTC) time-step ratios 127

Figure 5-8: Case 5 (MGC) sub-domain grids 127

Figure 5-9: Example 1 – Vertical displacement at free end (x = 10m) 130

Figure 5-10: Example 1 – Convergence of maximum vertical displacement at free end (x = 10m) 130

Figure 5-11: Example 2 – Horizontal displacement at free end (x = 10m)...131

Figure 5-12: Example 2 – Convergence of maximum horizontal displacement at free end (x = 10m) 131

Figure 5-13: Total number of equations and CPU solution time for Implicit UGUT cases 133

Figure 5-14: Example 1 – Global energies..135

Figure 5-15: Example 2 – Global energies..136

Figure 5-16: Example 1 – Augmented interface energies from component sub-domains 140

Figure 5-17: Example 2 – Augmented interface energies from component sub-domains 141

Figure 5-18: Example 1 – Augmented (total) interface energy 143

Figure 5-19: Example 2 – Augmented (total) interface energy 144

Figure 5-20: Example 1 – (MGMT1) Continuity of interface variables 145

Figure 5-21: Example 1 – (MGMT2) Continuity of interface variables 146

Figure 5-22: Example 2 – (MTC4) Continuity of interface variables 147

Figure 5-23: Example 2 – (MGC3) Continuity of interface variables 148

Figure 5-24: Example 1 – Comparison of vertical displacement at free end (x = 10m) 150

Figure 5-25: Example 1 – Comparison of Sigma-xx at a cross-section x = 1m and t = 0.05s 153

Figure 5-26: Example 1 – Deformed shape and Sigma-xx (N/m^2) at t = 0.05s 154

Figure 5-27: Example 2 – Comparison of horizontal displacement at free end (x = 10m) 155

Figure 5-28: Example 2 – Longitudinal vibrations 158

Figure 5-29: Example 2 – Longitudinal stress (Sigma-xx) as a function of space 159

Figure 5-30: Example 2 – Longitudinal displacement (U-x) as a function of space 161

Figure 5-31: Example 2 – Longitudinal stress (Sigma-xx) as a function of time 163

Figure 5-32: Example 2 – Longitudinal displacement (U-x) as a function of time 164

Figure 5-33: Example 2 – Longitudinal velocity ($\dot{U}$-x) as a function of time 165

Figure 5-34: Example 2 – Longitudinal acceleration ($\ddot{U}$-x) as a function of time 166

Figure 5-35: Summary of solved cases 170

Figure 6-1: Plate with a hole under uniaxial loading 176

Figure 6-2: Theoretical stress concentration factor K_t (Budynas & Nisbett 2008) 177

Figure 6-3: Sigma-xx as a function of space 178

Figure 6-4: Analyzed grids 179

Figure 6-5: Sigma-xx as a function of distance from the hole (along the width of the plate) 180

Figure 6-6: Sigma-xx as a function of distance from the hole (along the length of the plate) 181

Figure 6-7: Contour plots for Sigma-xx, Sigma-yy and Displacement-y around the hole 182

Figure 6-8: Contour plots for Sigma-xx and Sigma-yy outside the critical region 183

Figure 6-9: Distribution of element integration points 183

Figure 6-10: Global energies for UGUT3 v/s MGMT and augmented (total) interface energy 184

Figure 6-11: 3 point bending test 186

Figure 6-12: Domain grids: Transition Mesh and Structured Multiple Grids 187

Figure 6-13: Mesh quality..187

Figure 6-14: Stress as a function of space 188

Figure 6-15: Von Mises Stress (at t = 0.1s) as a function of beam height 189

Figure 6-16: Longitudinal Stress (at t = 0.1s) as a function of beam height 189

Figure 6-17: Von Mises stress distribution t = 0.1s 190

Figure 6-18: Contour lines for Displacement-x at t = 0.1s 190

Figure 6-19: Contour lines for Displacement-y at t = 0.1s 190

Figure 6-20: Heterogeneous material system...191

Figure 6-21: Heterogeneous material system – Domain grids 194

Figure 6-22: Global energies 195

Figure 6-23: Heterogeneous material system – Longitudinal stress and displacement measurement locations across the length of beam 196

Figure 6-24: Heterogeneous material system – Longitudinal stress (Sigma-xx) as a function of time 197

Figure 6-25: Longitudinal displacement (U-x) as a function of time.................................197

Figure 6-26: Longitudinal stress (Sigma-xx) as a function of space..................................199

Figure 6-27: Longitudinal displacement (U-x) as a function of space...............................199

Figure 6-28: Longitudinal stress wave at t = 0.00075s...200

Figure 6-29: Longitudinal stress wave at t = 0.0015s...200

Figure 6-30: Longitudinal stress wave at t = 0.00225s...201

Figure 6-31: Longitudinal stress wave at t = 0.003s 201

Figure 6-32: Steel girder subjected to short duration impulse loading 203

Figure 6-33: Analyzed cases: UGUT, MGMT1 and MGMT2 204

Figure 6-34: Global Energies....206

Figure 6-35: Analyzed result descriptions (a) Results measured at points A, B and C as functions of time (b) Results measured along segment D as functions of space....207

Figure 6-36: Global contour lines for Displacement-x at peak impulse. UGUT, MGMT1 and MGMT2 (L-R)....208

Figure 6-37: Global contour lines for Displacement-y at peak impulse. UGUT, MGMT1 and MGMT2 (L-R)....209

Figure 6-38: Global contour plots for Sigma-xx. UGUT, MGMT1 and MGMT2 (L-R) 210

Figure 6-39: Global contour plots for Sigma-yy. UGUT, MGMT1 and MGMT2 (L-R) 211

Figure 6-40: Displacement-x as a function of time....212

Figure 6-41: Sigma-xx as a function of time....213

Figure 6-42: Displacement-x as a function of space....214

Figure 6-43: Sigma-xx as a function of space....215

Figure 6-44: Interface disconnection (Deformed shape graphed at 5E7 magnification)....216

Figure 6-45: (a) Domain under analysis: Curved frame (b) Transient point loading....217

Figure 6-46: Finite element modeling of a double symmetric domain under symmetric loading condition 217

Figure 6-47: Analyzed cases 218

Figure 6-48: Global energies 220

Figure 6-49: Kinematic conformity and interface continuity (N1) 221

Figure 6-50: Kinematic conformity (N2) 222

Figure 6-51: Contour plots for Displacement-x. UGUT (Left) and MGMT (Right)...223

Figure 6-52: Contour plots for Displacement-y. UGUT (Left) and MGMT (Right)...224

Figure 6-53: Bridge analysis: Domain description and time proportional load function 227

Figure 6-54: Global energies 229

Figure 6-55: Augmented interface energies 230

Figure 6-56: Kinematic conformity and interface continuity 231

Figure 6-57: Displacement-x at t=0.5sec (Deformed shape graphed at x50)...233

Figure 6-58: Displacement-y at t=2.5sec (Deformed shape graphed at x50)...234

Figure 6-59: Sigma-xx at t=4.5sec (Deformed shape graphed at x50) 235

Figure 6-60: Sigma-yy at t=4.5sec (Deformed shape graphed at x50) 236

Figure A-1: (a) Example problem (b) Transient (linear) loading......................................249

Figure C-1: (a) Domain under consideration – 2D cantilever beam (b) Transverse (step) loading 260

Figure C-2: Decomposed sub-domains and corresponding time-stepping parameters 266

List of Tables

Table 2-1: Labels for sub-domain and interface degrees of freedom (DOF) 42

Table 3-1: Labels for FE degrees of freedom 69

Table 3-2: Sub-domain time-stepping parameters 78

Table 4-1: FEAPI material library 114

Table 5-1: Case 1 (UGUT) simulation parameters 122

Table 5-2: Case 2 (MGMT1) simulation parameters 124

Table 5-3: Case 3 (MGMT2) simulation parameters 125

Table 5-4: Case 4 (MTC) simulation parameters 126

Table 5-5: Case 5 (MGC) simulation parameters 128

Table 5-6: RMSE and NRMSE between respective UGUT cases 132

Table 5-7: Total number of equations and CPU solution time 133

Table 5-8: Example 1 – RMSE and NRMSE (%). Variable = Global energies 137

Table 5-9: Example 2 – RMSE and NRMSE (%). Variable = Global energies 137

Table 5-10: RMSE and NRMSE (%). Variable = Internal energy v/s External work 138

Table 5-11: Mean variance in augmented (total) interface energy contributions 142

Table 5-12: Example 1 – (MGMT1) RMSE and NRMSE (%). Variable = Interface displacement, velocity and accelerations 145

Table 5-13: Example 1 – (MGMT2) RMSE and NRMSE (%). Variable = Interface displacement, velocity and accelerations 146

Table 5-14: Example 2 – (MTC4) RMSE and NRMSE (%). Variable = Interface displacement, velocity and accelerations 147

Table 5-15: Example 2 – (MGC3) RMSE and NRMSE (%). Variable = Interface displacement, velocity and accelerations 148

Table 5-16: Example 1 – RMSE and NRMSE (%). Variable = displacement at free end (x = 10m) 151

Table 5-17: Example 1 – Comparison between UGUT2 and MTC cases 152

Table 5-18: Example 1 – Comparison between UGUT3 and MGMT1 152

Table 5-19: Example 1 – RMSE and NRMSE (%). Variable = Sigma-xx at x = 1m and t = 0.05s 153

Table 5-20: Example 2 – RMSE and NRMSE (%). Variable = displacement at free end (x = 10m) 156

Table 5-21: Example 2 – Comparison between UGUT2 and MTC cases 157

Table 5-22: Example 2 – Comparison between UGUT3 and MGMT1 157

Table 5-23: Example 2 – RMSE and NRMSE (%). Variable = Sigma-xx as a function of space 160

Table 5-24: Example 2 – RMSE and NRMSE (%). Variable = U-x as a function of space 162

Table 5-25: Example 2 – RMSE and NRMSE (%). Variable = Longitudinal stress (Sigma-xx) 167

Table 5-26: Example 2 – RMSE and NRMSE (%). Variable = Longitudinal displacement (U-x) 167

Table 5-27: Example 2 – RMSE and NRMSE (%). Variable = Longitudinal velocity (U̇-x) 167

Table 5-28: Example 2 – RMSE and NRMSE (%). Variable = Longitudinal acceleration (Ü-x) 167

Table 5-29: Example 2 – Average errors and corresponding rankings 168

Table 5-30: Comparison of computational resources for Example 1 and Example 2 169

Table 5-31: Summary of analyzed results 171

Table 5-32: Summary of computational efficiency 173

Table 6-1: Case parameters 179

Table 6-2: Example 1.1 – Comparison of computational resources 185

Table 6-3: Simulation parameters 187

Table 6-4: Mechanical properties for Steel and Tin 192

Table 6-5: Time-step to accurately resolve structural wave propagation 192

Table 6-6: UGUT heterogeneous material system – Simulation parameters 194

Table 6-7: MGMT1 heterogeneous material system – Simulation parameters 194

Table 6-8: MGMT2 heterogeneous material system – Simulation parameters 194

Table 6-9: RMSE and NRMSE (%). Variable = Global energies 196

Table 6-10: Mean variance in augmented (total) interface energy 196

Table 6-11: RMSE and NRMSE (%). Variable = Longitudinal stress (Sigma-xx) as function of time 198

Table 6-12: RMSE and NRMSE (%). Variable = Longitudinal displacement (U-x) as function of time 198

Table 6-13: Example 2 – Comparison of computational resources 202

Table 6-14: Simulation parameters 205

Table 6-15: RMSE and NRMSE (%). Variable = Global Energies 205

Table 6-16: RMSE and NRMSE (%). Variable = Displacement-x as a function of time 212

Table 6-17: RMSE and NRMSE (%). Variable = Sigma-xx as a function of time 213

Table 6-18: RMSE and NRMSE (%). Variable = Displacement-x as a function of space 214

Table 6-19: RMSE and NRMSE (%). Variable = Displacement-x as a function of space 215

Table 6-20: Example 3 – Comparison of computational resources 216

Table 6-21: Simulation parameters 218

Table 6-22: RMSE and NRMSE (%). Variable = Global Energies 220

Table 6-23: RMSE and NRMSE (%). Variable = Kinematic conformity/interface continuity (N1) 222

Table 6-24: Example 4 – Comparison of computational resources 225

Table 6-25: Simulation parameters 226

Table 6-26: RMSE and NRMSE (%). Variable = Global energies....230

Table 6-27: RMSE and NRMSE (%). Variable = Kinematic conformity (v/s UGUT results)....232

Table 6-28: RMSE and NRMSE (%). Variable = Interface continuity (Ω1 v/s Ω2)....232

Table 6-29: Example 5 – Comparison of computational resources 237

Table A-1: FEAPI::DOMAIN 250

Table A-2: FEAPI::COORDINATES 250

Table A-3: FEAPI::CONNECTIVITIES 251

Table A-4: FEAPI::MATERIAL 251

Table A-5: FEAPI::RESTRAINTS 252

Table A-6: FEAPI::LOADS 252

Table A-7: FEAPI::PRESCRIBED 253

Table A-8: FEAPI::TRANSIENT 254

Table A-9: FEAPI::INTERFACE 255

Table A-10: FEAPI::POST 256

List of Acronyms

1. AMR – Adaptive Mesh Refinement
2. API – Application Programming Interface
3. BDD – Balancing Domain Decomposition Method
4. BEC – Block Elimination using Crout Factorization
5. BEM – Block Elimination Method
6. BLAS – Basic Linear Algebra Subprograms
7. BVP – Boundary Value Problem
8. CDM – Central Difference Method
9. CST – Classical Stress Theory
10. CFL – Courant–Friedrichs–Lewy
11. CPU – Central Processing Unit
12. DDM – Domain Decomposition Method
13. DDT – Derived Data Type
14. DOF – Degrees of Freedom
15. FE – Finite Element
16. FEAPI – Finite Element Analysis Programming Interface
17. FEM – Finite Element Method
18. FETI – Finite Element Tearing and Interconnecting
19. GC – Gravouil-Combescure
20. GUI – Graphical User Interface
21. HHT –Hilber-Hughes-Taylor

22. LBB – Ladyzhenskaya-Babuska-Brezzi

23. LHS – Left Hand Side

24. LLM – Localized Lagrange Multipliers

25. MGC – Multiple Grid Coupling

26. MGMT – Multiple Grid Multiple Time-scale

27. MTC – Multiple Time-scale Coupling

28. MPI – Message Passing Interface

29. MM – Mixed Methods

30. M-FEM – Mortar Finite Element Method

31. NRMSE – Normalized Root Mean Square Error

32. PDE – Partial Differential Equation

33. PFEM – Programming the Finite Element Method

34. PH – Prakash-Hjelmstad

35. PCG – Preconditioned Conjugate Gradient Method

36. RHS – Right Hand Side

37. RMSD – Root Mean Square Deviation

38. RMSE – Root Mean Square Error

39. TCL – Tool Command Language

40. UGUT – Uniform Grid Uniform Time-scale

41. WBZ – Wood-Bossak-Zienkiewicz

Chapter 1: Introduction

1.1 Problem Statement and Motivation

Space and time are inherently coupled in the analysis of engineering problems. An approximate solution to these problems is usually obtained by using a mathematical tool suitable for the scale at which the physical phenomenon is addressed. Finite Element Method (FEM) is one such numerical technique that is used for solving problems in structural mechanics. It discretizes the governing equations over finite space and time and uses variational principles to minimize an error function in order to produce a stable solution. Numerical techniques as such, innately introduce discretization error due to the choice of finite space and time resolution (Shah 2002; Pointer 2002). Overall quality of the solution can be improved by spatial and temporal refinements, however, at the expense of increasing number of unknowns and consequently longer computation times.

Over the past couple of decades, researchers have devoted significant amount of effort towards element formulations, material modeling, non-linear formulations, efficient solution algorithms and several other numerical techniques in order to improve overall quality of the solution. Static mesh transition, Adaptive Mesh Refinement (AMR) (Bellenger & Coorevits 2005), Mortar Finite Element Method (Maday et al. 1988; Lamichhane & Wohlmuth 2004b) and Finite Element Tearing and Interconnecting (FETI) (Farhat & Roux 1991) are among the few techniques that are widely used to segregate problem size (total number of unknowns) between critical and remote regions. These techniques not only provide complete control over grid resolution, compared to fixed coarse or fine scale discretization, but also help in capturing local gradients and wave dynamics more accurately, hence yielding a better solution in the qualified region of interest. Domain Decomposition Method (DDM) (Smith et al. 1996; Toselli & Widlund 2005) is another approach that enables effective implementation of these techniques by decomposing the problem under consideration into component sub-domains, which can then be modeled independently, formulated and solved.

Within the context of time discretization, a common approach in coupling sub-domains is to use the same integration method (implicit or explicit) with the same time-step (Δt) globally over multiple grids. This is not recommended since it restricts one to analyze an entire domain using a single time-step that meets the stability and accuracy criteria for all elements. This is not desirable in the case of large scale problems since different regions could very well represent significantly different stability and accuracy requirements. In addition, different regions may exhibit high frequency (wave propagation type) and/or low frequency (vibration type) responses, requiring explicit and/or implicit time integration methods respectively. Accordingly, it is much more economical to use different time-steps or different time-stepping algorithms in different sub-domains in order to capture local behavior as accurately as possible.

In the analysis of large scale systems with complex geometries, the range of element sizes in a mesh varies several orders in magnitude. Certain parts of the mesh may contain very small elements, perhaps to capture high stress gradients, while large parts of the mesh may still be relatively coarse. Using an exclusive implicit or explicit time integration scheme with a uniform time-step, for such problems, is computationally very inefficient. If one were to use explicit scheme, the time-step would be restricted by the size of the smallest element in the mesh and it would take large number of steps to compute the response of the structure for the desired interval of time. On the other hand, using an implicit scheme with large time-steps, one would not be able to accurately capture the response in regions of the mesh with high gradients.

Aforementioned challenges motivate us to construct an enhanced computational method that can incorporate multiple scales, i.e. independently discretized (spatially and temporally) sub-domains, in order to preserve numerical accuracy and boost computational efficiency in the analysis of complex, large-scale structural systems, Figure 1-1.

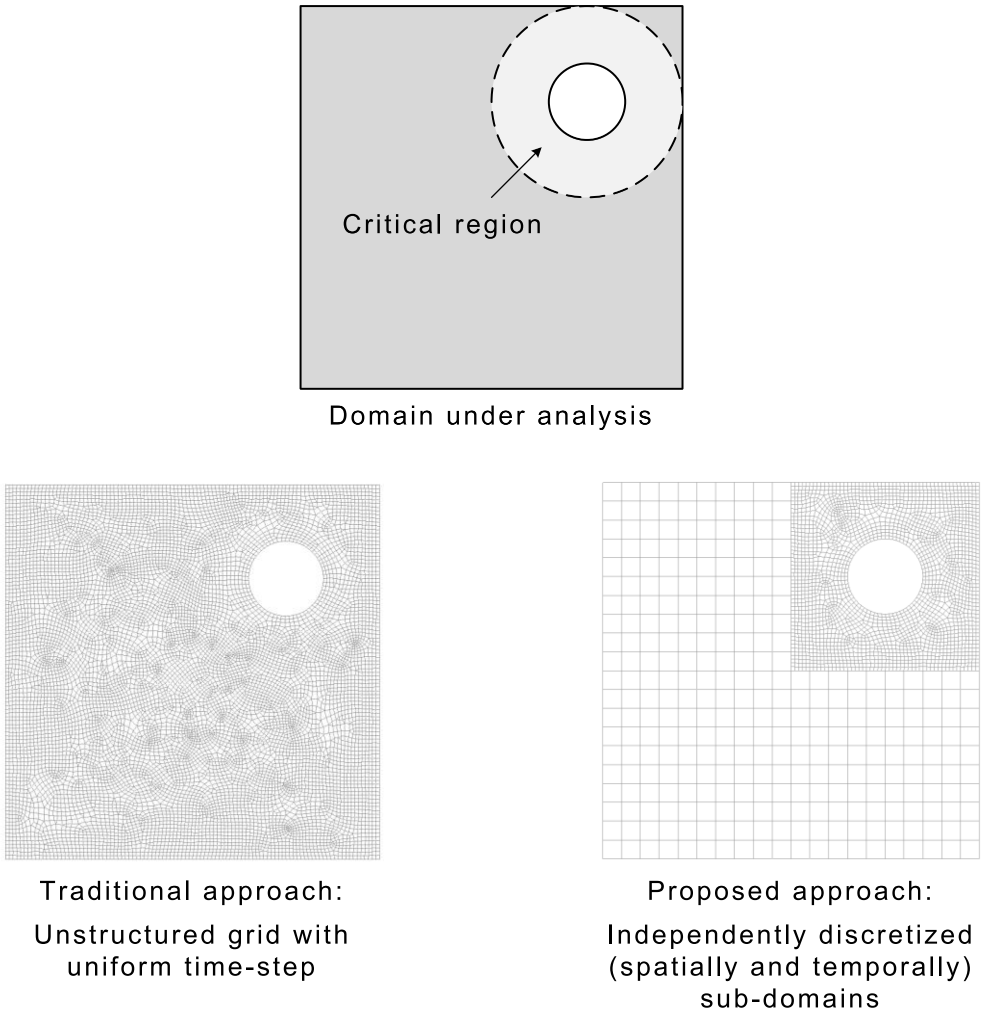


Figure 1-1: Proposed multiscale approach

1.2 Goals and Objectives

The objective for this dissertation can be broadly classified into two categories:

- The first objective is to explore, formulate, implement and verify a new computational algorithm that allows selective discretization of finite element problems in space and time. The goal within this objective is to establish a systematic approach for concurrent multiscale simulations that allow improved numerical accuracy in desired critical regions whilst enhancing computational efficiency globally. Proposed formulation shall be derived specifically for applications in solid continuum mechanics with a primary focus on problems in linear structural dynamics. Resulting Multiple Grid Multiple Time-scale (MGMT) approach and constituent assumptions shall be analyzed rigorously to provide a proof of stability using an established theoretical framework.
- The second objective of this dissertation is to implement the proposed formulation using FEM. The goal within this objective is to develop an efficient and an easy to implement solution algorithm and create a comprehensive, yet flexible computer program that can be used for the numerical simulation of linear structural dynamic systems using the proposed MGMT Method. Resulting computer program shall be used for numerical analysis and verification purposes by solving benchmark examples using traditional FEM and MGMT Method. Thorough performance evaluation, including stability analysis, assessment of numerical accuracy and computational efficiency shall also be performed using this computer program.

1.3 Contributions of the Dissertation

This dissertation contributes to the area of multiscale simulations in solid continuum mechanics. Specifically, it introduces an efficient and a systematic approach for Multiple Grid Multiple Time-scale (MGMT) simulations in linear structural dynamics.

The dissertation provides a précis of existing literature and theoretical foundation that supports the construction of a concurrent multiscale algorithm; it discusses their advantages and disadvantages, and builds upon their limitations and shortcomings to provide an advanced approach to MGMT simulations. Major contributions of this dissertation can be listed as follows:

1) Formulation of a consistent approach to implement user defined multiple grid and multiple time-scale discretizations in structural analysis
2) Development of an efficient and easy to implement solution algorithm
3) Implementation of proposed formulation and solution algorithm using FEM, creating a self-contained computer program for multiscale simulations
4) Verification of proposed formulation and its FE implementation for problems in linear structural dynamics
5) Evaluation of overall stability, numerical accuracy and computational efficiency, as assessed by solving baseline problems
6) Comprehensive comparison between traditional FEM and proposed multiscale approach by accessing various performance evaluation factors

1.4 Dissertation Organization

This dissertation elaborates on every aspect discussed in the previous sections and presents a comprehensive discussion of the proposed multiscale algorithm, its theory, formulation, implementation and verification. The content and structure of this dissertation is arranged as follows:

- **Chapter 2: Review of Literature and Theoretical Foundation**

Chapter 2 provides a comprehensive review of existing literature and theoretical foundation to help evaluate all theories and approaches relevant to multiscale modeling at continuum scales. Within the context of spatial/temporal discretization, traditional methods that allow improving accuracy and/or computational efficiency are briefly discussed. Focus is laid on mathematical techniques such as Domain Decomposition Methods (DDM), Lagrange Multipliers and Mortar Finite Element Method (M-FEM), since they innately assist multiscale coupling. Existing methods in coupling independently discretized sub-domains, GC Method (space and time) and PH Method (time only), are also discussed in details, along with their advantages shortcomings.

- **Chapter 3: MGMT Formulation**

Chapter 3 begins with the application of DDM, used to derived coupled equations of motion, for decomposed sub-domains augmented with an appropriate interface condition. M-FEM is then used to couple sub-domains that may be selectively discretized in space, resulting in conforming or non-conforming interfaces. Equations necessary for Multiple Grid (MG) coupling and their FE implementation is also discussed in details. Sub-domain specific time discretized equations are then obtained using Newmark integration method. Interface reactions (Lagrange Multipliers) from disparate time-scales are then condensed and expressed in terms of reactions at largest/global time-step, allowing Multiple Time-Scale (MT) coupling. Energy Method is then used to prove that resulting Multiple Grid and Multiple Time-Scale (MGMT) equations, and specifically the selected interface condition, results in unconditional stability for linear structural systems. Crout factorization and Block Elimination Method (BEM) are then used to design a systematic step-by-step algorithm for obtaining the solution of coupled MGMT equations that are synchronized at the global time-step.

- **Chapter 4: Programming the MGMT Method**

Following the formulation of MGMT Method, a FORTRAN 90 based FE code was developed for numerical simulations and verification purposes. Chapter 4 elaborates on the development and implementation of this computer program, highlighting its Object Oriented features, data structures and auxiliary FE libraries. Resulting computer program – Finite Element Analysis Programming Interface (FEAPI) is capable of solving linear structural dynamics problems that are modeled using independently discretized sub-domains. General structure of the program, information flow, implemented data types and its interaction with an external pre/post processor, GiD, is also described in Chapter 4.

- **Chapter 5: Numerical Analysis and Verification**

The focus in Chapter 5 is to evaluate the overall performance of MGMT Method by analyzing factors such as: 1) Numerical stability, 2) Numerical accuracy, 3) Computational efficiency. Two benchmark examples – 1) Transverse vibration 2) Longitudinal vibration of a 2D cantilever beam are first solved using traditional Uniform Grid Uniform Time-scale (UGUT) approach in order to establish baselien results. These examples are then solved using MGMT Method, taking into account various scenarios such as: 1) Implicit-Implicit, 4 sub-domain coupling, 2) Implicit-Explicit 2 sub-domain coupling, 3) Uniform/conforming grid with multiple time-stepping and 4) Multiple grids with uniform time-stepping. Results for global energy balance, interface energy dissipation/accumulation, evolution/distribution of kinematic quantities and propagation of structural waves across component sub-domains are presented with corresponding relative errors in comparison with converged UGUT cases. Computational efficiency is evaluated by comparing total number of nodes, elements, required number of equations in primary unknown variables and resulting computation times (CPU solution time). A comprehensive performance evaluation matrix is designed and the relative advantages/shortcomings of MGMT Method are highlighted in detail.

- **Chapter 6: MGMT Example Problems and Results**

Chapter 6 presents traditional FE application problems wherein MGMT Method can be potentially used to preserve numerical accuracy in desired regions, whilst improving computational efficiency globally. Example problems, such as stress resolution in critical regions, wave propagation across heterogeneous material systems, response under complex loading functions and large-scale structural problems are solved using UGUT and MGMT Method and their results are presented in this Chapter.

- **Chapter 7: Conclusions and Future Directions**

Chapter 7 concludes and summarizes the content of this dissertation and provides directives for related future work.

Chapter 2: Review of Literature and Theoretical Foundation

2.1 Governing Equations in Linear Structural Dynamics

Continuum mechanics characterizes the fundamental physical model that provides foundation for all physical theories concerning the modeling of material behavior at macroscopic scales. In solid mechanics, and particularly structural dynamics, the response of a structure is represented by a Boundary Value Problem (BVP); which is a set of Partial Differential Equations (PDE) describing the kinematics of deformation, conservative laws of continua and the constitutive laws along with appropriate boundary conditions and initial conditions.

This section presents a brief review of the mathematics and physical laws that approximate the macroscopic behavior of material that is subjected to mechanical loading. A comprehensive description on continuum/solid mechanics and structural dynamics can be found in: (Eringen 1980), (Chen et al. 2000), (Spencer 2004), (Zienkiewicz et al. 2005), (Reddy 2007), (Bower 2009) and (Hughes 2012).

2.1.1 Kinematics

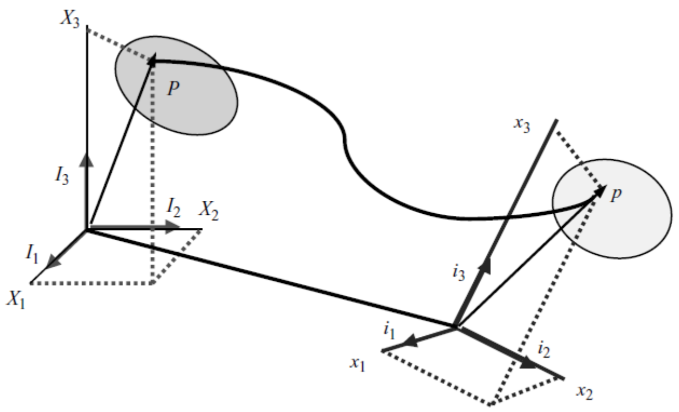


Figure 2-1: Motion and deformation of a point in continuum (Chen et al. 2000)

In continuum mechanics, and especially solid mechanics, motion is described by choosing some convenient configuration (reference/original configuration) of the solid that is in the initial, undeformed state. The material then changes its shape under the action of external loads, and at some time t occupies a new region, which is called the deformed or current configuration of the solid, Figure 2-1. If the position of point P in the reference configuration is expressed by $\boldsymbol{X} \equiv X_K \ (K = 1,2,3)$ in the Lagrange (material) coordinate system and its position in the deformed configurations, represented by p, is expressed by $\boldsymbol{x} \equiv x_k \ (k = 1,2,3)$ in the Eulerian (spatial) coordinate system, then the motion of the solid is expressed through a deformation mapping function as follows:

$$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{X}, t) \quad or \quad x_k = x_k(X_K, t) \tag{2.1}$$

$$\boldsymbol{X} = \boldsymbol{X}(\boldsymbol{x}, t) \quad or \quad X_K = X_K(x_k, t) \tag{2.2}$$

To be a physically admissible deformation (Bower 2009) the mapping function must be 1:1 on the full set of real numbers and must be invertible; it must also be continuous and continuously differentiable and must satisfy $\left|\partial x_k / \partial X_K\right| > 0$ or $\left|x_{k,K}\right| > 0$. <u>Note:</u> From now on indices after comma will indicate partial differentiation with respect to associated coordinate system, Lagrangian (majuscule) or Eulerian (minuscule).

The displacement $(\boldsymbol{u})$ of a material point expressed as a vector that extends from $\boldsymbol{X}$ in the reference state to $\boldsymbol{x}$ in the deformed state is then defined as:

$$\boldsymbol{u} = \boldsymbol{x} - \boldsymbol{X} + \boldsymbol{b} \tag{2.3}$$

Where $\boldsymbol{b}$ represents the vector extending from the origin of the Lagrangian (reference) coordinate system to the Eulerian (deformed) coordinate system. The displacement field in Eq. (2.3) completely specifies the change in shape of the solid. Velocity vector $\boldsymbol{v}$ is then expressed as the material time rate change of the position vector $p_k = x_k(\boldsymbol{X}, t)$ and is defined as:

$$\boldsymbol{v} \equiv \frac{d\boldsymbol{p}}{dt} \quad or \quad v_k = \frac{\partial x_k}{\partial t} \tag{2.4}$$

And consequently acceleration vector $\boldsymbol{a}$ is defined as:

$$\boldsymbol{a} \equiv \frac{d\boldsymbol{v}}{dt} \quad or \quad a_k(\boldsymbol{x}, t) = \frac{\partial v_k}{\partial t} + v_{k,l} v_l \tag{2.5}$$

From Eq. (2.1) and Eq. (2.2) we have $dx_k = x_{k,K} dX_K$ and $dX_K = X_{K,k} dx_k$, yielding the definition of deformation gradient $\boldsymbol{F} = \nabla \boldsymbol{x} = F_{kK} \equiv x_{k,K} \equiv \partial x_k / \partial X_K$ and its inverse $\boldsymbol{F}^{-1} \equiv X_{K,k} \equiv \partial X_K / \partial x_k$. The Jacobian is defined as $J = |\boldsymbol{F}|$ and is a measure of the volume change produced by a deformation. <u>Note:</u>

1) For any physically admissible deformation, the volume of the deformed element must be positive. Therefore, all physically admissible displacement fields must satisfy $J > 0$
2) If a material is incompressible, its volume remains constant and accordingly, it must satisfy $J = 1$
3) If the mass density of a material at a point in the undeformed state is ρ_0, its mass density in the deformed state is expressed as: $\rho = \rho_0 / J$

Green deformation tensor is defined as $\boldsymbol{C} = \boldsymbol{F}^T \boldsymbol{F}$ or $C_{KL} \equiv x_{k,K} x_{k,L}$ with corresponding Lagrangian strain tensor expressed as $\boldsymbol{E} = 1/2\left(\boldsymbol{C} - \boldsymbol{I}\right)$ or $E_{KL} \equiv 1/2\left(C_{KL} - \delta_{KL}\right)$, where δ_{KL} represents the Kronecker delta. Equivalent definitions for Cauchy deformation tensor and Eulerian strain tensors are expressed as $c_{kl} \equiv X_{K,k} X_{K,l}$ and $\varepsilon_{kl} \equiv 1/2\left(\delta_{kl} - c_{kl}\right)$ respectively. Lagrangian and Eulerian strain components expressed in terms of displacement gradients can be expressed as:

$$E_{KL} = \frac{1}{2}\left(\frac{\partial U_K}{\partial X_L} + \frac{\partial U_L}{\partial X_K} + \delta_{MN} \frac{\partial U_M}{\partial X_K} \frac{\partial U_N}{\partial X_L} \right) \tag{2.6}$$

$$\varepsilon_{kl} = \frac{1}{2}\left(\frac{\partial u_k}{\partial x_l} + \frac{\partial u_l}{\partial x_k} - \delta_{mn} \frac{\partial u_m}{\partial x_k} \frac{\partial u_n}{\partial x_l} \right) \tag{2.7}$$

Using above expression, infinitesimal strain tensor can be defined as:

$$E_{KL} = \frac{1}{2}\left(\frac{\partial U_K}{\partial X_L} + \frac{\partial U_L}{\partial X_K} \right) \tag{2.8}$$

$$\varepsilon_{kl} = \frac{1}{2}\left(\frac{\partial u_k}{\partial x_l} + \frac{\partial u_l}{\partial x_k} \right) \tag{2.9}$$

The infinitesimal strain tensor is an approximate deformation measure, which is only valid for small shape changes; however it is more convenient than the Lagrange or Eulerian strain definitions because it is linear. Comprehensive discussion and derivation of aforementioned discussion can be found in – ‘Mechanics of Continua’ (Eringen 1980) and ‘Meshless Methods in Solid Mechanics’ (Chen et al. 2000).

2.1.2 Balance Laws and Equilibrium

Fundamental balance laws include conservation of mass, momentum and energy. Equations of motion and equilibrium of deformable solids however are obtained by generalizing Newton's Laws of motion (conservation of linear and angular momentum) to deformable solids.

A. Balance of Linear Momentum

The time rate change of linear momentum $\mathcal{P}$ is equal to the resultant force $\mathcal{F}$ acting on the body.

The linear momentum of a volume (v) can be expressed as:

$$\mathcal{P} = \int_{\mathrm{v}} \rho \boldsymbol{v} d\mathrm{v} \tag{2.10}$$

Where ρ represents mass density and $\boldsymbol{v}$ is the velocity vector. Then the balance of linear momentum is established by:

$$\frac{d\mathcal{P}}{dt} = \frac{d}{dt}\int_{\mathrm{v}} \rho \boldsymbol{v} d\mathrm{v} = \mathcal{F} \tag{2.11}$$

The resultant force acting on any arbitrary internal volume (v) with a boundary surface (a) within a deformed solid is expressed as:

$$\mathcal{F} = \int_{\mathrm{a}} \boldsymbol{T}(\boldsymbol{n}) d\mathrm{a} + \int_{\mathrm{v}} \rho \boldsymbol{b} d\mathrm{v} \tag{2.12}$$

First term in Eq. (2.12) refers to the resultant force acting on the internal surface (a) where $\boldsymbol{T}(\boldsymbol{n})$ refers to the traction acting on any surface with a unit outward normal $\boldsymbol{n}$. The

Cauchy (true) stress tensor, representing force per unit area of the deformed solid is then denoted by $\boldsymbol{\sigma} = \boldsymbol{T}(\boldsymbol{n})$, see Figure 2-2 (a) and Eq. (2.13). The second term in Eq. (2.12) is the resultant body force; where $\boldsymbol{b}$ represents the body force vector denoting the external force acting on the interior of a solid (v), per unit mass. See Figure 2-2 (b) and Eq. (2.14).

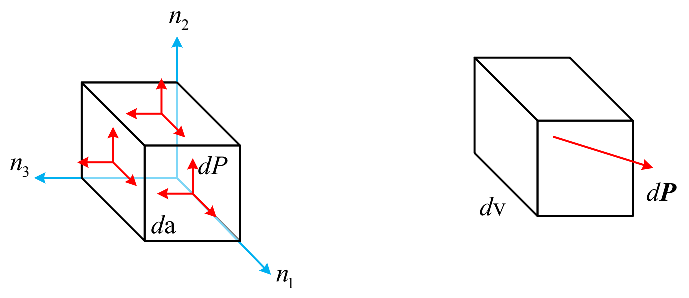


Figure 2-2: (a) Surface traction and (b) Internal body force

Using notations used in Figure 2-2, Cauchy stress components σ_{ij} and body force vector $\boldsymbol{b}$ are expressed as:

$$\sigma_{ij} = T_j(n_i) = \lim_{d\mathrm{a}\to 0} \frac{dP_j}{d\mathrm{a}_i} \tag{2.13}$$

$$\boldsymbol{b} = \frac{1}{\rho} \lim_{d\mathrm{v}\to 0} \frac{d\boldsymbol{P}}{d\mathrm{v}} \tag{2.14}$$

The balance of linear momentum, or Eq. (2.11), can then be expressed as:

$$\frac{d}{dt}\int_{\mathrm{v}} \rho \boldsymbol{v} d\mathrm{v} = \int_{\mathrm{a}} \boldsymbol{T}(\boldsymbol{n}) d\mathrm{a} + \int_{\mathrm{v}} \rho \boldsymbol{b} d\mathrm{v} \tag{2.15}$$

$$\sigma_{ij,i} + \rho b_j = \rho a_j \tag{2.16}$$

B. Balance of Angular Momentum

The time rate change of angular momentum $\mathcal{H}$ about any fixed point is equal to the resultant momentum $\mathcal{M}$ about that point.

The angular momentum of a volume (v) can be expressed as:

$$\mathcal{H} = \int_{\mathrm{v}} \boldsymbol{x} \times \rho \boldsymbol{v} d\mathrm{v} \tag{2.17}$$

Then the balance of angular momentum is established by:

$$\frac{d\mathcal{H}}{dt} = \frac{d}{dt}\int_{\mathrm{v}} \boldsymbol{x} \times \rho \boldsymbol{v} d\mathrm{v} = \mathcal{M} \tag{2.18}$$

The resultant moment (about a fixed point) exerted by tractions and body forces acting on a general region within a solid is expressed as:

$$\mathcal{M} = \int_{\mathrm{a}} \boldsymbol{x} \times \boldsymbol{T}(\boldsymbol{n}) d\mathrm{a} + \int_{\mathrm{v}} \boldsymbol{x} \times \rho \boldsymbol{b} d\mathrm{v} \tag{2.19}$$

Balance of angular momentum is then expressed as:

$$\frac{d}{dt}\int_{\mathrm{v}} \boldsymbol{x} \times \rho \boldsymbol{v} d\mathrm{v} = \int_{\mathrm{a}} \boldsymbol{x} \times \boldsymbol{T}(\boldsymbol{n}) d\mathrm{a} + \int_{\mathrm{v}} \boldsymbol{x} \times \rho \boldsymbol{b} d\mathrm{v} \tag{2.20}$$

Conservation of angular momentum for a continuum requires that the Cauchy stress must be symmetric, i.e.:

$$\sigma_{ji} = \sigma_{ij} \tag{2.21}$$

2.1.3 Constitutive Model

The governing equations in structural dynamics are completed by constitutive laws that provide the missing connection. Unlike kinematics and balance laws, a constitutive law cannot be calculated or predicted from first principles, except for a very few special cases, such as small deformations of crystalline materials, where elastic properties can be estimated using ab-initio techniques that approximate quantum mechanical level atomistic interactions in some way (Bower 2009). In solid (structural) mechanics we are primarily concerned with the stress-strain relationship. As described in (Chen et al. 2000), for a constitutive model to adequately represent a material, various axioms, such as axiom of Causality, Determinism, Equipresence, Neighborhood, Memory, Objectivity, Material Invariance and Admissibility, must be satisfied.

In this section we will briefly present the constitutive model for an isotropic, linear elastic material behavior. Assuming infinitesimal strain tensor defined by Eq. (2.9) and Cauchy stress tensor, the stress-strain relationship for an isotropic linear elastic solid, in terms of Young's modulus (E) and Poisson's ratio (ν) is expressed as:

$$\varepsilon_{ij} = \frac{1+\nu}{E}\sigma_{ij} - \frac{\nu}{E}\sigma_{kk}\delta_{ij} \tag{2.22}$$

$$\begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ 2\varepsilon_{23} \\ 2\varepsilon_{13} \\ 2\varepsilon_{12} \end{bmatrix} = \frac{1}{E}\begin{bmatrix} 1 & -\nu & -\nu & 0 & 0 & 0 \\ -\nu & 1 & -\nu & 0 & 0 & 0 \\ -\nu & -\nu & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2(1+\nu) & 0 & 0 \\ 0 & 0 & 0 & 0 & 2(1+\nu) & 0 \\ 0 & 0 & 0 & 0 & 0 & 2(1+\nu) \end{bmatrix}\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{13} \\ \sigma_{12} \end{bmatrix} \tag{2.23}$$

The inverse relationship can be expressed by:

$$\sigma_{ij} = \frac{E}{1+\nu}\left\{\varepsilon_{ij} + \frac{\nu}{1-2\nu}\varepsilon_{kk}\delta_{ij}\right\} \tag{2.24}$$

$$\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{23} \\ \sigma_{13} \\ \sigma_{12} \end{bmatrix} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} \end{bmatrix} \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ 2\varepsilon_{23} \\ 2\varepsilon_{13} \\ 2\varepsilon_{12} \end{bmatrix} \tag{2.25}$$

The stress-strain relationship in Eq. (2.25) is often expressed using the elastic modulus tensor (C_{ijkl}) as:

$$\sigma_{ij} = C_{ijkl}\varepsilon_{kl} \tag{2.26}$$

For Plane Strain or Plane Stress deformations, some strain or stress components are always zero (by definition) so the stress-strain laws can be simplified. Accordingly, for plane strain and plane stress deformations we have:

$$\textit{Plane Strain} \rightarrow \varepsilon_{33} = \varepsilon_{23} = \varepsilon_{13} = 0 \tag{2.27}$$

$$\textit{Plane Stress} \rightarrow \sigma_{33} = \sigma_{23} = \sigma_{13} = 0 \tag{2.28}$$

2.2 Finite Element Implementation and Spatial Discretization

PDEs, referred to as the strong form, discussed in preceding sections represent the governing equations for dynamic linear elasticity. An analytical solution for governing equations in strong form is almost never available, even for simple problems and accordingly, these equations are expressed in an integral form, referred to as the weak form, to obtain an approximate solution to the problem. Two distinct procedures are available for obtaining such approximation: 1) method of weighted residual (Galerkin Method) and 2) method of variational functionals. A comprehensive discussion on these methods can be found in (Zienkiewicz et al. 2005).

In this section we will discuss how the approximate solution to aforementioned PDEs can be obtained by using the Galerkin Method and FEM. As discussed in preceding sections, our goal is to calculate displacements, strains and stresses satisfying the following governing equations for dynamic linear elasticity:

1) The strain-displacement equation: $\varepsilon_{ij} = \frac{1}{2}\left(u_{i,j} + u_{j,i}\right)$ or $\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$

2) The elastic stress-strain law: $\sigma_{ij} = C_{ijkl}\varepsilon_{kl}$

3) The equation of motion (as derived under the balance of linear momentum):

$\sigma_{ij,i} + \rho b_j = \rho a_j$ or $\frac{\partial \sigma_{ij}}{\partial x_i} + \rho b_j = \rho \frac{\partial^2 u_j}{\partial t^2}$

4) And the necessary boundary conditions on displacement and stress: $u_i = u_i^* \ \ on \ \ \Gamma_u$

and $\sigma_{ij} n_i = T_j^*(\boldsymbol{n}) \ \ on \ \ \Gamma_T$

Now, in a general three-dimensional continuum denoted by Ω with a volume V, the weak form of equation of motion using Galerkin Method is expressed as:

$$\int_{\Omega} \Psi(\sigma_{ij,i} + \rho b_j - \rho a_j) dV = 0 \tag{2.29}$$

Here, Ψ represents a test weight function that reduces the residual error in an average sense. To solve the integral form of elasticity equation given in Eq. (2.29), we discretize the displacement field. That is, we calculate the displacement field at a set of n discrete points, called nodes, within the continuum that is discretized using finite number of elements (E).

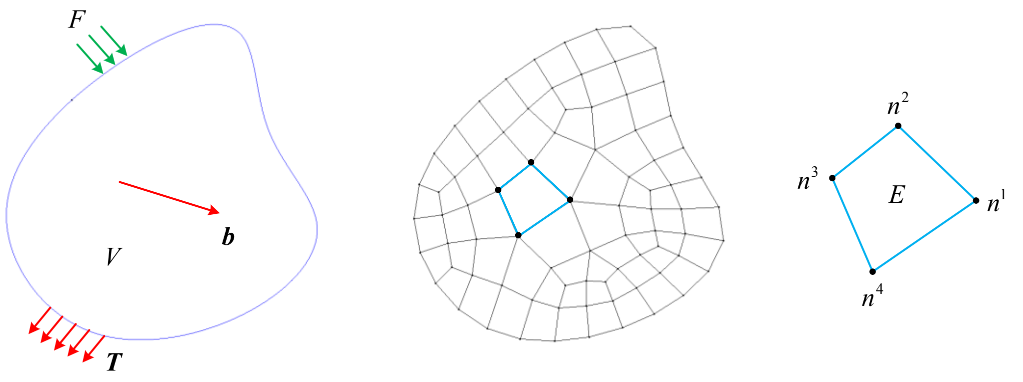


Figure 2-3: Discretization in space using Finite Elements

The displacement field at an arbitrary point within the solid is then specified by interpolating between nodal values. The continuous variable u_i is then approximated by $\hat{u}_i$ through simple functions of space variables, called shape functions as follows:

$$u_i \approx \hat{u}_i = N_\alpha U_{i\alpha} \tag{2.30}$$

Where, N represents the shape function and U represents the nodal displacements. Subscript i denotes the spatial dimensions $(i = 1,2,3)$ and $\alpha = 1,2,3,\ldots m$ represents an index denoting the node number on a particular element and m is the total number of nodes per element. Equation (2.30) is also expressed as:

$$u_i \approx \hat{u}_i = N_{i\beta} U_\beta \tag{2.31}$$

Here, $N_{i\beta}$ is the $(i,\beta)^{th}$ component of the $3 \times 3m$ matrix made from m shape functions (N_α) and U_β is the β^{th} component of the vector of nodal displacements of the finite element. Accordingly, $\beta = 1,2,3,\ldots 3m$.

Since the primary unknown variable in Eq. (2.29) is the displacement u_i, an appropriate choice of test functions is the coefficients from Eq. (2.31). Accordingly, Eq. (2.29) can be expressed in terms of shape functions as an approximate solution to the continuous variable u_i as follows:

$$\int_\Omega N_{j\beta}(\sigma_{ij,i} + \rho b_j - \rho a_j) dV = 0 \tag{2.32}$$

Using Green-Gauss theorem and integration by parts, above equation becomes:

$$\int_\Omega (\sigma_{ij} N_{j\beta,i} - \rho b_j N_{j\beta} + \rho a_j N_{j\beta})\, dV - \int_\Gamma \sigma_{ij} n_i N_{j\beta}\, d\Gamma = 0 \tag{2.33}$$

The infinitesimal strain tensor, in terms of selected weight functions and nodal displacements, can be expressed as:

$$\varepsilon_{ij} = \frac{1}{2}(N_{j\beta,i} + N_{i\beta,j})U_\beta \equiv B_{ij\beta} U_\beta \tag{2.34}$$

Introducing elastic stress-strain law and updated strain-displacement relationship from the above equation into Eq. (2.33) we get:

$$\int_{\Omega} (c_{ijkl} B_{kl\alpha} U_{\alpha} N_{j\beta,i} - \rho b_j N_{j\beta} + \rho N_{j\alpha} \ddot{U}_{\alpha} N_{j\beta})\, dV - \int_{\Gamma} \sigma_{ij} n_i N_{j\beta}\, d\Gamma = 0 \tag{2.35}$$

$$\begin{aligned} \therefore U_{\alpha} \int_{\Omega} c_{ijkl} B_{kl\alpha} B_{ji\beta}\, dV + \ddot{U}_{\alpha} \int_{\Omega} \rho N_{j\alpha} N_{j\beta}\, dV - \int_{\Omega} \rho b_j N_{j\beta}\, dV \\ - \int_{\Gamma_T} T_j^* N_{j\beta}\, d\Gamma - \int_{\Gamma_u} \sigma_{ij} n_i N_{j\beta}\, d\Gamma = 0 \end{aligned} \tag{2.36}$$

The fourth term in Eq. (2.36) is the natural boundary condition on stress and the fifth term is the essential (Dirichlet) boundary condition on displacement which vanishes when the test functions pass through nodes, as in FEM. Resulting space discretized (over the nodes of finite number of elements) equation of motion is expressed as:

$$M_{\beta\alpha} \ddot{U}_{\alpha} + K_{\beta\alpha} U_{\alpha} = F_{\beta} \quad or \quad M\ddot{U} + KU = F \tag{2.37}$$

Where:

$$\text{Mass Matrix:} \quad M_{\beta\alpha} = \int_{\Omega} \rho N_{j\alpha} N_{j\beta}\, dV \tag{2.38}$$

$$\text{Stiffness Matrix:} \quad K_{\beta\alpha} = \int_{\Omega} c_{ijkl} B_{kl\alpha} B_{ji\beta}\, dV \tag{2.39}$$

$$\text{Load Vector:} \quad F_{\beta} = \int_{\Omega} \rho b_j N_{j\beta}\, dV + \int_{\Gamma_T} T_j^* N_{j\beta}\, d\Gamma \tag{2.40}$$

First term in Eq. (2.37) represents inertia forces, whereas the second term represents elastic forces (in linear elastic constitutive modeling); however, solids in motion also experience a third type of force whose action is to dissipate energy. This force, in general, is dependent on velocity and is usually denoted by $C_{\beta\alpha}$ (also called: Damping Matrix). Ideally, this matrix is obtained by incorporating a rate-dependent constitutive model, but is often assumed to be a linear combination of mass and stiffness matrices as follows:

$$C_{\beta\alpha} = m_c M_{\beta\alpha} + k_c K_{\beta\alpha} \tag{2.41}$$

Where mass coefficient (m_c) and stiffness coefficient (k_c) are scalars and are referred to as Rayleigh damping coefficients. These coefficients are related to the damping ratio (ζ) as follows:

$$\zeta = \frac{m_c + k_c \omega^2}{2\omega} \tag{2.42}$$

Where ω is the natural (usually fundamental) frequency of vibration. The equation of motion, after incorporating damping forces, is then expressed as:

$$M\ddot{U} + C\dot{U} + KU = F \tag{2.43}$$

This equation is the final space discretized (semi-discretized) equation of motion for a linear structural dynamic system and represents the equilibrium of a deforming solid under the action of external forces. Note: 1) Initial conditions must include nodal velocities and 2) Inertia, damping and elastic stiffness matrices are constant for linear elastic problems, but load vector (F) will in general be a function of time.

Within the context of spatial discretization, grid segregation is commonly implemented using 1) Static Mesh Refinement, also called Transition Mesh, Figure 2-4, or 2) Adaptive (dynamic) Mesh Refinement (AMR), see Figure 2-5. These methods allow selective concentration of nodes in vicinity of the critical region as opposed to remote regions, hence improving accuracy whilst preserving computational efficiency.

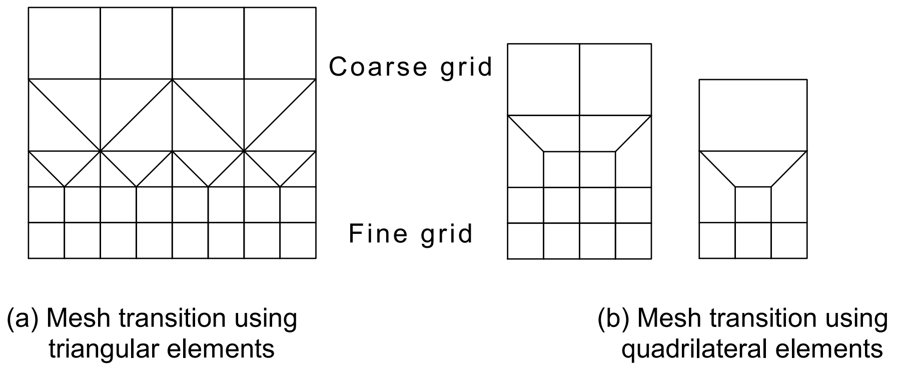

Coarse grid
Fine grid
(a) Mesh transition using triangular elements
(b) Mesh transition using quadrilateral elements

Figure 2-4: Static mesh refinement using transition elements

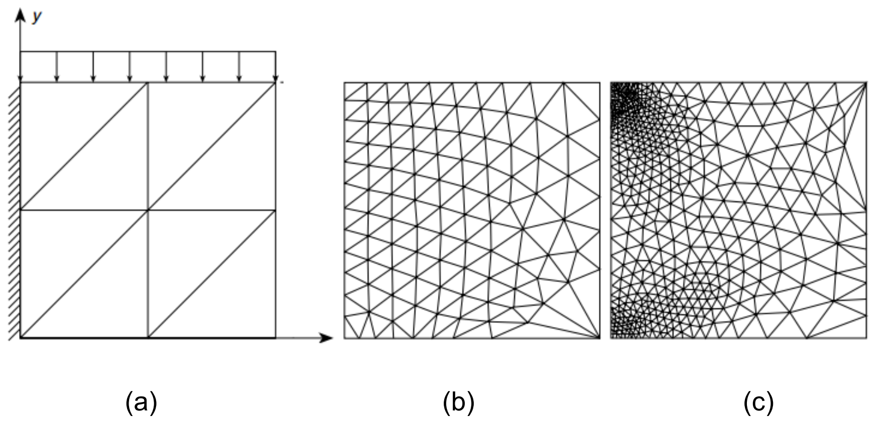


Figure 2-5: Adaptive mesh refinement (h-refinement) (a) Initial discretization: Mesh 1 (b) Mesh 2 and (c) Mesh 3 (Zienkiewicz et al. 2005)

Static mesh refinement requires the user to predict a critical zone and pre-define its grid density during the meshing/pre-processing phase. It is necessary to ensure node-to-node connectivity within the global computational grid and accordingly, transition zones utilizing triangular or quadrilateral elements must be used, as shown in Figure 2-4. AMR on the other hand, uses pre-defined error norm/criterion during the simulation phase to resolve steep gradients and coarsen flat gradients. Following two approaches are widely used for refinement of FE using AMR, (Zienkiewicz et al. 2005):

1) H-refinement uses the same class of elements, but with different sizes, making them smaller or larger, in order to attain maximum efficiency in reaching a desired solution.
2) P-refinement uses the same element size, but changes the order of the polynomial used to define their shape functions.

These mesh refinement techniques, although they allow improving grid resolution in a desired critical regions, they do not allow selective discretization in time-domain. This not only affects the solution accuracy in transient problems, but also affects the global computational efficiency. AMR also requires re-computation of global system matrices, for every refinement, further increasing simulation time and reducing computational efficiency.

2.3 Temporal Discretization and Direct Integration Methods

In structural dynamics, the FEM solves equilibrium equations discretized both in space and time. In a time-history or dynamic response problem, we usually solve the dynamic equation in the form of Eq. (2.43) for U, $\dot{U}$ and $\ddot{U}$ as functions of time. There are two distinct methods for time-history analysis: 1) Mode Superposition Method and 2) Direct Integration Method. For most problems in structural dynamics or wave propagation, direct integration method is more convenient. Using this method, the time interval of interest is divided into N time-steps of size Δt, and the equilibrium equation is enforced at discrete instants of time t_n where $n \in \{0,1,2,...N-1\}$. Direct integration algorithms as such, can be broadly classified into two categories: Implicit and Explicit methods. Since a description of these algorithms, within the context of their computational characteristics (accuracy, stability requirements, efficiency, etc.), can serve as an important basis for their implementation, we will now briefly discuss these time integration algorithms, their benefits/limitations and suitable FE implementation.

2.3.1 Explicit Methods

When the solution at time $(t+\Delta t)$ is computed based on quantities from the previous time-step only, the method is called Explicit Method. In general, it may be expressed as:

$$U_{n+1} = f\left(U_n, \dot{U}_n, \ddot{U}_n, U_{n-1}, \dot{U}_{n-1}, \ddot{U}_{n-1}, \ldots\right) \tag{2.44}$$

Direct integration algorithms derived from such expressions are referred to as single-step methods if the right hand side (RHS) contains variables at time t_n only, or two-step methods when the RHS contains variables from time t_n and t_{n-1}. Central Difference Method (CDM), for example, is a two-step explicit method that assumes the following:

$$\ddot{U}_n = \frac{\left(U_{n-1} - 2U_n + U_{n+1}\right)}{\Delta t^2} \tag{2.45}$$

$$\dot{U}_n = \frac{\left(U_{n+1} - U_{n-1}\right)}{2\Delta t} \tag{2.46}$$

The displacement solution at time $(t + \Delta t)$ or t_{n+1} is then obtained by considering Eq. (2.43) at time t_n as follows:

$$M\ddot{U}_n + C\dot{U}_n + KU_n = F_n \tag{2.47}$$

Substituting $\ddot{U}_n$ and $\dot{U}_n$ from Eq. (2.45) and Eq. (2.46) we obtain:

$$\left(\frac{M}{\Delta t^2} + \frac{C}{2\Delta t}\right)U_{n+1} = F_n - \left(K - \frac{2M}{\Delta t^2}\right)U_n - \left(\frac{M}{\Delta t^2} + \frac{C}{2\Delta t}\right)U_{n-1} \tag{2.48}$$

Equation (2.48) can now be used to solve for unknown displacements (U_{n+1}).

At the end of every explicit time-step, the stiffness matrix is computed based on changes in geometry and/or material properties. Stiffness matrix inversion/factorization is not required in this case (Bathe 1996). Although computationally desired, the accuracy of explicit methods largely depends on the time-step size and hence the total number of time increments. The solution tends to diverge for a large time-step or if the number of increments are not sufficient. Explicit methods also do not enforce equilibrium of internal forces with external loads.

The length of time-step in explicit integration schemes, such as CDM, is limited and subject to following conditions (Cook et al. 2001):

$$\Delta t \le \Delta t_{critical} = \frac{2}{\omega_{\max}} \qquad \text{(undamped system)} \tag{2.49}$$

$$\Delta t \le \Delta t_{critical} = \frac{2}{\omega_{\max}}\left(\sqrt{1-\zeta^2}-\zeta\right) \qquad \text{(damped system)} \tag{2.50}$$

In the above equations, $\omega_{\max}$ refers to the maximum natural frequency in the finite element mesh and ζ is the critical damping factor. For wave propagation problems, time-step Δt is also subject to the Courant–Friedrichs–Lewy (CFL) condition as follows:

$$\Delta t \le \Delta t_{critical} = \frac{H}{c} \tag{2.51}$$

$$c = \sqrt{\frac{\lambda + 2G}{\rho}} \qquad \lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \qquad G = \frac{E}{2(1+\nu)} \tag{2.52}$$

Where, H is the grid spacing and c is the speed of dilatational waves (P – waves) in a continuum, λ and G are the Lame constants and ρ is the mass density.

Alternately, stability conditions from Eq. (2.51) and (2.52) may also be expressed as, (Plesek et al. 2012):

$$Cr \le \frac{2}{\bar{\omega}} \qquad \left(\bar{\omega} = \frac{\omega_{\max} H}{c}\right) \tag{2.53}$$

Where the dimensionless Courant number is defined as:

$$Cr = \frac{c\Delta t}{H} \tag{2.54}$$

Here $Cr \leq 1$ exactly satisfies the stability requirement for linear elements integrated explicitly using CDM. Since the 'critical' explicit time-step is a function of grid spacing within a mesh, it is clear that stability and accuracy requirements of the smallest element within a mesh will dictate the global time-step size.

2.3.2 Implicit Methods

Implicit methods compute the solution at time $(t + \Delta t)$ based on itself and previous states of the problem. Contrary to Eq.(2.44), implicit methods may be described as:

$$U_{n+1} = f\left(\dot{U}_{n+1}, \ddot{U}_{n+1}, U_n, \dot{U}_n, \ddot{U}_n, U_{n-1}, \dot{U}_{n-1}, \ddot{U}_{n-1}, \ldots\right) \tag{2.55}$$

This method is similar to explicit methods in addition that it enforces equilibrium condition, i.e. Eq. (2.55) is combined with the equation of motion at time $(t + \Delta t)$. Hence it includes a convergence check (typically enforced by a user specified tolerance) which is usually performed using Newton–Raphson iterations. This method is more accurate since it takes into consideration the previous state of the structure, which accounts for geometric and/or material changes. It also allows for larger time-steps, however, at the cost of increased Newton–Raphson iterations. Since these iterations include stiffness matrix calculation and factorization, implicit methods can be computationally very expensive. These methods, however, are more desirable for non-linear analysis as they perform the much required convergence check.

Explicit methods in general utilize less computational resources per time-step, but require a large number of steps, whereas implicit methods utilize large amount of computational resources, but require fewer steps. The choice between explicit and implicit methods is hence dictated by the problem under consideration. Explicit methods are great when it comes to short duration analysis, such as crash, impact or blast analysis problems. Since these problems (categorized under the wave propagation type) result in several high frequency modes, small time-steps are required to accurately model the response. Implicit methods, however allow large time-steps enabling faster solutions, which is desirable in case of structural dynamic problems where the response is typically dominated by low frequency modes.

According to (Bajer 2002), the best time integration scheme should have the following features:

1) It should be unconditionally stable.
2) It should have parameter controlled dissipation or no dissipation.
3) Controlled dissipation, if present, should not affect the response of lower modes. (Numerical dissipation is sometimes desirable in the high frequency regime)
4) It should have better computational efficiency.
5) It should permit computation of non-inertia structures with the motion being kinematically enforced.

Following features are also important from a practical point of view:

1) Computational cost
2) Accuracy

3) Stability

4) Damping of high frequency response

5) Efficient information communication (important in wave propagation problems)

Among the most widely used time integration methods in structural dynamics, Newmark Method (Newmark 1959), HHT-α Method (Hilber et al. 1977), WBZ-α Method (Wood et al. 1980) and Generalized-α Method (Chung & Hulbert 1993) possess most of the above mentioned features. In addition, these algorithms may be used for implicit or explicit time integration by selecting appropriate algorithmic parameters.

2.3.3 Generalized-α Method

When incorporating fine-scale discretization in critical regions as opposed to remote regions, one must be careful about spurious high frequency modes introduced as a result of spatial FE discretization of critical regions. These high frequency modes are not characteristic of the physical structure and may affect the overall quality of solution. Hence it is desirable to be able to introduce controlled numerical dissipation to damp out any high frequency contributions. Numerical damping may be introduced within Newmark family, but it leads to undesirable low frequency damping and reduced order of accuracy. Aforementioned α-algorithms have been developed (Hilber et al. 1977; Wood et al. 1980; Chung & Hulbert 1993) to improve upon this situation. Here the equation of motion is satisfied in a weighted average sense, allowing improved algorithmic damping that is mainly concentrated in the high frequency domain. Generalized-α Method is one of these one step three stage numerically dissipative algorithms that incorporates an optimal combination of high frequency and low frequency damping. That is, for a desired level of high frequency dissipation, the low frequency dissipation is minimized. We will now briefly discuss the Generalized-α Method and its numerical implementation.

When the time interval of interest is divided into N time-steps of size Δt, equilibrium Eq. (2.43), using Generalized-α Method is enforced at discrete instants of time t_n where $n \in \{0,1,2,...N-1\}$ such that:

$$M\ddot{U}_{n+1-\alpha_m} + C\dot{U}_{n+1-\alpha_f} + KU_{n+1-\alpha_f} = F_{n+1-\alpha_f} \tag{2.56}$$

Where:

$$\left.\begin{aligned} x_{n+1-\alpha_m} &= (1-\alpha_m)x_{n+1} + \alpha_m x_n \\ x_{n+1-\alpha_f} &= (1-\alpha_f)x_{n+1} + \alpha_f x_n \end{aligned}\right\} \tag{2.57}$$

Equation (2.56) is supplemented by the essential initial conditions:

$$U_0 = U(t=0) \quad and \quad \dot{U}_0 = \dot{U}(t=0) \tag{2.58}$$

Using Eq. (2.57) fully discretized equilibrium equation is then expressed as:

$$\begin{aligned} &(1-\alpha_m)M\ddot{U}_{n+1} + (1-\alpha_f)C\dot{U}_{n+1} + (1-\alpha_f)KU_{n+1} \\ &= (1-\alpha_f)F_{n+1} + \alpha_f F_n - \alpha_m M\ddot{U}_n - \alpha_f C\dot{U}_n - \alpha_f KU_n \end{aligned} \tag{2.59}$$

Displacement and velocity updates in Generalized-α Method are identical to those for Newmark Method (Chung & Hulbert 1993) and are expressed in terms of algorithmic parameters β and γ. These are obtained by restricting the sum of the coefficients of their acceleration terms equal to the coefficients of the acceleration term in Taylor series expansion of U_{n+1} and $\dot{U}_{n+1}$ around t_n. Simple numerical experiments have shown that this update equation results in a monotone increase per period in the peak displacement and velocity errors (Chung & Hulbert 1993). Accordingly, expressions for displacement and velocity updates are as follows:

$$U_{n+1} = U_n + \Delta t \dot{U}_n + \Delta t^2 \left(0.5 - \beta\right) \ddot{U}_n + \Delta t^2 \beta \ddot{U}_{n+1} \tag{2.60}$$

$$\dot{U}_{n+1} = \dot{U}_n + \Delta t \left(1 - \gamma\right) \ddot{U}_n + \Delta t \gamma \ddot{U}_{n+1} \tag{2.61}$$

Substituting Eq. (2.60) and (2.61) back into Eq. (2.59), the acceleration-form of Generalized-α Method may be obtained as:

$$\mathbf{K} \ddot{U}_{n+1} = \mathbf{F_{n+1}} \tag{2.62}$$

Where effective stiffness matrix $\mathbf{K}$ and load vector $\mathbf{F_{n+1}}$ are expressed as:

$$\mathbf{K} = \left\{(1-\alpha_m)M + (1-\alpha_f)\Delta t \gamma C + (1-\alpha_f)\Delta t^2 \beta K\right\} \tag{2.63}$$

$$\begin{aligned} \mathbf{F_{n+1}} = {} & (1-\alpha_f)F_{n+1} + \alpha_f F_n - \alpha_m M \ddot{U}_n - \alpha_f C \dot{U}_n - \alpha_f K U_n \\ & - (1-\alpha_f) C \left\{\dot{U}_n + \Delta t \left(1-\gamma\right) \ddot{U}_n\right\} \\ & - (1-\alpha_f) K \left\{U_n + \Delta t \dot{U}_n + \Delta t^2 \left(0.5 - \beta\right) \ddot{U}_n\right\} \end{aligned} \tag{2.64}$$

Equation (2.62) can be solved for unknown accelerations by factorizing $\mathbf{K}$ using Cholesky decomposition (only once for linear systems with a uniform time-step). Displacements and velocities can then be obtained from Eq. (2.60) and (2.61) respectively. Hence, completing the solution at time t_{n+1}.

Generalized-α Method is second order accurate, Eq. (2.65), and unconditionally stable, Eq. (2.66), for the following conditions, (Chung & Hulbert 1993):

$$\gamma = \frac{1}{2} - \alpha_m + \alpha_f \tag{2.65}$$

$$\alpha_m \le \alpha_f \le \frac{1}{2} \quad \text{and} \quad \beta \ge \frac{1}{4} + \frac{1}{2}(\alpha_f - \alpha_m) \tag{2.66}$$

For a user defined high frequency dissipation factor δ, algorithmic parameters for unconditional stability are obtained using Eqs. (2.65) and (2.66). Accordingly for:

1) WBZ-α Method: $\delta \in [0,1]$

$$\alpha_m = -\delta, \quad \alpha_f = 0, \quad \gamma = \frac{1}{2} + \delta, \quad \beta = \frac{(1+\delta)^2}{4} \tag{2.67}$$

2) HHT-α Method: $\delta \in [0,1/3]$

$$\alpha_m = 0, \quad \alpha_f = \delta, \quad \gamma = \frac{1}{2} + \delta, \quad \beta = \frac{(1+\delta)^2}{4} \tag{2.68}$$

3) Generalized-α Method (Optimal case): $\delta \in [0,1]$

$$\alpha_m = \frac{1-3\delta}{2}, \quad \alpha_f = \frac{1-\delta}{2}, \quad \gamma = \frac{1}{2} + \delta, \quad \beta = \frac{(1+\delta)^2}{4} \tag{2.69}$$

In these equations, the lower bound of δ represents no dissipation, whereas the upper bound represents complete annihilation of high frequency modes.

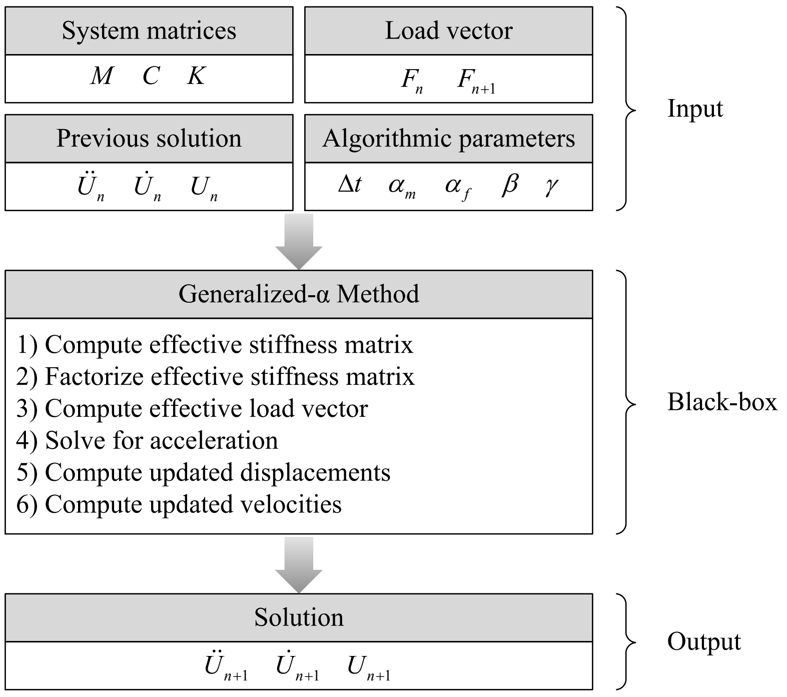
With appropriate algorithmic parameters, Generalized-α Method can be extended to:

1) HHT-α Method for $\alpha_m = 0$

2) WBZ-α Method for $\alpha_f = 0$ and

3) Newmark Method for $\alpha_m = \alpha_f = 0$

Finally, a pseudo-code to advance the solution of an input system of equations from time-step $n \in \{0,1,2,...N-1\}$ to $(n+1)$ using the Generalized-α Method can be expressed as follows:



System matrices
$M \quad C \quad K$
Load vector
$F_n \quad F_{n+1}$
Previous solution
$\ddot{U}_n \quad \dot{U}_n \quad U_n$
Algorithmic parameters
$\Delta t \quad \alpha_m \quad \alpha_f \quad \beta \quad \gamma$
Input
Generalized-α Method
1) Compute effective stiffness matrix
2) Factorize effective stiffness matrix
3) Compute effective load vector
4) Solve for acceleration
5) Compute updated displacements
6) Compute updated velocities
Black-box
Solution
$\ddot{U}_{n+1} \quad \dot{U}_{n+1} \quad U_{n+1}$
Output

Figure 2-6: Pseudo-code for direct time integration using Generalized-α Method acceleration form

2.4 Domain Decomposition Methods (DDM)

Domain Decomposition Methods (DDM) are among the most efficient and reliable techniques for the solution of engineering applications using FEM. One of the greatest advantages of this method is that the domain under analysis can be decomposed into several component sub-domains, which can then be formulated numerically, modeled and solved independently. Global solution is then obtained by assembling these sub-domains enforced by an interface condition, for example – continuity of a field variable.

The very fundamental idea of DDM can be explained using the following example for the solution of Poisson's equation, Figure 2-7, as presented in (Toselli & Widlund 2005).

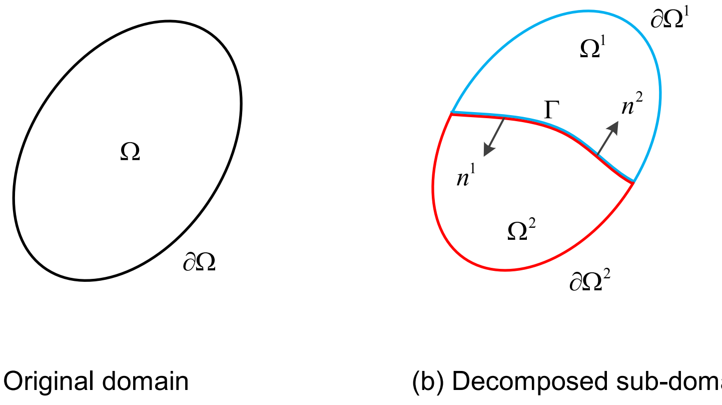


Figure 2-7: Domain decomposition method

Say we want to solve the Poisson's equation subject to boundary conditions as follows:

$$-\nabla^2 u = f \quad in \quad \Omega \tag{2.70}$$

$$u = 0 \quad on \quad \partial\Omega \tag{2.71}$$

In the above equations, u and f represent real or complex valued functions and Ω is a domain in 2 or 3 dimensions. Let the original domain Ω be decomposed into two non-overlapping sub-domains Ω^1 and Ω^2, Figure 2-7 (b), subject to following rules:

$$\begin{gathered}\Omega = \Omega^1 \cup \Omega^2 \\ \Omega^1 \cap \Omega^2 = 0 \\ \partial\Omega^1 \cap \partial\Omega^2 = \Gamma \\ \partial\Omega^1 \cap \partial\Omega > 0 \quad \partial\Omega^2 \cap \partial\Omega > 0\end{gathered} \tag{2.72}$$

As a coupled system, a new problem may now be defined as:

Sub-domain 1:	Poisson's Equation:	$-\nabla^2 u^1 = f^1 \quad in \quad \Omega^1$
	Boundary Condition:	$u^1 = 0 \quad on \ \partial\Omega^1 \setminus \Gamma$
Interface:	$u^1 = u^2 \quad on \ \Gamma$	$\frac{\partial u^1}{\partial n^1} = -\frac{\partial u^2}{\partial n^2}$
Sub-domain 2:	Poisson's Equation:	$-\nabla^2 u^2 = f^2 \quad in \quad \Omega^2$
	Boundary Condition:	$u^2 = 0 \quad on \ \partial\Omega^2 \setminus \Gamma$

Hence the basic idea is to decompose our problem into several sub-domains, each with its own discretized equations and boundary conditions, enforced by an appropriate interface condition. In several cases, Lagrange Multipliers may be used to enforce a weak statement of continuity across the interface (Bernardi et al. 1994).

Sub-domain interfaces created as a result of DDM may be non-overlapping or overlapping interfaces. Non-overlapping interfaces form the basis of Schur Compliment Approach (Roux 1990). Substructuring techniques in structural analysis uses this approach (Dodds & Lopez 1980) and the interface condition is usually applied to the unknown kinematic quantities, such as displacement, velocity or acceleration. Another application of non-overlapping DDM is the Balancing Domain Decomposition Method (BDD) where the interface problem is solved iteratively using a Preconditioned Conjugate Gradient (PCG) Method (Mandel 2005). On the other hand, overlapping interfaces form the basis of Schwarz Alternating Method (Lions 1987). Although this approach helps improving the stability of adjoining interfaces, it generates several additional unknowns as compared to non-overlapping interfaces.

A comprehensive discussion on various domain decomposition methods, theory, related algorithms, their implementation and analysis can be found under 'Domain Decomposition Methods – Algorithms and Theory' (Toselli & Widlund 2005), 'Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations' (Smith et al. 1996).

Once the problem domain has been decomposed, component sub-domains can be spatially discretized independent of each other. Within the framework of structural domains and FEM, node-cut partitioning (non-overlapping DDM) and element-cut partitioning (overlapping DDM) methods are the two most widely used discretization techniques.

2.4.1 Node-cut Partitioning

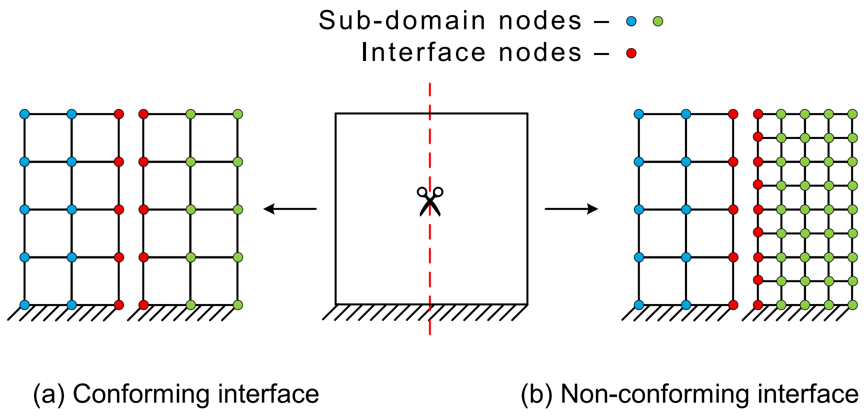


Figure 2-8: Node-cut grid partitioning

If the dividing interface is formed along the element edges, it is called node-cut partitioning, Figure 2-8. Resulting interface in this case is either a line (2D sub-domains) or a surface (3D sub-domains). Elements in this case are assigned uniquely to the sub-domains on either side of the interface. Interior nodes are private to component sub-domains, whereas nodes that fall on the interface are shared by adjacent elements and are responsible for communicating information across partitioned sub-domains.

2.4.2 Element-cut Partitioning

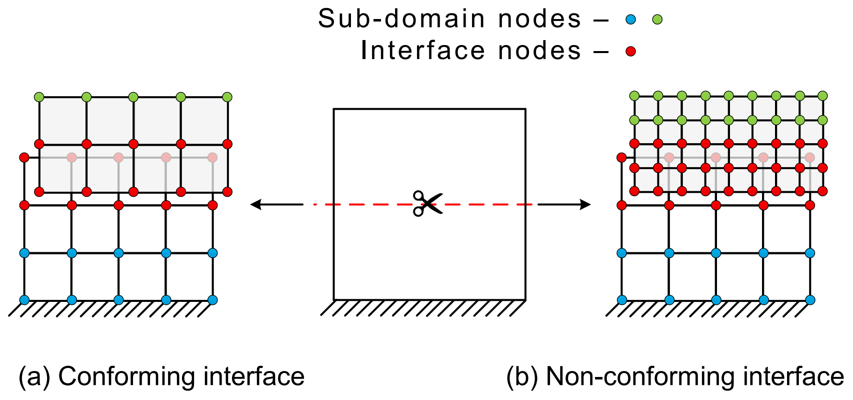


Figure 2-9: Element-cut grid partitioning

An alternate partitioning technique is the element-cut partitioning, Figure 2-9. Interface in this case is created by decomposing the domain across the element face. Accordingly, dividing interface is either a surface (2D sub-domains) or a volume (3D sub-domains). Nodes in this case are assigned uniquely to the partitioned elements and the elements which have been cut are duplicated for each sub-domain across the interface, hence creating an overlap region. Sub-domains in this case communicate not only with private nodes, but also with nodes coincident with the shared elements, which are part of other sub-domains.

Among these techniques, node-cut partitioning is a preferred approach since it generates lesser number of unknowns/interface degrees of freedom (DOF), as compared to element-cut partitioning, and it also results in a simpler interface. Once the domain has been partitioned and discretized in space, two different approaches can be employed to enforce interface conditions, (Becker et al. 2003):

1) An iterative procedure can be implemented to enforce the solution or its normal derivative or combinations to be continuous across the interface. This technique forms the foundation of the Standard Schwarz Alternating Method as discussed in (Lions 1989).
2) Another approach is to use the Lagrange Multiplier technique to achieve continuity of kinematic quantities across the interface. Several different methods have been discussed in (Tallec & Sassi 1995; Bernardi et al. 1994).

Since the Lagrange Multiplier approach does not require iterations to enforce continuity, this method is more suitable for large-scale problems. It also yields a direct global solution. However, the unknowns in this case have to be computed at the interface before enforcing a suitable condition on kinematic quantities. Accordingly, they must satisfy the *inf-sup* or equivalently the Ladyzhenskaya-Babuska-Brezzi (LBB) condition, which requires a special choice of multiplier space, such as mortar elements (Bernardi et al. 1994), or the use of special stabilization techniques (Baiocchi et al. 1992). When computing Lagrange Multipliers, another concern that comes into the picture is the grid conformity or the node-to-node connectivity at the interface. The usual domain decomposition approach would require that the nodes of a given element in a particular sub-domain intersect exactly with the nodes on an adjacent element from a sub-domain across the interface. Node-to-node connectivity at the interface can be very well modeled on structures with relatively simple boundaries; however, this is inconvenient when analyzing large structures with several integral parts. Conforming sub-domain interfaces are also difficult to model when one wishes to resolve a particular sub-domain (critical region) with a fine grid resolution, as opposed to a coarse grid in the surrounding/remote sub-domain. Since interface conditions are responsible for re-connecting partitioned sub-domains and hence obtaining the global solution, communication of information and transformation of nodal quantities across interfaces (conforming or non-conforming) has to be efficient. Another concern that needs to be answered is the gain in computational efficiency with respect to the actual accuracy that could be polluted as a result of non-conformity across the interface (Lacour & Maday 1997).

2.5 Coupling Sub-Domains in Space and Time

In previous sections we have seen how traditional FEM can be used to spatially and temporally discretize the governing equation of motion for a structural dynamic system. We have also seen how DDM can be used to partition the problem into several component sub-domains; each with its own governing equation, boundary condition and appropriate interface continuity condition. Following sections briefly describe various methodologies and algorithms used to couple sub-domains with distinct discretizations, hence concluding the review of literature and theoretical foundation in finite element multiscale coupling.

2.5.1 Mortar Finite Element Method (M-FEM)

Mortar Finite Element Method (M-FEM) is an interface discretization technique that is used to couple sub-domain grids across distinct spatial resolutions. In this method, node-to-node connectivity may or may not exist at the interface between adjacent sub-domains. Coupling is achieved in the form of point constraints, which are enforced by introducing Lagrange Multipliers, that are chosen to preserve the accuracy of the solution (Maday et al. 1988; Lamichhane & Wohlmuth 2004b).

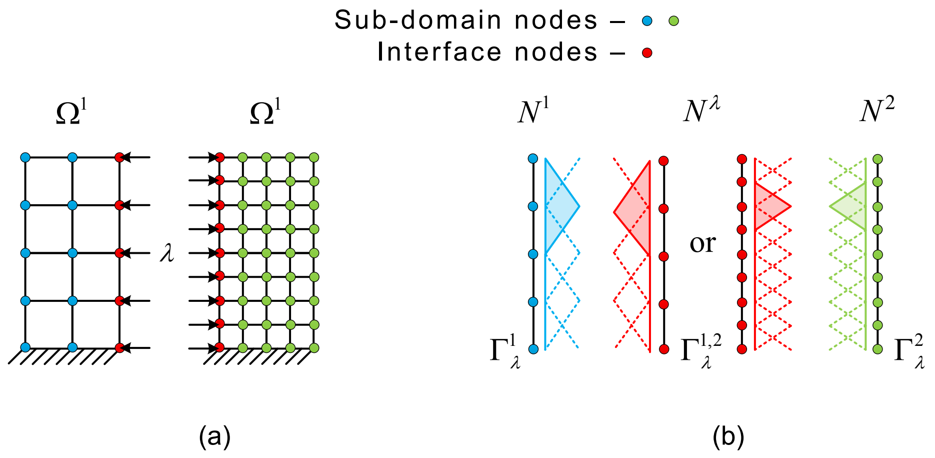


Figure 2-10: (a) Non-overlapping DD with independent (non-conforming) discretization in component sub-domains (b) Interface (Mortar) elements and corresponding Lagrange Multiplier space

Consider as an example a 2D domain, which has been decomposed into two non-overlapping, non-conforming sub-domains (Ω^1 and Ω^2) as shown in Figure 2-10 (a). Let them be spatially discretized using 4 node quadrilateral elements with standard bilinear shape functions. Let Lagrange Multipliers (λ) represent the sub-domain reactions in the form of interface fluxes or tractions. These unknowns, discretized over the interface, may be approximated using shape functions N^{λ} as shown in Figure 2-10 (b). Let the DOF associated with respective FE domains and their interfaces be as assumed in Table 2-1.

Table 2-1: Labels for sub-domain and interface degrees of freedom (DOF)

Finite element region	Corresponding number of DOF
Ω^1	a
Ω^2	b
Γ^1_λ	m (< a)
Γ^2_λ	n (< b)
$\Gamma^{1,2}_\lambda$	k

Here, if we choose coarse grid discretization to represent the Lagrange Multipliers, i.e. $k = m$, the adjacent interface on Ω^1 is referred to as the non-mortar interface, whereas the interface on Ω^2 is referred to as the mortar (glued) interface. If multiple sub-domains exist at a node on a mortar element, the usual linear interpolation shape functions associated with this particular node are replaced by a constant part (Zienkiewicz et al. 2005). The choice of mortar element discretization (for unknown Lagrange Multipliers) can be obtained from either sub-domain or independently. However, total number of DOF associated with the mortar interface should not be too rich in space so that they over constrain the coarse grid or too weak so that the constraints are not well enforced on the fine grid (Bernardi et al.

1994). Typically in M-FEM, the Lagrange Multiplier space is identical to one of the sub-domain interfaces, usually the coarser mesh. Hence for a non-overlapping DD, as shown in Figure 2-10 (a), the Lagrange Multiplier space is defined as obtained from Ω^1, i.e. $k = m$.

Recently, Park and Felippa (Park & Felippa 2000) proposed an approach using Localized Lagrange Multipliers method (LLM), as opposed to the typical global Lagrange Multipliers, Figure 2-11.

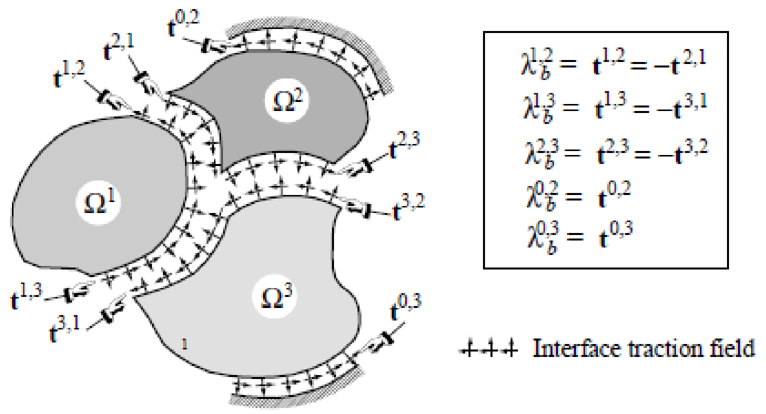

$\mathbf{t}^{0,2}$
$\mathbf{t}^{2,1}$
$\mathbf{t}^{1,2}$
Ω^2
Ω^1
$\mathbf{t}^{2,3}$
$\mathbf{t}^{3,2}$
Ω^3
$\mathbf{t}^{1,3}$
$\mathbf{t}^{3,1}$
$\mathbf{t}^{0,3}$
$\lambda_b^{1,2} = \mathbf{t}^{1,2} = -\mathbf{t}^{2,1}$
$\lambda_b^{1,3} = \mathbf{t}^{1,3} = -\mathbf{t}^{3,1}$
$\lambda_b^{2,3} = \mathbf{t}^{2,3} = -\mathbf{t}^{3,2}$
$\lambda_b^{0,2} = \mathbf{t}^{0,2}$
$\lambda_b^{0,3} = \mathbf{t}^{0,3}$
Interface traction field

Figure 2-11: Direct subdomain connection using global Lagrange Multipliers (Park & Felippa 2000)

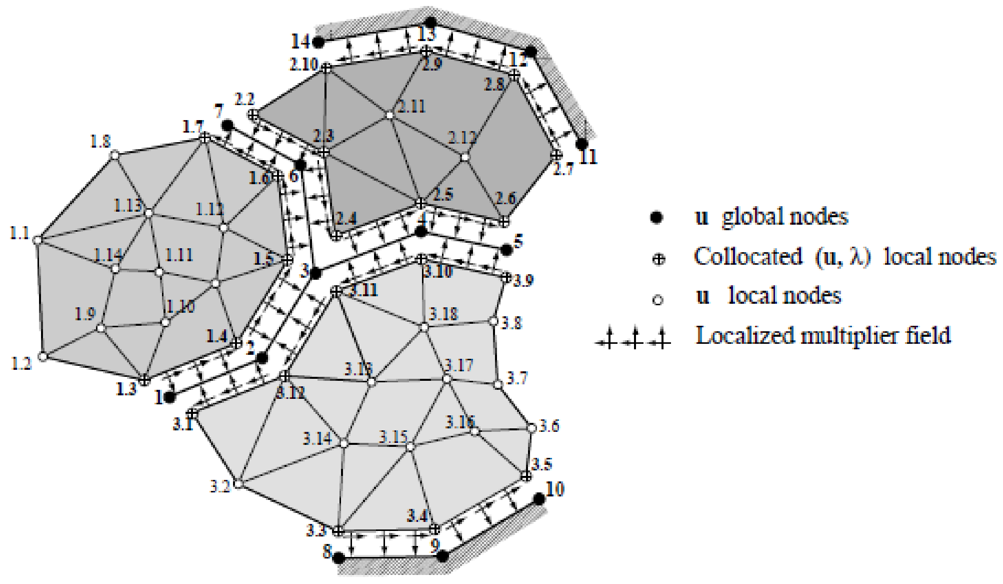


Figure 2-12: Localized-Multiplier FEM discretization (Park & Felippa 2000)

Although LLM was principally developed for contact-impact problems, it focuses on non-matching interfaces. Using LLM, a 'contact frame' with its own independent discretization is introduced between interfacing bodies, Figure 2-12, thus leading to a three field formulation. Each field variable in this case is expressed independently in order to ensure local equilibrium. They also enforce continuity of displacement across sub-domain interfaces and present an approach for positioning nodes on the contact frame.

More recently, Herry and Valentin (Herry et al. 2002) proposed a slightly modified version of global Lagrange Multiplier to connect sub-domains with non-matching grids. Henceforth, this approach will be referenced as the Hybrid M-FEM technique.

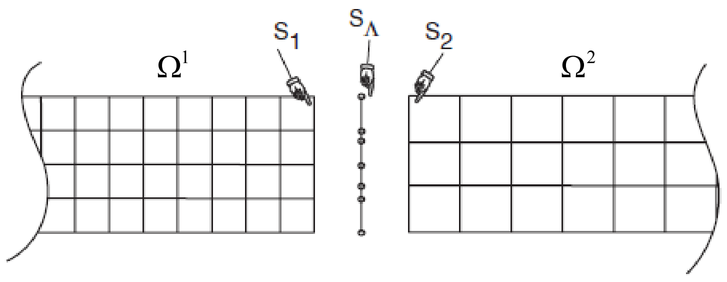


Figure 2-13: Interface definition (Herry et al. 2002)

In contrast to M-FEM, this approach uses interface discretization that is inherited from both, mortar and non-mortar interfaces; hence the name Hybrid M-FEM. For the DD shown in Figure 2-13, assume Ω^1 and Ω^2 have m and n DOF on interfaces S_1 and S_2 respectively, and let c be the total number of DOF common between these interfaces (common ≡ same coordinates). Then according to Herry and Valentin, most 'optimal' choice of interface (Lagrange Multiplier) discretization (k) is obtained as $k = m + n - c$. It is also shown that this approach results in exact continuity of kinematic quantities and equilibrium across the interface, however, only for a selected set of basis functions selectively defined for the Lagrange Multipliers, and the common interface nodes. They also note that the proposed optimal coupling can be used for elements with linear shape functions only. Regardless of the choice of interface discretization, interface connectivity constraints (Zienkiewicz et al. 2005) between adjacent sub-domains are obtained as follows: (refer to Figure 2-10 and Table 2-1 for subscript representations)

$$P_{km}^{1} = \int_{\Gamma_\lambda} N_k^{\lambda} N_m^{1} \; d\Gamma \tag{2.73}$$

$$P_{kn}^{2} = \int_{\Gamma_\lambda} N_k^{\lambda} N_n^{2} \; d\Gamma \tag{2.74}$$

Hence, the central idea in M-FEM is to decompose the domain of our interest into non-overlapping sub-domains (using node-cut partitioning) and impose a weak continuity condition across the interface by requiring that the jump of the solution is orthogonal to a suitable Lagrange Multiplier space (Bernardi et al. 1993; Bernardi et al. 1994). In the analysis of large structures, this approach has two noteworthy advantages. First, the discretization of domain can be selectively improved in localized regions, such as around corners or other features where error in solution is likely to be greatest. This will allow for greater accuracy without the computational burden associated with improving the discretization over entire global domain. Another practical benefit of this method is that it can be utilized to connect independently modeled and analyzed sub-structures in a large problem. For example, in analysis of an automobile the external framework and the chassis may be modeled independently by different engineers. It is unlikely that sub-structures like these will have exact node-to-node connectivity at the interface when assembled as a whole for a complete analysis. Transition meshes may be used in the vicinity of the interface, but this would require re-meshing and it would also make the analysis more complex and expensive. Non-conforming mortar method completely circumvents this difficulty and hence, naturally assists in multiscale modeling.

A good introduction to Mortar Methods can be found in (Bernardi et al. 1993; Bernardi et al. 1994; Lacour & Maday 1997; Maday et al. 1988; Lamichhane & Wohlmuth 2004b; Lamichhane & Wohlmuth 2004a; Lamichhane & Wohlmuth 2005).

2.5.2 Finite Element Tearing and Interconnecting (FETI)

Farhat and Roux (Farhat & Roux 1991) first introduced the Finite Element Tearing and Interconnecting (FETI) method for the solution of static problems. This method adopted Schur Compliment Approach, i.e. DD with non-overlapping sub-domains and element-cut partitioning, and introduced Lagrange Multipliers to enforce displacement compatibility at the interface nodes. Local singularities introduced as a result of static floating sub-domains are resolved in two phases: First, the rigid body modes are eliminated from each local problem and a direct scheme is used concurrently to recover partial local solution from each sub-domain. In the second phase, mode contributions from phase one are correlated to the Lagrange Multipliers through an orthogonality condition. Final coupled system of local rigid modes and Lagrange Multipliers is then solved using a Projected Conjugate Gradient (PCG) algorithm to complete the solution of the problem. This technique was later extended to the solution of transient problems (Farhat et al. 1994) along with a time parallel iterative method for structural dynamics (Farhat & Chandesris 2003; Farhat et al. 2006). Spectral stability analysis (Farhat et al. 1995) however, proved that FETI algorithm for structural dynamics is only weakly stable and instabilities grow linearly for any interface constraint. A characteristic feature of FETI, as opposed to M-FEM, is that the interface of Lagrange Multipliers is discretized using P-elements (P ≡ Polynomial) that inherit Lagrange polynomial shape functions, Figure 2-14. Polynomial shape functions are similar to linear basis functions in the sense that they are unity at the host node and zero on others.

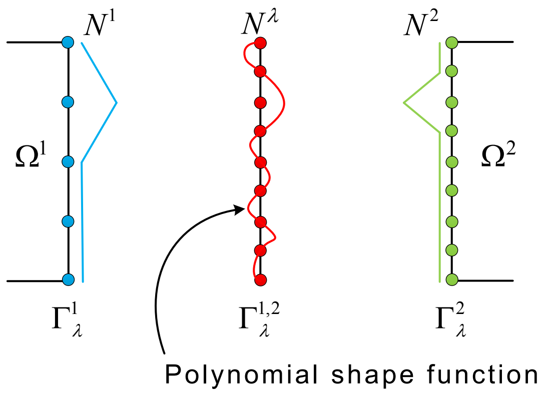


Figure 2-14: Finite element and interconnecting with Lagrange polynomial function

The upside in using polynomial shape functions is that accuracy can be improved by increasing the complexity of the polynomial, however, it requires a lot of computing time to solve a high order shape function. Another difficulty in using polynomial shape functions is the complexity in computation of interface connectivity constraints using Eq. (2.73) and (2.74). A comparison in implementing these interface connectivity constraints, and their accuracy using FEATI and M-FEM, Figure 2-15 (Lacour & Maday 1997), suggests that M-FEM has the following advantages: 1) It satisfies the compatibility condition between discrete spaces, 2) It provides an *inf-sup* condition that is independent of the discretization parameter and 3) It results in algebraic systems with well-conditioned matrices.

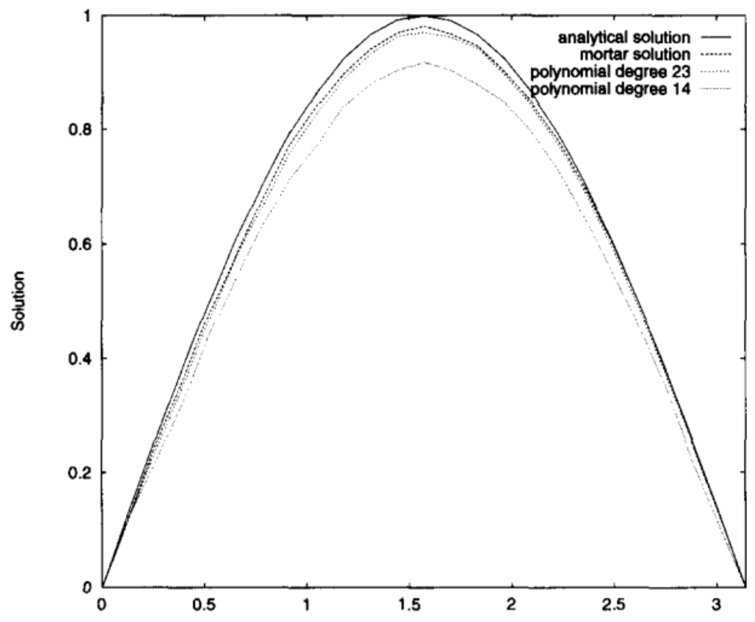


Figure 2-15: Comparison between polynomial and mortar Lagrange Multipliers (Lacour & Maday 1997)

2.5.3 Mixed Methods and Subcycling

Direct integration methods, Implicit and Explicit, both have their advantages and disadvantages. Researchers have long ago realized the need to synthesize these solution algorithms so that the time marching scheme can inherit benefits from each one of them. Similar to grid partitioning techniques, as discussed in Sections 2.4.1 and 2.4.2, the partitioning of time discretized equations is achieved through element or nodal time partitioning.

A. Element Time Partitioning

In element time partitioning, Figure 2-16, all equations associated with the elements of a particular sub-domain are integrated either implicitly or explicitly. Nodes at the interface are shared nodes and are responsible for communicating information across component sub-domains. This approach can handle shared nodes with more than two sub-domains since the assembly of system matrices is performed element-wise. Another advantage is that an element can be assigned, implicit or explicit, dynamically; subject to predefined user criterions.

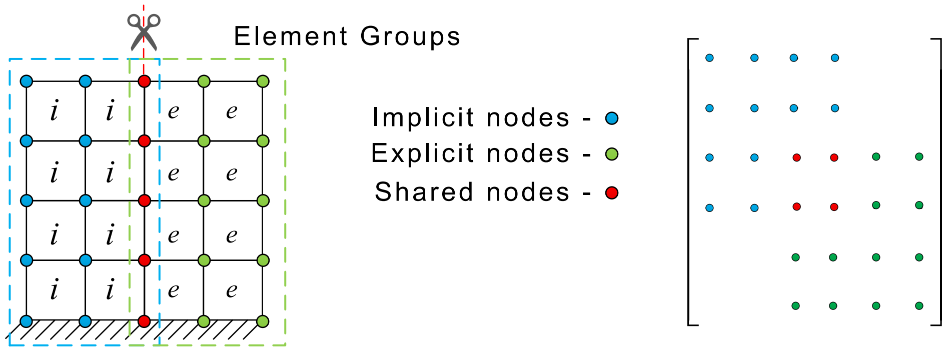


Figure 2-16: Element time partitioning

B. Nodal Time Partitioning

In nodal time partitioning, Figure 2-17, DOF associated with certain elements are integrated either implicitly or explicitly. Elements that have both implicit and explicit nodes are called interface elements and usually require special treatment. Since variables associated with these elements are integrated both implicitly and explicitly, the coupling algorithm is often difficult as compared to element partitioning technique.

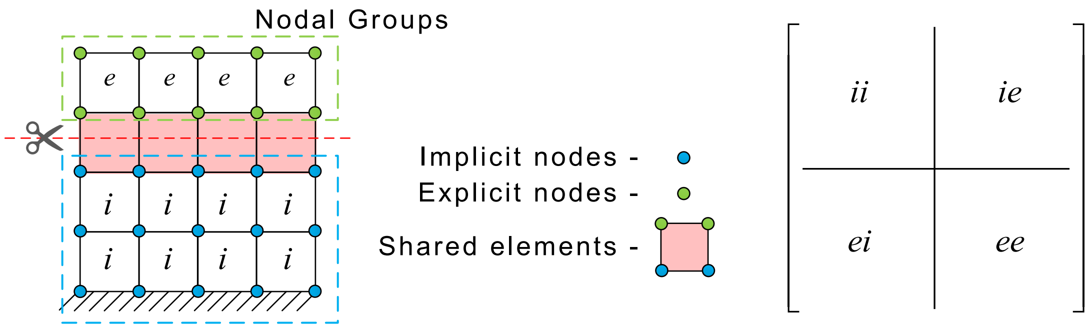


Figure 2-17: Nodal time partitioning

There are several techniques that couple different time integration schemes within a domain, with same time-step. For example, stiffer sub-domains employ implicit integration in order to circumvent the Courant time-step limit and flexible sub-domains use explicit integration to reduce overall computation costs. Such methods are referred to Mixed Methods (MM) (Belytschko et al. 1979). In classical MM, the time marching scheme uses same time-step in different sub-domains, but uses different algorithm (implicit/explicit) depending on the local sub-domain requirements. Accordingly, three different classes in MM are:

1) Implicit-Implicit (I-I) partitions
2) Explicit-Implicit (E-I) partitions
3) Explicit-Explicit (E-E) partitions

Implicit methods can adopt a direct solution approach, or an iterative method, to solve a given system of equations (Bathe & Wilson 1976). Iterative methods require less storage space, but for structural meshes the convergence rate is very poor (Belytschko et al. 1979). Direct solution method, on the other hand, yields converged results but requires more storage space. Accordingly, I-I MM (Park et al. 1977) utilizes direct solution and iterative methods as suited in different partitions.

E-I Mixed Methods were first introduced by (Belytschko & Mullen 1977) along with their stability analysis (Belytschko & Mullen 1978), allowing stiffer sub-domains to be integrated implicitly, whilst using explicit integration in flexible sub-domains. Using this element time partitioning and E-I approach, explicit nodes are integrated first and the results are used as boundary conditions for the implicit nodes. An alternate, and an easy to implement, nodal time partitioning E-I approach for linear problems was developed for the Newmark Method (Hughes & Liu 1978) and its stability analysis was provided using the Energy Method. This approach was later extended to non-linear problems (Hughes et al. 1979) with a proof for convergence (Hughes & Stephenson 1981). The E-I method was later augmented to control numerical dissipation of unwanted high frequency oscillation using a method similar to HHT-α Method (Miranda et al. 1989). Mixed time E-I method (Liu & Belytschko 1982) is another approach that uses the predictor-corrector algorithm to update implicit elements (once for every ΔT) and the Newmark Method to update explicit elements (m times every Δt).

Subcycling, or E-E integration technique, is another approach that use different time-steps in different sub-domains; however the stability and accuracy analysis of this method is significantly involved. A multi-time-step method for first-order equations, where sub-domains may be integrated using Subcycling and the entire domain is integrated at integer multiples for a global update, is presented in (Belytschko et al. 1984). This approach was later extended to second-order equations in (Smolinski 1992) with a proof of stability in (Smolinski et al. 1996). It was shown that Subcycling leads to reduction in total computer time in the analysis of large-scale problems, such as vehicle crash analysis, and can also cause an increase in accuracy for some applications (Bruijs 1990). An implicit Subcycling time integration method, based on the Trapezoidal Rule, was also proposed by (Smolinski & Wu 1998). Although energy analysis shows that this approach is unconditionally stable and conserves the same pseudo-energy as the standard algorithms, numerical tests have shown that the accuracy of the method degrades if the total cycle time or the time-step ratio becomes large.

2.5.4 GC Method

A time-space mortar method for coupling linear modal sub-domains and non-linear sub-domains in explicit structural dynamics was first proposed by Alain Combescure (Faucher & Combescure 2003). This approach introduced new opportunities in DDM, wherein each sub-domain can be discretized based on its effective loading conditions in order to reduce the computation time of the simulation. Using this technique, sub-domains that are subjected to small perturbations are assumed to represent linear elastic behavior and are suitably replaced by modal basis response of a much smaller size than the original problem. In order to establish compatibility between non-conforming grids, this approach adopts the hybrid version of M-FEM (Herry et al. 2002) as discussed earlier. This approach also extended the static FETI Method (also referred to as Dual Method) to incorporate multiple time-stepping using Newmark time integration scheme.

Another approach to handle sub-domain specific time-scale and space-scale characteristics in time-dependent non-linear problems was then proposed by Combescure and Gravouil (Gravouil & Combescure 2003). Under this approach, response from fine scale discretization and coarse scale discretization is obtained from a two-scale resolution technique inspired by the Multigrid Methods (Parsons & Hall 1990a; Parsons & Hall 1990b), Figure 2-18. They proposed an algorithm with a single iteration level to deal with both: non-linear equilibrium and two-space scale discretization; in which relaxation steps are performed using a non-linear PCG algorithm. However, this method permitted only two scale decomposition, i.e. it allowed only two spatial grids, each with its own time-scale, Figure 2-18.

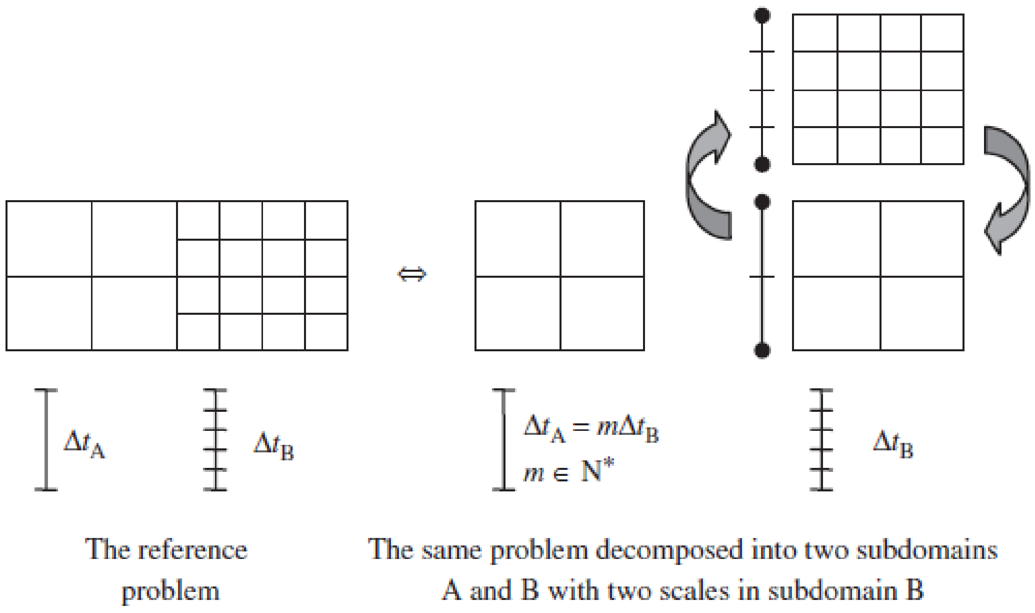

Δt_A
Δt_B
$\Delta t_A = m\Delta t_B$
$m \in N^*$
Δt_B
The reference problem
The same problem decomposed into two subdomains A and B with two scales in subdomain B

Figure 2-18: Treatment of incompatible meshes using two-grid approach (Gravouil & Combescure 2003)

Gravouil and Combescure (Combescure & Gravouil 2001; Combescure et al. 2003; Mahjoubi et al. 2009) later proposed a general time-space multi-scale method (GC Method) for the solution of transient problems based on node-cut grid partitioning. Using this method, partial differential equations were discretized independently over sub-domains, solved individually and then globally over the interface using Lagrange Multipliers. Assuming kinematic quantities, such as displacement, velocity and acceleration, are linear over the interface; Lagrange Multipliers are introduced to enforce continuity across the interface. Equilibrium equations for the coupled system are then formulated as a sum of total energy, obtained from each sub-domain and augmented by the interface energy. GC Method uses Hybrid M-FEM (Herry et al. 2002) and Newmark time integration scheme for the evolution of kinematic quantities. It is however shown that GC Method requires computation of interface reactions (Lagrange Multipliers) at the smallest time-step, which is indeed computationally very expensive for large-scale structures with large interfaces. They first present a procedure to couple different Newmark schemes (implicit/explicit) in each sub-domain with the same time-step. By enforcing continuity of velocities across sub-domain interfaces and utilizing the Energy Method, they show that this approach is energy preserving. They also show that the method is stable if and only if the interface energy is zero. For Newmark time integration scheme, the GC Method has zero interface energy under one of the following condition:

1) For continuity of Accelerations: γ is constant for each sub-domain
2) For continuity of Velocities: γ and β are independent parameters
3) For continuity of Displacements: $(2\beta - \gamma = 0)$ for each sub-domain

Therefore, it is shown that continuity of velocities at the interfaces leads to a stable algorithm. They further present their approach for non-linear cases with same time-step in all sub-domains, and linear cases with different time-steps in different sub-domains. GC Method was also extended to explicit-implicit formulation for non-linear problems with different time-steps in different sub-domains (Combescure et al. 2003). They derive coupled equations and stability conditions for both material and geometric non-linearity. A monolithic energy conserving approach, to couple heterogeneous time integrators such as Newmark, HHT-α, Simo, Krenk and Velocity Verlet with incompatible time-steps, was also built on the foundation of GC Method (Mahjoubi & Gravouil 2011) and a FE software enabling implicit-explicit multi-time-stepping co-computations, based on GC Method, was also developed by (Brun et al. 2012). However, it was shown that GC Method has unsuitable features of being dissipative on the interface between sub-domains. It was also shown that sub-domain computations are stable as long as Newmark stability requirements are fulfilled within each sub-domain, but the accuracy of the time-integration scheme reduced by one order when multi-time-stepping was introduced. Hence, indicating that for second order time-integration schemes (such as Newmark Method), the multi-time-step GC Method only lead to first order accuracy.

2.5.5 PH Method

Prakash and Hjelmstad, or PH Method, (Prakash & Hjelmstad 2004) further extended FETI algorithms and improvised on existing formulation for multiple time-stepping approach laid down by GC Method. They adopt Newmark time-integration method to selectively discretize component sub-domains, allowing distinct time-steps (with integer ratios) and implicit and/or explicit algorithms. Theoretical foundation of the newly developed multi-time-scale algorithm, in this dissertation, is largely inspired from the PH Method. Accordingly, we shall now take a brief look at this approach, as presented in (Prakash & Hjelmstad 2004).

For simplicity, we will assume the decomposition of a continuous domain Ω into two non-overlapping sub-domains: Ω^A and Ω^B as shown in Figure 2-19 (a) and (b). It is also assumed that Ω^A is integrated with a larger time-step ΔT (corresponding to the global time-step) and Newmark parameters $\left(\beta^A, \gamma^A\right)$ whereas Ω^B is integrated with a smaller time-step Δt (β^B, γ^B) such that the time-step ratio $(\Delta T / \Delta t = \xi)$ is an integer. Intermediate time-steps in integrating Ω^B are denoted by η such that $\eta = 0$ corresponds to time $T \equiv t_n$ and $\eta = \xi$ corresponds to $(T + \Delta T) = (T + \xi \Delta t) \equiv t_{n+1}$, see Figure 2-19 (c).

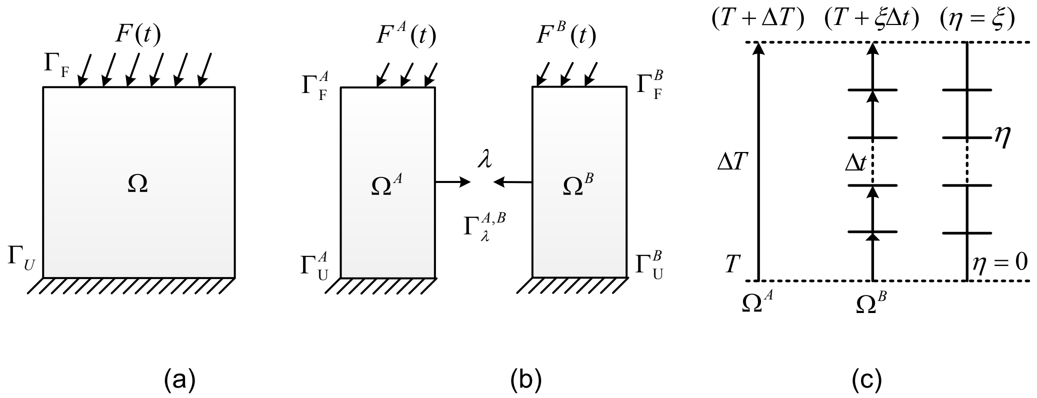


Figure 2-19: (a) Structural domain under consideration (b) Decomposed sub-domains linked through Lagrange Multipliers at the interface (c) sub-domain time-stepping parameters and intermediate time-step counter

Consider the structural domain Ω in Figure 2-19 (a) has prescribed displacements over Γ_U and transient external force over Γ_F. Using FE formulation for an un-damped system with linear elastic constitutive law the problem under consideration can be expressed as:

$$M\ddot{U}(t)+KU(t)=F(t) \qquad \forall t\in[0,T] \tag{2.75}$$

$$I.C \;\rightarrow\; U(0)=U_0 \;\;,\;\; \dot{U}(0)=\dot{U}_0 \tag{2.76}$$

$$B.C \;\rightarrow\; U(t)=U_{\mathrm{D}}(t) \;\text{ on }\; \Gamma_{\mathrm{U}} \;\;,\;\; F(t)=F_{\mathrm{D}}(t) \;\text{ on }\; \Gamma_{\mathrm{F}} \tag{2.77}$$

Using Newmark Method, fully-discretized equilibrium equation for the continuous domain Ω along with displacement and velocity updates is expressed as:

$$M\ddot{U}_{n+1}+KU_{n+1}=F_{n+1} \tag{2.78}$$

$$U_{n+1} = U_n + \Delta t \dot{U}_n + \Delta t^2 (0.5 - \beta) \ddot{U}_n + \Delta t^2 \beta \ddot{U}_{n+1} \tag{2.79}$$

$$\dot{U}_{n+1} = \dot{U}_n + \Delta t (1 - \gamma) \ddot{U}_n + \Delta t \gamma \ddot{U}_{n+1} \tag{2.80}$$

In order to formulate constituent equations for Ω^A and Ω^B augmented by an interface continuity condition, it is assumed that the interface reactions are discretized over $\Gamma_\lambda^{A,B}$ and represented using Lagrange Multipliers (λ). The Lagrangian for coupled sub-domains is then expressed as:

$$\mathcal{L} = \left[\frac{1}{2} \dot{U}^{A^T} M^A \dot{U}^A - \frac{1}{2} U^{A^T} K^A U^A \right] + \left[\frac{1}{2} \dot{U}^{B^T} M^B \dot{U}^B - \frac{1}{2} U^{B^T} K^B U^B \right] \tag{2.81}$$

As proposed by GC Method, PH Method imposes continuity of velocities across sub-domain interfaces using a linear constraint equation such as:

$$L^A \dot{U}^A + L^B \dot{U}^B = 0 \tag{2.82}$$

Where L represents a Boolean matrix that picks interface degrees of freedom from respective sub-domains and projects them onto $\Gamma_\lambda^{A,B}$.

Combining the constraint Eq. (2.82) with Eq. (2.81), the Lagrangian in terms of interface reactions (λ), and its variation is expressed as:

$$\tilde{\mathcal{L}} = \sum_i^{A,B} \left[\frac{1}{2} \dot{U}^{i^T} M^i \dot{U}^i - \frac{1}{2} U^{i^T} K^i U^i \right] + \lambda^T \sum_i^{A,B} L^i \dot{U}^i \tag{2.83}$$

$$\delta\tilde{\mathcal{L}} = \sum_{i}^{A,B}\left[\delta\dot{U}^{i^T} M^i \dot{U}^i - \delta U^{i^T} K^i U^i + \delta\dot{U}^{i^T} L^{i^T} \lambda\right] + \delta\lambda^T \sum_{i}^{A,B} L^i \dot{U}^i \tag{2.84}$$

External virtual work done on the system is expressed by:

$$\delta\mathcal{W} = \sum_{i}^{A,B} \delta U^{i^T} F^i \tag{2.85}$$

Applying Hamilton's principle:

$$\int_{t_1}^{t_2} (\delta\mathcal{L} + \delta\mathcal{W})\, dt = 0 \tag{2.86}$$

$$\int_{t_1}^{t_2} \sum_{i}^{A,B}\left[\delta\dot{U}^{i^T} (M^i \dot{U}^i + L^{i^T} \lambda) - \delta U^{i^T} (K^i U^i - F^i)\right] dt + \int_{t_1}^{t_2} \delta\lambda^T \left[\sum_{i}^{A,B} L^i \dot{U}^i\right] dt = 0 \tag{2.87}$$

Integrating first term by parts and assuming $\delta U^i(t_1) = \delta U^i(t_2) = 0$:

$$\int_{t_1}^{t_2} \sum_{i}^{A,B} -\delta U^{i^T} \left[M^i \ddot{U}^i + K^i U^i + L^{i^T} \dot{\lambda} - F^i\right] + \delta\lambda^T \left[\sum_{i}^{A,B} L^i \dot{U}^i\right] dt = 0 \tag{2.88}$$

Using fundamental theorem for calculus of variations, Eq. (2.88) can be expressed as:

$$M^i \ddot{U}^i + K^i U^i + L^{i^T} \Lambda = F^i \quad \forall i : A, B \tag{2.89}$$

$$\sum_{i}^{A,B} L^i \dot{U}^i = 0 \tag{2.90}$$

Introducing time-stepping parameters from Figure 2-19 (c), fully discretized equations for Ω^A and Ω^B are expressed as:

$$M^A\ddot{U}^A_\xi + K^A U^A_\xi + L^{A^T}\Lambda_\xi = F^A_\xi \tag{2.91}$$

$$M^B\ddot{U}^B_\eta + K^B U^B_\eta + L^{B^T}\Lambda_\eta = F^B_\eta \quad \forall\eta\in[1,2\ldots\xi] \tag{2.92}$$

Contrary to GC Method, PH Method enforces continuity of velocities only at the global time-step $(T+\Delta T)$ as:

$$L^A\dot{U}^A_\xi + L^B\dot{U}^B_\xi = 0 \tag{2.93}$$

In order to solve the system of equations represented by (2.91), (2.92) and (2.93), kinematic quantities from Ω^A are decomposed into two parts as follows:

$$U^A_\xi = V^A_\xi + W^A_\xi \tag{2.94}$$

Note: Eq. (2.94) represents the decomposition of displacements only; however, similar decomposition is adopted for velocity and acceleration vectors. Equation (2.91) is then expressed as a 'Free' problem (under the action of external forces) and a 'Link' problem (under the action of interface reactions) as follows:

$$M^A\ddot{V}^A_\xi + K^A V^A_\xi = F^A_\xi \tag{2.95}$$

$$M^A\ddot{W}^A_\xi + K^A W^A_\xi = -L^{A^T}\Lambda_\xi \tag{2.96}$$

Using Newmark Method, Eq. (2.95) is solved independently to compute the free response of Ω^A at time $(T+\Delta T)$. Resulting solution (V_ξ^A, $\dot{V}_\xi^A$ and $\ddot{V}_\xi^A$) is then linearly interpolated using Eq. (2.97), shown for displacements only, at intermediate time-steps corresponding to sub-domain Ω^B, i.e. for time-instants where $\eta \in [1, 2\ldots\xi]$.

$$V_\eta^A = \left(1 - \frac{\eta}{\xi}\right) U_T^A + \left(\frac{\eta}{\xi}\right) V_\xi^A \tag{2.97}$$

$$W_\eta^A = \left(\frac{\eta}{\xi}\right) W_\xi^A \tag{2.98}$$

Knowing the solution at intermediate time-steps, corresponding 'free' residuals (δf_j) are computed as:

$$\delta f_\eta = F_\eta^A - M^A \ddot{V}_\eta^A - K^A V_\eta^A \tag{2.99}$$

Similarly, 'link' residuals (δl_j) from Eq. (2.96) are obtained as:

$$\delta l_\eta = -L^{A^T} \Lambda_\eta - M^A \ddot{W}_\eta^A + K^A W_\eta^A \tag{2.100}$$

Now, in order to enforce equilibrium of Ω^A for every intermediate time-step, the combined residual from Eq. (2.99) and Eq. (2.100) is set to zero, i.e. $\left(\delta f_\eta + \delta l_\eta\right) = 0$.

$$\therefore \delta f_\eta - L^{A^T} \Lambda_\eta - M^A \ddot{W}_\eta^A + K^A W_\eta^A = 0 \tag{2.101}$$

Introducing Eq. (2.98), Eq. (2.101) can be expressed as:

$$\delta f_\eta - L^{A^T}\Lambda_\eta - \left(\frac{\eta}{\xi}\right)\left(M^A\ddot{W}_\xi^A + K^A W_\xi^A\right) = 0 \tag{2.102}$$

However, $M^A\ddot{W}_\xi^A + K^A W_\xi^A = -L^{A^T}\Lambda_\xi$

$$\therefore \delta f_\eta - L^{A^T}\Lambda_\eta + \left(\frac{\eta}{\xi}\right)L^{A^T}\Lambda_\xi = 0 \tag{2.103}$$

$$\therefore \Lambda_\eta = L^A\delta f_\eta + \left(\frac{\eta}{\xi}\right)\Lambda_\xi \qquad \because L^A L^{A^T} = I \tag{2.104}$$

Using the result from Eq. (2.104), intermediate interface reactions from Eq. (2.92) are condensed out and the equilibrium equation for Ω^B is expressed as:

$$M^B\ddot{U}_\eta^B + K^B U_\eta^B + \left(\frac{\eta}{\xi}\right)L^{B^T}\Lambda_\xi = F_\eta^B - L^A\delta f_\eta \quad \forall \eta \in [1, 2 \ldots \xi] \tag{2.105}$$

The final set of coupled equations, including equilibrium Eqs. (2.91), (2.105) and interface condition, Eq. (2.93), are solved using bordered system approach and the global solution to the original problem under consideration is obtained by summing responses from component sub-domain at global time-step ΔT.

Using Energy Method, it is shown that this method is unconditionally stable, energy preserving and computationally efficient than the former GC Method. Since the interface continuity condition is enforced at global time-steps only, corresponding interface reactions (Lagrange Multipliers) are also computed at the global time-step. This makes the PH

Method $\Delta T / \Delta t = \xi$ times faster than GC Method, Figure 2-20 (Prakash & Hjelmstad 2004).

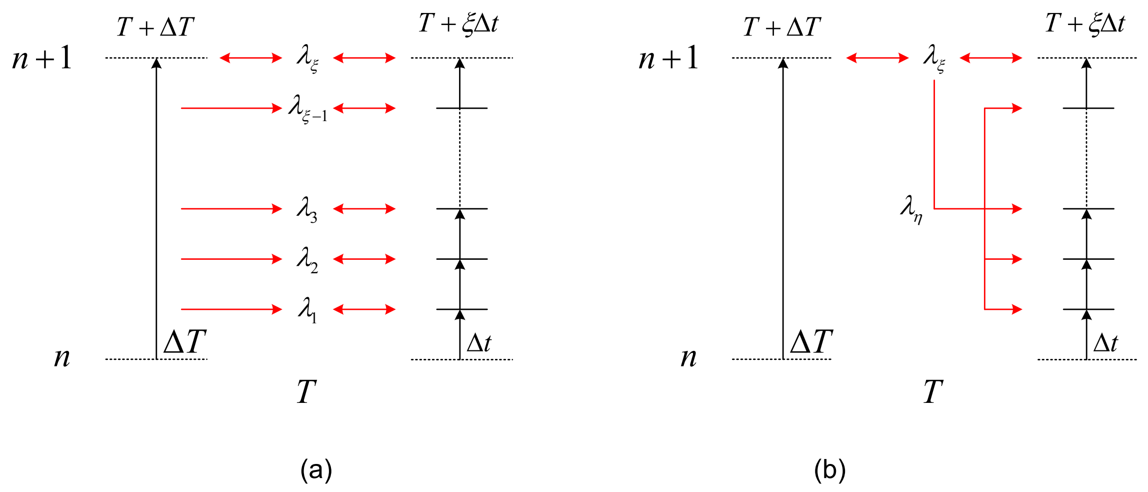


Figure 2-20: Comparison between (a) GC Method and (b) PH Method

However, the drawback in using PH Method is that Lagrange Multipliers are expressed in terms of 'free' residuals, Eq. (2.104), which are obtained from the coarse time-step sub-domain Eq. (2.99). This makes the fine time-step sub-domain dependent on other sub-domains, making it impossible to solve multiple sub-domains concurrently. Therefore, PH Method allows coupling only two sub-domains at a time; and consequently for multiple sub-domain coupling, one needs to implement a recursive solution algorithm (Prakash 2007) that can solve component sub-domains – one pair at a time. This dependence between component sub-domains significantly restricts the computational efficiency of multiple time-scale coupling.

Chapter 3: MGMT Formulation

3.1 Approach and Methodology

Since FEM is well established with vast amounts of literature in its theoretical background, mathematical foundation and applications using computer simulations; this dissertation builds upon the existing FEM in order to implement concurrent multiple grid and multiple time-scale coupling. We propose a hybrid approach which may be considered as a combination of:

1) Multiple grid formulation that allows efficient coupling of 'independently' discretized FE sub-domains
2) Multiple time-scale formulation that allows concurrent multi-time-stepping with energy preserving time interfaces

The framework for this approach is largely based on the fundamental principles of DDM, used to selectively decompose a structure into component sub-domains. Post decomposition, component sub-domains can be independently discretized in space. Subsequently, multiple grids with conforming/non-conforming interfaces can be coupled together using M-FEM and Lagrange Multipliers that are introduced as interface reactions. Space discretized equations of motion from component sub-domains can further be discretized in time, distinct from each other, allowing sub-domain specific time-stepping. Communication between component sub-domains is desired at the largest/global time-step and is achieved in a manner that preserves global/local sub-domain energy balance.

A distinction between traditional FE approach and the proposed hybrid methodology can be illustrated as follows:

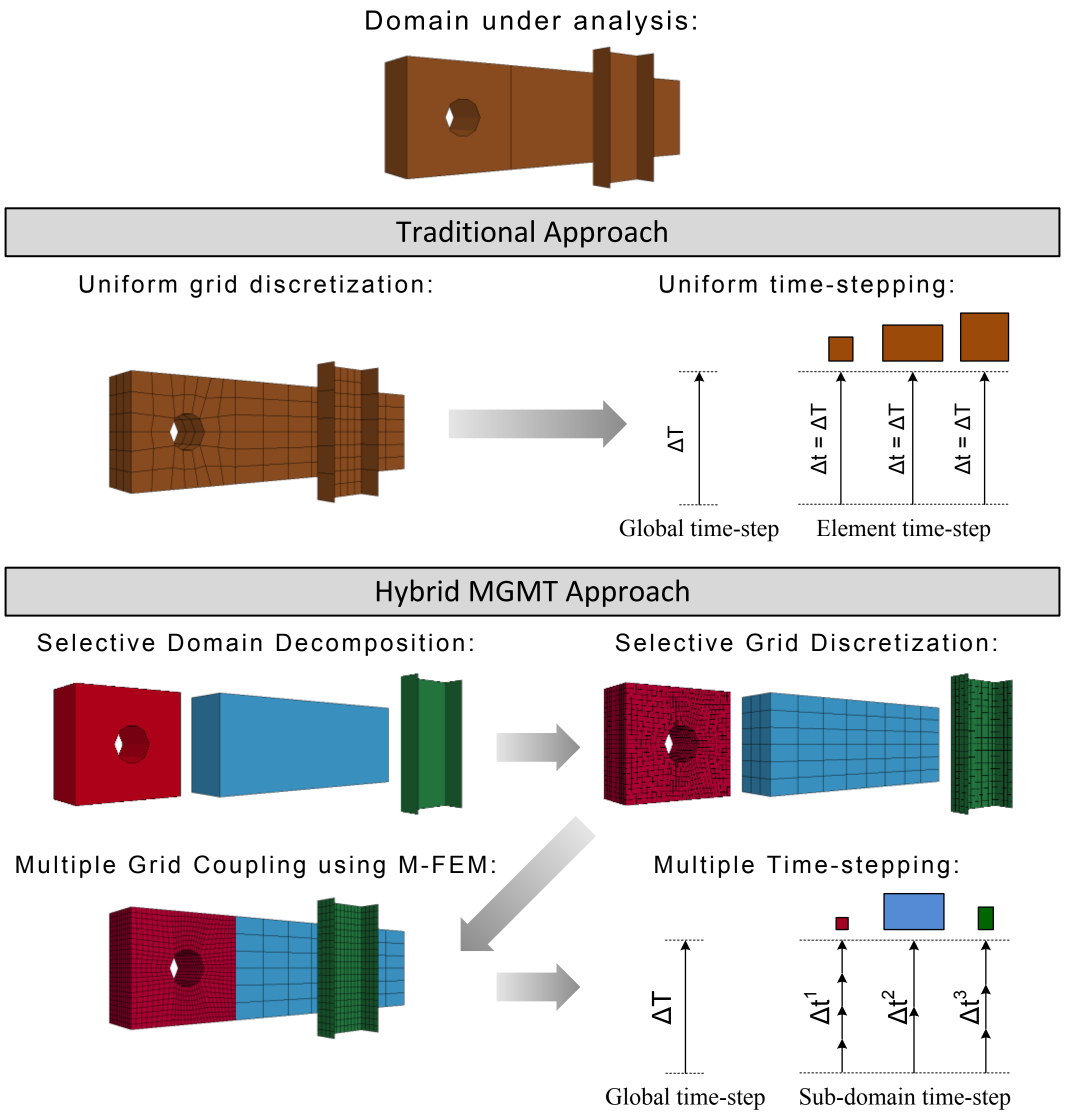


Figure 3-1: Hybrid MGMT approach

Following sections elaborate on every constituent step of this hybrid approach and rigorously derive necessary formulation that is required for the successful implementation of the proposed MGMT Method.

3.2 Domain Decomposition for Structural Dynamics

In this section, we present the DDM approach for deriving coupled equations for component sub-domains augmented with an appropriate interface condition. We limit our discussion to linear structural dynamic systems, as derived in Section 2.1.

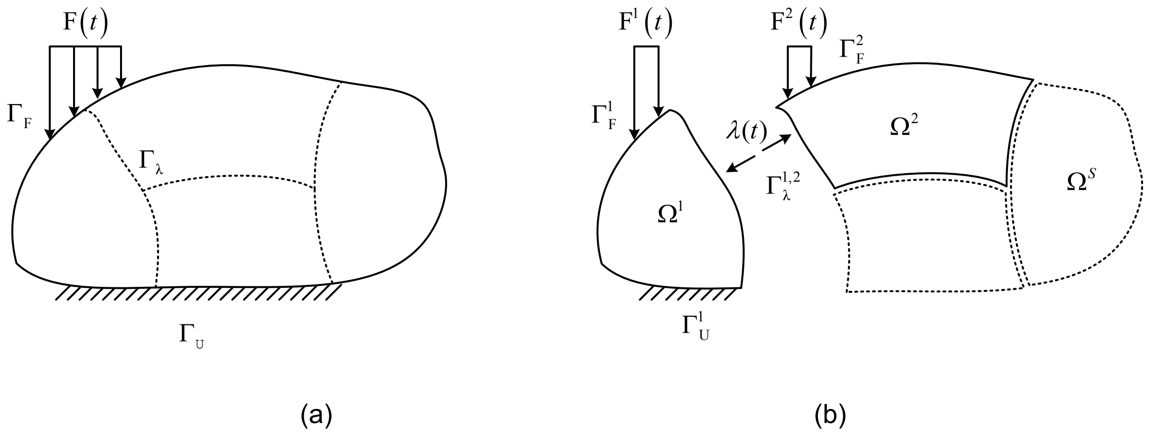


Figure 3-2: Structural domain decomposition (a) Domain under analysis (b) Decomposed sub-domains with inherited boundary conditions and augmented interface reactions

Consider a continuous domain Ω with prescribed displacements over Γ_U and prescribed tractions over Γ_F as shown in Figure 3-2 (a). Using finite element discretization in space, the governing equation is expressed as:

$$M\ddot{U}(t) + C\dot{U}(t) + KU(t) = F(t) \tag{3.1}$$

$$I.C \quad \rightarrow \quad U(0) = U_0 \quad , \quad \dot{U}(0) = \dot{U}_0 \tag{3.2}$$

$$B.C \quad \rightarrow \quad U(t) = U_D(t) \quad \text{on} \quad \Gamma_U \quad , \quad F(t) = F_D(t) \quad \text{on} \quad \Gamma_F \tag{3.3}$$

Here, M represents the mass, C is damping and K represents the domain stiffness. Primary unknowns are displacements, velocities and accelerations; as represented by $U(t)$, $\dot{U}(t)$ and $\ddot{U}(t)$. Initial displacements and velocities are defined by U_0 and $\dot{U}_0$ whereas $U_D(t)$ and $F_D(t)$ represent the prescribed displacements and force boundary conditions respectively. Energy balance equation corresponding to Eq. (3.1) can be expressed by pre-multiplying $\dot{U}^T$ (velocities transpose), followed by integration:

$$\frac{d}{dt}\left(\frac{1}{2}\dot{U}^T M\dot{U} + \frac{1}{2}U^T KU\right) = \dot{U}^T F - \dot{U}^T C\dot{U} \tag{3.4}$$

Using DDM let us now decompose our original domain into S component sub-domains such that $\Gamma_\lambda^{i,j}$ represents the interface between Ω^i and Ω^j, as shown in Figure 3-2 (b). If Lagrange Multipliers (λ) in the form of fluxes or tractions are used to represent interface reactions between connecting sub-domains the resulting equation of motion for a component sub-domain is obtained as:

$$M^i\ddot{U}^i(t) + C^i\dot{U}^i(t) + K^iU^i(t) = F^i(t) - L^{i^T}\lambda(t) \quad (i = 1,2\ldots S) \tag{3.5}$$

And the corresponding energy balance equation is expressed as:

$$\frac{d}{dt}\left(\frac{1}{2}\dot{U}^{i^T} M^i\dot{U}^i + \frac{1}{2}U^{i^T} K^iU^i\right) = \dot{U}^{i^T} F^i - \dot{U}^{i^T} C^i\dot{U}^i - \dot{U}^{i^T} L^{i^T}\lambda \quad (i = 1,2\ldots S) \tag{3.6}$$

Continuity of an unknown variable (say x) across sub-domain interface $\Gamma_\lambda^{i,j}$ can be represented by a linear constraint equation such as:

$$L^i x^i(t) = L^j x^j(t) \quad or \quad L^i x^i(t) - L^j x^j(t) = 0 \tag{3.7}$$

In above expressions, L represents a multi-constraint operator that is respectively zero and non-zero for interior and interface DOF respectively. It is used to project required constraints over the selected set of interface DOF. Variable x may represent displacement (d-continuity), velocity (v-continuity) or acceleration (a-continuity). Since Eq. (3.5) and (3.7) are coupled, it is necessary to ensure that enforcing continuity of variable x does not influence the global energy balance. It is also important to ensure that the time integration of Eq. (3.5), augmented with the interface condition, yields a stable solution within component sub-domains and globally. These requirements will help us select an appropriate variable for the interface condition. Accordingly, we enforce the following constraints on Eq. (3.6):

1) Energy is conserved within respective sub-domains, that is local equilibrium is satisfied
2) Interface energy, produced as a result of interface reactions, is identically zero

These constraints are reasonable since the global solution (obtained as an aggregate of local solutions) can be stable if and only if the solution is stable within respective sub-domains. Also, if the interface energy contributions are positive, numerical integration of Eq. (3.5) will eventually escalate with an unstable solution and if the interface energy is negative, artificial damping will be introduced across sub-domain interfaces. In MGMT, domain decomposition represents a mathematical division of a continuous domain and therefore, the dividing interfaces do not represent physical features within the system.

Accordingly, any energy produced as a result of introducing Lagrange Multipliers must be identically zero. This will ensure stable gluing of adjacent sub-domains and seamless communication across connecting interfaces.

Global energy balance equation, as obtained from component sub-domain contributions, can now be expressed as:

$$\sum_{i=1}^{S}\frac{d}{dt}\left(\frac{1}{2}\dot{U}^{i^T}M^i\dot{U}^i+\frac{1}{2}U^{i^T}K^iU^i\right)=\sum_{i=1}^{S}\left(\dot{U}^{i^T}F^i-\dot{U}^{i^T}C^i\dot{U}^i-\dot{U}^{i^T}L^{i^T}\lambda\right) \tag{3.8}$$

By enforcing the constraints discussed above, we obtain the following condition:

$$\sum_{i=1}^{S}\dot{U}^{i^T}L^{i^T}\lambda=0 \tag{3.9}$$

Equation (3.9) represents the necessary condition to ensure global energy balance; specifically it suggests that the interface energy should be identically zero. Comparing this result with Eq. (3.7), we see that zero interface energy naturally yields continuity of velocities across sub-domain interfaces. The final set of equations for coupled sub-domains, now augmented by interface condition, Eq. (3.9), can now be represented as:

$$M^i\ddot{U}^i(t)+C^i\dot{U}^i(t)+K^iU^i(t)=F^i(t)-L^{i^T}\lambda(t)\quad\left(i=1,2\ldots S\right) \tag{3.10}$$

$$\sum_{i=1}^{S}L^i\dot{U}^i\left(t\right)=0 \tag{3.11}$$

3.3 Multiple Grid Coupling

One clear advantage of using DDM is the ability to choose independent resolutions in component sub-domains so as to obtain a discretization that is well suited to the local characteristics of the solution to be approximated. In this section we will discuss how interface constraints can be implemented using node-cut partitioning, hence allowing multiple grid coupling.

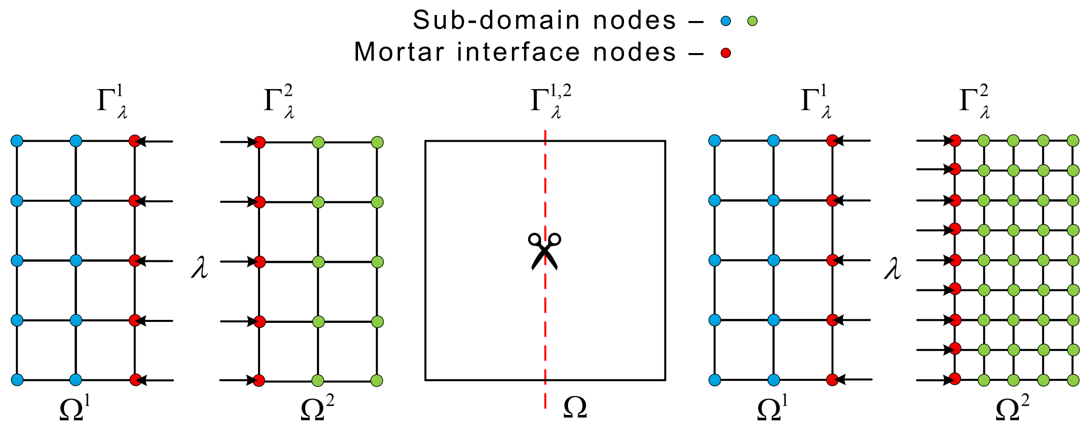


Figure 3-3: Domain decomposition and resulting conforming/non-conforming grids

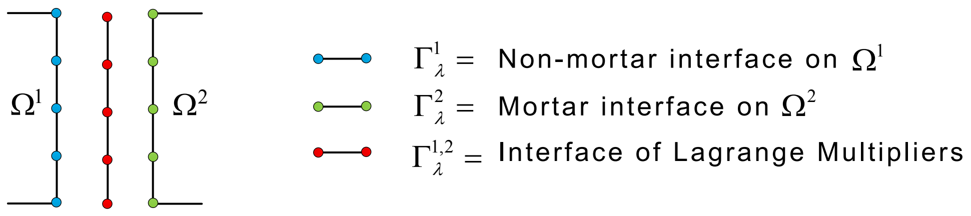
Table 3-1: Labels for FE degrees of freedom

FE region	Corresponding number of DOF
Ω^1	a
Ω^2	b
Γ^1_λ	m (< a)
Γ^2_λ	n (< b)
$\Gamma^{1,2}_\lambda$	k

Consider the decomposition of a continuous region Ω (2D) into two sub-domains, Ω^1 and Ω^2 which are joined together by introducing Lagrange Multipliers (interface reactions) at the dividing interface, Figure 3-3. Since we are using node-cut partitioning, the interface for Lagrange Multipliers is a 1D segment. As discussed earlier, the choice for mortar interface discretization is arbitrary, but fixed. In our derivation for multiple grid coupling, we choose the coarse grid interface as the non-mortar surface; therefore DOF assigned to the interface of Lagrange Multipliers is equal to the interface DOF from Ω^1, that is $k = m$.

3.3.1 Coupling Conforming Grids

An interface between adjacent sub-domains is said to be conforming if and only if $m = n$ and the coordinates for corresponding DOF are coincident. Therefore, we have $k = m = n$.



Ω^1
Ω^2
Γ_λ^1 = Non-mortar interface on Ω^1
Γ_λ^2 = Mortar interface on Ω^2
$\Gamma_\lambda^{1,2}$ = Interface of Lagrange Multipliers

Figure 3-4: Interface of Lagrange Multipliers (conforming grids)

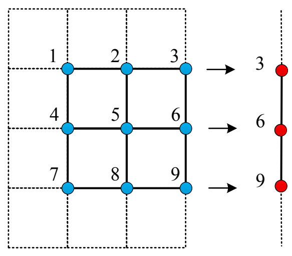
In this particular case, multi-constraint operator L is simply a Boolean projection matrix with 1's and 0's assigned to sub-domain interface and interior DOF respectively. If B^1 and B^2 represent the Boolean projection matrix for Ω^1 and Ω^2 respectively, then the sub-domain DOF, located on interface Γ_λ^i $(i = 1, 2)$ can be selectively projected as:

$$x_m^1 = B_{ma}^1 x_a^1 \qquad \therefore L^1 = B^1 \tag{3.12}$$

$$x_n^2 = B_{nb}^2 x_b^2 \qquad \therefore L^2 = B^2 \tag{3.13}$$

In the above expressions, variable x represents the primary kinematic unknowns, such as: displacement (U), velocity $(\dot{U})$ or acceleration $(\ddot{U})$.

As an example for implementing and constructing the Boolean projection matrix, consider the following illustration, Figure 3-5:



$$B = \begin{array}{c|ccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \hline 3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

Figure 3-5: Reference sub-domain and corresponding Boolean projection matrix

3.3.2 Coupling Non-Conforming Grids

An interface between adjacent sub-domains is said to be non-conforming if $m \neq n$ and/or the coordinates of corresponding DOF are not coincident. The total number of Lagrange Multipliers in this case is selected as $k = m$, as shown in Figure 3-6.

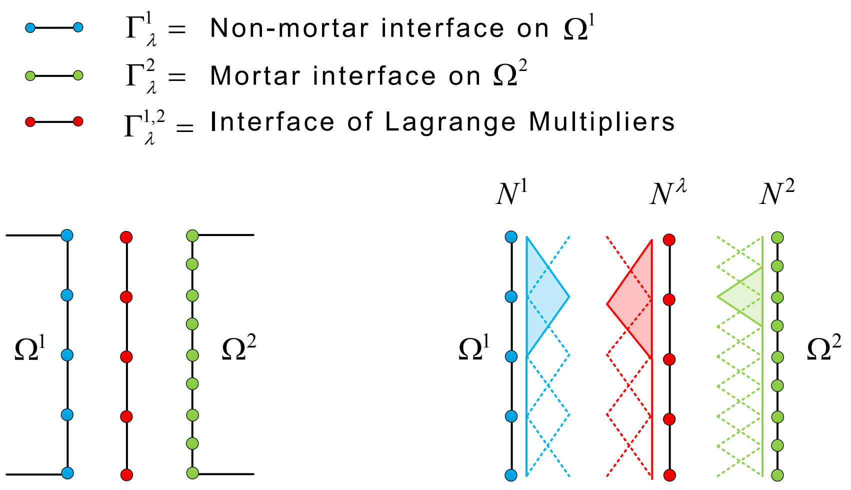

Γ_{λ}^{1} = Non-mortar interface on Ω^1
Γ_{λ}^{2} = Mortar interface on Ω^2
$\Gamma_{\lambda}^{1,2}$ = Interface of Lagrange Multipliers
Ω^1
Ω^2
N^1
N^λ
N^2
Ω^1
Ω^2

Figure 3-6: Interface of Lagrange Multipliers (non-conforming grids) and corresponding shape functions

In order to connect non-conforming grids, we need to define multiple constraints between interface DOF from either side. These constraints can be obtained using Eq. (2.73) and (2.74). Since the component sub-domains are discretized using 4-node (bilinear) quadrilateral elements, we assume equivalent shape functions over the interface of Lagrange Multipliers. Also, since these constraints are applied over interface DOF only; multi-constraint operator is expressed as:

$$L^1_{ka} = P^1_{km} B^1_{ma} \tag{3.14}$$

$$L^2_{kb} = P^2_{kn} B^2_{nb} \tag{3.15}$$

As an example for implementing and constructing multiple constraints, consider the following illustration, Figure 3-7. Assume element edge from Ω^1 and Ω^2 is 1 and 0.5 unit length respectively and shape functions used across either interfaces are bilinear functions.

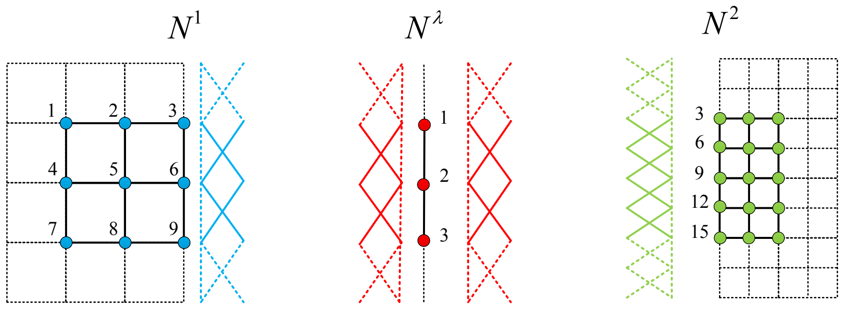


Figure 3-7: Reference sub-domains, interface of Lagrange Multipliers and corresponding shape functions

$$\therefore P^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 3 & 6 & 9 \end{array} \\ \begin{bmatrix} \int N^\lambda_1 N^1_3 & \int N^\lambda_1 N^1_6 & \int N^\lambda_1 N^1_9 \\ \int N^\lambda_2 N^1_3 & \int N^\lambda_2 N^1_6 & \int N^\lambda_2 N^1_9 \\ \int N^\lambda_3 N^1_3 & \int N^\lambda_3 N^1_6 & \int N^\lambda_3 N^1_9 \end{bmatrix} \end{array} = \begin{bmatrix} 0.3333 & 0.1666 & 0 \\ 0.1666 & 0.6666 & 0.1666 \\ 0 & 0.1666 & 0.3333 \end{bmatrix} \tag{3.16}$$

$$P^2 = \begin{array}{c|ccccc} & 3 & 6 & 9 & 12 & 15 \\ \hline 1 & \int N_1^\lambda N_3^2 & \cdots & \cdots & \cdots & \int N_1^\lambda N_{15}^2 \\ 2 & \int N_2^\lambda N_3^2 & \cdots & \cdots & \cdots & \int N_2^\lambda N_{15}^2 \\ 3 & \int N_3^\lambda N_3^2 & \cdots & \cdots & \cdots & \int N_3^\lambda N_{15}^2 \end{array} \tag{3.17}$$

$$= \begin{bmatrix} 0.2083 & 0.25 & 0.0416 & 0 & 0 \\ 0.0416 & 0.25 & 0.4167 & 0.25 & 0.0416 \\ 0 & 0 & 0.0416 & 0.25 & 0.2083 \end{bmatrix}$$

Resulting constraint equations are expressed as:

$$\Rightarrow \quad 0.3333x_3^1 + 0.1666x_6^1 = 0.2083x_3^2 + 0.25x_6^2 + 0.0416x_9^2$$

$$\Rightarrow \quad 0.1666x_3^1 + 0.6666x_6^1 + 0.1666x_9^1 = 0.0416x_3^2 + 0.25x_6^2 + 0.4167x_9^2 + 0.25x_{12}^2 + 0.0416x_{15}^2 \tag{3.18}$$

$$\Rightarrow \quad 0.1666x_6^1 + 0.3333x_9^1 = 0.0416x_9^2 + 0.25x_{12}^2 + 0.20835x_{15}^2$$

Hence, multiple grid coupling between conforming and non-conforming sub-domains can be achieved using Eq. (3.12) – (3.13) and Eq. (3.14) – (3.15) respectively. Note: the multi-constraint operator for coupled sub-domains, as described by Eq. (3.7) and (3.11), is expressed as the sum of component sub-domain contributions. Accordingly, the interface connectivity matrix (L) should be assigned a positive bias $(+L)$ for non-mortar DOF and a negative bias $(-L)$ for the mortar DOF.

In the case of non-conforming sub-domains, the numerical integration in Eq. (2.73) and (2.74), or equivalently Eq. (3.16) and (3.17), is typically performed over each element edge using the Quadrature rule (Zienkiewicz et al. 2005). MGMT Method however, uses

Trapezoidal rule (Lacour & Maday 1997) for the numerical integration of interface constraints due to its ease of implementation. Under this approach, the interface of Lagrange Multipliers (non-mortar interface) is divided into N equally spaced panels, or $N+1$ grid points such that $a = x_1 < x_2 < \ldots < x_{N+1} = b$ and the grid spacing $\Delta x = (b-a)/N$.

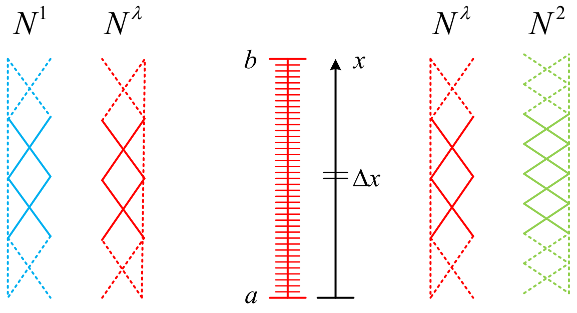


Figure 3-8: Interface integration grid

The product of shape functions, and their numerical integration, across the length of the interface of Lagrange Multipliers is then obtained as follows:

$$\begin{aligned}\int_{\Gamma_\lambda^{1,2}} N_k^\lambda N_m^1 \, d\Gamma &= \int_a^b N_k^\lambda(x)\, N_m^1(x)\, dx \\ &= \frac{\Delta x}{2} \sum_{i=1}^{N} \left[N_k^\lambda(x_{i+1}) N_m^1(x_{i+1}) + N_k^\lambda(x_i) N_m^1(x_i) \right]\end{aligned} \tag{3.19}$$

$$\begin{aligned}\int_{\Gamma_\lambda^{1,2}} N_k^\lambda N_n^2 \, d\Gamma &= \int_a^b N_k^\lambda(x)\, N_n^2(x)\, dx \\ &= \frac{\Delta x}{2} \sum_{i=1}^{N} \left[N_k^\lambda(x_{i+1}) N_n^2(x_{i+1}) + N_k^\lambda(x_i) N_n^2(x_i) \right]\end{aligned} \tag{3.20}$$

3.4 Multiple Time-Scale Coupling

3.4.1 Newmark Time Integration

In this section we will briefly discuss the Newmark time integration method (implicit and explicit) and its implementation algorithm as suited for MGMT coupling. We begin with fully discretized equation of motion, as derived using Generalized-α Method in Section 2.3.3, with $\alpha_m = \alpha_f = 0$.

$$M\ddot{U}_{n+1} + C\dot{U}_{n+1} + KU_{n+1} = F_{n+1} \tag{3.21}$$

Displacement and velocity updates using Newmark parameters β and γ are expressed as:

$$U_{n+1} = U_n + \Delta t\dot{U}_n + \Delta t^2\left(0.5-\beta\right)\ddot{U}_n + \Delta t^2\beta\ddot{U}_{n+1} \tag{3.22}$$

$$\dot{U}_{n+1} = \dot{U}_n + \Delta t\left(1-\gamma\right)\ddot{U}_n + \Delta t\gamma\ddot{U}_{n+1} \tag{3.23}$$

Substituting these equations back into Eq. (3.21), the acceleration-form of Newmark Method is obtained as:

$$\mathbf{K}\ddot{U}_{n+1} = \mathbf{F_{n+1}} \tag{3.24}$$

Where effective stiffness matrix $\mathbf{K}$ and load vector $\mathbf{F_{n+1}}$ are expressed as:

$$\mathbf{K} = \left\{M + \Delta t\gamma C + \Delta t^2\beta K\right\} \tag{3.25}$$

$$\mathbf{F_{n+1}} = F_{n+1} - C\left\{\dot{U}_n + \Delta t(1-\gamma)\ddot{U}_n\right\} - K\left\{U_n + \Delta t\dot{U}_n + \Delta t^2(0.5-\beta)\ddot{U}_n\right\} \tag{3.26}$$

Equation (3.24) can now be solved for unknown accelerations by factorizing $\mathbf{K}$ using Cholesky decomposition as $\mathbf{K} = LDL^T$. Displacements and velocities can then be obtained from Eq. (3.22) and (3.23) respectively.

The Newmark algorithm (for an undamped case) is stable if $\gamma \geq 1/2$ and is unconditionally stable if $\beta \geq (\gamma + 1/2)^2/4$. With appropriate expression for β and γ, the Newmark Method can be further classified into (Cook et al. 2001):

1) Implicit constant average acceleration method, also known as Trapezoidal rule

$$\gamma = \frac{1}{2} \quad and \quad \beta = \frac{1}{4}$$

2) Implicit linear acceleration method (corresponds to θ = 1 in the Wilson Method)

$$\gamma = \frac{1}{2} \quad and \quad \beta = \frac{1}{6}$$

3) Explicit Central Difference Method (CDM), also known as Velocity Verlet Method

$$\gamma = \frac{1}{2} \quad and \quad \beta = 0$$

In order to facilitate the derivation of MGMT equations, it is helpful to visualize the Newmark Method, Eq. (3.21), (3.22) and (3.23), in a matrix form as follows:

$$\begin{bmatrix} M & C & K \\ -\Delta t\gamma & I & 0 \\ -\Delta t^2\beta & 0 & I \end{bmatrix} \begin{bmatrix} \ddot{U}_{n+1} \\ \dot{U}_{n+1} \\ U_{n+1} \end{bmatrix} = \begin{bmatrix} F_{n+1} \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 \\ -\Delta t(1-\gamma)I & -I & 0 \\ -\Delta t^2(0.5-\beta) & -\Delta tI & -I \end{bmatrix} \begin{bmatrix} \ddot{U}_n \\ \dot{U}_n \\ U_n \end{bmatrix} \tag{3.27}$$

Equation (3.27) can be further expressed in compact form as follows:

$$\tilde{M}\tilde{U}_{n+1} = \tilde{F}_{n+1} - \tilde{N}\tilde{U}_n \tag{3.28}$$

If the time interval of interest is divided into N time-steps of size Δt, with Eq. (3.28) enforced at discrete instants of time t_n where $n \in \{0,1,2,...N-1\}$, entire time evolution can be expressed as:

$$\begin{bmatrix} \tilde{M} & 0 & 0 & 0 \\ \tilde{N} & \tilde{M} & 0 & 0 \\ 0 & \ddots & \ddots & 0 \\ 0 & 0 & \tilde{N} & \tilde{M} \end{bmatrix} \begin{bmatrix} \tilde{U}_1 \\ \tilde{U}_2 \\ \vdots \\ \tilde{U}_{N-1} \end{bmatrix} = \begin{bmatrix} \tilde{F}_1 - \tilde{N}\tilde{U}_0 \\ \tilde{F}_2 \\ \vdots \\ \tilde{F}_{N-1} \end{bmatrix} \tag{3.29}$$

A pseudo-code to advance the solution of an input system of equations from time-step n to $(n+1)$ using the Newmark Method can now be expressed as:

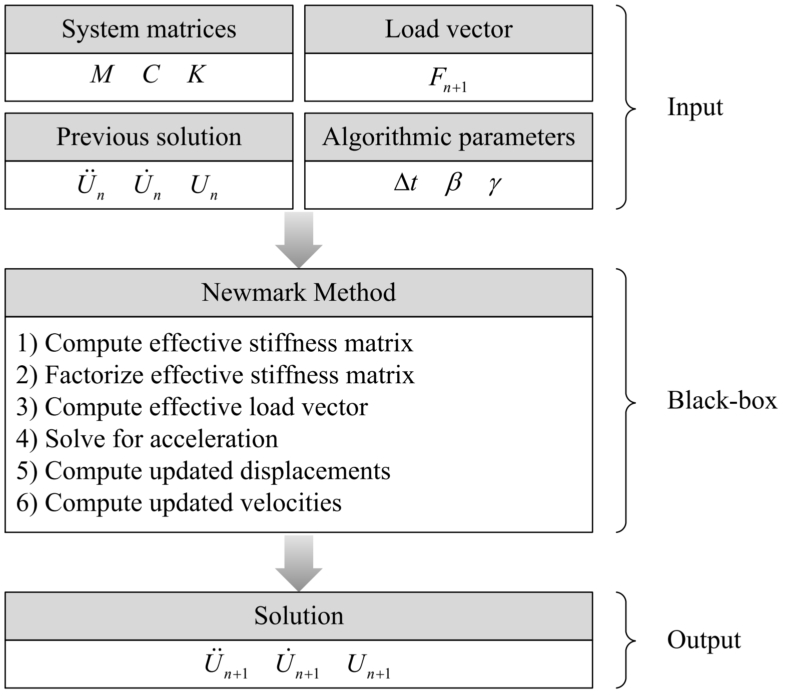

System matrices
$M \quad C \quad K$
Load vector
F_{n+1}
Previous solution
$\ddot{U}_n \quad \dot{U}_n \quad U_n$
Algorithmic parameters
$\Delta t \quad \beta \quad \gamma$
Input
Newmark Method
1) Compute effective stiffness matrix
2) Factorize effective stiffness matrix
3) Compute effective load vector
4) Solve for acceleration
5) Compute updated displacements
6) Compute updated velocities
Black-box
Solution
$\ddot{U}_{n+1} \quad \dot{U}_{n+1} \quad U_{n+1}$
Output

Figure 3-9: Pseudo-code for direct time integration using Newmark Method acceleration form

3.4.2 Interface Condensation

In this section we will derive the fully discretized equations of motion for decomposed sub-domains, coupled over multiple time scales. We shall use the DD from Figure 3-2 (b). It is assumed that every component sub-domain is discretized (spatially and temporally) independent of each other and multiple grid coupling between adjacent sub-domains is established using M-FEM.

Let the global time-step for evolving MGMT equations from t_n to t_{n+1}, where $n \in \{0,1,2,...N-1\}$, be ΔT. For reference purposes, we also assume that Ω^1 is integrated at the global time-step ΔT. Every other component sub-domain (Ω^i) can be integrated with time-steps Δt^i such that $(T+\Delta T) = (T+\xi^i \Delta t^i)$. In order to obtain global solution at synchronous time instants it is necessary to ensure that the time-step ratio ξ^i is an integer factor of the global time-step ΔT. Let us also introduce an intermediate time-step counter η^i such that $\eta^i \in \{0,1,2,\ldots \xi^i\}$. See Table 3-2 and Figure 3-10 to get a clear understanding of these time-stepping parameters.

Table 3-2: Sub-domain time-stepping parameters

Sub-domain	Time-step (Δt)	Algorithmic parameters	Time-step ratio ($\xi = \Delta T / \Delta t$)	Intermediate step counter (η)
Ω^1	$\Delta t^1 = \Delta T$	β^1, γ^1	$\xi^1 = 1$	$\eta^1 = 0, 1$
Ω^2	Δt^2	β^2, γ^2	ξ^2	$\eta^2 = 0, 1, 2 \ldots \xi^2$
:	:	:	:	:
Ω^S	Δt^S	β^S, γ^S	ξ^S	$\eta^S = 0, 1, 2, 3 \ldots \xi^S$

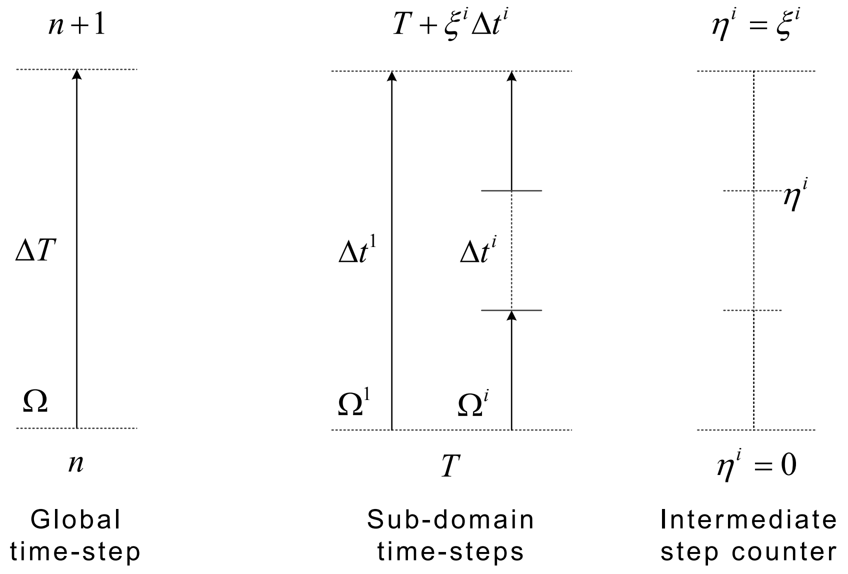


Figure 3-10: MGMT time-stepping

Using Newmark Method, fully discretized equations of motion for component sub-domains along with the interface condition enforced at $\eta^i=\xi^i$, i.e. global time-steps, are expressed as:

$$M^i\ddot{U}^i_{\eta^i}+C^i\dot{U}^i_{\eta^i}+K^iU^i_{\eta^i}+L^{i^T}\lambda_{\eta^i}=F^i_{\eta^i}\quad (i=1,2\ldots S) \tag{3.30}$$

$$\sum_{i=1}^{S}L^i\dot{U}^i_{\xi^i}=0 \tag{3.31}$$

In Eq. (3.30) unknown interface reactions (Lagrange Multipliers) are computed at every intermediate time-step, for every component sub-domain. This is computationally very expensive, especially in cases where the time-step ratio between adjoining sub-domains is large. Intermediate interface reactions (at $0<\eta^i<\xi^i$) can however be condensed, so that

Eq. (3.30) can be expressed in terms of Lagrange Multipliers at global time-steps, that is in terms of $\lambda_n\ (\eta^i = 0)$ and $\lambda_{n+1}\ (\eta^i = \xi^i)$.

Interface condensation is performed using an energy preserving approach that is similar to PH Method. This approach makes the interface computation ξ^i times faster than GC Method, making it computationally superior. However, the drawback in using PH Method is that it allows coupling only two sub-domains at a time. Hence for multiple sub-domain coupling, one needs to implement a recursive solution algorithm (Prakash 2007) that can solve component sub-domains – one pair at a time. This again restricts the computational efficiency of multiple time-scale coupling. Also, in PH Method, the Lagrange Multipliers are expressed in terms of 'unbalanced' interface reactions, which are obtained from the coarse time-step sub-domain. This makes the fine time-step sub-domain dependent on other sub-domains, making it impossible to solve multiple sub-domains concurrently using parallel processing. MGMT Method, however, uses the energy preserving approach with a slight modification, such that the intermediate interface reactions are expressed only in the terms of known (λ_n) and unknown (λ_{n+1}) Lagrange Multipliers. We will also show that our implementation results in exclusively independent sub-domains, so they may be solved concurrently independent of each other, further improving computational efficiency.

We begin with kinematic decomposition, Eq. (3.32), which will enable us to define the equation of motion independently under the action of external forces and under the response of interface reactions. Accordingly, let:

$$\tilde{U} = \tilde{V} + \tilde{W} \tag{3.32}$$

Where the notation (~) was established earlier in Eq. (3.27) and (3.28). Using Eq. (3.32), equilibrium of Ω^1 can be expressed as:

$$\left(M^1\ddot{V}_{n+1}^1 + C^1\dot{V}_{n+1}^1 + K^1V_{n+1}^1\right) + \left(M^1\ddot{W}_{n+1}^1 + C^1\dot{W}_{n+1}^1 + K^1W_{n+1}^1\right) = F_{n+1}^1 - L^{1^T}\lambda_{n+1} \tag{3.33}$$

<u>Note:</u> for Ω^1, $\xi^1 = 1$ which corresponds to $\eta^1 = 1$ and hence the time instant $(n+1)$. Equation (3.33) can now be decomposed into two equations as follows:

$$M^1\ddot{V}_{n+1}^1 + C^1\dot{V}_{n+1}^1 + K^1V_{n+1}^1 = F_{n+1}^1 \tag{3.34}$$

$$M^1\ddot{W}_{n+1}^1 + C^1\dot{W}_{n+1}^1 + K^1W_{n+1}^1 = -L^{1^T}\lambda_{n+1} \tag{3.35}$$

Equation (3.34) represents the equilibrium of Ω^1 and its contribution to the kinematic quantities under the action of external forces only. Equation (3.35) on other hand represents the equilibrium under the action of Lagrange Multipliers or unknown interface reactions. In order to avoid interface dissipation across intermediate time-steps, the equilibrium of Ω^1 is also enforced at every intermediate time-step η^i $(i = 2,3\ldots S)$ instant by requiring that the combined residual of Eq. (3.34) and (3.35), represented by δf and δl respectively, is equal to zero. That is:

$$\delta f_{\eta^i}^1 + \delta l_{\eta^i}^1 = 0 \quad \left(i = 1,2\ldots S\right) \tag{3.36}$$

For $i = 1$ or $\eta = 1$, Eq. (3.36) is identical to solving the equilibrium of Ω^1 as defined by Eq. (3.30). However, for $1 < i < S$ and $\eta > 1$, it is equivalent to enforcing the equilibrium of Ω^1 across intermediate time-steps.

The first term in Eq. (3.36) represents the free residuals obtained under the action of external forces and the second term represents the link residuals obtained under the action of unknown interface reactions:

$$\therefore \delta f^1_{\eta^i} = F^1_{\eta^i} - M^1\ddot{V}^1_{\eta^i} - C^1\dot{V}^1_{\eta^i} - K^1V^1_{\eta^i} \quad (i = 2\ldots S) \tag{3.37}$$

$$\therefore \delta l^1_{\eta^i} = -L^{1^T}\lambda_{\eta^i} - M^1\ddot{W}^1_{\eta^i} - C^1\dot{W}^1_{\eta^i} - K^1W^1_{\eta^i} \quad (i = 2\ldots S) \tag{3.38}$$

Before enforcing Eq. (3.36) let us define necessary linear interpolation functions for obtaining intermediate quantities used in Eq. (3.37) and (3.38):

$$\tilde{V}^1_{\eta^i} = \left(1 - \frac{\eta^i}{\xi^i}\right)\tilde{U}^1_n + \left(\frac{\eta^i}{\xi^i}\right)\tilde{V}^1_{n+1} \tag{3.39}$$

$$\tilde{W}^1_{\eta^i} = \left(\frac{\eta^i}{\xi^i}\right)\tilde{W}^1_{n+1} \tag{3.40}$$

$$F^1_{\eta^i} = \left(1 - \frac{\eta^i}{\xi^i}\right)F^1_n + \left(\frac{\eta^i}{\xi^i}\right)F^1_{n+1} \tag{3.41}$$

Equation (3.34) can be solved independently using the Newmark Method, Figure 3-9, yielding the solution vector at time instant $(n+1)$ or $(\eta^1 = \xi^1 = 1)$. Equation (3.37) can now be expressed as:

$$\delta f^1_{\eta^i} = \left\{\left(1-\frac{\eta^i}{\xi^i}\right)F^1_n + \left(\frac{\eta^i}{\xi^i}\right)F^1_{n+1}\right\} - M^1\left\{\left(1-\frac{\eta^i}{\xi^i}\right)\ddot{U}^1_n + \left(\frac{\eta^i}{\xi^i}\right)\ddot{V}^1_{n+1}\right\}$$
$$-C^1\left\{\left(1-\frac{\eta^i}{\xi^i}\right)\dot{U}^1_n + \left(\frac{\eta^i}{\xi^i}\right)\dot{V}^1_{n+1}\right\} - K^1\left\{\left(1-\frac{\eta^i}{\xi^i}\right)U^1_n + \left(\frac{\eta^i}{\xi^i}\right)V^1_{n+1}\right\} \tag{3.42}$$

$$\therefore \delta f^1_{\eta^i} = \left(1-\frac{\eta^i}{\xi^i}\right)\left\{F^1_n - M^1\ddot{U}^1_n - C^1\dot{U}^1_n - K^1U^1_n\right\}$$
$$+ \left(\frac{\eta^i}{\xi^i}\right)\left\{F^1_{n+1} - M^1\ddot{V}^1_{n+1} - C^1\dot{V}^1_{n+1} - K^1V^1_{n+1}\right\} \tag{3.43}$$

Using Eq. (3.30) and (3.34) we have:

$$\delta f^1_{\eta^i} = \left(1-\frac{\eta^i}{\xi^i}\right)L^{1^T}\lambda_n \tag{3.44}$$

Using Eq. (3.40), link residuals from Eq. (3.38) can be expressed as:

$$\delta l^1_{\eta^i} = -L^{1^T}\lambda_{\eta^i} - M^1\left(\frac{\eta^i}{\xi^i}\right)\ddot{W}^1_{n+1} - C^1\left(\frac{\eta^i}{\xi^i}\right)\dot{W}^1_{n+1} - K^1\left(\frac{\eta^i}{\xi^i}\right)W^1_{n+1} \tag{3.45}$$

$$\therefore \delta l^1_{\eta^i} = -L^{1^T}\lambda_{\eta^i} - \left(\frac{\eta^i}{\xi^i}\right)\left\{M^1\ddot{W}^1_{n+1} + C^1\dot{W}^1_{n+1} + K^1W^1_{n+1}\right\} \tag{3.46}$$

Further using Eq. (3.35):

$$\delta l^1_{\eta^i} = -L^{1^T}\lambda_{\eta^i} + \left(\frac{\eta^i}{\xi^i}\right)L^{1^T}\lambda_{n+1} \tag{3.47}$$

Enforcing Eq. (3.36) we get:

$$\left(1-\frac{\eta^i}{\xi^i}\right)L^{1^T}\lambda_n - L^{1^T}\lambda_{\eta^i} + \left(\frac{\eta^i}{\xi^i}\right)L^{1^T}\lambda_{n+1} = 0$$

$$\therefore L^{1^T}\lambda_{\eta^i} = \left(1-\frac{\eta^i}{\xi^i}\right)L^{1^T}\lambda_n + \left(\frac{\eta^i}{\xi^i}\right)L^{1^T}\lambda_{n+1}$$

$$\therefore \lambda_{\eta^i} = \left(1-\frac{\eta^i}{\xi^i}\right)\lambda_n + \left(\frac{\eta^i}{\xi^i}\right)\lambda_{n+1} \tag{3.48}$$

Equation (3.48) represents the interpolation function to compute intermediate Lagrange Multipliers (interface reactions) at every intermediate time-step η^i for Ω^i $(i = 2,3\ldots S)$. We can now condense the intermediate Lagrange Multipliers and express the equilibrium of Ω^i as follows:

$$M^i\ddot{U}^i_{\eta^i} + C^i\dot{U}^i_{\eta^i} + K^iU^i_{\eta^i} + \left(\frac{\eta^i}{\xi^i}\right)L^{i^T}\lambda_{n+1} = F^i_{\eta^i} - \left(1-\frac{\eta^i}{\xi^i}\right)L^{i^T}\lambda_n \tag{3.49}$$

Comparing Eq. (3.49) with Eq. (2.105) derived in Section 2.5.5 for PH Method, we clearly see that this approach does not impose any sub-domain dependency. Intermediate interface reactions are now defined only in terms of unknown Lagrange Multipliers at $(n+1)$ and known Lagrange Multipliers at n, as shown in Figure 3-11. This further enables us to solve every component sub-domain independent of each other followed by a direct solution of λ_{n+1}.

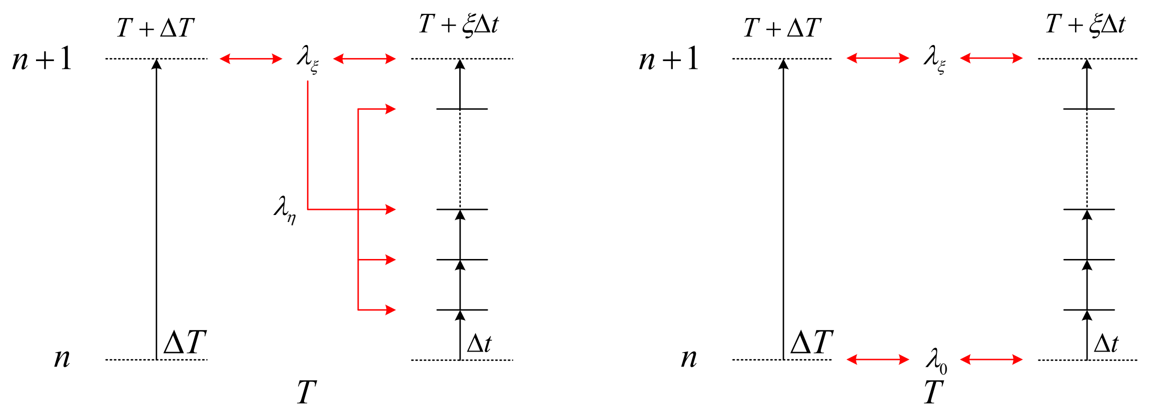


Figure 3-11: Comparison between (a) PH Method and (b) MGMT Method

Using compact notations discussed earlier in Eq. (3.27) and (3.28), global system of fully discretized equations along with displacement and velocity updates can now be expressed as:

$$\tilde{M}^i\tilde{U}^i_{\eta^i} + \left(\frac{\eta^i}{\xi^i}\right)\tilde{L}^i\lambda_{n+1} = \tilde{F}^i_{\eta^i} - \tilde{N}^i\tilde{U}^i_{\eta^i-1} - \left(1-\frac{\eta^i}{\xi^i}\right)L^{i^T}\lambda_n \quad (i=1,2\ldots S) \tag{3.50}$$

$$\sum_{i=1}^{S}\tilde{B}^i\tilde{U}^i_{n+1} = 0 \tag{3.51}$$

$$\text{Where: } \tilde{B}^i = \begin{bmatrix} 0 & L^i & 0 \end{bmatrix} \tag{3.52}$$

3.5 Stability Analysis Using Energy Method

When deriving coupled MGMT equations for component sub-domains, we assumed that the solution is stable if energy is conserved within respective sub-domains (local equilibrium is satisfied) and interface energy produced as a result of introducing Lagrange Multipliers is identically zero. This allowed us to define an appropriate variable for enforcing continuity across dividing interfaces.

In this section we will show, using Energy Method (Richtmyer & Morton 1967; Hughes 2012), that the change in energy due to MGMT coupling over time-step $(t_n \to t_{n+1})$ or $(T \to T + \Delta T)$ is identically zero if the time integration scheme is stable within respective sub-domains. We will also show that enforcing continuity of velocities conserves global energy by yielding zero interface energy contributions.

Some useful definitions and identities used in this Section are, (Hughes 2012):

$$\textit{Undivided forward difference operator} \to [x_n] = x_{n+1} - x_n \tag{3.53}$$

$$\textit{mean value operator} \to \langle x_n \rangle = \frac{x_{n+1} + x_n}{2} \tag{3.54}$$

$$\langle x_n \rangle^T A[x_n] = \frac{1}{2}\left[x_n^T A x_n\right] \tag{3.55}$$

$$x_{n+\alpha}^T\, A\left[x_n\right] = \langle x_n \rangle^T A\left[x_n\right] + \left(\alpha - \frac{1}{2}\right)\left[x_n\right]^T A\left[x_n\right] \tag{3.56}$$

Before obtaining stability conditions for coupled sub-domains, we shall derive the necessary equations for a single sub-domain. For the purpose of stability, it suffices to restrict our discussion to homogeneous cases where the external applied forces on a system are equal to zero. Accordingly, the governing equation, discretized in time using Newmark Method, along with displacement and velocity updates is expressed as:

$$M\ddot{U}_{n+1} + C\dot{U}_{n+1} + KU_{n+1} = -L^T \lambda_{n+1} \tag{3.57}$$

$$\dot{U}_{n+1} = \dot{U}_n + \Delta t\left(1-\gamma\right)\ddot{U}_n + \Delta t\gamma\ddot{U}_{n+1} \tag{3.58}$$

$$U_{n+1} = U_n + \Delta t\dot{U}_n + \Delta t^2\left(0.5-\beta\right)\ddot{U}_n + \Delta t^2\beta\ddot{U}_{n+1} \tag{3.59}$$

Let domain energies be represented by:

$$\textit{Kinetic Energy} \rightarrow \hat{T}(x) = \frac{1}{2}x^T M\, x \tag{3.60}$$

$$\textit{Stiffness Energy} \rightarrow \hat{V}(x) = \frac{1}{2}x^T K\, x \tag{3.61}$$

$$\textit{Dissipation Energy} \rightarrow \hat{D}(x) = -x^T C\, x \tag{3.62}$$

Equations (3.58) and (3.59), in terms of discrete operators can be expressed as:

$$\left[\dot{U}_n\right] = \Delta t\, \ddot{U}_{n+\gamma} \tag{3.63}$$

$$\left[U_n\right] = \Delta t\left\langle \dot{U}_n \right\rangle + \frac{\Delta t^2}{2}\left(2\beta - \gamma\right)\left[\ddot{U}_n\right] \tag{3.64}$$

Using Eq. (3.63), Dissipation energy can be expressed as:

$$-\hat{D}\left(\left[\dot{U}_n\right]\right) = \Delta t^2 \ddot{U}_{n+\gamma}^T C \ddot{U}_{n+\gamma} \tag{3.65}$$

Further using Eq. (3.56), Eq. (3.65) can be expressed as:

$$-\hat{D}\left(\left[\dot{U}_n\right]\right) = \Delta t^2 \left\{ \left\langle \ddot{U}_n \right\rangle^T C \left\langle \ddot{U}_n \right\rangle + 2\left(\gamma - \frac{1}{2}\right)\left\langle \ddot{U}_n \right\rangle^T C\left[\ddot{U}_n\right] + \left(\gamma - \frac{1}{2}\right)^2 \left[\ddot{U}_n\right]^T C\left[\ddot{U}_n\right] \right\} \tag{3.66}$$

$$\therefore -\hat{D}\left(\left[\dot{U}_n\right]\right) = -\Delta t^2 \left\{ \hat{D}\left(\left\langle \ddot{U}_n \right\rangle\right) + \left(\gamma - \frac{1}{2}\right)\left[\hat{D}\left(\ddot{U}_n\right)\right] + \left(\gamma - \frac{1}{2}\right)^2 \hat{D}\left(\left[\ddot{U}_n\right]\right) \right\} \tag{3.67}$$

The energy balance equation from time-step $n \to n+1$ is expressed by applying the forward difference operator $[\cdot]$ to the fully discretized equation of motion and then pre-multiplying by $[\dot{U}_n]^T$:

$$\left[\dot{U}_n\right]^T M\left[\ddot{U}_n\right] + \left[\dot{U}_n\right]^T C\left[\dot{U}_n\right] + \left[\dot{U}_n\right]^T K\left[U_n\right] + \left[\dot{U}_n\right]^T L^T\left[\lambda_n\right] = 0 \tag{3.68}$$

Substituting Eq. (3.63) and (3.64):

$$\begin{aligned}&\Delta t\,\ddot{U}^T_{n+\gamma} M\left[\ddot{U}_n\right]+\left[\dot{U}_n\right]^T C\left[\dot{U}_n\right]+\left\langle \dot{U}^T_n\right\rangle \Delta t K\left[\dot{U}_n\right]\\&+\ddot{U}^T_{n+\gamma}\ \Delta t K\frac{\Delta t^2}{2}\left(2\beta-\gamma\right)\left[\ddot{U}_n\right]+\left[\dot{U}_n\right]^T L^T\left[\lambda_n\right]=0\end{aligned} \tag{3.69}$$

$$\begin{aligned}&\therefore \Delta t\left\{\left[\hat{T}\left(\ddot{U}_n\right)\right]+2\left(\gamma-\frac{1}{2}\right)\hat{T}\left(\left[\ddot{U}_n\right]\right)\right\}-\hat{D}\left(\left[\dot{U}_n\right]\right)+\Delta t\left[\hat{V}\left(\dot{U}_n\right)\right]\\&+\frac{\Delta t^3}{2}\left(2\beta-\gamma\right)\left\{\left[\hat{V}\left(\ddot{U}_n\right)\right]+2\left(\gamma-\frac{1}{2}\right)\hat{V}\left(\left[\ddot{U}_n\right]\right)\right\}+\left[\dot{U}_n\right]^T L^T\left[\lambda_n\right]=0\end{aligned} \tag{3.70}$$

Substituting Eq. (3.65) and dividing both sides by Δt:

$$\begin{aligned}&\left[\hat{T}\left(\ddot{U}_n\right)\right]+\frac{\Delta t^2}{2}\left(2\beta-\gamma\right)\left[\hat{V}\left(\ddot{U}_n\right)\right]-\Delta t\left(\gamma-\frac{1}{2}\right)\left[\hat{D}\left(\ddot{U}_n\right)\right]+\left[\hat{V}\left(\dot{U}_n\right)\right]=\\&-2\left(\gamma-\frac{1}{2}\right)\hat{T}\left(\left[\ddot{U}_n\right]\right)-\frac{\Delta t^2}{2}\left(2\beta-\gamma\right)2\left(\gamma-\frac{1}{2}\right)\hat{V}\left(\left[\ddot{U}_n\right]\right)+\\&\Delta t\left(\gamma-\frac{1}{2}\right)^2\hat{D}\left(\left[\ddot{U}_n\right]\right)+\Delta t\hat{D}\left(\left\langle\ddot{U}_n\right\rangle\right)-\frac{1}{\Delta t}\left[\dot{U}_n\right]^T L^T\left[\lambda_n\right]\end{aligned} \tag{3.71}$$

Using the notations described in Eq. (3.60), (3.61) and (3.62), we can now express the above equations as:

$$\begin{aligned}&\left\langle\ddot{U}_n\right\rangle^T\left\{M+\frac{\Delta t^2}{2}\left(2\beta-\gamma\right)K+2\Delta t\left(\gamma-\frac{1}{2}\right)C\right\}\left[\ddot{U}_n\right]+\left[\hat{V}\left(\dot{U}_n\right)\right]=\\&-\left(\gamma-\frac{1}{2}\right)\left[\ddot{U}_n\right]^T\left\{M+\frac{\Delta t^2}{2}\left(2\beta-\gamma\right)K+2\Delta t\left(\gamma-\frac{1}{2}\right)C\right\}\left[\ddot{U}_n\right]\\&-\Delta t\left\langle\ddot{U}_n\right\rangle^T C\left\langle\ddot{U}_n\right\rangle-\frac{1}{\Delta t}\left[\dot{U}_n\right]^T L^T\left[\lambda_n\right]\end{aligned} \tag{3.72}$$

Introducing (Hughes 2012):

$$
\begin{aligned}
B &= M + \Delta t\left(\gamma - \frac{1}{2}\right)C + \Delta t^2\left(\beta - \frac{\gamma}{2}\right)K \\
A &= B + \Delta t\left(\gamma - \frac{1}{2}\right)C
\end{aligned}
\tag{3.73}
$$

Equation (3.72) can be expressed as:

$$
\begin{aligned}
&\frac{1}{2}\ddot{U}_{n+1}^T A \ddot{U}_{n+1} - \frac{1}{2}\ddot{U}_n^T A \ddot{U}_n + \frac{1}{2}\dot{U}_{n+1}^T K \dot{U}_{n+1} - \frac{1}{2}\dot{U}_n^T K \dot{U}_n = \\
&-\left(\gamma - \frac{1}{2}\right)\left[\ddot{U}_n\right]^T B\left[\ddot{U}_n\right] - \Delta t\left\langle \ddot{U}_n\right\rangle^T C\left\langle \ddot{U}_n\right\rangle \underline{\underline{- \frac{1}{\Delta t}\left[\dot{U}_n\right]^T L^T\left[\lambda_n\right]}}
\end{aligned}
\tag{3.74}
$$

Equation (3.74) is similar to the energy balance equation of a continuous domain, except for the interface energy term. The system of MGMT equations can now be considered numerically stable if the total energy change (the RHS of Eq. (3.74)) under no external loads is less than or equal to zero. Hence, if we can show that the interface contributions are less than or equal to zero, the burden of stability relies solely upon the stability of component sub-domains, as integrated using Newmark Method.

We shall now assume DD of a continuous region Ω into S component sub-domains, Figure 3-2. (b), with sub-domain time-stepping parameters as described in Figure 3-10 and Table 3-2. Using Eq. (3.74), interface contributions from Ω^i can be expressed as:

$$
\hat{E}_{\Gamma_\lambda^i} = \sum_{\eta^i=1}^{\xi^i} \frac{1}{\Delta t^i}\left[\dot{U}_{\eta^i-1}^i\right]^T L^{i^T}\left[\lambda_{\eta^i-1}\right]
\tag{3.75}
$$

Augmented interface contributions from all component sub-domains can then be expressed as:

$$
\begin{aligned}
\hat{E}_{\Gamma_\lambda} = \sum_{i=1}^{S}\sum_{\eta^i=1}^{\xi^i} \hat{E}_{\Gamma_\lambda^i} &= \sum_{i=1}^{S}\sum_{\eta^i=1}^{\xi^i} \frac{1}{\Delta t^i}\left[\dot{U}^i_{\eta^i-1}\right]^T L^{i^T}\left[\lambda_{\eta^i-1}\right] \\
&= \sum_{i=1}^{S}\sum_{\eta^i=1}^{\xi^i} \frac{1}{\Delta t^i}\left(\dot{U}^i_{\eta^i} - \dot{U}^i_{\eta^i-1}\right)^T L^{i^T}\left(\lambda_{\eta^i} - \lambda_{\eta^i-1}\right)
\end{aligned}
\tag{3.76}
$$

Using the relationship derived in Eq. (3.48) the forward difference of interface reactions in terms of λ_n and λ_{n+1} is expressed as:

$$
\begin{aligned}
\left[\lambda_{\eta-1}\right] &= \lambda_\eta - \lambda_{\eta-1} \\
&= \left\{\left(1-\frac{\eta}{\xi}\right)\lambda_n + \left(\frac{\eta}{\xi}\right)\lambda_{n+1}\right\} - \left\{\left(1-\frac{\eta-1}{\xi}\right)\lambda_n + \left(\frac{\eta-1}{\xi}\right)\lambda_{n+1}\right\} \\
&= \frac{1}{\xi}\left(\lambda_{n+1} - \lambda_n\right)
\end{aligned}
\tag{3.77}
$$

Equation (3.76) now becomes:

$$
\begin{aligned}
\hat{E}_{\Gamma_\lambda} &= \sum_{i=1}^{S} \frac{1}{\xi^i \Delta t^i}\left\{\sum_{\eta^i=1}^{\xi^i}\left(\dot{U}^i_{\eta^i} - \dot{U}^i_{\eta^i-1}\right)^T L^{i^T}\left(\lambda_{n+1} - \lambda_n\right)\right\} \\
&= \sum_{i=1}^{S}\left(\lambda_{n+1} - \lambda_n\right)^T \left\{\frac{L^i}{\xi^i \Delta t^i}\sum_{\eta^i=1}^{\xi^i}\left(\dot{U}^i_{\eta^i} - \dot{U}^i_{\eta^i-1}\right)\right\}
\end{aligned}
\tag{3.78}
$$

<u>Note:</u>

$$
\begin{aligned}
\sum_{\eta^i=1}^{\xi^i}\left(\dot{U}^i_{\eta^i} - \dot{U}^i_{\eta^i-1}\right) &= \left(\dot{U}^i_1 - \dot{U}^i_0\right) + \left(\dot{U}^i_2 - \dot{U}^i_1\right) + \ldots + \left(\dot{U}^i_{\xi^i} - \dot{U}^i_{\xi^i-1}\right) \\
&= \dot{U}^i_{\xi^i} - \dot{U}^i_0 \quad = \dot{U}^i_{n+1} - \dot{U}^i_n
\end{aligned}
\tag{3.79}
$$

And:

$$\xi^i \Delta t^i = \Delta T \tag{3.80}$$

Therefore we have:

$$\begin{aligned}
\hat{E}_{\Gamma_\lambda} &= \sum_{i=1}^{S} \left(\lambda_{n+1} - \lambda_n\right)^T \left\{ \frac{L^i}{\Delta T} \left(\dot{U}^i_{n+1} - \dot{U}^i_n \right) \right\} \\
&= \frac{\left(\lambda_{n+1} - \lambda_n\right)^T}{\Delta T} \sum_{i=1}^{S} L^i \left(\dot{U}^i_{n+1} - \dot{U}^i_n \right) \\
&= \frac{\left(\lambda_{n+1} - \lambda_n\right)^T}{\Delta T} \sum_{i=1}^{S} L^i \dot{U}^i_{n+1} - L^i \dot{U}^i_n
\end{aligned} \tag{3.81}$$

Since we have enforced continuity of velocities across sub-domain interfaces, Eq. (3.11), the above expression yields:

$$\hat{E}_{\Gamma_\lambda} = \frac{\left(\lambda_{n+1} - \lambda_n\right)^T}{\Delta T} \left\{ \cancelto{=0}{\sum_{i=1}^{S} L^i \dot{U}^i_{n+1}} - \cancelto{=0}{\sum_{i=1}^{S} L^i \dot{U}^i_n} \right\} \tag{3.82}$$

$$\therefore \hat{E}_{\Gamma_\lambda} = 0 \tag{3.83}$$

Equation (3.83) proves that introducing Lagrange Multipliers in conjugation with the continuity of velocity constraint, results in identically zero energy contributions. Accordingly, the stability of MGMT coupling only depends on the stability of Newmark Method in integrating component sub-domains. Hence, as long as the stability requirements are satisfied within the time integration of respective sub-domains, MGMT coupling is stable and energy preserving.

3.6 Solution Algorithm and its Finite Element Implementation

In this section we will derive the implementation algorithm and a step-by-step solution procedure for obtaining the solution of coupled MGMT equations as described by Eq. (3.50) and (3.51). Let us use the DD from Figure 3-2 (b) and the time-stepping parameters from Table 3-2, Figure 3-10. In order to communicate information across component sub-domains at global time-step ΔT or $(\eta^i = \xi^i)$, equilibrium of Ω^i where $\xi^i > 1$ needs to be advanced from $\eta^i = 1, 2 \ldots \xi^i$. These intermediate steps combined together in a matrix format can be expressed as:

$$\left[\begin{array}{cccc:c} \tilde{M}^i & 0 & 0 & 0 & \left(1/\xi^i\right)\tilde{L}^i \\ \tilde{N}^i & \tilde{M}^i & 0 & 0 & \left(2/\xi^i\right)\tilde{L}^i \\ 0 & \ddots & \ddots & 0 & \vdots \\ 0 & 0 & \tilde{N}^i & \tilde{M}^i & \tilde{L}^i \end{array}\right] \left[\begin{array}{c} \tilde{U}_1^i \\ \tilde{U}_2^i \\ \vdots \\ \tilde{U}_{\xi^i}^i \\ \hdashline \lambda_{n+1} \end{array}\right] = \left[\begin{array}{c} \tilde{F}_1^i - \tilde{N}^i\tilde{U}_n^i - \left(1 - \dfrac{1}{\xi^i}\right) L^{i^T} \lambda_n \\ \tilde{F}_2^i - \left(1 - \dfrac{2}{\xi^i}\right) L^{i^T} \lambda_n \\ \vdots \\ \tilde{F}_{\xi^i}^i \end{array}\right] \tag{3.84}$$

Introducing the notation $(\approx)$ Eq (3.84) can be expressed as:

$$\tilde{\tilde{M}}^i \tilde{\tilde{U}}_{\xi^i}^i + \tilde{\tilde{L}}^i \lambda_{n+1} = \tilde{\tilde{F}}_{\xi^i}^i \tag{3.85}$$

Where:

$$\tilde{\tilde{M}}^i = \begin{bmatrix} \tilde{M}^i & 0 & 0 & 0 \\ \tilde{N}^i & \tilde{M}^i & 0 & 0 \\ 0 & \ddots & \ddots & 0 \\ 0 & 0 & \tilde{N}^i & \tilde{M}^i \end{bmatrix} \tag{3.86}$$

$$\tilde{\tilde{U}}^{i}_{\xi^i} = \begin{bmatrix} \tilde{U}^{i}_{1} & \tilde{U}^{i}_{2} & \cdots & \tilde{U}^{i}_{\xi^i} \end{bmatrix}^{T} \tag{3.87}$$

$$\tilde{\tilde{L}}^{i} = \begin{bmatrix} \dfrac{\tilde{L}^{i}}{\xi^i} & \dfrac{2\tilde{L}^{i}}{\xi^i} & \cdots & \tilde{L}^{i} \end{bmatrix}^{T} \tag{3.88}$$

$$\tilde{\tilde{F}}^{i}_{\xi^i} = \begin{bmatrix} \tilde{F}^{i}_{1} - \tilde{N}^{i}\tilde{U}^{i}_{n} - \left(1 - \dfrac{1}{\xi^i}\right) L^{i^{T}} \lambda_n \\ \tilde{F}^{i}_{2} - \left(1 - \dfrac{2}{\xi^i}\right) L^{i^{T}} \lambda_n \\ \vdots \\ \tilde{F}^{i}_{\xi^i} \end{bmatrix} \tag{3.89}$$

For sub-domain Ω^1 where $\xi^1 = 1$ we have:

$$\tilde{M}^{1}\tilde{U}^{1}_{n+1} + \tilde{L}^{1}\lambda_{n+1} = \tilde{F}^{1}_{n+1} - \tilde{N}^{1}\tilde{U}^{1}_{n} \tag{3.90}$$

Final system of equations for coupled sub-domains (discretized independently in space and time) and synchronized at every global time-step ΔT or time-instant $(n+1)$ can now be expressed as:

$$\left[\begin{array}{cccc|c} \tilde{\tilde{M}}^{S} & \cdots & 0 & 0 & \tilde{\tilde{L}}^{S} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & \tilde{\tilde{M}}^{2} & 0 & \tilde{\tilde{L}}^{2} \\ 0 & \cdots & 0 & \tilde{M}^{1} & \tilde{L}^{1} \\ \hline \tilde{\tilde{B}}^{S} & \cdots & \tilde{\tilde{B}}^{2} & \tilde{B}^{1} & 0 \end{array}\right] \left[\begin{array}{c} \tilde{\tilde{U}}^{S}_{\xi^S} \\ \vdots \\ \tilde{\tilde{U}}^{2}_{\xi^2} \\ \tilde{U}^{1}_{n+1} \\ \hline \lambda_{n+1} \end{array}\right] = \left[\begin{array}{c} \tilde{\tilde{F}}^{S}_{\xi^S} \\ \vdots \\ \tilde{\tilde{F}}^{2}_{\xi^2} \\ \tilde{F}^{1}_{n+1} - \tilde{N}^{1}\tilde{U}^{1}_{n} \\ \hline 0 \end{array}\right] \tag{3.91}$$

With:

$$\tilde{\tilde{B}}^i = \begin{bmatrix} 0 & 0 & \cdots & \tilde{B}^i \end{bmatrix} \tag{3.92}$$

Above set of equations can be conveniently grouped block wise as follows:

$$\left[\begin{array}{c|c} A & b \\ \hline c & d \end{array}\right] \left[\begin{array}{c} x \\ \hline y \end{array}\right] = \left[\begin{array}{c} f \\ \hline g \end{array}\right] \tag{3.93}$$

Using Block Elimination using Crout factorization or BEC algorithm (Govaerts 1991), the LHS coefficient matrix in Eq. (3.93) can be decomposed as follows:

$$\begin{bmatrix} A & b \\ c & d \end{bmatrix} = \begin{bmatrix} A & 0 \\ c & \delta \end{bmatrix} \begin{bmatrix} I & v \\ 0 & 1 \end{bmatrix} \tag{3.94}$$

Global solution to the original problem at time $(T + \Delta T)$ can now be obtained using the following steps:

3.6.1 Step 1: Solve Av = b

$$\begin{bmatrix} \tilde{\tilde{M}}^S & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & \tilde{\tilde{M}}^2 & 0 \\ 0 & \cdots & 0 & \tilde{M}^1 \end{bmatrix} \begin{bmatrix} \tilde{\tilde{v}}^S_{\xi^S} \\ \vdots \\ \tilde{\tilde{v}}^2_{\xi^2} \\ \tilde{v}^1_{n+1} \end{bmatrix} = \begin{bmatrix} \tilde{\tilde{L}}^S \\ \vdots \\ \tilde{\tilde{L}}^2 \\ \tilde{L}^1 \end{bmatrix} \tag{3.95}$$

Where:

$$\tilde{\tilde{v}}^i_{\xi^i} = \begin{bmatrix} \tilde{v}^i_1 & \tilde{v}^i_2 & \cdots & \tilde{v}^i_{\xi^i} \end{bmatrix}^T \tag{3.96}$$

Since the LHS (coefficient) matrix and the RHS vector in Eq. (3.95) remain constant in case of linear structural systems, the above equation is computed only once before entering the global time-stepping loop. Starting with Ω^1:

$$\tilde{M}^1 \tilde{v}^1_{n+1} = \tilde{L} \quad \Rightarrow \quad \begin{bmatrix} M^1 & C^1 & K^1 \\ -\Delta T \gamma^1 & I & 0 \\ -\Delta T^2 \beta^1 & 0 & I \end{bmatrix} \begin{bmatrix} \ddot{v}^1_{n+1} \\ \dot{v}^1_{n+1} \\ v^1_{n+1} \end{bmatrix} = \begin{bmatrix} L^{1^T} \\ 0 \\ 0 \end{bmatrix} \tag{3.97}$$

Comparing Eq. (3.97) with Eq. (3.28) we see that LHS coefficient matrix represents the Newmark set of equations for Ω^1 with zero initial conditions. The solution vector can hence be obtained by using the Black box routine for Newmark Method, Figure 3-9.

Notice that the RHS (load) vector consists of interface connectivity matrix (Boolean projection matrix in case of conforming sub-domains), accordingly the solution array v in Eq. (3.96) is also referred to as the unit load response matrix. The first index of interface connectivity matrix is equal to the total number of DOF associated with the interface of Lagrange Multipliers, i.e. $\mathrm{DOF}(\lambda)$. Hence the above system needs to be solved successively by considering one RHS column at a time; that is by loading one sub-domain interface DOF at a time. Introducing the notation $[:, j]$ to represent [all rows, jth column] of the corresponding RHS vector, the solution array will be an aggregate of $\mathrm{DOF}(\lambda)$ columns as follows:

$$\tilde{v}^1_{n+1} = \begin{array}{c} \begin{array}{ccc} \underset{\downarrow}{j=1} & \ldots & \underset{\downarrow}{j=DOF(\lambda)} \end{array} \\ \begin{bmatrix} \ddot{v}^1_{n+1} & \cdots & \ddot{v}^1_{n+1} \\ \dot{v}^1_{n+1} & \cdots & \dot{v}^1_{n+1} \\ v^1_{n+1} & \cdots & v^1_{n+1} \end{bmatrix} \end{array} \tag{3.98}$$

In general, for any sub-domain Ω^i $(i = 1, 2, \ldots S)$ the solution array is structured as:

$$\tilde{v}^i_{\eta^i} = \begin{array}{c} \begin{array}{ccc} \overset{j=1}{\downarrow} & \cdots & \overset{j=DOF(\lambda)}{\downarrow} \end{array} \\ \begin{bmatrix} \ddot{v}^i_{\eta^i} & \cdots & \ddot{v}^i_{\eta^i} \\ \dot{v}^i_{\eta^i} & \cdots & \dot{v}^i_{\eta^i} \\ v^i_{\eta^i} & \cdots & v^i_{\eta^i} \end{bmatrix} \end{array} \tag{3.99}$$

Step 1 represented as a sub-domain independent pseudo-code can be illustrated as follows:

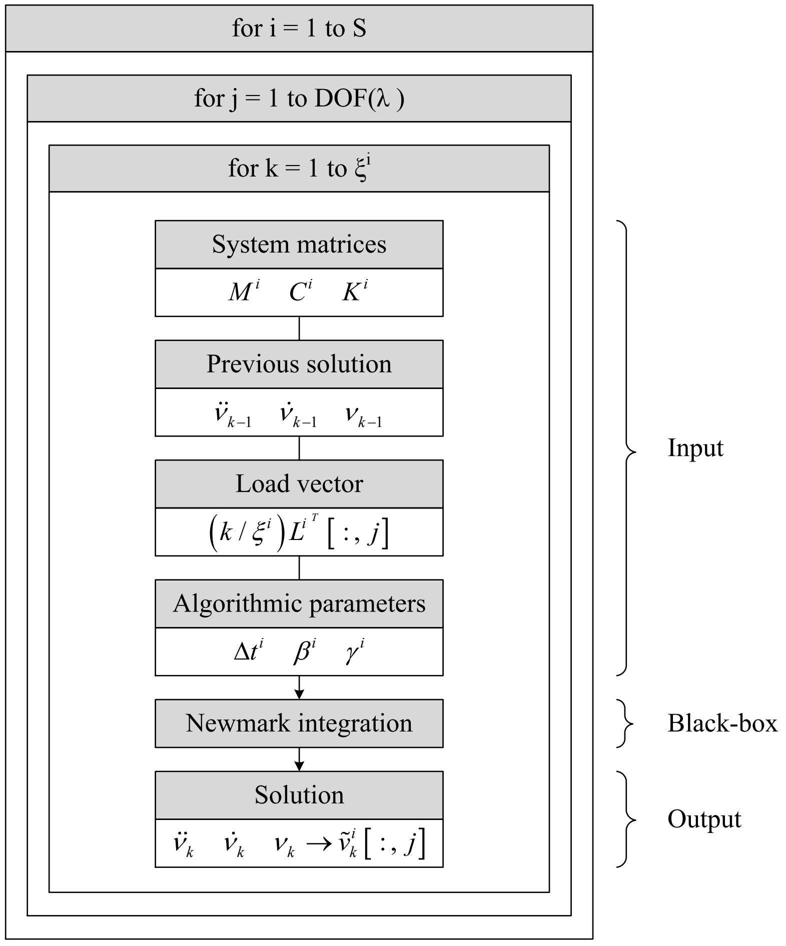


Figure 3-12: Pseudo-code for computing unit load response matrix for component sub-domains

3.6.2 Step 2: Compute δ = d – cv

$$\begin{aligned}\delta &= [0]-\left[\tilde{\tilde{B}}^S \ \cdots \ \tilde{\tilde{B}}^2 \ \tilde{B}^1\right]\left[\tilde{\tilde{v}}^S_{\xi^S} \ \cdots \ \tilde{\tilde{v}}^2_{\xi^2} \ \tilde{v}^1_{n+1}\right]^T \\ &= -\left[\tilde{\tilde{B}}^S\tilde{\tilde{v}}^S_{\xi^S}+\ldots+\tilde{\tilde{B}}^2\tilde{\tilde{v}}^2_{\xi^2}+\tilde{B}^1\tilde{v}^1_{n+1}\right] \\ &= -\left[\tilde{B}^S\tilde{v}^S_{n+1}+\ldots+\tilde{B}^2\tilde{v}^2_{n+1}+\tilde{B}^1\tilde{v}^1_{n+1}\right] \\ &= -\left\{\sum_{i=1}^{S} L^i\left[\overset{\overset{j=1}{\downarrow}}{\dot{v}^i_{n+1}} \ \overset{\cdots}{\cdots} \ \overset{\overset{j=DOF(\lambda)}{\downarrow}}{\dot{v}^i_{n+1}}\right]\right\}\end{aligned} \tag{3.100}$$

Notice that in order to compute δ, we need the solution from **Step 1** at $(n+1)$ time instants. Hence for component sub-domains with $\xi^i > 1$, we only require the solution at $\xi^i = \eta^i$. This observation can be used to our advantage by storing results only for $\xi^i = \eta^i$ in Eq. (3.96) and (3.99). Matrix δ is similar to that obtained in the GC Method for the case of $\Delta T = \Delta t$, (Gravouil & Combescure 2001). As we can see, it may be calculated only once before entering the global time-stepping loop.

3.6.3 Step 3: Solve Aw = f

$$\begin{bmatrix} \tilde{\tilde{M}}^S & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & \tilde{\tilde{M}}^2 & 0 \\ 0 & \cdots & 0 & \tilde{M}^1 \end{bmatrix} \begin{bmatrix} \tilde{\tilde{w}}^S_{\xi^S} \\ \vdots \\ \tilde{\tilde{w}}^2_{\xi^2} \\ \tilde{w}^1_{n+1} \end{bmatrix} = \begin{bmatrix} \tilde{\tilde{F}}^S_{\xi^S} \\ \vdots \\ \tilde{\tilde{F}}^2_{\xi^2} \\ \tilde{F}^1_{n+1}-\tilde{N}^1\tilde{U}^1_n \end{bmatrix} \tag{3.101}$$

Where:

$$\tilde{\tilde{w}}^i_{\xi^i} = \left[\tilde{w}^i_1 \ \ \tilde{w}^i_2 \ \cdots \ \tilde{w}^i_{\xi^i}\right]^T \tag{3.102}$$

For Ω^1 we have:

$$\tilde{M}^1 \tilde{w}^1_{n+1} = \tilde{F}^1_{n+1} - \tilde{N}^1 \tilde{U}^1_n \tag{3.103}$$

Since the above system of equations is solved under the action of external forces only, it is equivalent to solving the free problem for Ω^1 as expressed by Eq. (3.34). **Step 3** represented as a sub-domain independent pseudo-code can be illustrated as follows:

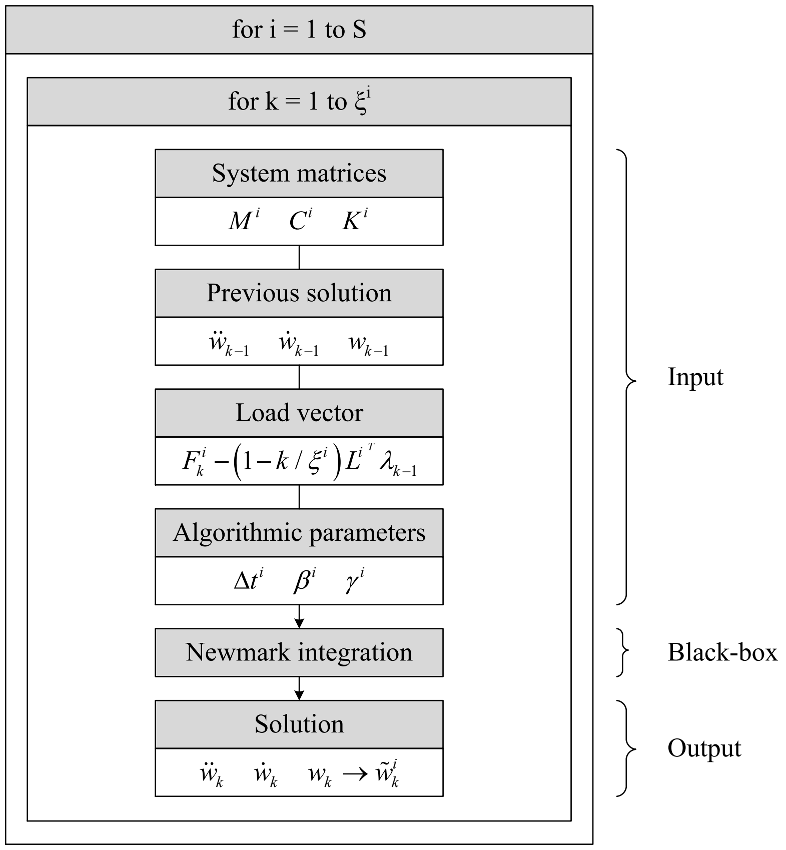

for i = 1 to S
for k = 1 to ξ^i
System matrices
$M^i \quad C^i \quad K^i$
Previous solution
$\ddot{w}_{k-1} \quad \dot{w}_{k-1} \quad w_{k-1}$
Load vector
$F^i_k - \left(1 - k/\xi^i\right) L^{i^T} \lambda_{k-1}$
Algorithmic parameters
$\Delta t^i \quad \beta^i \quad \gamma^i$
Newmark integration
Solution
$\ddot{w}_k \quad \dot{w}_k \quad w_k \rightarrow \tilde{w}^i_k$
Input
Black-box
Output

Figure 3-13: Pseudo-code for computing sub-domain response under the action of (applied) external forces and known interface reactions

3.6.4 Step 4: Compute y = δ^{-1}(g – cw)

$$\begin{aligned}
cw &= \begin{bmatrix} \tilde{\tilde{B}}^S & \cdots & \tilde{\tilde{B}}^2 & \tilde{B}^1 \end{bmatrix} \begin{bmatrix} \tilde{\tilde{w}}^S_{\xi^S} & \cdots & \tilde{\tilde{w}}^2_{\xi^2} & \tilde{w}^1_{n+1} \end{bmatrix}^T \\
&= \tilde{\tilde{B}}^S \tilde{\tilde{w}}^S_{\xi^S} + \ldots + \tilde{\tilde{B}}^2 \tilde{\tilde{w}}^2_{\xi^2} + \tilde{B}^1 \tilde{w}^1_{n+1} \\
&= \tilde{B}^S \tilde{w}^S_{n+1} + \ldots + \tilde{B}^2 \tilde{w}^2_{n+1} + \tilde{B}^1 \tilde{w}^1_{n+1} \\
&= \sum_{i=1}^{S} L^i \dot{w}^i_{n+1}
\end{aligned} \tag{3.104}$$

$$\begin{aligned}
\therefore y &= \delta^{-1} \left\{ [0] - \sum_{i=1}^{S} L^i \dot{w}^i_{n+1} \right\} \\
&= -\delta^{-1} \sum_{i=1}^{S} L^i \dot{w}^i_{n+1} \\
&= \left\{ \sum_{i=1}^{S} L^i \begin{bmatrix} \overset{\overset{j=1}{\downarrow}}{\dot{v}^i_{n+1}} & \overset{\ldots}{\cdots} & \overset{\overset{j=dof(\lambda)}{\downarrow}}{\dot{v}^i_{n+1}} \end{bmatrix} \right\}^{-1} \sum_{i=1}^{S} L^i \dot{w}^i_{n+1}
\end{aligned} \tag{3.105}$$

Since we have already condensed out the interface reactions at intermediate time-steps, we only need to compute the Lagrange Multipliers at time $(T + \Delta T)$. Equation (3.105) hence represents the direct solution of unknown Lagrange Multipliers at time $(T + \Delta T)$.

3.6.5 Step 5: Compute x = w – vy

Global solution (combined from all component sub-domains) at the end of time-step $(T + \Delta T)$ and local solution at intermediate time-steps $(\eta = 1, 2, \ldots \xi)$ is computed in this particular step.

$$x = \begin{bmatrix} \tilde{\tilde{w}}^S_{\xi^S} & \cdots & \tilde{\tilde{w}}^2_{\xi^2} & \tilde{w}^1_{n+1} \end{bmatrix}^T - \begin{bmatrix} \tilde{\tilde{v}}^S_{\xi^S} & \cdots & \tilde{\tilde{v}}^2_{\xi^2} & \tilde{v}^1_{n+1} \end{bmatrix}^T y \tag{3.106}$$

Equation (3.106) is similar to Eq. (3.32) in the sense that the final solution is obtained as a sum of free and link responses, in this case however, one does not need to compute interface reactions at the intermediate time-steps. Now that the global solution is obtained for time $(T+\Delta T)$, we can repeat **Steps 3, 4** and **5** in order to compute the desired evolution of kinematic quantities.

Aforementioned solution algorithm can be conveniently integrated into an existing structural finite element code. DD of the original problem may be performed prior to the analysis (user defined) or during analysis (based on a user defined criterion). Assuming user defined DD, we will now describe a general approach for MGMT implementation:

1) For every sub-domain $\Omega^i (i=1,2,\ldots S)$
    a) Input nodes, coordinates, element connectivity's, material properties, boundary conditions, initial conditions, applied loads, time integration parameters and interface information
    b) Form sub-domain arrays $(M^i, C^i, K^i, F^i \ldots etc)$
    c) Form Boolean projection matrices (B^i)
    d) Form interface constraint equations (P^i)
    e) Form interface connectivity matrices $(L^i = P^i B^i)$
    f) Form unit load response matrices using **Step 1**
2) Compute interface condensation matrix (δ) using **Step 2**
3) Factorize δ (this will be used in **Step 4** to compute unknown Lagrange Multipliers).
4) Compute initial accelerations based on initial conditions.

5) Initialize $\lambda_0 = 0.0$

6) Loop for total number of integration steps based on global time-step ΔT

   a) Computing sub-domain response under the action of (applied) external forces and known interface reactions using **Step 3**

   b) Compute new interface reactions using **Step 4**

   c) Compute global and local response using **Step 5**

7) End loop.

Chapter 4: Programming the MGMT Method

4.1 Finite Element Analysis Programming Interface (FEAPI)

Finite Element Analysis Programming Interface (FEAPI) is a FORTRAN 90 library for the numerical solution of partial differential equations using the FEM. It provides basic building-blocks that allow engineers and software developers to build their own computer programs in order to solve engineering problems using FEM. As a result the library is very flexible and enables its users to apply various FE techniques to any new research in the fields of computational mechanics; e.g. Multiple Grid Multiple Time-Scale (MGMT) Simulations.

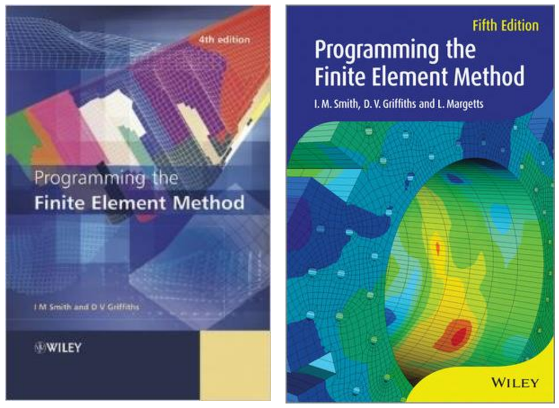


Figure 4-1: Programming the Finite Element Method (Smith et al. 2013)

General purpose (black box) sub-routines and functions for FE computations such as: numerical integration, multi-element assembly of symmetric/un-symmetric systems, factorization and solution algorithms were adapted from ‘Programming the Finite Element Method’ – PFEM (Smith et al. 2013). These program routines are essentially used as building blocks to construct FEAPI kernel, while extending their capabilities to efficiently allow multiple domain, multiscale simulations. Updated procedures are also compatible with Basic Linear Algebra Subprograms (BLAS), Message Passing Interface (MPI) and OpenMP. A primary concern when incorporating FORTRAN 90 sub-routines from PFEM was the flexibility in scaling FE programs to allow multiscale simulations. Since PFEM essentially uses Uniform Grid Uniform Time-scale (UGUT) approach for the forced vibration analysis of linear elastic solids, the driver program and constituent sub-routines from PFEM, were principally designed to handle only single continuous domains, with uniform structured grid and uniform time-steps.

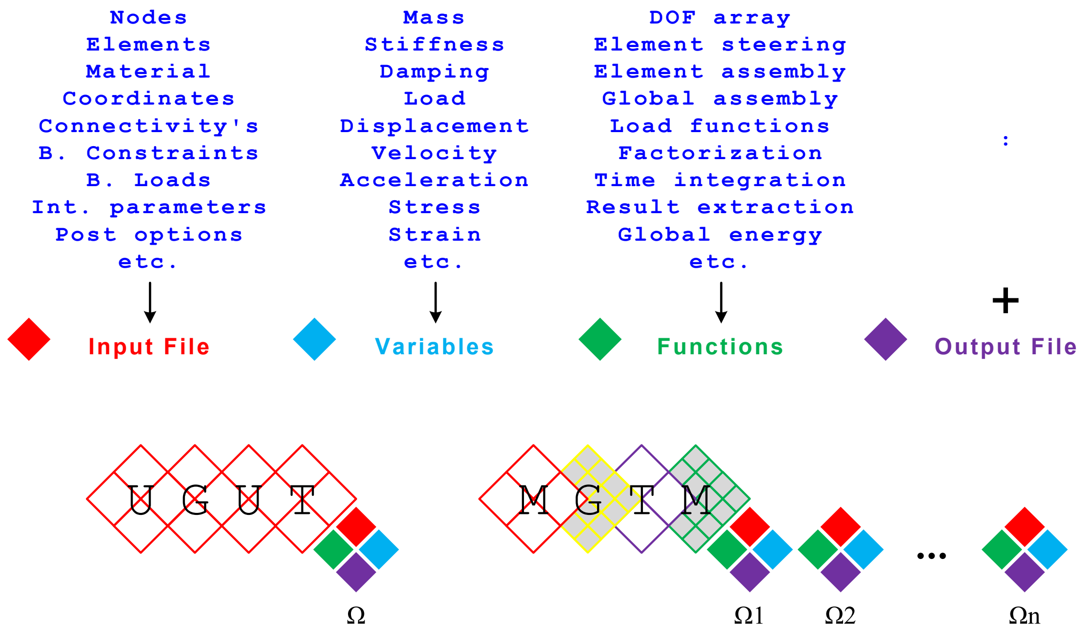


Figure 4-2: PFEM (UGUT) and FEAPI (UGUT/MGMT) program components

As seen in Figure 4-2, a finite element program needs to host several integral components: input/output file handling, program variables and parameters, black-box sub-routines and functions. PFEM, in regards with UGUT, uses only one instance of these components, however for MGMT simulations; these components were appropriately scaled in order to accommodate concurrent simulation of $\Omega 1, \Omega 2, \ldots \Omega n$ sub-domains. FEAPI utilizes advanced FORTRAN 90 programming features such as objects/derived data types (DDT), data encapsulation, function overloading and type inheritance to incorporate scalability. Accordingly, the programming interface is very flexible in allowing easy implementation of new numerical methods.

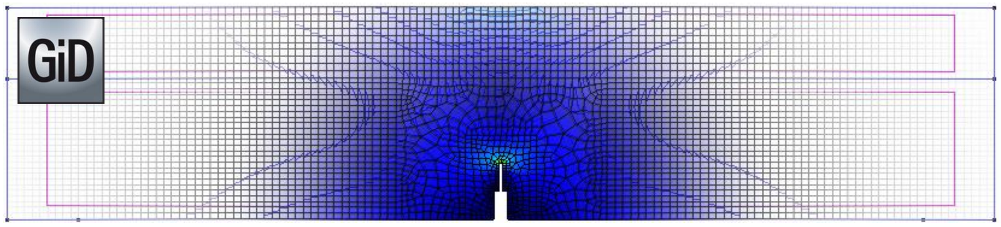


Figure 4-3: GiD – Universal, adaptive and user-friendly pre- and post-processor

FEAPI also hosts auxiliary sub-routine libraries and customized programmable interfaces that allow seamless integration with GiD (http://www.gidhome.com/), a universal, adaptive and user-friendly pre- and post-processor. These program blocks provide an interactive Graphical User Interface (GUI) for the definition, preparation and visualization of all data related to FEAPI numerical simulations; pre and post analysis. Pre analysis data includes the definition of domain geometry, material properties, boundary conditions, solution parameters and other simulation options; whereas post analysis data includes FEAPI results – nodal, elemental, domain; which can be visualized using contour fill / contour lines, vector plots, isosurfaces, deformation, graphs etc.

Complete Application Programming Interface (API) including source codes, list of program variables/parameters, sub-routine/function input-output arguments, call/caller graphs and collaboration diagrams is available at: http://home.gwu.edu/~truparel/.

4.1.1 Program Structure and Component Interfaces

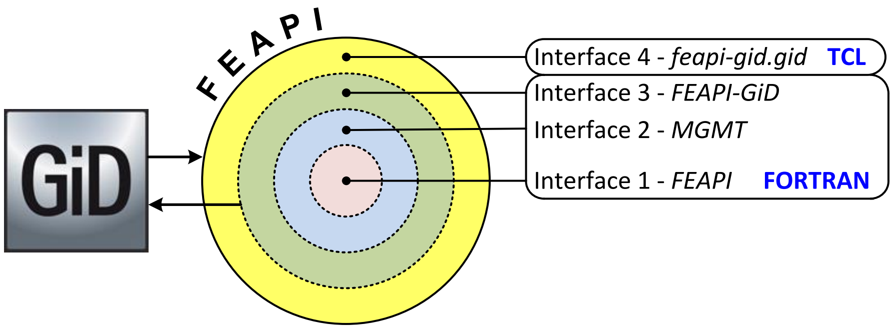
FEAPI consists of following programmable interfaces:

1) FEAPI (collection of basic FEA, auxiliary MATH and program sub-routines)

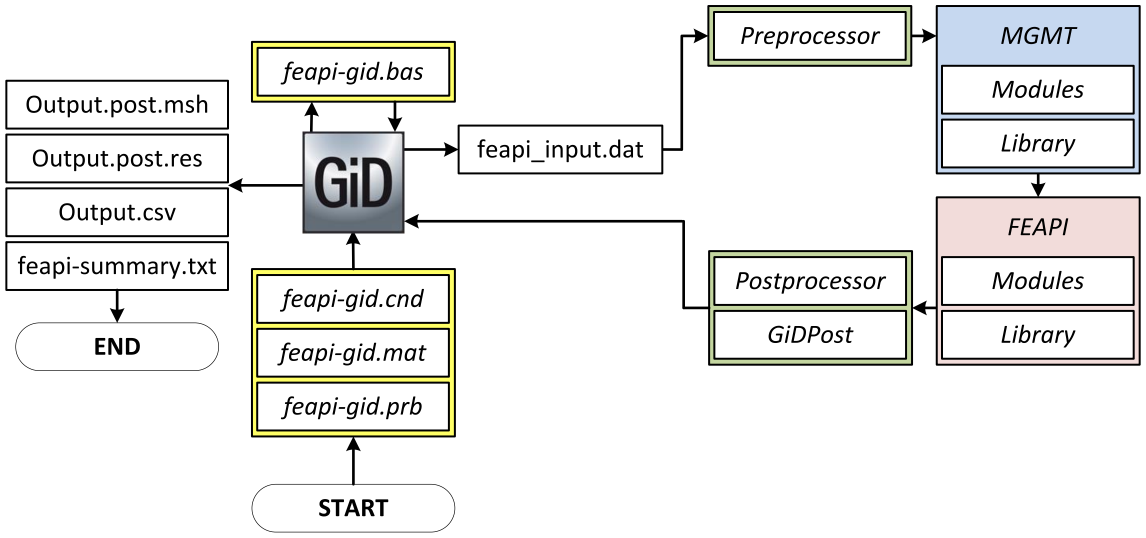
2) MGMT (sub-routines used to perform MGMT simulations)

3) FEAPI-GiD (pre/post processing sub-routines interfacing with GiD)

4) feapi-gid.gid (GiD problem type to define FEAPI characteristic pre-processing)



(a) Interface hierarchy



(b) Interface sub-components and process flow

Figure 4-4: Finite Element Analysis Programming Interface

Figure 4-4 (a) shows the hierarchy between FEAPI interfaces and their interaction levels whereas Figure 4-4 (b) shows the overall process flow and relevant interface sub-components. Interfaces 1, 2 and 3 (FORTRAN libraries) comprise of pre/post processing, computational and solver programs while Interface 4 provides necessary Tool Command Language (TCL) scripts and files to establish GUI communication with GiD. Following sections provide a brief overview of FEAPI interfaces and constituent sub-components.

```
Source
├──+- FEAPI     ├──+ Library
   |               |- Modules
   |
   |- MGMT      ├──+- Library
   |               |- Modules
   |
   |- FEAPI-GiD ├──+- GiDPost
                   |- Postprocessor
                   |- Preprocessor
```

Figure 4-5: FEAPI code structure

A. Interface 1 – FEAPI

Interface 1 hosts the main program *feapi.f90* and primary program variables, sub-routines and functions.

```
FEAPI
├──+- Library ├──+- FEA
|                |- MATH
|                |- Program
|
├──+- Modules ├──+- data_domain.f90
                 |- data_feapi.f90
                 |- data_material.f90
                 |- data_structural.f90
                 |- data_transient.f90
                 |- precision.f90
```

Figure 4-6: FEAPI interface sub-components – Library and Modules

FEAPI > Module represent a collection of program variables, parameters, their declaration and initial values. Detailed description of these variables, their collaboration / inheritance, kind and array size can be found in Appendix D.1.1. Following is a brief description about each module:

- `data_domain` contains FEAPI domain variables, such as DDT for domain parameters (number of nodes and elements, DOF per node, number integration points, nodal coordinates, element connectivity's etc.), domain boundary conditions (restrained, loaded, prescribed nodes and corresponding magnitude vectors) and domain interface info (mortar, non-mortar declaration and interface nodes).
- `data_feapi` contains global program variables, such as DDT for input files (name, location, ID), number of FE domain blocks used, and program block simulation times.
- `data_material` contains variables for domain specific material properties (number of materials, number of material properties, mass matrix formulation).
- `data_structural` contains variables for structural systems, such as mass, stiffness, load vector, solution vector, and stress / strain arrays.
- `data_transient` contains domain specific transient analysis variables, such as integration method, time-step, algorithmic parameters etc.
- `data_precision` contains the minimum KIND necessary to store real numbers with a precision of 15 decimal digits and an exponent in the range $10x10^{-307}$ to $10x10^{307}$.

```
FEAPI
 Library ├──+- FEA ├──+- beemat.f90      ├──+- gsteer.f90
                      |- deemat.f90         |- iniaccl.f90
                      |- domainfx.f90       |- lcontri.f90
                      |- domainie.f90       |- sample.f90
                      |- domainke.f90       |- shapeder.f90
                      |- domainse.f90       |- shapefun.f90
                      |- ecmat.f90          |- sk2chol.f90
                      |- elmat.f90          |- sk2gaus.f90
                      |- elres1.f90         |- skvmul.f90
                      |- esq2gsk.f90        |- slskchol.f90
                      |- esteer.f90         |- slskgaus.f90
                      |- fkdiag.f90         |- slsqlub.f90
                      |- formkdiag.f90      |- slsqluf.f90
                      |- formnf.f90         |- slsqlup.f90
                      |- formsky.f90        |- solvedtrans.f90
                      |- fpstiff.f90        |- sq2lu.f90
                      |- fresidual.f90      |- sq2lup.f90
                      |- fstfearry.f90      |- stressinvar.f90
                      |- gafamily.f90       |- tpfunction.f90
```

Figure 4-7: FEAPI interface sub-components – Library > FEA

```
FEAPI
 Library ├──+- MATH ├──+- crossproduct.f90   ├──+ BLAS
                       |- determinant.f90
                       |- distance.f90
                       |- identity.f90
                       |- inversem.f90
                       |- invert.f90
                       |- l2norm.f90
                       |- piksrt.f90
                       |- scalarproduct.f90
```

Figure 4-8: FEAPI interface sub-components – FEAPI > Library > MATH

```
FEAPI
 Library ├──+- Program ├──+- cputime.f90      ├──+- ppost.f90
                          |- findblock.f90       |- ppres.f90
                          |- getname.f90         |- ppropn.f90
                          |- lnblnk.f90          |- presult.f90
                          |- palloc.f90          |- psaver.f90
                          |- pfilename.f90       |- psetup.f90
                          |- pinput.f90          |- psolve.f90
                          |- postcsv.f90         |- psummary.f90
                          |- ppcsv.f90           |- pterminal.f90
                          |- ppmesh.f90          |- timestamp.f90
```

Figure 4-9: FEAPI interface sub-components – FEAPI > Library > Program

FEAPI library contains fundamental FE, MATH and program sub-routines/functions. A detailed description of these sub-components, list of input/output arguments, inherited modules and hierarchical call/caller diagrams can be found in Appendix D.1.2. Library highlights and their brief introduction is as follows:

- Library > FEA contains sub-routines and functions used to perform FE operations, such as formation of nodal DOF array, computation of element shape functions and their derivatives, computation of strain-displacement and stress-strain matrices, computation and assembly of global system matrices (symmetric/skyline and unsymmetric), computation of augmented stiffness matrix for prescribed boundary conditions, computation of time proportional load functions, computation of global energies, computation of stresses and strains at Gauss integration points, direct integration of structural dynamic systems and LU factorization (with and without pivoting).
- Library > Math contains sub-routines used to perform scalar/array operations such as scalar product, cross product, matrix inverse, matrix determinant and L2 Norm calculations. It also contains efficient and portable library BLAS (Lawson et al. 1979) to perform various vector and matrix operations.
- Library > Program contains auxiliary sub-routines/functions that are essentially used for program operations, such as reading input file, allocating/initializing and updating FEAPI variables, initializing domain data blocks, saving requested simulation results, generating *CSV*, **.post.msh*, **.post.res* and **.txt* output files and computing CPU solution times.

B. Interface 2 – MGMT

Interface 2 contains a collection of black box sub-routines used to perform concurrent multiple grid multiple time-scale simulations for linear structural dynamic equations.

```
MGMT
├──+- Library ├──+- fbpmat.f90  ├──+- mgmtdid.f90
|                |- fmcmat.f90     |- mgmtgdb.f90
|                |- fulmat.f90     |- mgmtvar.f90
|                |- immat.f90      |- solvemgmt.f90
|                |- invmap.f90     |- valued.f90
|                |- mgmtall.f90    |- valuem.f90
|
├──+- Modules ├──+- data_mgmt.f90
```

Figure 4-10: MGMT interface sub-components – Library and Modules

A brief introduction to MGMT sub-components is as follows:

- MGMT > Library contains sub-routines designed specifically for operations such as allocation of MGMT variables, setup of MGMT sub-domain hierarchy (according to time-step ratio) and initializing of corresponding domain data blocks, formation of Boolean projection matrix, formation of multi-constraint interface matrix, computation of sub-domain unit load response and solution of MGMT equations.
- `data_mgmt` stores global/sub-domain specific MGMT variables, such as sub-domain ID's, time-step ratios, list of interface nodes/DOF, sub-domain coupling matrices, interface condensation matrix, sub-domain unit load response arrays, interface projection matrix and array of Lagrange Multipliers,

A detailed description of these sub-routines, DDTs, their inheritance, variable/array dimensions and collaboration can be found in Appendix D.3.

C. Interface 3 – FEAPI–GiD

Interface 3 forms the outer core layer of FEAPI that enables interaction with GiD.

```
FEAPI-GiD
├──+-Preprocessor ├──+- ip_dbcs.f90
|                    |- ip_intf.f90
|                    |- ip_mats.f90
|                    |- ip_mesh.f90
|                    |- ip_post.f90
|                    |- ip_tran.f90
|
├──+-Postprocessor ├──+- gidgnum.f90
|                     |- gidgxyz.f90
|                     |- gidmesh.f90
|                     |- gidrmat.f90
|                     |- gidropn.f90
|                     |- gidrvec.f90
|
├──+- GiDPost ├──+- data_post.f90
                 |- GiDPost.f90
                 |- GiD_hdf5.lib
                 |- GiD_post.lib
                 |- GiD_zlib.lib
```

Figure 4-11: FEAPI-GiD interface sub-components – Preprocessor

- Preprocessor is used to read input files generated using Interface 4 – *feapi-gid.gid*. It is also used to allocate program variables and initialize corresponding data within FEAPI.
- Postprocessor is used to output FEAPI post result files for visualization using GiD.
- GiDPost is a set of FORTRAN 90 modules, functions and sub-routines used to write FEAPI post result files in ASCII or binary compressed format. This library is copyrighted by CIMNE (http://www.gidhome.com/)

Refer Appendix D.2 for a comprehensive description on these sub-components.

D. Interface 4 – *feapi-gid.gid*

This interface provides an interactive GUI for the definition, preparation and visualization of all input data related to FEAPI numerical simulations. It works in conjugation with GiD and is utilized to define pre analysis data, such as domain geometry, material properties, boundary conditions, solution parameters and other simulation options. The principle advantage in implementing this interface is that it lets you perform all necessary declarations and configurations using GiD GUI, hence avoiding the need to modify the solver itself. Pre-processing is initiated via customized GiD problem type *feapi-gid.gid*. This package is required to define, assign and output all the necessary information that is required to perform a FEAPI simulation. In order to achieve this, some files are used to define global parameters, conditions, materials, general analysis options, unit systems, and the structured input file format for FEAPI program solver. These files, Figure 4-12, combined together constitute the GiD problem type auxiliary interface *feapi-gid.gid*.

```
feapi-gid.gid ├──+- feapi-gid.bas
                 |- feapi-gid.cnd
                 |- feapi-gid.mat
                 |- feapi-gid.prb
                 |- feapi-gid.win.bat
```

Figure 4-12: feapi-gid.gid interface sub-components

Conditions file (*feapi-gid.cnd)* contains information about conditional variables that can be applied to different geometrical and FE entities. These conditions are accessible via *GiD* > *Data* > *Conditions* and contain following definitions:

1) Restrained (constrained) DOF
2) Forced DOF w/ respective magnitudes and time functions

3) Prescribed (displacement) DOF w/ respective magnitudes and time functions

4) Domain interface nodes

5) Interface info (Mortar/Non-mortar interface declaration)

Materials file (*feapi-gid.mat*) contains a list of base materials and their mechanical properties. In a fashion similar to defining the conditional entity, a material declaration can be considered as a group of fields containing its name, related mechanical properties and their corresponding values. Accordingly, any of these base materials and associated fields can be used to define new materials during the pre-processing phase. Contrary to the conditions file, a material can be assigned to different levels of geometrical entities (lines, surfaces or volumes) and can even be assigned directly to grid elements. Included material definitions, Table 4-1, are accessible via *GiD* > *Data* > *Materials*.

Table 4-1: FEAPI material library

Name	Modulus of Elasticity (E)	Poisson's Ratio (ν)	Mass Density (ρ)
Steel	207E9 N/m^2	0.3	7830 Kg/m^3
Aluminum	69e9 N/m^2	0.33	2712 Kg/m^3
Concrete	40e9 N/m^2	0.2	2400 Kg/m^3
Copper	117e9 N/m^2	0.36	8940 Kg/m^3
Lead	13790 N/m^2	0.425	11340 Kg/m^3
Ph-Bronze	116e9 N/m^2	0.33	8900 Kg/m^3
Zinc	82737 N/m^2	0.25	7135 Kg/m^3
Tin	47e9 N/m^2	0.33	7280 Kg/m^3

Problem Data File (*feapi-gid.prb*) contains information about common simulation parameters, program variables and corresponding interval data. Most simulation parameters are global in nature, i.e. they do not concern any particular geometrical or FE entity. This differs from the previous definitions of conditions and materials properties since they were all assigned to specific entities. Examples of entity independent global variables include: DOF per node, type of solution algorithm, algorithmic parameters, convergence criteria, post result options and so on. Within these declarations, one may consider the definition of specific problem data for the whole process or intervals of data to allow different variable values for different solution intervals. Typically, one can define a different load case for different time intervals in dynamic problems and consequently define variable loads, variations in time-steps, changes in boundary conditions and so on. FEAPI relevant definitions included in this file are as follows:

1) **Global Variables**

   GiD > Data > Problem Data > Global Variables

- Analysis type = Static, Transient
- Problem type = Rod, Beam, PlaneStress, PlaneStrain, Axisymmetric, General3D
- Element type = Line, Triangle, Quadrilateral, Tetrahedron, Hexahedron
- Degrees of freedom per node
- Number of Gauss integration points

2) **Transient Analysis Options**

   GiD > Data > Problem Data > Transient Analysis Options

- Direct integration methods = Newmark, WBZ, HHT, Generalized-α
- Algorithmic parameters = Alpha-m, Alpha-f, Beta, Gamma, Delta, Integration time-step, total number of steps, simulation end time
- Inertia = Consistent or lumped mass matrix formulation
- Damping = Rayleigh damping

3) **Post Result Options**

GiD > Data > Problem Data > Post Result Options

- Post frequency
- Nodal results = Displacement, Velocity, Acceleration
- Element results = Stress (Cauchy, Von Mises), Strains
- Domain results = Kinetic, Stiffness, External, Interface Energy

Once a geometrical entity has been created, meshed, assigned boundary conditions, material properties and other program variables, **Base File** (*feapi-gid.bas*) is used to generate a structured FEAPI input file *(*.dat*). The file primarily consists of Tool Command Language (TCL) scripts that synthesize information from constituent problem files (**.cnd*, **.mat* and **.prb*) and sequentially print them in a structured format that is readable via FEAPI. A typical input file generation process can be summarized using Figure 4-13. This process is initiated via *GiD > Calculate > Calculate*. An example FEAPI input file and the description of basic data blocks can be found in Appendix A.1 and Appendix A.2 respectively.

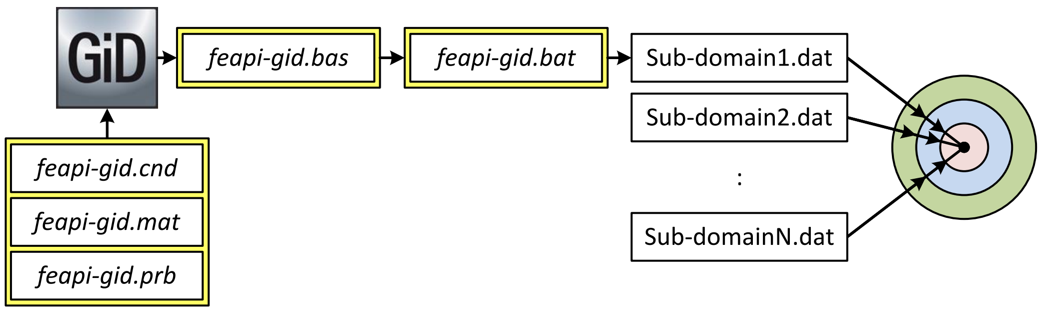


Figure 4-13: Input file creation flowchart

4.1.2 Program Installation

Interfaces 1 – 4 are included as an integral part of the FEAPI source distribution (*feapi.zip*) available at: http://home.gwu.edu/~truparel/. Following files are also included with the source code:

1) *feapi-configuration.txt* – Configuration file.
2) *feapi.exe* – Binary executable file.
3) *feapi.chm* – Application programming interface (compressed HTML help)

To install the binary version, copy the executable file (*feapi.exe*), configuration file (*feapi-configuration.txt*) and the API (*feapi.chm*) to desired location (*example C:\FEAPI\*) and create input/output directories (*C:\FEAPI\Input and C:\FEAPI\Output*). Configuration file contains pointers for FEAPI input/output directory paths and additional MGMT post result options. These keywords must be defined before running FEAPI. An example file is described below:

```
FEAPI::INPUT
C:\FEAPI\INPUT\! FEAPI input directory

FEAPI::OUTPUT
C:\FEAPI\OUTPUT\! FEAPI output directory

MGMT::POST
1! Post frequency
1! Displacements: 1 = Yes, 0 = No
1! Velocities: 1 = Yes, 0 = No
1! Accelerations: 1 = Yes, 0 = No
1 1! Stresses (Cauchy/Von Mises): 1 = Yes, 0 = No; (1/2)
1! Strains: 1 = Yes, 0 = No
1! Kinetic energy: 1 = Yes, 0 = No
1! Stiffness energy: 1 = Yes, 0 = No
1! External work: 1 = Yes, 0 = No
1! Interface energy: 1 = Yes, 0 = No
```

Before using *feapi-gid.gid*, download and install the latest official version of GiD from http://www.gidhome.com/download/official-versions.

Instructions for installing *feapi-gid.gid* are as follows:

1) Unzip the contents of *feapi-gid.zip*
2) Copy *feapi-gid.gid* to ...\GiD_installation_directory\Problemtypes\\
3) *feapi-gid* should now be accessible via *GiD > Data > Problem Types*

Alternately, *feapi-gid.gid* may also be copied to another preferred location. In this case *feapi-gid* is accessible via *GiD > Data > Problem Types > Load*. Command execution file (*feapi-gid.win.bat*) requires the path to FEAPI input directory that is defined using `FEAPI::INPUT` keyword in *feapi-configuration.txt*. After the input file has been created and saved inside GiD, it can be generated and exported directly into the FEAPI input directory via *GiD > Calculate > Calculate*. To provide this path edit the windows batch file as follows:

```
@ECHO OFF
rem FEAPI Input File Base Name = %1
rem FEAPI Input File Location = %2
rem FEAPI-GiD Location = %3
rem Copy FEAPI input file %1.dat to FEAPI Input Directory
rem FEAPI input directory is defined in feapi-configuration.txt
copy %2\%1.dat FEAPI input directory <defined using FEAPI::INPUT>
```

4.1.3 Driver Programs

A. Uniform Grid Uniform Time-scale Simulations

This program solves linear structural dynamic systems using direct time-integration methods such as Newmark, HHT-α, WBZ or Generalized-α.

```
→ FEAPI.f90 - Main code
 → pterminal.90 - Select program 1, Read project title
 → pfilename.f90 - Read input filename
 → palloc.f90 - Allocate necessary derived data types for program 1
 → pinput.f90 - Read FEAPI input data
  → ip_mesh.f90 - Read global parameters and mesh data
  → ip_mats.f90 - Read material data
  → ip_dbcs.f90 - Read domain boundary conditions
  → ip_trans.f90 - Read transient analysis options
  → ip_post.f90 - Read post analysis options
 → psetup.f90 - Setup program variables
  → fstfearray.f90 - Allocate and initialize FE structural arrays
   → formsky.f90 - Form FE structural arrays as skyline vectors
 → presult.f90 - Allocate and initialize result storage space
 → psolve.f90 - Solve transient dynamic system
  → solvedtrans.f90 - Solve using direct time integration (program 1)
   → iniaccl.f90 - Solve for initial accelerations using initial conditions
   ┌ Loop for total number of simulation time-steps
   ├ Read externally applied loads at current time-step
   ├ gafamily.f90 - Solve current time-step using Generalized-α method
   ├ elres.f90 - Compute element results (stresses and strains)
   ├ domainke.f90 - Compute kinetic energy for current time-step
   ├ domainse.f90 - Compute stiffness energy for current time-step
   ├ domainfx.f90 - Compute external work for current time-step
   ├ pasver.f90 - Save results for current time-step
   └ End loop
 → ppost.f90 - Post simulation results
 → psummary.f90 - Post simulation summary
```

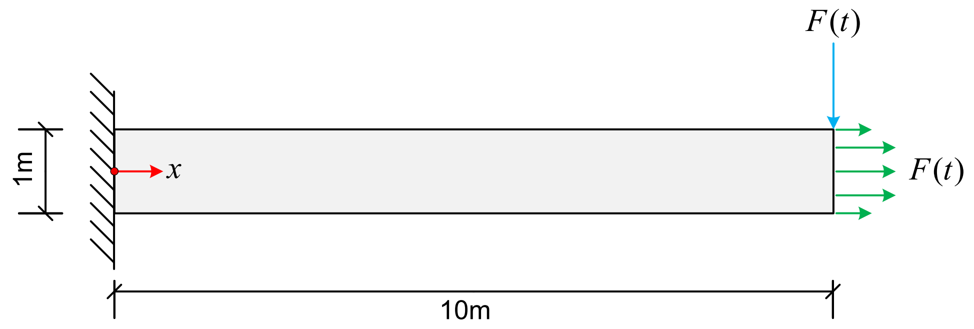
B. Multiple Grid Multiple Time-scale Simulations

This program solves linear structural dynamic systems using MGMT method.

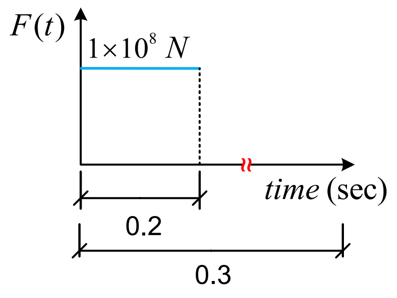
```
→ FEAPI.f90 - Main code
 → pterminal.90 - Select program 2, read project title and number of sub-domains
 ┌ Loop for total number of sub-domains
 ├ pfilename.f90 - Read sub-domain input filename
 └ End loop
 → palloc.f90 - Allocate necessary derived data types for program 2
 → pinput.f90 - Read FEAPI input data
  ┌ Loop for total number of sub-domains
  ├ ip_mesh.f90 - Read sub-domain global parameters and mesh data
  ├ ip_mats.f90 - Read sub-domain material data
  ├ ip_dbcs.f90 - Read sub-domain boundary conditions
  ├ ip_trans.f90 - Read sub-domain transient analysis options
  ├ ip_post.f90 - Read sub-domain post analysis options
  ├ ip_intf.f90 - Read sub-domain interface data
  └ End loop
→ psetup.f90 – Setup program variables
  ┌ Loop for total number of sub-domains
  ├ fstfearray.f90 – Allocate and initialize FE structural arrays
  │ → formsky.f90 – Form FE structural arrays as skyline vectors
  └ End loop
  → mgmtdid.f90 – Setup sub-domain hierarchy
  → mgmtall.f90 – Allocate and initialize MGMT specific program variables
  → mgmtvar.f90 – Setup MGMT variables
    ┌ Loop for total number of sub-domains
    ├ fbpmat.f90 – Form sub-domain Boolean projection matrix
    └ End loop
    ┌ Loop for total number of sub-domains
    ├ Assign interface projection bias
    └ End loop
    ┌ Loop for total number of sub-domains
    ├ fmcmat.f90 – Form sub-domain multi Constraint matrix
    └ End loop
    ┌ Loop for total number of sub-domains
    ├ Form sub-domain interface connectivity matrix
    └ End loop
    ┌ Loop for total number of sub-domains
    ├ fulmat.f90 – Form sub-domain unit load matrix
    └ End loop
    → Form global interface condensation matrix
    → sq2lu.f90 – Factorize global interface condensation matrix
  → mgmtgdb.f90 – Form global domain data block
→ presult.f90 – Allocate and initialize result storage space
→ psolve.f90 – Solve transient dynamic system
  → solvemgmt.f90 – Solve using MGMT method (program 2)
    ┌ Loop for total number of sub-domains
    ├ iniaccl.f90 – Solve for initial accelerations using initial conditions
    └ End loop
    ┌ Loop for total number of global time-steps
    │┌ Loop for local sub-domain time-steps between global time-steps
    │├ Read externally applied loads at current time-step
    │├ gafamily.f90 – Solve current time-step using Generalized-α method
    │└ End loop
    ├ Compute updated vector of Lagrange multipliers (interface reactions)
    │┌ Loop for total number of sub-domains
    │├ Compute link response
    │├ Update local sub-domain solution (free + link response)
    │└ End loop
    │┌ Loop for total number of sub-domains
    │├ elres.f90 – Compute sub-domain element results (stresses and strains)
    │├ domainke.f90 – Compute sub-domain kinetic energy for current time-step
    │├ domainse.f90 – Compute sub-domain stiffness energy for current time-step
    │├ domainfx.f90 – Compute sub-domain external work for current time-step
    │├ domainie.f90 – Compute sub-domain interface energy for current time-step
    │├ pasver.f90 – Save results for current time-step
    │└ End loop
    ├ Update global solution for current time-step
    └ End loop
→ ppost.f90 – Post sub-domain/global simulation results
→ psummary.f90 – Post simulation summary
```

Chapter 5: Numerical Analysis and Verification

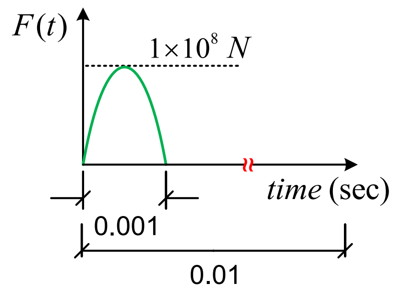
In this chapter we will present some examples that will help us evaluate – 1) stability, 2) accuracy and 3) computational efficiency of the MGMT Method. We will analyze a 2D cantilever beam under forced vibrations; transverse (Example 1) and longitudinal (Example 2). The domain under analysis and transient forces applied at the free end are as shown in Figure 5-1. In either case, the domain is discretized using 4-node quadrilateral elements with 2 DOF/node, plane stress formulation, consistent mass matrix and zero damping. Isotropic linear elastic material properties with modulus of elasticity (E) = 2.07x10^{11} N/m^2, Poisson's ratio (ν) = 0.3 and mass density (ρ) = 7.83x10^3 Kg/m^3 are used.



(a) Structural domain under consideration



(b) Example 1 – Transverse load



(c) Example 2 – Longitudinal load

Figure 5-1: Forced vibrations analysis of a cantilever beam

Different scenarios analyzed under these examples are as follows:

Case 1) Uniform grid uniform time-step (UGUT) reference cases

Case 2) Multiple Grid Multiple Time-Step I-I Coupling (MGMT1)

Case 3) Multiple Grid Multiple Time-Step I-E Coupling (MGMT2)

Case 4) Multiple Time-Step I-I Coupling (MTC)

Case 5) Multiple Grid Coupling with Implicit Time Integration (MGC)

5.1 Benchmark Case Descriptions

5.1.1 Case 1 – Uniform grid uniform time-step (UGUT) reference cases

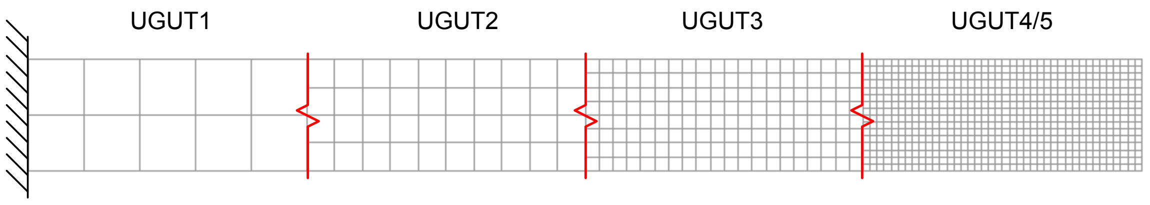


Figure 5-2: Case 1 (UGUT) domain grids

Table 5-1: Case 1 (UGUT) simulation parameters

Sub-case ID	Grid spacing (H)	Newmark parameters	Time-step (ΔT)	
			Example 1	Example 2
UGUT1	0.5	β=0.25, γ=0.5 (Implicit)	$1x10^{-3}$	$1x10^{-4}$
UGUT2	0.25	β=0.25, γ=0.5 (Implicit)	$0.5x10^{-3}$	$0.5x10^{-4}$
UGUT3	0.125	β=0.25, γ=0.5 (Implicit)	$0.25x10^{-3}$	$0.25x10^{-4}$
UGUT4	0.0625	β=0.25, γ=0.5 (Implicit)	$0.125x10^{-3}$	$0.125x10^{-4}$
UGUT5	0.0625	β=0.0, γ=0.5 (Explicit)	$1x10^{-6}$	$1x10^{-6}$

Reference uniform grid uniform time-step (UGUT) simulations are performed by improving refinement, both in spatial as well as temporal domains, in order to establish numerical convergence using conventional FEM (FEAPI – Program 1). This will allows us to establish a solution that is desirable, in accuracy and/or efficiency, and one that can only be obtained by using traditional FE algorithms. Accordingly, primary conclusions for evaluating overall performance (stability, accuracy and efficiency) of MGMT Method are all derived by comparing MGMT results with equivalently converged UGUT cases. Sub-cases UGUT1 to UGUT4 employ implicit constant average acceleration time integration algorithms, whereas UGUT5 uses explicit Central Difference Method (CDM), also referred to as the Velocity Verlet Method. Corresponding spatial discretizations and simulation parameters, for Example 1 and Example 2, are as shown in Figure 5-2 and Table 5-1.

When comparing results, we will use Root Mean Square Error (RMSE), also called the root mean square deviation (RMSD), and Normalized RMSE (NRMSE) to provide a quantitative measure of accuracy between measured variables. NRSME is often expressed as a percentage, where lower values indicate less variance. This will enable us to quantify the error (on an average sense) between respective UGUT and MGMT cases. RMSE and NRMSE are computed as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\mathrm{var}_i^{\mathrm{MGMT}} - \mathrm{var}_i^{\mathrm{UGUT}}\right)^2} \tag{5.1}$$

$$NRMSE = \frac{RMSE}{\mathrm{var}_{\mathrm{max}}^{\mathrm{UGUT}} - \mathrm{var}_{\mathrm{min}}^{\mathrm{UGUT}}} \times 100\% \tag{5.2}$$

In above equations, n represents the total number of data points recorded at global time increments. Based on NRMSE we will also highlight the error ranking between respective MGMT cases, and constituent special cases: MTC and MGC, so one can identify the best scenario for a particular performance evaluation factor.

5.1.2 Case 2 – Multiple Grid Multiple Time-Step I-I Coupling (MGMT1)

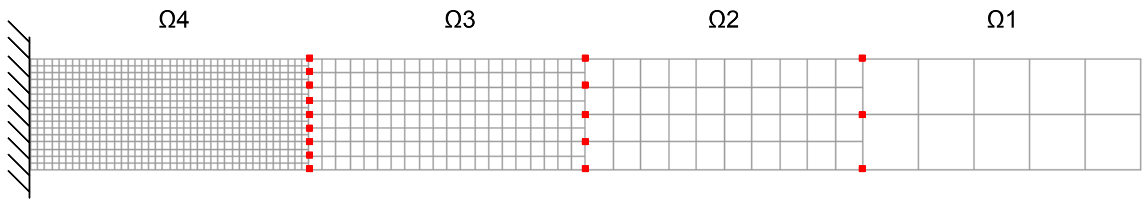


Figure 5-3: Case 2 (MGMT1) sub-domain grids

Table 5-2: Case 2 (MGMT1) simulation parameters

Sub-domain	Grid spacing (H)	Newmark parameters	Time-step (Δt)	
			Example 1	Example 2
Ω1	0.5	β=0.25, γ=0.5 (Implicit)	$1x10^{-3}$	$1x10^{-4}$
Ω2	0.25	β=0.25, γ=0.5 (Implicit)	$0.5x10^{-3}$	$0.5x10^{-4}$
Ω3	0.125	β=0.25, γ=0.5 (Implicit)	$0.25x10^{-3}$	$0.25x10^{-4}$
Ω4	0.0625	β=0.25, γ=0.5 (Implicit)	$0.125x10^{-3}$	$0.125x10^{-4}$

Case 2 represents a 4 sub-domain multiple grid multiple time-scale approach, in which every component sub-domain has a distinct spatial and temporal resolution, as shown in Figure 5-3 and Table 5-2. Grid decomposition and corresponding time-scale discretization begins with a coarse resolution on the RHS (free end) and gradually refines to fine scale resolution on the LHS (fixed end) of the structural domain under consideration. The total number of interface (mortar) nodes in this case is 17 (🟥). This case employs I-I time integration coupling with a global time-step $\Delta T = 1\times 10^{-3}$ for Example 1 and $\Delta T = 1\times 10^{-4}$ for Example 2. Corresponding time-step ratios for both Example 1 and Example 2 are $\xi^1 = 1$, $\xi^2 = 2$, $\xi^3 = 3$ and $\xi^4 = 4$, see Table 5-2. A graphical representation of relative sub-domain time-stepping is shown in Figure 5-4.

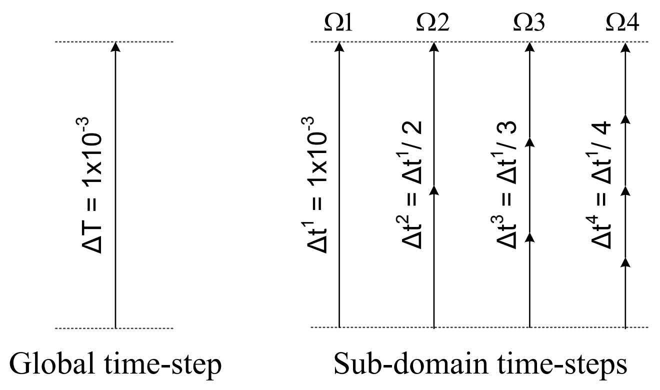


Figure 5-4: Case 2 sub-domain time increments

5.1.3 Case 3 – Multiple Grid Multiple Time-Step I-E Coupling (MGMT2)

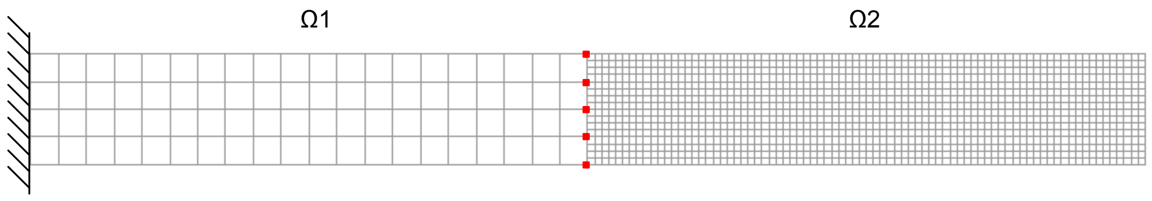


Figure 5-5: Case 3 (MGMT2) sub-domain grids

Table 5-3: Case 3 (MGMT2) simulation parameters

Sub-domain	Grid spacing (H)	Newmark parameters	Time-step (Δt)	
			Example 1	Example 2
Ω1	0.25	β=0.25, γ=0.5 (Implicit)	$0.5x10^{-3}$	$0.5x10^{-4}$
Ω2	0.0625	β=0.0, γ=0.5 (Explicit)	$1x10^{-6}$	$1x10^{-6}$

Case 3 represents a 2 sub-domain multiple grid multiple time-scale approach in which every component sub-domain has a distinct spatial and temporal resolution, as shown in Figure 5-5 and Table 5-3. Additionally, Ω1 employs implicit time integration and Ω2 employs explicit time integration (CDM). Grid decomposition and the corresponding time-scale discretization begin with a coarse resolution on the LHS (fixed end) and fine scale resolution on the RHS (free end) of the structural domain under consideration. The total number of interface (mortar) nodes in this case is 5 (🟥). The global integration time-step is $\Delta T = 0.5\times10^{-3}$ for Example 1 and $\Delta T = 0.5\times10^{-4}$ for Example 2. Corresponding sub-domain time-step ratios are $\xi^1 = 1$, $\xi^2 = 500$ for Example 1 and $\xi^1 = 1$, $\xi^2 = 50$ for Example 2. It should be noted that the explicit time-step $\Delta t^2 = 1\times10^{-6}$ is dictated by the CFL condition, Eq. (2.51), discussed under Section 2.3.1.

5.1.4 Case 4 – Multiple Time-Step I-I Coupling (MTC)

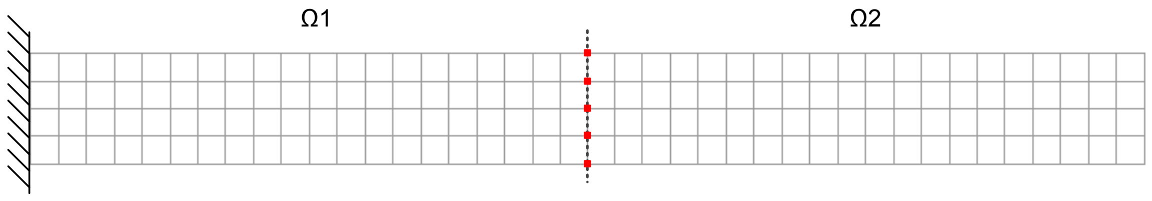


Figure 5-6: Case 4 (MTC) sub-domain grids

Table 5-4: Case 4 (MTC) simulation parameters

Sub-domain	Grid spacing (H)	Newmark parameters
Ω1	0.25	β=0.25, γ=0.5 (Implicit)
Ω2	0.25	β=0.25, γ=0.5 (Implicit)

		Time-step (Δt)			
		Example 1		Example 2	
Case ID	Time-step ratio (ξ^2)	Ω1	Ω2	Ω1	Ω2
MTC1	10	$5x10^{-4}$	$5x10^{-5}$	$5x10^{-5}$	$5x10^{-6}$
MTC2	100	$5x10^{-4}$	$5x10^{-6}$	$5x10^{-5}$	$5x10^{-7}$
MTC3	1000	$5x10^{-4}$	$5x10^{-7}$	$5x10^{-5}$	$5x10^{-8}$
MTC4	10000	$5x10^{-4}$	$5x10^{-8}$	$5x10^{-5}$	$5x10^{-9}$

Under Case 4, we will evaluate the effects of coupling 2 sub-domains with conforming grids, but with distinct time-steps, as shown in Figure 5-6 and Table 5-4. The goal is to quantify the cumulative interface dissipation or accumulation across sub-domain interfaces caused solely due to multiple time-stepping. Increasingly higher order time-step ratios $(\xi_{\max}^2 = 10,000)$ are analyzed under this case in order to ensure that MGMT performance does not degrade with increasing time-step ratios, Figure 5-7. Sub-domains, $\Omega 1$ and $\Omega 2$, employ implicit time integration algorithms and the total number of interface (mortar) nodes in this case is 5 (🟥).

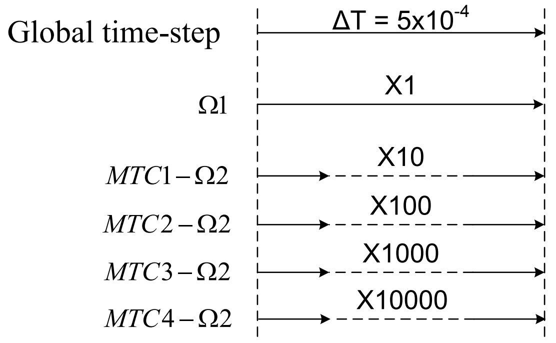


Figure 5-7: Case 4 (MTC) time-step ratios

5.1.5 Case 5 – Multiple Grid Coupling with Implicit Time Integration (MGC)

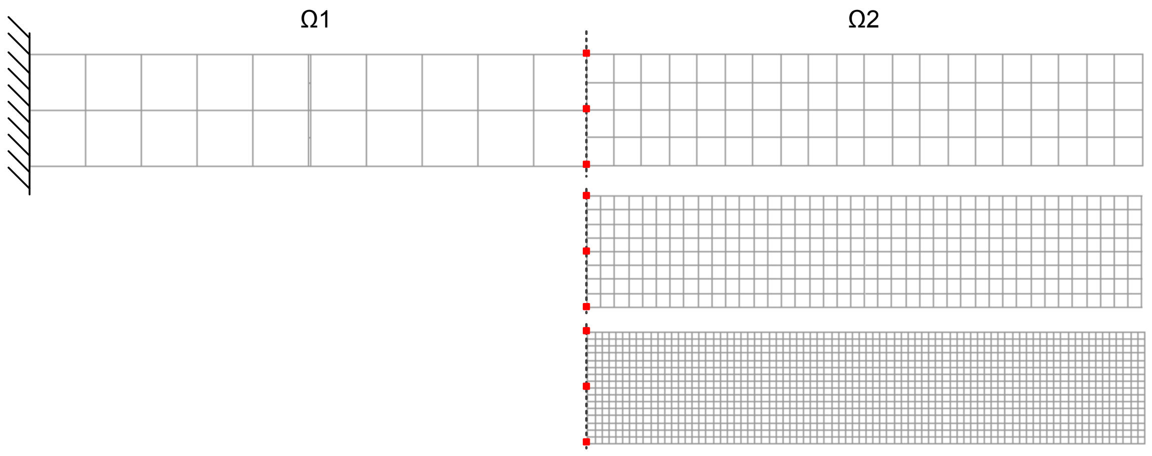


Figure 5-8: Case 5 (MGC) sub-domain grids

Table 5-5: Case 5 (MGC) simulation parameters

Sub-domain	Newmark parameters	Time-step (Δt)	
		Example 1	Example 2
Ω1 and Ω2	β=0.25, γ=0.5 (Implicit)	1x10^{-3}	1x10^{-4}

		Grid Spacing (H)	
		Example 1 and Example 2	
Case ID	Interface DOF (λ)	Ω1	Ω2
MGC1	6	0.5	0.25
MGC2	6	0.5	0.125
MGC3	6	0.5	0.0625

Under Case 5, we will evaluate the effects of coupling 2 sub-domains with non-conforming grids and uniform time-steps, as shown in Figure 5-8 and Table 5-5. The goal is to quantify the cumulative interface dissipation or accumulation across sub-domain interfaces caused as a result of introducing Lagrange Multipliers with increasing number of projected constraints over mortar interface on $\Omega2$. Sub-domains $\Omega1$ and $\Omega2$ employ implicit time integration algorithms and maintain a time-step ratio of $\xi = 1$.

5.2 UGUT Convergence

When modeling a problem using FEM, convergence is an essential attribute to instill confidence in FEM results from the standpoint of mathematics. The word convergence is used because the FEM output is 'expected' to converge to a single correct solution. In order to check the convergence, at least two solutions to the same problem are required. The solution from a particular FE approach is then checked with one that employs an improved approach. If the solution from an improved approach is dramatically different from the primal approach, then the solution is not converged. However, if the solution does not change much, then the solution is considered converged. The primary motivation for convergence is that the FEM solution should approach the analytical solution of the equivalent mathematical model.

Convergence can be tested differently depending on the solution technique in use. Two available methods are P-Method and H-Method. P-Method utilizes large elements with improved complex shape functions. In this approach, polynomial degree of the implemented shape functions is increased in order to obtain an improved solution. This method does not mandate grid refinement. H-Method, on other hand, uses simple shape functions and many small elements to improve the overall quality of the solution. Accordingly, H-Method or grid refinements essentially translate to increasing the number discretization points over a domain in order to obtain a better/more accurate solution. These discretized points are represented by domain degrees of freedom (DOF) in FEM and mathematically represent the total number of equations that a FE algorithm solves. It goes without saying, larger the number of equations – larger is the computational cost. Therefore, it is important to realize that when a domain is discretized with larger DOF, the gain in accuracy is inherently accompanied by greater computational costs and vice versa.

Since FEAPI implements simple shape functions (linear for 4-node quadrilateral elements), we employ H-Method to establish a discretization that yields converged results. Accordingly, grid spacing parameter (H) and corresponding time-steps are successively refined to obtain convergence in reference UGUT cases.

For Example 1 (transverse loading), we measure the vertical displacement of a neutral layer node on the free end (x = 10), Figure 5-9, and compare the maximum displacements and relative changes with respect to other UGUT cases in Figure 5-10. It is observed that the difference sharply falls to 0.04 (UGUT4 v/s UGUT3) and 0.043 (UGUT5 v/s UGUT3) ensuring convergence discretizations for Example 1.

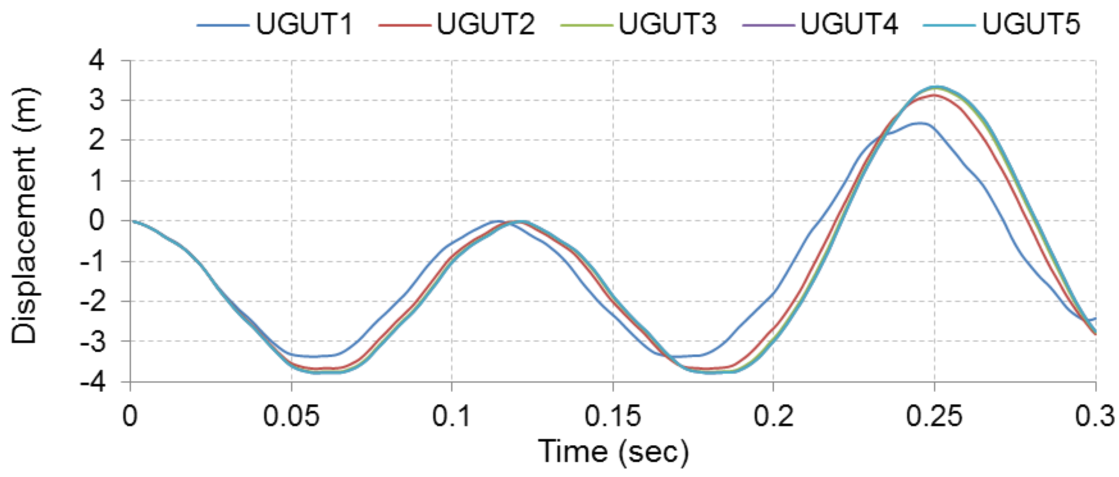


Figure 5-9: Example 1 – Vertical displacement at free end (x = 10m)

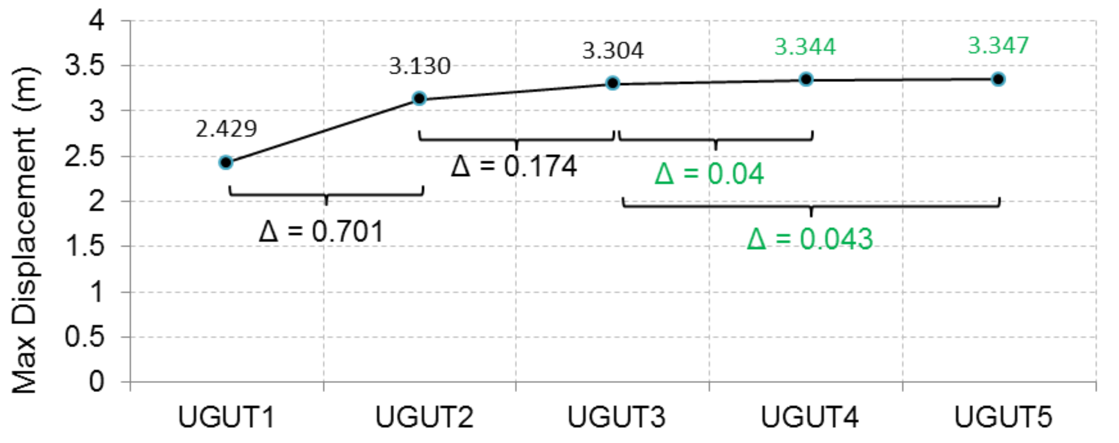


Figure 5-10: Example 1 – Convergence of maximum vertical displacement at free end (x = 10m)

In Example 2 (longitudinal loading), we measure the horizontal displacement of a neutral layer node on the free end (x = 10), Figure 5-11, and compare the maximum displacements (and relative changes) with respect to other UGUT cases in Figure 5-12. It is observed that the difference sharply falls to 1x10^{-5} for both (UGUT4 v/s UGUT3) and (UGUT5 v/s UGUT3) ensuring convergence discretizations for Example 2.

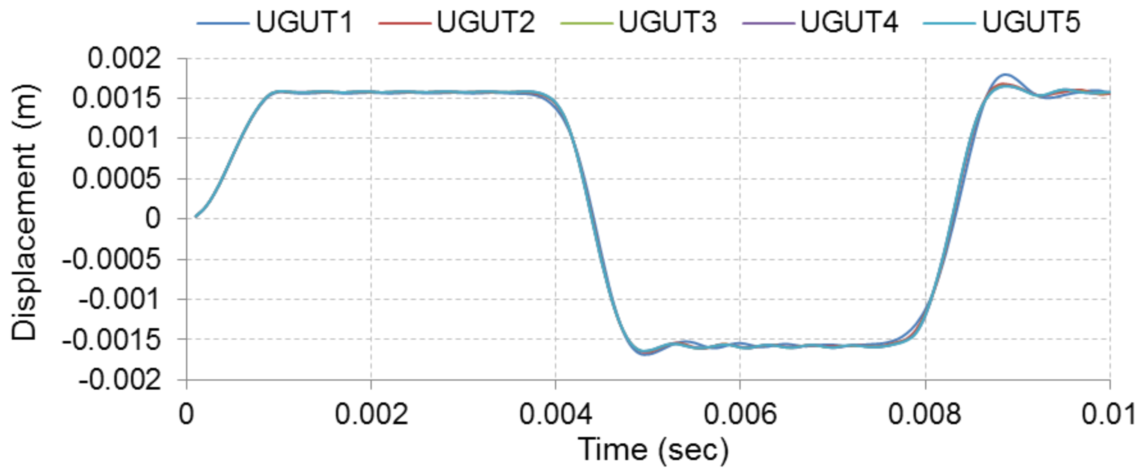


Figure 5-11: Example 2 – Horizontal displacement at free end (x = 10m)

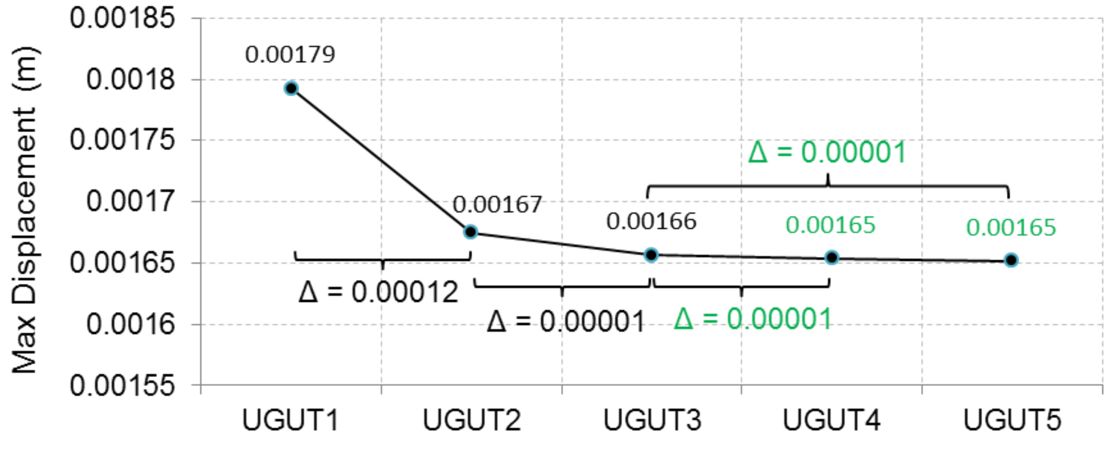


Figure 5-12: Example 2 – Convergence of maximum horizontal displacement at free end (x = 10m)

Table 5-6: RMSE and NRMSE between respective UGUT cases

		UGUT1	UGUT2	UGUT3	UGUT4	UGUT5
Example 1	UGUT1	-	**0.5667 (8.3%)**	0.7314 (10.36%)	0.7715 (10.83%)	0.7740 (10.86%)
	UGUT2	-	-	**0.1119 (1.59%)**	0.1146 (1.59%)	0.1414 (1.97%)
	UGUT3	-	-	-	**0.0247 (0.34%)**	0.0258 (0.36%)
	UGUT4	-	-	-	-	**0.0016 (0.002%)**
	UGUT5	-	-	-	**0.0016 (0.002%)**	-
Example 2	UGUT1	-	**3.97×10^{-5} (1.19%)**	4.28×10^{-5} (1.28%)	4.57×10^{-5} (1.38%)	5.59×10^{-5} (1.59%)
	UGUT2	-	-	**3.57×10^{-6} (0.106%)**	7.00×10^{-6} (0.212%)	7.00×10^{-6} (0.212%)
	UGUT3	-	-	-	**3.77×10^{-6} (0.106%)**	3.58×10^{-6} (0.106%)
	UGUT4	-	-	-	-	**2.91×10^{-6} (0.088%)**
	UGUT5	-	-	-	**2.91×10^{-6} (0.088%)**	-

Table 5-6 lists the relative errors between respective UGUT cases. Case pairs that were used to establish convergence in Figure 5-10 and Figure 5-12 are highlighted in **Green**. We can see that there is a similar trend/drop in RMSE and NRMSE as UGUT discretizations are refined, ensuring convergence over entire spectrum of time; in contrast to the convergence of maximum value at one particular time instant in Figure 5-10 and Figure 5-12.

Finally, we look at the total number of equations (unconstrained DOF) used in each UGUT case and corresponding CPU solution times required to solve these equations.

Table 5-7: Total number of equations and CPU solution time

	Total Number of Equations	Solution Time (sec)	
		Example 1	Example 2
UGUT1 (I)	120	0.23	0.07
UGUT2 (I)	400	2.01	0.65
UGUT3 (I)	1440	22.23	7.36
UGUT4 (I)	5440	320.92 (~5 min)	107.53 (~ 2 min)
UGUT5 (E)	5440	24822.27 (~ 7 hr)	830.14 (~13 min)

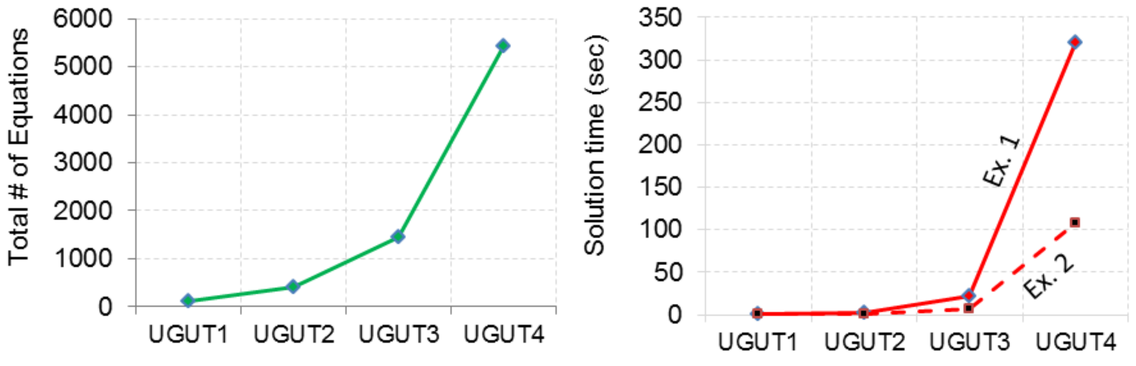


Figure 5-13: Total number of equations and CPU solution time for Implicit UGUT cases

For the example problems under consideration, results show that converged solution can be obtained via grid refinement (from discretizations inherited from UGUT4/5), however at the cost of larger CPU times. Since converged cases, UGUT4 (Implicit) and UGUT5 (Explicit), provide the best available approximation of measured variables, these cases are used as baseline/reference solutions when evaluating MGMT solutions.

5.3 Stability Analysis

Under our first performance evaluation criteria, we analyze the numerical stability of MGMT Method by looking at the global energy trends and augmented interface energy contributions from component sub-domains.

For a non-homogeneous linear structural system without damping, we know that Internal Energy (IE) is equal to the external work performed on the system. In UGUT simulations, IE is obtained as a sum of Kinetic Energy (KE) and Stiffness or Potential Energy (SE), whereas in MGMT simulations total IE is obtained by summing KE and SE contributions from all component sub-domains. Furthermore, interface energy produced as a result of introducing Lagrange Multipliers is augmented into IE, yielding global/total IE.

Since MGMT Method involves coupling of non-conforming grids as well as concurrent time-integration of sub-domains with distinct time-steps, it is critical to ensure that the global aspect of the simulation, i.e. energy balance, is established in order to evaluate performance of MGMT formulation and implementation. We have already established in Section 3.5 that MGMT coupling is numerically stable if and only if the total interface energy accumulates to zero. Consequently, IE from UGUT and MGMT simulations must be in complete conformance with each other.

Accordingly, in following sections we analyze global energy balance, augmented interface energy contributions and continuity of velocity across component sub-domain interfaces in order to provide a comprehensive evaluation of numerical stability of the proposed MGMT Method.

5.3.1 Global Energy Balance

Figure 5-14 shows the global energies from Example 1 (transverse vibration) for converged cases – UGUT4, UGUT5 and MGMT cases – MGMT1, MGMT2, MTC4 and MGC3. RMSE and NRMSE for these plots are listed in Figure 5-8.

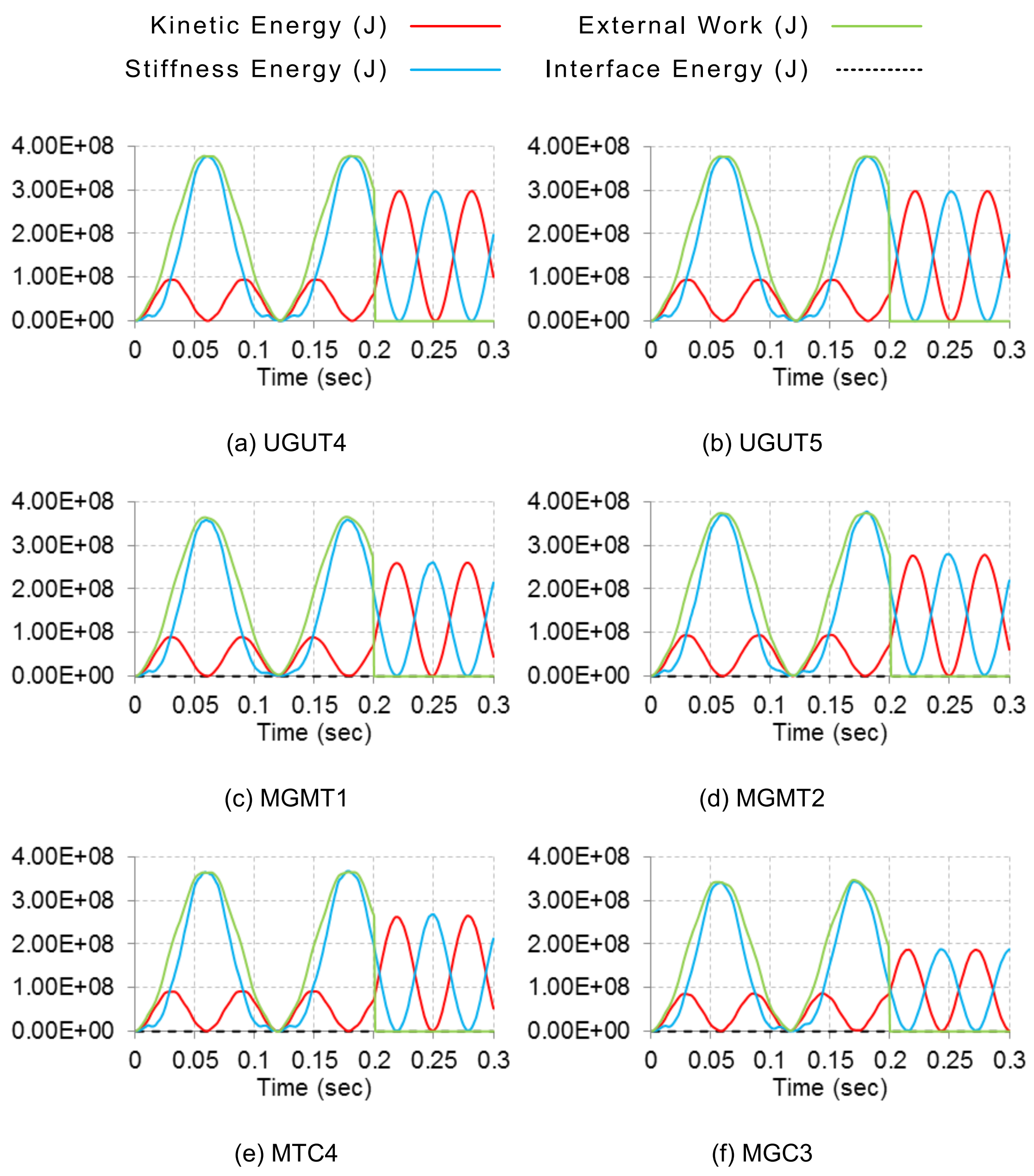
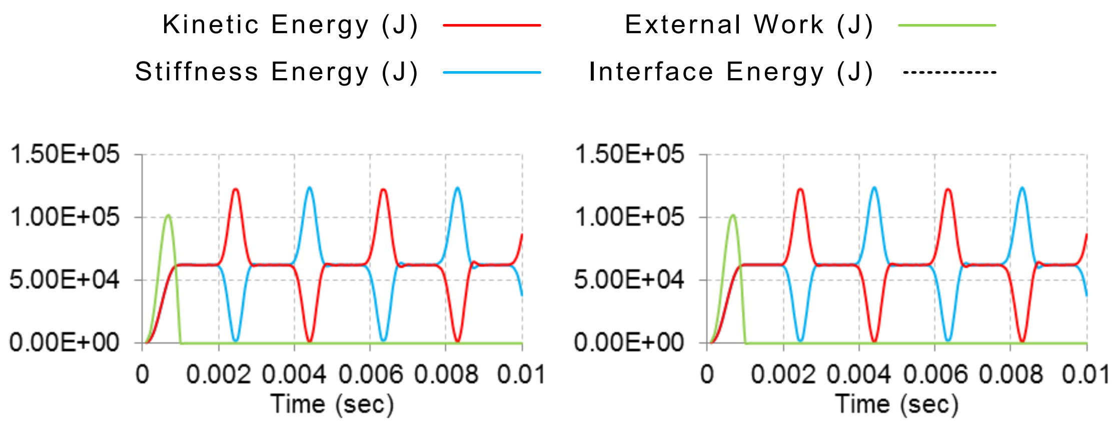


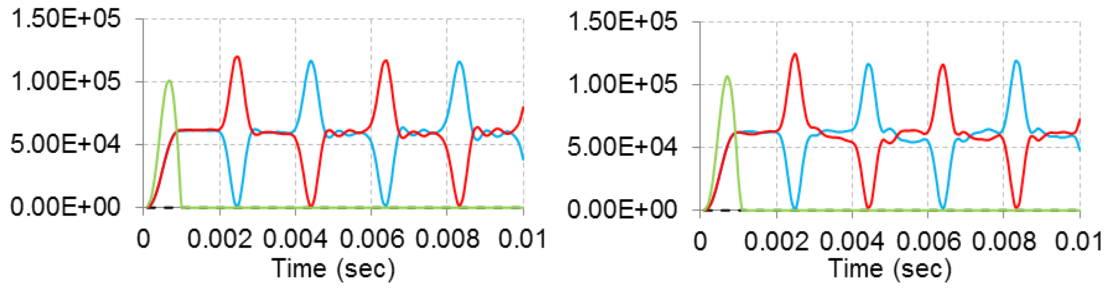
Figure 5-14: Example 1 – Global energies

Figure 5-15 shows the global energies from Example 2 (longitudinal vibration) for converged cases – UGUT4, UGUT5 and MGMT cases – MGMT1, MGMT2, MTC4 and MGC3. RMSE and NRMSE for these plots are listed in Table 5-9.



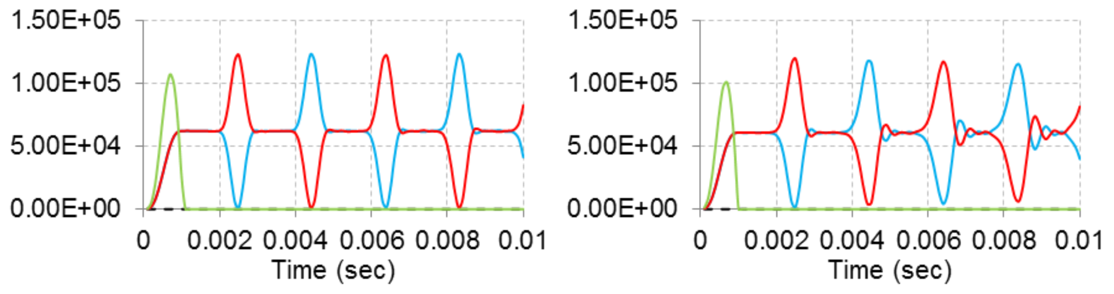
(a) UGUT4

(b) UGUT5



(c) MGMT1

(d) MGMT2



(e) MTC4

(f) MGC3

Figure 5-15: Example 2 – Global energies

Table 5-8: Example 1 – RMSE and NRMSE (%). Variable = Global energies

	Kinetic Energy		Stiffness Energy	
	UGUT4	UGUT5	UGUT4	UGUT5
UGUT4	-	3.94×10^5 (0.13%)	-	4.34×10^5 (0.11%)
UGUT5	3.94×10^5 (0.13%)	-	4.34×10^5 (0.11%)	-
MGMT1	2.37×10^7 (7.98%)	2.37×10^7 (7.97%)	2.54×10^7 (6.74%)	2.53×10^7 (6.72%)
MGMT2	1.55×10^7 (5.24%)	1.55×10^7 (5.22%)	1.69×10^7 (4.49%)	1.68×10^7 (4.46%)
MTC4	2.06×10^7 (6.95%)	2.07×10^7 (6.95%)	2.15×10^7 (5.71%)	2.14×10^7 (5.69%)
MGC3	6.37×10^7 (21.47%)	6.38×10^7 (21.44%)	6.70×10^7 (17.79%)	6.70×10^7 (17.79%)

Table 5-9: Example 2 – RMSE and NRMSE (%). Variable = Global energies

	Kinetic Energy		Stiffness Energy	
	UGUT4	UGUT5	UGUT4	UGUT5
UGUT4	-	102.16 (0.08%)	-	204.43 (0.16%)
UGUT5	102.16 (0.08%)	-	204.43 (0.16%)	-
MGMT1	3120.46 (2.59%)	3257.84 (2.63%)	3237.35 (2.66%)	3308.00 (2.71%)
MGMT2	4844.60 (3.91%)	4884.33 (3.95%)	5099.79 (4.19%)	5188.40 (4.26%)
MTC4	2258.88 (1.82%)	2314.85 (1.87%)	2227.30 (1.83%)	2341.72 (1.92%)
MGC3	4657.38 (3.76%)	4727.48 (3.82%)	4604.52 (3.78%)	4719.54 (3.88%)

Internal energy (kinetic + stiffness + interface energy) must equal the total amount of work performed on the system. Accordingly, RMSE and NRMSE errors between these curves are listed in Table 5-10. These errors are computed only for the time in which external work is non-zero, since internal energy remains constant, as residual energy, in the absence of damping. Although the errors in Table 5-8 and Table 5-9 are comparatively high, global energy balance (internal energy = external work) is verified by relatively small errors (less than 1%) in Table 5-10.

Table 5-10: RMSE and NRMSE (%). Variable = Internal energy v/s External work

	Example 1	Example 2
UGUT4	0.0	0.0
UGUT5	0.0	0.0
MGMT1	3.22×10^6 (0.88%)	1.07×10^4 (0.13%)
MGMT2	2.12×10^6 (0.56%)	8.85×10^4 (0.12%)
MTC4	3.00×10^6 (0.82%)	8.86×10^4 (0.12%)
MGC3	2.65×10^6 (0.07%)	1.11×10^4 (0.14%)

Error rankings, in ascending order, for kinetic energy and stiffness energy in Example 1 and Example 2 are {MGMT$\underline{2}$ < MTC$\underline{4}$ < MGMT$\underline{1}$ < MGC$\underline{3}$} ≡ {2413} and {MTC$\underline{4}$ < MGMT$\underline{1}$ < MGC$\underline{3}$ < MGMT$\underline{2}$} ≡ {4132} respectively whereas those for internal energy v/s external work are {3241} and {2413}.

5.3.2 Augmented Interface Energy

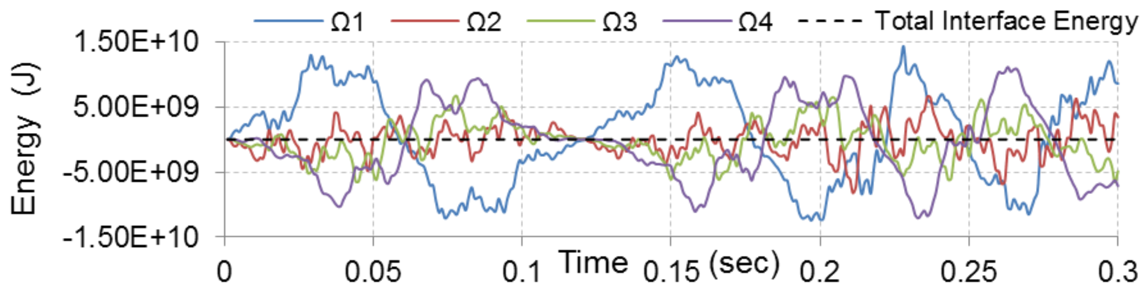
Another way to verify numerical stability is to ensure that augmented interface energies from component sub-domains accumulate to zero, as indicated by Eq. (3.9) and Eq. (3.83). Interface energy is representative of total work performed by sub-domain interface reactions (λ) while communicating interactions at intermediate time-steps. These forces enable coupling of multiple-grids and multiple time-scales and consequently also affect the numerical stability of component sub-domains. Hence, if augmented interface energies from all sub-domains accumulate to zero, one can derive that energy balance is established between at component sub-domain levels for a particular time-step. Furthermore, it ensures efficient and accurate multiscale coupling.

Augmented interface energy, as derived in Section 3.2, is computed as follows:

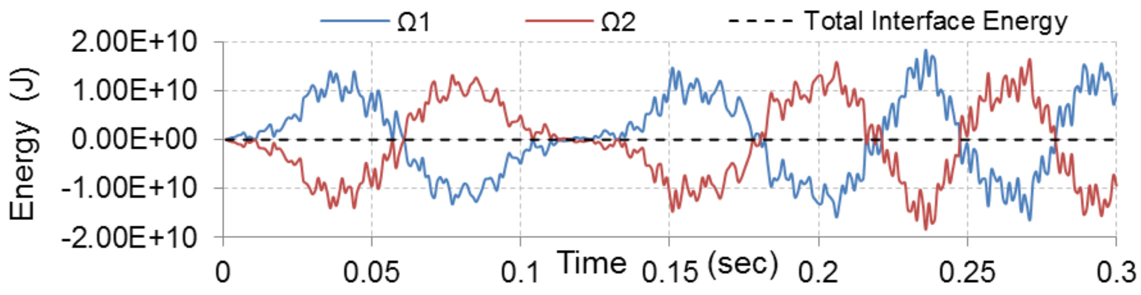
$$\text{Total Interface Energy } = \sum_{i=1}^{S} \dot{U}^{i^T} L^{i^T} \lambda \tag{5.3}$$

Where, S represents total number of sub-domains, λ is the total number of DOF discretized using M-FEM over each non-mortar sub-domain interface and L^i is the multi-constraint operator that is used to project sub-domain interface velocities, as represented by $\dot{U}$.

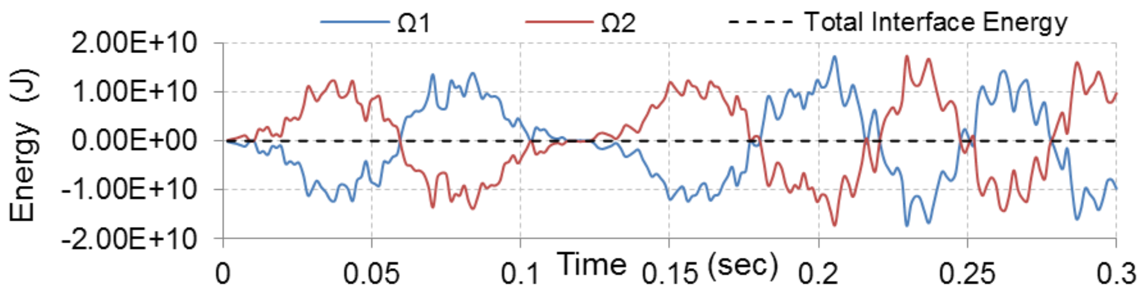
Subsequent plots, Figure 5-16 (Example 1) and Figure 5-17 (Example 2), show the evolution of total interface energy relative to energy contributions from component sub-domain interfaces. These plots clearly indicate that sub-domain interface energies annihilate each other, verifying stability at component sub-domain levels and accurate MGMT coupling.



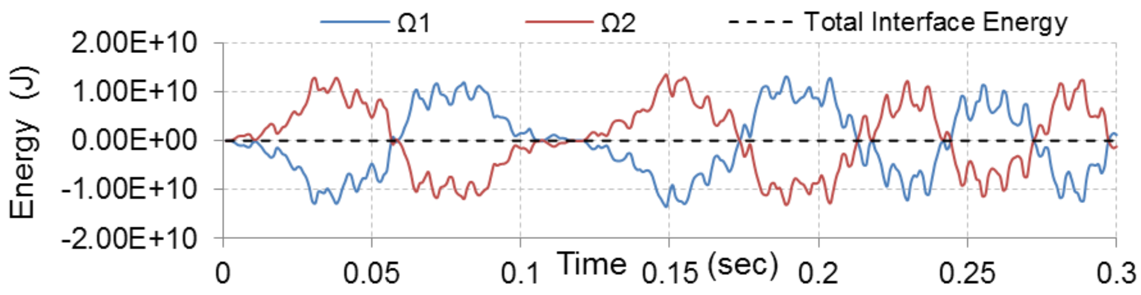
(a) MGMT1



(b) MGMT2

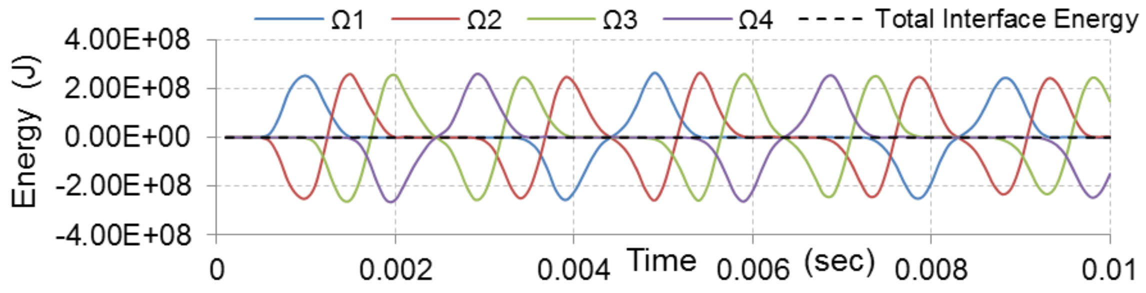


(c) MTC4

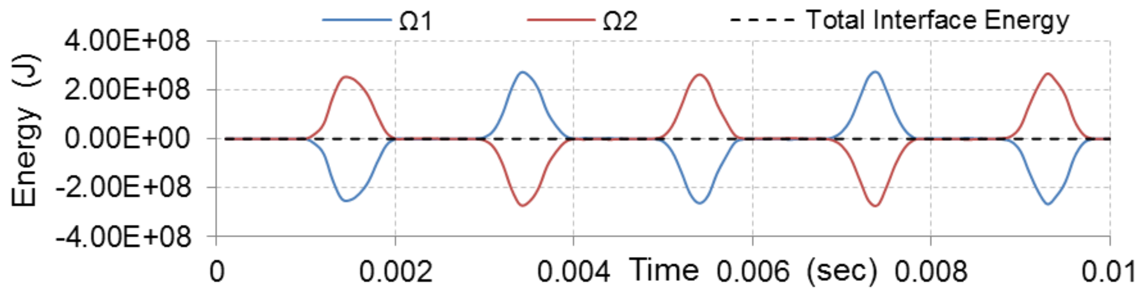


(d) MGC3

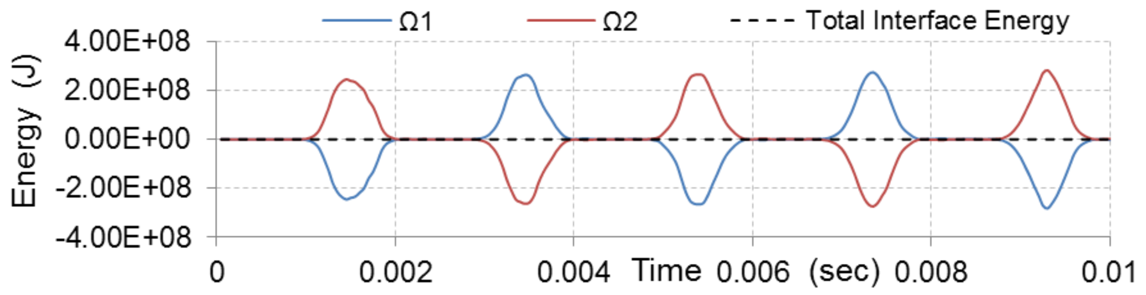
Figure 5-16: Example 1 – Augmented interface energies from component sub-domains



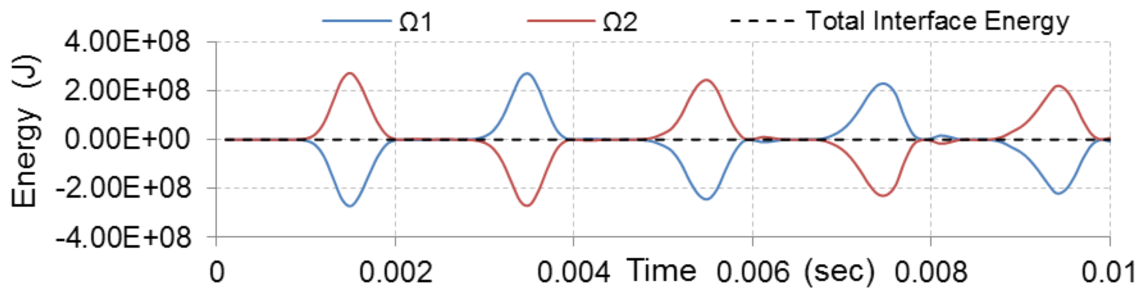
(a) MGMT1



(b) MGMT2



(c) MTC4



(d) MGC3

Figure 5-17: Example 2 – Augmented interface energies from component sub-domains

Total interface energy seems infinitesimally small relative to component sub-domain energy contributions. However, subsequent figures show that there is a small variance in interface energy over entire simulation time. This variance is partly due to machine tolerance/round-off errors and can be reduced by refining the increments used in numerical integration of interface connectivity constraints, Eq. (2.73) and (2.74), See also Figure 3-8.

Table 5-11 lists the RSME in total interface energy indicating its mean variance about zero, since corresponding UGUT interface energy is identically zero. These errors represent average accumulation or dissipation of total interface energy over entire simulation time. Table 5-11 also lists the total number of global integration loops, representing the frequency of recorded data. It should be noted that for every global integration loop, sub-domains with $\xi > 1$ are integrated ξ times before synchronizing with global time-step.

Table 5-11: Mean variance in augmented (total) interface energy contributions

	Example 1	Example 2
MGMT1	$9.48\text{x}10^{-5}$ (@ 300)	$4.99\text{x}10^{-8}$ (@ 100)
MGMT2	$2.08\text{x}10^{-5}$ (@ 600)	$1.91\text{x}10^{-8}$ (@ 200)
MTC1	$1.71\text{x}10^{-5}$ (@ 600)	$1.99\text{x}10^{-8}$ (@ 200)
MTC2	$1.33\text{x}10^{-5}$ (@ 600)	$2.12\text{x}10^{-8}$ (@ 200)
MTC3	$1.88\text{x}10^{-5}$ (@ 600)	$1.95\text{x}10^{-8}$ (@ 200)
MTC4	$1.48\text{x}10^{-5}$ (@ 600)	$1.85\text{x}10^{-8}$ (@ 200)
MGC1	$1.15\text{x}10^{-5}$ (@ 300)	$1.85\text{x}10^{-8}$ (@ 100)
MGC2	$1.31\text{x}10^{-5}$ (@ 300)	$2.09\text{x}10^{-8}$ (@ 100)
MGC3	$1.17\text{x}10^{-5}$ (@ 300)	$2.42\text{x}10^{-8}$ (@ 100)

These errors are orders of magnitude smaller than the global energy scales, hence their contribution is considered negligible. Error rankings for total interface energy, Example 1 and Example 2, are {3421} and {4231}.

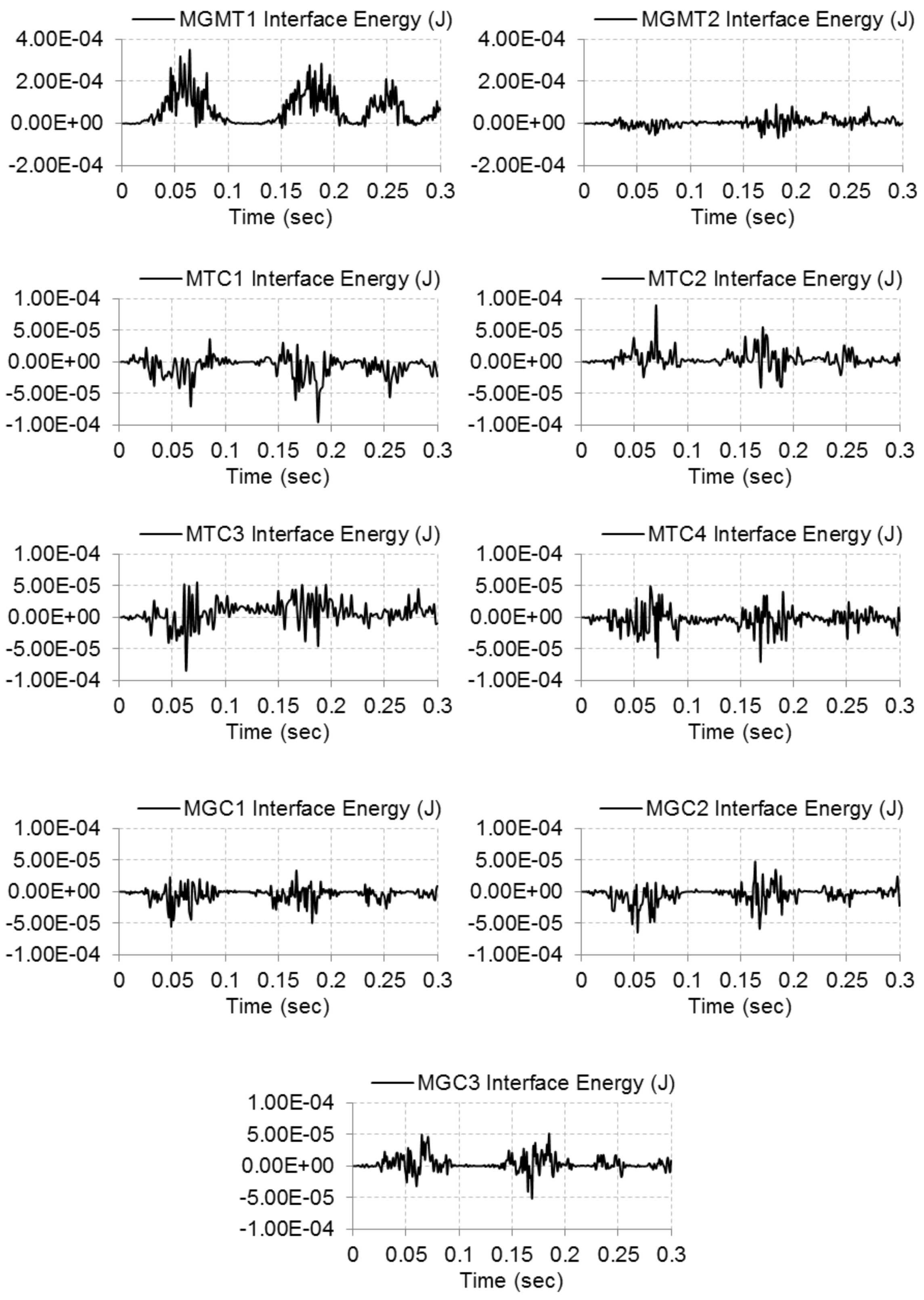


Figure 5-18: Example 1 – Augmented (total) interface energy

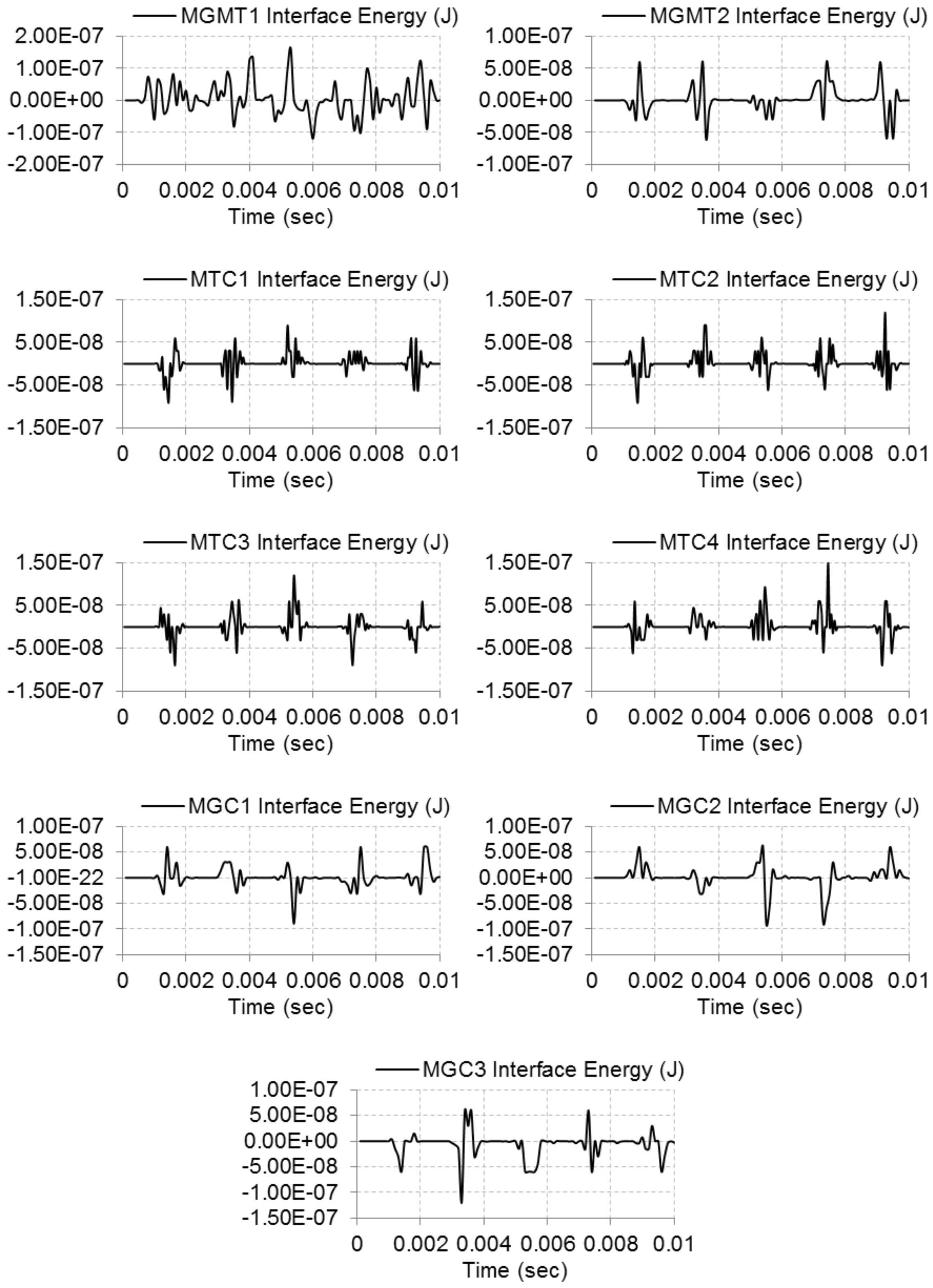


Figure 5-19: Example 2 – Augmented (total) interface energy

5.3.3 Interface Continuity

Another consequence of the global stability requirement is the continuity of velocities at sub-domain interfaces, as enforced by Eq. (3.11). In this section, we compare conformance between interface variables (displacement, velocity and acceleration) at x = 5, i.e. half way across the length of the beam, as obtained from adjoining sub-domains. Figure 5-20, compares U_y, $\dot{U}_y$ and $\ddot{U}_y$ for Example 1, as obtained from MGMT1 sub-domains $\Omega 2$ and $\Omega 3$. The grid ratio and time-step ratio between these sub-domains is 2 and a total of 10 mortar DOF (λ) are used to communicate interactions at the dividing interface.

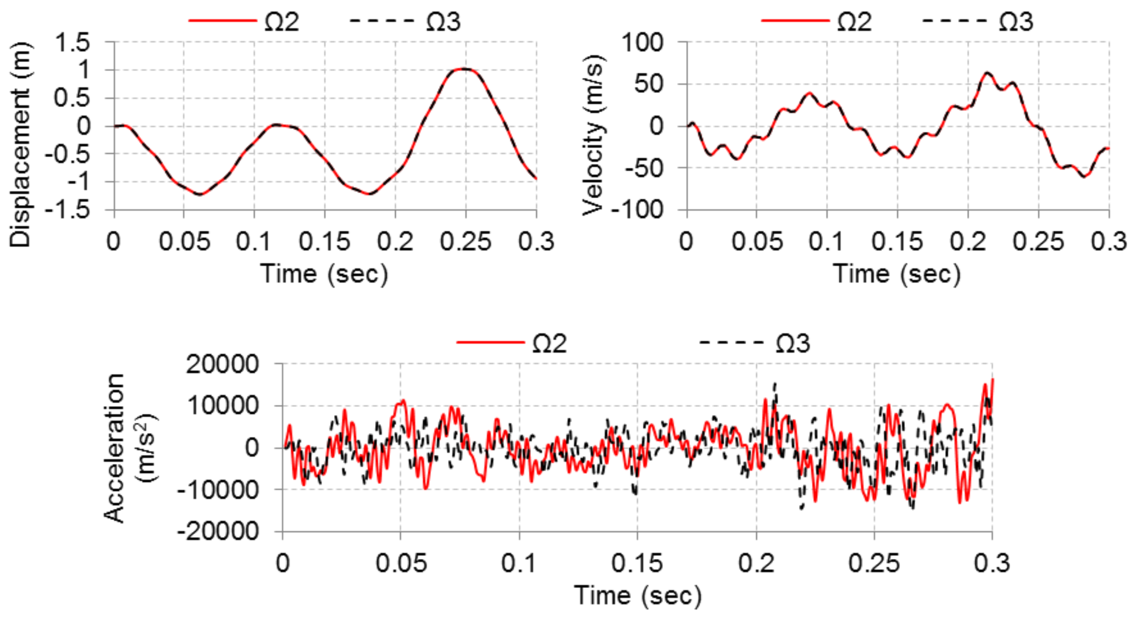


Figure 5-20: Example 1 – (MGMT1) Continuity of interface variables

Table 5-12: Example 1 – (MGMT1) RMSE and NRMSE (%). Variable = Interface displacement, velocity and accelerations

		Example 1
MGMT1	U^2 v/s U^3	1.31×10^{-3} (0.058%)
	$\dot{U}^2$ v/s $\dot{U}^3$	4.88×10^{-2} (0.039%)
	$\ddot{U}^2$ v/s $\ddot{U}^3$	6871.41 (23.27%)

Figure 5-21 compares U_y, $\dot{U}_y$ and $\ddot{U}_y$ for Example 1, as obtained from MGMT2 sub-domains $\Omega1$ and $\Omega2$. The grid ratio and time-step ratio between these sub-domains is 4 and 500 respectively with a total of 10 mortar DOF (λ) at the sub-domain interface. It should be noted that $\Omega1$ and $\Omega2$ in this case are coupled via Implicit and Explicit time integration algorithms.

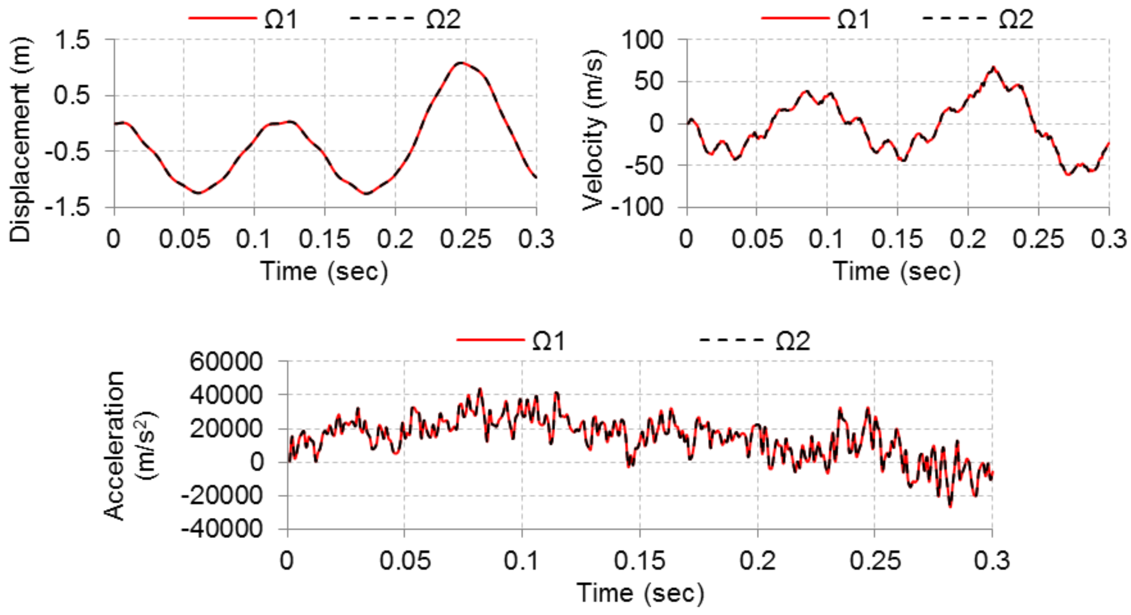


Figure 5-21: Example 1 – (MGMT2) Continuity of interface variables

Table 5-13: Example 1 – (MGMT2) RMSE and NRMSE (%). Variable = Interface displacement, velocity and accelerations

		Example 1
MGMT2	U^1 v/s U^2	$2.03\text{x}10^{-3}$ (0.087%)
	$\dot{U}^1$ v/s $\dot{U}^2$	0.2521 (0.195%)
	$\ddot{U}^1$ v/s $\ddot{U}^2$	1.4030 (0.002%)

Figure 5-22 compares U_x, $\dot{U}_x$ and $\ddot{U}_x$ for Example 2, as obtained from MTC4 sub-domains $\Omega 1$ and $\Omega 2$. The grid ratio in this case is 1 (conforming interface), however the time-step ratio between these sub-domains is 10,000. A total of 10 mortar DOF (λ) are used at the sub-domain interfaces to enable multiple time-scale coupling.

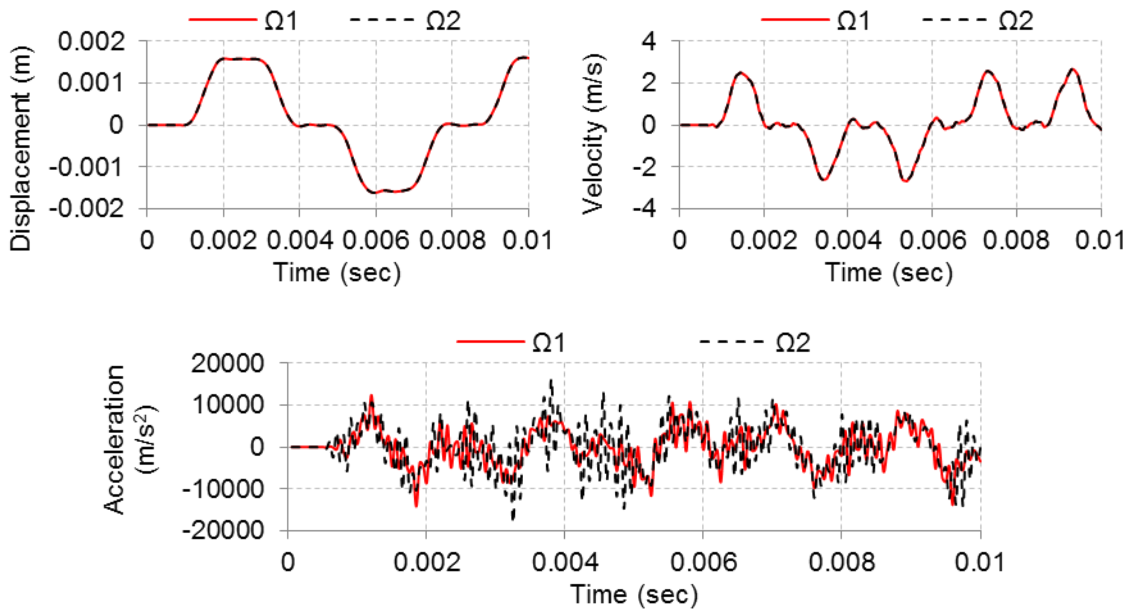


Figure 5-22: Example 2 – (MTC4) Continuity of interface variables

Table 5-14: Example 2 – (MTC4) RMSE and NRMSE (%). Variable = Interface displacement, velocity and accelerations

		Example 2
MTC4	U^1 v/s U^2	$1.53\text{x}10^{-6}$ (0.004%)
	$\dot{U}^1$ v/s $\dot{U}^2$	0.0 (0.0%)
	$\ddot{U}^1$ v/s $\ddot{U}^2$	5988.58 (22.52%)

Figure 5-23 compares U_x, $\dot{U}_x$ and $\ddot{U}_x$ for Example 2, as obtained from MGC3 sub-domains $\Omega 1$ and $\Omega 2$. The time-step ratio in this case is 1; however the grid-step ratio between these sub-domains is 8. A total of 10 mortar DOF (λ) are used at the sub-domain interfaces to enable multiple grid coupling.

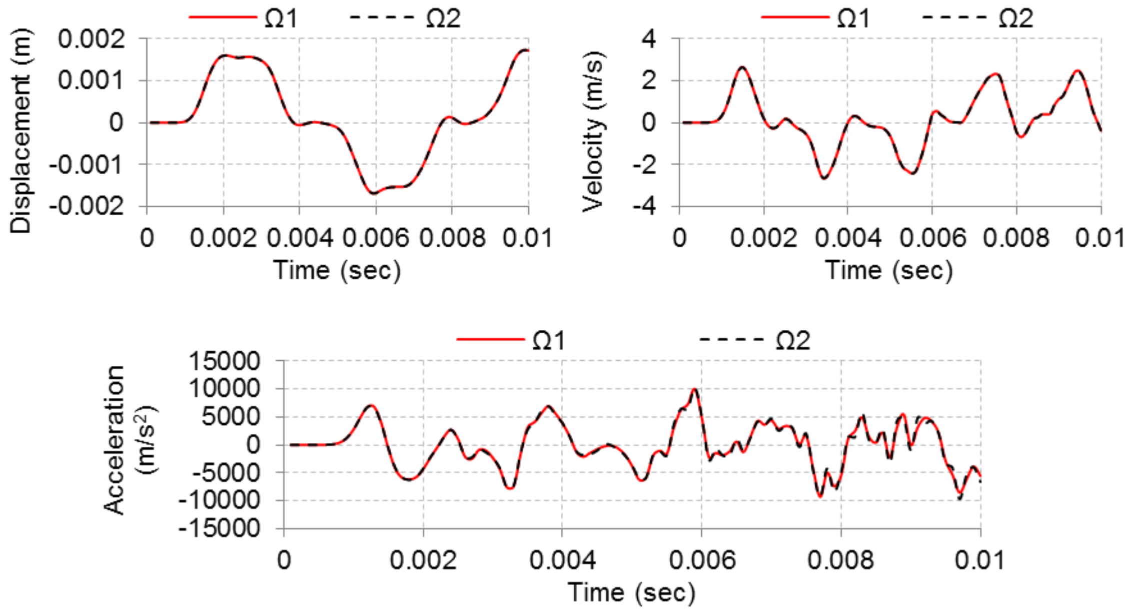


Figure 5-23: Example 2 – (MGC3) Continuity of interface variables

Table 5-15: Example 2 – (MGC3) RMSE and NRMSE (%). Variable = Interface displacement, velocity and accelerations

		Example 2
MGC3	U^1 v/s U^2	1.79x10^{-6} (0.005%)
	$\dot{U}^1$ v/s $\dot{U}^2$	1.43x10^{-2} (0.269%)
	$\ddot{U}^1$ v/s $\ddot{U}^2$	466.58 (2.418%)

For the results discussed above, NRMSE for displacement and velocity are well below 1%, whereas errors in accelerations (in certain cases) are significantly higher. However, the key variable under consideration here is velocity, since we enforced continuity of velocities across sub-domain interfaces using Eq. (3.11) in order to ensure global and local stability.

For Example 2 (MTC4) we notice that conforming grids with a time-step ratio of 10,000 yield continuous velocities with 0% NRMSE, Table 5-14. We have already seen that multi-constraint cooperator in this case is a Boolean projection operator $(L = B)$. For non-conforming grids, however, $L = BP$ where P represents interface connectivity constraints modeled using M-FEM. This indicates that any positive error in velocity continuity (for cases with non-conforming interfaces) can be reduced by efficient numerical integration of interface constraints, as discussed earlier in Section 5.3.2.

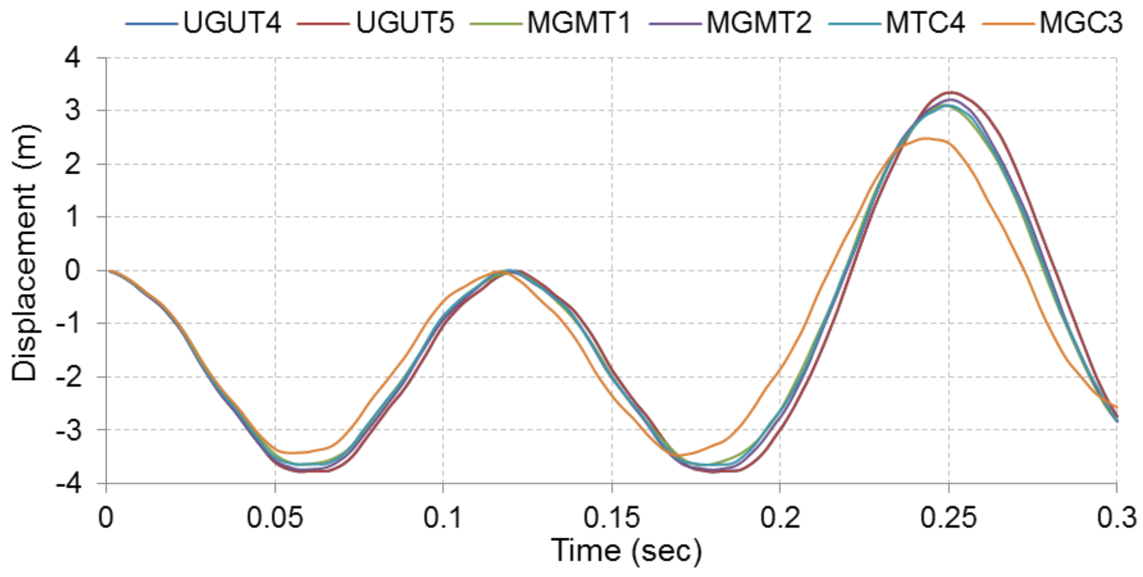
Since aforementioned results exhibit good conformance in continuity of interface velocity, we establish that Eq. (3.11) is efficiently implemented, further verifying MGMT stability.

Velocity error rankings for Example 1 and Example 2 are {12} and {43} respectively.

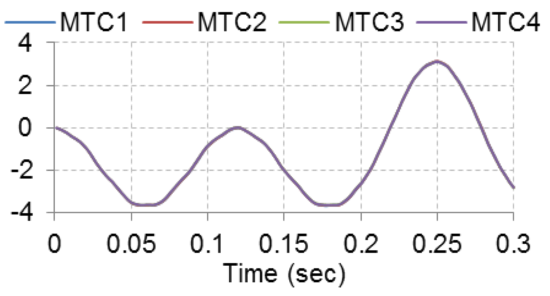
5.4 Evaluation of Numerical Accuracy

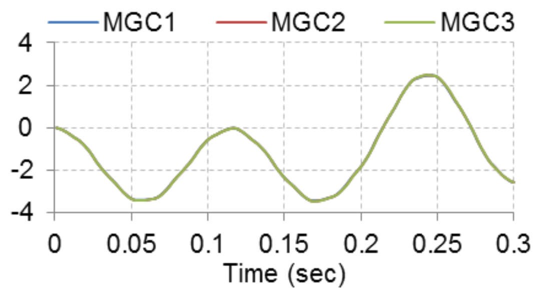
5.4.1 Example 1: Transverse Vibrations

As our first evaluation criteria for numerical accuracy in Example 1, we look at the time evolution of the beam tip deflection. Figure 5-24 shows the vertical component of tip deflection for UGUT4, UGUT5, MGMT1, MGMT2, MTC1, MTC2, MTC3, MTC4, MGC1, MGC2 and MGC3. Corresponding errors and total number of equations (unconstrained DOF) are listed in Table 5-16.



(a) UGUT4, UGUT5, MGMT1, MGMT2, MTC4, MGC3





(b) MTC1, MTC2, MTC3, MTC4

(c) MGC1, MGC2, MGC3

Figure 5-24: Example 1 – Comparison of vertical displacement at free end (x = 10m)

Table 5-16: Example 1 – RMSE and NRMSE (%). Variable = displacement at free end (x = 10m)

	UGUT4	UGUT5	Number of Equations
UGUT4	-	3.09×10^{-3} (0.043%)	5440
UGUT5	3.09×10^{-3} (0.043%)	-	5440
MGMT1	0.154 (2.24%)	0.1572 (2.28%)	1884
MGMT2	0.020 (0.29%)	0.0246 (0.34%)	2954
MTC1	0.2619 (3.83%)	0.2647 (3.89%)	410
MTC2	0.2672 (3.84%)	0.2727 (3.89%)	410
MTC3	0.2748 (3.84%)	0.2796 (3.90%)	410
MTC4	0.2816 (3.85%)	0.2686 (3.90%)	410
MGC1	0.362 (5.16%)	0.3668 (5.20%)	270
MGC2	0.375 (5.26%)	0.3736 (5.31%)	798
MGC3	0.3839 (5.37%)	0.3832 (5.42%)	2814

In Section 5.2 we discussed how H-Method, or grid refinement, can be used to achieve numerical convergence when using FEM to obtain the solution to a particular problem. We also established that UGUT4 and UGUT5 discretization, each with 5440 DOF, yield converged solutions to Example 1 problem. Accordingly, Table 5-16 also lists the total number of equations (unconstrained DOF) used to solve a particular MGMT case.

Errors are relatively small for most cases. MGMT2 has the smallest error since it hosts maximum (2954) DOF in comparison with other cases; whereas MGC3 (2814 DOF) has maximum error resulting in a conclusion that higher grid ratios with fewer interface coupling DOF may result in poor connections and hence higher errors. Additionally, if we look at MGMT1, it also hosts higher grid ratios, but the error in this case is almost half of MGC cases, indicating that one may use higher grid ratios, however, these should be modeled with a gradual grid refinement and relatively larger interface coupling DOF. This interpretation is further verified by noticing the gradual drop in MGC errors. That is, encountered error reduces as the grid ratio between $\Omega 1$ and $\Omega 2$ reduces from 8 $\rightarrow$ 4 $\rightarrow$ 2 for MGC3 $\rightarrow$ MGC2 $\rightarrow$ MGC1. Similar trend, gradual drop in error, is also noticed for respective MTC cases suggesting that accumulated error increases as the time-step ratio between component sub-domains increases. However; the errors in MTC (maximum of 3.9% for $\xi_{\max} = 10,000$) is relatively smaller than MGC (5.42% for a grid ratio of 8) clearly indicating the necessity, influence and the importance of efficient multiple grid coupling.

We also notice a significant effect on numerical accuracy due to multiple time-scale coupling by comparing equivalently discretized UGUT2 and MTC cases, as follows:

Table 5-17: Example 1 – Comparison between UGUT2 and MTC cases

	UGUT4	UGUT5	Number of Equations
UGUT2	0.1146 (1.59%)	0.1414 (1.97%)	400
MTC1	0.2619 (3.83%)	0.2647 (3.89%)	410
MTC2	0.2672 (3.84%)	0.2727 (3.89%)	410
MTC3	0.2748 (3.84%)	0.2796 (3.90%)	410
MTC4	0.2816 (3.85%)	0.2686 (3.90%)	410

Augmented effect of multiple grids and multiple time-scales can be observed by comparing UGUT3 and MGMT1 as follows:

Table 5-18: Example 1 – Comparison between UGUT3 and MGMT1

	UGUT4	UGUT5	Number of Equations
UGUT3	0.0247 (0.34%)	0.0258 (0.36%)	1440
MGMT1	0.154 (2.24%)	0.1572 (2.28%)	1884

Errors trends in the measurement of displacement at free end for Example 1, Table 5-16, are {2143}.

Figure 5-25 and Table 5-19 show the comparison of Sigma-xx for a cross-section at x = 1m from the fixed end (at t = 0.05s). Most cases have errors less than 0.5% indicating good conformance. Highest error occurs in MGC3 (~1%) since the stress is computed over a coarse grid (MGC3-$\Omega 1$). A global overview of Sigma-xx and deformed shape is plotted in Figure 5-26.

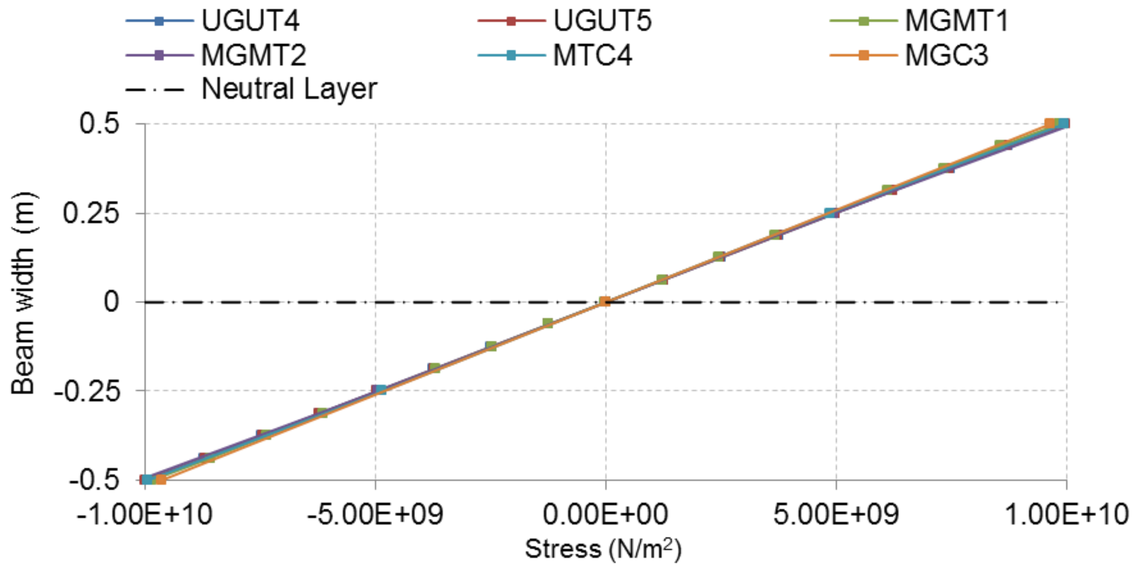


Figure 5-25: Example 1 – Comparison of Sigma-xx at a cross-section x = 1m and t = 0.05s

Table 5-19: Example 1 – RMSE and NRMSE (%). Variable = Sigma-xx at x = 1m and t = 0.05s

	UGUT4	UGUT5
UGUT4	-	1.53×10^7 (0.076%)
UGUT5	1.53×10^7 (0.076%)	-
MGMT1	9.91×10^7 (0.495%)	8.57×10^7 (0.420%)
MGMT2	6.30×10^7 (0.315%)	6.97×10^7 (0.346%)
MTC4	9.60×10^7 (0.480%)	8.16×10^7 (0.408%)
MGC3	2.01×10^7 (1.009%)	1.88×10^7 (0.945%)

* Error ranking for the measurement of Sigma-xx is {2413}

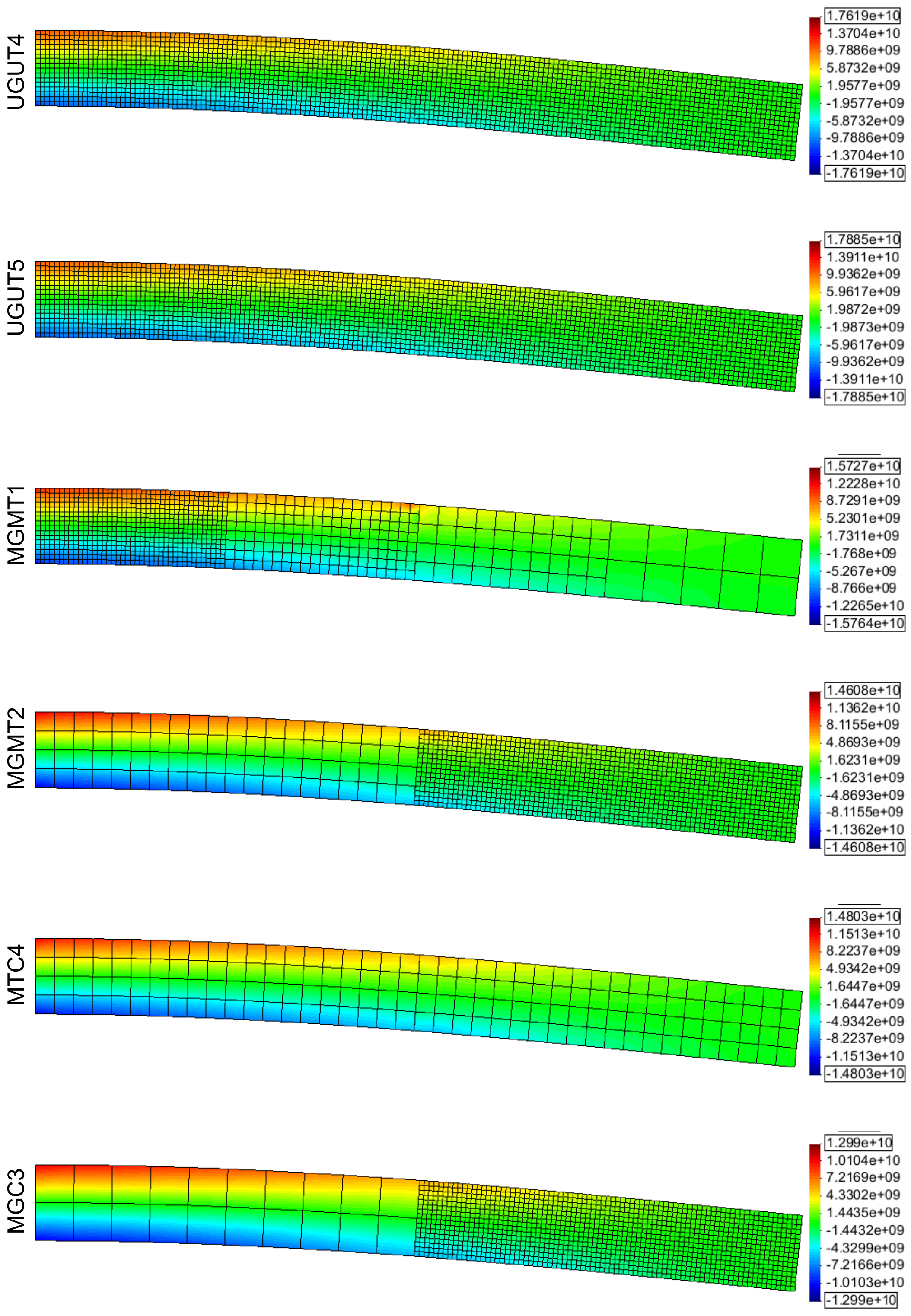
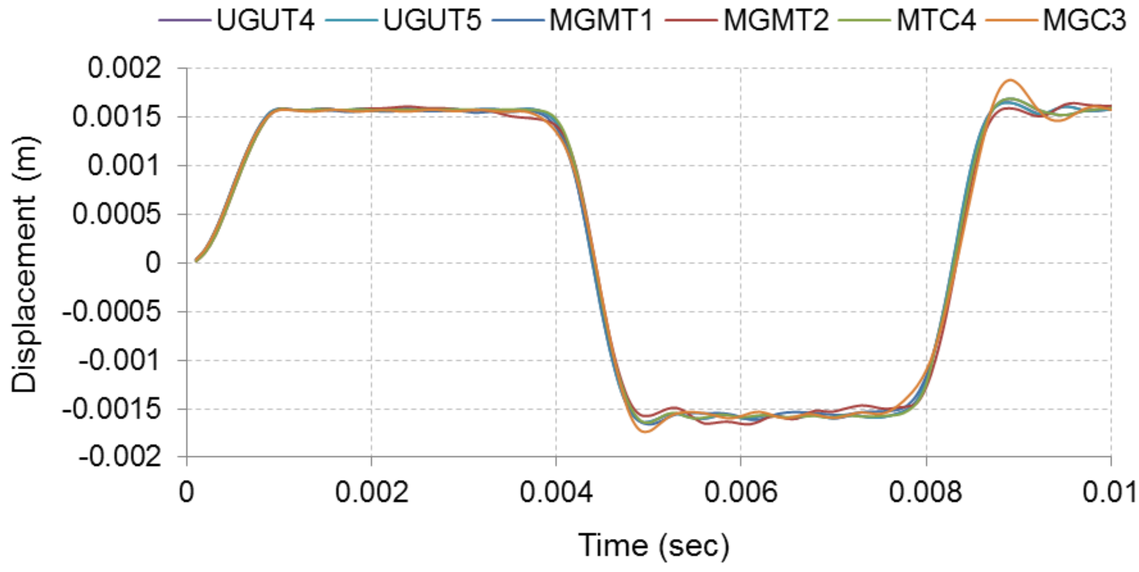


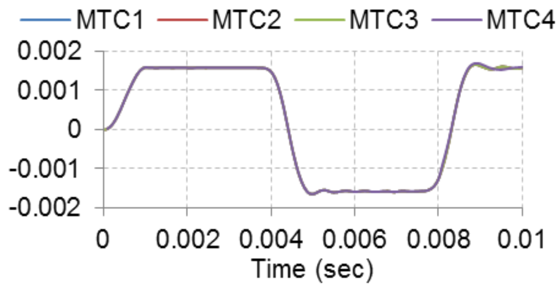
Figure 5-26: Example 1 – Deformed shape and Sigma-xx (N/m^2) at t = 0.05s

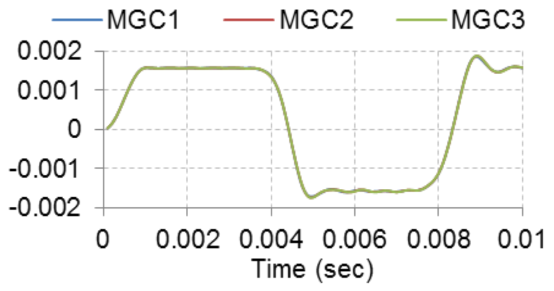
5.4.2 Example 2: Longitudinal Vibrations

In Example 2, we first look at the time evolution of the beam tip deflection. Figure 5-27 shows the horizontal component of tip deflection for UGUT4, UGUT5, MGMT1, MGMT2, MTC1, MTC2, MTC3, MTC4, MGC1, MGC2 and MGC3. Corresponding errors and total number of equations (unconstrained DOF) are listed in Table 5-20.



(a) UGUT4, UGUT5, MGMT1, MGMT2, MTC4, MGC3





(b) MTC1, MTC2, MTC3, MTC4 (c) MGC1, MGC2, MGC3

Figure 5-27: Example 2 – Comparison of horizontal displacement at free end (x = 10m)

Table 5-20: Example 2 – RMSE and NRMSE (%). Variable = displacement at free end (x = 10m)

	UGUT4	UGUT5	Number of Equations
UGUT4	-	2.91×10^{-6} (0.088%)	5440
UGUT5	2.91×10^{-6} (0.088%)	-	5440
MGMT1	1.13×10^{-5} (0.34%)	1.15×10^{-5} (0.35%)	1884
MGMT2	0.43×10^{-5} (0.13%)	0.45×10^{-5} (0.13%)	2954
MTC1	0.97×10^{-5} (0.29%)	1.00×10^{-5} (0.30%)	410
MTC2	0.98×10^{-5} (0.29%)	1.02×10^{-5} (0.31%)	410
MTC3	0.98×10^{-5} (0.29%)	1.04×10^{-5} (0.31%)	410
MTC4	0.99×10^{-5} (0.31%)	1.33×10^{-5} (0.35%)	410
MGC1	1.26×10^{-5} (0.38%)	1.28×10^{-5} (0.39%)	270
MGC2	1.29×10^{-5} (0.39%)	1.31×10^{-5} (0.39%)	798
MGC3	1.31×10^{-5} (0.39%)	1.36×10^{-5} (0.41%)	2814

Errors are relatively small for most cases and very small compared to displacement measurements in Example 1. Once again, MGMT2 has the smallest error since it hosts maximum DOF in comparison with other cases; whereas MGC3 has the maximum error.

As observed in Example 1 results, the error gradually increases for respective MTC and MGC cases, validating the conclusion that accumulated error does increase with increasing grid ratios and time-step ratios. MTC cases yield relatively small errors than MGC cases suggesting that regardless of grid discretization refinement, higher grid ratios with fewer coupling interface DOF can significantly affect the results. The error in MGMT1 is also relatively high, but it should be noted that the measured variable in this case is recorded at in a coarse grid sub-domain (MGMT1-$\Omega 1$). Once again, comparing equivalently discretized cases, UGUT2 and MTC cases in Table 5-21, we can observe the effect of multiple time-scale coupling. As we can see, the effective accumulated error due to MTC is relatively smaller in case of wave propagation type problems, in contrast with the observations made in Table 5-17 for Example 1.

Table 5-21: Example 2 – Comparison between UGUT2 and MTC cases

	UGUT4	UGUT5	Number of Equations
UGUT2	7.00x10^{-6} (0.212%)	7.00x10^{-5} (0.212%)	400
MTC1	0.97x10^{-5} (0.29%)	1.00x10^{-5} (0.30%)	410
MTC2	0.98x10^{-5} (0.29%)	1.02x10^{-5} (0.31%)	410
MTC3	0.98x10^{-5} (0.29%)	1.04x10^{-5} (0.31%)	410
MTC4	0.99x10^{-5} (0.31%)	1.33x10^{-5} (0.35%)	410

Augmented effect of multiple grids and multiple time-scales can be observed by comparing UGUT3 and MGMT1 as follows:

Table 5-22: Example 2 – Comparison between UGUT3 and MGMT1

	UGUT4	UGUT5	Number of Equations
UGUT3	3.77x10^{-6} (0.106%)	3.58x10^{-6} (0.106%)	1440
MGMT1	1.13x10^{-5} (0.34%)	1.15x10^{-5} (0.35%)	1884

Errors trends in the measurement of displacement at free end for Example 2, Table 5-20, are {2413}.

We will now evaluate the capability of MGMT Method in simulating wave propagation across component sub-domains. Since structural waves are coupled in space and time, we need to ensure that MGMT coupling does not damage the characteristic features of a wave (such as amplitude and phase) as it propagates through distinct spatial and temporal resolutions.

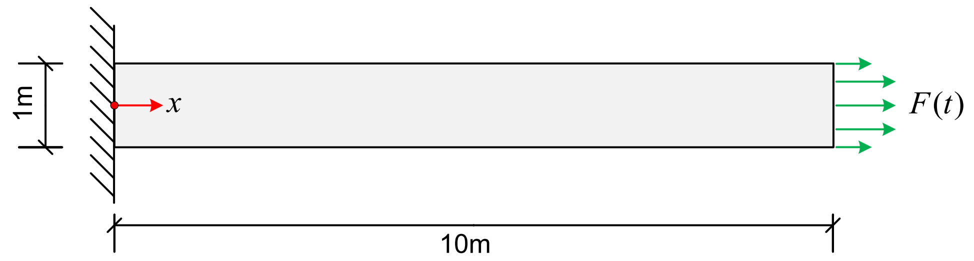


Figure 5-28: Example 2 – Longitudinal vibrations

First, we will look at the distribution of longitudinal stress (Sigma-xx) and longitudinal displacement (U-x) along the length of the beam as a function of space. This will help us gauge the effects of sub-domain interfaces on possible wave reflection or deterioration. Analyzed quantities are measured at the neutral layer and recorded at 5 time instants (t = $1x10^{-4}$, $5x10^{-4}$, $10x10^{-4}$, $15x10^{-4}$ and $20x10^{-4}$) that are within the time frame required by the wave to travel once across the beam. Results are plotted for UGUT4 v/s MGMT1, UGUT5 v/s MGMT2, UGUT4 v/s MTC4 and UGUT4 v/s MGC3. Stresses are plotted in Figure 5-29 and displacements in Figure 5-30, with corresponding errors listed in Table 5-23 and Table 5-24 respectively.

	t = $1x10^{-4}$	t = $5x10^{-4}$	t = $1x10^{-3}$	t = $15x10^{-4}$	t = $2x10^{-3}$
Plot	🟥	🟦	🟩	🟪	🟧

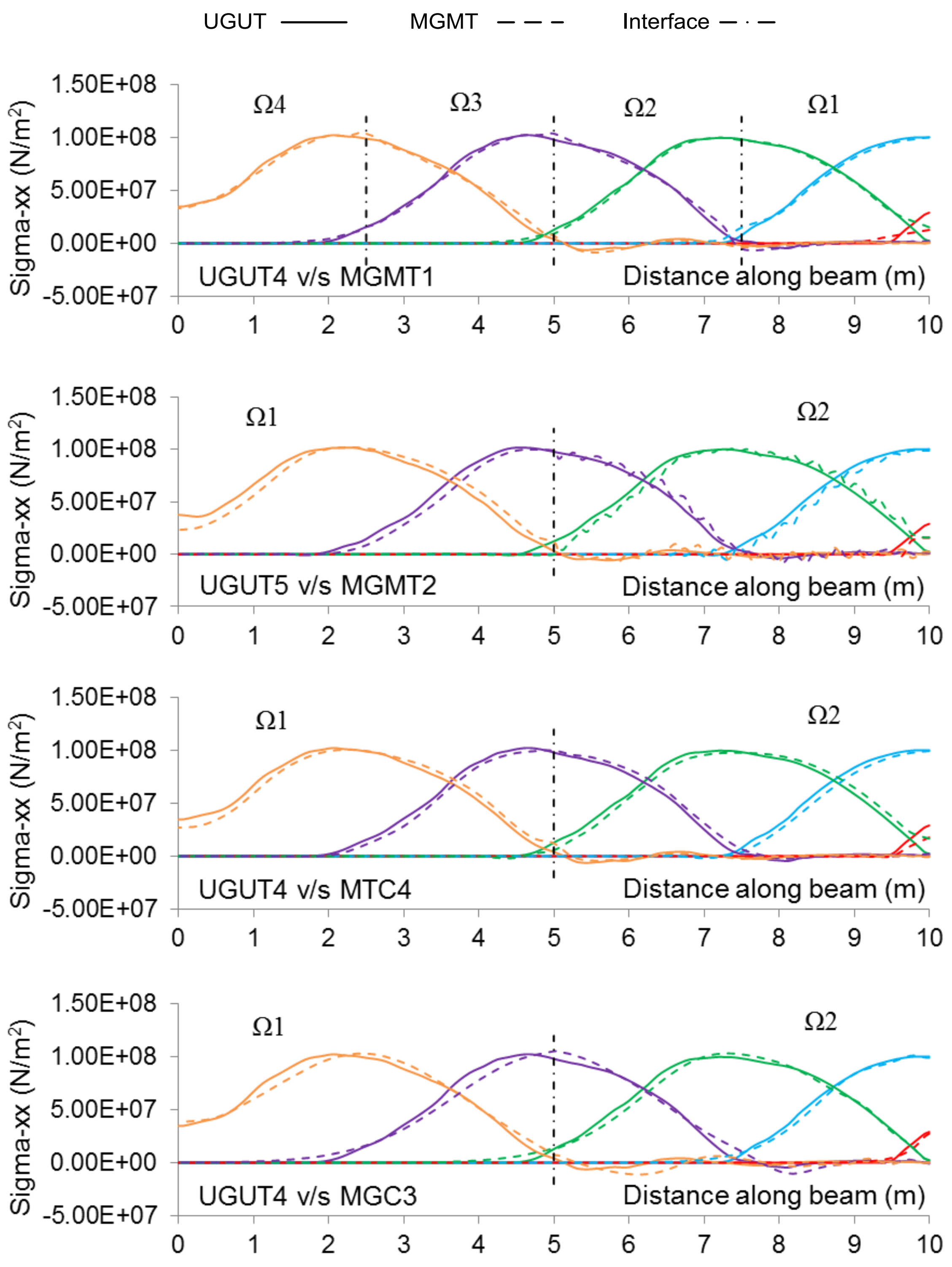


Figure 5-29: Example 2 – Longitudinal stress (Sigma-xx) as a function of space

Table 5-23: Example 2 – RMSE and NRMSE (%). Variable = Sigma-xx as a function of space

	t = 1x10^{-4}	t = 5x10^{-4}	t = 1x10^{-3}	t = 15x10^{-4}	t = 2x10^{-3}
MGMT1	1.96x10^{6} (6.71%)	1.08x10^{6} (1.07%)	1.82x10^{6} (1.83%)	2.15x10^{6} (2.02%)	2.64x10^{6} (2.43%)
MGMT2	2.64x10^{6} (8.86%)	4.06x10^{6} (4.00%)	5.86x10^{6} (5.82%)	4.20x10^{6} (3.97%)	4.39x10^{6} (4.08%)
MTC4	1.79x10^{6} (6.13%)	2.65x10^{6} (2.65%)	3.93x10^{6} (3.93%)	3.50x10^{6} (3.26%)	4.04x10^{6} (3.70%)
MGC3	8.87x10^{6} (3.03%)	1.85x10^{6} (1.85%)	3.15x10^{6} (3.15%)	5.02x10^{6} (4.67%)	4.94x10^{6} (4.52%)

The average error over all selected time-instants is 2.81% (MGMT1), 5.34% (MGMT2), 3.93% (MTC4) and 3.44% (MGC3). From Figure 5-29 we can see that MGMT2 and MGC3 show a nominal phase lag in stress measurements. This can be a result of using relatively high grid density in $\Omega 2$ in disparity with coarse grid density in $\Omega 1$.

A very similar phase lag is observed in MTC4, but not in MGMT1, indicating that multiple grid and multiple time-scale interfaces can slow down the propagation of stress waves in finely discretized (space and time) sub-domains.

Implicit-Explicit coupling in MGMT2 also affects the stress distribution in $\Omega 2$ in the form of jagged undulations which sharply disappear in the implicit sub-domain $\Omega 1$.

MGC3 shows a significant error at the interface of $\Omega 1$ and $\Omega 2$, once again suggesting that higher grid ratios should not be coupled with fewer interface DOF.

Error rankings for 'average over time' stress measurements in this case is {1342}.

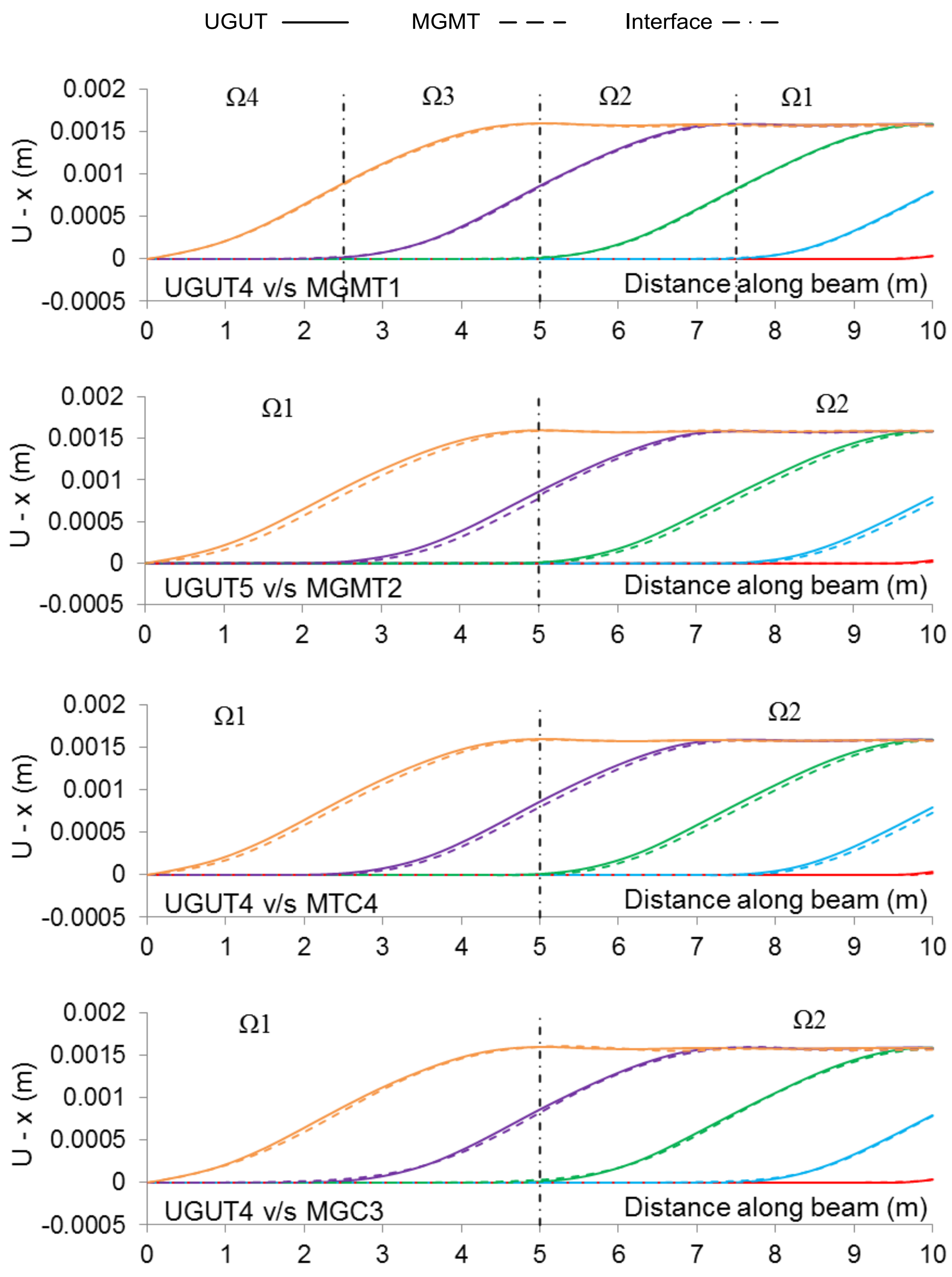


Figure 5-30: Example 2 – Longitudinal displacement (U-x) as a function of space

Table 5-24: Example 2 – RMSE and NRMSE (%). Variable = U-x as a function of space

	$t = 1x10^{-4}$	$t = 5x10^{-4}$	$t = 1x10^{-3}$	$t = 15x10^{-4}$	$t = 2x10^{-3}$
MGMT1	$5.80x10^{-7}$ (1.57%)	$2.46x10^{-6}$ (0.31%)	$4.47x10^{-6}$ (0.28%)	$8.38x10^{-6}$ (0.52%)	$1.23x10^{-5}$ (0.77%)
MGMT2	$3.02x10^{-6}$ (8.12%)	$2.90x10^{-5}$ (3.65%)	$4.14x10^{-5}$ (2.60%)	$2.88x10^{-5}$ (1.81%)	$2.70x10^{-5}$ (1.69%)
MTC4	$2.59x10^{-6}$ (7.03%)	$2.27x10^{-5}$ (2.86%)	$3.20x10^{-5}$ (2.01%)	$3.13x10^{-5}$ (1.97%)	$3.17x10^{-5}$ (1.98%)
MGC3	$1.03x10^{-6}$ (2.79%)	$6.04x10^{-6}$ (0.75%)	$1.35x10^{-5}$ (0.85%)	$1.98x10^{-5}$ (1.24%)	$2.25x10^{-5}$ (1.40%)

The average error over all selected time-instants is 0.73% (MGMT1), 4.46% (MGMT2), 3.96% (MTC4) and 1.75% (MGC3). A 'nominal' phase lag is observed in displacement measurements, however, jagged undulations due to explicit time-stepping in MGMT2-$\Omega 2$ disappear in this case, ensuring better conformance in displacement wave propagation. Error rankings for 'average over time' displacement measurements in this case is {1342}.

We will now look at the time evolution of longitudinal stress (Sigma-xx), longitudinal displacement (U_x), velocity $(\dot{U}_x)$ and acceleration $(\ddot{U}_x)$ as it propagates across the beam as a function of time. Results are measured at 4 different locations, x = 10, 7.5, 5 and 2.5 and are plotted for UGUT4 v/s MGMT1, UGUT5 v/s MGMT2. UGUT4 v/s MTC4 and UGUT4 v/s MGC3 with respective errors listed in subsequent tables.

	x = 10	x = 7.5	x = 5	x = 2.5
Plot	🟥	🟦	🟩	🟪

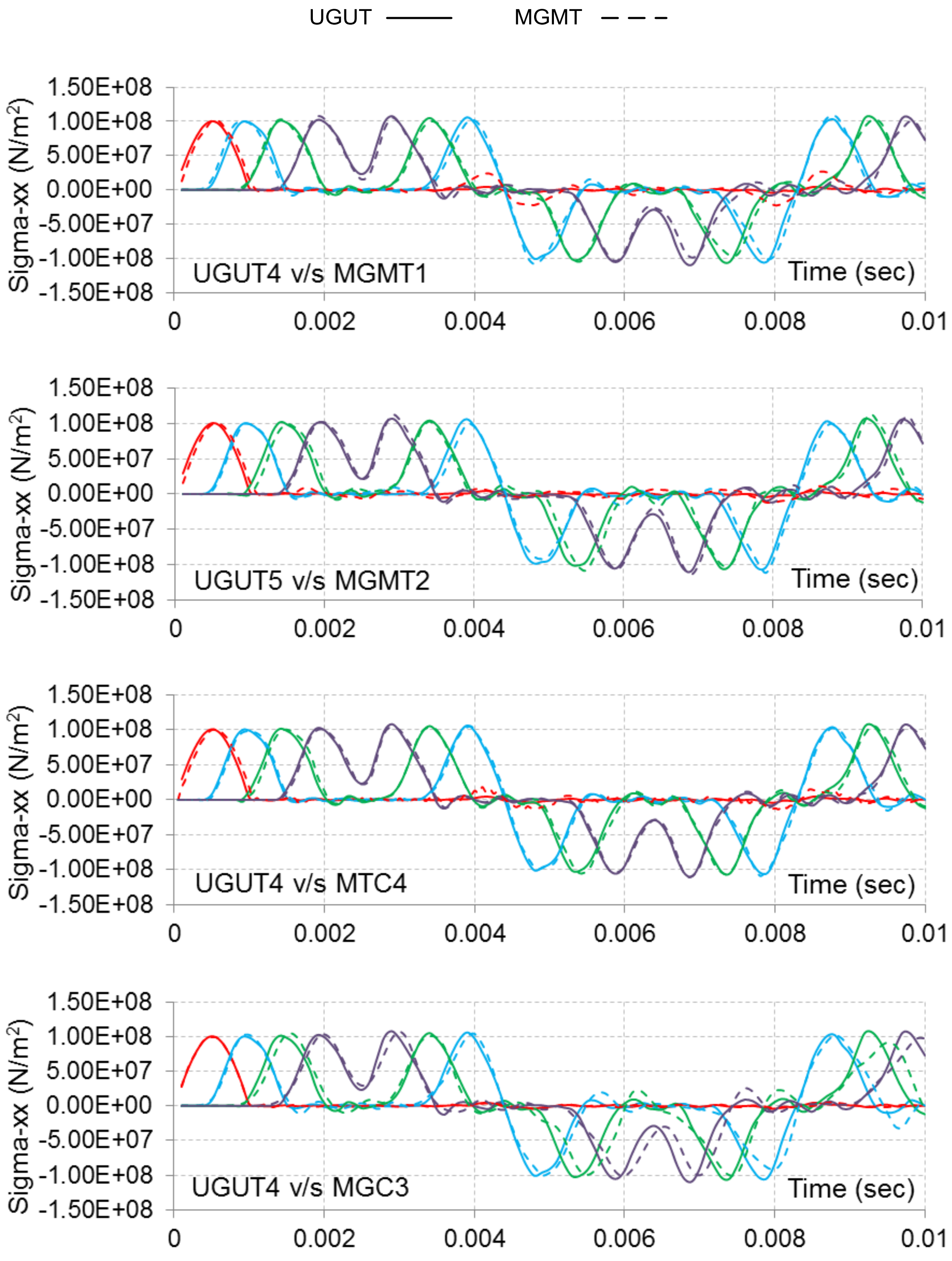


Figure 5-31: Example 2 – Longitudinal stress (Sigma-xx) as a function of time

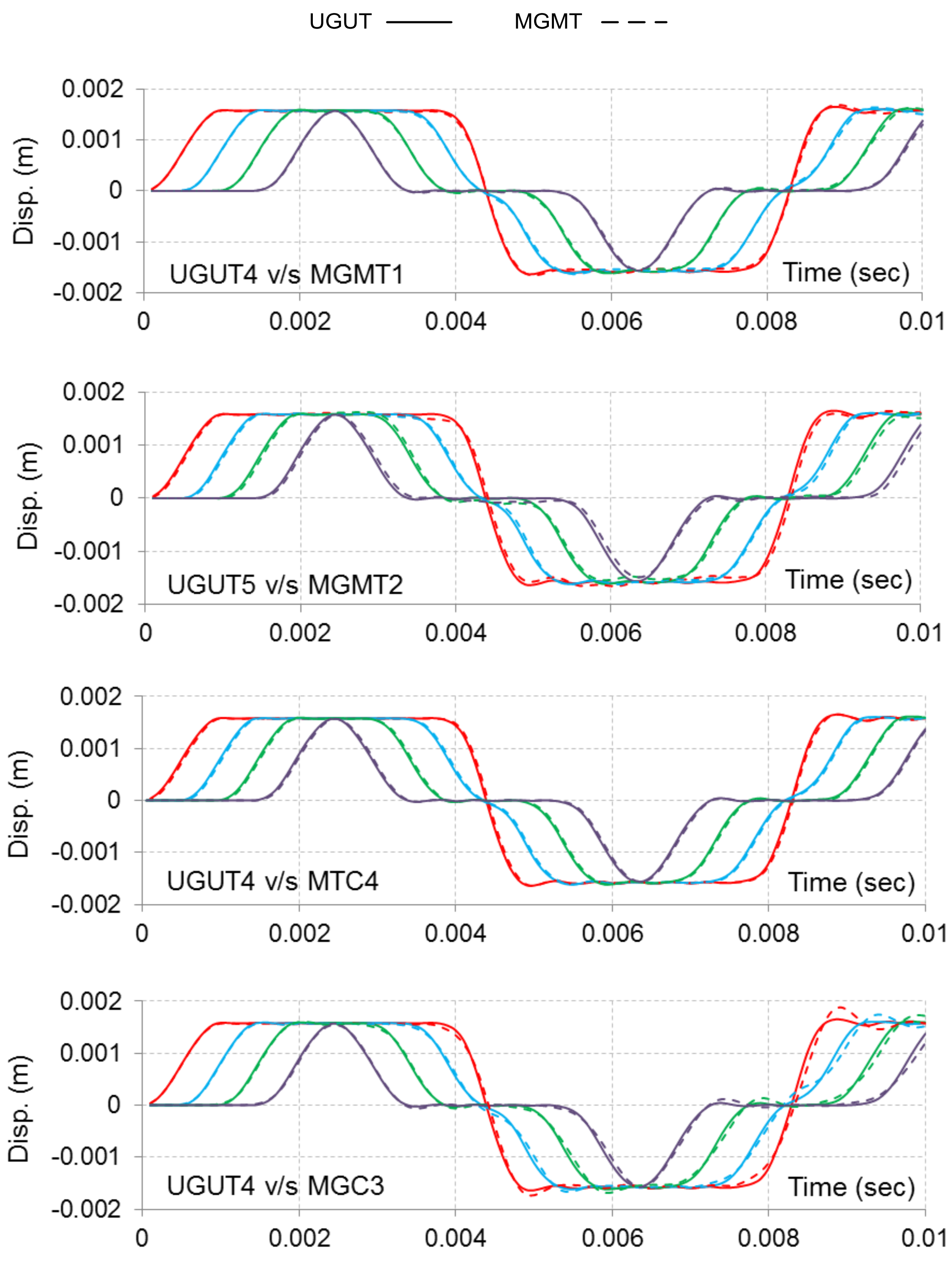


Figure 5-32: Example 2 – Longitudinal displacement (U-x) as a function of time

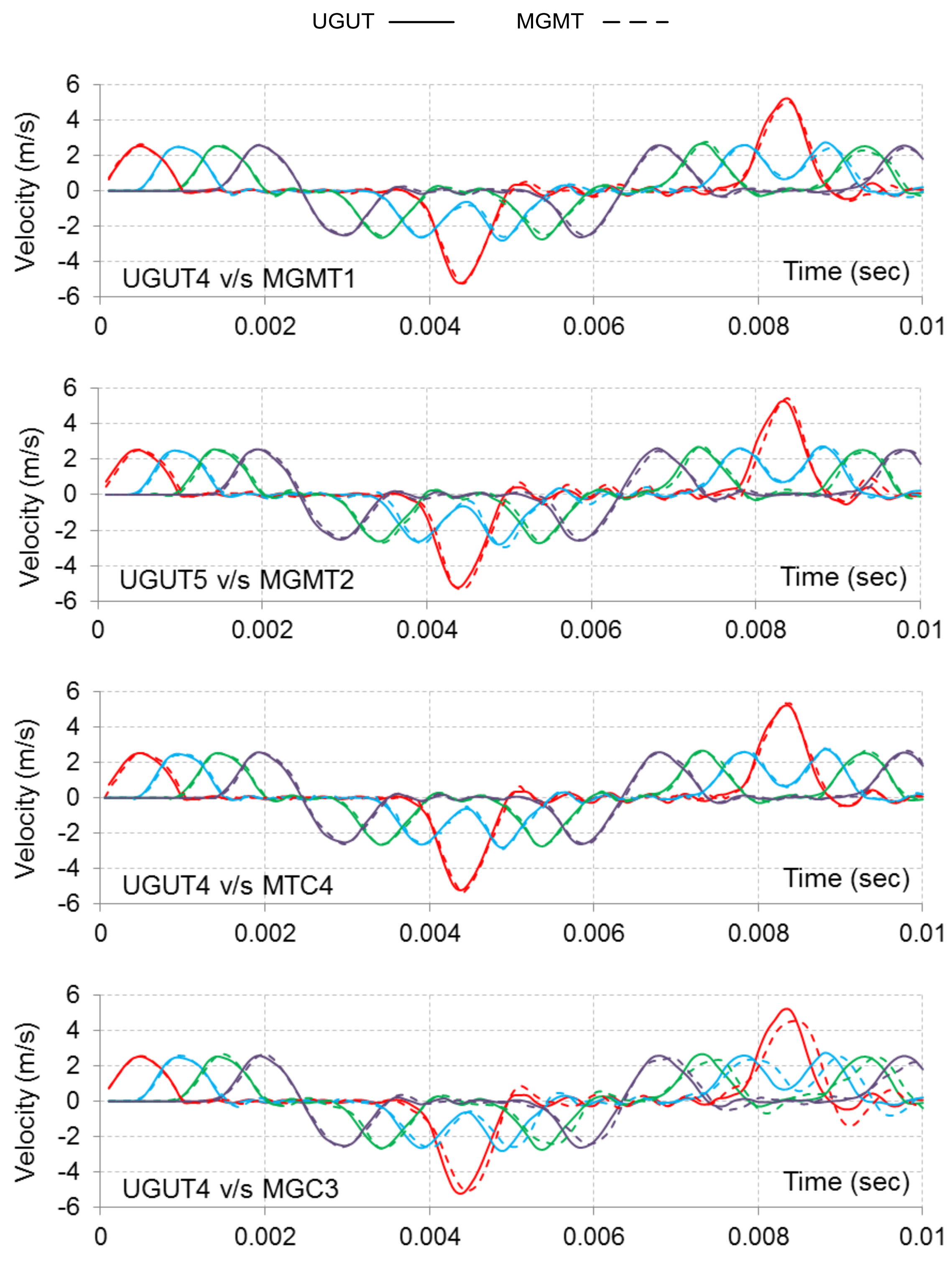


Figure 5-33: Example 2 – Longitudinal velocity ($\dot{U}$-x) as a function of time

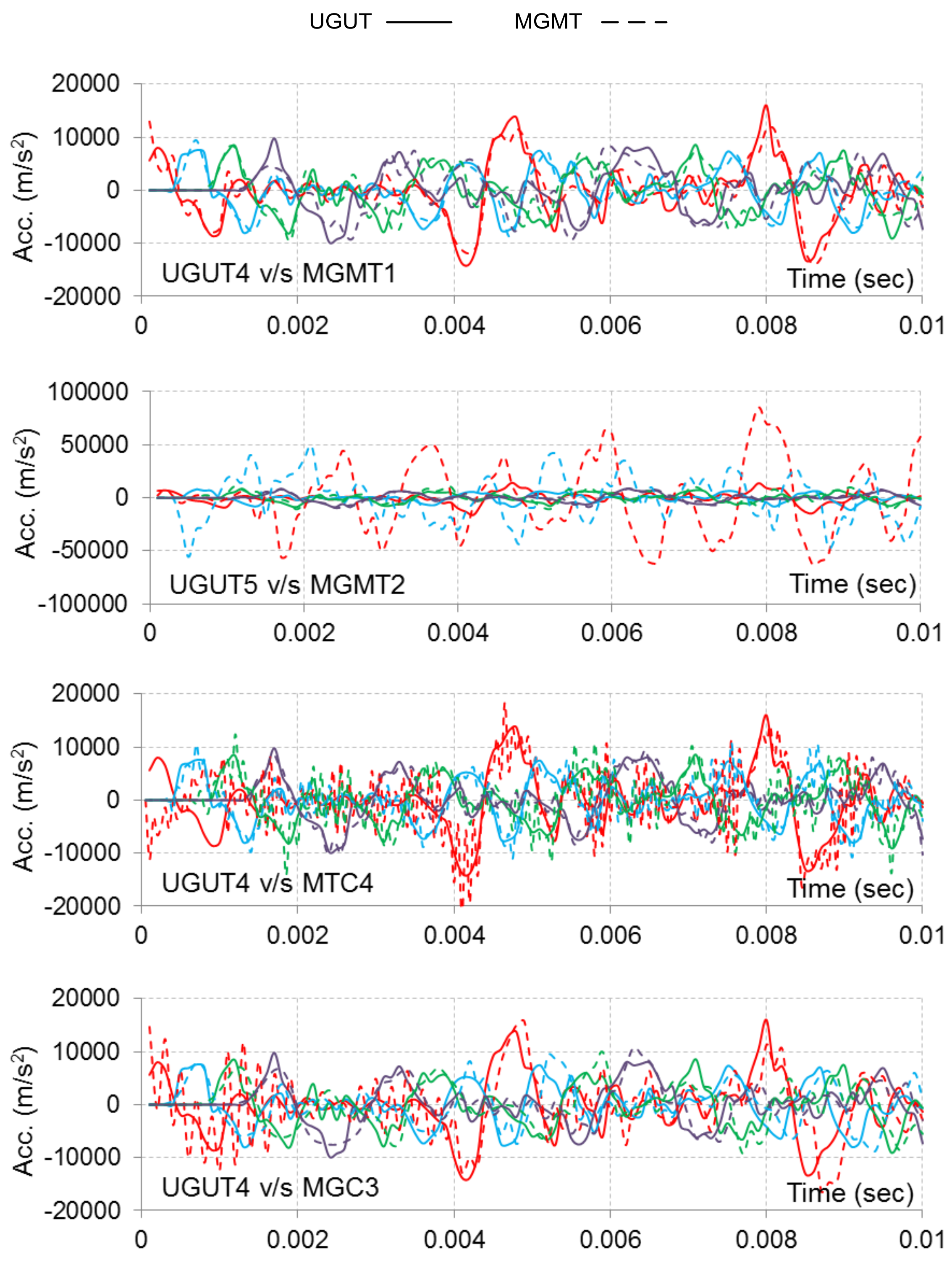


Figure 5-34: Example 2 – Longitudinal acceleration (Ü-x) as a function of time

Table 5-25: Example 2 – RMSE and NRMSE (%). Variable = Longitudinal stress (Sigma-xx)

	x = 10	x = 7.5	x = 5	x = 2.5
MGMT1	8.42×10^6 (8.0%)	7.60×10^6 (3.6%)	6.05×10^6 (2.9%)	7.10×10^6 (3.3%)
MGMT2	5.06×10^6 (4.8%)	7.15×10^6 (3.4%)	9.27×10^6 (4.4%)	5.87×10^6 (2.7%)
MTC4	5.22×10^6 (5.0%)	4.05×10^6 (1.9%)	5.77×10^6 (2.7%)	3.45×10^6 (1.5%)
MGC3	1.44×10^6 (1.3%)	1.03×10^7 (4.9%)	1.54×10^7 (7.3%)	1.25×10^7 (5.7%)

Table 5-26: Example 2 – RMSE and NRMSE (%). Variable = Longitudinal displacement (U-x)

	x = 10	x = 7.5	x = 5	x = 2.5
MGMT1	3.46×10^{-5} (1.1%)	2.86×10^{-5} (0.9%)	3.23×10^{-5} (0.8%)	2.01×10^{-5} (0.6%)
MGMT2	7.25×10^{-5} (2.2%)	4.11×10^{-5} (1.2%)	6.08×10^{-5} (1.9%)	6.60×10^{-5} (2.1%)
MTC4	3.93×10^{-5} (1.1%)	2.92×10^{-5} (0.9%)	2.91×10^{-5} (0.9%)	2.87×10^{-5} (0.9%)
MGC3	7.77×10^{-5} (2.3%)	6.13×10^{-5} (1.9%)	5.43×10^{-5} (1.6%)	4.79×10^{-5} (1.5%)

Table 5-27: Example 2 – RMSE and NRMSE (%). Variable = Longitudinal velocity (U̇-x)

	x = 10	x = 7.5	x = 5	x = 2.5
MGMT1	0.1978 (1.91%)	0.1573 (2.85%)	0.1430 (2.66%)	0.1133 (2.20%)
MGMT2	0.2828 (2.71%)	0.1876 (3.41%)	0.1728 (3.21%)	0.1722 (3.34%)
MTC4	0.1735 (1.67%)	0.092 (1.66%)	0.100 (1.86%)	0.100 (1.95%)
MGC3	0.4716 (4.55%)	0.3853 (6.96%)	0.3156 (5.88%)	0.2283 (4.42%)

Table 5-28: Example 2 – RMSE and NRMSE (%). Variable = Longitudinal acceleration (Ü-x)

	x = 10	x = 7.5	x = 5	x = 2.5
MGMT1	2660.03 (8.9%)	1747.2 (11.2%)	2134.57 (12.1%)	2947.2 (15.0%)
MGMT2	32370.4 (106%)	21195.2 (109%)	2825.17 (16.2%)	1519.43 (8.6%)
MTC4	5594.7 (18.7%)	2469.6 (15.8%)	3067.39 (17.4%)	1501.08 (7.6%)
MGC3	5570.8 (18.6%)	3070.2 (19.6%)	2658.09 (15.0%)	2207.2 (11.2%)

Table 5-29: Example 2 – Average errors and corresponding rankings

	Sigma-xx	U-x	$\dot{U}$-x	$\ddot{U}$-x
MGMT1	5.23%	0.85%	2.40%	11.8%
MGMT2	3.82%	1.85%	3.16%	59.96%
MTC4	2.77%	0.95%	1.78%	14.87%
MGC3	4.80%	1.82%	5.45%	16.10%

Ranking	{4231}	{1432}	{4123}	{1342}

Results for structural wave propagation (Sigma-xx, U-x and $\dot{U}$-x) show fairly good conformance with converged UGUT cases. For longitudinal stress results, MTC4 has minimum errors whereas MGMT1 has the highest errors. MGMT1 records the highest error at x=10 since the stresses here are computed over a coarse grid ($\Omega 1$) but the error sharply falls by almost 50% as the grid density increases at x=10 in MGMT2 and MGC3. Hence finely discretized sub-domains are recommended in regions where stress (or strain) gradients are critical so that resulting errors can be kept to their minimum. This is inference is further validated by noticing that errors also drop at x=7.5, x=5 and x=2.5 for MGMT1, i.e. as the mesh is refined. MGMT2 and MGC3 have the same grid density at x=10 but MGC3 records the minimum error indicating that explicit integration in MGMT2-$\Omega 2$ resulted in higher errors. Results for longitudinal displacement are very satisfactory with an average error of less than 2% for all analyzed cases but they are relatively higher for velocity and acceleration results. Overlooking the effect of grid density on recorded values values, we conclude that the structural wave (stress and displacement) propagates seamlessly across component sub-domains without any significant loss in amplitude or phase.

5.5 Evaluation of Computational Efficiency

Finally we look at the principal advantage of using MGMT Method as opposed to UGUT approach by comparing the improvements in invested computational resources. Table 5-30 lists the total computation time (in sec) required to obtain the global solution of the problem, total number of nodes, elements, number of equations representing the primary unconstrained DOF and the total amount of skyline storage that is descriptive of the total CPU memory in use.

Table 5-30: Comparison of computational resources for Example 1 and Example 2

Case ID	Nodes	Elements	Number of equations	Skyline storage	Solution time (sec)	
					Example 1	Example 2
UGUT4	2737	2560	5440	221352	320.92 (~5 min)	107.53 (~2 min)
UGUT5	2737	2560	5440	221352	24822.27 (~7 hr)	830.14 (~13 min)
MGMT1	959	850	1884	74110	127.98 (~2 min)	41.80
MGMT2	1482	1360	2954	122615	13678.54 (~4 hr)	490.27 (~8 min)
MTC1	210	160	410	5543	16.17	5.50
MTC2					144.81	48.56
MTC3					1396.67 (~23 min)	474.74 (~8 min)
MTC4					13685.74 (~4 hr)	4605.30 (~1 hr)
MGC1	138	100	270	3361	1.02	0.35
MGC2	402	340	798	17949	4.43	1.46
MGC3	1410	1300	2814	120433	28.12	9.25

5.6 Numerical Analysis and Verification Summary

An overview of various scenarios analyzed in this chapter is presented in Figure 5-35. It also illustrates various grid ratios and time-step ratios taken into consideration and also shows the total number of mortar element nodes (■) used to couple distinct discretization.

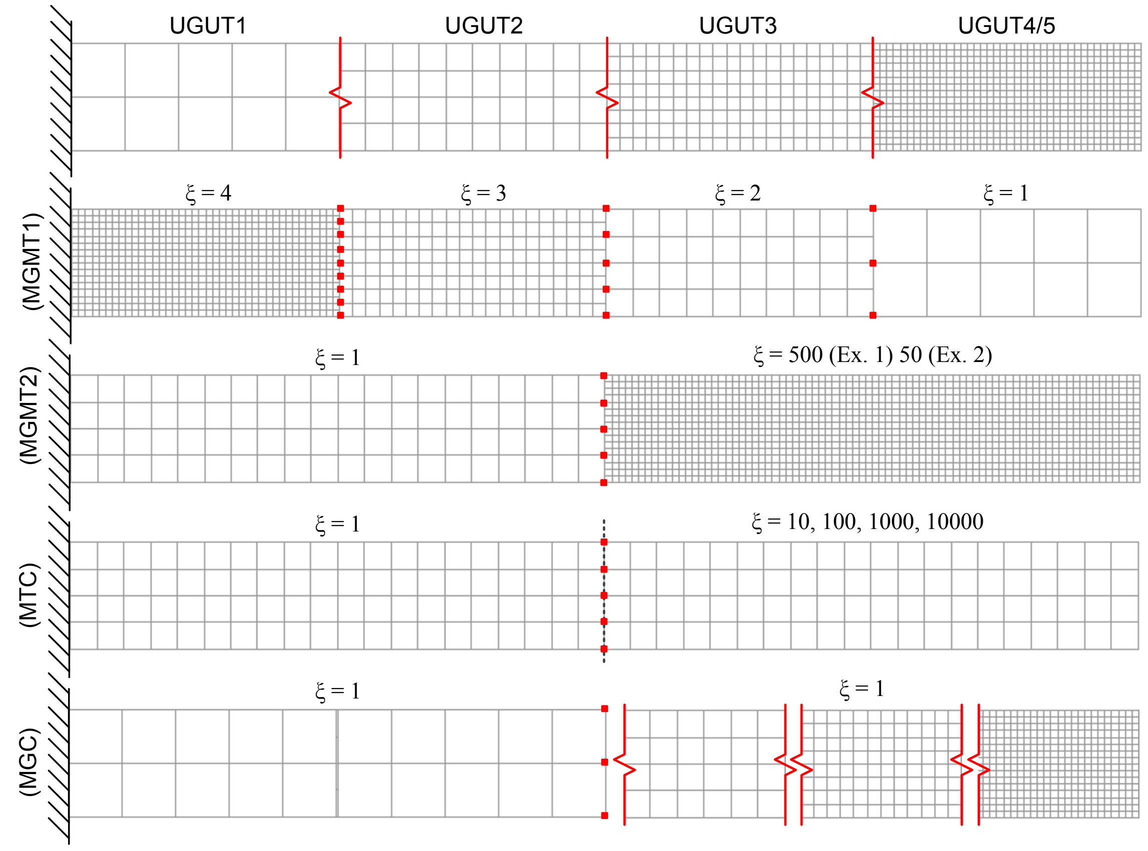


Figure 5-35: Summary of solved cases

Table 5-31 shows a quick overview of various performance evaluation factors analyzed using aforementioned cases and corresponding error rankings (NRMSE in ascending order). Note: Notation for error rankings is – MGMT1 ≡ 1, MGMT2 ≡ 2, MTC4 ≡ 4 and MGC3 ≡ 3.

Table 5-31: Summary of analyzed results

		UGUT4	UGUT5	MGMT1	MGMT2	MTC1	MTC2	MTC3	MTC4	MGC1	MGC2	MGC3	Ranking
Stability Analysis	Kinetic Energy	✓ ✗	✓ ✗	✓ ✗	✓ ✗				✓ ✗			✓ ✗	{2413} {4132}
	Stiffness Energy	✓ ✗	✓ ✗	✓ ✗	✓ ✗				✓ ✗			✓ ✗	{2413} {4132}
	IE v/s External Work	✓ ✗	✓ ✗	✓ ✗	✓ ✗				✓ ✗			✓ ✗	{3241} {2413}
	Domain Interface Energy			✓ ✗	✓ ✗				✓ ✗			✓ ✗	
	Total Interface Energy			✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	{3421} {4231}
	Displacement Continuity			✓	✓				✗			✗	
	Velocity Continuity			✓	✓				✗			✗	{12} {43}
	Acceleration Continuity			✓	✓				✗			✗	
Numerical Accuracy	Tip Deflection	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	{2143} {2413}
	Stress-xx (@ x = 1)	✓	✓	✓	✓				✓			✓	{2413}
	Stress-xx = f(x)	✗	✗	✗	✗				✗			✗	{1342}
	U-x = f(x)	✗	✗	✗	✗				✗			✗	{1342}
	Stress-xx = f(t)	✗	✗	✗	✗				✗			✗	{4231}
	U-x = f(t)	✗	✗	✗	✗				✗			✗	{1432}
	$\dot{U}$-x = f(t)	✗	✗	✗	✗				✗			✗	{4123}
	$\ddot{U}$-x = f(t)	✗	✗	✗	✗				✗			✗	{1342}

✓ = Example 1 (Transverse Vibration) ✗ = Example 2 (Longitudinal Vibration)

Table 5-32 shows the effective computational gain in MGMT implementation relative to corresponding UGUT cases. Here we will use ‘Percentage Change’ (Wikipedia 2014) to express a change in a variable, such as nodes, elements solution time etc. It is calculated by comparing the initial (reference/old) value and the final (updated/new) value, representing the change between them. More generally, if V_1 represents the old value and V_2 the new one:

$$Percentage\ change = \frac{\Delta V}{V_1} = \frac{V_2 - V_1}{V_1} \times 100 \tag{5.4}$$

For example: the solution time for Example 1, using traditional (reference) FE approach was 320.92 sec; and the same solution, using (new) approach, MGMT1, was 127.98 sec. Then, according to Eq. (5.4):

$$Percentage\ change = \frac{127.98 - 320.82}{320.82} \times 100 = -60.12\% \tag{5.5}$$

The negative sign in Eq. (5.5) represents percentage decrease. Accordingly, it can be said that the solution time for MGMT1 decreased by 60.12%.

When comparing variables such as nodes, elements, number of equations, skyline storage and CPU solution time, between UGUT and MGMT cases, we prefer a negative percentage change since these variables represent computational resources that are invested to obtain the same numerical solution. Therefore, fewer the computational resources, i.e. larger the negative percentage change, better is the computational efficiency. Accordingly, percentage change measurements for such variables will be represented by (▲) indicating percentage increase (positive change – requiring more computational resources) and by

(▼) indicating percentage decrease (negative change – requiring less computational resources). For other variables, such as in the comparison of numerical accuracy, we will respectively use ▼ or ▲ to indicate a negative or a positive percentage change.

Consequently, Table 5-32 presents a summary of computational efficiency between UGUT and MGMT cases, with implicit MGMT cases (MGMT1, MTC4 and MGC3 compared) against UGUT4 (I) and explicit case (MGMT2) compared against UGUT5 (E).

Table 5-32: Summary of computational efficiency

	Nodes	Elements	Number of equations	Skyline storage	Solution time (sec)	
					Example 1	Example 2
MGMT1 v/s UGUT4	▼64.96%	▼66.8%	▼65.37%	▼66.52%	▼60.12%	▼61.13%
MGMT2 v/s UGUT5	▼45.85%	▼46.88%	▼45.7%	▼44.61%	▼44.89%	▼40.94%
MTC4 v/s UGUT4	▼92.33%	▼93.75%	▼92.46%	▼97.5%	▲ 4164.5%	▲ 4182.8%
MGC3 v/s UGUT4	▼48.48%	▼49.22%	▼48.27%	▼45.59%	▼91.24%	▼91.4%

Results show that invested computational resources, both memory and processing time, show significant reduction than corresponding UGUT cases; with the exception of MTC4 which has a considerably higher CPU solution time since it employs a significantly large time-step ratio ($\xi = 10,000$) and a smaller time-step in $\Omega 2$: 5×10^{-8} sec compared to 0.125×10^{-3} sec for Example 1 and 5×10^{-9} sec compared to 0.125×10^{-4} sec for Example 2.

Chapter 6: MGMT Example Problems and Results

In this chapter we will present some FE problems wherein MGMT Method can be potentially used to preserve numerical accuracy in desired critical regions whilst improving computational efficiency globally. Discussed examples provide a comprehensive matrix of result analysis by comparing variables such as: global energies, augmented interface energies, kinematic conformity, interface continuity, displacement/stress contour plots and relative errors (RMSE/NRMSE) where relevant. Principal advantage in using MGMT Method, which is the gain in computational efficiency, is also discussed by comparing invested computational resources and CPU solution times between various UGUT and MGMT cases.

Various examples analyzed in this chapter are as follows:

- **Example 1: Stress Resolution in Critical Regions**

  Here we will analyze two classical FE problems that primarily focus on stress resolution. In Example 1.1: Plate with a Hole, we compare MGMT results with both: analytical and UGUT results whereas in Example 1.2: 3 Point Bending Test, we compare results between transition mesh (UGUT) and structured mesh (MGMT) approach.

- **Example 2: Analysis of Heterogeneous Structural Systems**

  The focus in this example problem is to ensure accurate and efficient modeling of structural wave propagation across heterogeneous (material) structural systems, whilst highlighting the limitations in traditional FE approach and corresponding advantages in using MGMT Method.

- **Example 3: Steel Girder Subjected to Impulse Loading**

  This example analyzes the effects of isolating a small region, which is subjected to short duration impulse loads, with a fine scale discretization so the critical region can be solved using explicit time-integration whilst using implicit integration in the larger, remote region. Global vibration characteristics are measured and compared with a fine grid, explicit UGUT discretization.

- **Example 4: Curved Frame under Point Loading**

  Similar to Example 3, the goal in this example is to isolate the concentrated point load with fine discretization whilst ensuring that the dynamic behavior in remote regions is retained. However, this example uses 8 node quadrilateral elements with quadratic shape functions, and since interface reactions (or Lagrange Multipliers) are discretized using linear shape functions, this example tests MGMT interface coupling by comparing kinematic conformity and continuity at the interface.

- **Example 5: Bridge Analysis**

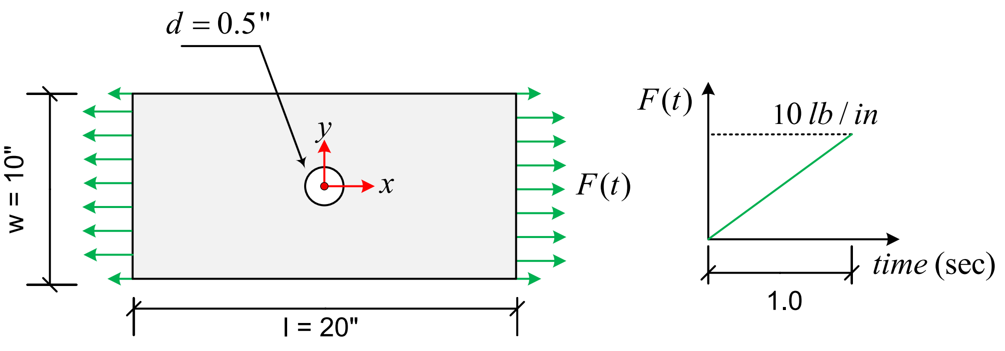
  Here we analyze a large domain with both: heterogeneous material components and application of complex loading functions; to observe overall structural dynamics, kinematic conformity at material interfaces and gain in computational efficiency. Reference results are obtained from a UGUT simulation and are compared with multiple time-step (MTC), multiple grid (MGC) and MGMT coupling approaches.

6.1 Example 1: Stress Resolution in Critical Regions

6.1.1 Example 1.1: Plate with a Hole

The "plate with a hole" problem is one of the fundamental learning steps in any study involving finite element analysis as it illustrates a number of key points essential to the accurate application of the FEM to stress analysis. In this example, we will examine the distribution of stress in a flat plate (2D) with a hole under uniaxial loading (simple tension). We will approach this problem with both, uniform grid uniform time-scale (UGUT) and multiple grid multiple time-scale (MGMT) discretizations.

The domain under analysis, Figure 6-1, is 10” wide, 20” long and 0.1” thick with a circular hole of diameter 0.5” at the center. The domain is discretized using 4-node quadrilateral elements (2 DOF/node) with plane stress formulation, consistent mass matrix and zero damping. Isotropic linear elastic material properties with modulus of elasticity (E) = 3x10^7 Psi, Poisson’s ratio (ν) = 0.3 and mass density (ρ) = 7.33x10^{-4} lbf-sec^2/in^4 are used.



(a) Domain under consideration (b) Linear (ramp) load

Figure 6-1: Plate with a hole under uniaxial loading

Using Classical Stress Theory (CST), longitudinal stress in the plate (ignoring the hole) can be obtained as:

$$\sigma = \frac{F}{w \times t} = \frac{10}{10 \times 0.1} = 10 Psi \tag{6.1}$$

For a plate with a hole, under uniaxial tension, nominal stress is obtained as:

$$\sigma_{nom} = \frac{F}{(w-d)t} = \frac{10}{(10-0.5) \times 0.1} = 10.5263 Psi \tag{6.2}$$

Maximum stress in the plate, in vicinity of the hole, can then expressed as:

$$\sigma_{\max} = K_t \, \sigma_{nom} \tag{6.3}$$

Where theoretical stress concentration factor (K_t) can be obtained from the following chart (Budynas & Nisbett 2008):

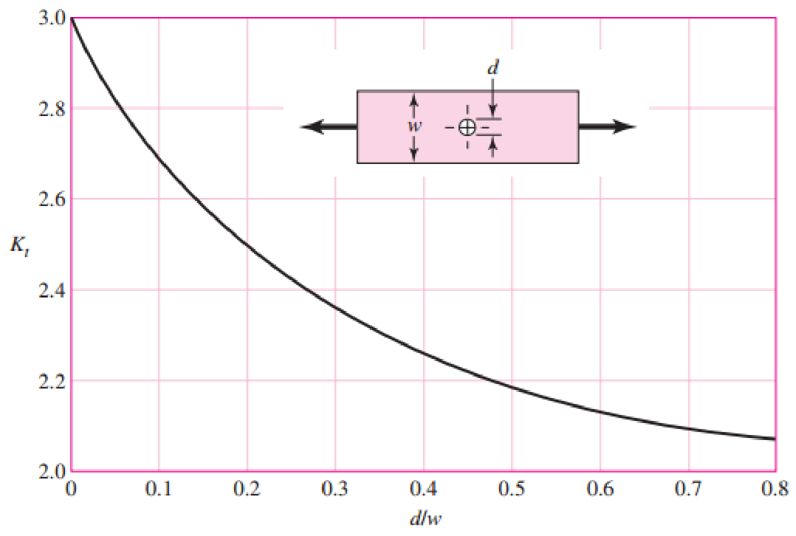


Figure 6-2: Theoretical stress concentration factor K_t (Budynas & Nisbett 2008)

For the example problem described in Figure 6-1 we have:

$$\frac{d}{w} = \frac{0.5}{10} = 0.05 \tag{6.4}$$

Therefore, from Figure 6-2, we have K_t = 2.8 and accordingly the maximum stress in the vicinity of the hole is:

$$\sigma_{\max} = 2.8 \times \sigma_{nom} = 29.4736 Psi \tag{6.5}$$

Now that we have established the maximum value of stress in the vicinity of the hole via analytical method, we will approach this problem using FEM. We will compare the computed stress (Sigma-xx) not only in the vicinity of the hole, but also as a function of space along the width and length of the plate as shown in Figure 6-3.

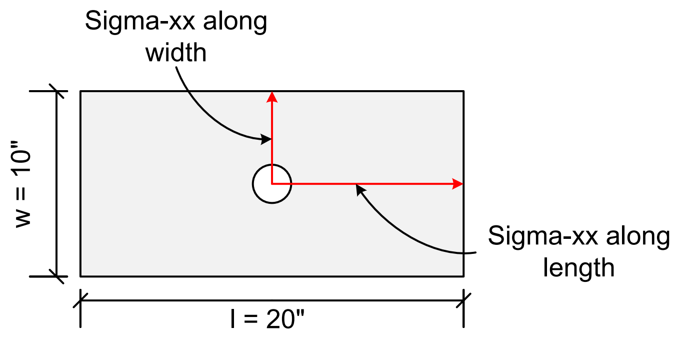
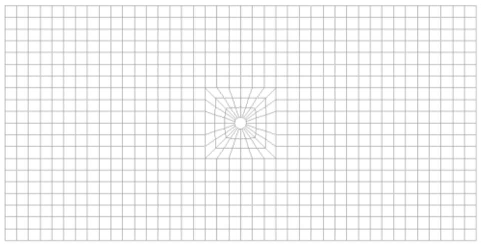


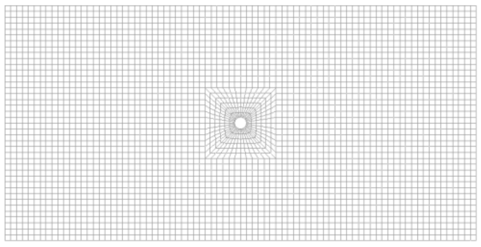
Figure 6-3: Sigma-xx as a function of space

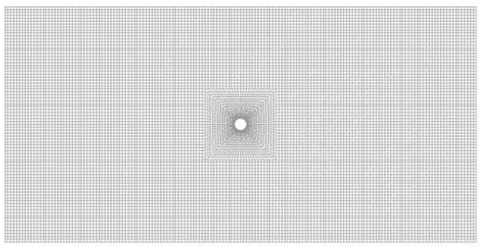
Figure 6-5 (UGUT1) shows the distribution of longitudinal stress as a function of space along the width of the plate. The computed maximum stress in this case is 20.64 Psi. This result is far from the theoretical value; in fact it is almost ▼30%. The difference can be explained by the coarse mesh in the stress concentration region that is in the vicinity of the hole. In order to achieve desired/converged value of maximum stress, two different approaches can be used for this problem. The classical approach is to refine the grid discretization, using H-method as discussed under Section 5.2, or to refine the mesh only in the vicinity of the hole and establish multiple grid connections between critical region and the rest of the structure. Accordingly, we adopt both: refined UGUT and MGMT method with the following parameters.

Table 6-1: Case parameters

Case ID	Nodes	Elements	Grid Spacing (H)	Time-step (ΔT)
UGUT1	908	836	$\overline{H} = 0.5$	$1\text{x}10^{-3}$
UGUT2	3488	3344	$\overline{H} = 0.25$	$0.5\text{x}10^{-3}$
UGUT3	13664	13376	$\overline{H} = 0.125$	$0.25\text{x}10^{-3}$
MGMT-Ω1	836	764	$\overline{H} = 0.5$	$1\text{x}10^{-3}$
MGMT-Ω2	4800	4608	$\overline{H} = 0.125$	$0.25\text{x}10^{-3}$


(a) UGUT1


(b) UGUT2


(c) UGUT3

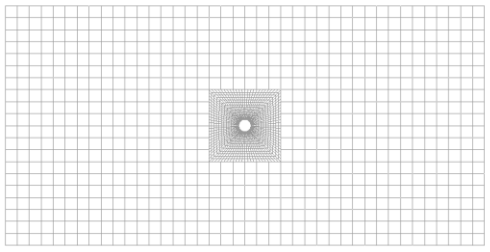

(d) MGMT

Figure 6-4: Analyzed grids

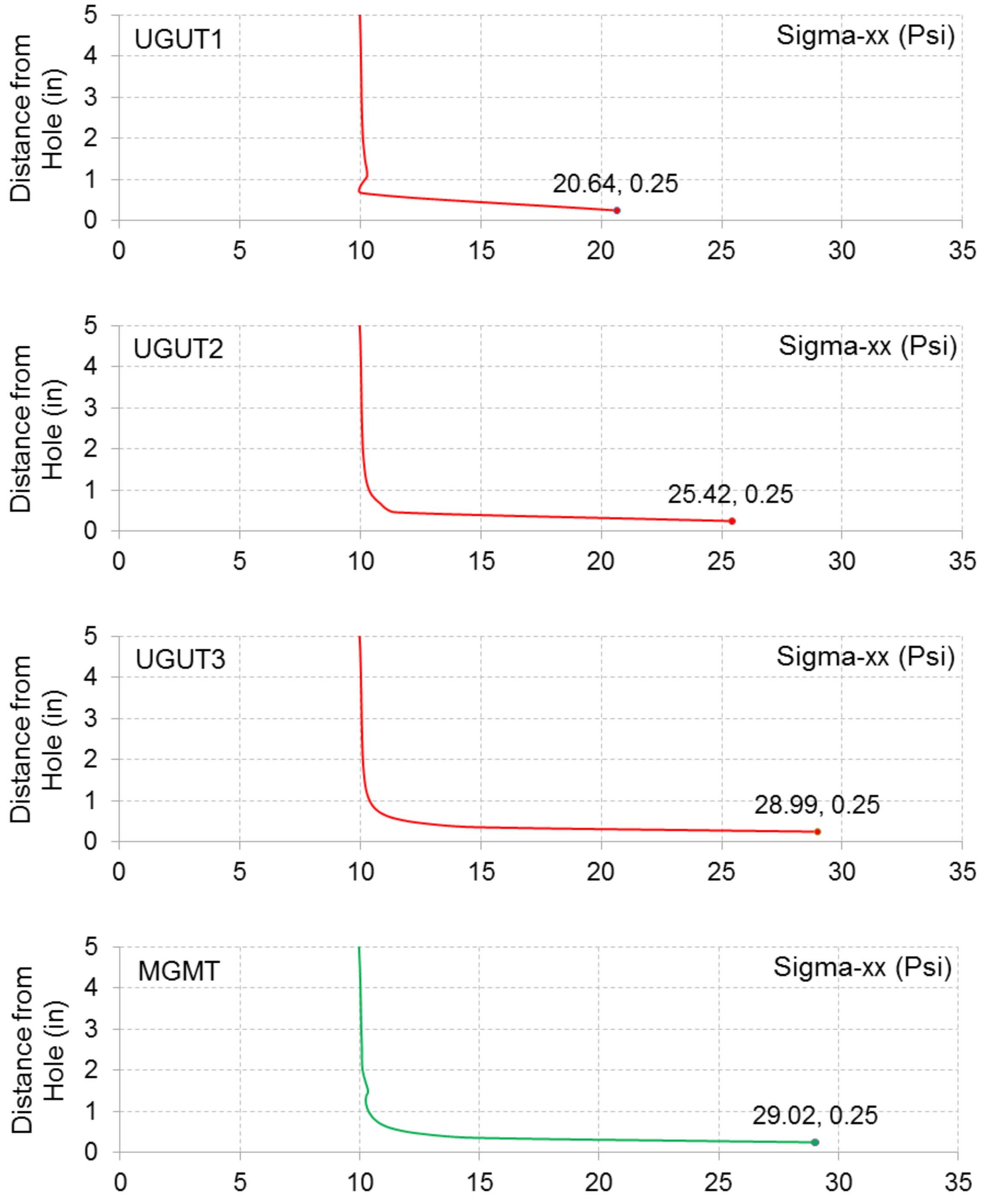


Figure 6-5: Sigma-xx as a function of distance from the hole (along the width of the plate)

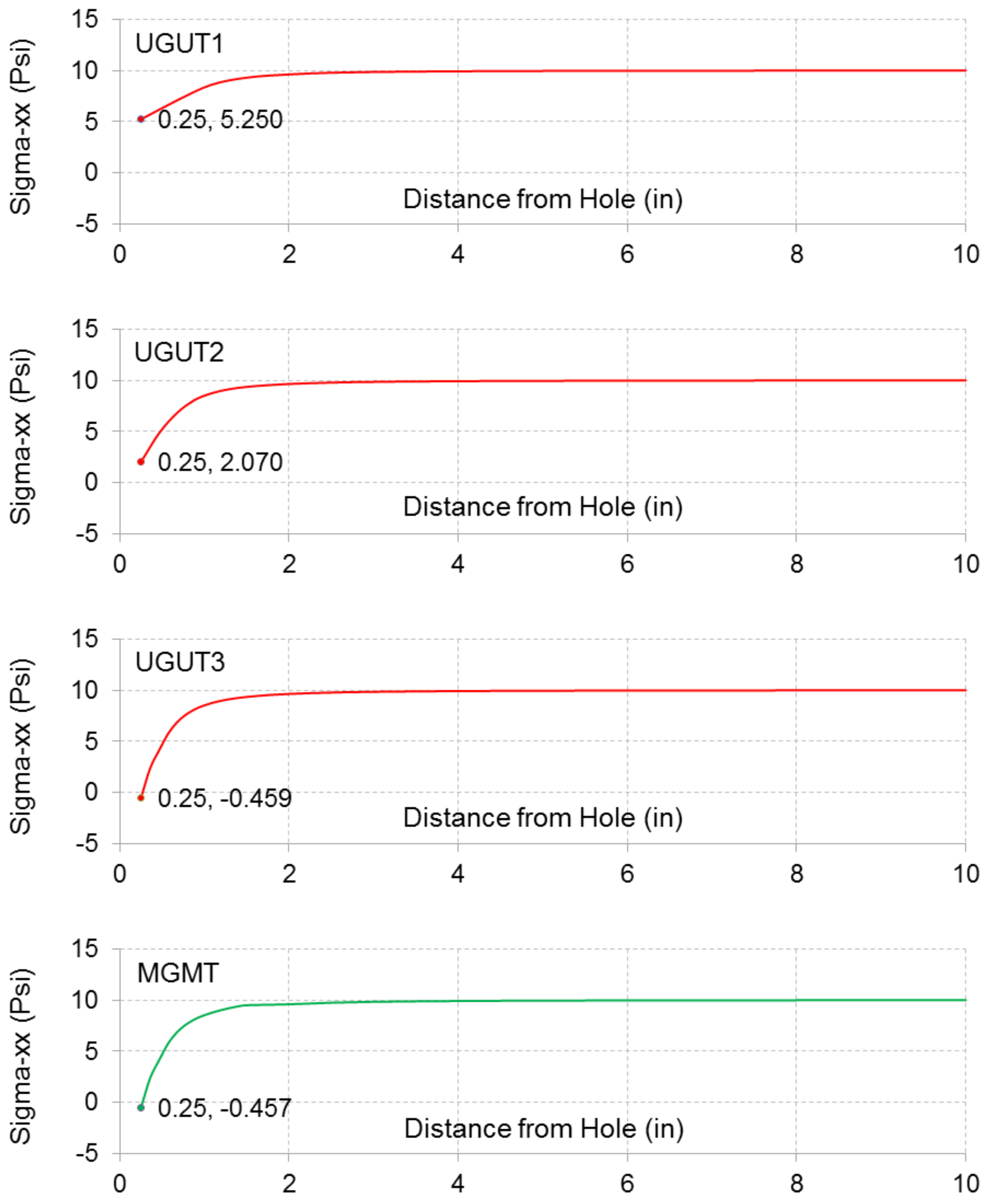


Figure 6-6: Sigma-xx as a function of distance from the hole (along the length of the plate)

We can see from Figure 6-5 that the maximum longitudinal stress increases (and converges towards theoretical value) as the domain gird is refined. It is 25.42 Psi (▼13.75%) for UGUT2, 28.99 (▼1.64%) for UGUT3 and 29.02 (▼1.54%) for MGMT. Similar trend is also observed for the minimum longitudinal stress, Figure 6-6. In fact, only UGUT3 (-0.459 Psi) and MGMT (-0.46 Psi) discretizations are capable of predicting compression on the lateral edge of the hole. We also compare the overall distribution of longitudinal and lateral stress, and lateral displacement and notice that the contours get smoother with grid refinement.

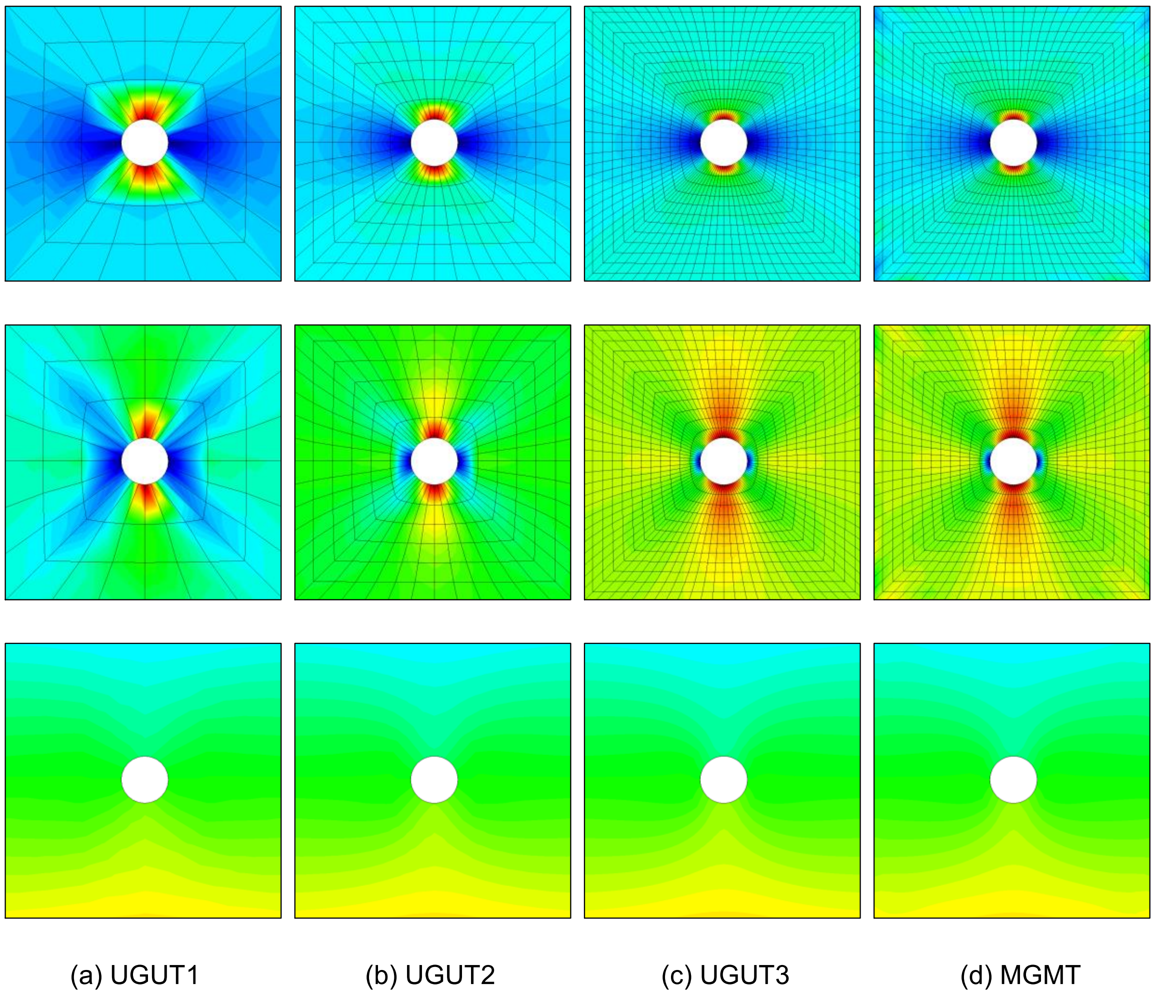
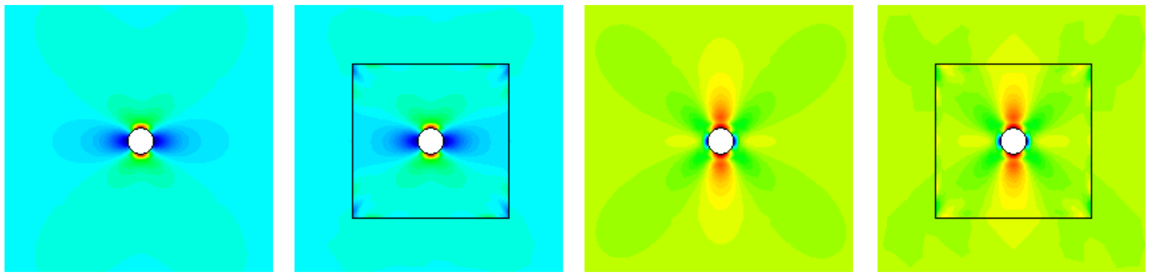


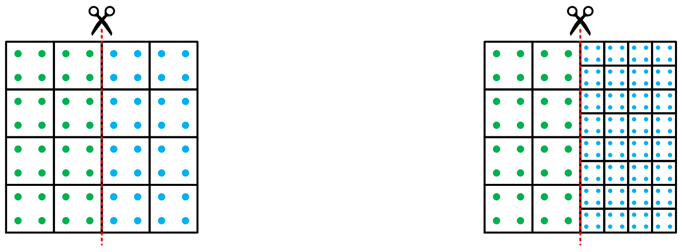
Figure 6-7: Contour plots for Sigma-xx, Sigma-yy and Displacement-y around the hole



(a) Sigma-xx UGUT3 (b) Sigma-xx MGMT (c) Sigma-yy UGUT3 (d) Sigma-yy MGMT

Figure 6-8: Contour plots for Sigma-xx and Sigma-yy outside the critical region

Zooming out of the critical region, Figure 6-8, we notice a discontinuity in stress contours for MGMT discretization; however this discontinuity does not exist in Figure 6-5 and Figure 6-6. This is because former plots were obtained by stresses computed at nodal location, whereas latter contour plots are obtained by stresses computed at element integration points. We clearly see from Figure 6-9 that non-conforming interfaces also result in non-conforming element integration points, and it should be noted that the multi-constraint operator (L) works on nodal quantities only. Accordingly, in order to establish continuity of quantities computed at integration points; one should implement a multi-constraint operator that connects disparate integration points from adjacent sub-domains or simply project these quantities onto nodes using element shape functions and L.



(a) Conforming interface (d) Non-conforming interface

Figure 6-9: Distribution of element integration points

Following figures plot the global energies (kinetic energy and stiffness energy) for UGUT3 and MGMT along with the total augmented interface energy for MGMT. Also, listed is the RMSE for respective plots.

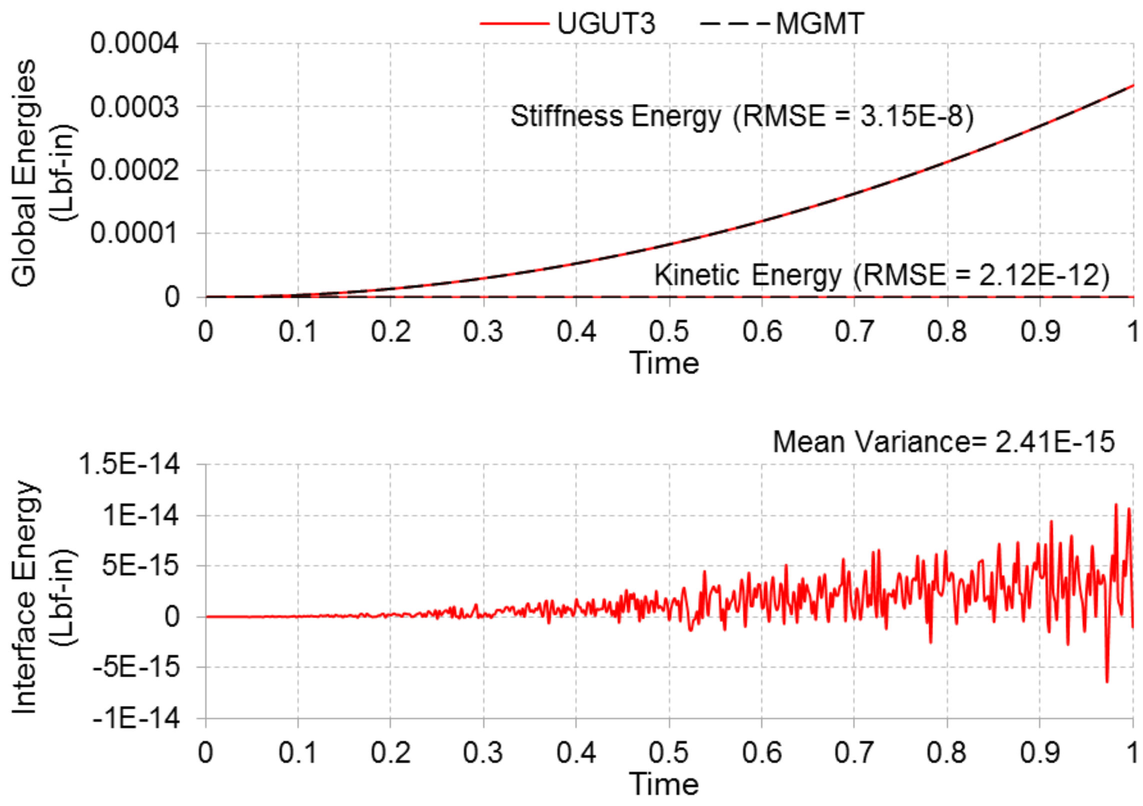


Figure 6-10: Global energies for UGUT3 v/s MGMT and augmented (total) interface energy

Augmented interface energy does accumulate over time, and once again is the result of fewer number of interface coupling DOF (or high grid ratio). MGMT-$\Omega 1$ in this case has a total 24 'non-mortar' nodes whereas MGMT-$\Omega 2$ has 96 'mortar' nodes, resulting in only 24 mortar element nodes and hence only 24 Lagrange Multipliers (λ) to communicate information across connecting sub-domains.

Finally we look the invested computational resources for all analyzed cases. UGUT3 has the smallest error (1.64%) amongst UGUT cases however it requires ~12 hours of computation time. MGMT on other hand takes only 18 min to solve with an even smaller error (1.54%).

Table 6-2: Example 1.1 – Comparison of computational resources

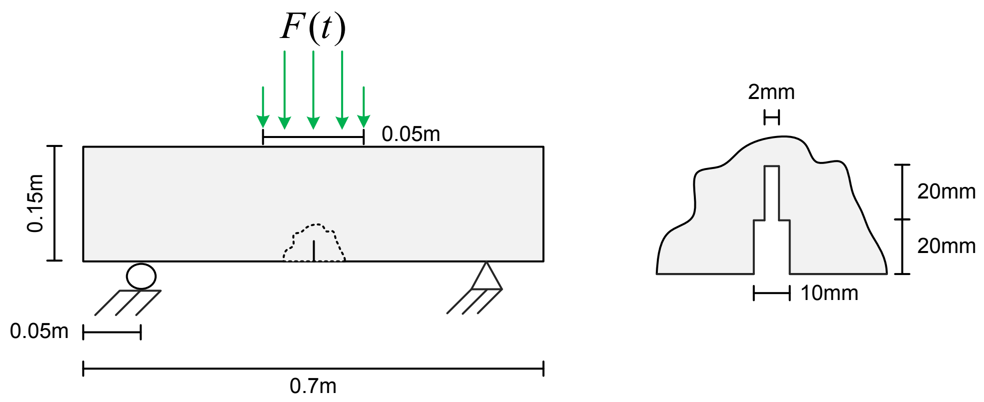
Case ID	Number of equations	Skyline storage	Solution time (sec)	% change in Sigma-xx
UGUT1	1795	108723	81.57	▼29.97%
UGUT2	6935	814215	1749.16 (~30 min)	▼13.75%
UGUT3	27247	6296845	43967.58 (~12 hr)	▼1.64%
MGMT	4147	336151	1103.33 (~18 min)	▼1.54%

MGMT v/s UGUT3	▼ 84.78%	▼ 94.66%	▼ 97.49%

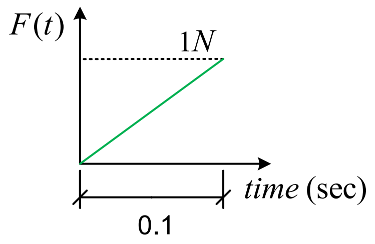
6.1.2 Example 1.2: 3 Point Bending Test

In this example we will analyze a single edge notch concrete beam under bending. The goal is to demonstrate the capability of MGMT Method in resolving critical regions with structured fine scale discretizations in the vicinity of the notch/crack.

The domain under consideration is shown in Figure 6-11 (a) with a linear ramp load of 1N as shown in Figure 6-11 (a). The domain is discretized using 4-node quadrilateral elements (2 DOF/node) with plane stress formulation, consistent mass matrix and zero damping. Isotropic linear elastic material properties for concrete with a modulus of elasticity (E) = 40×10^9 N/m^2, Poisson's ratio (ν) = 0.2 and mass density (ρ) = 2400 Kg/m^3 are used.



(a) Domain under consideration



(b) Linear (ramp) load

Figure 6-11: 3 point bending test

Since the edge notch (crack) is significantly smaller in size than rest of the beam, typical FE approach would require a transition or an unstructured mesh so the crack can be resolved whilst using coarser elements in the remote region. However, unstructured mesh can inherently result in elements that are poor in shape quality within the transition zone. This is not recommended since it can adversely affect the quality of the solution. MGMT Method on other hand allows the user to decompose the original problem into sub-domains which can then be discretized independent of each other. Hence permitting structured fine scale discretization in desired critical regions. Accordingly, in this example we will approach the problem with unstructured transition mesh and structured multiple grids, Table 6-3 and Figure 6-12, for comparison.

Table 6-3: Simulation parameters

Case	Grid spacing (H)	Newmark parameters	Time-step (ΔT)
Transition Mesh	$\overline{H} = 0.0055$	$\beta = 0.25$, $\gamma = 0.5$ (Implicit)	$1\text{x}10^{-4}$
Structured Multiple Grids	$H^1 = 0.025$ $H^2 = 0.001$	$\beta^1 = 0.25$, $\gamma^1 = 0.5$ (Implicit) $\beta^2 = 0.25$, $\gamma^2 = 0.5$ (Implicit)	$\Delta t^1 = 1\text{x}10^{-3}$ $\Delta t^2 = 1\text{ x}10^{-4}$

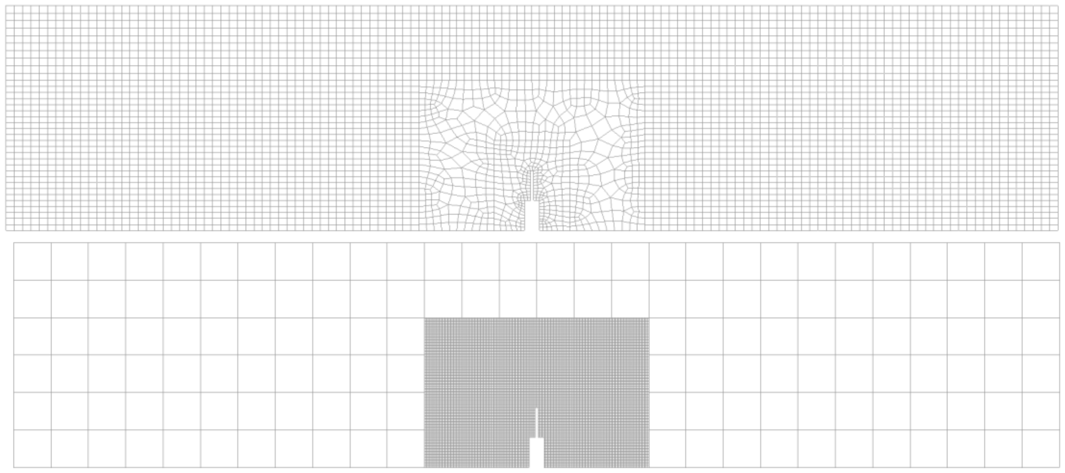


Figure 6-12: Domain grids: Transition Mesh and Structured Multiple Grids

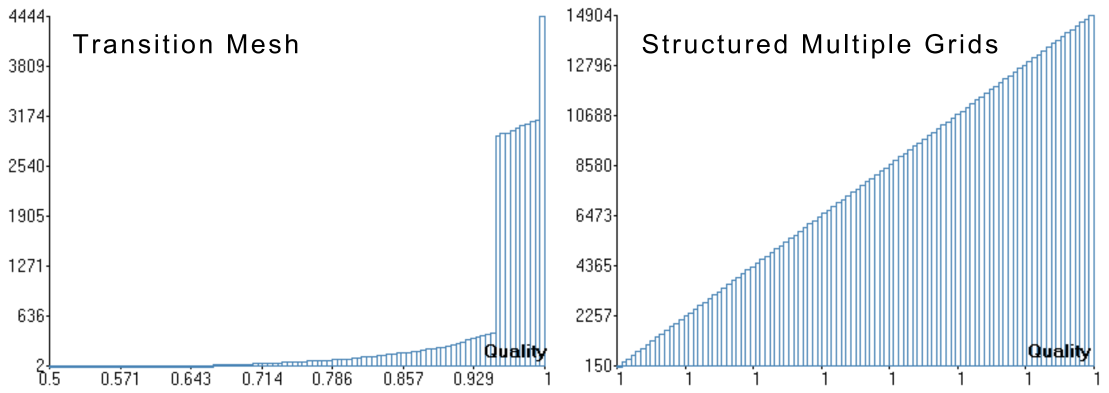


Figure 6-13: Mesh quality

In Figure 6-13 we compare the overall mesh quality for both – transition mesh and structured multiple grids. The criterion 'Shape Quality' measures the likeness of an element to be a reference element (an equilateral triangle in the case of triangles, a regular tetrahedron in the case of tetrahedra, a square in the case of quadrilaterals and a cube in the case of hexahedra). Its value is 1 for a perfect element (reference element), and it decreases as the element degrades in shape in comparison with the reference element. A negative value would represent that the element has a negative Jacobian at some point. In Figure 6-13 we clearly see that the mesh quality quickly degrades for elements within transition zone. Although it allows resolving critical regions in the vicinity of the crack, element quality can adversely affect the quality of solution in this region. Structured multiple grids, on other hand maintain a consistent element shape quality of 1, allowing the user to resolve critical regions in the vicinity of crack whilst achieving optimum shape quality.

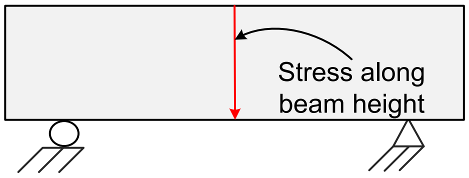


Figure 6-14: Stress as a function of space

Figure 6-15 and Figure 6-16 show the Von Mises Stress and longitudinal stress as a function of space at the center of the beam and along its height, See Figure 6-14. Structured MGMT clearly shows a smoother stress distribution around the crack tip with a maximum Von Mises stress of 343.19 N/m^2 versus 272.24 N/m^2 for transition mesh, and a maximum longitudinal stress of 353.69 N/m^2 versus 310.65 N/m^2 for transition mesh.

Finally, Figure 6-17, Figure 6-18 and Figure 6-19 show the overall (across the beam) distribution of Von Mises stress, longitudinal and lateral displacement at t = 0.1s.

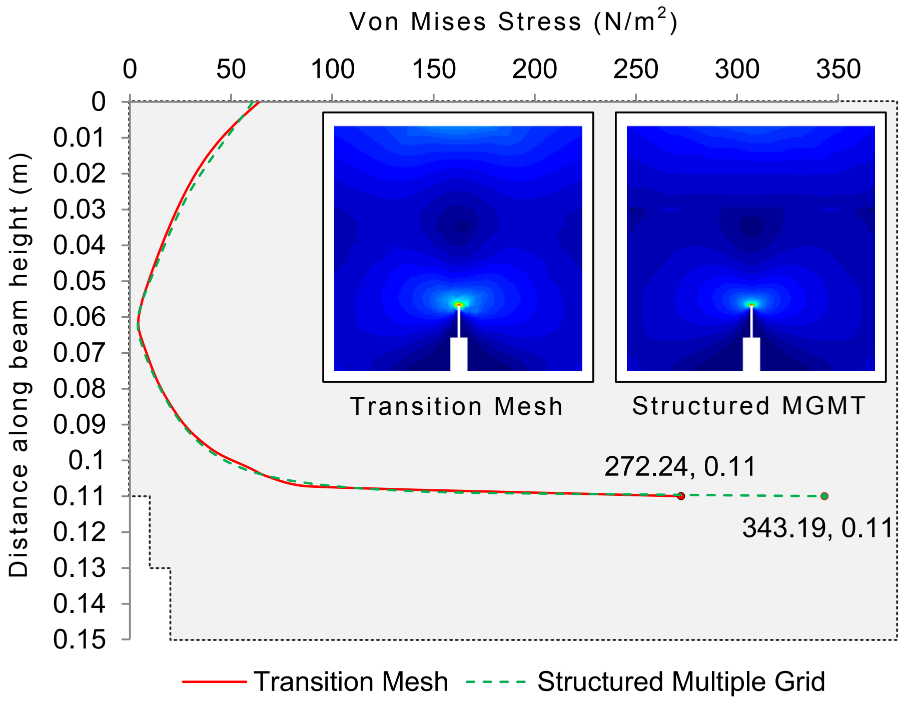


Figure 6-15: Von Mises Stress (at t = 0.1s) as a function of beam height

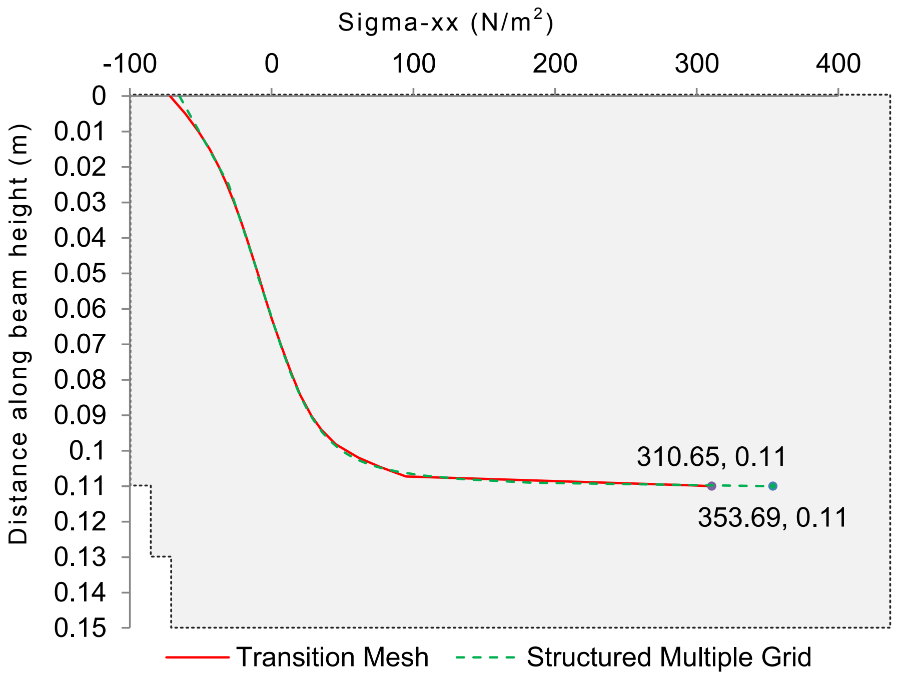


Figure 6-16: Longitudinal Stress (at t = 0.1s) as a function of beam height

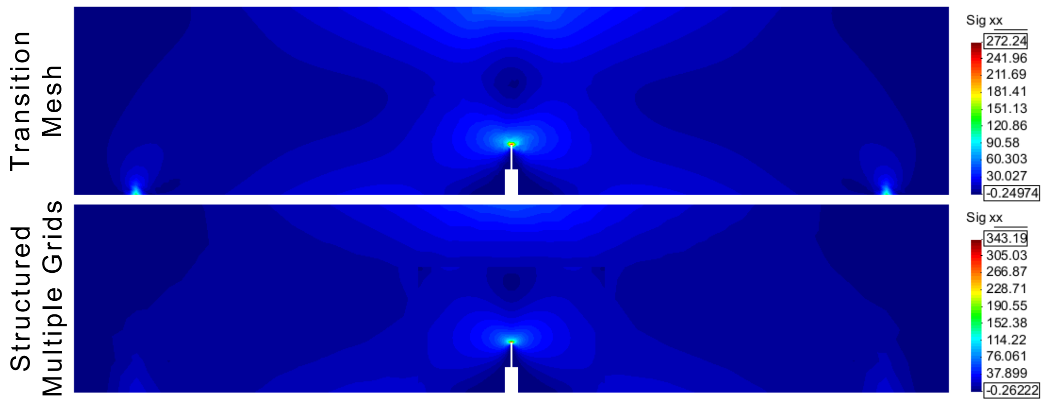


Figure 6-17: Von Mises stress distribution t = 0.1s

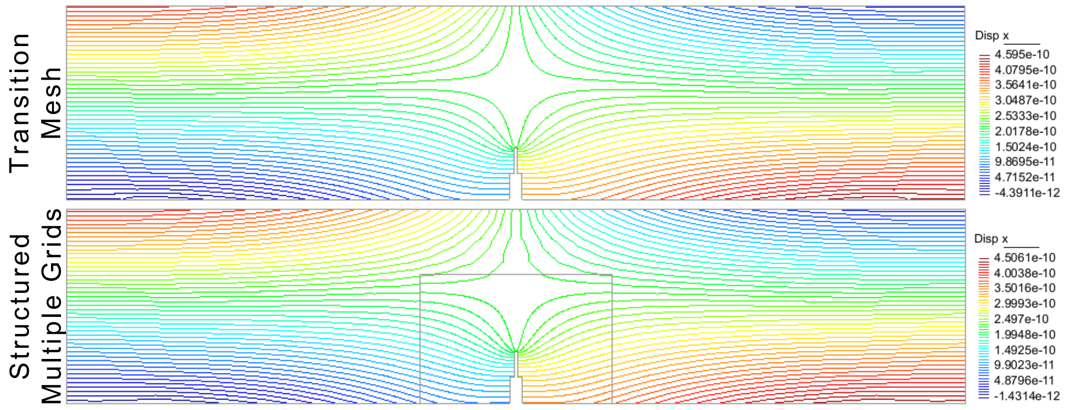


Figure 6-18: Contour lines for Displacement-x at t = 0.1s

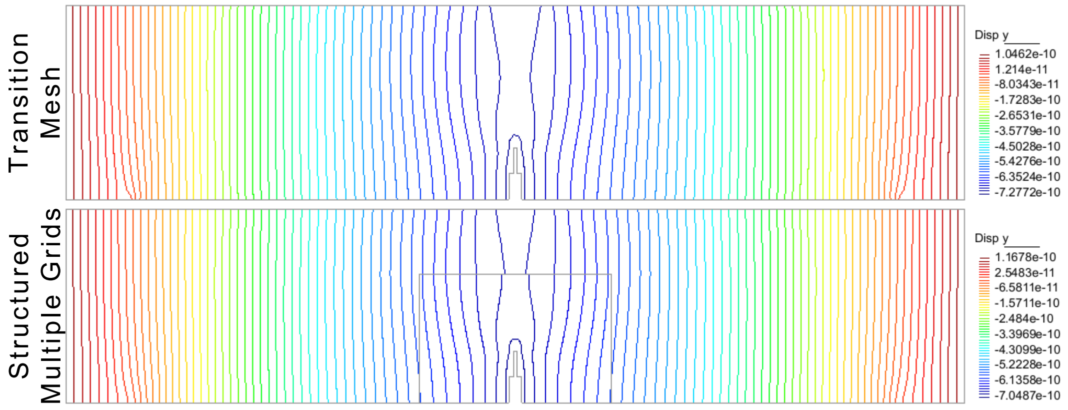
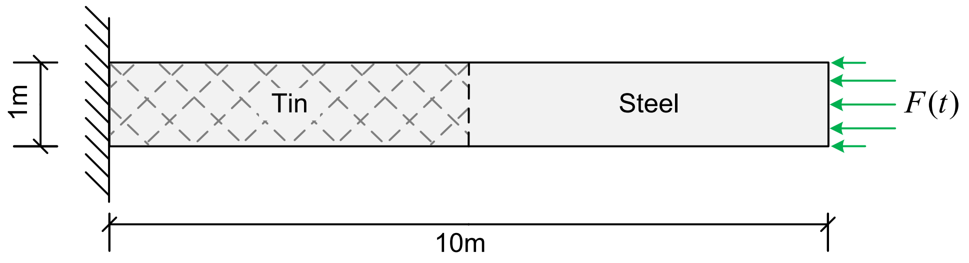


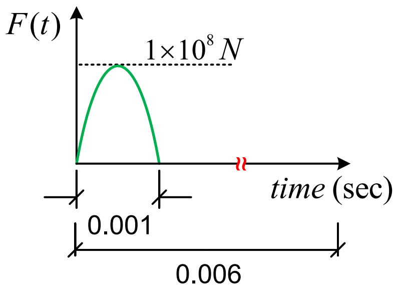
Figure 6-19: Contour lines for Displacement-y at t = 0.1s

6.2 Example 2: Analysis of Heterogeneous Structural Systems

Most structural systems used in contemporary mechanical and civil engineering applications are heterogeneous (composites) and multicomponent in nature. An essential characteristic feature of such multi-material, multicomponent systems is that structural waves travel at different speeds through different mediums with possible reflection at the material interface. Accordingly, accurate and efficient numerical modeling of wave propagation in heterogeneous media is important for several applications. Following example shows the fundamental limitations of conventional uniform grid uniform time-scale FEM and demonstrates the relative advantages in using MGMT Method.



(a) Domain under consideration



(b) Transient impact load

Figure 6-20: Heterogeneous material system

Figure 6-20 shows an overview of a heterogeneous domain under analysis. It consists of a 2D cantilever beam subjected to longitudinal impact loading of sinusoidal form. The beam is divided halfway across the length and consists of Steel and Tin on either side of the dividing interface. Mechanical properties for these materials, Modulus of Elasticity (E), Poisson's Ratio (v) and Mass Density (ρ) are listed in Table 6-4. Ignoring Poisson's Ratio, wave speed of the material is calculated as $c = \sqrt{E/\rho}$ and is also listed in Table 6-4.

Table 6-4: Mechanical properties for Steel and Tin

	E (N/m^2)	v	ρ (Kg/m^3)	c (m/sec)
Steel	207x10^9	0.3	7830	5141.67
Tin	47x10^9	0.33	7280	2540.87

In order the model the longitudinal wave as accurately as possible, the domain under analysis needs to be adequately discretized, both in space and time. A span of 10 elements seems appropriate to spread the impact load across the width of the beam, accordingly 4 node quadrilateral elements of width 0.1m are selected. In order to achieve structured mesh across the domain, as well as to maintain the aspect ratio of individual elements, the length of the element is also selected as 0.1m. Accordingly, as a preliminary spatial discretization, a grid consisting of 100x10 elements is selected. An appropriate time-step for each material media is then obtained by dividing the characteristic length of the element with respective material wave speed.

Table 6-5: Time-step to accurately resolve structural wave propagation

	Critical time-step ($\Delta t_c = H/c$)	Selected time-step ($\Delta t < \Delta t_c$)
Steel	$0.1/5141.67 = 1.94\times10^{-5}$	1×10^{-5}
Tin	$0.1/2540.87 = 3.93\times10^{-5}$	3×10^{-5}

It is evident from Table 6-5 that different material systems, depending on their mechanical properties, require different time-steps in order to accurately resolve structural wave propagation across the domain. Traditional FE approach requires uniform time-scale discretization and consequently the solution algorithm is forced to use the lowest stable time-step for the entire range of material systems under analysis. In this case, the global time-step would be limited to $1\text{x}10^{-5}$ sec. Clearly, this is computationally inefficient since half the domain (Tin) does not require this small time-step to accurately capture the wave.

As an alternative, elements from the steel domain may be appropriately scaled to match the larger time-step ($3\text{x}10^{-5}$ sec); however this introduces non-conforming connections at the interface which cannot be handled by traditional FE techniques. Usual FEM is therefore incapable of resolving heterogeneous material systems according to their time-step requirements, hence resulting in lower computational efficiency and compromised solution accuracy. In order to demonstrate relative advantages in using MGMT Method, whilst overcoming aforementioned limitations, following cases will be considered under this example:

1) UGUT – Uniform grid uniform time-step with lowest stable time-step
2) MGMT1 – Uniform gird with material appropriate time-steps
3) MGMT2 – Uniform time-step with appropriately scaled elements

Following tables and accompanying figure give a summary of simulation parameters for various cases analyzed under this example.

Table 6-6: UGUT heterogeneous material system – Simulation parameters

Case	Grid spacing (H)	Newmark parameters	Time-step (ΔT)
UGUT	0.1	$\beta = 0.25$, $\gamma = 0.5$ (Implicit)	$1x10^{-5}$

Table 6-7: MGMT1 heterogeneous material system – Simulation parameters

Number of sub-domains = 2 Total number of interface DOF = 22

Global time-step $\Delta T = 3x10^{-5}$ Time-step ratio (ξ) = 1, 3

Sub-domain	Grid spacing (H)	Newmark parameters	Time-step (Δt)
Ω1 (Tin)	0.1	$\beta = 0.25$, $\gamma = 0.5$ (Implicit)	$3x10^{-5}$
Ω2 (Steel)	0.1	$\beta = 0.25$, $\gamma = 0.5$ (Implicit)	$1x10^{-5}$

Table 6-8: MGMT2 heterogeneous material system – Simulation parameters

Number of sub-domains = 2 Total number of interface DOF = 14

Global time-step $\Delta T = 3x10^{-5}$ Time-step ratio (ξ) = 1, 1

Sub-domain	Grid spacing (H)	Newmark parameters	Time-step (Δt)
Ω1 (Tin)	0.1	$\beta = 0.25$, $\gamma = 0.5$ (Implicit)	$3x10^{-5}$
Ω2 (Steel)	0.6666	$\beta = 0.25$, $\gamma = 0.5$ (Implicit)	$3x10^{-5}$

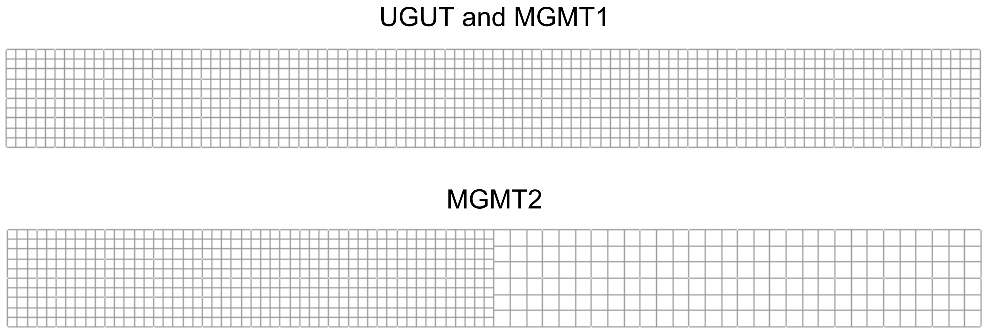


Figure 6-21: Heterogeneous material system – Domain grids

Initial comparison is made by plotting global energies, Figure 6-22, and by measuring relative RMSE and NRMS errors, Table 6-9, in order to ensure stability and overall conformance between global energies.

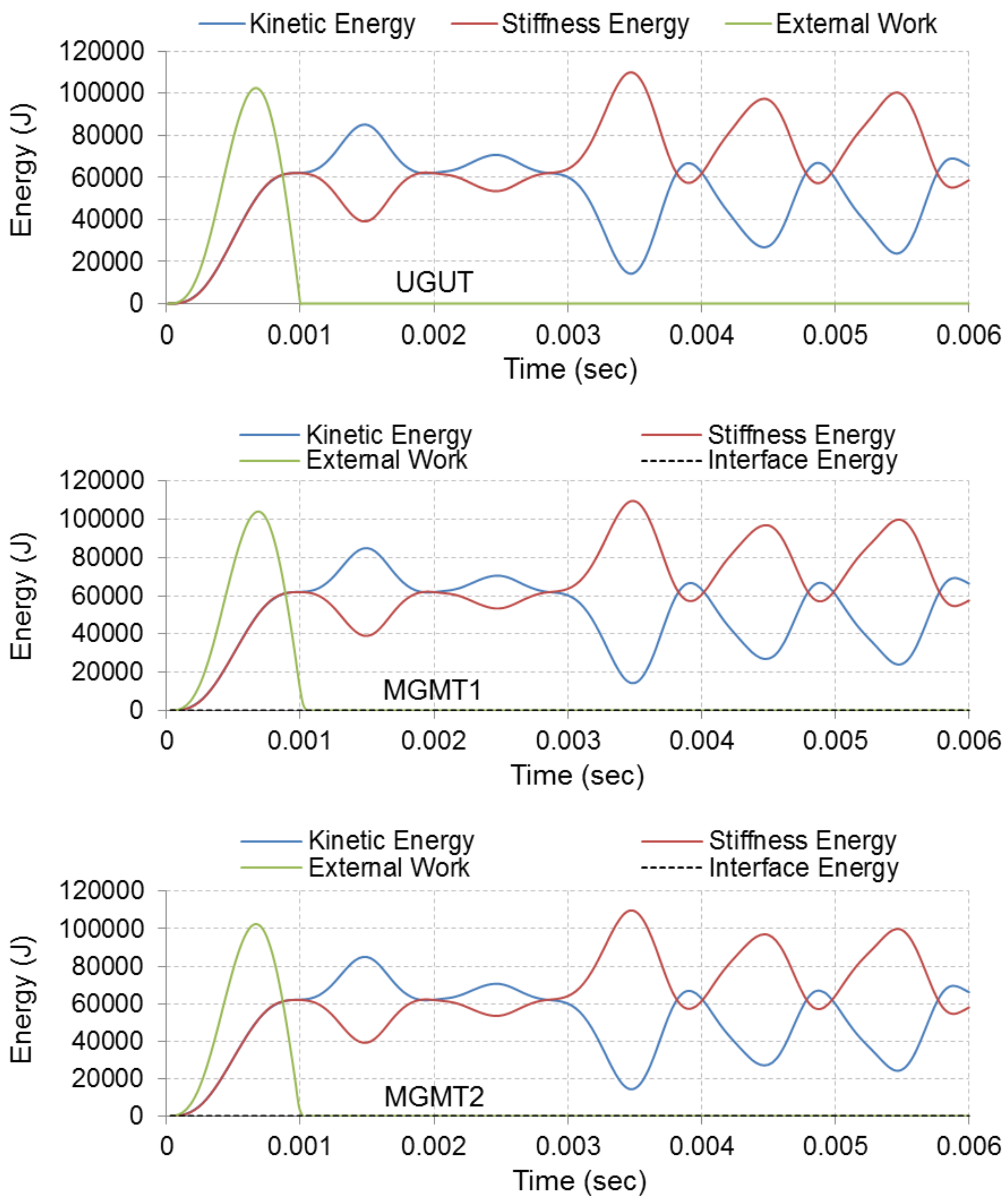


Figure 6-22: Global energies

Table 6-9: RMSE and NRMSE (%). Variable = Global energies

	Kinetic Energy	Stiffness Energy
UGUT v/s MGMT1	1109.42 (1.30%)	1108.30 (1.00%)
UGUT v/s MGMT2	297.58 (0.35%)	304.60 (0.27%)

Table 6-10: Mean variance in augmented (total) interface energy

	Total Interface Energy
MGMT1	1.31×10^{-8} (@ 200)
MGMT2	1.07×10^{-8} (@ 200)

Table 6-9 shows very good conformance between global energies and validates MGMT stability by ensuring negligible interface energy accumulation/dissipation, Table 6-10.

Subsequent results include comparison of stress wave (Sigma-xx), Figure 6-24, and displacement wave, Figure 6-25, as a function of time, as it propagates across the length of the beam. Measurements are made at 4 different locations as shown in Figure 6-23. Resulting RMSE and NRMSE errors are listed in Table 6-11 and Table 6-12 respectively.

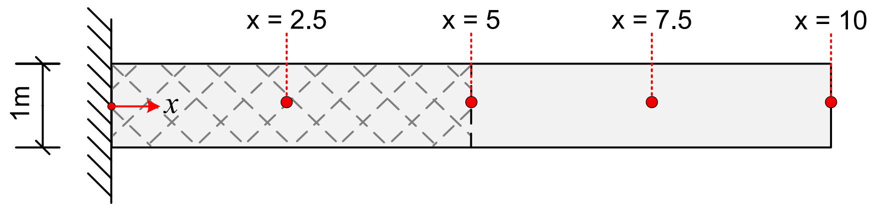


Figure 6-23: Heterogeneous material system – Longitudinal stress and displacement measurement locations across the length of beam

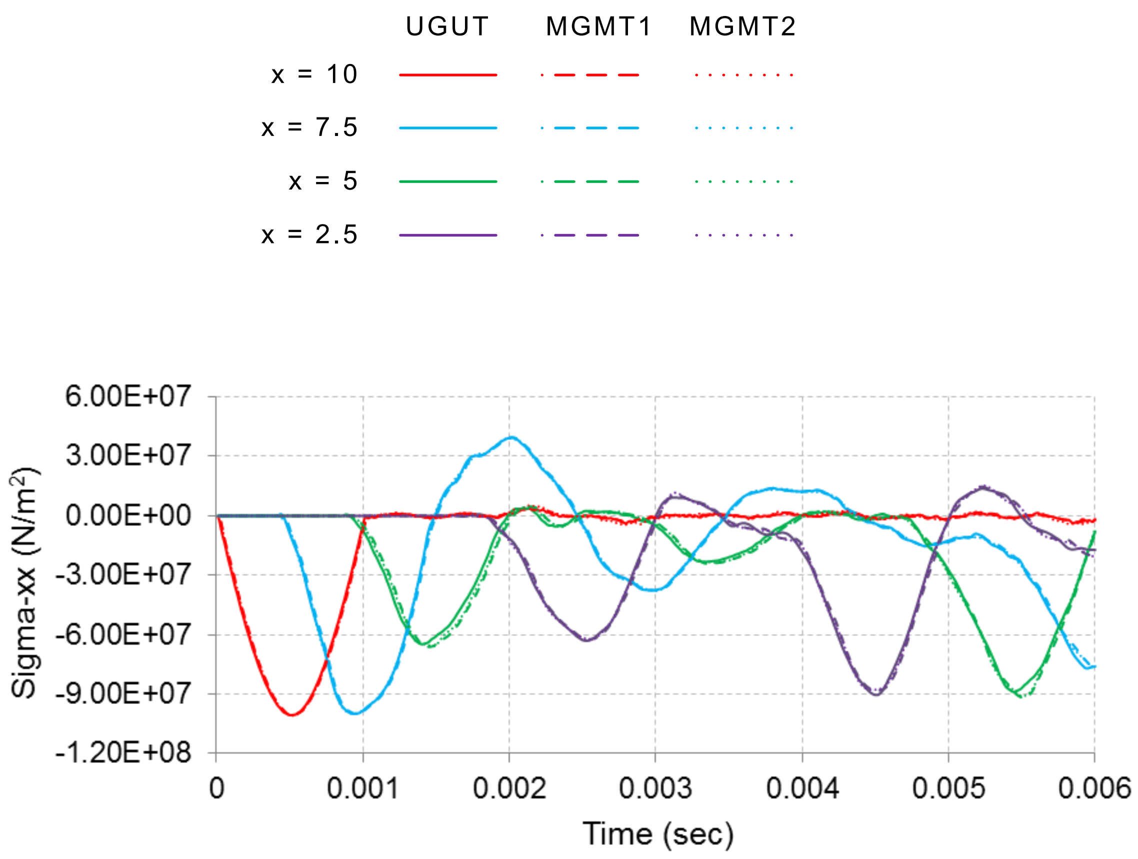


Figure 6-24: Heterogeneous material system – Longitudinal stress (Sigma-xx) as a function of time

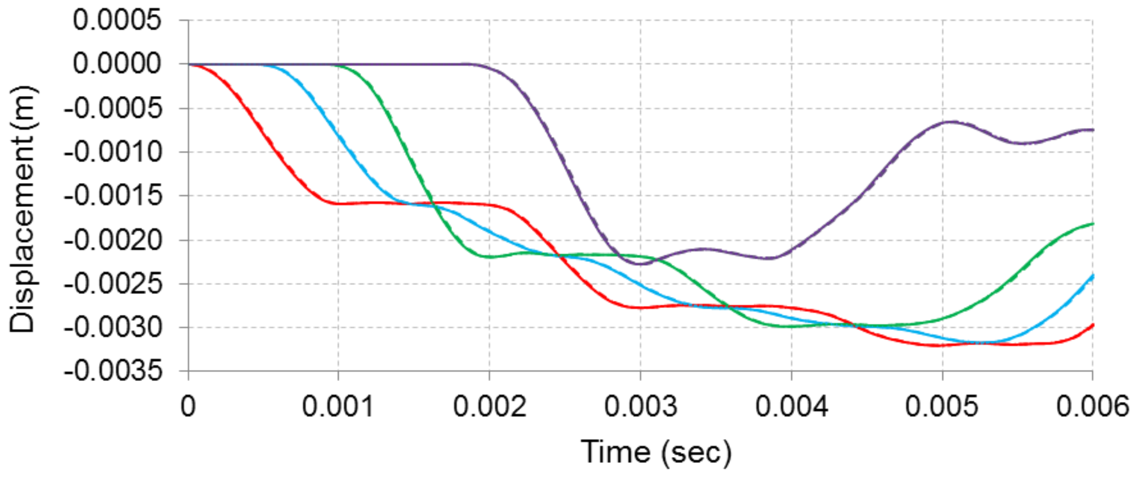


Figure 6-25: Longitudinal displacement (U-x) as a function of time

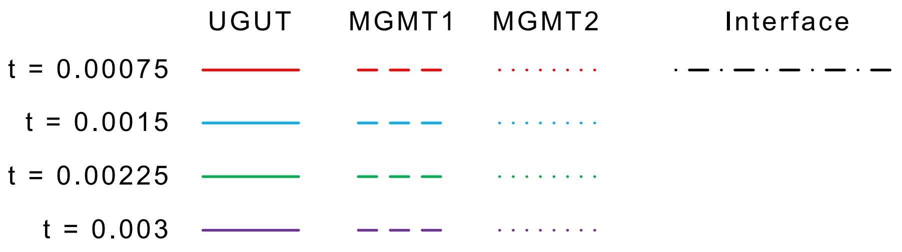
Table 6-11: RMSE and NRMSE (%). Variable = Longitudinal stress (Sigma-xx) as function of time

	Stress (Sigma-xx)	
	MGMT1	MGMT2
x = 10	973568.46 (0.93%)	1010704.63 (0.96%)
x = 7.5	1462419.42 (1.04%)	868346.19 (0.622%)
x = 5	2684557.54 (2.90%)	2030655.68 (2.19%)
x = 2.5	1740279.38 (1.67%)	1289447.07 (1.23%)

Table 6-12: RMSE and NRMSE (%). Variable = Longitudinal displacement (U-x) as function of time

	Displacement-x	
	MGMT1	MGMT2
x = 10	$1.10\text{x}10^{-6}$ (0.34%)	$3.20\text{x}10^{-6}$ (0.1%)
x = 7.5	$1.10\text{x}10^{-6}$ (0.34%)	$3.02\text{x}10^{-6}$ (0.09%)
x = 5	$1.39\text{x}10^{-6}$ (0.46%)	$3.41\text{x}10^{-6}$ (0.11%)
x = 2.5	$1.61\text{x}10^{-6}$ (0.70%)	$5.38\text{x}10^{-6}$ (0.23%)

Following results include plots of longitudinal stress, Figure 6-26, and displacement, Figure 6-27, as a function of space, measured at 4 different time instants t = 0.00075, 0.0015, 0.00225 and 0.003 sec. Subsequent are the contour plots for longitudinal stress with global deformation, Figure 6-28 through Figure 6-31.



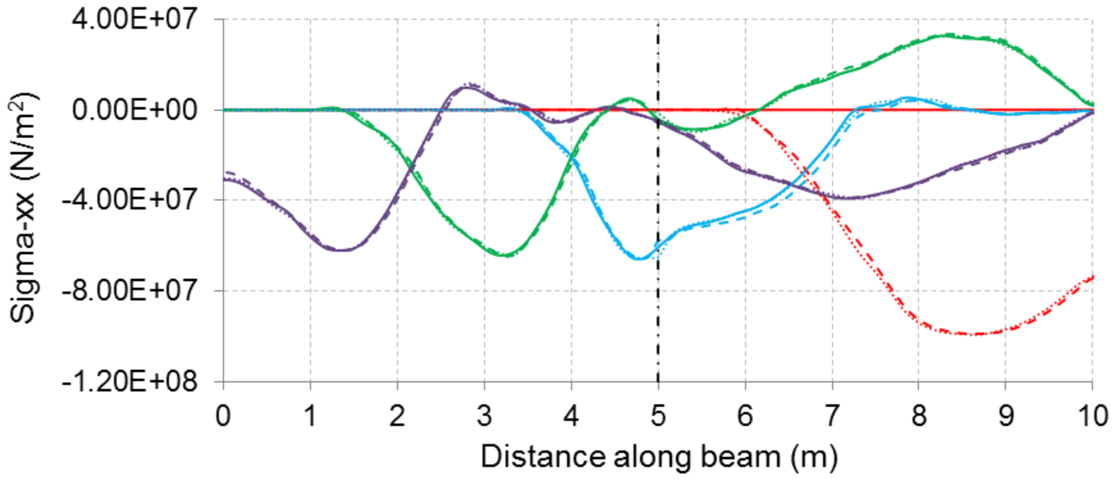


Figure 6-26: Longitudinal stress (Sigma-xx) as a function of space

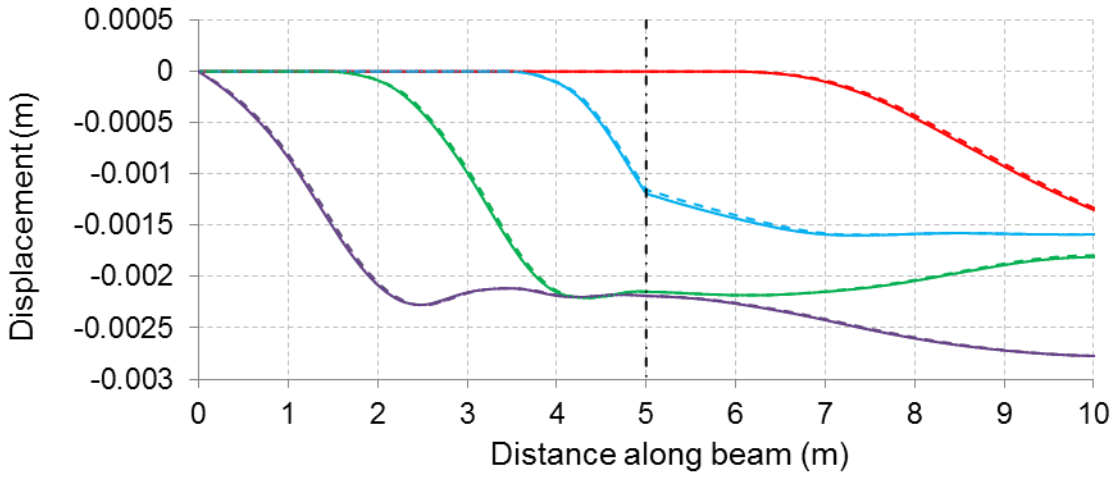


Figure 6-27: Longitudinal displacement (U-x) as a function of space

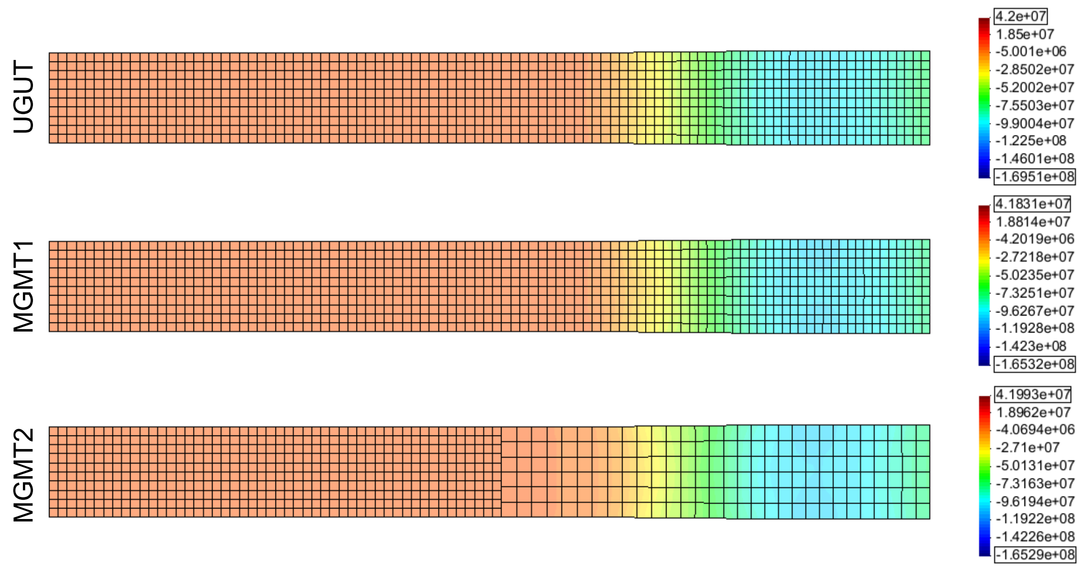


Figure 6-28: Longitudinal stress wave at t = 0.00075s

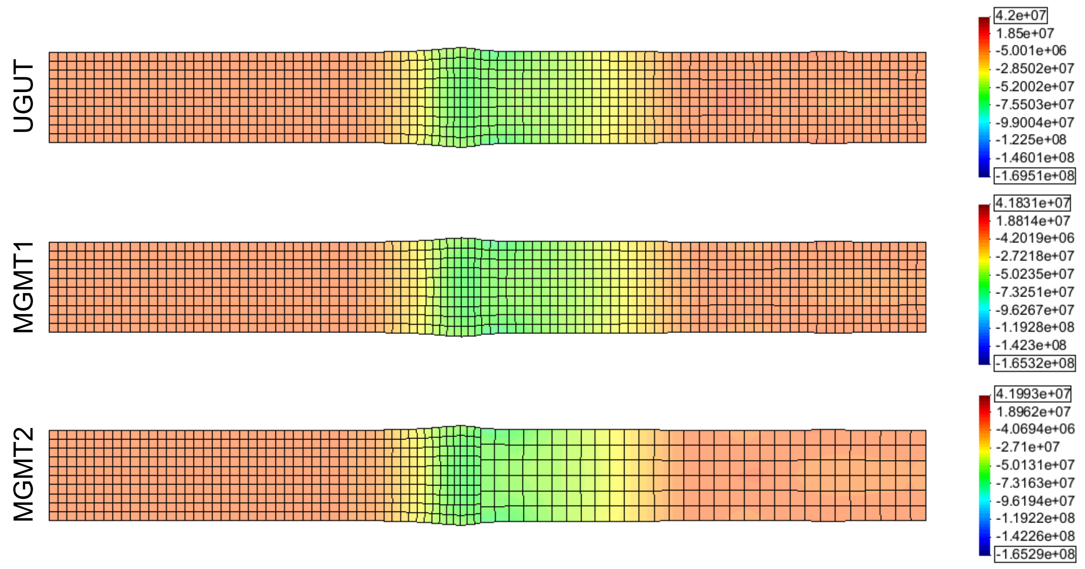


Figure 6-29: Longitudinal stress wave at t = 0.0015s

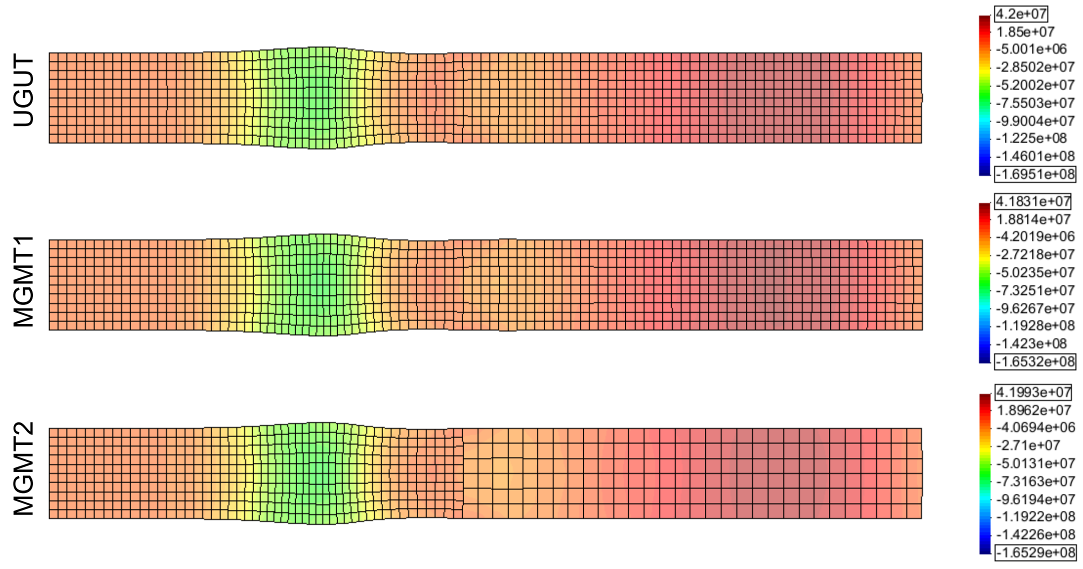


Figure 6-30: Longitudinal stress wave at t = 0.00225s

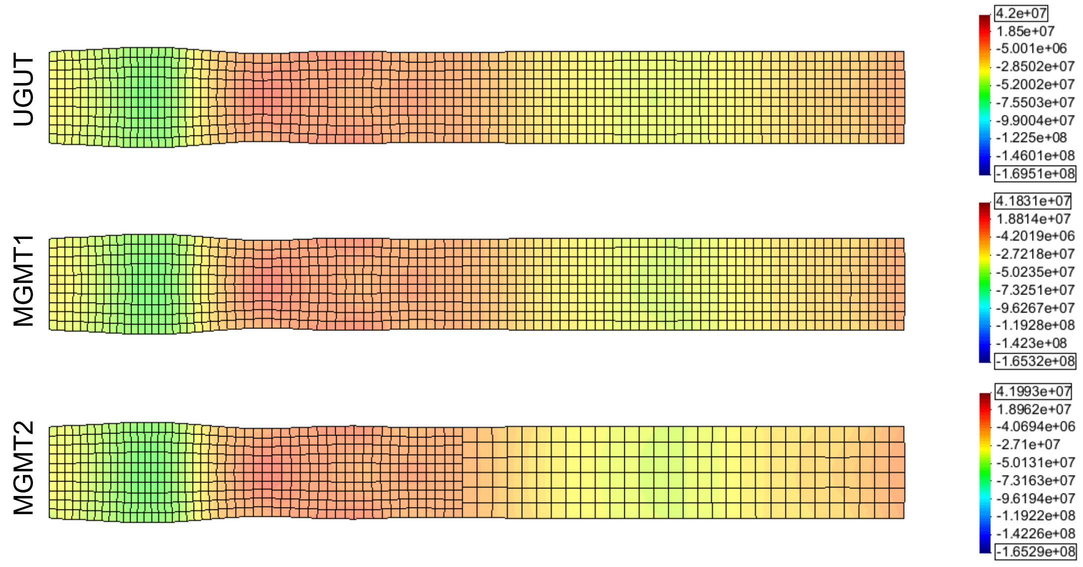


Figure 6-31: Longitudinal stress wave at t = 0.003s

Results show very good conformance between UGUT and MGMT cases. Stress wave propagates seamlessly across the length of the beam without any significant change in amplitude or phase. Table 6-11 shows that maximum NRMSE error in longitudinal stress occurs at x = 5 which is 2.9% for UGUT v/s MGMT1 and 2.19% for UGUT v/s MGMT2. However x = 5 not only represents the interface between Steel and Tin sub-domains but also represents an interface between multiple grid (MGMT2) and multiple time-scale (MGMT1) sub-domains; accordingly certain discontinuity in stress is expected as a result of multiscale computations. Corresponding errors for longitudinal displacement are much lower, 0.46% for UGUT v/s MGMT1 and 0.11% for UGUT v/s MGMT2, and hence represent smoother continuity in displacement across the interface. Longitudinal stress and displacement, plotted as function of space, also show very good conformance with UGUT case, ensuring no delay in communicating information across heterogeneous MGMT sun-domains. Table 6-13 shows the comparison of computational resources utilized in analyzing respective cases. MGMT1 with uniform grid and material specific time-step reduces the computation time by 9.3%, whereas MGMT2 with uniform time-step and appropriately scaled element sizes reduces the computation time by 67.2%.

Table 6-13: Example 2 – Comparison of computational resources

	Nodes	Elements	Number of equations	Skyline storage	Solution time (sec)
UGUT	1111	1000	2200	59536	22.68
MGMT1	1122 ▲ 0.99%	1000 ▲▼	2222 ▲ 1.00%	63849 ▲ 7.24%	20.57 ▼ 9.30%
MGMT2	778 ▼ 29.97%	680 ▼ 32%	1534 ▼ 30.27%	39681 ▼ 33.34%	7.44 ▼ 67.19%

6.3 Example 3: Steel Girder Subjected to Impulse Loading

In this example we will analyze a large compound structure used for building bridges and the frameworks of large buildings. This girder, as shown in Figure 6-32, is analyzed for its response under the action of a sudden impulse loading applied at the mid-section connection. Since the load signifies a large magnitude force applied over a short duration of time, the girder is expected to represent an impact response and accordingly, wave propagation is the primary concern in this example. In addition to modeling response under impulse loading, we would also like to analyze the global, structural dynamic behavior of the girder. Traditional FE approach in such scenarios would require small time-steps and preferably an explicit integration scheme in order to capture the wave dynamics; and limited by the inherent nature of single domain discretization, one would be forced to utilize a uniform grid uniform time-scale discretization throughout the analysis domain under consideration.

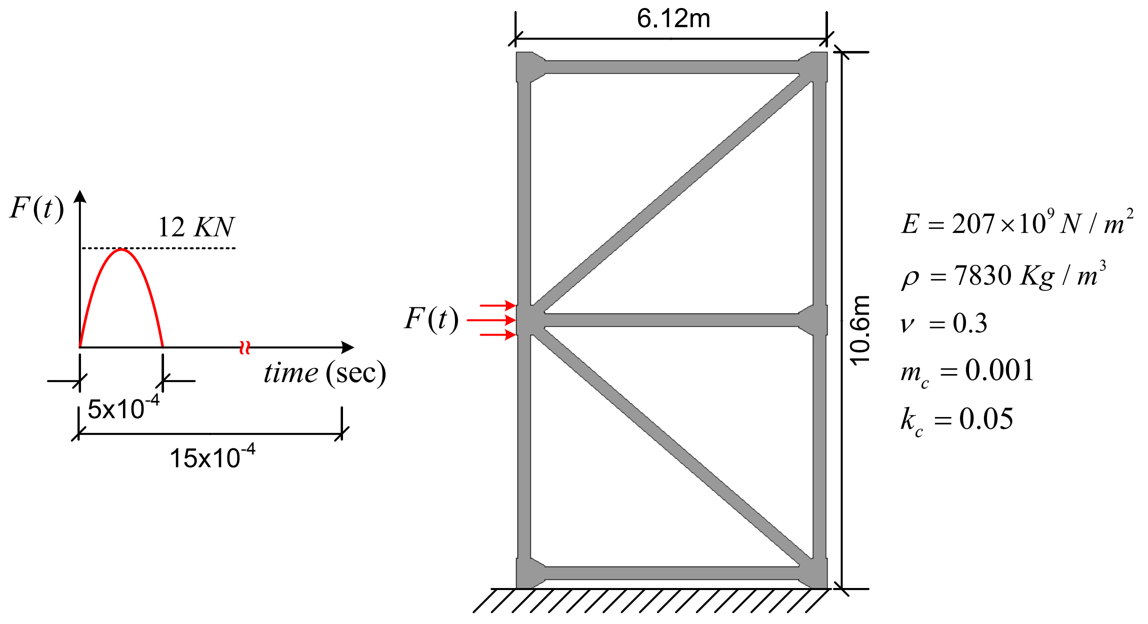


Figure 6-32: Steel girder subjected to short duration impulse loading

In this example, we will begin with UGUT discretization, Figure 6-33 and Table 6-14, that uses explicit time integration scheme and a domain wide spatial discretization of $\overline{H}$ = 0.025. We then selectively isolate the girder section that is subjected to impulse loading $(\Omega 2)$ and discretize it with same parameters as UGUT. We then employ two different discretizations within the remote region of the girder $(\Omega 1)$ with coarser grids and implicit time integration algorithms.

Resulting analysis domains and their respective simulation parameters are as shown in Figure 6-33 and Table 6-14. In either case, the domain is discretized using 4-node quadrilateral elements (2 DOF/node) with plane stress formulation and consistent mass matrix. Rayleigh damping is assumed in this example and the corresponding damping coefficients are: m_c = 0.001 for mass and k_c = 0.05 for stiffness. Isotropic linear elastic material properties with modulus of elasticity (E) = 2.07x10^{11} N/m^2, Poisson's ratio (ν) = 0.3 and mass density (ρ) = 7.83x10^3 Kg/m^3 are used.

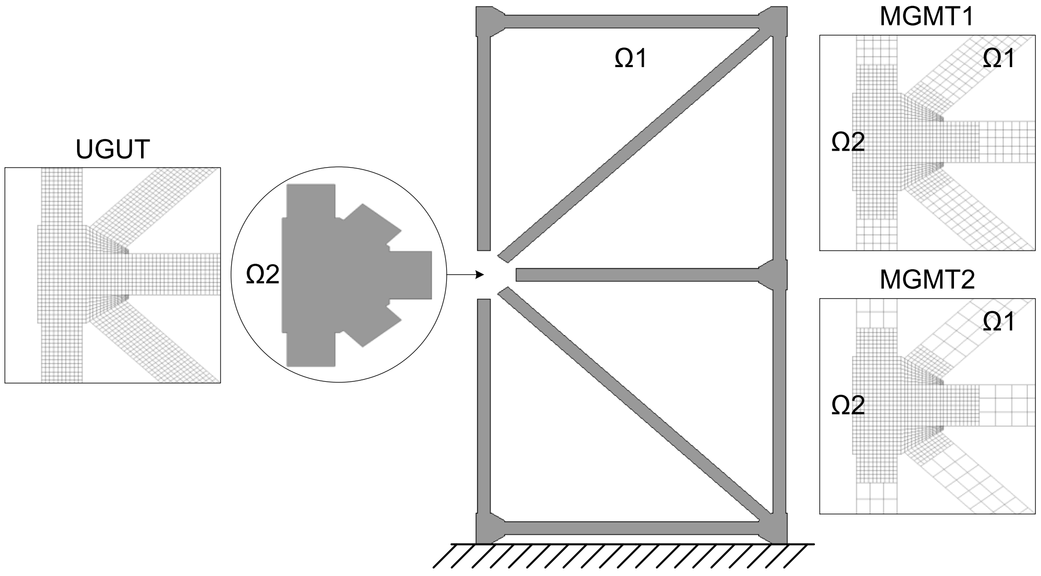


Figure 6-33: Analyzed cases: UGUT, MGMT1 and MGMT2

Table 6-14: Simulation parameters

	Grid spacing (H)	Interface DOF (λ)	Newmark parameters	Time-step (ΔT)	Time-step ratio (ξ)
UGUT	$\overline{H} = 0.025$	-	β=0.0, γ=0.5 (Explicit)	$1\text{x}10^{-6}$	-
MGMT1-Ω1	$\overline{H} = 0.05$	60	β=0.25, γ=0.5 (Implicit)	$2\text{x}10^{-4}$	1
MGMT1-Ω2	$\overline{H} = 0.025$		β=0.0, γ=0.5 (Explicit)	$1\text{x}10^{-6}$	20
MGMT2-Ω1	$\overline{H} = 0.083$	40	β=0.25, γ=0.5 (Implicit)	$2\text{x}10^{-4}$	1
MGMT2-Ω2	$\overline{H} = 0.025$		β=0.0, γ=0.5 (Explicit)	$1\text{x}10^{-6}$	20

We first look at the global energy distribution in order to establish numerical stability in UGUT and MGMT simulations. Figure 6-34 plots the evolution of kinetic, stiffness energy and external work for all three cases with augmented interface energy contributions for MGMT cases. As expected, the total internal energy (kinetic + stiffness + interface) in all three cases approaches zero as soon as the impulse load terminates. This is due to system damping present in the form of Rayleigh damping coefficients. From the evolution of interface energy we see that highest augmented energies (although several orders of magnitude smaller than domain energies) are generated at the peak of applied impulse load; but again: approach zero due to system damping. Results from Figure 6-34 and Table 6-15 show overall good conformance with UGUT and hence numerical stability is verified.

Table 6-15: RMSE and NRMSE (%). Variable = Global Energies

	Kinetic Energy	Stiffness Energy	Interface Energy
MGMT1	$5.77\text{x}10^{-7}$ (1.79%)	$7.10\text{x}10^{-9}$ (0.96%)	$2.91\text{x}10^{-17}$
MGMT2	$8.99\text{x}10^{-7}$ (1.86%)	$8.57\text{x}10^{-9}$ (1.16%)	$3.03\text{x}10^{-17}$

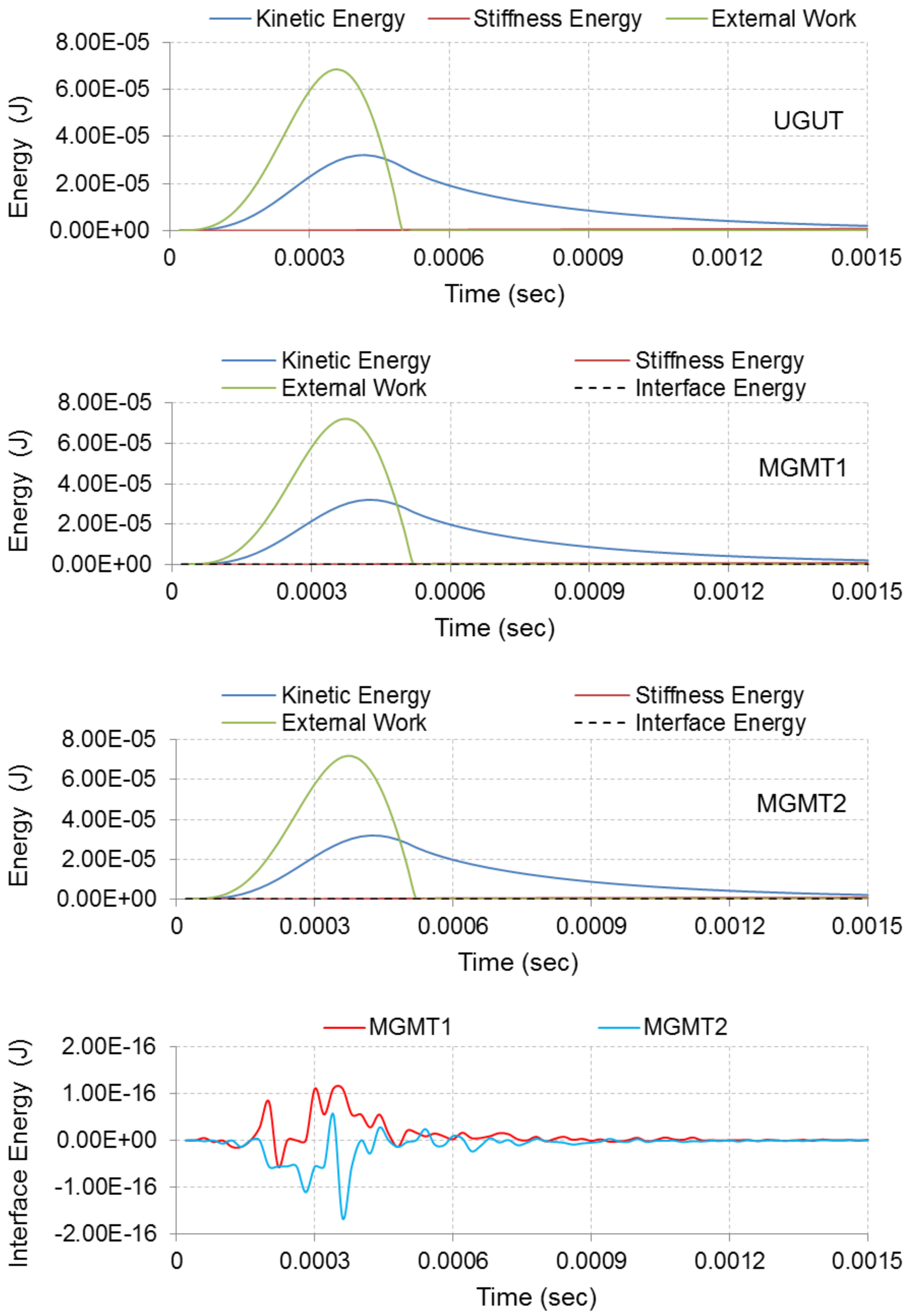


Figure 6-34: Global Energies

Subsequently, we will look at:

1) Distribution of Displacement-x and Displacement-y (Figure 6-36 and Figure 6-37) at peak impulse along with the deformed shape of the girder.
2) Distribution of Sigma-xx and Sigma-yy (Figure 6-38 and Figure 6-39) at simulation end time.
3) Evolution of Displacement-x and Sigma-xx as function of time (Figure 6-40 and Figure 6-41) along with respective RMSE and NRMSE. Recorded at A → x = 0.125m, B → x = 3m and C → x = 6m, See Figure 6-35.
4) Evolution of Displacement-x and Sigma-xx as function of space (Figure 6-42 and Figure 6-43) along with respective RMSE and NRMSE. Recorded along segment D, Figure 6-35 (b), and at t = 0.0005sec, 0.001sec and 0.0015sec.

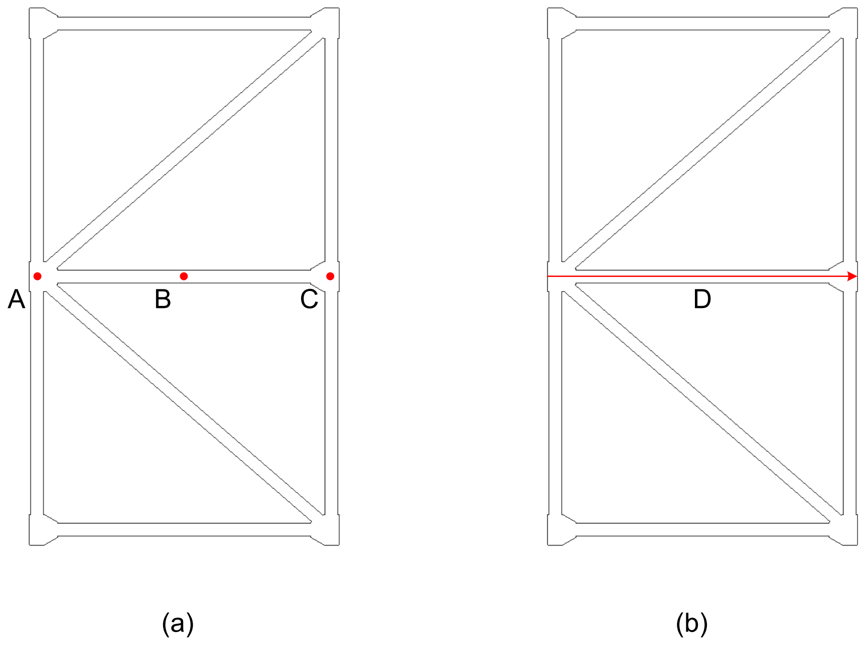


Figure 6-35: Analyzed result descriptions (a) Results measured at points A, B and C as functions of time (b) Results measured along segment D as functions of space

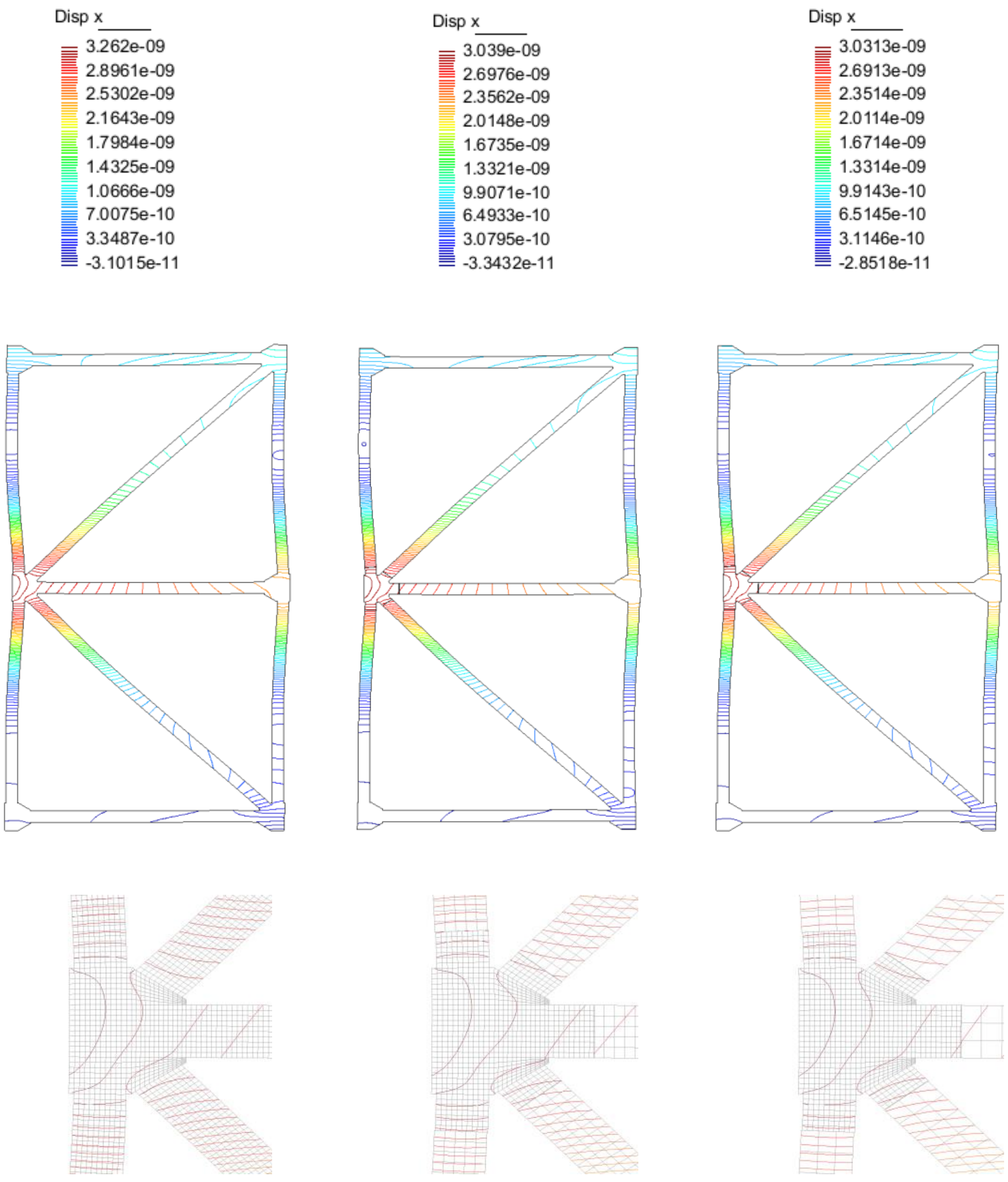


Figure 6-36: Global contour lines for Displacement-x at peak impulse. UGUT, MGMT1 and MGMT2 (L-R)

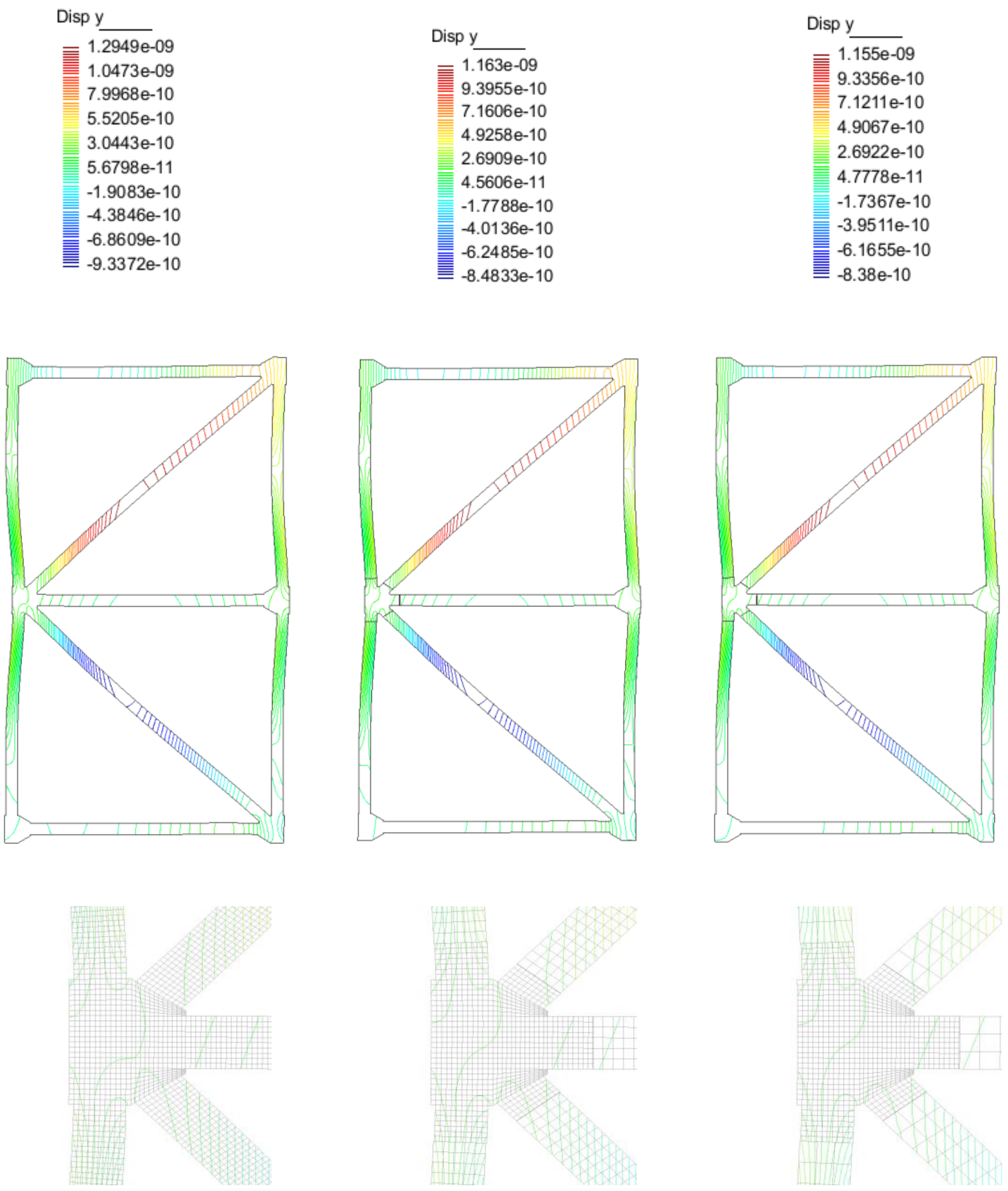


Figure 6-37: Global contour lines for Displacement-y at peak impulse. UGUT, MGMT1 and MGMT2 (L-R)

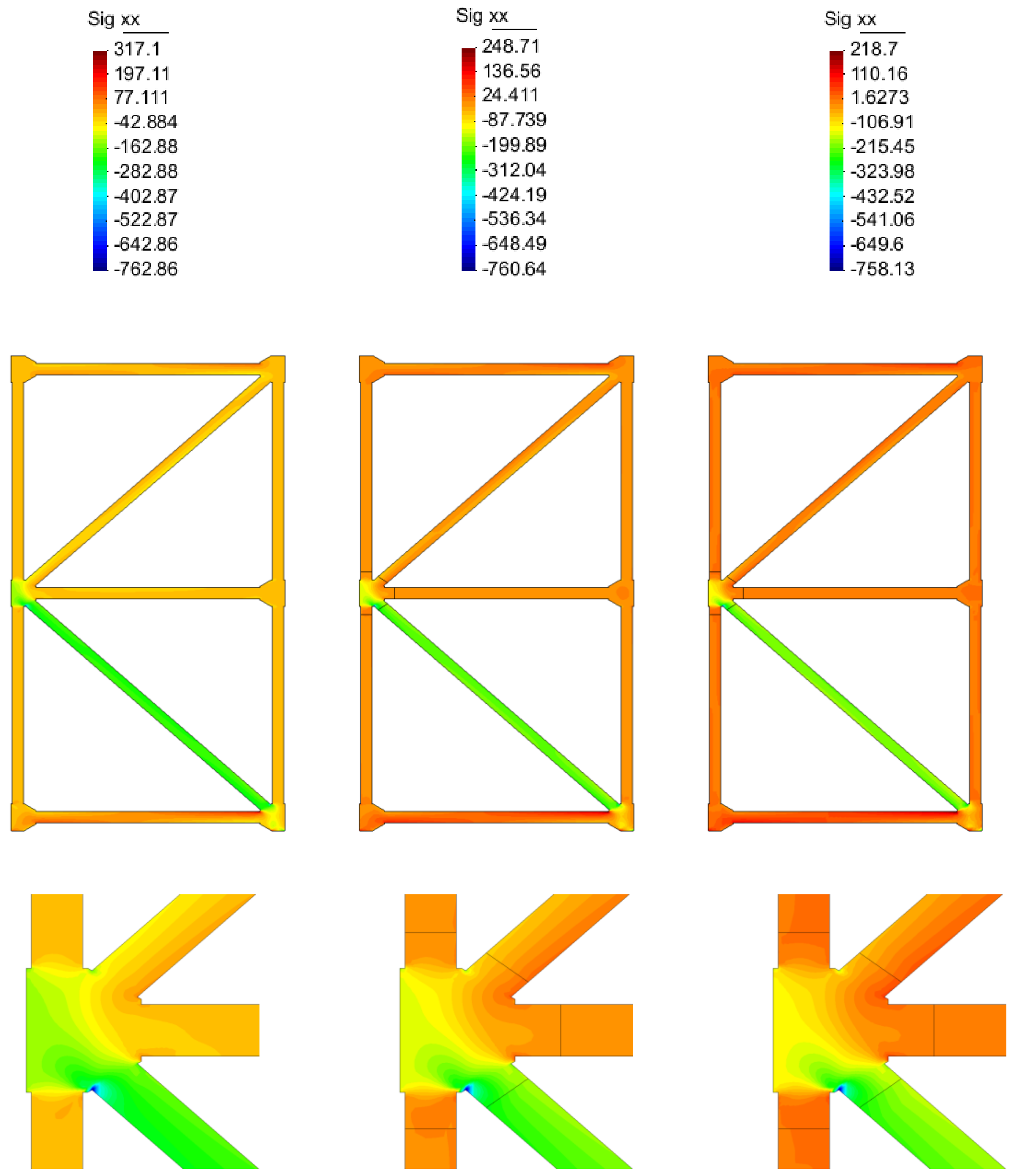


Figure 6-38: Global contour plots for Sigma-xx. UGUT, MGMT1 and MGMT2 (L-R)

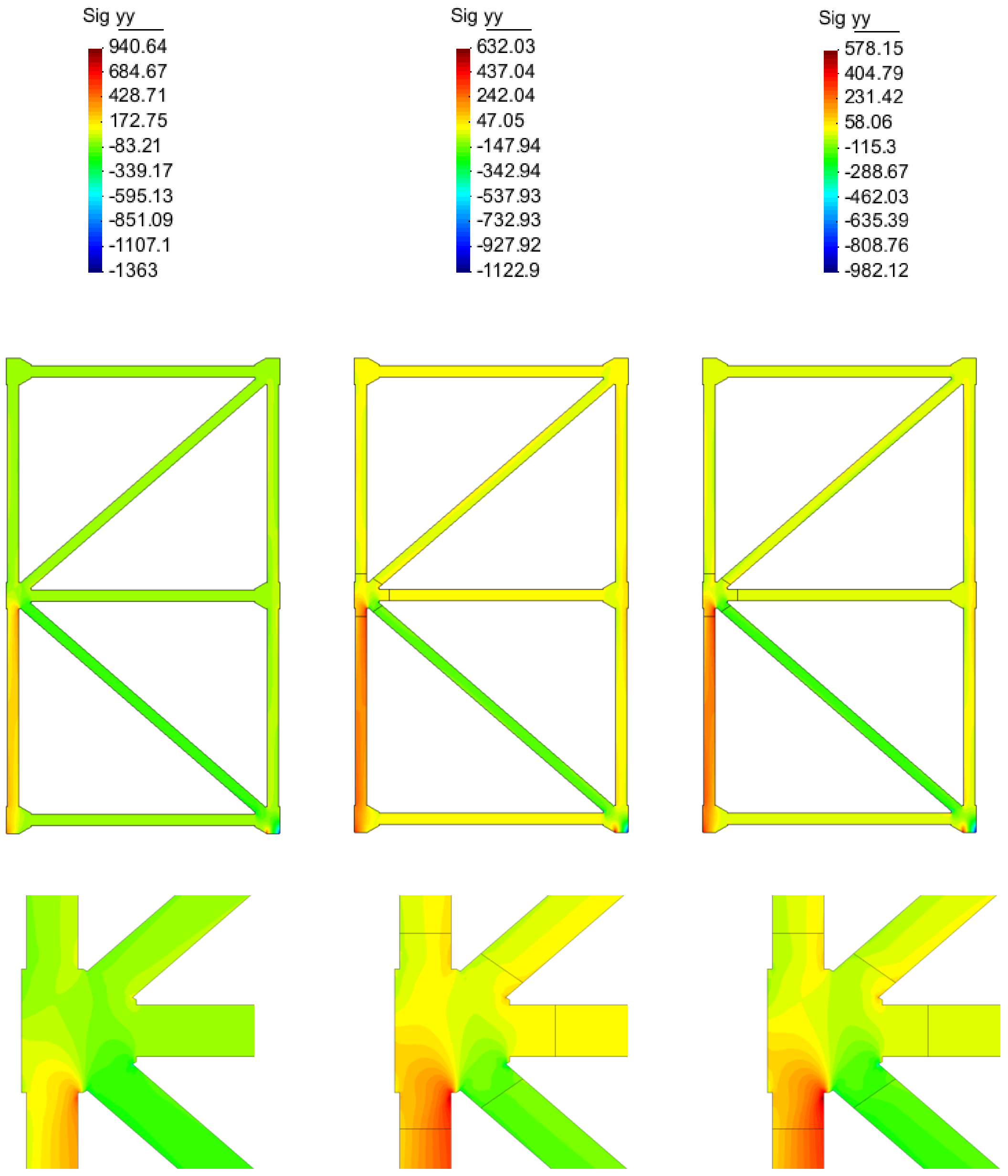


Figure 6-39: Global contour plots for Sigma-yy. UGUT, MGMT1 and MGMT2 (L-R)

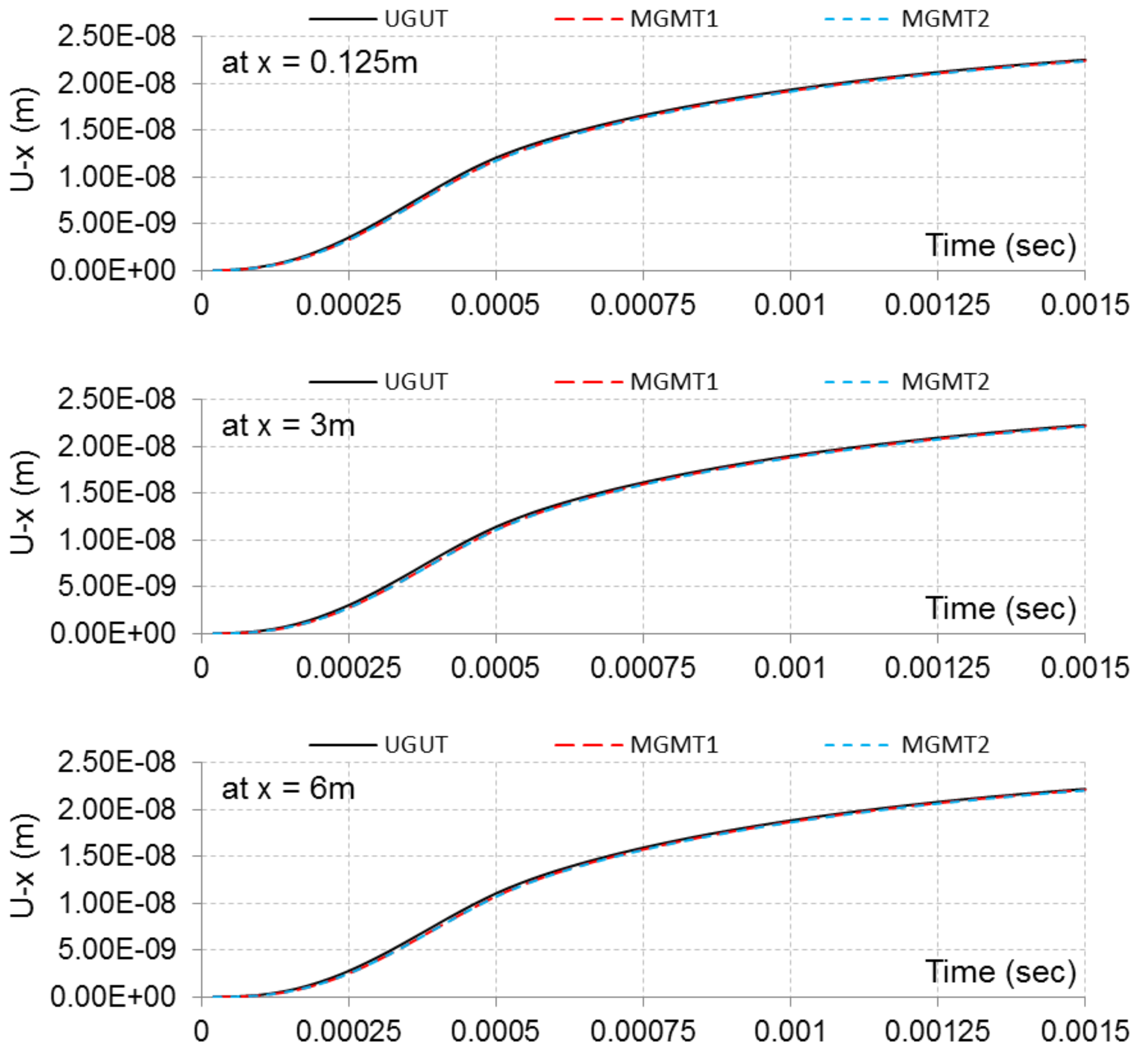


Figure 6-40: Displacement-x as a function of time

Table 6-16: RMSE and NRMSE (%). Variable = Displacement-x as a function of time

	MGMT1	MGMT2
x = 0.125m	$1.93\text{x}10^{-10}$ (0.85%)	$2.25\text{x}10^{-10}$ (1.00%)
x = 3m	$2.07\text{x}10^{-10}$ (0.92%)	$2.43\text{x}10^{-10}$ (1.09%)
x = 6m	$2.09\text{x}10^{-10}$ (0.94%)	$2.51\text{x}10^{-10}$ (1.13%)

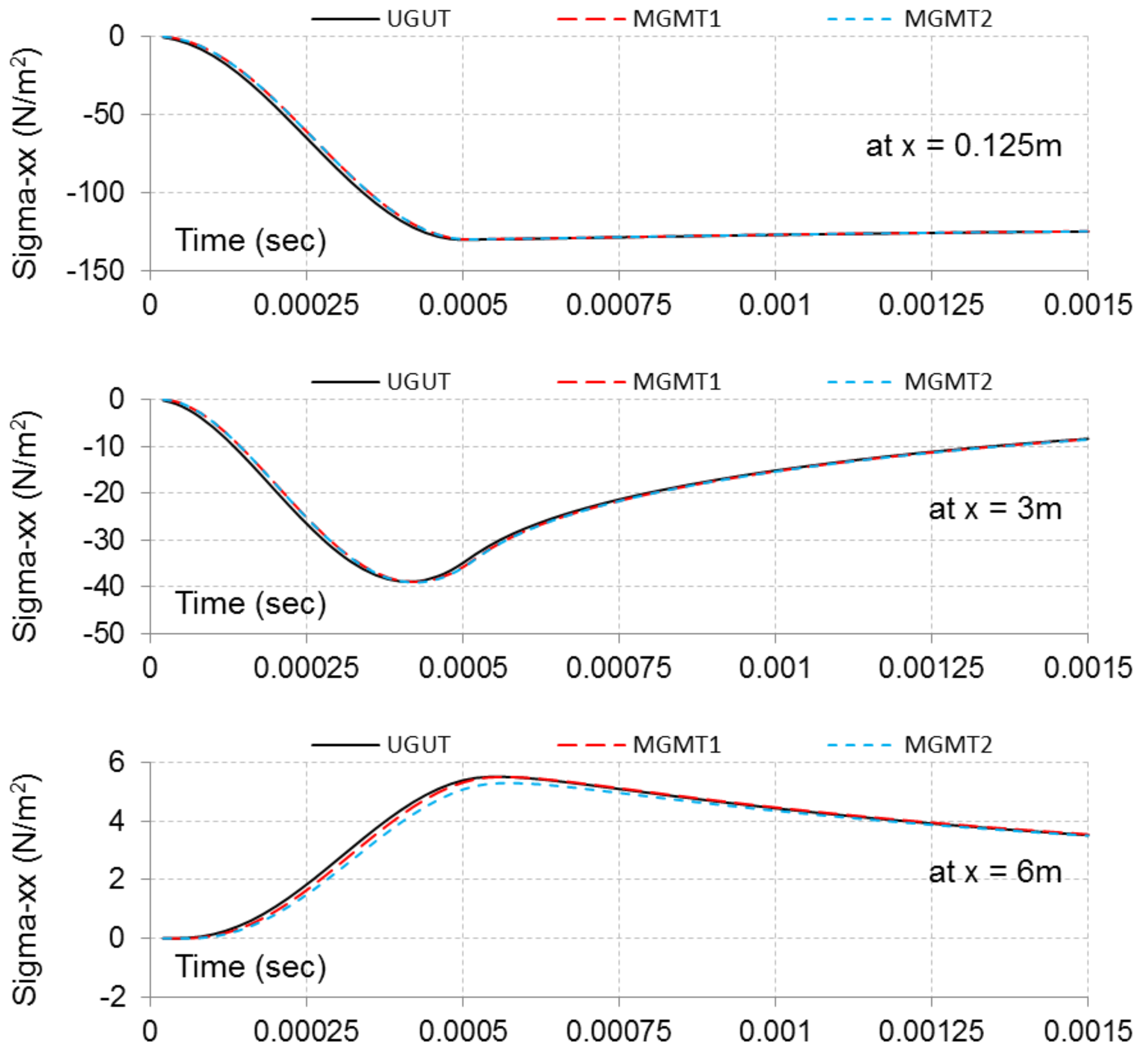


Figure 6-41: Sigma-xx as a function of time

Table 6-17: RMSE and NRMSE (%). Variable = Sigma-xx as a function of time

	MGMT1	MGMT2
x = 0.125m	1.70 (1.31%)	1.69 (1.30%)
x = 3m	0.62 (1.60%)	0.62 (1.62%)
x = 6m	$8.72\text{x}10^{-2}$ (1.58%)	0.202 (3.67%)

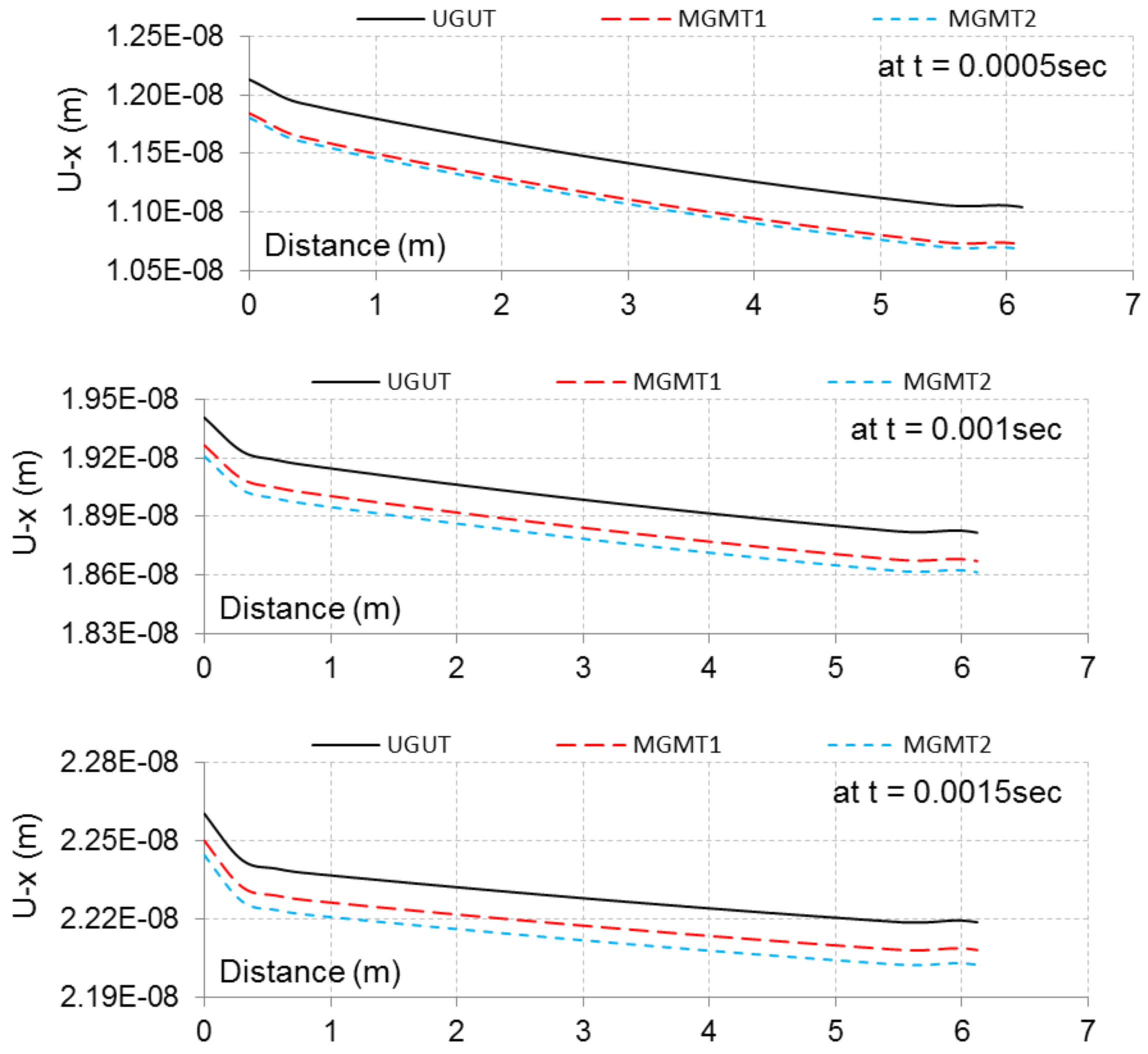


Figure 6-42: Displacement-x as a function of space

Table 6-18: RMSE and NRMSE (%). Variable = Displacement-x as a function of space

	MGMT1	MGMT2
t = 0.0005sec	3.17×10^{-10} (28.88%)	3.54×10^{-10} (32.23%)
t = 0.001sec	1.46×10^{-10} (24.47%)	2.00×10^{-10} (33.23%)
t = 0.0015sec	1.05×10^{-10} (26.46%)	1.73×10^{-10} (43.38%)

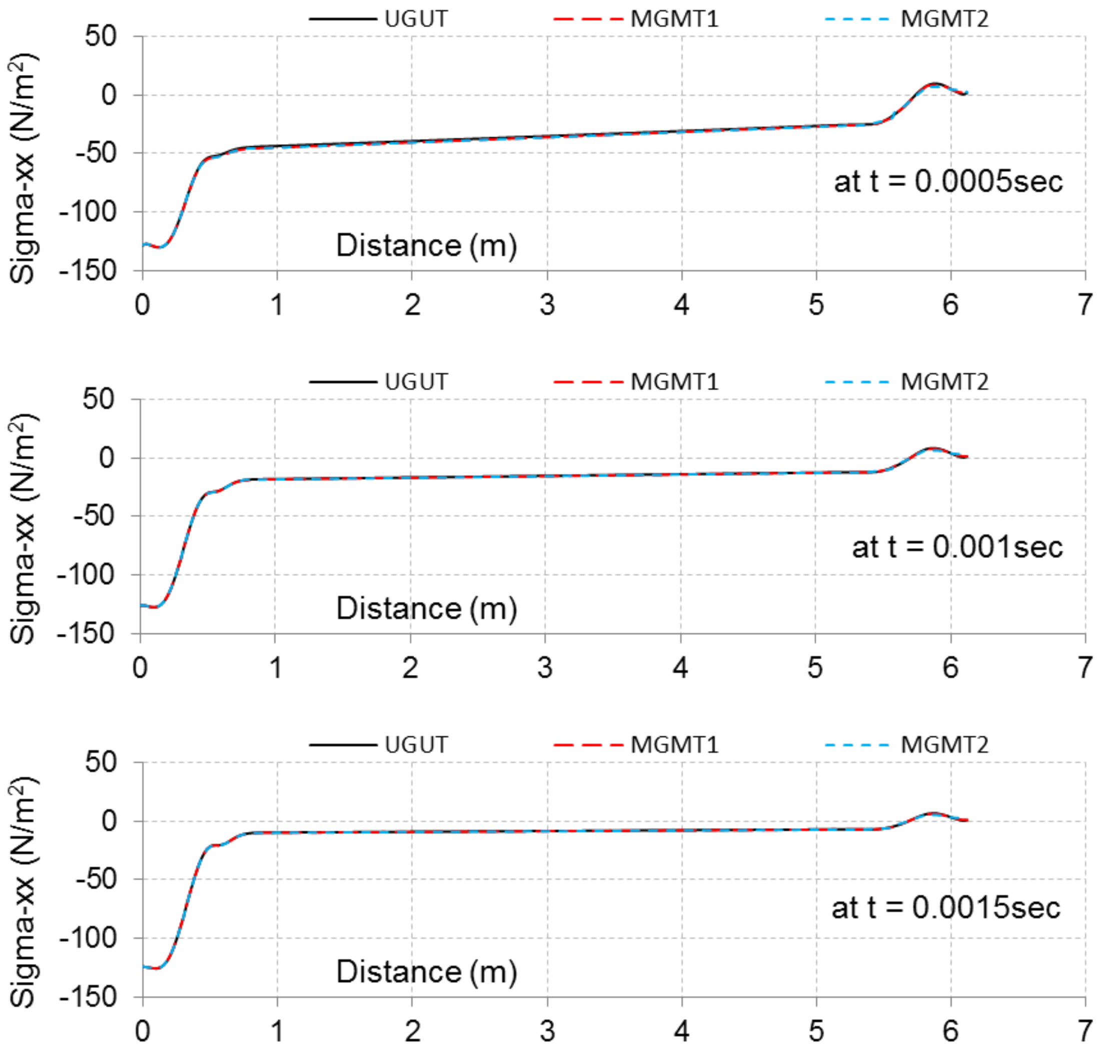


Figure 6-43: Sigma-xx as a function of space

Table 6-19: RMSE and NRMSE (%). Variable = Displacement-x as a function of space

	MGMT1	MGMT2
t = 0.0005sec	0.89 (0.63%)	1.19 (0.85%)
t = 0.001sec	0.22 (0.16%)	0.53 (0.39%)
t = 0.0015sec	0.14 (0.10%)	0.38 (0.29%)

From aforementioned results we see that MGMT1 has a better conformance (smaller NRMSE than MGMT2) with UGUT since it hosts more DOF than MGMT2. There is a significant error in displacement-x as a function of space for both MGMT1 and MGMT2 due to $\Omega 1-\Omega 2$ interface disconnection, Figure 6-44, under the action of sudden impulse loading. Accordingly, utmost care must be taken when modeling critical regions subjected to impact conditions. All other MGMT1 results show relatively small errors, averaging to less than 1%, which is significantly smaller in comparison to the gain in simulation speedup (~99%) as shown in Table 6-20.

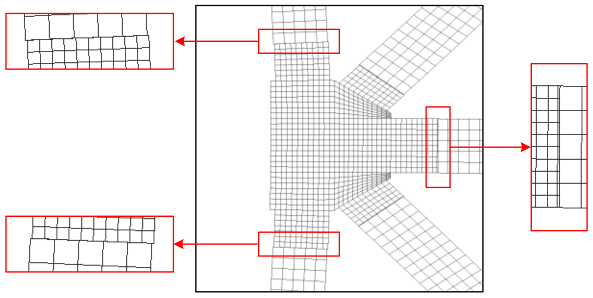


Figure 6-44: Interface disconnection (Deformed shape graphed at 5E7 magnification)

Table 6-20: Example 3 – Comparison of computational resources

	Nodes	Elements	Number of equations	Skyline storage	Solution time (sec)
UGUT	24133	22076	48214	10444765	38268.63 (~11 hr)
MGMT1	7390 ▼69.37%	6284 ▼71.53%	14748 ▼69.41%	1473410 ▼85.89%	300.97 (~5 min) ▼99.24%
MGMT2	3236 ▼86.59%	2624 ▼88.11%	6448 ▼86.62%	387348 ▼96.29%	160.64 (~3 min) ▼99.54%

6.4 Example 4: Curved Frame under Point Loading

In this example we will analyze a thin curved frame subjected to concentrated point load as shown in Figure 6-45. Since the domain under analysis is doubly symmetric with symmetric loading conditions, we shall model only a quarter of the original domain with symmetric boundary condition as shown Figure 6-46. 8 node quadrilateral elements with 2DOF/node and 3x3 Gauss integration rule will be used to discretize the domain along with plane stress formulation, consistent mass matrix and zero damping.

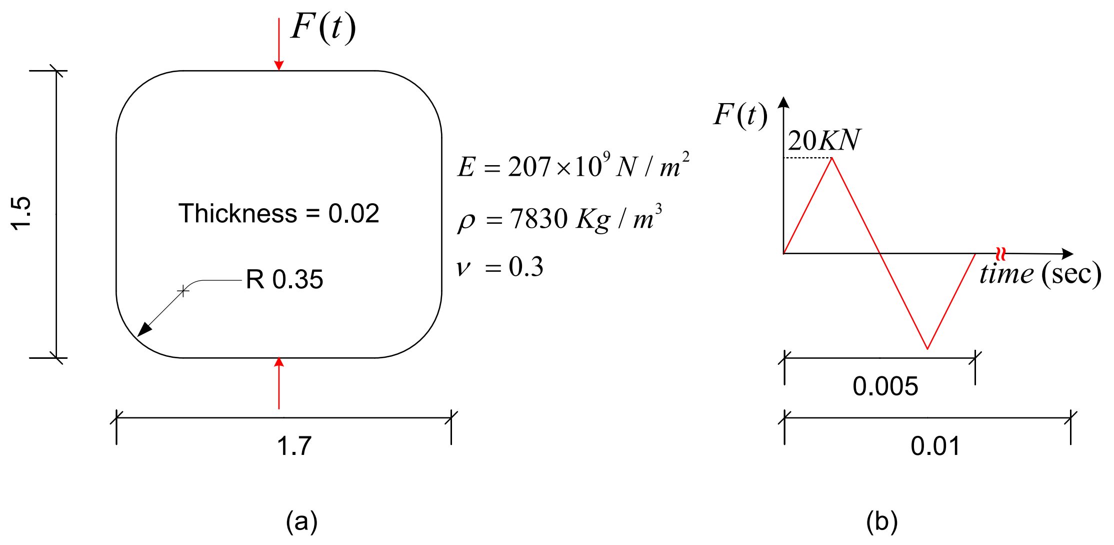
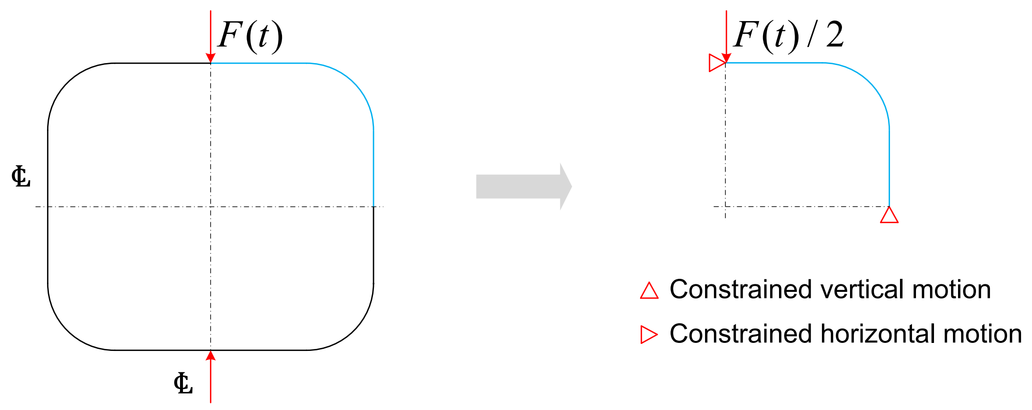


Figure 6-45: (a) Domain under analysis: Curved frame (b) Transient point loading



$F(t)$
℄
℄
$F(t)/2$
△ Constrained vertical motion
▷ Constrained horizontal motion

Figure 6-46: Finite element modeling of a double symmetric domain under symmetric loading condition

Two cases, Figure 6-47, will be analyzed under this example. For MGMT analysis, we will isolate the region around concentrated point load (Ω2) and analyze the overall vibration characteristics of the thin frame, primarily by comparing the conformance of displacement, velocity and accelerations of N1 (y-component) and N2 (x-component), see Figure 6-47, with results obtained from corresponding UGUT nodes. The goal is to ensure that MGMT interface is capable of efficiently communicating information between component sub-domains (grid ratio = 5, time-step ratio = 100) so the dynamic behavior in remote regions is retained.

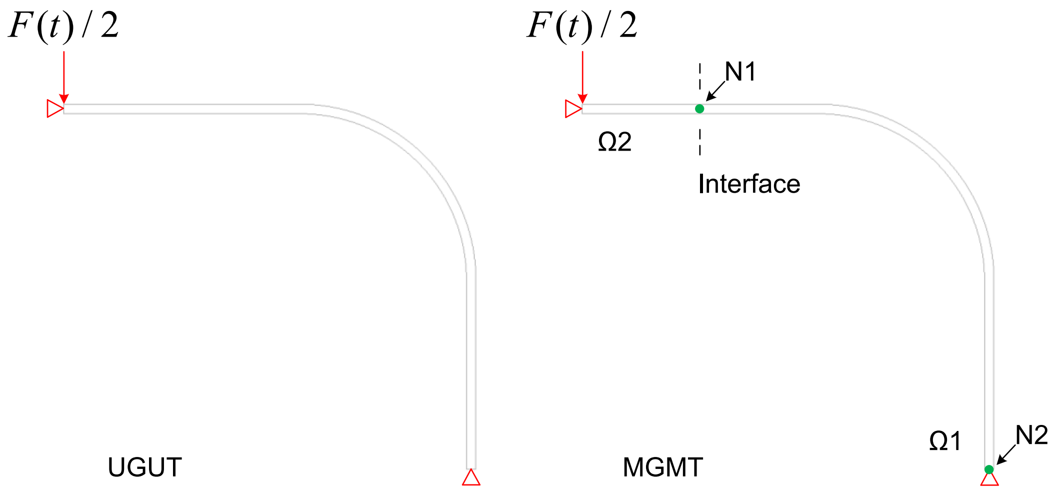


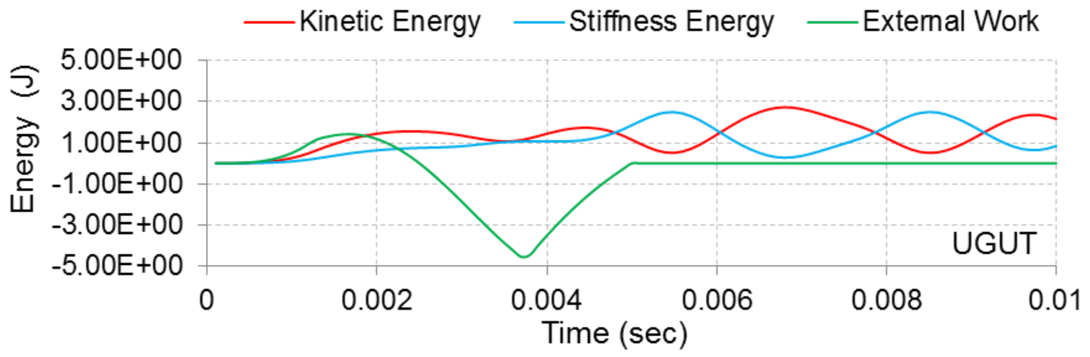
Figure 6-47: Analyzed cases

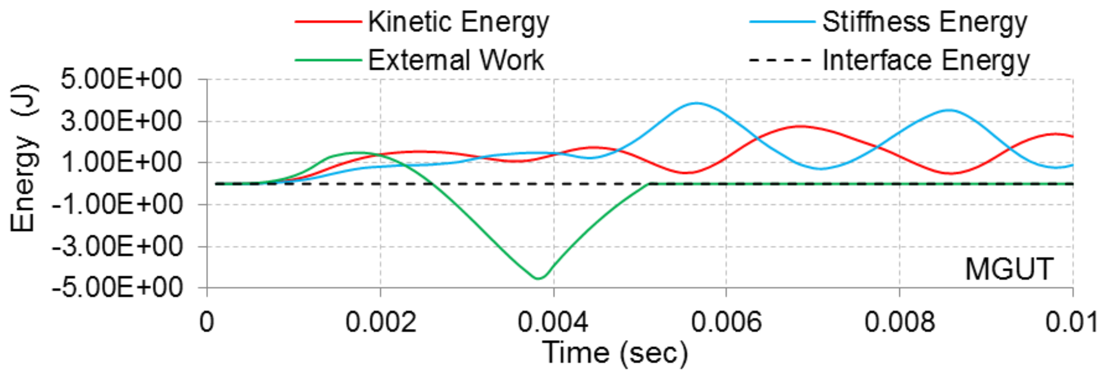
Table 6-21: Simulation parameters

	Grid spacing (H)	Interface DOF (λ)	Newmark parameters	Time-step (ΔT)	Time-step ratio (ξ)
UGUT	$\overline{H}=0.002$	-	β=0.25, γ=0.5 (Implicit)	$1x10^{-6}$	-
MGMT-Ω1	$\overline{H}=0.01$	10	β=0.25, γ=0.5 (Implicit)	$1x10^{-4}$	1
MGMT-Ω2	$\overline{H}=0.002$		β=0.25, γ=0.5 (Implicit)	$1x10^{-6}$	100

Subsequently, we will look at:

1) Evolution of global energies: kinetic energy, stiffness energy and external work with relative errors in MGMT analysis (Figure 6-48 and Table 6-22).
2) Augmented interface energy and its mean variance about zero (Figure 6-48 and Table 6-22).
3) Kinematic conformity (comparison between UGUT and MGMT) and interface continuity (comparison between MGMT-Ω1 and MGMT-Ω2) of displacement-y, velocity-y and acceleration-y for interface node N1, See Figure 6-47, with corresponding errors (Figure 6-49 and Table 6-23).
4) Kinematic conformity (comparison between UGUT and MGMT) of displacement-x, velocity-x and acceleration-x for node N2, See Figure 6-47, with corresponding errors (Figure 6-50).
5) Overall deformed shape and displacement contour plots at various time-instants. Displacement-x (Figure 6-51) and Displacement-y (Figure 6-52).
6) Comparison on of computational resources (Table 6-24).





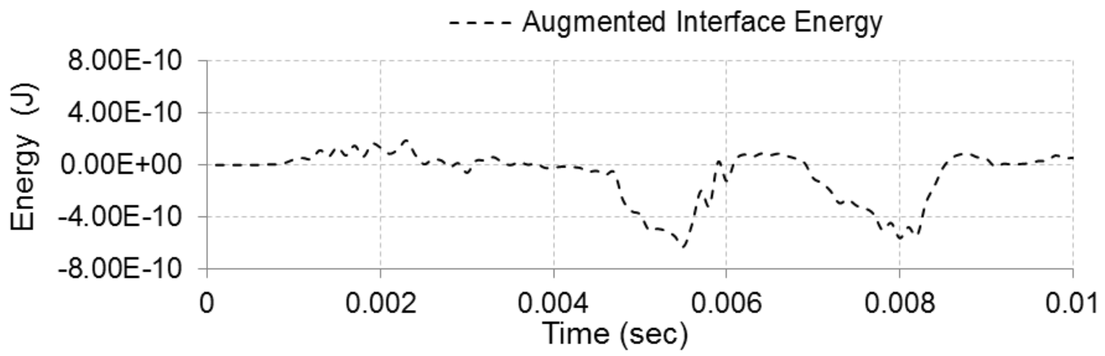


Figure 6-48: Global energies

Table 6-22: RMSE and NRMSE (%). Variable = Global Energies

	Kinetic Energy	Stiffness Energy	Interface Energy
MGMT	6.41×10^{-2} (2.35%)	0.762 (26.89%)	2.09×10^{-10}

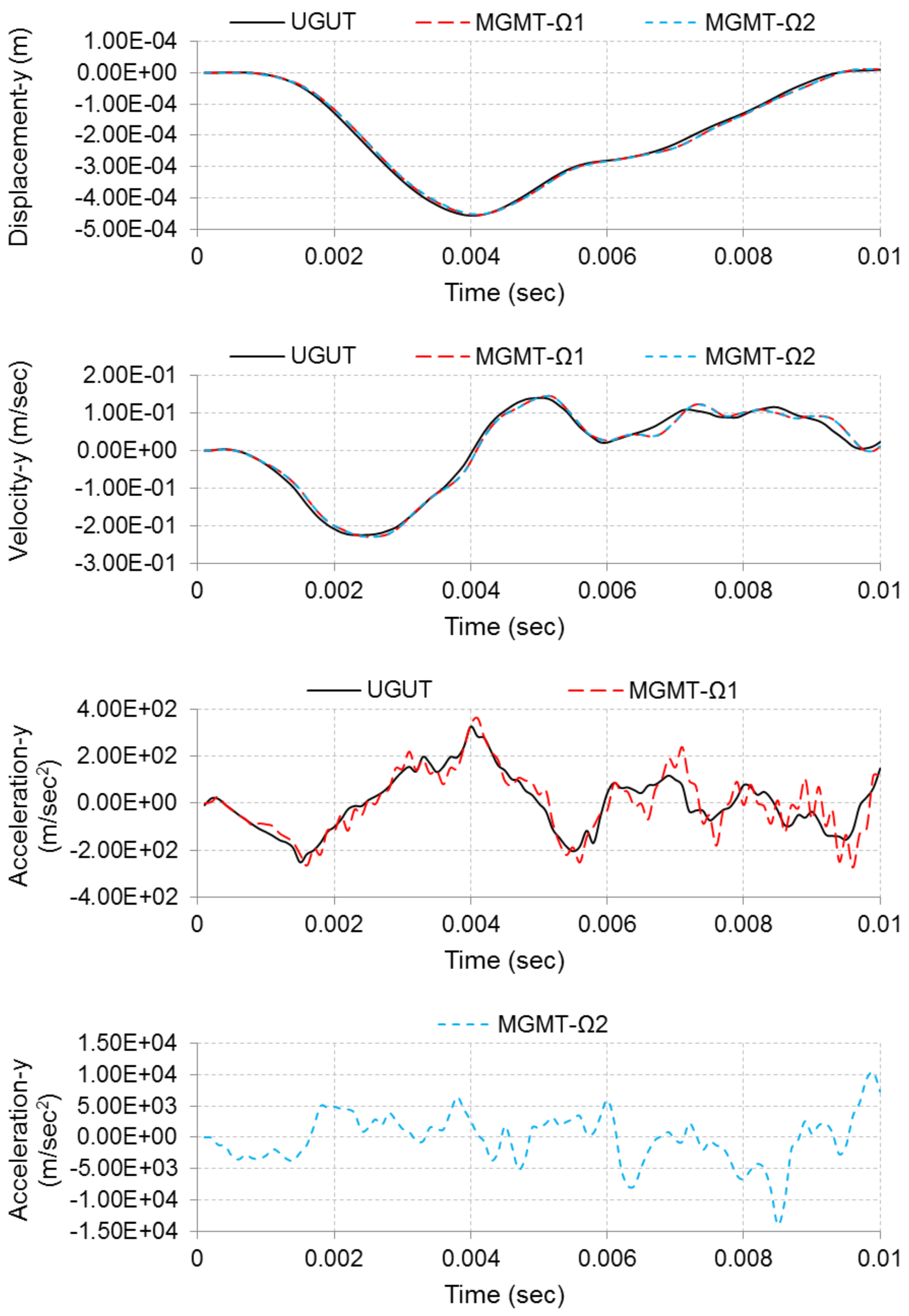


Figure 6-49: Kinematic conformity and interface continuity (N1)

Table 6-23: RMSE and NRMSE (%). Variable = Kinematic conformity/interface continuity (N1)

	UGUT v/s MGMT-Ω1	UGUT v/s MGMT-Ω2	MGMT-Ω1 v/s MGMT-Ω2
Displacement	$7.11\text{x}10^{-6}$ (1.52%)	$7.60\text{x}10^{-6}$ (1.63%)	$1.26\text{x}10^{-6}$ (0.27%)
Velocity	$1.18\text{x}10^{-2}$ (3.25%)	$1.16\text{x}10^{-2}$ (3.20%)	$4.58\text{x}10^{-4}$ (0.12%)
Acceleration	60.96 (10.53%)	3956.27 (683.8%)	3958.18 (621.5%)

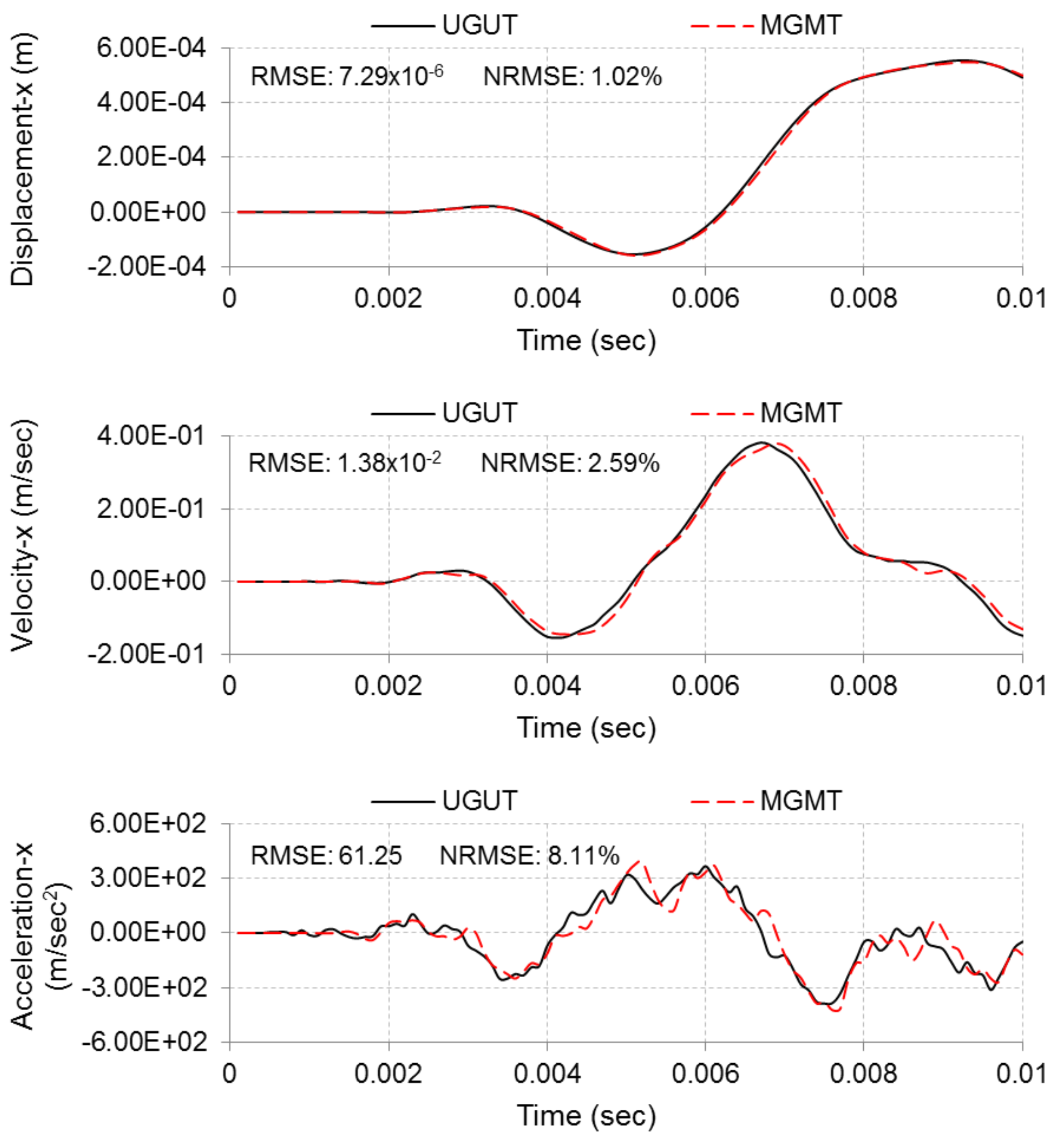


Figure 6-50: Kinematic conformity (N2)

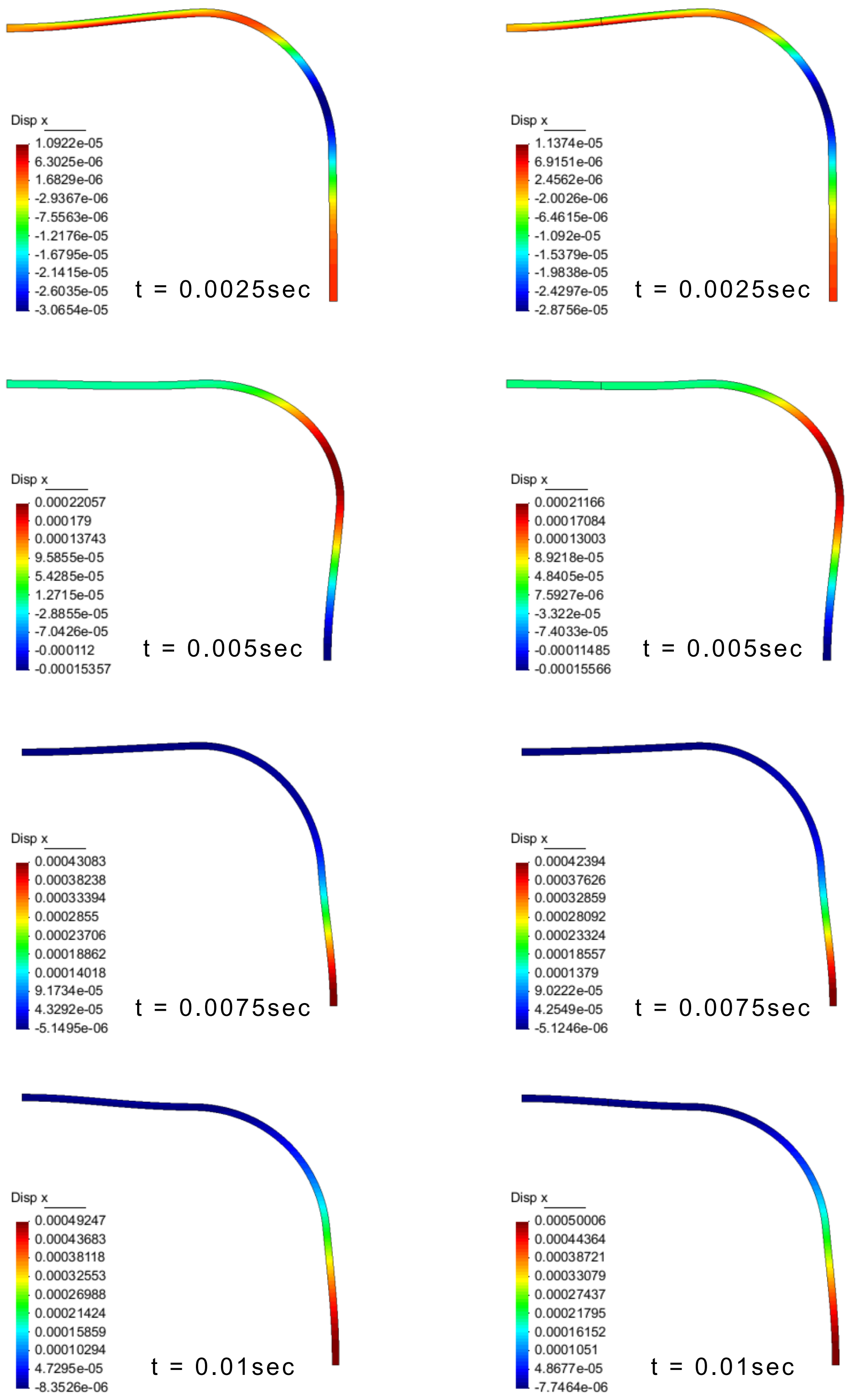


Figure 6-51: Contour plots for Displacement-x. UGUT (Left) and MGMT (Right)

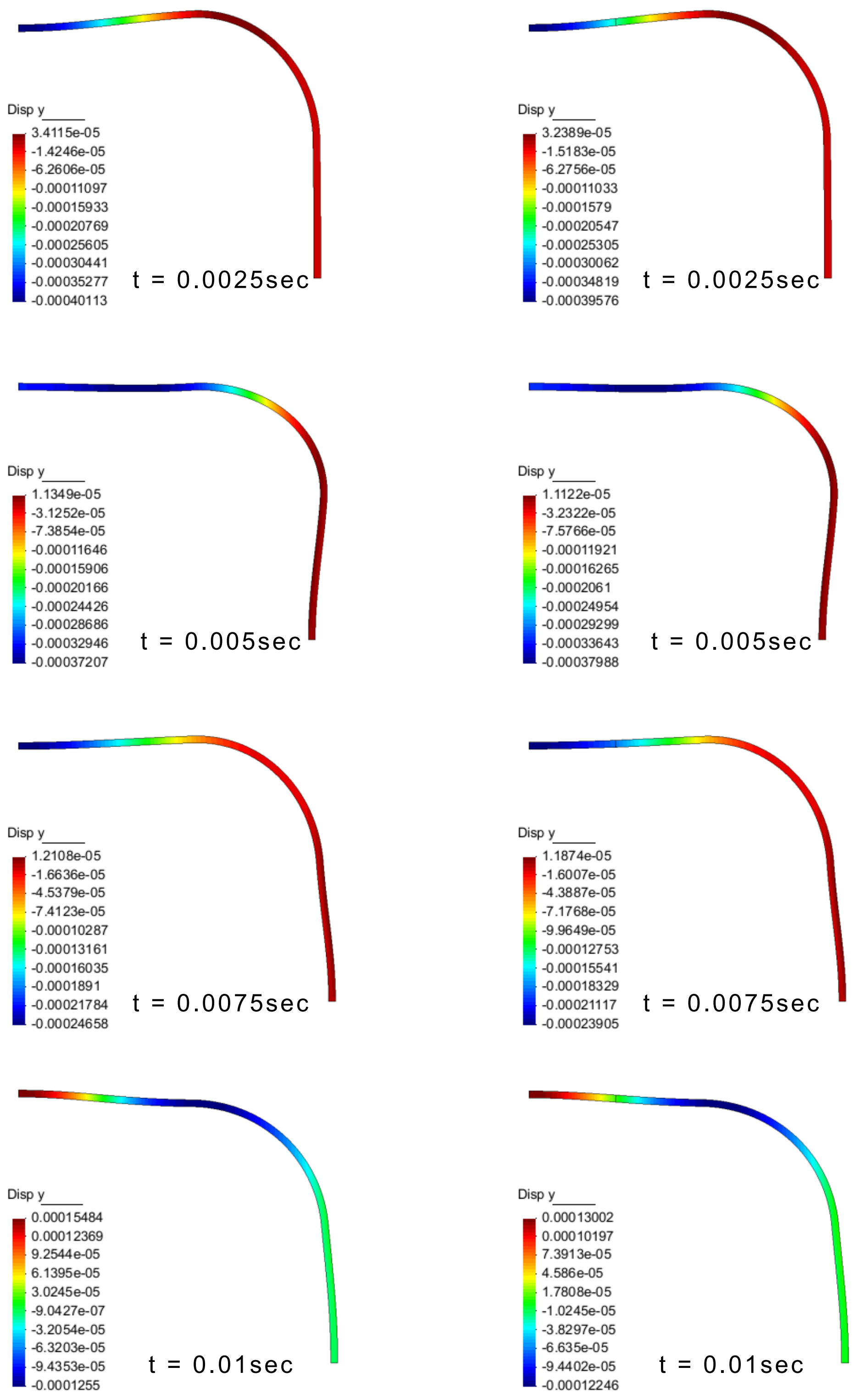


Figure 6-52: Contour plots for Displacement-y. UGUT (Left) and MGMT (Right)

Aforementioned results show very good conformance between UGUT and MGMT simulations. The augmented interface energy (10^{-10}) is several orders of magnitude smaller than global energies (10^{0}) ensuring numerical stability. Kinematic comparison between UGUT and MGMT results measured at N1 yields significantly larger errors (1.6% in displacement and 3.2% in velocity) but the comparison of interface continuity between Ω1 and Ω2 results measured at N1 yields relatively low errors (0.2% in displacement and 0.1% in velocity) ensuring efficient grid coupling. Errors in kinematic comparison of remote node N2 are also sufficiently low (1% in displacement and 2.5% in velocity) ensuring that global dynamic behavior can be efficiently retained in MGMT simulations. Comparison between deformed shape and displacement counter plots further substantiates this inference.

As shown in Table 6-24, this example certainly demonstrates the principal advantage in using MGMT Method. It is evident that reducing the total number of space discretized variables (number of equations) will inherently yield errors but the resulting gain in computational speed up due to MGMT coupling is indeed significant.

Table 6-24: Example 4 – Comparison of computational resources

	Nodes	Elements	Number of equations	Skyline storage	Solution time (sec)
UGUT	23733	7410	47424	4483371	41929.49 (~12 hr)
MGMT	5010 ▼78.89%	1496 ▼79.81%	9994 ▼78.92%	541396 ▼87.92%	3724.36 (~1 hr) ▼91.11%

6.5 Example 5: Bridge Analysis

In this example we will analyze a large scale bridge problem with heterogeneous materials subjected to complex loading conditions. The goal is to ensure conformance in global structural dynamics, kinematic continuity at material interface and gain in computational efficiency as a result of MGMT implementation. Various cases discussed under this example are listed in Table 6-25. The bridge, Figure 6-53, is discretized using 4-node quadrilateral elements with 2 DOF/node, plane stress formulation and consistent mass matrix. Rayleigh damping is assumed and the corresponding damping coefficients are: $m_c = 0.01$ for mass and $k_c = 0.05$ for stiffness. Isotropic linear elastic material model is used with the following properties: Steel: E = 2.07x10^{11} N/m^2, ν = 0.3 and ρ = 7.83x10^3 Kg/m^3 and Concrete: E = 40x10^9 N/m^2, ν = 0.2 and ρ = 2400 Kg/m^3.

Table 6-25: Simulation parameters

	Grid spacing (H)	Interface DOF (λ)	Newmark parameters	Time-step (ΔT)	Time-step ratio (ξ)
UGUT	$\overline{H} = 0.3$	-	β=0.25, γ=0.5 (Implicit)	1.25x10^{-3}	-
MTC-Ω1	$\overline{H} = 0.3$	72	β=0.25, γ=0.5 (Implicit)	5x10^{-3}	1
MTC-Ω2			β=0.25, γ=0.5 (Implicit)	1.25x10^{-3}	4
MGC-Ω1	$\overline{H} = 1.0$	24	β=0.25, γ=0.5 (Implicit)	1.25x10^{-3}	1
MGC-Ω2	$\overline{H} = 0.3$		β=0.25, γ=0.5 (Implicit)	1.25x10^{-3}	
MGMT-Ω1	$\overline{H} = 1.0$	24	β=0.25, γ=0.5 (Implicit)	5x10^{-3}	1
MGMT-Ω2	$\overline{H} = 0.3$		β=0.25, γ=0.5 (Implicit)	1.25x10^{-3}	4

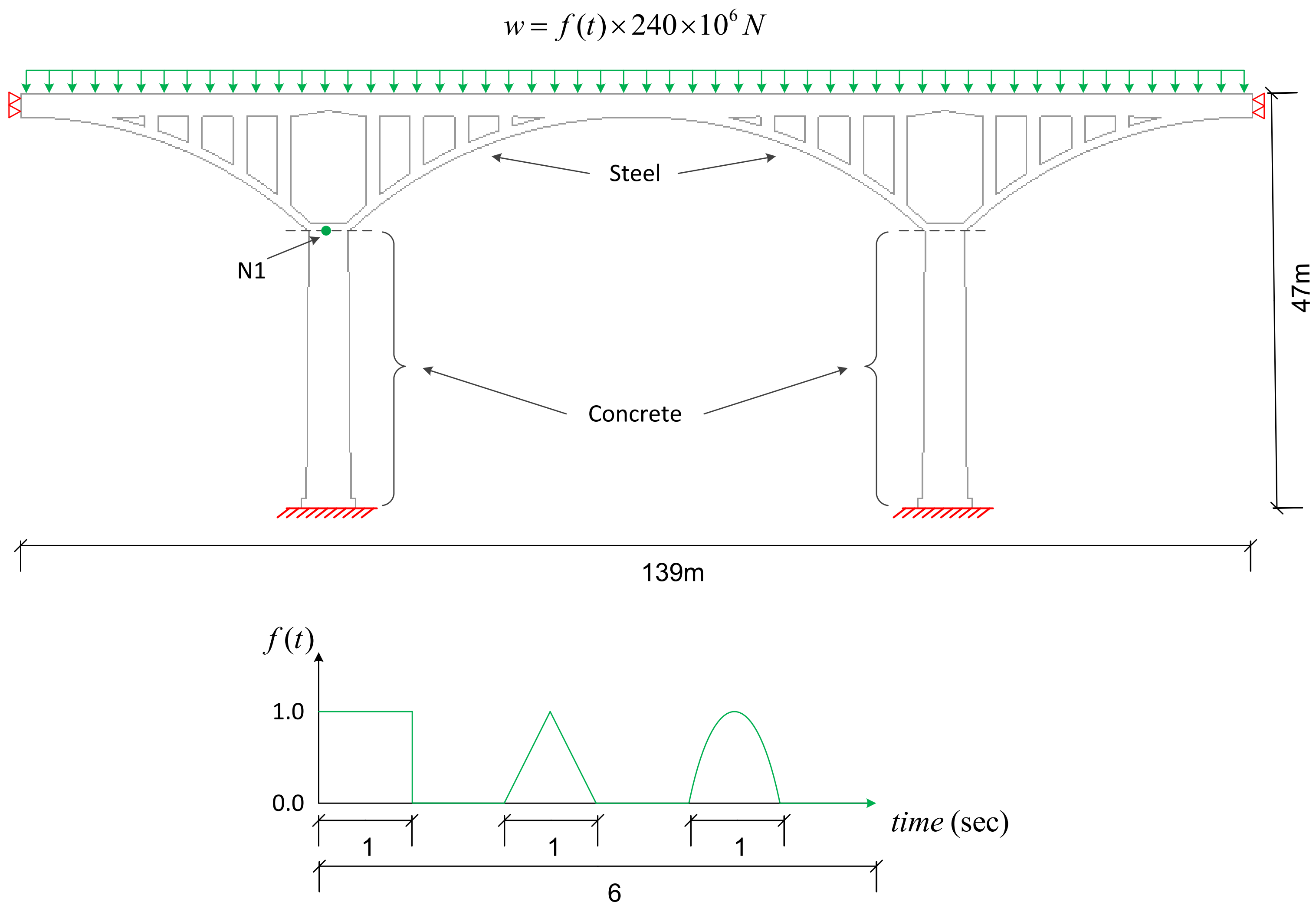


Figure 6-53: Bridge analysis: Domain description and time proportional load function

Spatial discretization listed in Table 6-25 results in UGUT w/ 12,886 nodes, MTC w/ 12,886 + 36 interface nodes, MGC and MGMT w/ 9,482 + 12 interface nodes. Accordingly, analysis domain is first solved using UGUT to establish baseline results and computational efficiency followed by MTC $(\xi = 4)$ to quantify the gain in computational efficiency as a result of introducing multiple time-stepping only. MGC, with an average grid ratio of 4 and $\xi = 1$, is then solved to measure the computational efficiency and the resulting gain produced due to reducing the total number of domain DOF. Finally, a combination of aforementioned MTC and MGC cases: MGMT is used to observe the collective effect of reducing the total number of DOF and multiple time-stepping on total CPU solution time. In addition to measuring the computational scaling introduced as a result of MGMT (and constituent special case) simulations; a comprehensive comparison of global FE results (in reference with UGUT) is presented in order to ensure that the dynamic behavior of the domain is captured with desired accuracy. Accordingly, we will also look at: 1) Evolution of global energies with relative errors (Figure 6-54 and Table 6-26). Note: The kinetic energy in these plots is magnified x100 for visual resolution and comparison only. 2) Augmented interface energy and its mean variance about zero (Figure 6-55 and Table 6-26). 3) Kinematic conformity (comparison between UGUT and MTC/MGC/MGMT) and interface continuity (comparison between -Ω1 and -Ω2) for displacement-y, velocity-y and acceleration-y of interface node N1, see Figure 6-53, with corresponding errors (Figure 6-56 and Table 6-27). 4) Deformed shape and displacement/stress contour plots at various time instants (Figure 6-57, Figure 6-58, Figure 6-59, Figure 6-60. Note: Deformed shape is graphed at x50 magnification

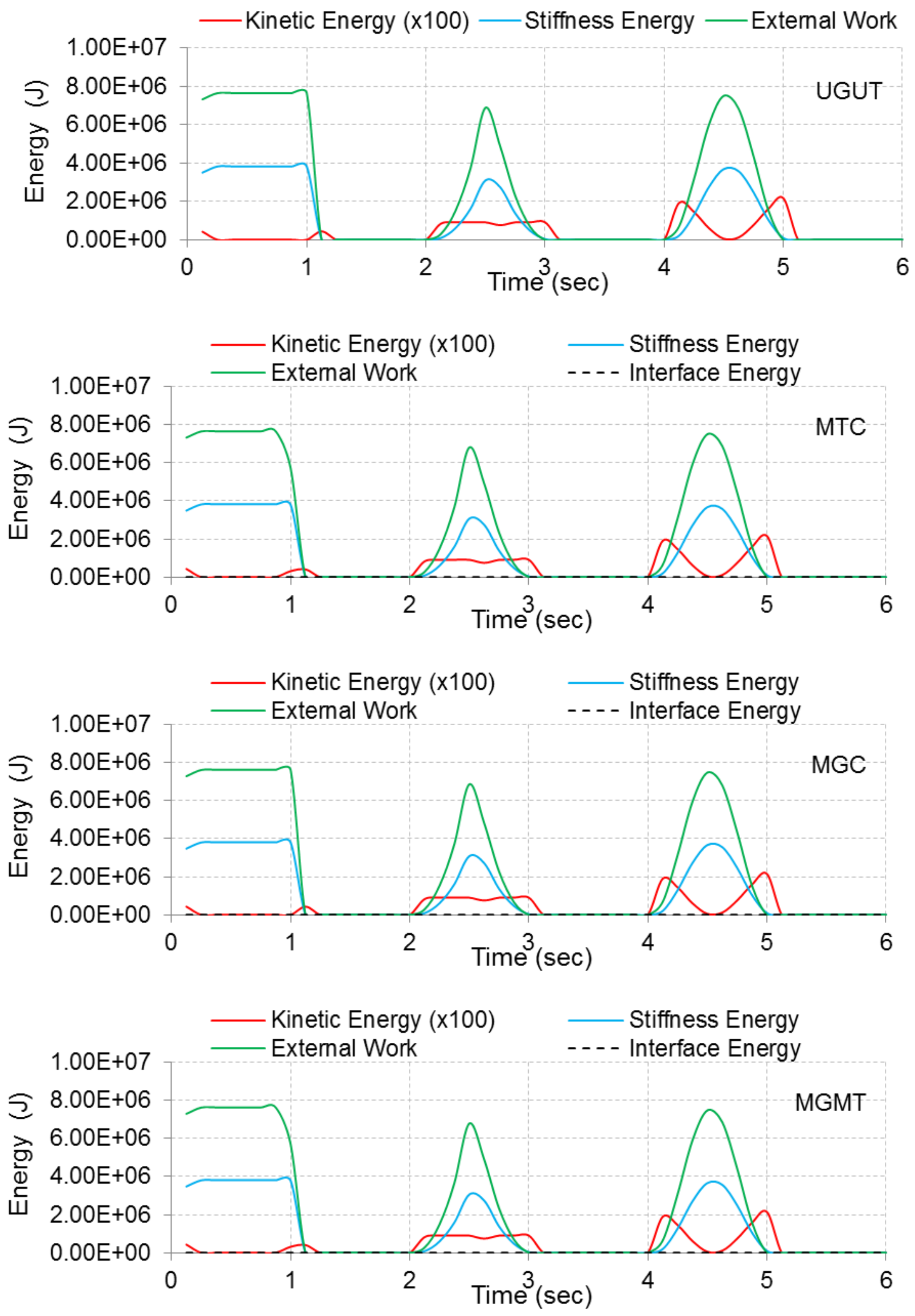


Figure 6-54: Global energies

Table 6-26: RMSE and NRMSE (%). Variable = Global energies

	Kinetic Energy	Stiffness Energy	Interface Energy
MTC	461.91 (2.12%)	10537.26 (0.27%)	$5.96\text{x}10^{-9}$
MGC	57.02 (0.26%)	7225.80 (0.18%)	$7.13\text{x}10^{-9}$
MGMT	466.37 (2.14%)	12633.63 (0.33%)	$7.36\text{x}10^{-9}$

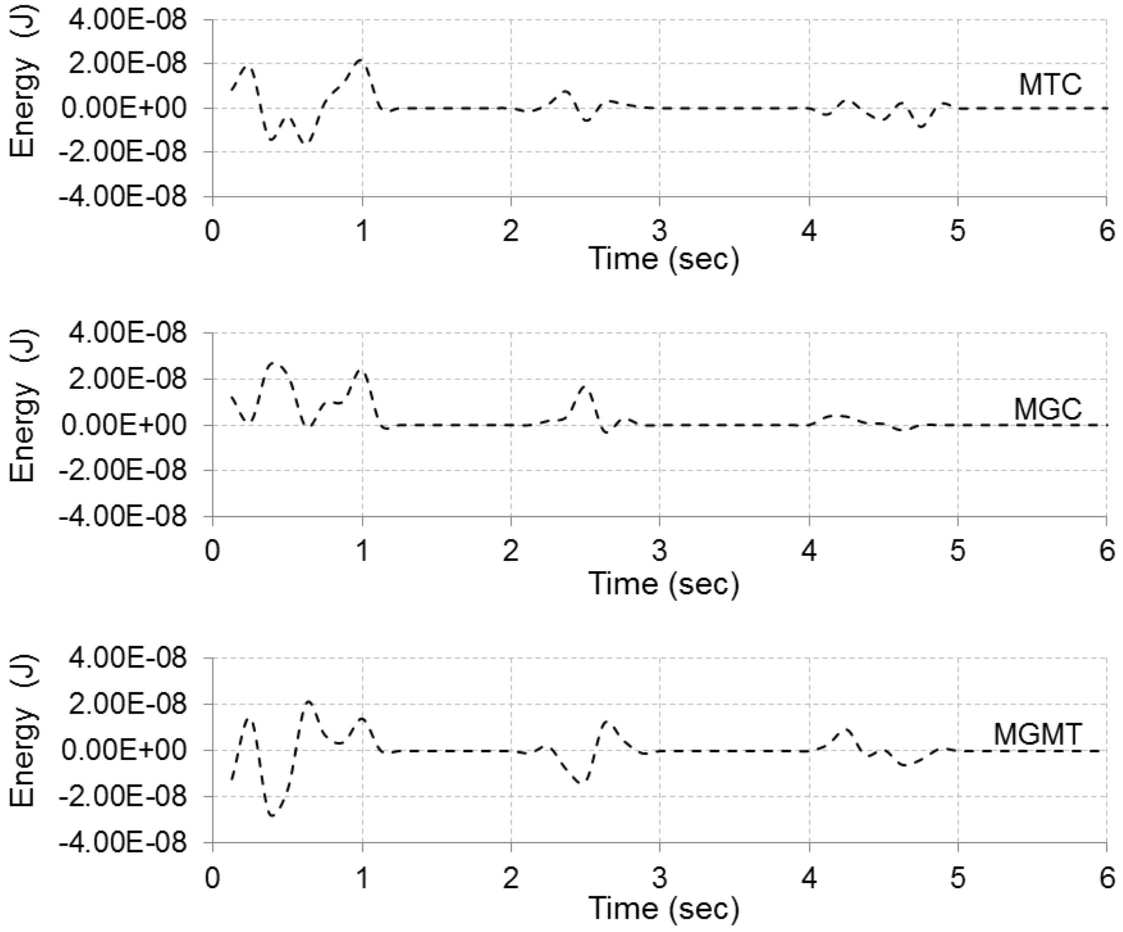


Figure 6-55: Augmented interface energies

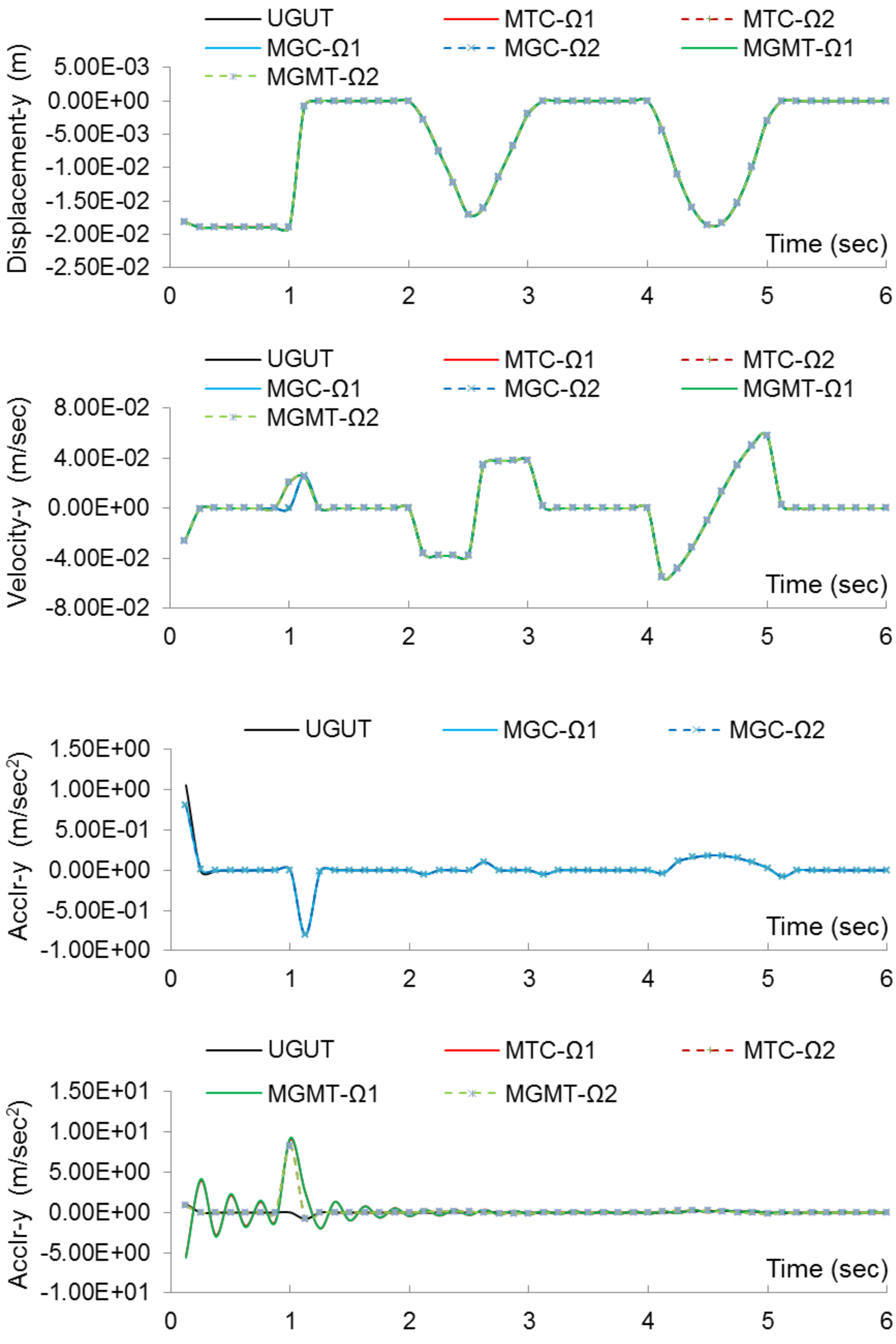


Figure 6-56: Kinematic conformity and interface continuity

Table 6-27: RMSE and NRMSE (%). Variable = Kinematic conformity (v/s UGUT results)

	Displacement	Velocity	Acceleration
MTC-Ω1	4.87×10^{-5} (0.25%)	2.97×10^{-3} (2.62%)	1.93 (103.75%)
MTC-Ω2	4.84×10^{-5} (0.25%)	2.97×10^{-3} (2.62%)	1.18 (63.74%)
MGC-Ω1	1.14×10^{-5} (0.06%)	8.40×10^{-5} (0.07%)	3.63×10^{-2} (1.94%)
MGC-Ω2	3.31×10^{-6} (0.01%)	7.55×10^{-5} (0.06%)	3.64×10^{-2} (1.95%)
MGMT-Ω1	4.93×10^{-5} (0.26%)	2.99×10^{-3} (2.63%)	1.99 (106.7%)
MGMT-Ω2	4.80×10^{-5} (0.25%)	2.98×10^{-3} (2.63%)	1.18 (63.73%)

Table 6-28: RMSE and NRMSE (%). Variable = Interface continuity (Ω1 v/s Ω2)

	Displacement	Velocity	Acceleration
MTC	1.50×10^{-5} (0.07%)	0.0 (0.0%)	1.41 (9.7%)
MGC	1.27×10^{-5} (0.06%)	2.87×10^{-5} (0.02%)	2.30×10^{-4} (0.01%)
MGMT	6.02×10^{-6} (0.03%)	2.90×10^{-5} (0.02%)	1.46 (9.84%)

Energy plots for constituent MGMT cases are in very good conformance with each other and with the reference UGUT results. Augmented interface energy, with an average mean variance of 10^{-9}, remains consistently small compared to global energy scales (10^{6}), hence ensuring numerical stability. Kinematic agreement (UGUT v/s MTC/MGC/MGMT) for displacement and velocity is also very reasonable with the least errors occurring in MGC. MTC accounts for significant disagreement between kinematic conformances, however the 0.0% error in interface continuity ensure efficient multiple time-scale coupling. It is clear from these results that MTC guarantees interface continuity and MGC ensure close conformance with desired UGUT results. Accordingly and as expected, MGMT yields averaged errors from both MTC and MGC cases.

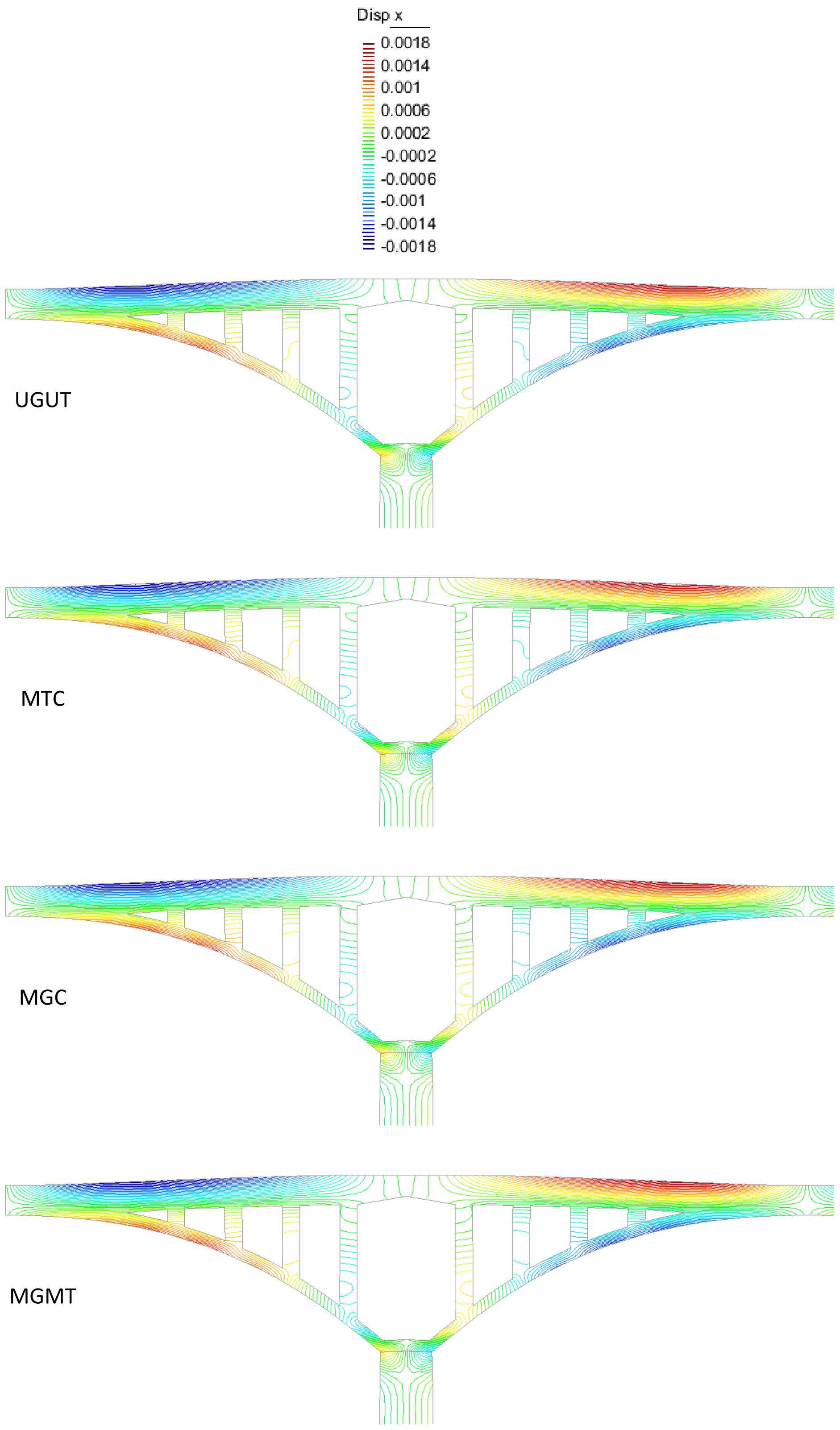


Figure 6-57: Displacement-x at t=0.5sec (Deformed shape graphed at x50)

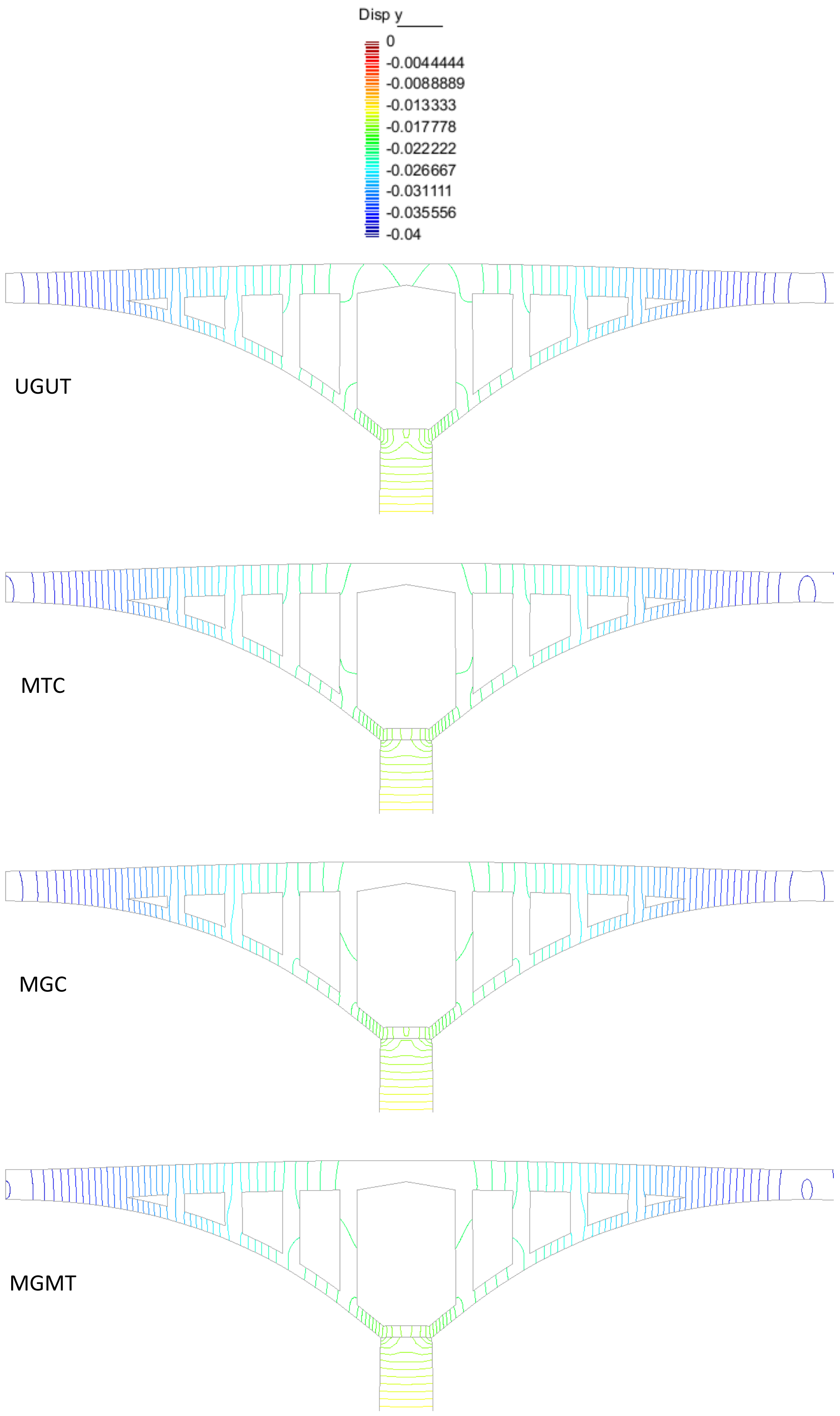


Figure 6-58: Displacement-y at t=2.5sec (Deformed shape graphed at x50)

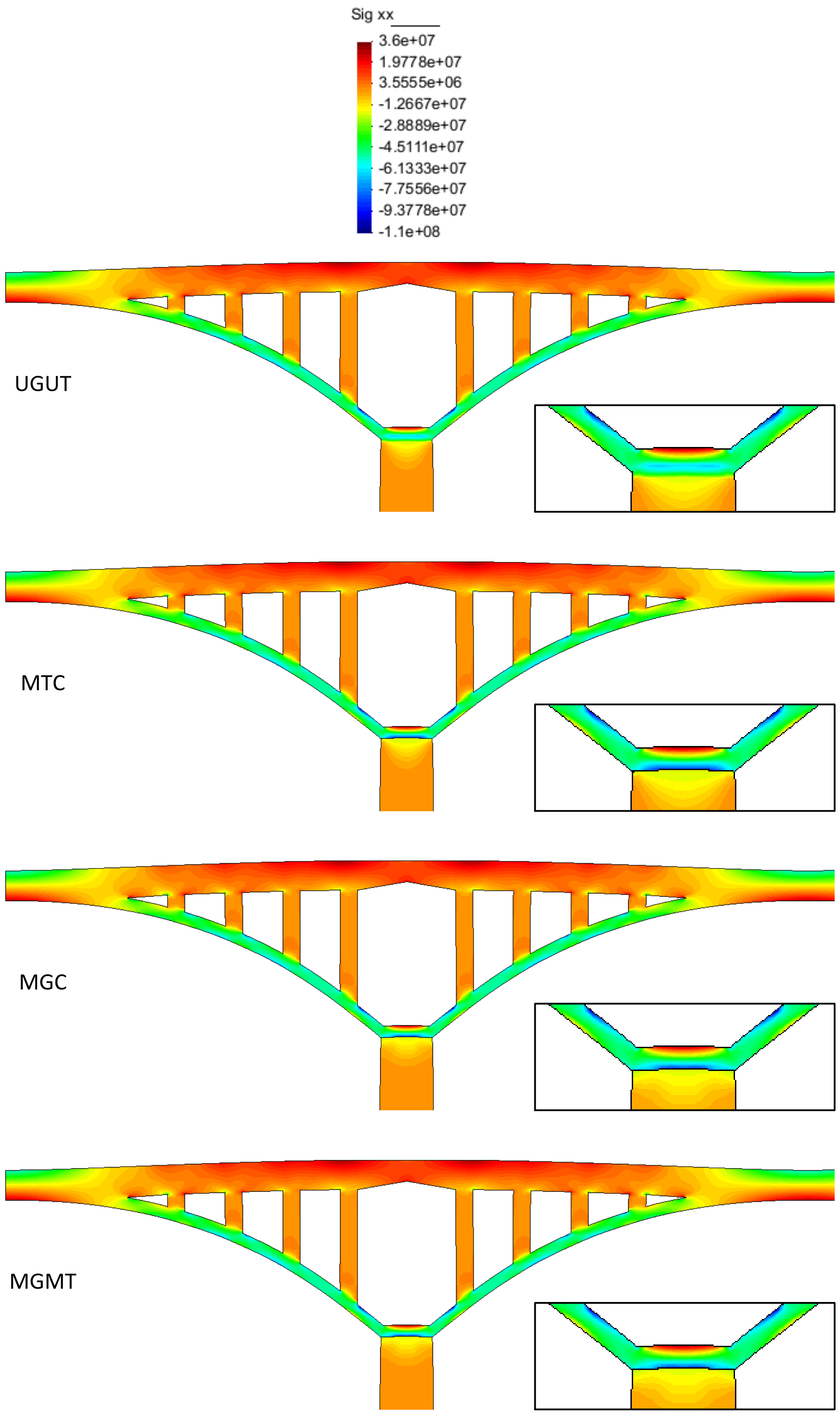


Figure 6-59: Sigma-xx at t=4.5sec (Deformed shape graphed at x50)

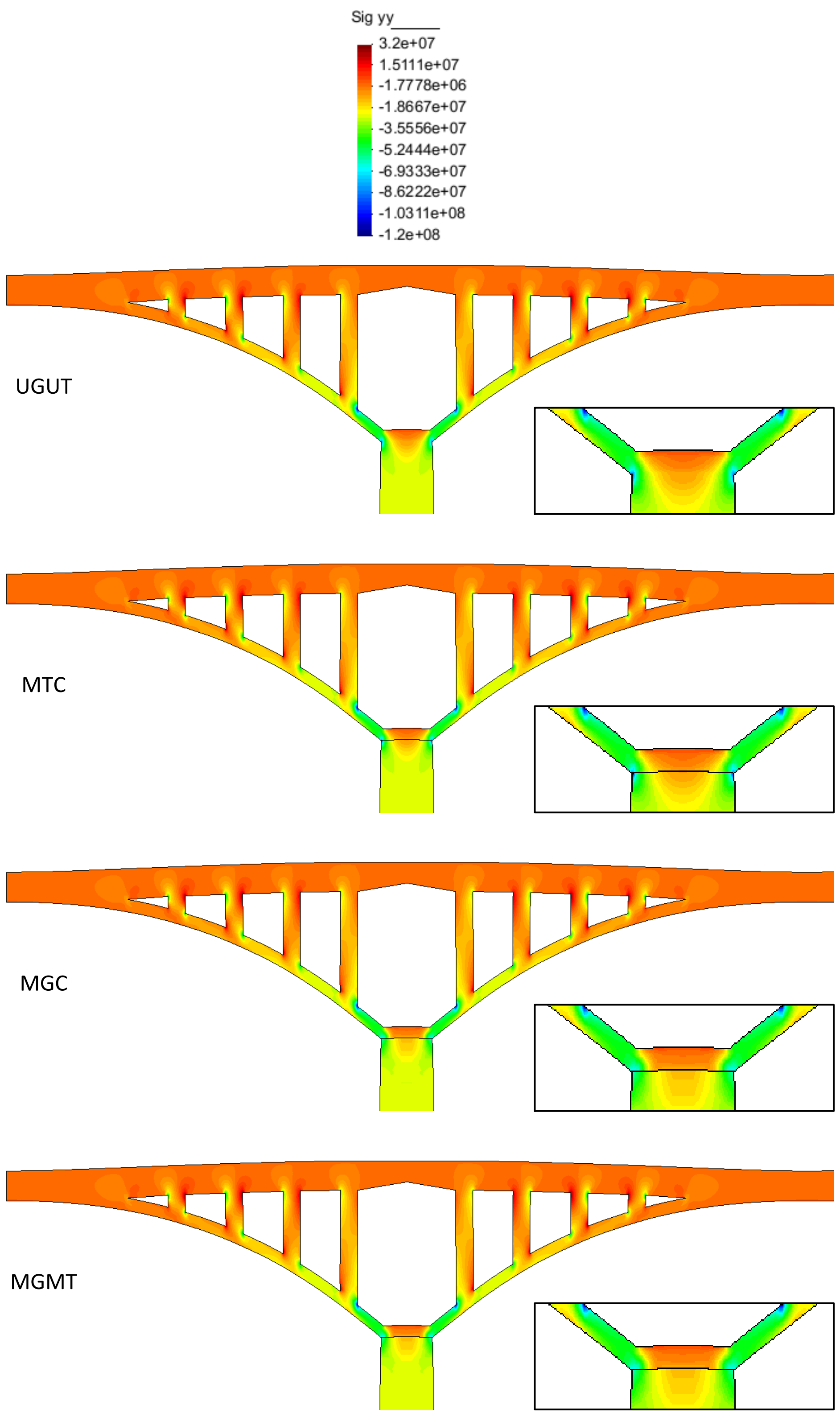


Figure 6-60: Sigma-yy at t=4.5sec (Deformed shape graphed at x50)

Aforementioned results show very conformance between UGUT and constituent MGMT cases. As we can see from Table 6-29, MGC effectively reduces computation time by 70%. Certainly, the error incurred in MGC coupling is primarily due to the loss in available DOF, but this is in-turn is complimented with a significant gain in computational efficiency. MTC with a small time-step ratio of 4, effectively reduces simulation time by 58%, and as discussed earlier allows efficient time-scale coupling with very good conformance in continuity of interface variables (with the exception of acceleration).

As expected, MGMT coupling enhances computational efficiency whilst averaging errors in MGC and MTC scenarios. Accordingly, it is certainly more desirable to compliment distinct grid discretizations with respective time-stepping parameters. Overall, we can see that MGMT Method is capable of preserving accuracy in desired critical regions (Ω2 in this example) and is also efficient in modeling global behavior of the domain under analysis.

Table 6-29: Example 5 – Comparison of computational resources

	Nodes	Elements	Number of equations	Skyline storage	Solution time (sec)
UGUT	12886	11628	25656	2381836	9766.96 (~3 hr)
MTC	12922 ▲0.27%	11628 ▲▼	25728 ▲0.28%	1574408 ▼33.89%	4070.47 (~1 hr) ▼58.32%
MGC	9494 ▼26.32%	8374 ▼27.98%	18928 ▼26.22%	896648 ▼62.35%	2865.57 (~48 min) ▼70.66%
MGMT	9494 ▼26.32%	8374 ▼27.98%	18928 ▼26.22%	896648 ▼62.35%	2535.82 (~42 min) ▼74.03%

Chapter 7: Conclusions and Future Directions

7.1 Conclusions

A systematic approach to perform concurrent multiscale simulations within the purview of continuum mechanics, and as applicable to linear structural dynamic systems is presented. Derived simulation strategy is largely based upon the fundamental principles of DDM, allowing selective discretization (spatial and temporal) of component sub-domains. Constituent governing equations for decomposed sub-domains are linked together through Lagrange Multipliers, used to represent pseudo interface reactions, and the resulting system of coupled equations is augmented with an appropriate interface condition that demands interface energy, produced as a result of introducing interface reactions, to be equal to zero. It is shown that enforcing this particular condition naturally results in the continuity of velocities across sub-domain interfaces. Conversely, continuity of velocities is enforced as an interface (sub-domain boundary) condition and it is shown, using Energy Method, that resulting interface reactions from adjacent sub-domains completely annihilate each other and therefore yield zero interface energy accumulation or dissipation.

Association between space discretized equations, form component sub-domains (conforming or non-conforming), is established via M-FEM and uses the same old Lagrange Multipliers to communicate multiple constraints across sub-domain interfaces; hence allowing Multiple Grid (MG-) coupling. Semi-discretized, and grid coupled, equilibrium equations are then selectively discretized in time using Newmark time integration method. Multiple Time-scale (-MT) coupling is then established by requiring the equilibrium equation, from fully-discretized sub-domains, to be identically satisfied at every intermediate time-step. Subsequently, it is shown that as long as the stability requirements are satisfied within the time integration of component sub-domains, MGMT coupling is stable and energy preserving. Additionally, interface reactions (Lagrange Multipliers) from intermediate time-step are condensed and expressed in terms of global time-steps, further enhancing computational efficiency. Final set of equations for MGMT sub-domains are solved using block elimination and Crout factorization and the corresponding step-by-step algorithm to obtain global solution at synchronous time-steps is presented.

MGMT Method is rigorously implemented for the numerical simulation of linear structural dynamic systems and an in-house computer program – Finite Element Analysis Programming Interface (FEAPI) is developed for analysis and verification purposes. Overall performance of the proposed formulation is assessed by solving benchmark problems; followed by stability analysis, and evaluation of numerical accuracy and computational efficiency. Comparisons are made against reference uniform grid uniform time-scale simulations and relative errors (RMSE and NRMSE) are presented for the entire set of analyzed results. It is shown that MGMT Method is numerically stable, reasonably accurate and yields good conformance with reference results. It is also shown that numerical accuracy is proportional to grid density (in a constituent sub-domain) and is inversely proportional to the computational efficiency (of the global problem). Hence, the loss in available degrees of freedom is clearly reflected in the NRMSE; however it is relatively insignificant compared to the advantage gained in simulation speedup. Thus, a comprehensive matrix of error rankings and computational efficiency is presented to allow competent and proficient application of the MGMT Method.

Furthermore, examples involving stress resolution in critical regions, wave propagation across heterogeneous material systems, complex loading functions and evaluation of global structural behavior show the potential advantage (and limitations) in using MGMT Method in such application problems. Results show that MGMT Method yields very good conformance in global energies with relatively infinitesimal interface energies, hence ensuring overall numerical stability. Kinematic comparison between reference UGUT and MGMT results also yields good conformance. Continuity of interface variables, especially velocity, ensures efficient implementation of augmented interface conditions and further validates zero interface energy. It is seen that MGC primarily contributes to simulation speedup while MTC allows accurate conformance in overall structural dynamics and accordingly MGMT produces the best results with averaged errors (from MTC and MGC) with improved computational efficiency. It is also shown that distinct grid resolutions should be accompanied with appropriate time-step scaling to achieve optimal balance between numerical accuracy and simulation speedup.

7.2 Future Directions

The task of multiscale modeling at macroscopic levels, whilst integrating a fair balance in numerical accuracy and computational efficiency, is immensely challenging. Owing to the widespread scope continuum mechanics, and structural dynamics in general, many specific application domains still remain unexplored; some of which are listed as follows:

1) Several engineering materials exhibit characteristic orientation, such as fiber reinforced composite materials. Even though the material may deform elastically, the simple isotropic model implemented under this dissertation is unable to describe the response of such materials accurately. This implementation can however be easily extended to include a more general stress-strain relationship, including the description of anisotropic solids, in order to accurately capture the response of such materials.
2) Multiple grid coupling using M-FEM can be certainly extended to incorporate 3-dimensionsal sub-domains and contact problems. Accordingly, the performance and efficiency of MGMT coupling should be evaluated and verified in scenarios involving large (3D) geometries and contact conditions.
3) M-FEM, although efficient, should be further explored and analyzed for optimal Lagrange Multiplier spaces that are more suitable for MGMT coupling interfaces.
4) MGMT Method, with appropriate implementation, can also be used to improve computational efficiency in non-linear (geometric or material) problems.
5) Domain Decomposition Method and MGMT Method in turn inherently yield to the capability of parallel computing. Since component sub-domain, each with their own boundary conditions and discretizations, are solved independent of each other, sub-

domain specific computations can be assigned to individual processors further boosting computational efficiency.

6) Adaptive Mesh Refinement (AMR) is an attractive FE feature that enables minimizing errors in desired regions – as triggered by pre-defined user criterions during the course of simulation. This technique can certainly be combined with MGMT Method, allowing problem specific sub-domain discretizations – only in triggered regions and only when required.
7) MGMT Method can also be extended to handle transient field problem, such as heat conduction or fluid flow.
8) Coupled problems such as fluid-structure interaction or soil pore-fluid interaction can also benefit from MGMT Method since it allows sub-domain specific physical modeling and solution algorithms.

In addition to allowing sub-domain specific discretizations in the analysis of large-scale structural dynamic systems, MGMT Method has a promising field of application in coupling microscopic discrete atomic systems with macroscopic continuum models. Atomistic modeling of explicit interactions between atoms provides valuable insight into material behavior and its failure. Accordingly, coupling atomistic simulations with continuum based models can help predict structural behavior with better accuracy. Atomistic modeling however uses nanometer (10^{-9} m) space scales and picosecond (10^{-12} sec) time-scales, making it computationally impossible to augment large-scale (10^{0}m) systems with complementary discretizations. MGMT Method can however provide a smooth transition between macro and micro scales, enabling atomistic-continuum coupling.

Bibliography

Baiocchi, C., Brezzi, F. & Marini, L.D., 1992. Stabilization of Galerkin methods and applications to domain decomposition. In A. Bensoussan & J.-P. Verjus, eds. *Future Tendencies in Computer Science, Control and Applied Mathematics SE - 23*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 343–355.

Bajer, C., 2002. Time integration methods - still questions. In W. Szczesniak, ed. *Theoretical Foundations of Civil Engineering*. Warsaw, pp. 45–54.

Bathe, K. & Wilson, E., 1976. Numerical methods in finite element analysis. *International Journal for Numerical Methods in Engineering*, 11(9), p.1485.

Bathe, K.-J., 1996. *Finite element procedures*, Prentice Hall.

Becker, R., Hansbo, P. & Stenberg, R., 2003. A finite element method for domain decomposition with non-matching grids. *ESAIM: Mathematical Modelling and Numerical Analysis*, 37(02), pp.209–225.

Bellenger, E. & Coorevits, P., 2005. Adaptive mesh refinement for the control of cost and quality in finite element analysis. *Finite Elements in Analysis and Design*, 41(15), pp.1413–1440.

Belytschko, T. & Mullen, R., 1977. Mesh partitions of explicit-implicit time integration. In K.-J. Bathe, J. T. Oden, & W. Wunderlich, eds. *Formulations of Computational Algorithms in Finite Element Analysis*. Cambridge: MIT Press.

Belytschko, T. & Mullen, R., 1978. Stability of explicit-implicit mesh partitions in time integration. *International Journal for Numerical Methods in Engineering*, 12(10), pp.1575–1586.

Belytschko, T., Smolinski, P. & Liu, W.K., 1984. Multistepping implicit-explicit procedures in transient analysis. In W. K. Liu, ed. *Innovative Methods for Nonlinear Problems*. Swansea: Pineridge Press, pp. 135–154.

Belytschko, T., Yen, H.J. & Mullen, R., 1979. Mixed methods for time integration. *Computer Methods in Applied Mechanics and Engineering*, 17–18 Part(February), pp.259–275.

Bernardi, C., Maday, Y. & Patera, A.T., 1994. A new nonconforming approach to domain decomposition: the mortar element method. In H. Brezis, ed. *Nonlinear partial differential equations and their applications*. Paris, France.

Bernardi, C., Maday, Y. & Patera, A.T., 1993. Domain Decomposition by the Mortar Element Method. In H. Kaper, M. Garbey, & G. Pieper, eds. *Asymptotic and Numerical Methods for Partial Differential Equations with Critical Parameters*. Springer Netherlands, pp. 269–286.

Bower, A.F., 2009. *Applied Mechanics of Solids*, CRC Press.

Bruijs, M., 1990. *Subcycling in transient finte element analysis*. Eindhoven University of Technology.

Brun, M. et al., 2012. Implicit/explicit multi-time step co-computations for predicting reinforced concrete structure response under earthquake loading. *Soil Dynamics and Earthquake Engineering*, 33(1), pp.19–37.

Budynas, R. & Nisbett, J., 2008. *Shigley's mechanical engineering design*, McGraw-Hill.

Chen, Y., Lee, J.D. & Eskandarian, A., 2000. *Meshless method for solid mechanics*, New York, NY: Springer New York.

Chung, J. & Hulbert, G.M., 1993. A Time Integration Algorithm for Structural Dynamics With Improved Numerical Dissipation: The Generalized-α Method. *Journal of Applied Mechanics*, 60(2), pp.371–375.

Combescure, A. & Gravouil, A., 2001. A time-space multi-scale algorithm for transient structural non-linear problems. *Mécanique & Industries*, 2(1), pp.43–55.

Combescure, A., Gravouil, A. & Herry, B., 2003. An algorithm to solve transient structural non-linear problems for non-matching time-space domains. *Computers & structures*, 81(12), pp.1211–1222.

Cook, R.D. et al., 2001. *Concepts and applications of finite element analysis* 4th ed., Wiley.

Dodds, R. & Lopez, L., 1980. Substructuring in linear and nonlinear analysis. *International Journal for Numerical Methods in Engineering*, 15(June 1979), pp.583–597.

Eringen, A., 1980. Mechanics of continua.

Farhat, C. et al., 2006. Time-parallel implicit integrators for the near-real-time prediction of linear structural dynamic responses. *International Journal for Numerical Methods in Engineering*, 67(5), pp.697–724.

Farhat, C. & Chandesris, M., 2003. Time-decomposed parallel time-integrators: theory and feasibility studies for fluid, structure, and fluid-structure applications. *International Journal for Numerical Methods in Engineering*, 58(9), pp.1397–1434.

Farhat, C., Crivelli, L. & Géradin, M., 1995. Implicit time integration of a class of constrained hybrid formulations—Part I: Spectral stability theory. *Computer Methods in Applied Mechanics and Engineering*, 125(1-4), pp.71–107.

Farhat, C., Crivelli, L. & Roux, F.-X., 1994. A transient FETI methodology for large-scale parallel implicit computations in structural mechanics. *International Journal for Numerical Methods in Engineering*, 37(11), pp.1945–1975.

Farhat, C. & Roux, F.-X., 1991. A method of finite element tearing and interconnecting and its parallel solution algorithm. *International Journal for Numerical Methods in Engineering*, 32(6), pp.1205–1227.

Faucher, V. & Combescure, A., 2003. A time and space mortar method for coupling linear modal subdomains and non-linear subdomains in explicit structural dynamics. *Computer methods in applied mechanics and Engineering*.

Govaerts, W., 1991. Stable solvers and block elimination for bordered systems. *SIAM journal on matrix analysis and applications*, 12(3), pp.469–483.

Gravouil, A. & Combescure, A., 2003. Multi-time-step and two-scale domain decomposition method for non-linear structural dynamics. *International Journal for Numerical Methods in Engineering*, 58(10), pp.1545–1569.

Gravouil, A. & Combescure, A., 2001. Multi-time-step explicit–implicit method for non-linear structural dynamics. *International Journal for Numerical Methods in Engineering*, 50(1), pp.199–225.

Herry, B., Di Valentin, L. & Combescure, A., 2002. An approach to the connection between subdomains with non-matching meshes for transient mechanical analysis. *International Journal for Numerical Methods in Engineering*, 55(8), pp.973–1003.

Hilber, H.M., Hughes, T.J.R. & Taylor, R.L., 1977. Improved numerical dissipation for time integration algorithms in structural dynamics. *Earthquake Engineering & Structural Dynamics*, 5(3), pp.283–292.

Hughes, T.J.R., 2012. *The finite element method: linear static and dynamic finite element analysis*, Mineola, New York: Dover Publications.

Hughes, T.J.R. & Liu, W.K., 1978. Implicit-Explicit Finite Elements in Transient Analysis: Stability Theory. *Journal of Applied Mechanics*, 45(2), pp.371–374.

Hughes, T.J.R., Pister, K.S. & Taylor, R.L., 1979. Implicit-explicit finite elements in nonlinear transient analysis. *Computer Methods in Applied Mechanics and Engineering*, 17–18, Par(0), pp.159–182.

Hughes, T.J.R. & Stephenson, R.A., 1981. Convergence of implicit-explicit algorithms in nonlinear transient analysis. *International Journal of Engineering Science*, 19(2), pp.295–302.

Lacour, C. & Maday, Y., 1997. Two different approaches for matching nonconforming grids: The mortar element method and the FETI method. *BIT Numerical Mathematics*, 37(3), pp.720–738.

Lamichhane, B.P. & Wohlmuth, B.I., 2004a. A quasi-dual Lagrange multiplier space for serendipity mortar finite elements in 3D. *ESAIM: Mathematical Modelling and Numerical Analysis*, 38(1), pp.73–92.

Lamichhane, B.P. & Wohlmuth, B.I., 2004b. Mortar finite elements for interface problems. *Computing*, 72, pp.333–348.

Lamichhane, B.P. & Wohlmuth, B.I., 2005. Mortar finite elements with dual lagrange multipliers: Some applications. *Domain Decomposition Methods in Science and Engineering*, 40, pp.319–326.

Lawson, C. et al., 1979. Basic linear algebra subprograms for Fortran usage. *ACM Transactions on Mathematical Software*, 5(3), pp.308–323.

Lions, P., 1987. On the Schwarz alternating method. I. In R. Glowinski et al., eds. *Proceedings of the 1st International Conference on Domain Decomposition Methods*. Paris, France.

Lions, P., 1989. On the Schwarz alternating method. III: a variant for nonoverlapping subdomains. In T. F. Chan, R. Glow, & O. Widlund, eds. *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*. Houston, Texas: SIAM, pp. 202–223.

Liu, W.K. & Belytschko, T., 1982. Mixed-time implicit-explicit finite elements for transient analysis. *Computers & Structures*, 15(4), pp.445–450.

Maday, Y., Mavriplis, C. & Patera, A.T., 1988. Nonconforming mortar element methods: Application to spectral discretizations. In T. F. Chan et al., eds. *Proceedings of the 2nd International Conference on Domain Decomposition Methods*. California: SIAM, pp. 392–418.

Mahjoubi, N. & Gravouil, A., 2011. A monolithic energy conserving method to couple heterogeneous time integrators with incompatible time steps in structural dynamics. *Computer Methods in Applied Mechanics and Engineering*, 200(9-12), pp.1069–1086.

Mahjoubi, N., Gravouil, A. & Combescure, A., 2009. Coupling subdomains with heterogeneous time integrators and incompatible time steps. *Computational Mechanics*, 44(6), pp.825–843.

Mandel, J., 2005. Balancing domain decomposition. *Communications in Numerical Methods in Engineering*, 9(March 1992), pp.1–10.

Miranda, I., Ferencz, R.M. & Hughes, T.J.R., 1989. An improved implicit-explicit time integration method for structural dynamics. *Earthquake Engineering & Structural Dynamics*, 18(5), pp.643–653.

Newmark, N., 1959. A method of computation for structural dynamics. *Journal of Engineering Mechanics*, 85(EM3), pp.67–94.

Park, K. & Felippa, C., 2000. A variational principle for the formulation of partitioned structural systems. *International Journal for Numerical Methods in Engineering*, 47(1-3), pp.395–418.

Park, K., Felippa, C. & DeRuntz, J., 1977. Stabilization of staggered solution procedures for fluid-structure interaction analysis. *ASME Applied Mechanics Division Symposia Series*, 26, pp.94–124.

Parsons, I. & Hall, J., 1990a. The multigrid method in solid mechanics: part I—algorithm description and behaviour. *International Journal for Numerical Methods in Engineering*, 29(July 1989), pp.719–737.

Parsons, I. & Hall, J., 1990b. The multigrid method in solid mechanics: part II—practical applications. *International Journal for Numerical Methods in Engineering*, 29(July 1989), pp.739–753.

Plesek, J., Kolman, R. & Gabriel, D., 2012. Estimation of critical time step for explicit integration. In *18th International Conference on Engineering Mechanics*. Svratka, Czech Republic, pp. 1001–1010.

Pointer, J., 2002. Understanding Accuracy and Discretization Error in an FEA Model. In *Ansys*.

Prakash, A., 2007. *Multi-time-step domain decomposition and coupling methods for non-linear structural dynamics*. University of Illinois at Urbana-Champaign.

Prakash, A. & Hjelmstad, K.D., 2004. A FETI-based multi-time-step coupling method for Newmark schemes in structural dynamics. *International Journal for Numerical Methods in Engineering*, 61(13), pp.2183–2204.

Reddy, J.N., 2007. *An Introduction to Continuum Mechanics*, Cambridge University Press.

Richtmyer, R.D. & Morton, K.W., 1967. *Difference methods for initial-value problems* Second., New York, NY, USA: Interscience.

Roux, F.-X., 1990. Domain decomposition methods for static problems. *La Recherche Aerospatiale(English Edition)*, (1), pp.37–48.

Shah, C., 2002. Mesh Discretization Error and Criteria for Accuracy of Finite Element Solutions. In *Ansys*.

Smith, B.F., Bjorstad, P.E. & Gropp, W.D., 1996. *Domain decomposition: parallel multilevel methods for elliptic partial differential equations*, New York, NY, USA: Cambridge University Press.

Smith, I.M., Griffiths, D.V. & Margetts, L., 2013. *Programming the finite element method* 5th Editio., John Wiley & Sons, Inc.

Smolinski, P., 1992. An explicit multi-time step integration method for second order equations. *Computer Methods in Applied Mechanics and Engineering*, 94(1), pp.25–34.

Smolinski, P., Sleith, S. & Belytschko, T., 1996. Stability of an explicit multi-time step integration algorithm for linear structural dynamics equations. *Computational mechanics*, 18, pp.236–244.

Smolinski, P. & Wu, Y.-S., 1998. An implicit multi-time step integration method for structural dynamics problems. *Computational Mechanics*, 22(4), pp.337–343.

Spencer, A.J.M., 2004. *Continuum Mechanics* Dover Ed e., Dover Publications.

Tallec, P. Le & Sassi, T., 1995. Domain decomposition with non-matching grids: Augmented Lagrangian approach. *Mathematics of Computation*, 64(212), pp.1367–1396.

Toselli, A. & Widlund, O., 2005. *Domain decomposition methods-algorithms and theory*, Springer Series in Computational Mathematics.

Wikipedia, 2014. Relative change and difference. *Wikipedia, The Free Encyclopedia.* Available at: http://en.wikipedia.org/w/index.php?title=Relative_change_and_difference&oldid=636200665.

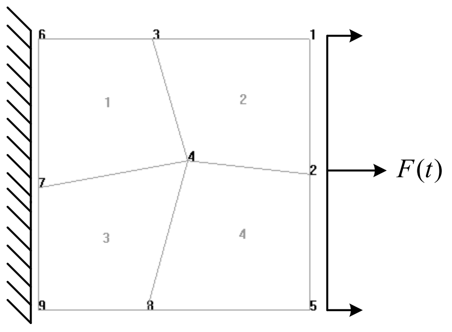
Wood, W.L., Bossak, M. & Zienkiewicz, O.C., 1980. An alpha modification of Newmark's method. *International Journal for Numerical Methods in Engineering*, 15(10), pp.1562–1566.

Zienkiewicz, O.C., Taylor, R.L. & Zhu, J., 2005. *The finite element method: its basis and fundamentals*, Elsevier.

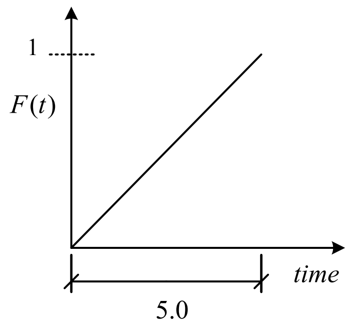
Appendix A: FEAPI Input File

FEAPI input file **.dat* may be created manually or by using FEAPI-GiD interface. This file contains basic information about analysis domain, nodal coordinates, element connectivity's, boundary conditions and other FEAPI simulation parameters.

A.1 Example Input File



(a)



(b)

Figure A-1: (a) Example problem (b) Transient (linear) loading

```
FEAPI::DOMAIN
2 9 4
Transient
PlaneStress
Quadrilateral 4 2 4

FEAPI::COORDINATES
10.00000    10.00000
10.00000    5.00000
4.20000     10.00000
5.50000     5.50000
10.00000    0.00000
0.00000     10.00000
0.00000     4.50000
4.00000     0.00000
0.00000     0.00000

FEAPI::CONNECTIVITIES
1 7 6 3 4
1 1 2 4 3
1 9 7 4 8
1 5 8 4 2
```

```
FEAPI::MATERIAL
1
207e9 0.3 7830
0 0.0 0.0
0

FEAPI::RESTRAINTS
3
6 0 0
7 0 0
9 0 0

FEAPI::LOADS
3
1 0.25 0.0 Linear 5.0 0.0
2 0.5 0.0 Linear 5.0 0.0
5 0.25 0.0 Linear 5.0 0.0

FEAPI::PRESCRIBED
0
```

```
FEAPI::TRANSIENT
Newmark
0.25
0.5
0.0
0.1 0 5.0

FEAPI::POST
1
1
1
1
1 1
1
1
1
1
0
```

A.2 Input File Data Blocks

Table A-1: FEAPI::DOMAIN

This data block defines FEAPI global variables.

```
FEAPI::DOMAIN
ndim nn nels
atype
ptype
etype nod nodof nip
```

ndim	Spatial dimensions of the FE domain under analysis
nn	Total number of nodes
nels	Total number of elements
atype	Analysis Type
ptype	Problem Type
etype	Element Type
nod	Number of nodes per element
nodof	Number of degrees of freedom per node (DOF)
nip	Number of Gauss integration points

Table A-2: FEAPI::COORDINATES

This data block defines global nodal coordinates.

```
FEAPI::COORDINATES
x-1 y-1 z-1
:
:
x-nn y-nn z-nn
```

(x,y,z)-nn	Global x, y, z coordinates for node 1 to nn

Table A-3: FEAPI::CONNECTIVITIES

This data block defines element node connectivity's and material ID for the corresponding element.

```
FEAPI::CONNECTIVITIES
nel1-matID n-1 … n-nod
:
:
nels-matID n-1 … n-nod
```

nels-matID	Material ID for element 1 to nels
n-nod	Element connectivity's starting from node 1 to node nod

Table A-4: FEAPI::MATERIAL

This data block defines finite element material properties.

```
FEAPI::MATERIAL
nmats
! IF (atype == static)
1mats-E 1mats-Nu
:
:
nmats-E nmats-Nu
! END IF
! IF (atype == transient)
1mats-E 1mats-Nu 1mats-Rho
:
:
nmats-E nmats-Nu nmats-Rho
! END IF
damping rmdc rkdc
mmf
```

nmats	Total number of materials used
nmats-E	Modulus of elasticity for materials 1 to nmats
nmats-Nu	Poisson's ratio for materials 1 to nmats
nmats-Rho	Mass density for materials 1 to nmats
damping	Rayleigh system damping (1 = Yes, 0 = No)
rmdc	Rayleigh mass coefficient
rkdc	Rayleigh stiffness coefficient
mmf	Lumped mass matrix (1 = Yes, 0 = No)

Table A-5: FEAPI::RESTRAINTS

This data block defines restrained/constrained degrees of freedom.

```
FEAPI::RESTRAINTS
rdof
! IF (rdof > 0)
n-1 dof-1 … dof-nodof
:
:
n-rdof dof-1 … dof-nodof
! END IF
```

rdof	Total number of nodes with restrained DOF
n-rdof	Restrained node number from 1 to rdof
dof-nodof	Restraint flag 1 to nodof (1 = restrained, 0 = free)

Table A-6: FEAPI::LOADS

This data block defines loaded degrees of freedom.

```
FEAPI::LOADS
ldof
IF (ldof > 0)
n-1 dof-1 … dof-nodof lfun ldur lwav
:
:
n-ldof dof-1 … dof-nodof lfun ldur lwav
END IF
```

ldof	Total number of nodes with loaded DOF
n-ldof	Loaded node number 1 to ldof
dof-nodof	Load magnitude for DOF 1 to nodof
lfun	Time proportional function (Linear/Step/Square/Since/HalfSine/Triangle/Sawtooth)
ldur	Load duration
lwav	Load wavelength

Table A-7: FEAPI::PRESCRIBED

This data block defines prescribed (displacement) degrees of freedom.

```
FEAPI::PRESCRIBED
pdof
! IF (pdof > 0)
n-1 dof-1 … dof-nodof pfun pdur pwav
:
:
n-pdof dof-1 … dof-nodof pfun pdur pwav
! END IF
```

pdof	Total number of nodes with prescribed DOF
n-pdof	Loaded node number 1 to pdof
dof-nodof	Load magnitude for DOF 1 to nodof
pfun	Time proportional function (Linear/Step/Square/Since/HalfSine/Triangle/Sawtooth)
pdur	Prescribed displacement duration
pwav	Prescribed displacement wavelength

Table A-8: FEAPI::TRANSIENT

This data block defines FEAPI transient analysis options.

```
! IF (atype == transient)
FEAPI::TRANSIENT
meth
! IF (meth == Newmark)
beta gamma delta
! END IF
! IF (meth == WBZ)
beta gamma alpham delta
! END IF
! IF (meth == HHT)
beta gamma alphaf delta
! END IF
! IF (meth == Generalized)
beta gamma alpham alphaf delta
! END IF
dt nsteps endt
```

`meth`	Direct integration method (Newmark/WBZ/HHT/Generalized)
`beta`	Newmark parameter β
`gamma`	Newmark parameter γ
`delta`	Amplification decay factor δ
`alpham`	Generalized-α parameter α_m
`alphaf`	Generalized-α parameter α_f
`dt`	Integration time-step Δt
`nsteps`	Total number of integration steps N
`endt`	Simulation termination time

Table A-9: FEAPI::INTERFACE

This data block defines sub-domain interface data.

```
! IF (inn > 0)
FEAPI::INTERFACE
inum inn
n-1
n-2
:
:
n-inn
! DO (1 to inum)
iid itype nin
! IF (nin > 0)
n-1
n-2
:
:
n-nin
! END IF
! END DO
! END IF
```

inum	Total number of sub-domain interfaces
inn	Total number of sub-domain interface nodes
n-inn	Interface node number t to inn
iid	Interface ID
itype	Interface type (Master/Slave)
nin	Total number of nodes on interface iid

Table A-10: FEAPI::POST

This data block defines FEAPI post analysis options.

```
FEAPI::POST
resf
rflag1
rflag2
rflag3
rflag4 res4k
rflag5
rflag6
rflag7
rflag8
rflag9
```

`resf`	Result frequency
`rflag1`	Post nodal displacements (1 = Yes, 0 = No)
`rflag2`	Post nodal velocities (1 = Yes, 0 = No)
`rflag3`	Post nodal accelerations (1 = Yes, 0 = No)
`rflag4` `res4k`	Post element stresses (1 = Yes, 0 = No) Result keyword (1 = Cauchy Stress, 2 = Von Mises Stress)
`rflag5`	Post element strains (1 = Yes, 0 = No)
`rflag6`	Post domain kinetic energy (1 = Yes, 0 = No)
`rflag7`	Post stiffness energy kinetic energy (1 = Yes, 0 = No)
`rflag8`	Post domain external work (1 = Yes, 0 = No)
`rflag9`	Post domain interface energy (1 = Yes, 0 = No)

Appendix B: FEAPI Output Files

FEAPI generates a series of output files that contain a variety of post-simulation results.

B.1 Post Mesh File

Post mesh file (**.post.msh*) contains nodal coordinates and element connectivity's for displaying the mesh while post-processing overlaid results in GiD.

B.2 Post Result Files

Post result file (**.post.res*) contains post-simulation data such as vector results (displacements, velocities, accelerations), matrix results (stresses, strains). This file, in combination with the post mesh file, is used for result post-processing using GiD.

B.3 Comma Separated Value File

'Comma separated value' file (**.csv*) contains user defined outputs for vector results. By default, options are available for extracting domain results such as kinetic energy, potential energy, external work and interface energy.

B.4 Simulation Summary File

Simulation summary file (*FEAPI-summary.txt*) contains brief summary about the completed simulation. It outputs global/local domain information as well as FEAPI block execution times. Following is an example of the FEAPI simulation summary file:

```
=======================================================================

         Finite Element Analysis Programming Interface (FEAPI)
                             Version 1.0

                    Department Of Civil Engineering
               School of Engineering and Applied Science
                     George Washington University

=======================================================================
 >> MULTIPLE GRID MULTIPLE TIME-SCALE (MGMT) SIMULATIONS
 >> PROJECT                  :
 >> NUMBER OF DOMAINS        :
 >> DATE AND TIME            :
=======================================================================
 >> MGMT SUB-DOMAIN SUMMARY
    ----------------------|------|------------|------------|-------
    File Name             | Rank | dt         | nsteps     | mratio
    ----------------------|------|------------|------------|-------
  +                       |      |            |            |
  +                       |      |            |            |
    ----------------------|------|------------|------------|-------
=======================================================================
 >> GLOBAL INFO
  + FEAPI GLOBAL BLOCK       : 0
    GiD Post Mesh File       :
    GiD Post Result File     :
    GiD Post Result File     :
  + GLOBAL PARAMETERS
    Analysis Type            :
    Problem Type             :
    Element Type             :
    Number of Dimensions     :
    Number of Nodes          :
    Number of Elements       :
    Number of Nodes/Element  :
    Degrees of Freedoms/Node :
    Number of Gauss Points   :
    Number of Equations      :
=======================================================================
 >> DOMAIN INFO
  + FEAPI DOMAIN BLOCK       : 1
  + DOMAIN FILES
    GiD Input File           :
    GiD Post Mesh File       :
    GiD Post Result File     :
    GiD Post Result File     :
  + DOMAIN PARAMETERS
    Analysis Type            :
    Problem Type             :
    Element Type             :
    Number of Dimensions     :
    Number of Nodes          :
    Number of Elements       :
    Number of Nodes/Element  :
    Degrees of Freedoms/Node :
    Number of Gauss Points   :
    Number of Equations      :
    Skyline storage          :
    Integration Time-Step    :
    Total Number of Steps    :
=======================================================================
 >> FEAPI BLOCK EXECUTION TIMES
    PROGRAM BLOCK                SECONDS         % TOTAL
  + PTERMINAL                 :
  + PFILENAME                 :
  + PALLOC                    :
  + PINPUT                    :
  + PSETUP                    :
  + PRESULT                   :
  + PSOLVE                    :
  + PPOST                     :
    TOTAL                     :
======================================================================
```

Appendix C: Pre/Post Example

Here, we will consider an example of forced vibration analysis of a 2D cantilever beam as shown in Figure C-1. The domain will be discretized using 4-node quadrilateral elements (2 DOF/node) with plane stress formulation, consistent mass matrix and zero damping. Isotropic linear elastic material properties with modulus of elasticity (E) = 207x10^9 N/m^2, Poisson's ratio (ν) = 0.3 and mass density (ρ) = 7830 Kg/m^3 will be used.

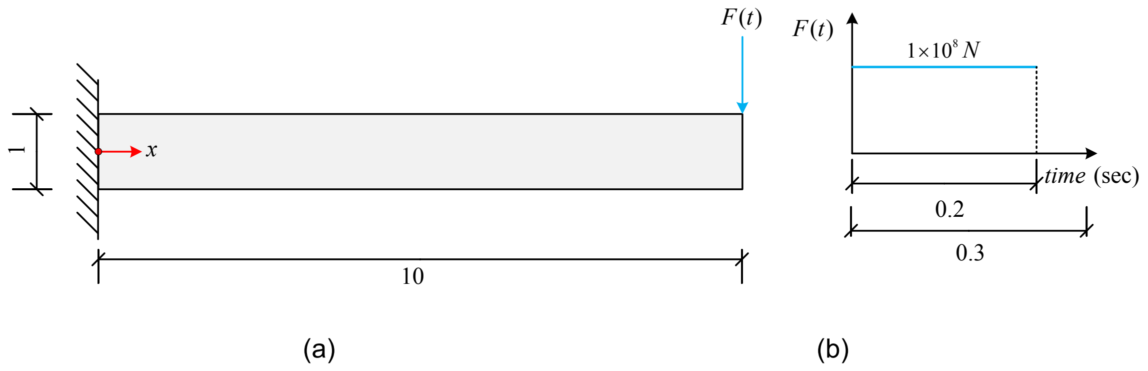


Figure C-1: (a) Domain under consideration – 2D cantilever beam (b) Transverse (step) loading

C.1 Program 1: Uniform Grid Uniform Time-scale Simulations

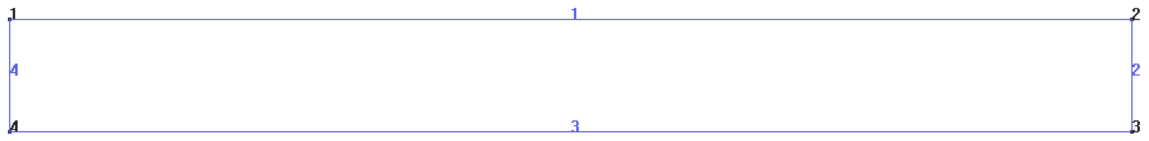
Grid spacing (H)	Newmark parameters	Time-step (ΔT)
0.25	β=0.25, γ=0.5 (Implicit)	0.5x10^{-3}

C.1.1 Pre-processing (Input File Creation)

1) Start GiD.

2) Select 'Create Line'. (Geometry > Create > Straight Line)

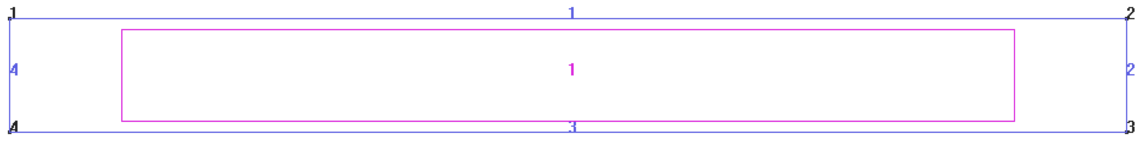
   - Enter points to define line (0, 0.5) (10, 0.5) (10, -0.5) (0, -0.5) (0, 0.5). Join > Escape.
   - Right click > Label > All.



3) Select 'Create NURBS surface'. (Geometry > Create > NURBS surface > By contour)

   ▪ Enter lined to define NURBS surface (1, 2, 3, and 4). Escape.

   ▪ Right click > Label > All.



4) Save workspace.

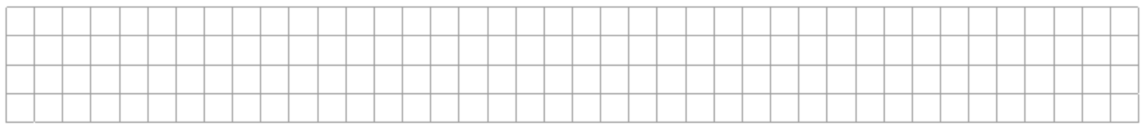
   ▪ (File > Save > 'example1')

5) Assign element type.

   ▪ (Mesh > Element Type > Quadrilateral). Enter surfaces to assign this element type (1). Enter > Escape.

   ▪ (Mesh > Quadratic Type > Normal).

6) (Mesh > Structured > Surfaces > Assign Size). Select 4-sided or NURBS surface to define structured mesh (1). Enter > Escape. Enter mesh size to assign to lines (0.25). Select lines to define structured mesh (2 and 4). Enter mesh size to assign to lines (0.25). Select lines to define structured mesh (1 and 3). Enter > Escape > Close. (Mesh > Generate Mesh). OK > View Mesh.



7) Select problem type. (Data > Problem Type > feapi-gid)

8) Assign material properties. (Data > Materials)

- Select Steel. Assign > Elements > Enter elements to assign Material: Steel. Select all elements. Escape > Close.

9) Define global variables. (Data > Problem Data > Global Variables)

- Analysis type: Transient
- Problem type: PlaneStress
- Element type: Quadrilateral
- DOF per node: 2
- Gauss points: 4
- Accept > Close.

10) Define boundary conditions. (Data > Conditions)

- Assign Restrained DOFs. Check DOF 1 (x) and DOF 2 (y). Assign. Select all nodes on line 4. Escape.
- Assign Forced DOFs. Enter -0.2e8 (1x10^8 /5 nodes) for DOF 2 (y). Assign. Select all nodes on line 2. Escape.

  Load function: Step

  Load duration: 0.2

  Load wavelength: 0.0

- Close.

11) Define analysis options. (Data > Problem Data > Transient Analysis Options)

- Direct Integration

  Integration method: Newmark

  Newmark parameter (Beta): 0.25

  Newmark parameter (Gamma): 0.5

  Generalized parameter (Alpha m): 0.0

Generalized parameter (Alpha f): 0.0

Amplification decay factor: 0.0

Time step: 0.5e-3

Number of steps: 600

End time: 0.3

- Inertial and System Damping

  Uncheck lumped mass approximation

  Uncheck Rayleigh damping

- Accept. Close.

12) Define post analysis options. (Data > Problem Data > Post Result Options)

- General

  Post frequency: 2

- Nodal results. Check all (Displacements, Velocities and Accelerations)

- Element results.

  Select Cauchy stresses

  Check Strains.

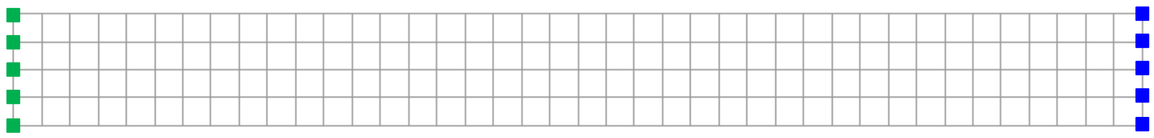
- Domain results. Select kinetic energy, stiffness energy and external work.

- Accept. Close.

13) Save workspace.

14) Generate FEAPI input file. (Calculate > Calculate)

Windows batch file (feapi-gid.win.bat) is executed on Calculate. It will copy the input file (example1.dat) to FEAPI input directory defined using `FEAPI::INPUT`.



🟩 Fixed nodes

🟦 Loaded nodes

C.1.2 Solver

1) Run FEAPI.

2) Enter program number: 1 (Forced vibration analysis of linear elastic solids)

3) Enter project title: Example 1

4) Enter base name for input file: example1

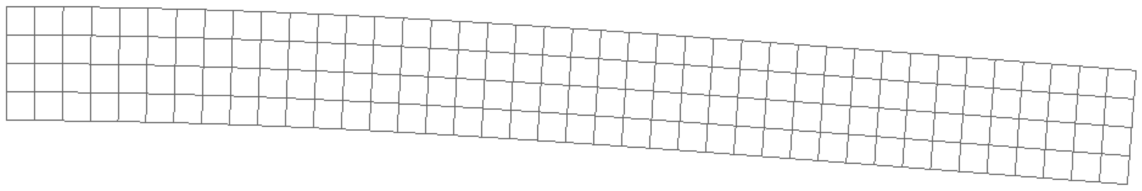
C.1.3 Post-processing (Result Visualization)

1) Start GiD in Post process mode. (File > Postprocess)

2) Browse to FEAPI output directory and open example1.post.msh

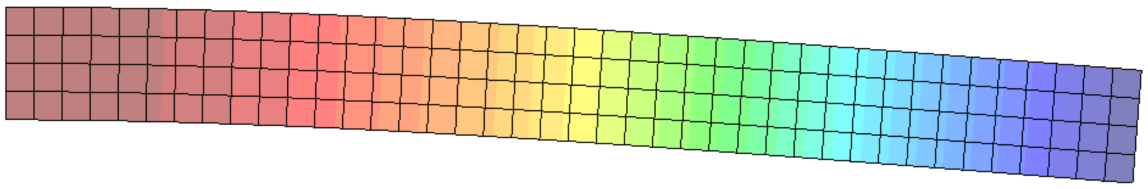
3) Plot deformation.

- (Window > View Results)
- Select Main Mesh as Deformed (Step: 0.3, Result: Displacement, Factor: 0.2).



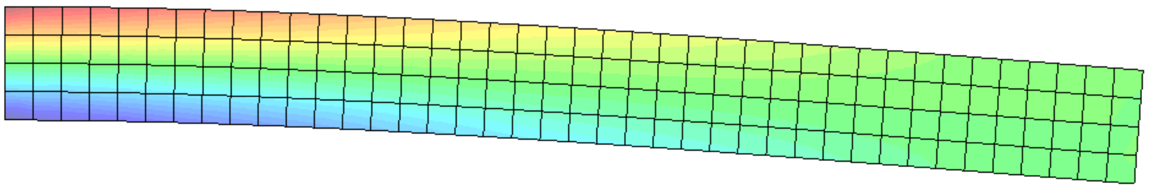
4) Plot deformation contours.

- (View Results > Contour Fill > Displacement > Disp y).



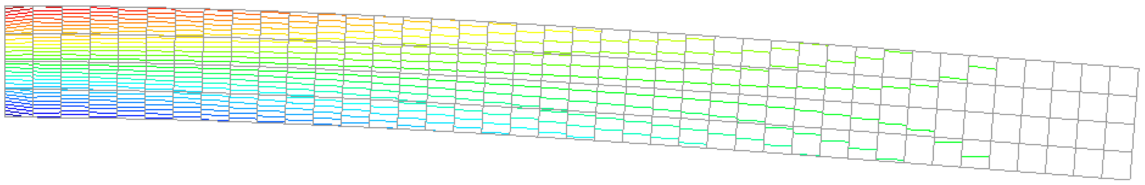
5) Plot stress contours.

- (View Results > Smooth Contour Fill > Stress > Sig xx).



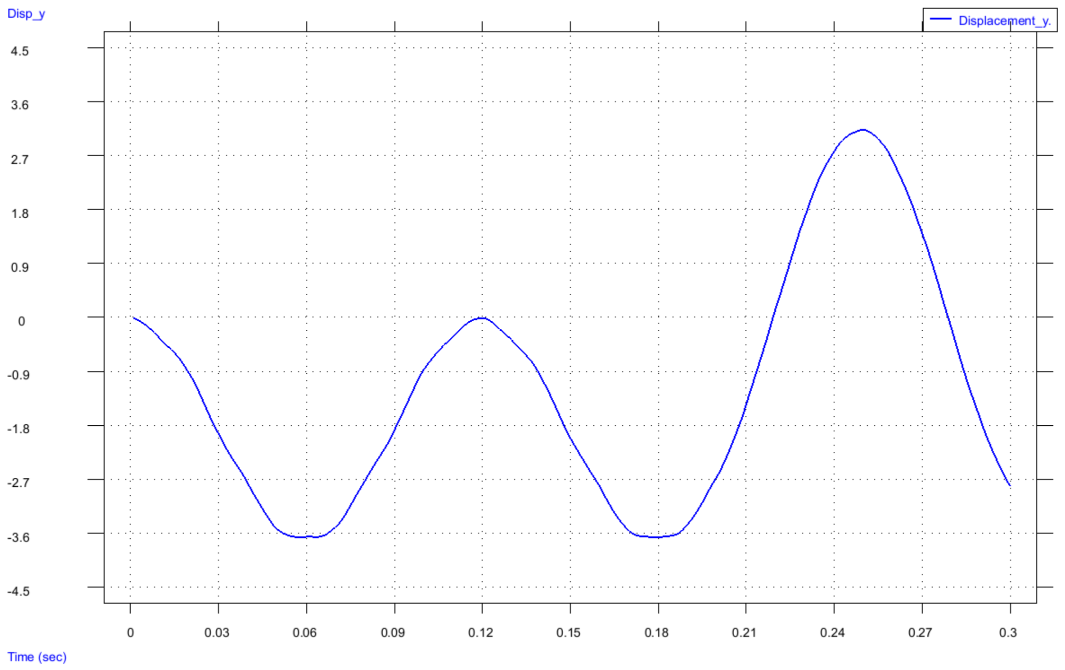
6) Plot stress contour lines.

- (View Results > Contour Lines > Stress > Sig xx).



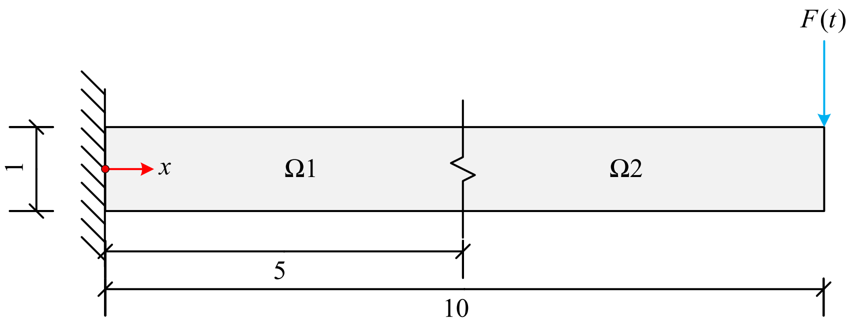
7) Plot deformation graph.

- (View Results > Graphs > Point Evolution > Displacement > Disp y)
- Enter the coordinates of the point to see its evolution (10, 0.0). Enter.



C.2 Program 2: Multiple Grid Multiple Time-scale Simulations

Here, we will decompose the domain under analysis from Figure C-2 into two component sub-domains each with an exclusive spatial and temporal discretization.



Sub-domain	Grid spacing (H)	Newmark parameters	Time-step (Δt)
Ω1	0.0625	β=0.25, γ=0.5 (Implicit)	$0.125\text{x}10^{-3}$
Ω2	0.25	β=0.25, γ=0.5 (Implicit)	$0.5\text{x}10^{-3}$

Figure C-2: Decomposed sub-domains and corresponding time-stepping parameters

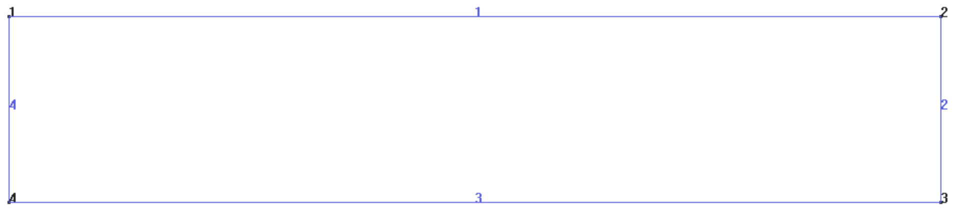
C.2.1 Pre-processing (Input File Creation)

A. Sub-domain 1

1) Start GiD.

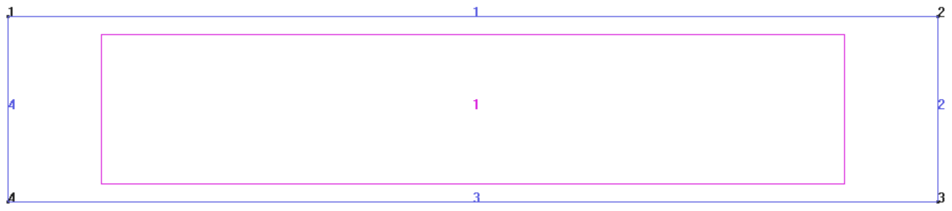
2) Select ‘Create Line’. (Geometry > Create > Straight Line)

- Enter points to define line (0, 0.5) (5, 0.5) (5, -0.5) (0, -0.5) (0, 0.5). Join > Escape.
- Right click > Label > All.



3) Select ‘Create NURBS surface’. (Geometry > Create > NURBS surface > By contour)

- Enter lined to define NURBS surface (1, 2, 3, and 4). Escape.
- Right click > Label > All.



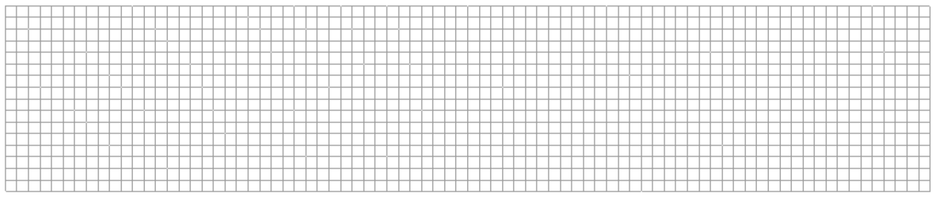
4) Save workspace.

- (File > Save > ‘example1-d1.gid’)

5) Assign element type.

- (Mesh > Element Type > Quadrilateral). Enter surfaces to assign this element type (1). Enter > Escape.
- (Mesh > Quadratic Type > Normal).

6) (Mesh > Structured > Surfaces > Assign Size). Select 4-sided or NURBS surface to define structured mesh (1). Enter > Escape. Enter mesh size to assign to lines (0.0625). Select lines to define structured mesh (2 and 4). Enter mesh size to assign to lines (0.0625). Select lines to define structured mesh (1 and 3). Enter > Escape > Close. (Mesh > Generate Mesh). OK > View Mesh.



7) Select problem type. (Data > Problem Type > feapi-gid)

8) Assign material properties. (Data > Materials)

- Select Steel. Assign > Elements > Enter elements to assign Material: Steel. Select all elements. Escape > Close.

9) Define global variables. (Data > Problem Data > Global Variables)

- Analysis type: Transient
- Problem type: PlaneStress
- Element type: Quadrilateral
- DOF per node: 2
- Gauss points: 4
- Accept > Close.

10) Define boundary conditions. (Data > Conditions)

- Assign Restrained DOFs. Check DOF 1 (x) and DOF 2 (y). Assign. Select all nodes on line 4. Escape.
- Assign Domain Interface Nodes. Assign. Select all nodes on line 2. Escape.
- Define interface info. Select Interface Type – Slave, Interface ID – 1. Select all nodes on line 2. Escape.
- Close.

11) Define analysis options. (Data > Problem Data > Transient Analysis Options)

- Direct Integration

  Integration method: Newmark

  Newmark parameter (Beta): 0.25

  Newmark parameter (Gamma): 0.5

  Generalized parameter (Alpha m): 0.0

  Generalized parameter (Alpha f): 0.0

  Amplification decay factor: 0.0

  Time step: 0.125e-3

Number of steps: 0

End time: 0.3

- Inertial and System Damping

Uncheck lumped mass approximation

Uncheck Rayleigh damping

- Accept. Close.

12) Define post analysis options. (Data > Problem Data > Post Result Options)

- General

Post frequency: 8

- Nodal results. Check all (Displacements, Velocities and Accelerations)
- Element results.

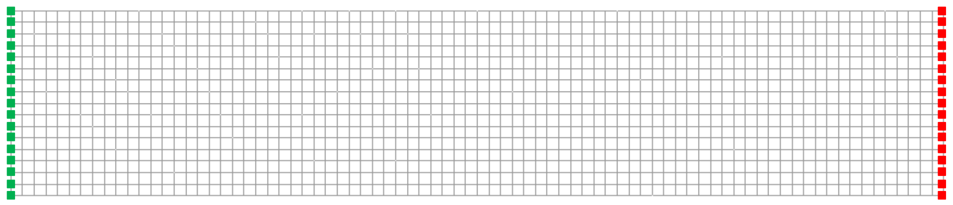
Select Cauchy stresses

Check Strains.

- Domain results. Check kinetic energy, stiffness energy, external work and interface energy.
- Accept. Close.

13) Save workspace.

14) Generate FEAPI input file. (Calculate > Calculate)



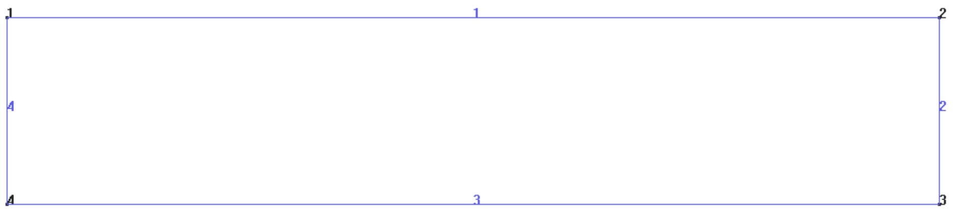
🟩 Fixed nodes
🟥 Interface nodes (Slave, ID 1)

B. Sub-domain 2

1) Start GiD.

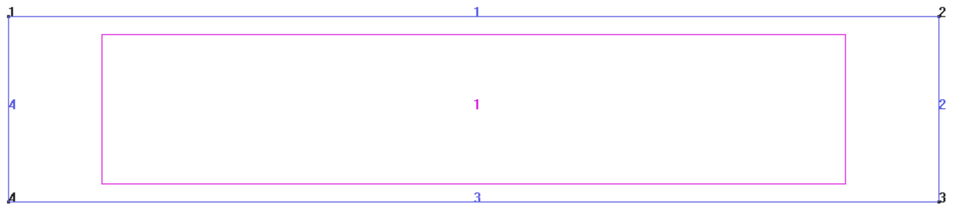
2) Select 'Create Line'. (Geometry > Create > Straight Line)

- Enter points to define line (5, 0.5) (10, 0.5) (10, -0.5) (5, -0.5) (5, 0.5). Join > Escape.
- Right click > Label > All.



3) Select 'Create NURBS surface'. (Geometry > Create > NURBS surface > By contour)

- Enter lined to define NURBS surface (1, 2, 3, and 4). Escape.
- Right click > Label > All.



4) Save workspace.

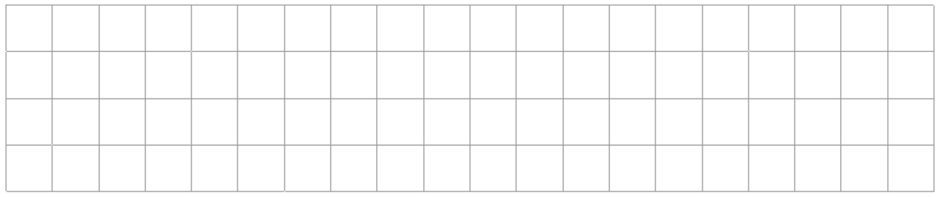
- (File > Save > 'example1-d2.gid')

5) Assign element type.

- (Mesh > Element Type > Quadrilateral). Enter surfaces to assign this element type (1). Enter > Escape.
- (Mesh > Quadratic Type > Normal).

6) (Mesh > Structured > Surfaces > Assign Size). Select 4-sided or NURBS surface to define structured mesh (1). Enter > Escape. Enter mesh size to assign to lines

(0.25). Select lines to define structured mesh (2 and 4). Enter mesh size to assign to lines (0.25). Select lines to define structured mesh (1 and 3). Enter > Escape > Close. (Mesh > Generate Mesh). OK > View Mesh.



7) Select problem type. (Data > Problem Type > feapi-gid)

8) Assign material properties. (Data > Materials)

- Select Steel. Assign > Elements > Enter elements to assign Material: Steel. Select all elements. Escape > Close.

9) Define global variables. (Data > Problem Data > Global Variables)

- Analysis type: Transient
- Problem type: PlaneStress
- Element type: Quadrilateral
- DOF per node: 2
- Gauss points: 4
- Accept > Close.

10) Define boundary conditions. (Data > Conditions)

- Assign Forced DOFs. Enter -0.2e8 (1x108 /5 nodes) for DOF 2 (y). Assign. Select all nodes on line 2. Escape.
- Assign Domain Interface Nodes. Assign. Select all nodes on line 4. Escape.
- Define interface info. Select Interface Type – Master, Interface ID – 1. Select all nodes on line 4. Escape.
- Close.

11) Define analysis options. (Data > Problem Data > Transient Analysis Options)

▪ Direct Integration

Integration method: Newmark

Newmark parameter (Beta): 0.25

Newmark parameter (Gamma): 0.5

Generalized parameter (Alpha m): 0.0

Generalized parameter (Alpha f): 0.0

Amplification decay factor: 0.0

Time step: 0.5e-3

Number of steps: 0

End time: 0.3

▪ Inertial and System Damping

Uncheck lumped mass approximation

Uncheck Rayleigh damping

▪ Accept. Close.

12) Define post analysis options. (Data > Problem Data > Post Result Options)

▪ General

Post frequency: 2

▪ Nodal results. Check all (Displacements, Velocities and Accelerations)

▪ Element results.

Select Cauchy stresses

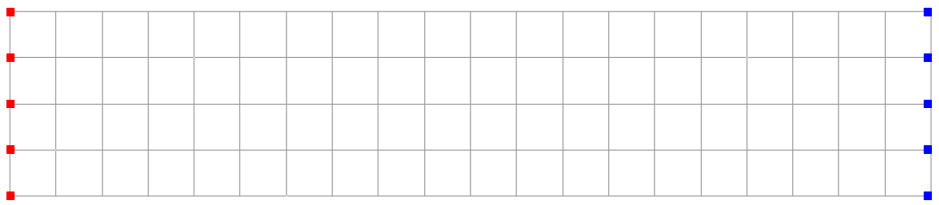
Check Strains.

▪ Domain results. Check kinetic energy, stiffness energy, external work and interface energy.

▪ Accept. Close.

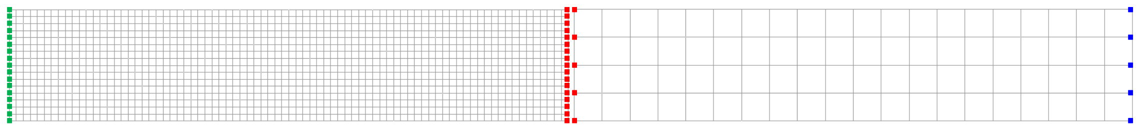
13) Save workspace.

14) Generate FEAPI input file. (Calculate > Calculate)



🟦 Forced nodes
🟥 Interface nodes (Master, ID 1)

Global representation:



Before starting the simulation, make sure feapi-configuration.txt has necessary values defined under data block – `MGMT::POST.`

```
MGMT::POST
1! 1 = Post frequency is same as global time-step sub-domain
1! Displacements: 1 = Yes
1! Velocities: 1 = Yes
1! Accelerations: 1 = Yes
1 1! Stresses (Cauchy/Von Mises): 1 = Yes
1! Strains: 1 = Yes
1! Kinetic energy: 1 = Yes
1! Stiffness energy: 1 = Yes
1! External work: 1 = Yes
1! Interface energy: 1 = Yes
```

C.2.2 Solver

1) Run FEAPI.

2) Enter program number: 2 (Multiple grid multiple time-scale simulations)

3) Enter project title: Example 1 MGMT

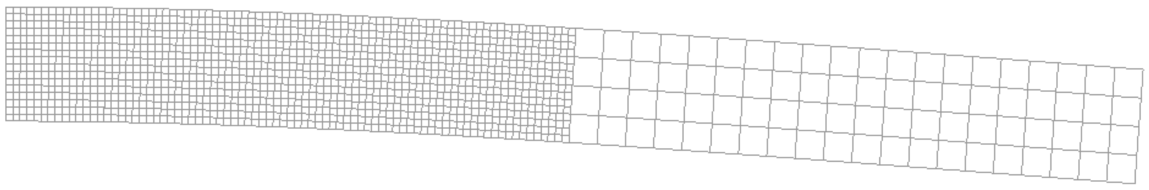
4) Enter base name for input file 1: example1-d1

5) Enter base name for input file 2: example1-d2

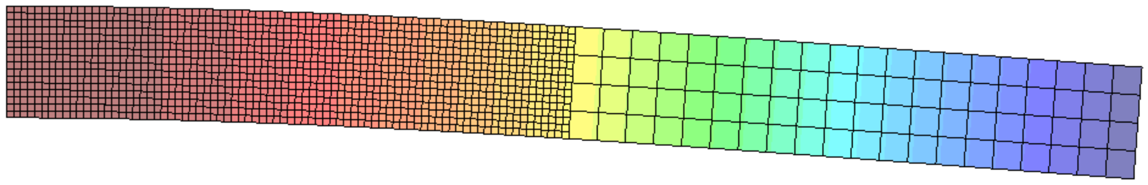
C.2.3 Post-processing (Result Visualization)

In this example, FEAPI generates outputs for local sub-domain results as well as global results plotted over multiple grids. Local sub-domain results may be post-processed using corresponding **.post.msh* files and global results for the original problem may be post-processed using *global.post.msh* as follows:

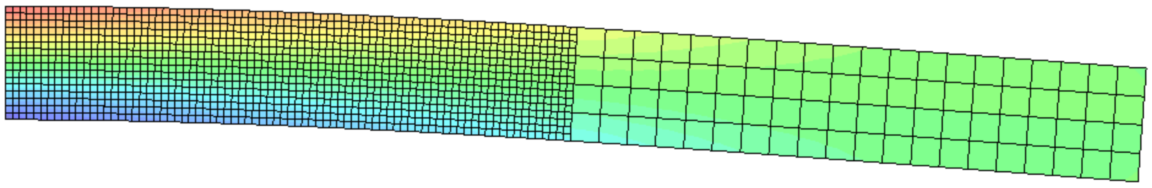
1) Start GiD in Post process mode. (File > Postprocess)
2) Browse to FEAPI output directory and open Global.post.msh
3) Plot deformation.
   - (Window > View Results)
   - Select Main Mesh as Deformed (Step: 0.3, Result: Displacement, Factor: 0.2).



4) Plot deformation contours.
   - (View Results > Contour Fill > Displacement > Disp y).

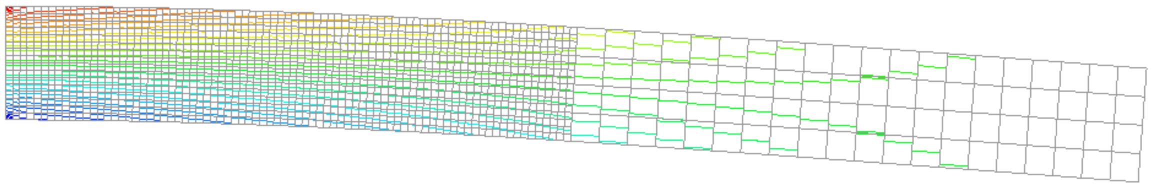


5) Plot stress contours.
   - (View Results > Smooth Contour Fill > Stress > Sig xx).



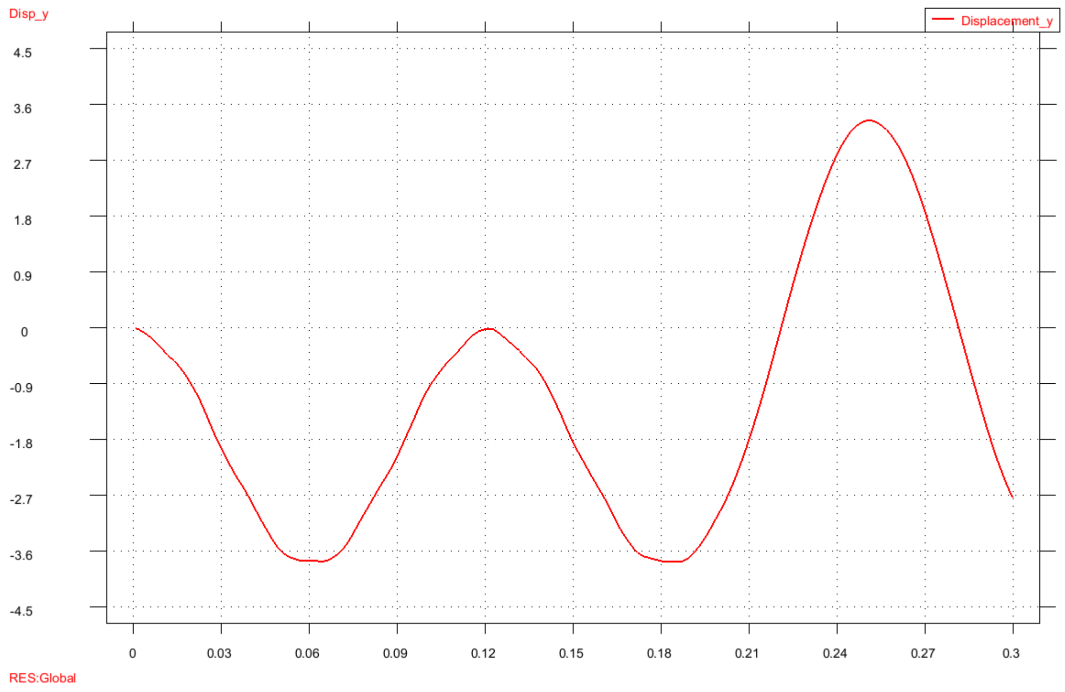
6) Plot stress contour lines.

- (View Results > Contour Lines > Stress > Sig xx).



7) Plot deformation graph.

- (View Results > Graphs > Point Evolution > Displacement > Disp y)
- Enter the coordinates of the point to see its evolution (10, 0.0). Enter.



Appendix D: Library Routines

D.1 FEAPI

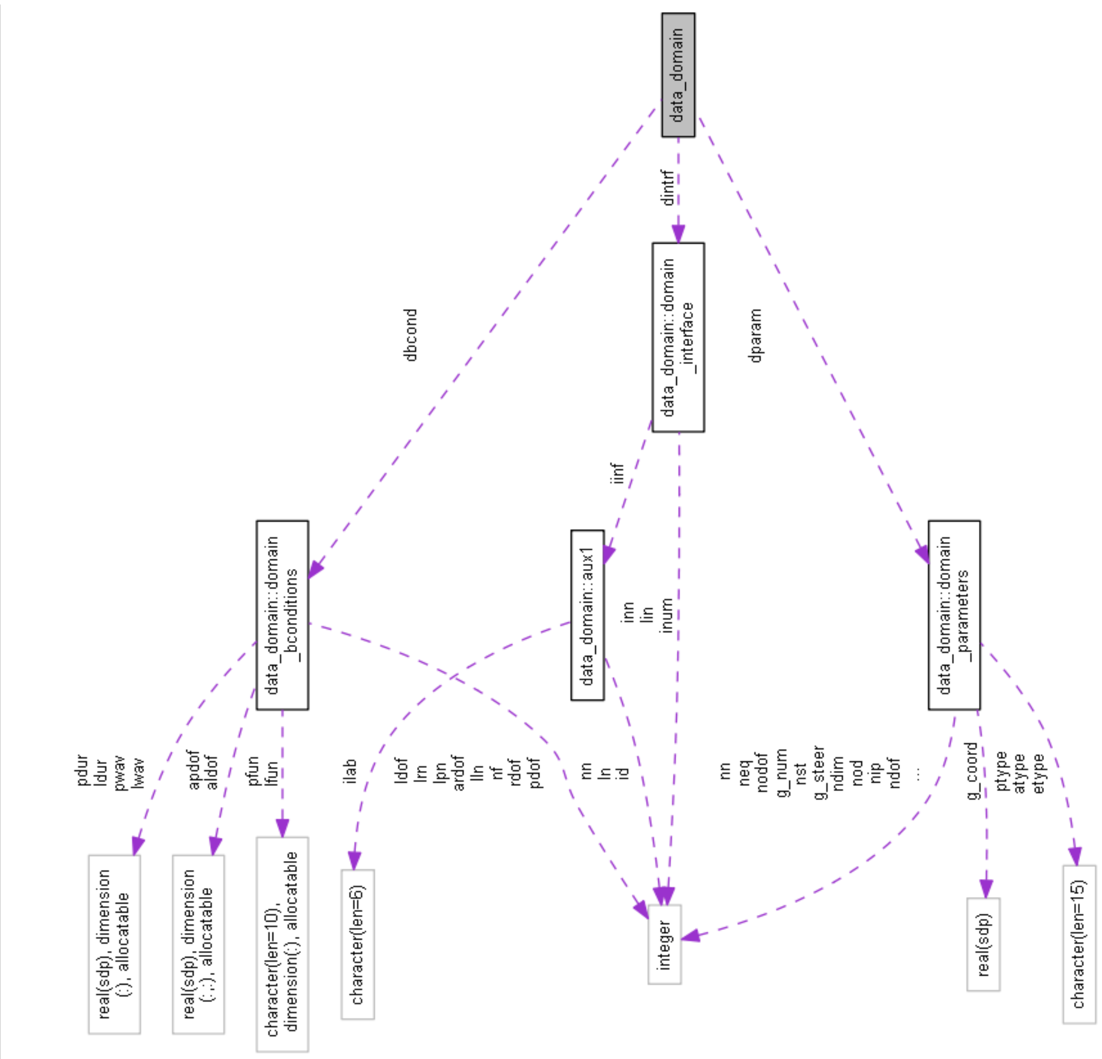
D.1.1 Modules

A. data_domain.f90

```fortran
! This module contains FEAPI domain variables.
MODULE data_domain
USE precision
IMPLICIT NONE
! Data type for FEAPI-domain parameters.
TYPE :: domain_parameters
       ! Spatial dimensions of finite element domain.
       INTEGER :: ndim
       ! Total number of nodes.
       INTEGER :: nn
       ! Total number of elements.
       INTEGER :: nels
       ! Total number of nodes per element.
       INTEGER :: nod
       ! Number of degrees of freedom per node.
       INTEGER :: nodof
       ! Number of degrees of freedom per element.
       INTEGER :: ndof
       ! Number of Gauss integration points.
       INTEGER :: nip
       ! Number of stress/strain terms.
       INTEGER :: nst
       ! Total number of equations.
       INTEGER :: neq
       ! Analysis Type.
       CHARACTER(LEN = 15) :: atype
       ! Problem Type.
       CHARACTER(LEN = 15) :: ptype
       ! Element Type.
       CHARACTER(LEN = 15) :: etype
       ! Array of nodal coordinates.
       REAL(sdp), ALLOCATABLE :: g_coord(:,:)
       ! Array of element connectivity.
       INTEGER, ALLOCATABLE :: g_num(:,:)
       ! Element degree of freedom steering vectors.
       INTEGER, ALLOCATABLE :: g_steer(:,:)
END TYPE domain_parameters
! Data type for FEAPI-domain boundary conditions.
TYPE :: domain_bconditions
       ! Array of numbered nodal degrees of freedom.
       INTEGER, ALLOCATABLE :: nf(:,:)
       ! Total number of nodes with restrained degrees of freedom.
       INTEGER :: rdof
       ! List of restrained nodes.
       INTEGER, ALLOCATABLE :: lrn(:)
       ! Array of restrained degrees of freedom.
       INTEGER, ALLOCATABLE :: ardof(:,:)
       ! Total number of nodes with loaded degrees of freedom.
       INTEGER :: ldof
```

```
        ! List of loaded nodes.
        INTEGER, ALLOCATABLE :: lln(:)
        ! Array of loaded degrees of freedom.
        REAL(sdp), ALLOCATABLE :: aldof(:,:)
        ! Total number of nodes with prescribed (displacement) degrees of freedom.
        INTEGER :: pdof
        ! List of (displacement) prescribed nodes.
        INTEGER, ALLOCATABLE :: lpn(:)
        ! Array of (displacement) prescribed degrees of freedom.
        REAL(sdp), ALLOCATABLE :: apdof(:,:)
        ! Load function
        CHARACTER(LEN = 10), ALLOCATABLE :: lfun(:)
        ! Load duration
        REAL(sdp), ALLOCATABLE :: ldur(:)
        ! Load wavelength
        REAL(sdp), ALLOCATABLE :: lwav(:)
        ! Prescribed (displacement) funtion
        CHARACTER(LEN = 10), ALLOCATABLE :: pfun(:)
        ! Prescribed (displacement) duration
        REAL(sdp), ALLOCATABLE :: pdur(:)
        ! Prescribed (displacement) wavelength
        REAL(sdp), ALLOCATABLE :: pwav(:)
END TYPE domain_bconditions
! Auxiliary data type for domain interface info.
TYPE :: aux1
        ! Interface ID.
        INTEGER :: id
        ! Interface label.
        CHARACTER(LEN = 6)    :: ilab
        ! Number of nodes on a particular interface.
        INTEGER :: nn
        ! List of nodes on a particular interface.
        INTEGER, ALLOCATABLE :: ln(:)
END TYPE aux1
! Data type for domain interface info.
TYPE :: domain_interface
        ! Total number of interfaces.
        INTEGER  :: inum
        ! Total number of interface nodes.
        INTEGER  :: inn
        ! List of interface nodes.
        INTEGER, ALLOCATABLE :: lin(:)
        ! Domain interface info.
        TYPE(aux1), ALLOCATABLE :: iinf(:)
END TYPE domain_interface
! Derived data type for FEAPI-domain parameters.
TYPE(domain_parameters), ALLOCATABLE :: dparam(:)
! Derived data type for FEAPI-domain boundary conditions.
TYPE(domain_bconditions), ALLOCATABLE :: dbcond(:)
! Derived data type for domain interface info.
TYPE(domain_interface), ALLOCATABLE :: dintrf(:)
END MODULE data_domain
```

DATA COLLABORATION

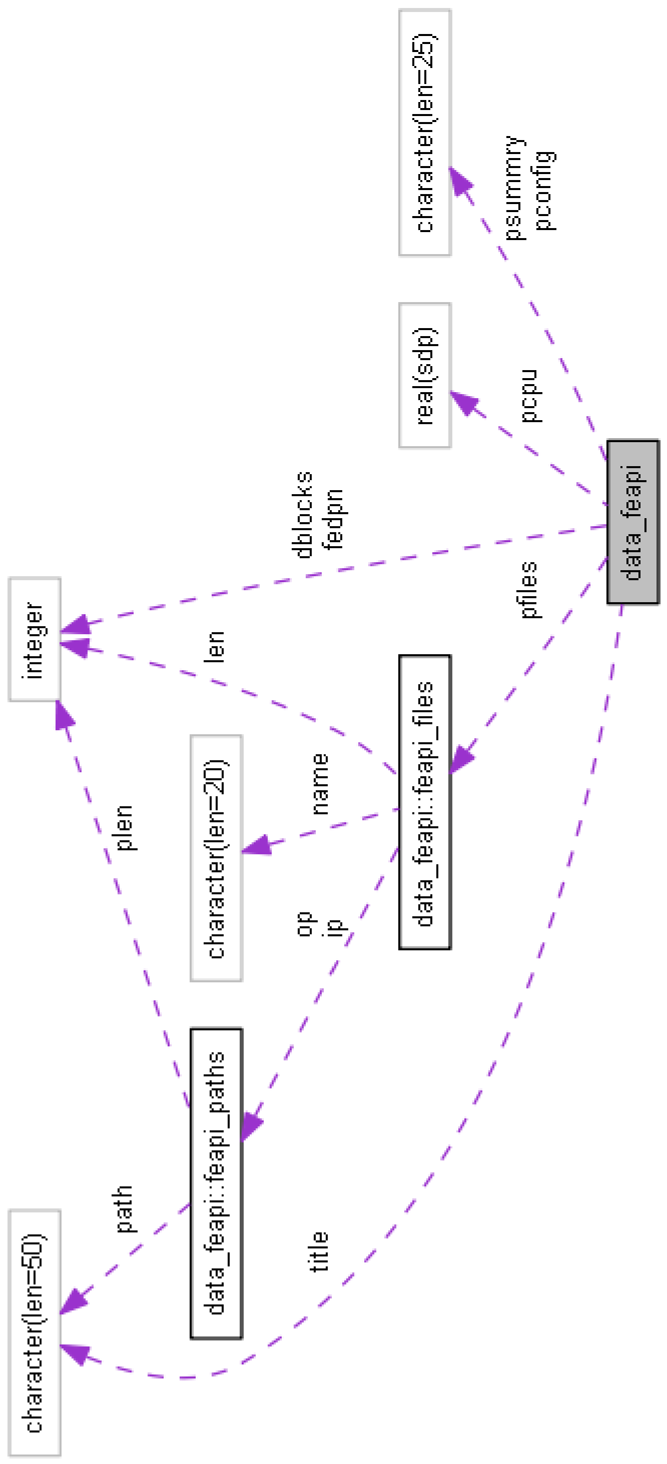

data_domain
dintrf
dbcond
dparam
data_domain::domain_interface
iinf
data_domain::domain_bconditions
data_domain::aux1
data_domain::domain_parameters
inn lin inum
pdur ldur pwav lwav
apdof aldof
pfun lfun
ilab
ldof lrn lpn ardof lln nf rdof pdof
nn ln id
nn neq nodof g_num nst g_steer ndim nod nip ndof ...
g_coord
ptype atype etype
real(sdp), dimension(:), allocatable
real(sdp), dimension(:,:), allocatable
character(len=10), dimension(:), allocatable
character(len=6)
integer
real(sdp)
character(len=15)

B. data_feapi.f90

```
! This module contains FEAPI program variables.
MODULE data_feapi
USE precision
IMPLICIT NONE
! FEAPI configuration file name.
CHARACTER(LEN = 25) :: pconfig = 'feapi-configuration.txt'
! FEAPI summary file name.
CHARACTER(LEN = 25) :: psummry = 'feapi-summary.txt'
! FEAPI driver program number.
INTEGER :: fedpn
! Project title.
CHARACTER(LEN = 50) :: title
! Number of finite element domain blocks.
INTEGER :: dblocks
! FEAPI computation times.
REAL(sdp),  ALLOCATABLE :: pcpu(:)
! Data type for FEAPI include directories.
TYPE :: feapi_paths
```

```
       ! Directory path.
       CHARACTER(LEN = 50) :: path
       ! Directory path character length.
       INTEGER :: plen
END TYPE feapi_paths
! Data type for FEAPI files.
TYPE :: feapi_files
       ! File (base) name.
       CHARACTER(LEN = 20) :: name
       ! File name character length.
       INTEGER :: len
       ! Input file location.
       TYPE(feapi_paths) :: ip
       ! Output file location.
       TYPE(feapi_paths) :: op
END TYPE feapi_files
! Derived data type for FEAPI input files.
TYPE(feapi_files), ALLOCATABLE :: pfiles(:)
END MODULE data_feapi
```

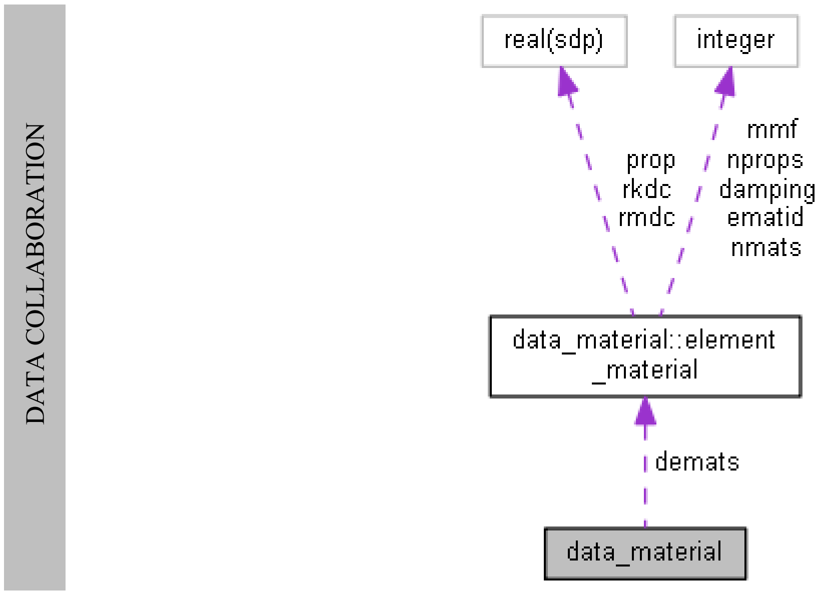
DATA COLLABORATION



C. data_material.f90

```
! This module contains FEAPI domain material variables.
MODULE data_material
USE precision
IMPLICIT NONE
```

```
! Data type for FEAPI-domain element material properties.
TYPE :: element_material
       ! Number of materials used.
       INTEGER :: nmats
       ! Number of material properties.
       INTEGER :: nprops
       ! Element material ID.
       INTEGER, ALLOCATABLE :: ematid(:)
       ! Array of material properties.
       REAL(sdp), ALLOCATABLE :: prop(:,:)
       ! System damping.
       INTEGER :: damping = 0
       ! Rayleigh mass coefficient.
       REAL(sdp):: rmdc = 0.0_sdp
       ! Rayleigh stiffness coefficient.
       REAL(sdp):: rkdc = 0.0_sdp
       ! Mass matrix formulation.
       INTEGER :: mmf = 0
END TYPE element_material
! Derived data type for FEAPI-domain element material properties.
TYPE(element_material), ALLOCATABLE :: demats(:)
END MODULE data_material
```

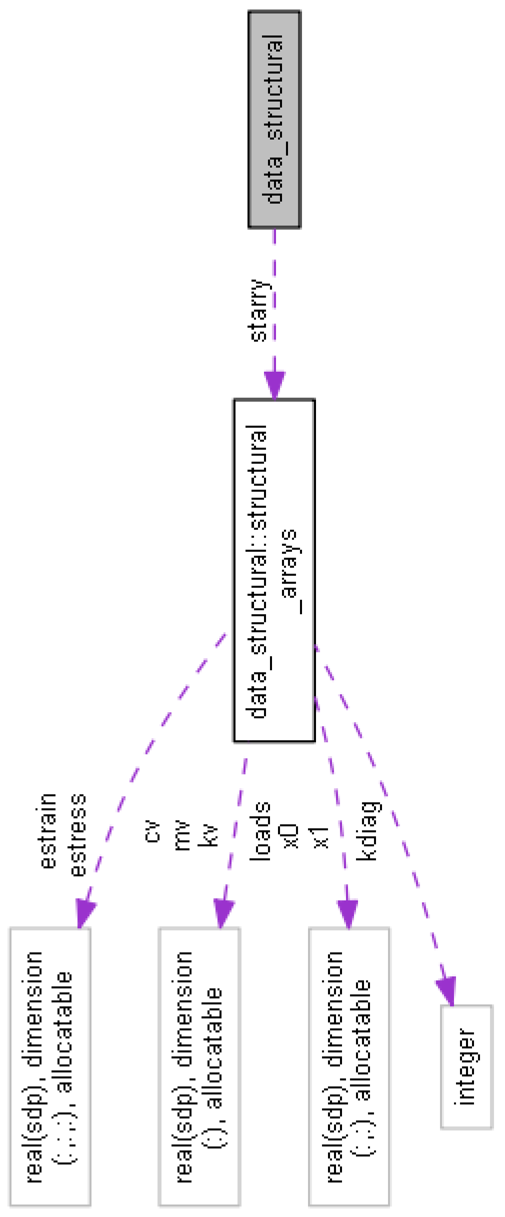


D. data_structural.f90

```
! This module contains FEAPI structural system variables.
MODULE data_structural
USE precision
IMPLICIT NONE
! Data type for structural (symmetric/skyline) system arrays.
TYPE :: structural_arrays
       ! Mass matrix (skyline).
       REAL(sdp), ALLOCATABLE :: mv(:)
       ! Damping matrix (skyline).
       REAL(sdp), ALLOCATABLE :: cv(:)
       ! Stiffness matrix (skyline).
       REAL(sdp), ALLOCATABLE :: kv(:)
       ! Diagonal term locator for skyline mapping.
```

```
        INTEGER, ALLOCATABLE :: kdiag(:)
        ! Initial kinematic quantities.
        REAL(sdp), ALLOCATABLE :: x0(:,:)
        ! Final kinematic quantities.
        REAL(sdp), ALLOCATABLE :: x1(:,:)
        ! R.H.S load vector.
        REAL(sdp), ALLOCATABLE :: loads(:,:)
        ! Element stresses (computed at Gauss integration points).
        REAL(sdp), ALLOCATABLE :: estress(:,:,:)
        ! Element strains (computed at Gauss integration points).
        REAL(sdp), ALLOCATABLE :: estrain(:,:,:)
END TYPE structural_arrays
! Derived data type for structural (symmetric/skyline) system arrays.
TYPE(structural_arrays), ALLOCATABLE :: starry(:)
END MODULE data_structural
```

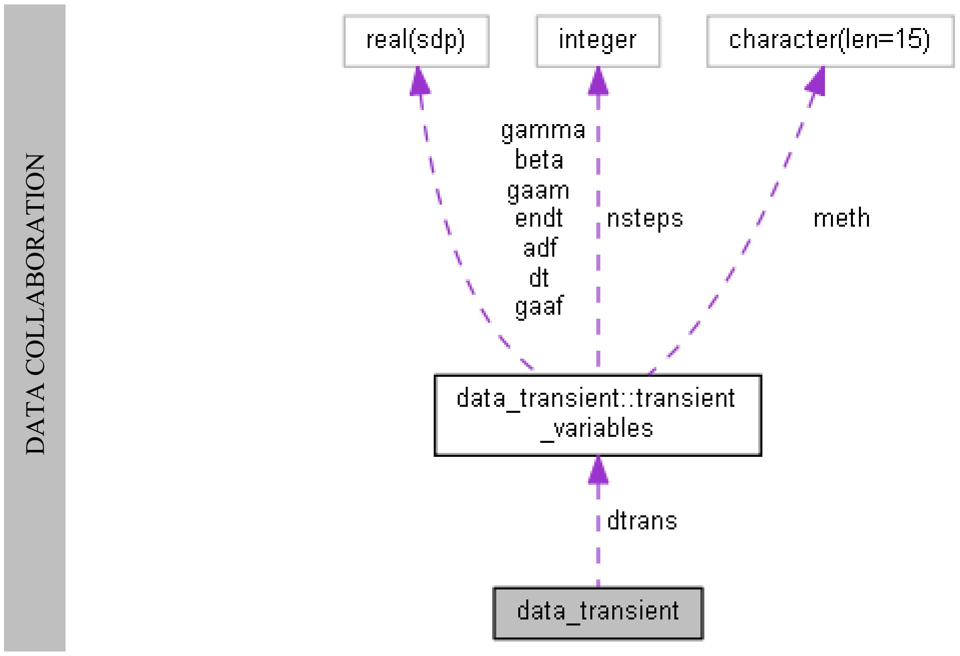
DATA COLLABORATION



data_structural
starry
data_structural::structural_arrays
estrain
estress
cv
mv
kv
loads
x0
x1
kdiag
real(sdp), dimension (:,:,:), allocatable
real(sdp), dimension (:), allocatable
real(sdp), dimension (:,:), allocatable
integer

E. data_transient.f90

```
! This module contains FEAPI domain transient analysis variables.
MODULE data_transient
USE precision
IMPLICIT NONE
! Data type for FEAPI-domain transient analysis variables.
TYPE :: transient_variables
       ! Direct integration method.
       CHARACTER(LEN = 15) :: meth
       ! Newmark parameter.
       REAL(sdp) :: beta = 0.25_sdp
       ! Newmark parameter.
       REAL(sdp) :: gamma = 0.5_sdp
       ! Generalized Alpha parameter.
       REAL(sdp) :: gaam
       ! Generalized Alpha parameter.
       REAL(sdp) :: gaaf
       ! Amplification Decay Factor.
       REAL(sdp) :: adf = 0.0_sdp
       ! Time-step.
       REAL(sdp) :: dt
       ! Termination time.
       REAL(sdp) :: endt
       ! Total number of steps.
       INTEGER :: nsteps
END TYPE transient_variables
! Derived data type for FEAPI-domain transient analysis variables.
TYPE(transient_variables), ALLOCATABLE :: dtrans(:)
END MODULE data_transient
```



F. precision.f90

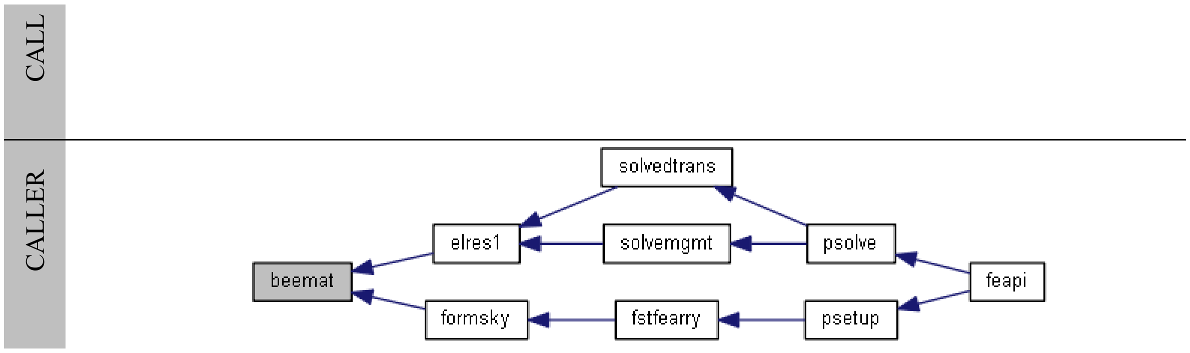
```
! This module contains FEAPI precision parameters.
MODULE precision
IMPLICIT NONE
```

```
! Selected double precision.
!
! sdp returns the minimum KIND necessary to store real numbers with a
! precision of 15 decimal digits and an exponent in the range 10^-307 to
! 10^307.
INTEGER, PARAMETER   :: sdp = SELECTED_REAL_KIND(15,307)
END MODULE precision
```

D.1.2 Library – FEA

A. beemat.f90

```
! This sub-routine computes strain-displacement matrix.
SUBROUTINE beemat(deriv,bee)
USE precision
IMPLICIT NONE
! - Arguments.
! Shape function derivatives with respect to global coordinates.
REAL(sdp),  INTENT(IN)   :: deriv(:,:)
! Strain-displacement matrix.
REAL(sdp),  INTENT(OUT) :: bee(:,:)
! :
! :
RETURN
END SUBROUTINE beemat
```

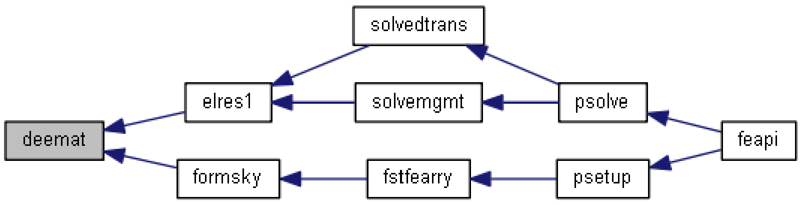


B. deemat.f90

```
! This sub-routine computes stress-strain matrix.
SUBROUTINE deemat(ptype,e,v,dee)
USE precision
IMPLICIT NONE
! - Arguments.
! Modulus of elasticity.
REAL(sdp),      INTENT(IN)      :: e
! Poisson's ratio.
REAL(sdp),      INTENT(IN)      :: v
! Formulation
CHARACTER(*),   INTENT(IN)      :: ptype
! Element stress-strain matrix.
REAL(sdp),      INTENT(INOUT)   :: dee(:,:)
! :
! :
RETURN
END SUBROUTINE deemat
```

CALL

CALLER

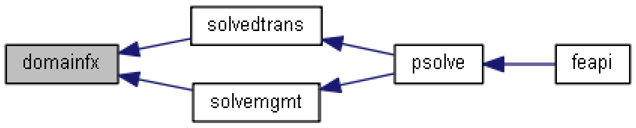


C. domainfx.f90

```
! This sub-routine computes domain external work.
SUBROUTINE domainfx(loads,disp,fx)
USE precision
USE interfaces
IMPLICIT NONE
! - Arguments.
! Load vector.
REAL(sdp),  INTENT(IN)  :: loads(0:)
! Displacement vector.
REAL(sdp),  INTENT(IN)  :: disp(0:)
! External work.
REAL(sdp),  INTENT(OUT) :: fx
! :
! :
RETURN
END SUBROUTINE domainfx
```

CALL

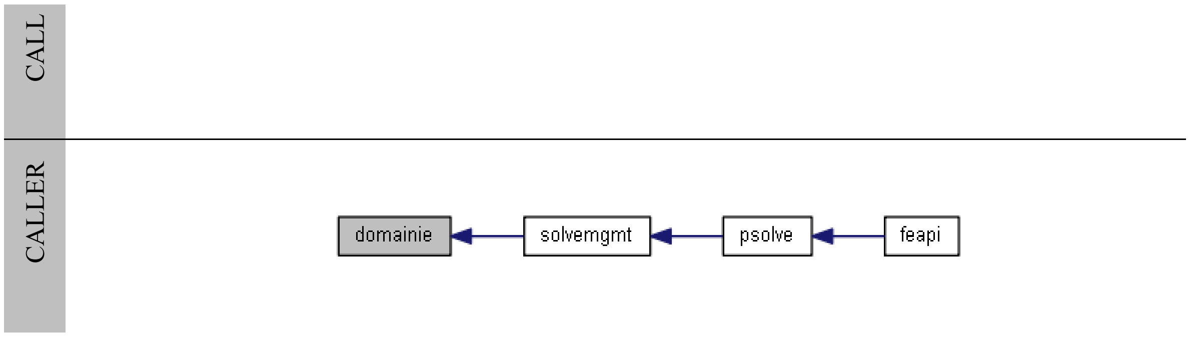
CALLER



D. domainie.f90

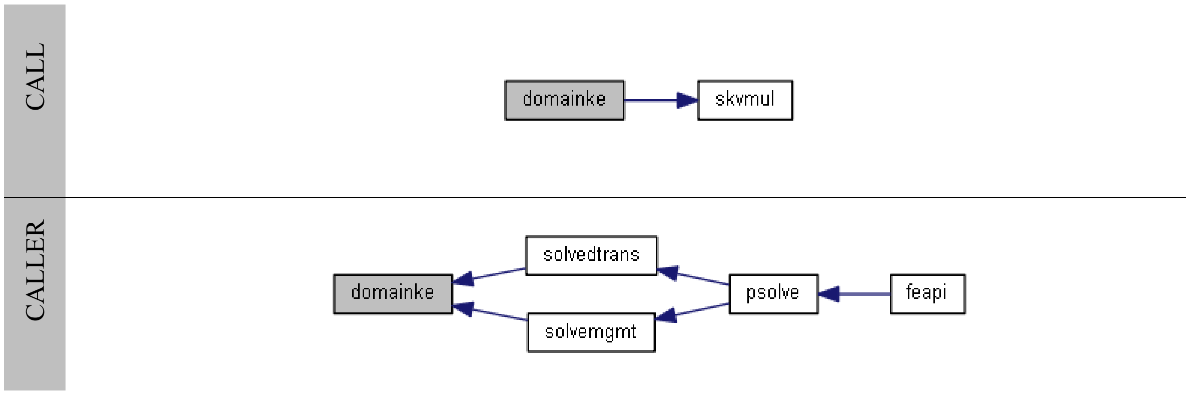
```
! This sub-routine computes domain interface energy.
SUBROUTINE domainie(ipro,ifor,velo,ie)
USE precision
USE interfaces
USE data_mgmt
IMPLICIT NONE
! - Arguments.
! Interface projection matrix.
REAL(sdp),  INTENT(IN)  :: ipro(:,:)
! Interface force.
REAL(sdp),  INTENT(IN)  :: ifor(:)
! Nodal velocities.
REAL(sdp),  INTENT(IN)  :: velo(0:)
```

```
! Interface Energy.
REAL(sdp),   INTENT(OUT) :: ie
! :
! :
RETURN
END SUBROUTINE domainie
```



E. domainke.f90

```
! This sub-routine computes domain kinetic energy.
SUBROUTINE domainke(mv,velo,kdiag,ke)
USE precision
USE interfaces
IMPLICIT NONE
! - Arguments.
! Mass matrix (symmetric/skyline).
REAL(sdp),   INTENT(IN)  :: mv(:)
! Velocity vector.
REAL(sdp),   INTENT(IN)  :: velo(0:)
! Diagonal term locator.
INTEGER,     INTENT(IN)  :: kdiag(:)
! Kinetic Energy.
REAL(sdp),   INTENT(OUT) :: ke
! :
! :
RETURN
END SUBROUTINE domainke
```

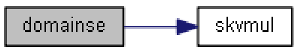


F. domainse.f90

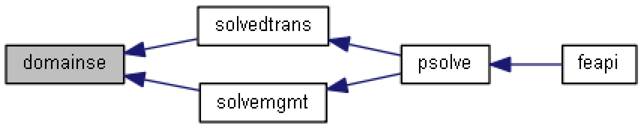
```
! This sub-routine computes domain stiffness energy.
SUBROUTINE domainse(kv,disp,kdiag,se)
```

```
USE precision
USE interfaces
IMPLICIT NONE
! - Arguments.
! Stiffness matrix (symmetric/skyline).
REAL(sdp),   INTENT(IN)  :: kv(:)
! Displacement vector.
REAL(sdp),   INTENT(IN)  :: disp(0:)
! Diagonal term locator.
INTEGER,     INTENT(IN)  :: kdiag(:)
! Stiffness Energy.
REAL(sdp),   INTENT(OUT) :: se
! :
! :
RETURN
END SUBROUTINE domainse
```

CALL



CALLER

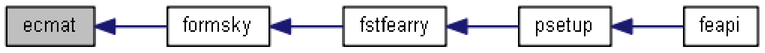


G. ecmat.f90

```
! This sub-routine computes the consistent mass matrix for an element.
SUBROUTINE ecmat(ecm,fun,ndof,nodof)
USE precision
IMPLICIT NONE
! - Arguments.
! Shape functions.
REAL(sdp),   INTENT(IN)  :: fun(:)
! Number of Degrees of freedom per node.
INTEGER,     INTENT(IN)  :: nodof
! Number of Degrees of freedom per element.
INTEGER,     INTENT(IN)  :: ndof
! Element consistent mass matrix.
REAL(sdp),   INTENT(OUT) :: ecm(:,:)
! :
! :
RETURN
END SUBROUTINE ecmat
```

CALL

CALLER

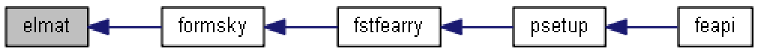


H. elmat.f90

```
! This sub-routine computes the lumped mass matrix for an element.
SUBROUTINE elmat(area,rho,emm)
USE precision
IMPLICIT NONE
! - Arguments.
! Element area.
REAL(sdp),  INTENT(IN)  :: area
! Material mass density.
REAL(sdp),  INTENT(IN)  :: rho
! Lumped element mass matrix.
REAL(sdp),  INTENT(OUT) :: emm(:,:)
! :
! :
RETURN
END SUBROUTINE elmat
```

CALL

CALLER

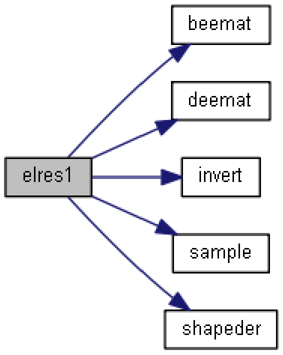


I. elres1.f90

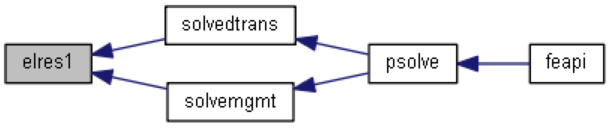
```
! This sub-routine computes element stresses and strains at Gauss
! integration points.
SUBROUTINE elres1(db,disp,stress,strain)
USE precision
USE interfaces
USE data_domain
USE data_material
USE data_structural
IMPLICIT NONE
! - Arguments.
! Domain block.
INTEGER,    INTENT(IN)     :: db
! Nodal displacement vector.
REAL(sdp),  INTENT(IN)     :: disp(0:)
! Element stresses.
REAL(sdp),  INTENT(INOUT)  :: stress(:,:,:)
! Element strains.
REAL(sdp),  INTENT(INOUT)  :: strain(:,:,:)
! :
```

```
! :
RETURN
END SUBROUTINE elres1
```

CALL



CALLER

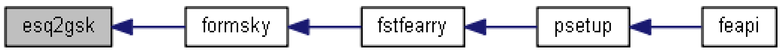


J. esq2gsk.f90

```
! This sub-routine is used in the assembly of symmetric/skyline matrices.
SUBROUTINE esq2gsk(kv,km,g,diagtl)
USE precision
IMPLICIT NONE
! - Arguments.
! Element steering vector.
INTEGER,    INTENT(IN)  :: g(:)
! Diagonal term locator.
INTEGER,    INTENT(IN)  :: diagtl(:)
! Element matrix.
REAL(sdp),  INTENT(IN)  :: km(:,:)
! Global matrix (stored as a skyline vector).
REAL(sdp),  INTENT(OUT) :: kv(:)
! :
! :
RETURN
END SUBROUTINE esq2gsk
```

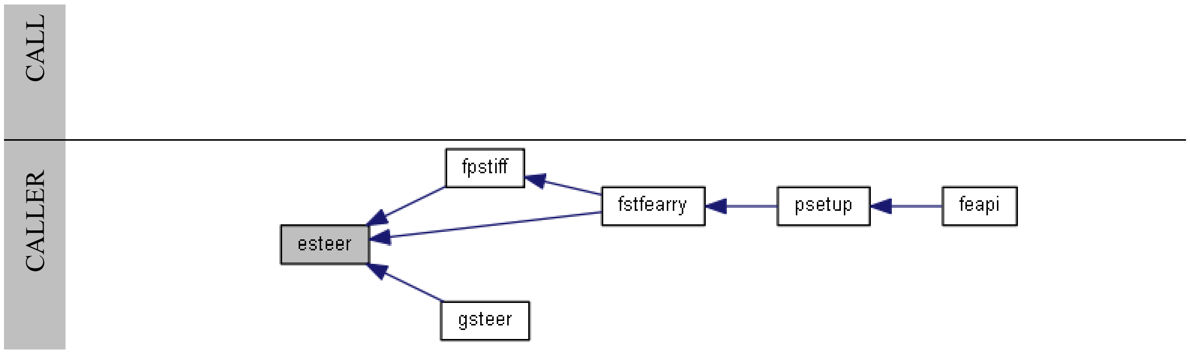
CALL

CALLER



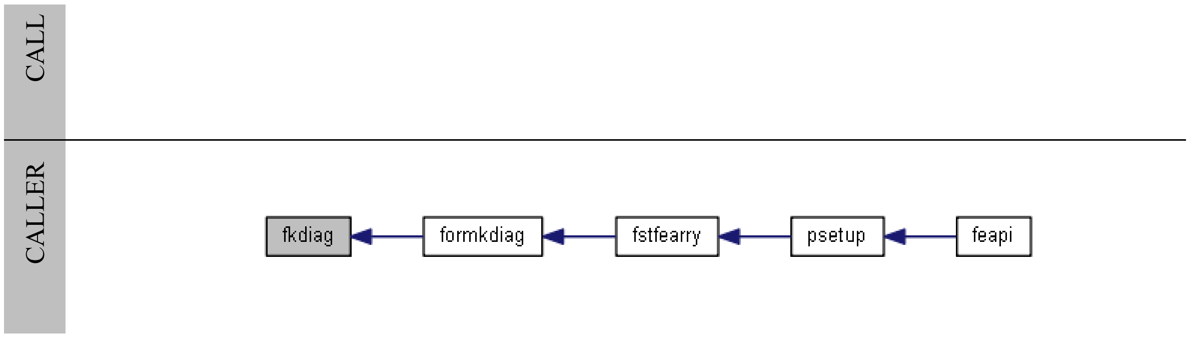
K. esteer.f90

```
! This sub-routine returns the element degree of freedom steering vector from
! element node numbering and nodal degree of freedom array.
SUBROUTINE esteer(num,nf,g)
IMPLICIT NONE
! - Arguments.
! Element node numbers vector.
INTEGER, INTENT(IN)  :: num(:)
! Numbered nodal freedom matrix.
INTEGER, INTENT(IN)  :: nf(:,:)
! Element degree of freedom steering vector.
INTEGER, INTENT(OUT) :: g(:)
! :
! :
RETURN
END SUBROUTINE esteer
```



L. fkdiag.f90

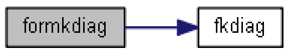
```
! This sub-routine computes the skyline profile for symmetric system
! matrices.
SUBROUTINE fkdiag(diagtl,g)
IMPLICIT NONE
! - Arguments.
! Element degree of freedom steering vector.
INTEGER, INTENT(IN)  :: g(:)
! Skyline profile.
INTEGER, INTENT(OUT) :: diagtl(:)

! :
! :
RETURN
END SUBROUTINE fkdiag
```



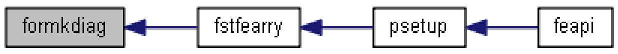
M. formkdiag.f90

```
! This sub-routine computes the array of diagonal term locators for a skyline
! storage system.
SUBROUTINE formkdiag(gg,diagtl)
USE interfaces
IMPLICIT NONE
! - Arguments.
! Element degree of freedom steering array.
INTEGER, INTENT(IN)     :: gg(:,:)
! Skyline profile -> Diagonal term locator.
INTEGER, INTENT(INOUT)  :: diagtl(:)
! :
! :
RETURN
END SUBROUTINE formkdiag
```

CALL



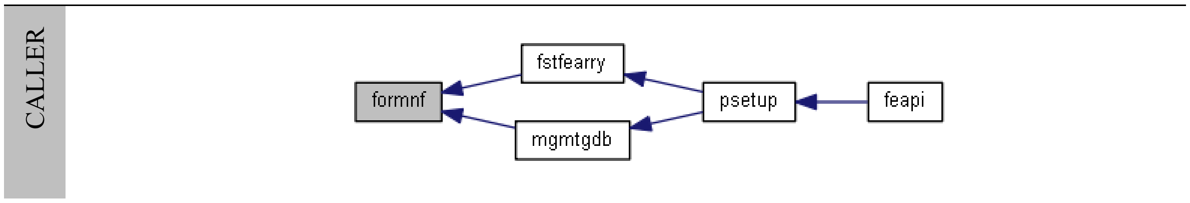
CALLER



N. formnf.f90

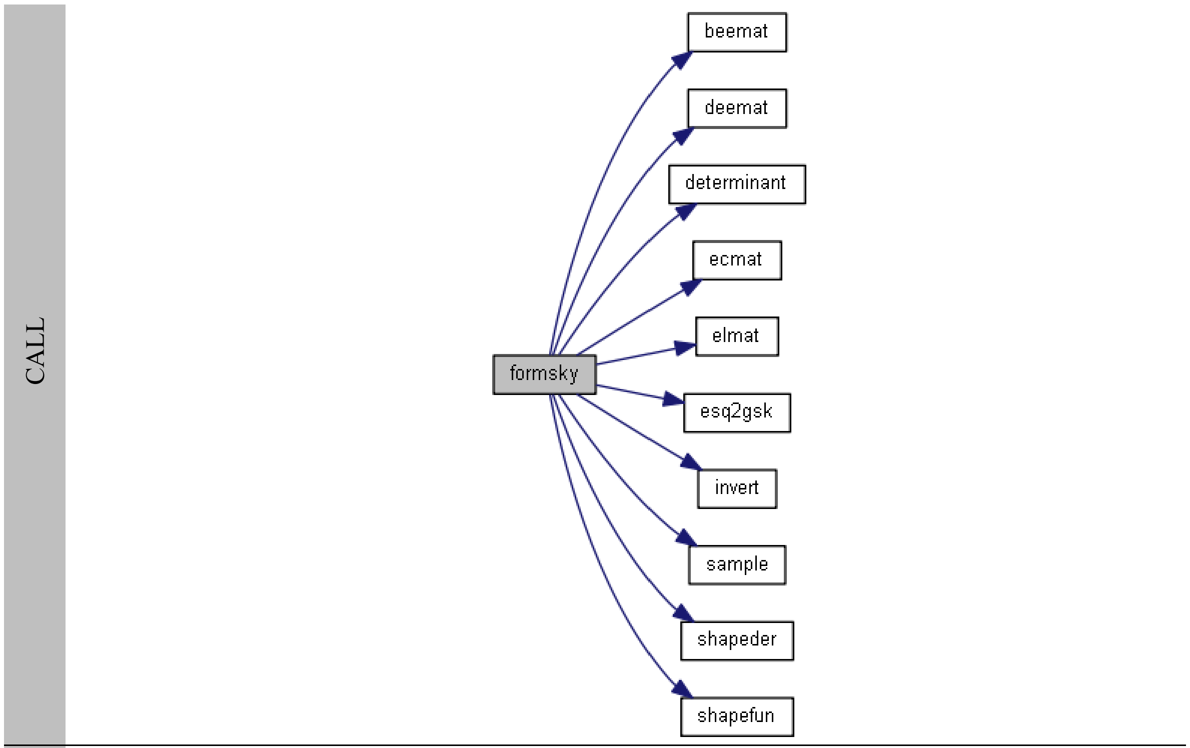
```
! This sub-routine forms the (numbered) nodal degree of freedom array.
SUBROUTINE formnf(lrn,ardof,nf)
IMPLICIT NONE
! - Arguments.
! List of constrained nodes.
INTEGER, INTENT(IN)     :: lrn(:)
!> Array of constrained degrees of freedom.
INTEGER, INTENT(IN)     :: ardof(:,:)
!> Nodal freedom matrix.
INTEGER, INTENT(INOUT)  :: nf(:,:)
! :
! :
RETURN
END SUBROUTINE formnf
```

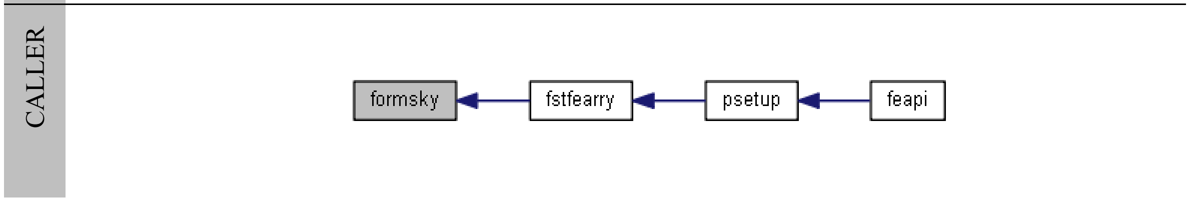
CALL



O. formsky.f90

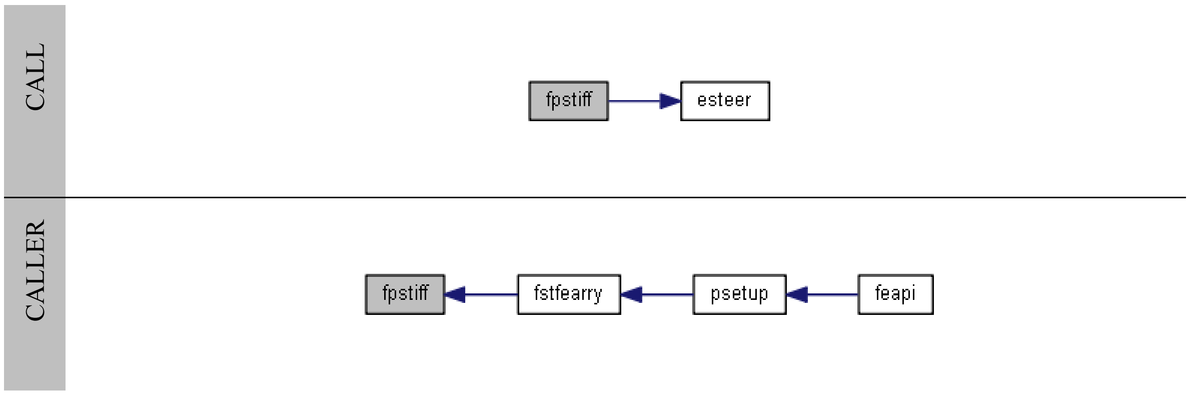
```
! This sub-routine computes and assembles symmetric element matrices
! into global skylines arrays.
SUBROUTINE formsky(db)
USE precision
USE interfaces
USE data_domain
USE data_material
USE data_structural
IMPLICIT NONE
! - Arguments.
! Domain block .
INTEGER,     INTENT(IN)   :: db
! :
! :
RETURN
END SUBROUTINE formsky
```





P. fpstiff.f90

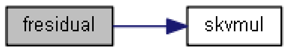
```fortran
! This sub-routine forms a penalty augmented stiffness matrix for including
! prescribed (displacement) degrees of freedom.
SUBROUTINE fpstiff(db)
USE precision
USE interfaces
USE data_domain
USE data_structural
IMPLICIT NONE
! - Arguments.
! Domain block.
INTEGER, INTENT(IN)   :: db
! :
! :
RETURN
END SUBROUTINE fpstiff
```



Q. fresidual.f90

```fortran
! This sub-routine computes the residual for the equilibrium equation.
SUBROUTINE fresidual(dn,rloads,rx0,rx1,resd)
USE precision
USE interfaces
USE data_domain
USE data_structural
USE data_transient
IMPLICIT NONE
! - Arguments.
! Domain number.
INTEGER,     INTENT(IN)      :: dn
! Loads.
REAL(sdp),   INTENT(IN)      :: rloads(:,:)
! Initial solution vector.
REAL(sdp),   INTENT(IN)      :: rx0(:,:)
! Updated solution vector.
REAL(sdp),   INTENT(IN)      :: rx1(:,:)
! Residual.
REAL(sdp),   INTENT(INOUT)   :: resd(:)
! :
! :
RETURN
END SUBROUTINE fresidual
```

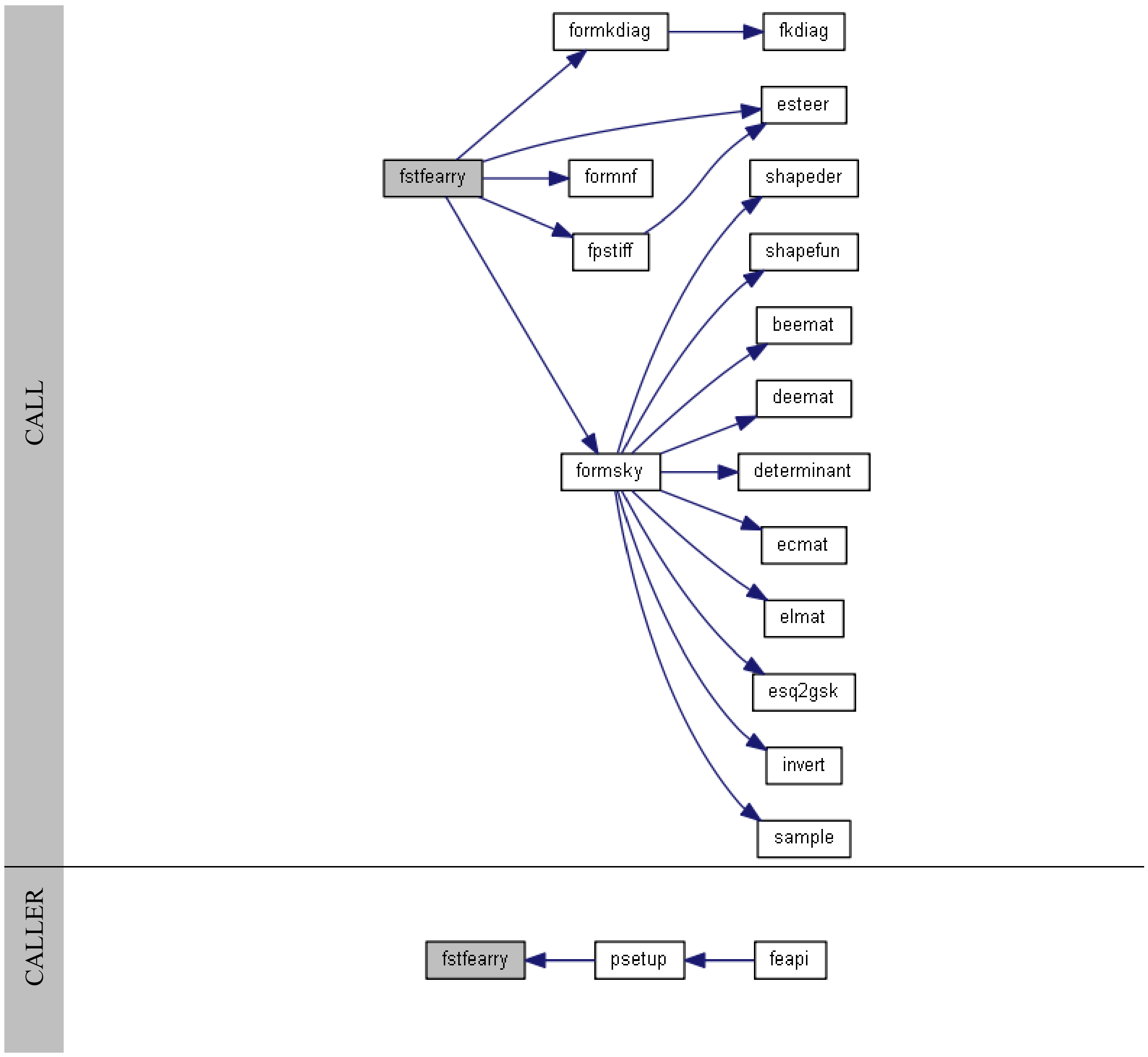
CALL



CALLER

R. fstfearry.f90

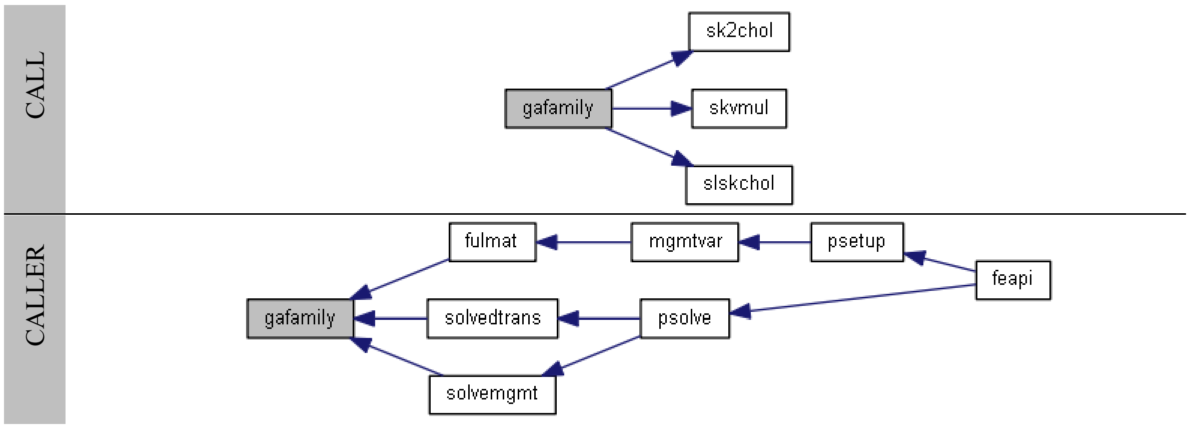
```
! This sub-routine forms structural finite element arrays.
SUBROUTINE fstfearry(db)
USE precision
USE interfaces
USE data_domain
USE data_material
USE data_structural
IMPLICIT NONE
! - Arguments.
! Domain block.
INTEGER, INTENT(IN)  :: db
! :
! :
RETURN
END SUBROUTINE fstfearry
```



S. gafamily.f90

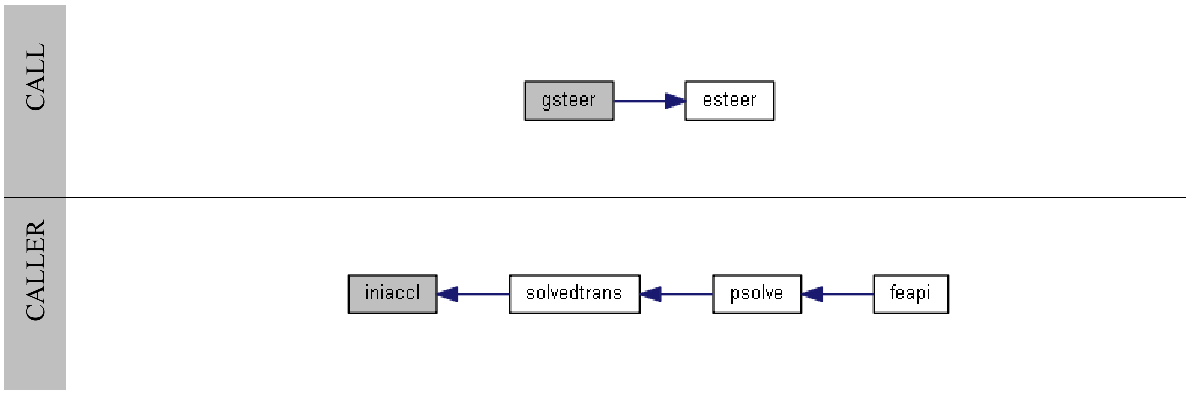
```
! This sub-routine uses the Generalized-alpha family of algorithms for direct
! time integration of (symmetric/skyline) structural dynamic equations.
SUBROUTINE gafamily(af,am,beta,gamma,dt,sm,sc,sk,l0,l1,d0,v0,a0,d1,v1,a1,diagtl)
USE precision
USE interfaces
IMPLICIT NONE
! - Arguments.
! Generalized Alpha parameter.
REAL(sdp),  INTENT(IN)  :: af
! Generalized Alpha parameter.
REAL(sdp),  INTENT(IN)  :: am
! Newmark parameter
REAL(sdp),  INTENT(IN)  :: beta
! Newmark parameter.
REAL(sdp),  INTENT(IN)  :: gamma
! Integration time step.
REAL(sdp),  INTENT(IN)  :: dt
! Structural mass matrix.
```

```
REAL(sdp),   INTENT(IN)   :: sm(:)
! Structural damping matrix.
REAL(sdp),   INTENT(IN)   :: sc(:)
! Structural stiffness matrix.
REAL(sdp),   INTENT(IN)   :: sk(:)
! Loads at the beginning of the time-step.
REAL(sdp),   INTENT(IN)   :: l0(:)
! Loads at the end of the time-step.
REAL(sdp),   INTENT(IN)   :: l1(:)
! Displacements at the beginning of the time-step.
REAL(sdp),   INTENT(IN)   :: d0(:)
! Velocities at the beginning of the time-step.
REAL(sdp),   INTENT(IN)   :: v0(:)
! Accelerations at the beginning of the time-step.
REAL(sdp),   INTENT(IN)   :: a0(:)
! Diagonal term locator
INTEGER,     INTENT(IN)   :: diagtl(:)
! Updated displacements.
REAL(sdp),   INTENT(OUT)  :: d1(:)
! Updated velocities.
REAL(sdp),   INTENT(OUT)  :: v1(:)
! Updated accelerations.
REAL(sdp),   INTENT(OUT)  :: a1(:)
! :
! :
RETURN
END SUBROUTINE gafamily
```



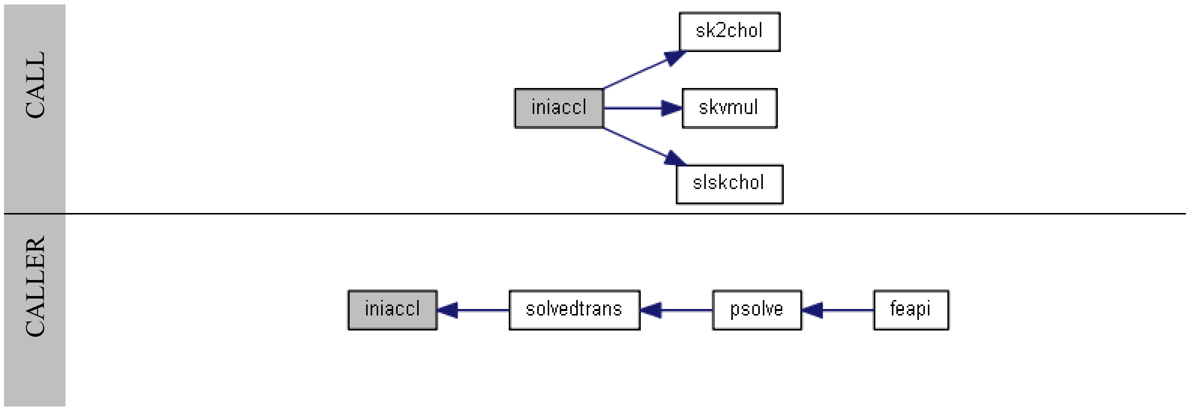
T. gsteer.f90

```
! This sub-routine returns the global element steering matrix.
SUBROUTINE gsteer(connect,nf,g_g)
USE interfaces
IMPLICIT NONE
! - Arguments.
! Nodal connectivities.
INTEGER, INTENT(IN)     :: connect(:,:)
! Nodal freedom matrix.
INTEGER, INTENT(IN)     :: nf(:,:)
! Element degree of freedom steering matrix.
INTEGER, INTENT(INOUT)  :: g_g(:,:)
! :
! :
RETURN
END SUBROUTINE gsteer
```



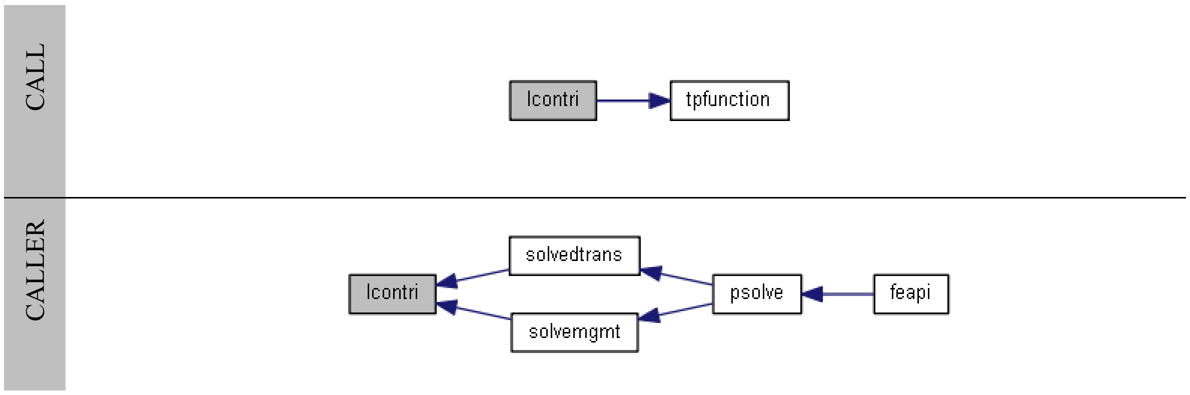
U. iniaccl.f90

```
! This sub-routine computes initial accelerations from given initial conditions.
SUBROUTINE iniaccl(sm,sc,sk,l0,d0,v0,a0,diagtl)
USE interfaces
USE precision
IMPLICIT NONE
! - Arguments.
! Symmetric mass matrix (skyline).
REAL(sdp),   INTENT(IN)   :: sm(:)
! Symmetric damping matrix (skyline).
REAL(sdp),   INTENT(IN)   :: sc(:)
! Symmetric stiffness matrix (skyline).
REAL(sdp),   INTENT(IN)   :: sk(:)
! Initial loads.
REAL(sdp),   INTENT(IN)   :: l0(:)
! Initial displacements.
REAL(sdp),   INTENT(IN)   :: d0(:)
! Initial velocities.
REAL(sdp),   INTENT(IN)   :: v0(:)
! Diagonal term locator.
INTEGER,     INTENT(IN)   :: diagtl(:)
! Computed initial accelerations.
REAL(sdp),   INTENT(OUT) :: a0(:)
! :
! :
RETURN
END SUBROUTINE iniaccl
```



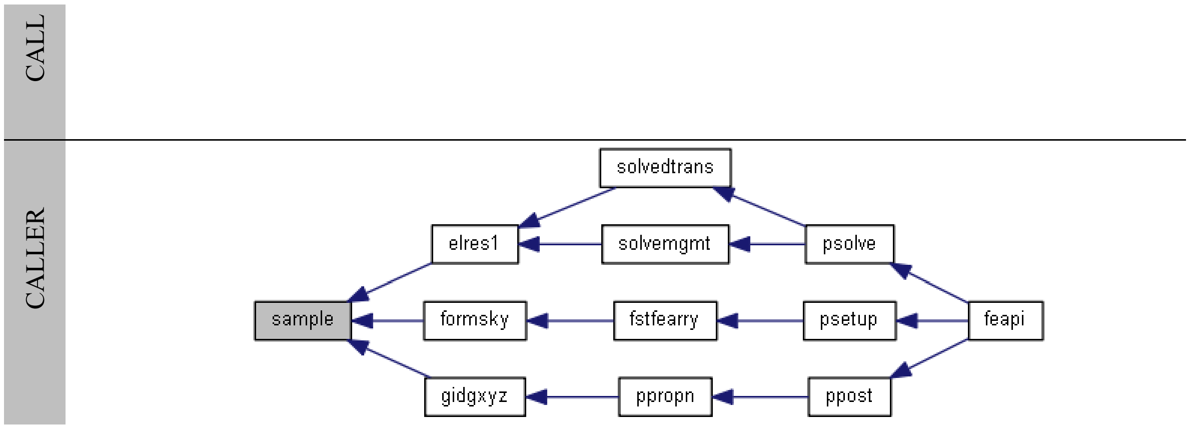
V. lcontri.f90

```
! This sub-routine computes domain (degree of freedom) load contributions.
SUBROUTINE lcontri(db,idf,ctime)
USE precision
USE interfaces
USE data_domain
USE data_transient
USE data_structural
IMPLICIT NONE
! - Arguments.
! Domain block.
INTEGER,     INTENT(IN)   :: db
! Initial load / final load identifier (0/1)
INTEGER,     INTENT(IN)   :: idf
! Current time.
REAL(sdp),   INTENT(IN)   :: ctime
! :
! :
RETURN
END SUBROUTINE lcontri
```



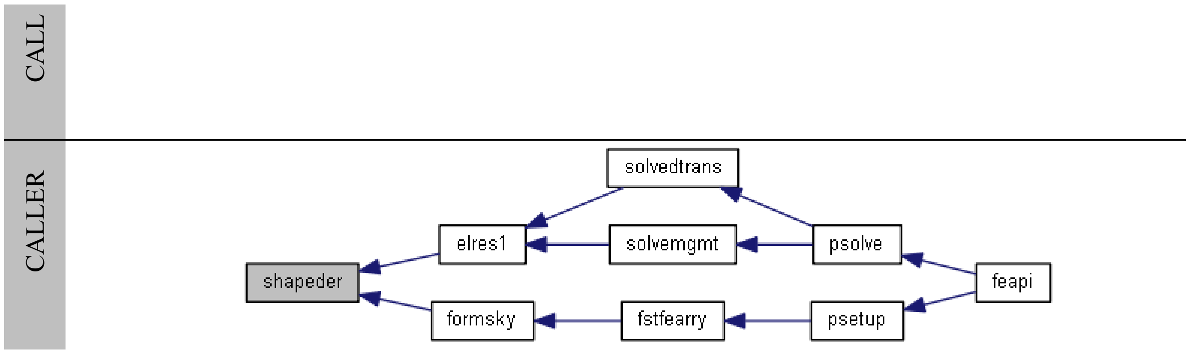
W. sample.f90

```
! This sub-routine returns local coordinates and weighting coefficients for
! the Gauss integration points.
SUBROUTINE sample(etype,s,wt)
USE precision
IMPLICIT NONE
! - Arguments.
! Element type.
CHARACTER(*),  INTENT(IN)              :: etype
! Gauss point coordinates.
REAL(sdp),     INTENT(OUT)             :: s(:,:)
! Gauss point weights.
REAL(sdp),     INTENT(OUT),   OPTIONAL :: wt(:)
! :
! :
RETURN
END SUBROUTINE sample
```



X. shapeder.f90

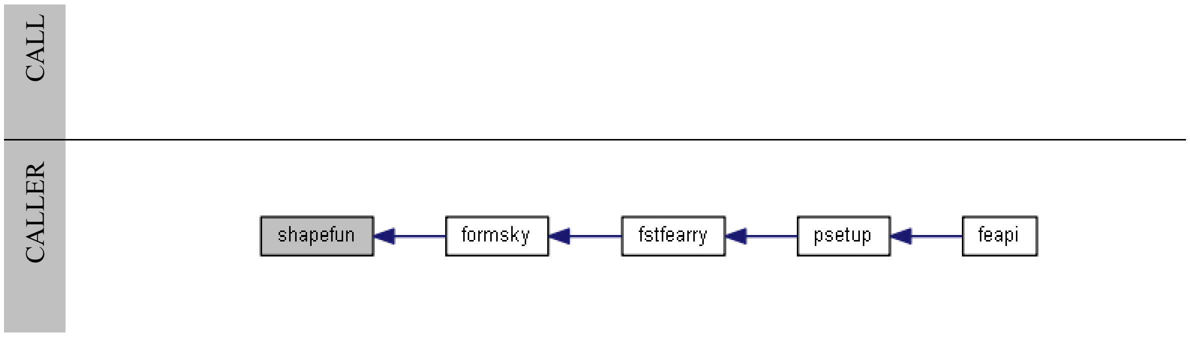
```
! This sub-routine computes the value of shape function derivatives at
! selected Gauss integration points.
SUBROUTINE shapeder(der,points,i)
USE precision
IMPLICIT NONE
! - Arguments.
! Shape function derivatives with respect to local coordinates
REAL(sdp),  INTENT(OUT) :: der(:,:)
! Integration point local coordinates.
REAL(sdp),  INTENT(IN)  :: points(:,:)
! Selected integration point.
INTEGER,    INTENT(IN)  :: i
! :
! :
RETURN
END SUBROUTINE shapeder
```



Y. shapefun.f90

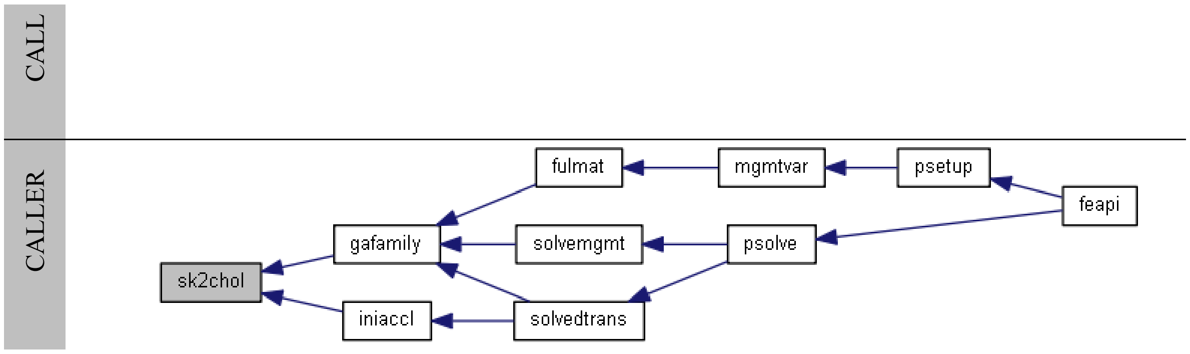
```
! This sub-routine computes the value of shape functions at
! selected Gauss integration points.
SUBROUTINE shapefun(fun,points,i)
USE precision
IMPLICIT NONE
! - Arguments.
! Shape functions.
REAL(sdp),  INTENT(OUT) :: fun(:)
! Integration point local coordinates.
```

```
REAL(sdp),  INTENT(IN)  :: points(:,:)
! Selected integration point.
INTEGER,    INTENT(IN)  :: i
! :
! :
RETURN
END SUBROUTINE shapefun
```



Z. sk2chol.f90

```
! This sub-routine performs Cholesky factorization on a symmetric matrix
! stored as a skyline vector.
SUBROUTINE sk2chol(kv,diagtl)
USE precision
IMPLICIT NONE
! - Arguments.
! Diagonal term locator.
INTEGER,    INTENT(IN)     :: diagtl(:)
! Global matrix -> Cholesky factorized.
REAL(sdp),  INTENT(INOUT)  :: kv(:)
! :
! :
RETURN
END SUBROUTINE sk2chol
```



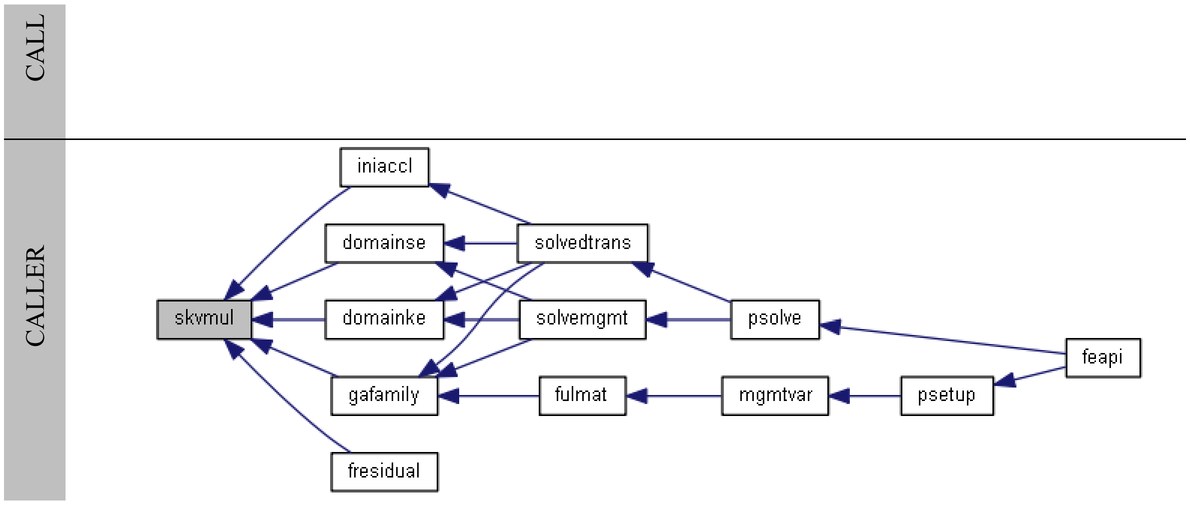
AA. sk2gaus.f90

```
! This sub-routine performs Gauss factorization of a symmetric
! matrix stored as a skyline vector.
SUBROUTINE sk2gaus(kv,diagtl)
USE precision
IMPLICIT NONE
! - Arguments.
! Diagonal term locator.
```

```
INTEGER,     INTENT(IN)  :: diagtl(:)
! Global matrix -> Gauss factorized.
REAL(sdp),   INTENT(OUT) :: kv(:)
! :
! :
RETURN
END SUBROUTINE sk2gaus
```

BB. skvmul.f90

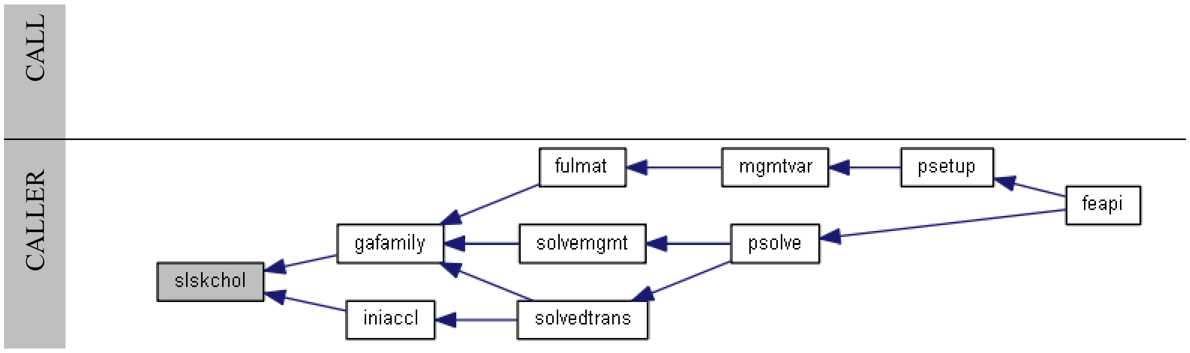
```
! This sub-routine performs matrix-vector multiplication on a
! symmetric matrix stored as a skyline vector.
SUBROUTINE skvmul(kv,disps,loads,diagtl)
USE precision
IMPLICIT NONE
! - Arguments.
! Global coefficient matrix stored as a skyline.
REAL(sdp),   INTENT(IN)  :: kv(:)
! Multiplying vector.
REAL(sdp),   INTENT(IN)  :: disps(0:)
! Diagonal term locator.
INTEGER,     INTENT(IN)  :: diagtl(:)
! Resulting matrix-vector multiplication vector.
REAL(sdp),   INTENT(OUT) :: loads(0:)
! :
! :
RETURN
END SUBROUTINE skvmul
```



CC. slskchol.f90

```
! This sub-routine performs Cholesky forward and backward substitution on a
! symmetric matrix stored as a skyline vector.
SUBROUTINE slskchol(kv,loads,diagtl)
USE precision
IMPLICIT NONE
! - Arguments.
! Global matrix stored as a skyline.
REAL(sdp),   INTENT(IN)     :: kv(:)
! Diagonal term locator.
INTEGER,     INTENT(IN)     :: diagtl(:)
```

```
! RHS vector -> Solution vector.
REAL(sdp),   INTENT(INOUT)   :: loads(0:)
! :
! :
RETURN
END SUBROUTINE slskchol
```

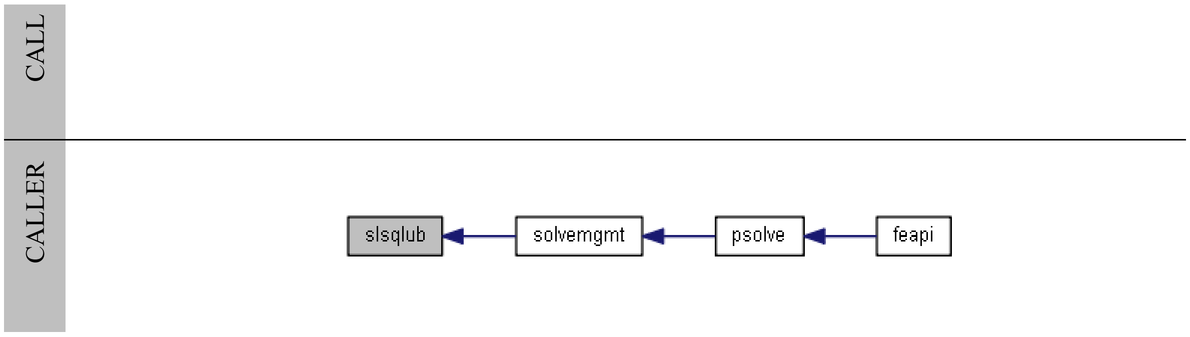


DD. slskgaus.f90

```
! This sub-routine performs Gauss forward and backward substitution on a
! symmetric matrix stored as a skyline vector.
SUBROUTINE slskgaus(kv,loads,diagtl)
USE precision
IMPLICIT NONE
! - Arguments.
! Global matrix stored as a skyline.
REAL(sdp),   INTENT(IN)      :: kv(:)
! Diagonal term locator.
INTEGER,     INTENT(IN)      :: diagtl(:)
! RHS vector -> Solution vector.
REAL(sdp),   INTENT(INOUT)   :: loads(0:)
! :
! :
RETURN
END SUBROUTINE slskgaus
```

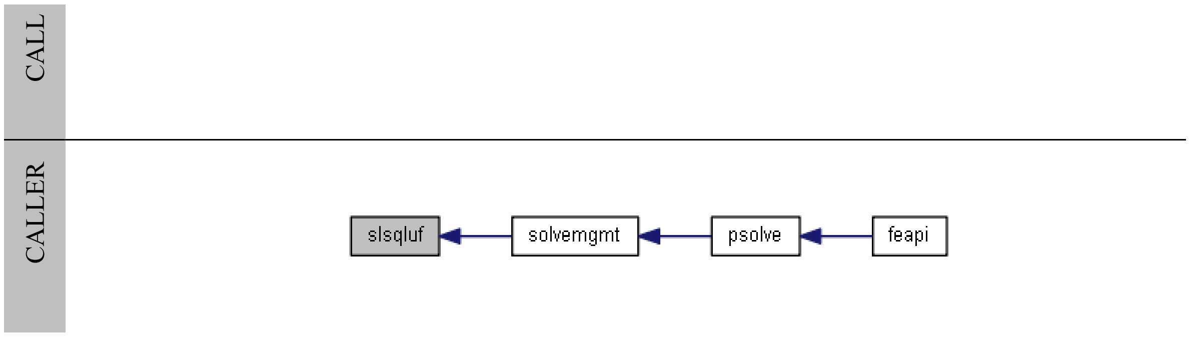
EE. slsqlub.f90

```
! This sub-routine performs backward substitution on an upper triangular
! square matrix obtained after LU factorization.
SUBROUTINE slsqlub(a,b)
USE precision
IMPLICIT NONE
! - Arguments.
! LU factorized matrix.
REAL(sdp),   INTENT(IN)      :: a(:,:)
! RHS vector -> Solution vector.
REAL(sdp),   INTENT(INOUT)   :: b(:)
! :
! :
RETURN
END SUBROUTINE slsqlub
```



FF. slsqluf.f90

```
! This sub-routine performs forward substitution on a lower triangular square
! matrix obtained after LU factorization.

SUBROUTINE slsqluf(a,b)
USE precision
IMPLICIT NONE
! - Arguments.
! LU factorized matrix.
REAL(sdp),  INTENT(IN)     :: a(:,:)
! RHS vector -> Solution vector.
REAL(sdp),  INTENT(INOUT)  :: b(:)
! :
! :
RETURN
END SUBROUTINE slsqluf
```



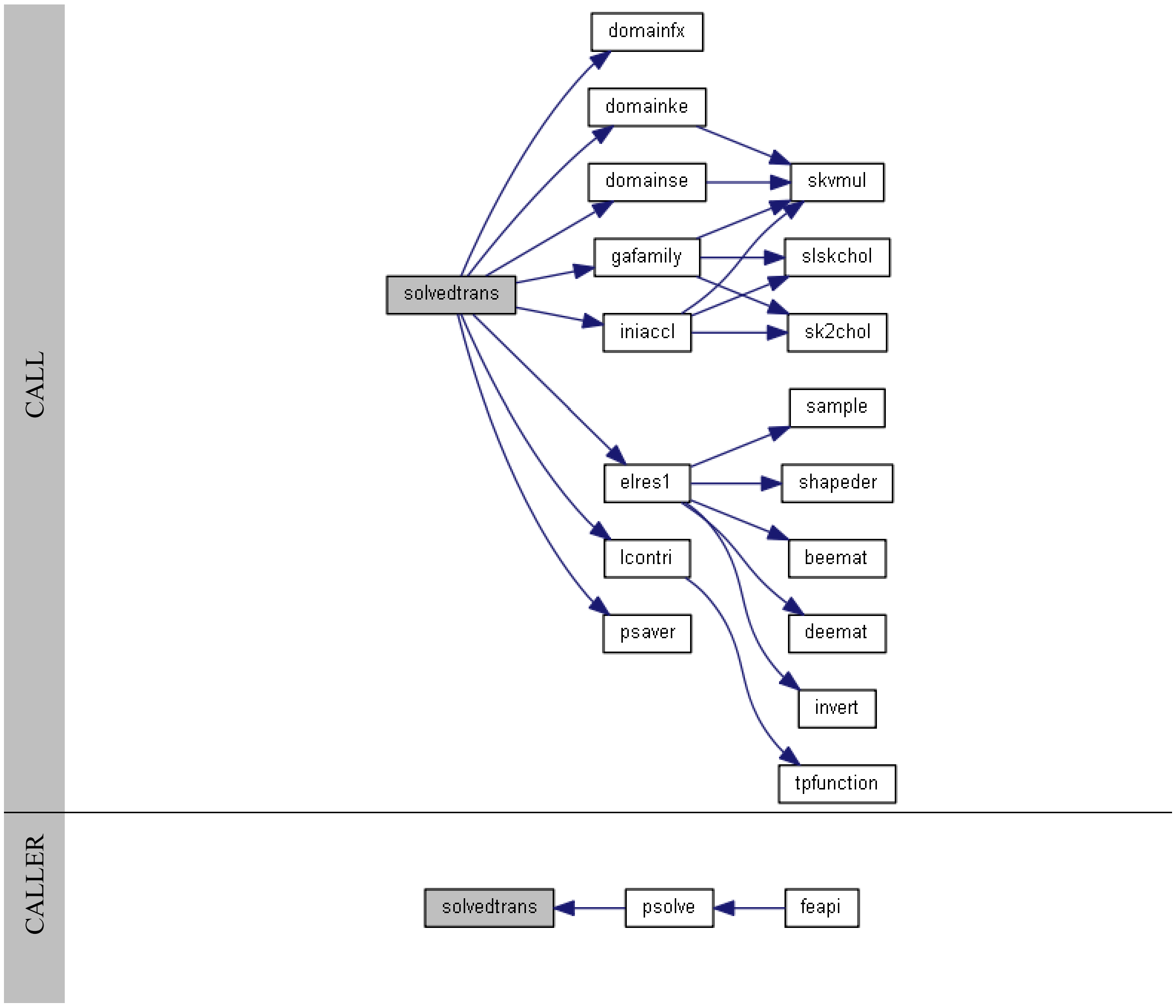
GG. slsqlup.f90

```
! This sub-routine performs forward and backward substitution following
! LU factorization with pivoting.
SUBROUTINE slsqlup(a,b,sol,row)
Use precision
IMPLICIT NONE
! - Arguments.
! LU factorized matrix.
REAL(sdp),  INTENT(IN)  :: a(:,:)
! RHS vector.
REAL(sdp),  INTENT(IN)  :: b(:)
! Solution vector.
REAL(sdp),  INTENT(OUT) :: sol(:)
! Pivot mapping.
INTEGER,    INTENT(IN)  :: row(:)
```

```
! :
! :
RETURN
END SUBROUTINE slsqlup
```

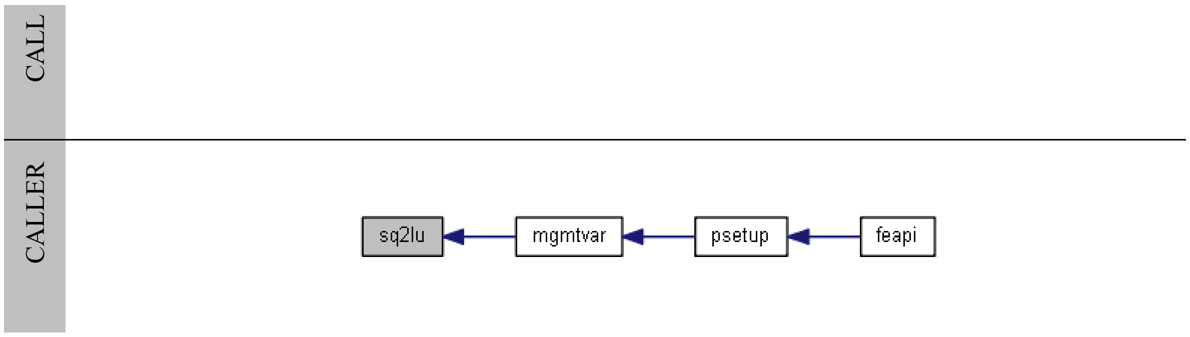
HH. solvedtrans.f90

```
! This sub-routine solves transient structural dynamic
! equations using direct integration.
SUBROUTINE solvedtrans()
USE precision
USE interfaces
USE data_feapi
USE data_transient
USE data_structural
USE data_post
IMPLICIT NONE
! :
! :
RETURN
END SUBROUTINE solvedtrans
```



II. sq2lu.f90

```
! This sub-routine performs LU factorization on a square matrix returning
! lower triangular and upper triangular square matrices.
SUBROUTINE sq2lu(a,lower,upper)
USE precision
IMPLICIT NONE
! - Arguments.
! Input square matrix.
REAL(sdp),  INTENT(IN)  :: a(:,:)
! Factorized lower triangular matrix.
REAL(sdp),  INTENT(OUT) :: lower(:,:)
! Factorized upper triangular matrix.
REAL(sdp),  INTENT(OUT) :: upper(:,:)
! :
! :
RETURN
END SUBROUTINE sq2lu
```



JJ. sq2lup.f90

```
! This sub-routine performs LU factorization on a square matrix with
! pivoting.
SUBROUTINE sq2lup(a,row,error)
USE precision
IMPLICIT NONE
! - Arguments.
! Input square matrix -> LU factorized matrix.
REAL(sdp),  INTENT(INOUT)  :: a(:,:)
! Pivot mapping.
INTEGER,    INTENT(OUT)    :: row(:)
! Error flag.
LOGICAL,    INTENT(OUT)    :: error
! :
! :
RETURN
END SUBROUTINE sq2lup
```

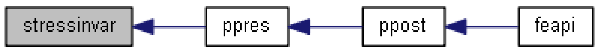
KK. stressinvar.f90

```
! This sub-routine computes the stress invariants.
SUBROUTINE stressinvar(stress,sigma_m,sigma_eq)
USE precision
IMPLICIT NONE
! - Arguments.
! Cauchy stress components.
REAL(sdp),  INTENT(IN)               :: stress(:)
```

```
! Mean stress invariant (Hydrostatic stress).
REAL(sdp),  INTENT(OUT),   OPTIONAL :: sigma_m
! Equivalent stress (Von Mises effective stress).
REAL(sdp),  INTENT(OUT),   OPTIONAL :: sigma_eq
! :
! :
RETURN
END SUBROUTINE stressinvar
```

CALL

CALLER

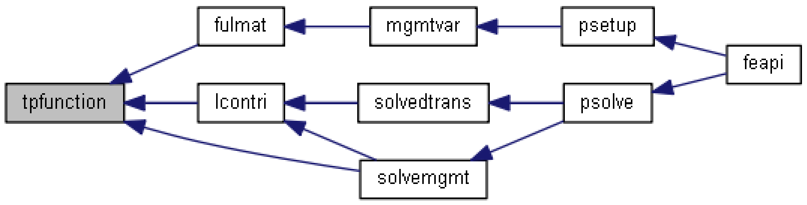


LL. pfunction.f90

```
! This sub-routine returns the coefficient for time proportional functions.
SUBROUTINE tpfunction(ctime,func,duration,wavelength,coefficient)
USE precision
IMPLICIT NONE
! - Arguments.
! Current time.
REAL(sdp),      INTENT(IN)  :: ctime
! Time proportional function.
CHARACTER(*),   INTENT(IN)  :: func
! Function duration.
REAL(sdp),      INTENT(IN)  :: duration
! Function wavelength.
REAL(sdp),      INTENT(IN)  :: wavelength
! Time proportional coefficient \f$ \in \f$ [0,1].
REAL(sdp),      INTENT(OUT) :: coefficient
! :
! :
RETURN
END SUBROUTINE tpfunction
```

CALL

CALLER



D.1.3 Library – MATH

A. BLAS

Basic Linear Algebra Subprograms (BLAS) are a set of low-level kernel subroutines that perform common linear algebra operations such as copying, vector scaling, vector dot products, linear combinations, and matrix multiplication. A quick reference guide to these sub-routines can be found here: http://www.netlib.org/lapack/lug/node145.html.

B. crossproduct.f90

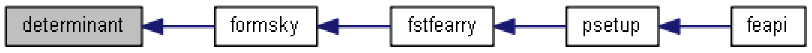
```fortran
! This sub-routine forms the cross product of two REAL vectors, a = b x c.
SUBROUTINE crossproduct(b,c,a)
USE precision
IMPLICIT NONE
! - Arguments.
! Real vector 1.
REAL(sdp),  INTENT(IN)  :: b(:)
! Real vector 2.
REAL(sdp),  INTENT(IN)  :: c(:)
! Cross product of b and c.
REAL(sdp),  INTENT(OUT) :: a(:,:)
! :
! :
RETURN
END SUBROUTINE crossproduct
```

C. determinant.f90

```fortran
! This function returns the determinant of a 1x1, 2x2 or 3x3 matrix.
FUNCTION determinant(jac) RESULT(det)
USE precision
IMPLICIT NONE
! - Arguments.
! Jacobian matrix.
REAL(sdp),  INTENT   (IN)  :: jac(:,:)
REAL(sdp)                  :: det
! :
! :
RETURN
END FUNCTION determinant
```

CALL

CALLER

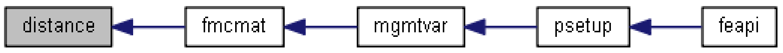


D. distance.f90

```
! This sub-routine returns the distance between two points a and b with
! respect to global coordinates.
SUBROUTINE distance(a,b,ab)
USE precision
IMPLICIT NONE
! - Arguments.
! Point 1.
REAL(sdp),  INTENT(IN)  :: a(:)
! Point 2.
REAL(sdp),  INTENT(IN)  :: b(:)
! Distance between a and b.
REAL(sdp),  INTENT(OUT) :: ab
! :
! :
RETURN
END SUBROUTINE distance
```

CALL

CALLER



E. identity.f90

```
! This sub-routine returns a REAL identity matrix.
SUBROUTINE identity(n,a)
USE precision
IMPLICIT NONE
! - Arguments.
! Matrix dimension.
INTEGER,    INTENT(IN)                 :: n
! Identity matrix.
REAL(sdp),  INTENT(OUT),   ALLOCATABLE :: a(:,:)
! :
! :
RETURN
END SUBROUTINE identity
```

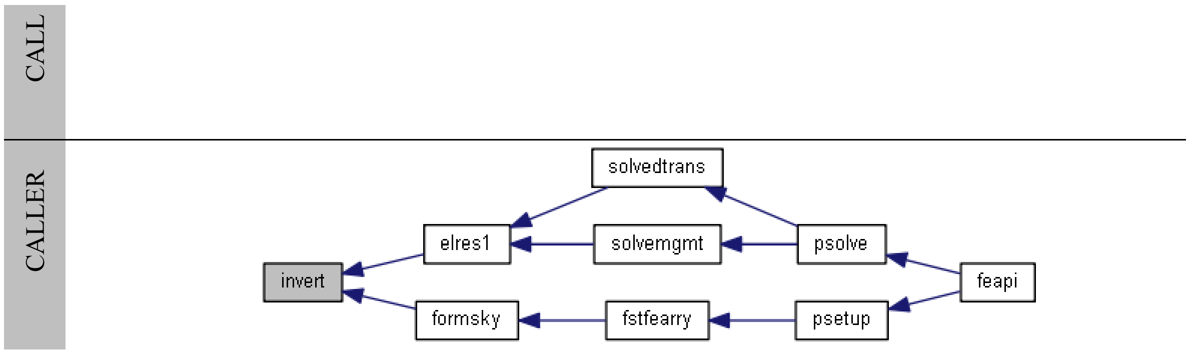
F. inversem.f90

```
! This sub-routine computes the inverse of a (nxn) square matrix.
```

```
SUBROUTINE inversem(matrix,inverse,n)
USE precision
IMPLICIT NONE
! - Arguments.
! Input matrix size.
INTEGER,     INTENT(IN)                       :: n
! Input matrix.
REAL(sdp),   INTENT(IN),     DIMENSION(n,n)  :: matrix
! Inverse matrix.
REAL(sdp),   INTENT(OUT),    DIMENSION(n,n)  :: inverse
! :
! :
RETURN
END SUBROUTINE inversem
```

G. invert.f90

```
! This sub-routine computes the inverse of a small square matrix.
SUBROUTINE invert(matrix)
USE precision
IMPLICIT NONE
! - Arguments.
! Matrix -> Inverse
REAL(sdp),   INTENT(INOUT)   :: matrix(:,:)
! :
! :
RETURN
END SUBROUTINE invert
```



H. l2norm.f90

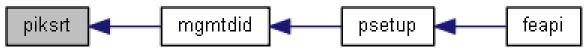
```
! This function returns the L2 norm of a vector.
FUNCTION l2norm(vec) RESULT(norm)
USE precision
IMPLICIT NONE
! - Arguments.
! Vector.
REAL(sdp),   INTENT(IN)   :: vec(:)
REAL(sdp)                 :: norm
! :
! :
RETURN
END FUNCTION l2norm
```

I. piksrt.f90

```
! This sub-routine sorts a REAL array in descending order by straight
! insertion.
SUBROUTINE piksrt(arr)
USE precision
IMPLICIT NONE
! - Arguments.
! Array to be sorted -> Sorted array.
REAL(sdp), INTENT(INOUT)   :: arr(:)
! :
! :
RETURN
END SUBROUTINE piksrt
```

CALL

CALLER



J. scalarproduct.f90

```
! This sub-routine forms the scalar product of two REAL vectors, a = b . c
SUBROUTINE scalarproduct(b,c,a)
USE precision
IMPLICIT NONE
! - Arguments.
! Vector 1.
REAL(sdp),  INTENT(IN)  :: b(:)
! Vector 2.
REAL(sdp),  INTENT(IN)  :: c(:)
! Scalar product of b and c.
REAL(sdp),  INTENT(OUT) :: a
! :
! :
RETURN
END SUBROUTINE scalarproduct
```

D.1.4 Library – Program

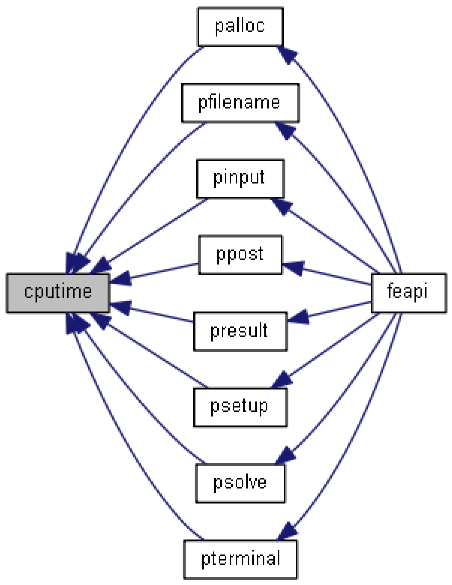
A. cputime.f90

```
! This function returns the current CPU time in seconds.
FUNCTION cputime()
USE precision
IMPLICIT NONE
REAL(sdp)   :: cputime  ! Return value.
REAL(sdp)   :: time     ! Time.
! :
! :
RETURN
```

```fortran
END FUNCTION cputime
```

CALL

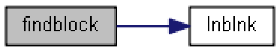
CALLER

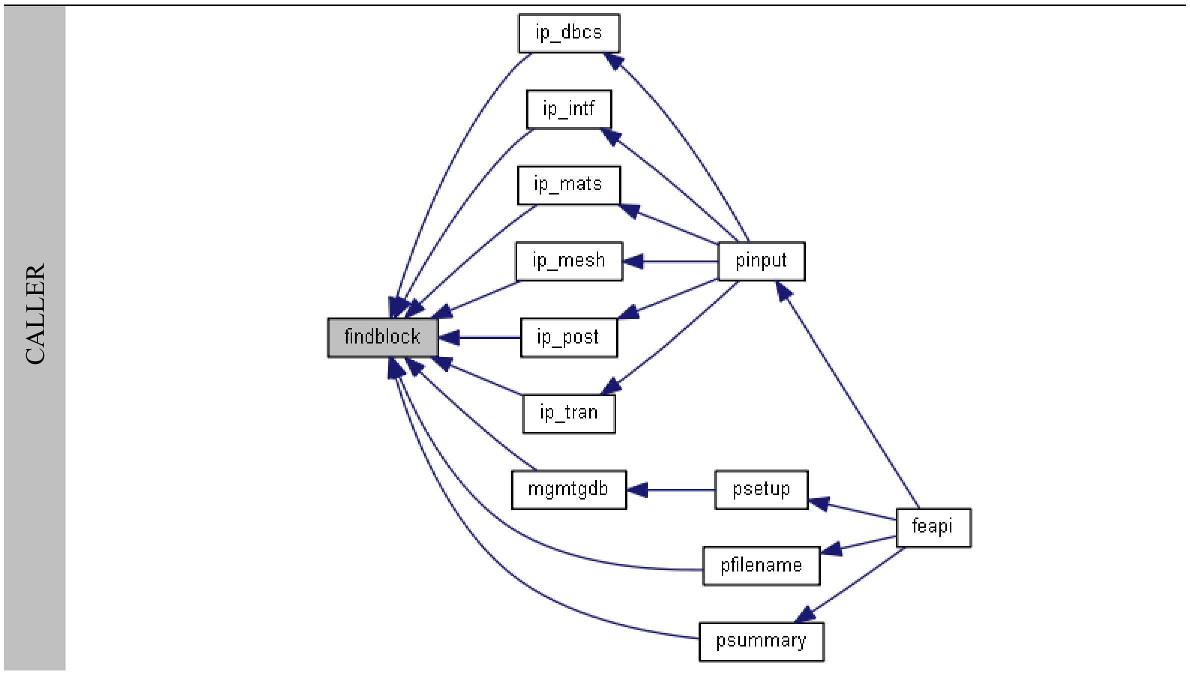


B. findblock.f90

```fortran
! This sub-routine is used to find data blocks within FEAPI program files.
SUBROUTINE findblock(uid,path,name,search)
USE interfaces
IMPLICIT NONE
! - Arguments
! File unit identifier.
INTEGER,        INTENT(IN)   :: uid
! File location.
CHARACTER(*),   INTENT(IN)   :: path
! File name with extension.
CHARACTER(*),   INTENT(IN)   :: name
! 'Search' character string.
CHARACTER(*),   INTENT(IN)   :: search
! :
! :
RETURN
END SUBROUTINE findblock
```

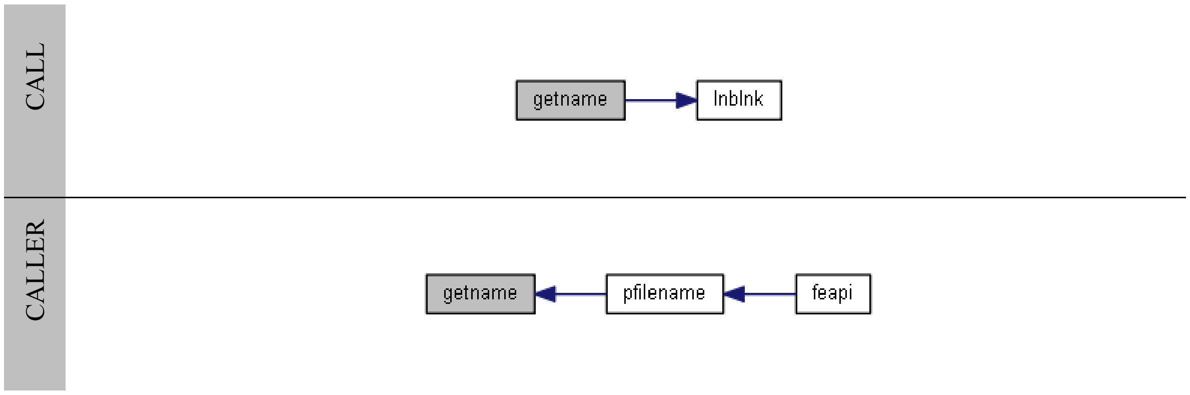
CALL





C. getname.f90

```
! This sub-routine prompts the input of terminal commands.
SUBROUTINE getname(argv,nlen)
USE interfaces
IMPLICIT NONE
! - Arguments.
! Terminal argument.
CHARACTER(*),  INTENT(OUT) :: argv
! Argument character length.
INTEGER,       INTENT(OUT) :: nlen
! :
! :
RETURN
END SUBROUTINE getname
```



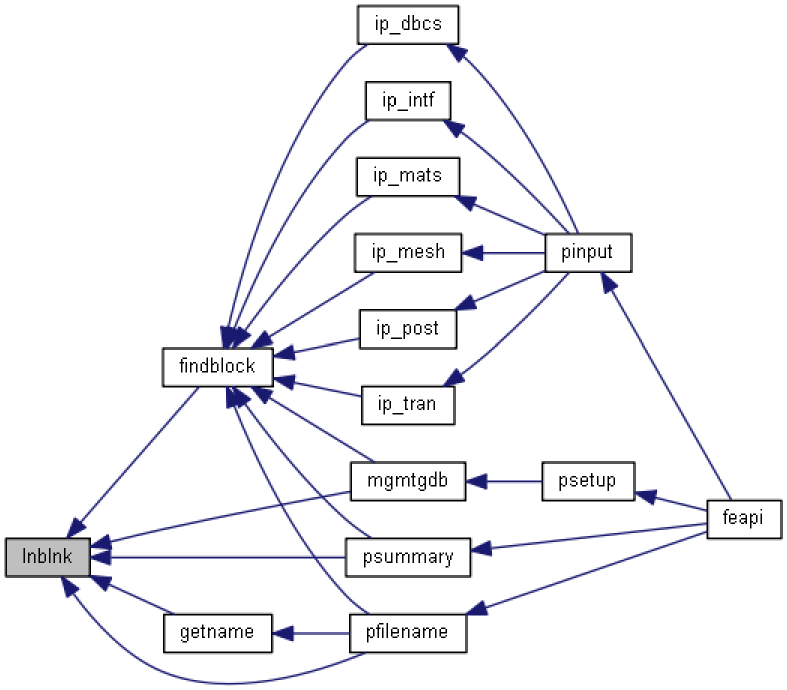
D. lnblnk.f90

```
! This function computes the character length for an input argument.
FUNCTION lnblnk(string) RESULT(alen)
```

```
IMPLICIT NONE
! - Arguments.
! :
! :
RETURN
END FUNCTION lnblnk
```

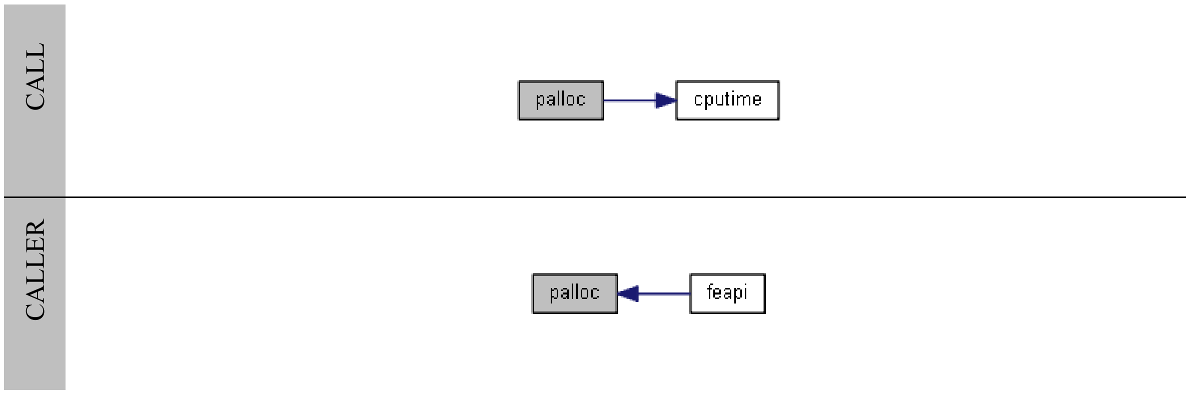
CALL

CALLER



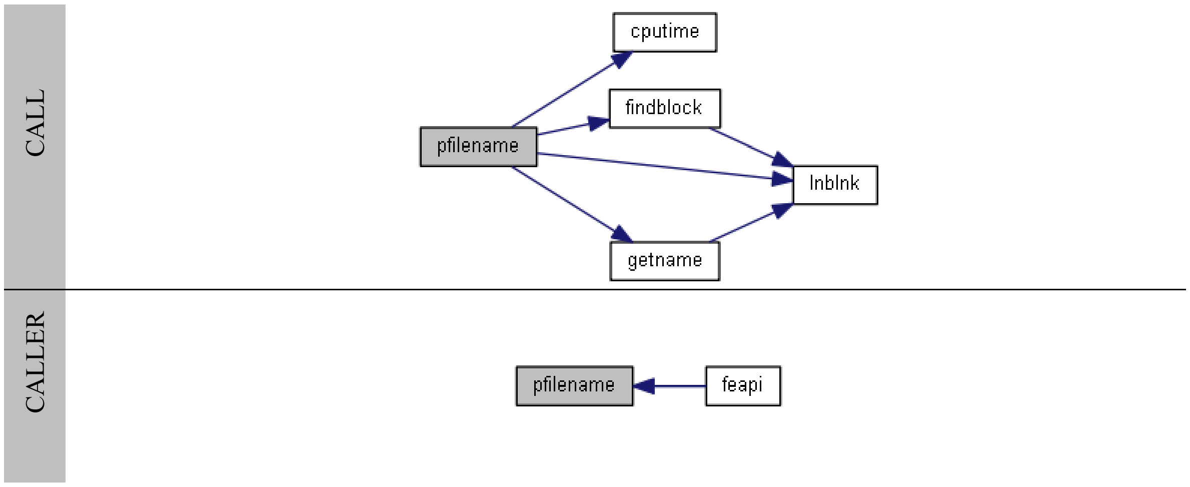
E. palloc.f90

```
! This sub-routine allocates FEAPI TYPE variables.
SUBROUTINE palloc()
USE interfaces
USE data_feapi
USE data_domain
USE data_material
USE data_transient
USE data_structural
USE data_post
IMPLICIT NONE
! - Local variables.
! :
! :
RETURN
END SUBROUTINE palloc
```



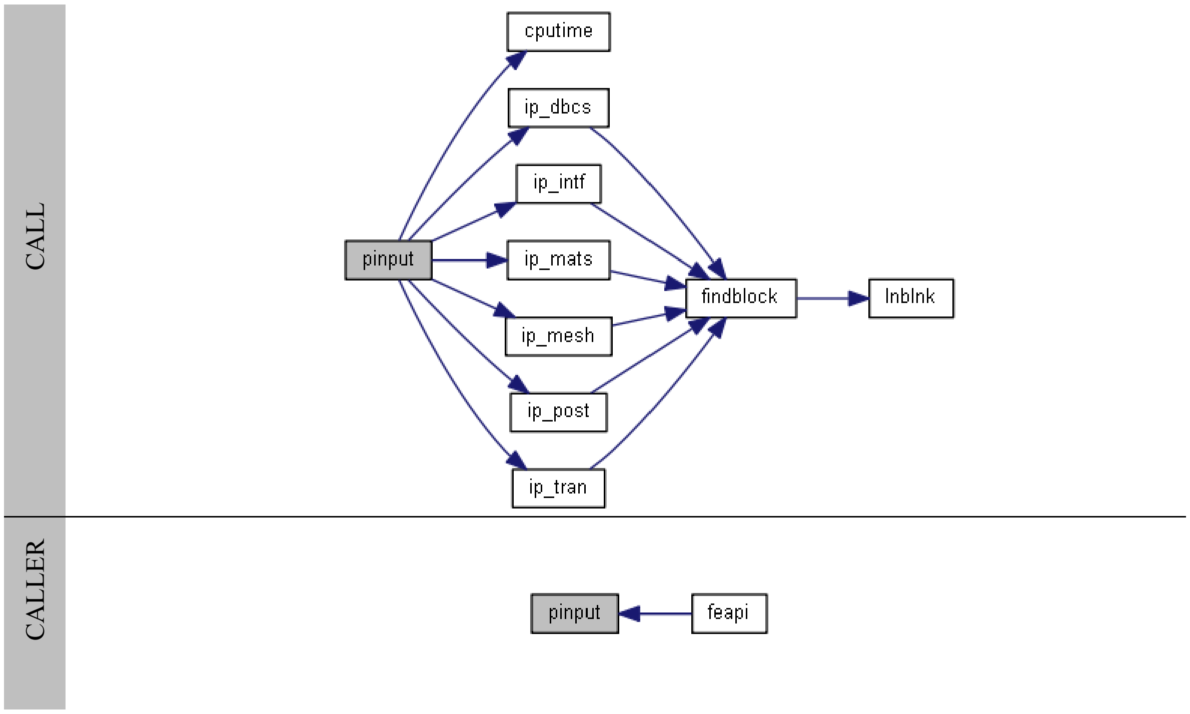
F. pfilename.f90

```
! This sub-routine allocates and assigns FEAPI program files.
SUBROUTINE pfilename()
USE interfaces
USE data_feapi
IMPLICIT NONE
! - Local variables.
! :
! :
RETURN
END SUBROUTINE pfilename
```



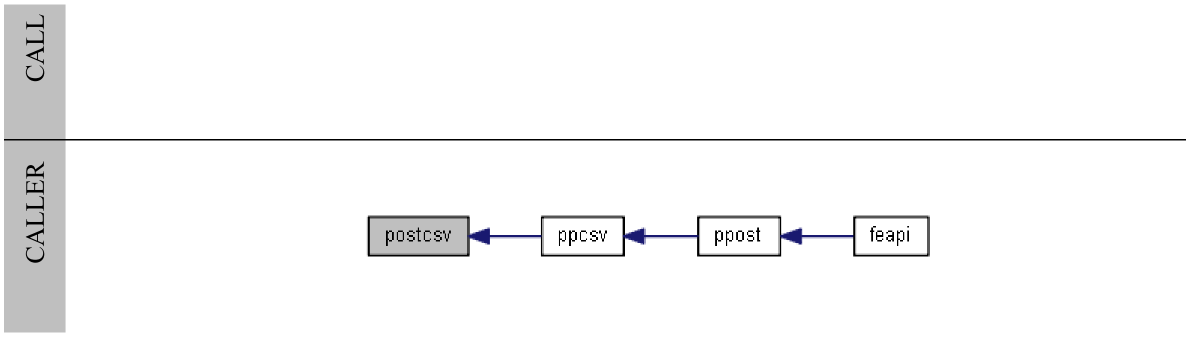
G. pinput.f90

```
! This sub-routine reads FEAPI input data.
SUBROUTINE pinput()
USE interfaces
USE data_feapi
IMPLICIT NONE
! :
! :
RETURN
END SUBROUTINE pinput
```



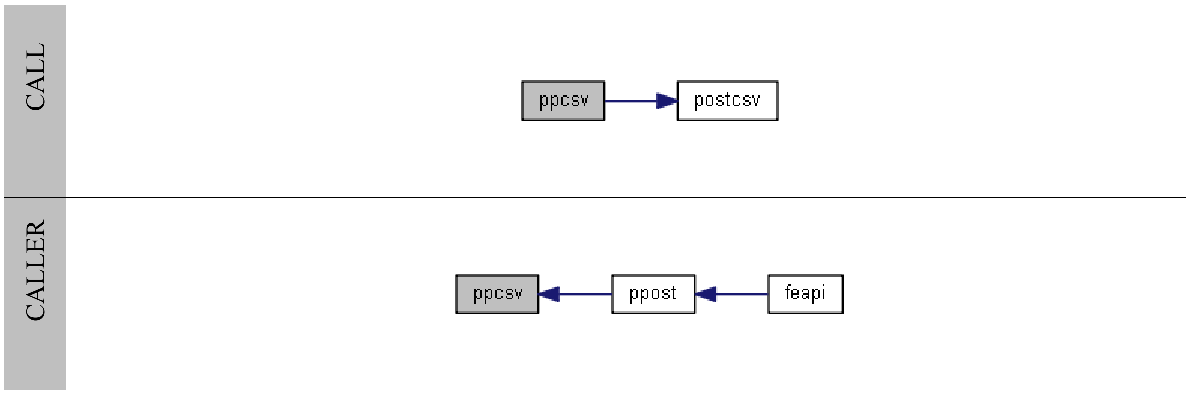
H. postcsv.f90

```
! This sub-routine generates FEAPI Comma Separated Values (CSV) result file.
SUBROUTINE postcsv(db)
USE data_feapi
USE data_post
IMPLICIT NONE
! - Arguments
! Domain block.
INTEGER, INTENT(IN)   :: db
! :
! :
RETURN
END SUBROUTINE postcsv
```



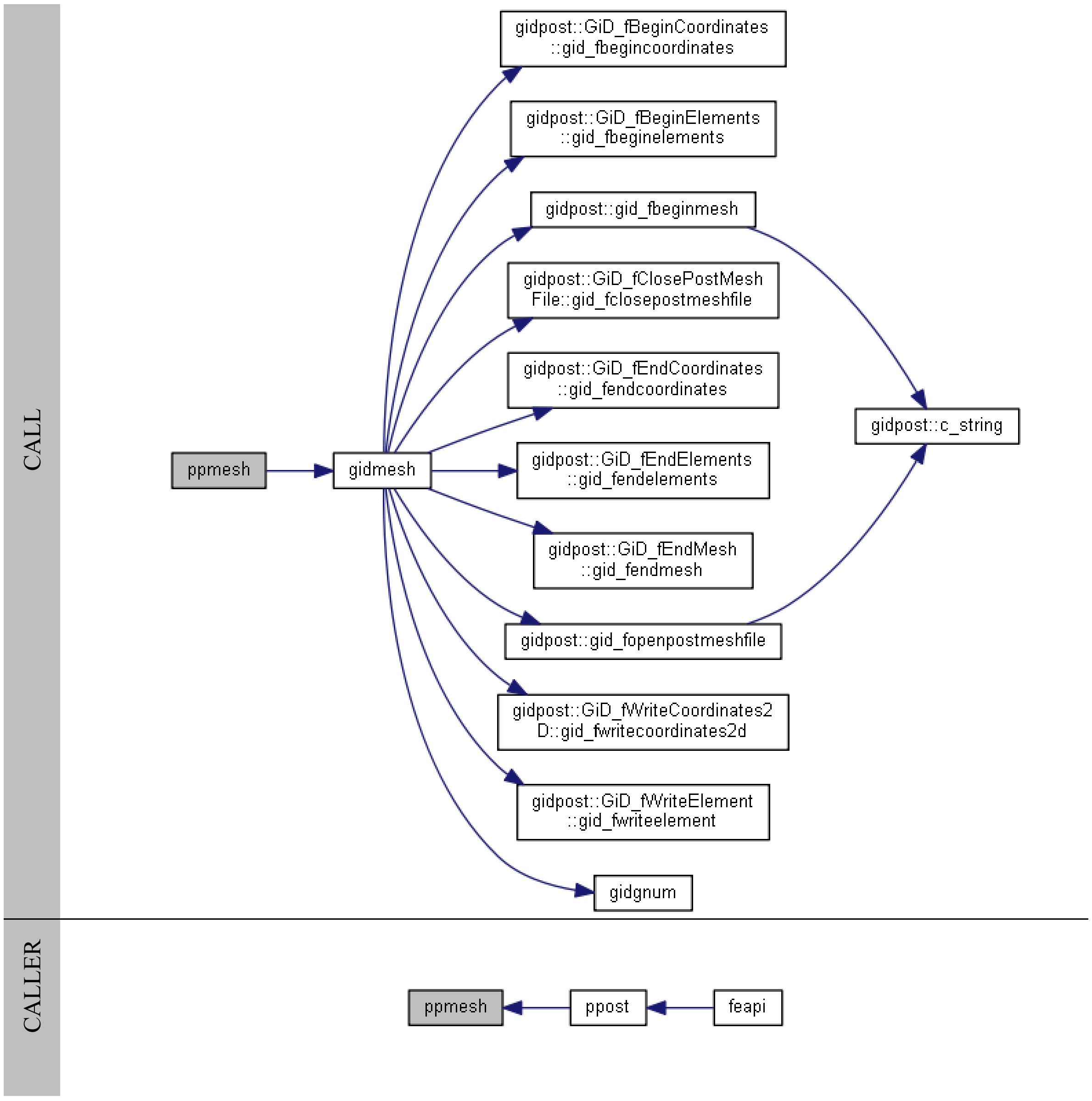
I. ppcsv.f90

```
! This sub-routine is used to print FEAPI CSV results.
SUBROUTINE ppcsv()
USE interfaces
USE data_feapi
USE data_post
IMPLICIT NONE
! :
! :
RETURN
END SUBROUTINE ppcsv
```



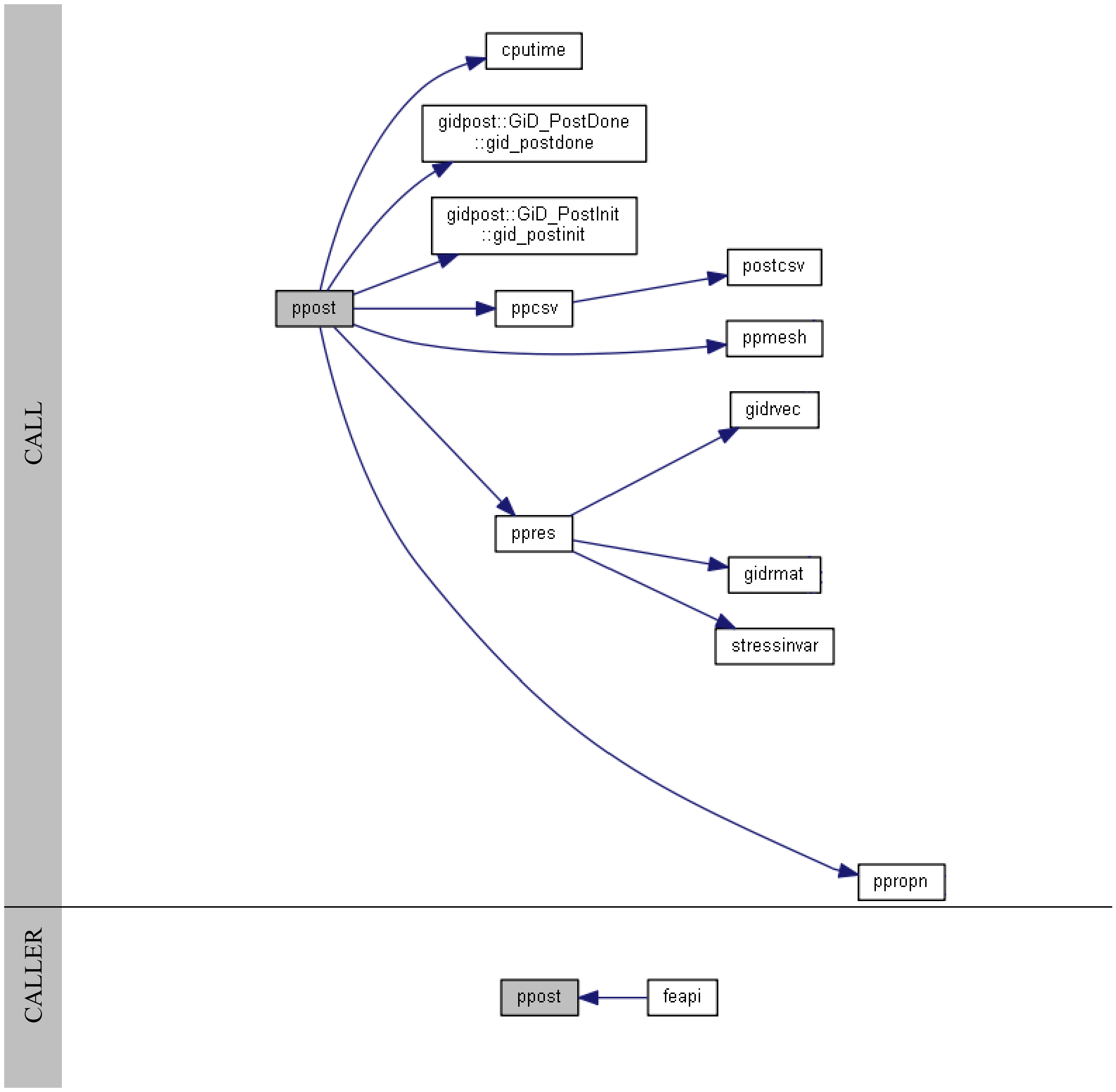
J. ppmesh.f90

```
! This sub-routine generates FEAPI GiD post mesh files.
SUBROUTINE ppmesh()
USE interfaces
USE data_feapi
USE data_post
IMPLICIT NONE
! :
! :
RETURN
END SUBROUTINE ppmesh
```


CALL
ppmesh
gidmesh
gidpost::GiD_fBeginCoordinates ::gid_fbegincoordinates
gidpost::GiD_fBeginElements ::gid_fbeginelements
gidpost::gid_fbeginmesh
gidpost::GiD_fClosePostMesh File::gid_fclosepostmeshfile
gidpost::GiD_fEndCoordinates ::gid_fendcoordinates
gidpost::GiD_fEndElements ::gid_fendelements
gidpost::GiD_fEndMesh ::gid_fendmesh
gidpost::gid_fopenpostmeshfile
gidpost::GiD_fWriteCoordinates2 D::gid_fwritecoordinates2d
gidpost::GiD_fWriteElement ::gid_fwriteelement
gidgnum
gidpost::c_string
CALLER
ppmesh
ppost
feapi

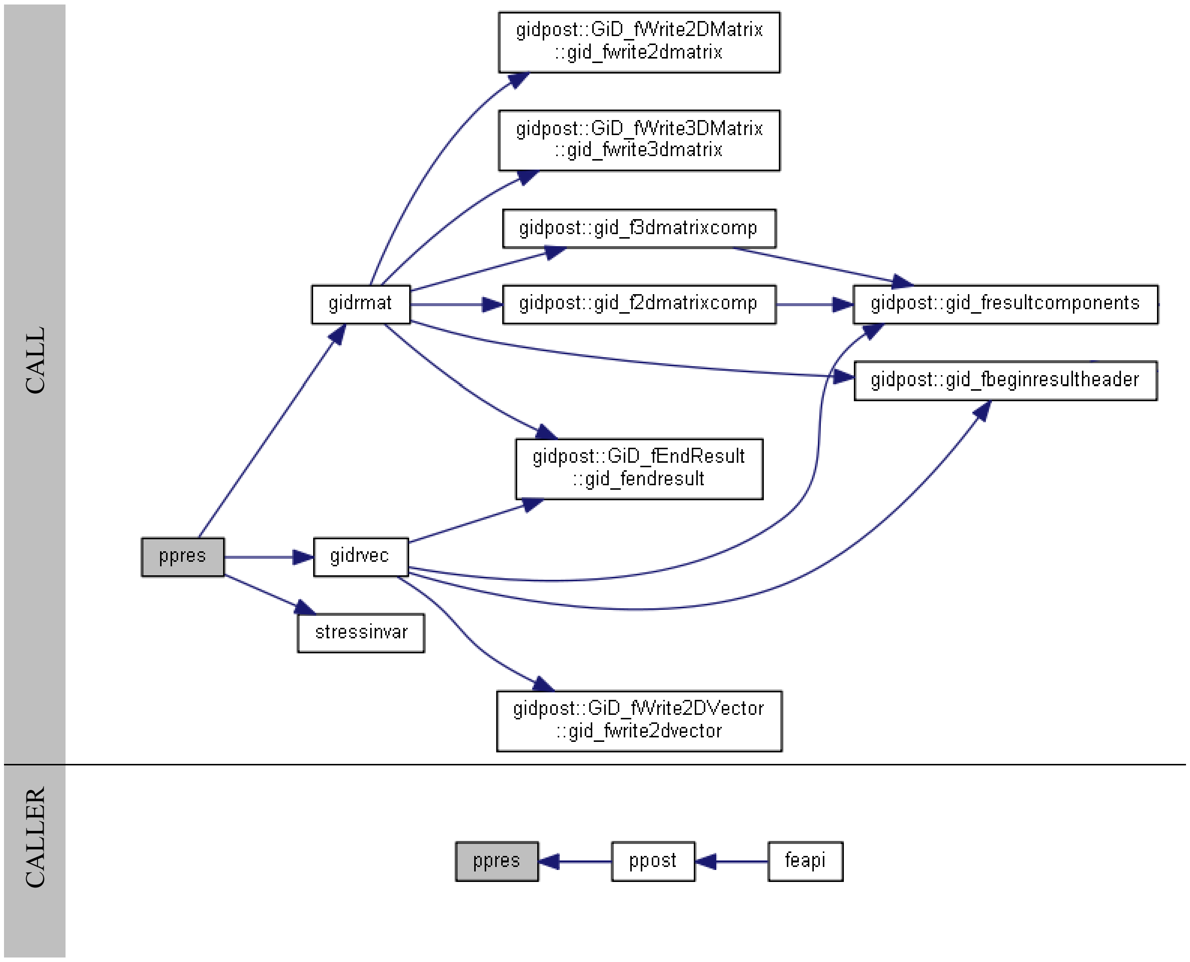
K. ppost.f90

```
! This sub-routine is the primary FEAPI post driver.
SUBROUTINE ppost()
USE interfaces
USE GiDPost
USE data_post
USE data_feapi
IMPLICIT NONE
! :
! :
RETURN
END SUBROUTINE ppost
```



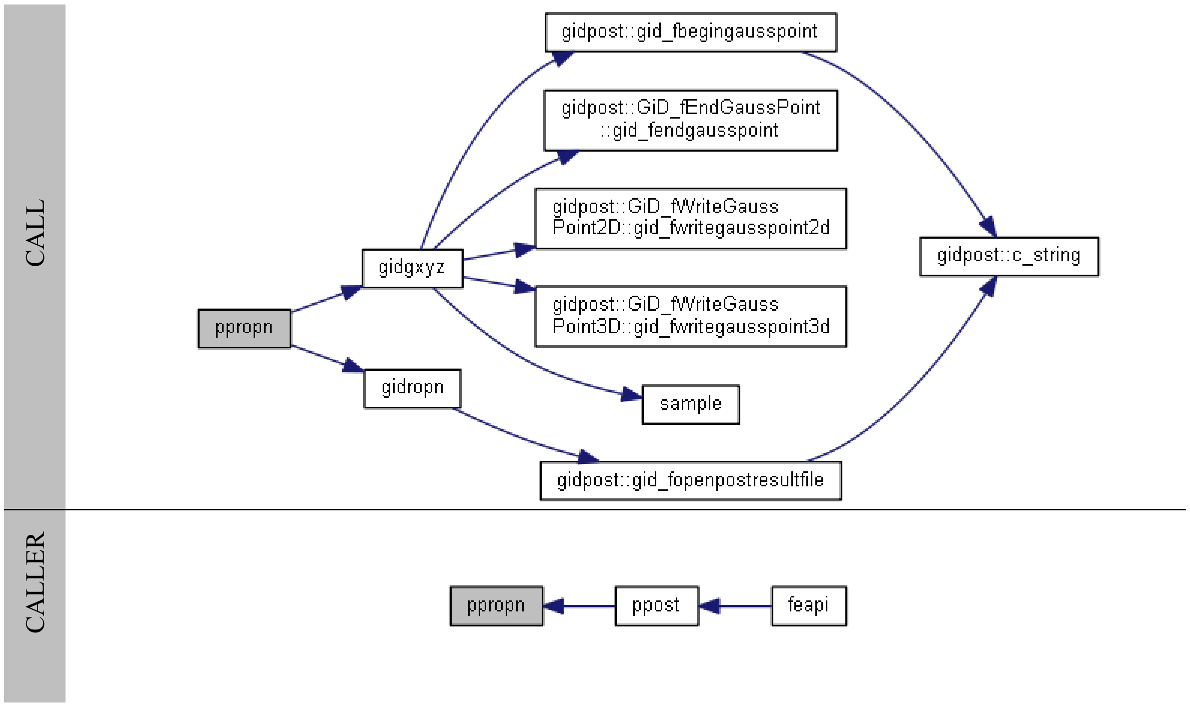
L. ppres.f90

```
! This sub-routine is used to print FEAPI post results.
SUBROUTINE ppres()
USE precision
USE interfaces
USE data_feapi
USE data_post
IMPLICIT NONE
! :
! :
RETURN
END SUBROUTINE ppres
```


CALL
gidpost::GiD_fWrite2DMatrix ::gid_fwrite2dmatrix
gidpost::GiD_fWrite3DMatrix ::gid_fwrite3dmatrix
gidpost::gid_f3dmatrixcomp
gidrmat
gidpost::gid_f2dmatrixcomp
gidpost::gid_fresultcomponents
gidpost::gid_fbeginresultheader
gidpost::GiD_fEndResult ::gid_fendresult
ppres
gidrvec
stressinvar
gidpost::GiD_fWrite2DVector ::gid_fwrite2dvector
CALLER
ppres
ppost
feapi

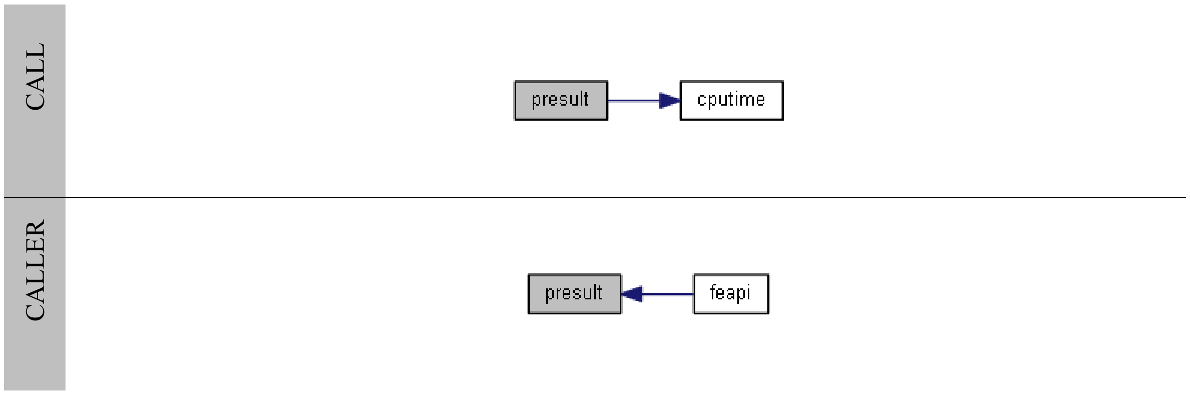
M. ppropn.f90

```
! This sub-routine opens FEAPI GiD post result files.
SUBROUTINE ppropn()
USE data_feapi
USE data_post
IMPLICIT NONE
! :
! :
RETURN
END SUBROUTINE ppropn
```


CALL
gidpost::gid_fbegingausspoint
gidpost::GiD_fEndGaussPoint ::gid_fendgausspoint
gidpost::GiD_fWriteGauss Point2D::gid_fwritegausspoint2d
gidgxyz
gidpost::c_string
ppropn
gidpost::GiD_fWriteGauss Point3D::gid_fwritegausspoint3d
gidropn
sample
gidpost::gid_fopenpostresultfile
CALLER
ppropn
ppost
feapi

N. presult.f90

```
! This sub-routine is used to allocate FEAPI result storage.
SUBROUTINE presult()
USE precision
USE interfaces
USE data_post
USE data_feapi
USE data_domain
USE data_transient
IMPLICIT NONE
! :
! :
RETURN
END SUBROUTINE presult
```

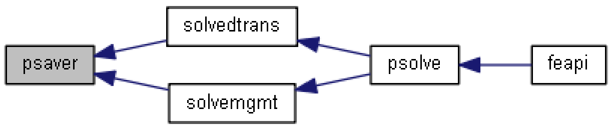


O. psaver.f90

```
! This sub-routine saves FEAPI results to corresponding storage.
SUBROUTINE psaver(db,rt,rstep,resv,resm,cvsr,rn)
USE precision
USE data_feapi
USE data_post
IMPLICIT NONE
! - Arguments
! Domain block.
INTEGER,     INTENT(IN)   :: db
! Result step number.
INTEGER,     INTENT(IN)   :: rstep
! Result time.
REAL(sdp),   INTENT(IN)   :: rt
! Result vector.
REAL(sdp),   INTENT(IN)   :: resv(:)
! Result matrix.
REAL(sdp),   INTENT(IN)   :: resm(:,:,:)
! CVS result.
REAL(sdp),   INTENT(IN)   :: cvsr
! Result name.
CHARACTER(LEN = *),   INTENT(IN)   :: rn
! :
! :
RETURN
END SUBROUTINE psaver
```

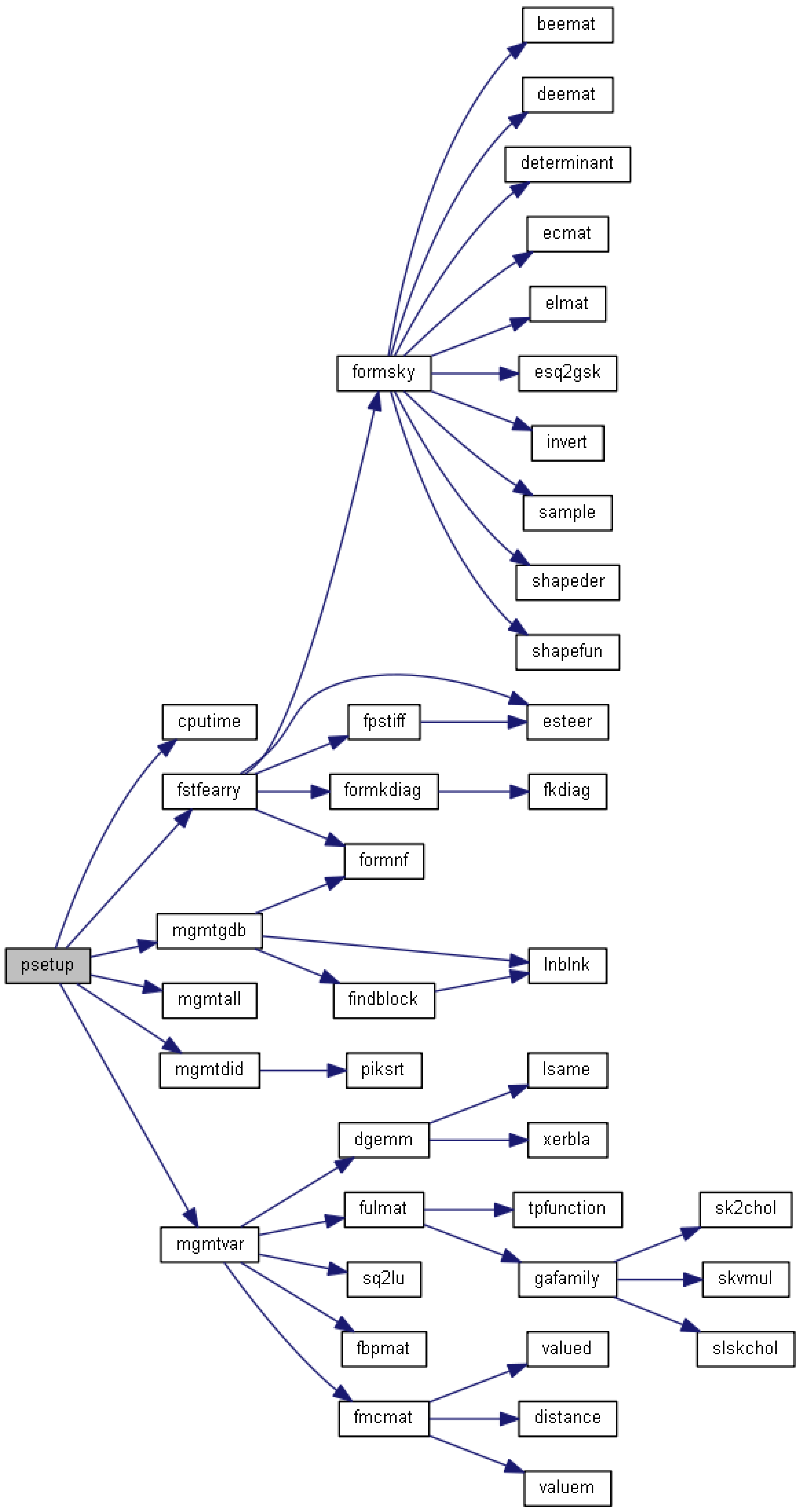
CALL

CALLER

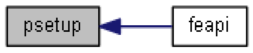


P. psetup.f90

```
! This sub-routine is used to setup FEAPI program variables.
SUBROUTINE psetup()
USE interfaces
USE data_feapi
USE data_mgmt
IMPLICIT NONE

! :
! :
RETURN
END SUBROUTINE psetup
```


beemat
deemat
determinant
ecmat
elmat
formsky
esq2gsk
invert
sample
shapeder
shapefun
cputime
fpstiff
esteer
fstfearry
formkdiag
fkdiag
formnf
mgmtgdb
psetup
lnblnk
mgmtall
findblock
mgmtdid
piksrt
lsame
dgemm
xerbla
fulmat
tpfunction
sk2chol
mgmtvar
sq2lu
gafamily
skvmul
fbpmat
valued
slskchol
fmcmat
distance
valuem

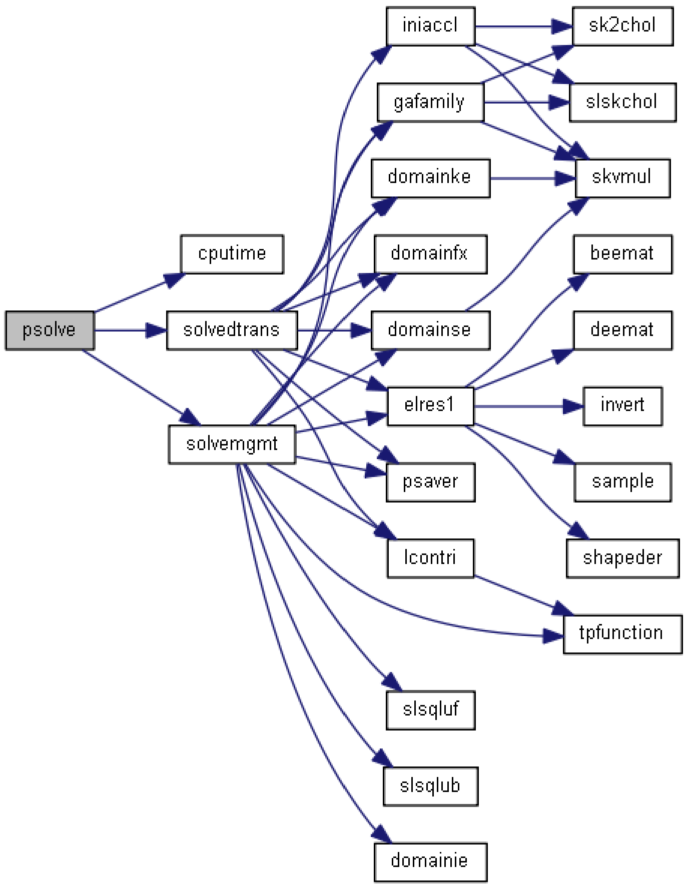
CALLER

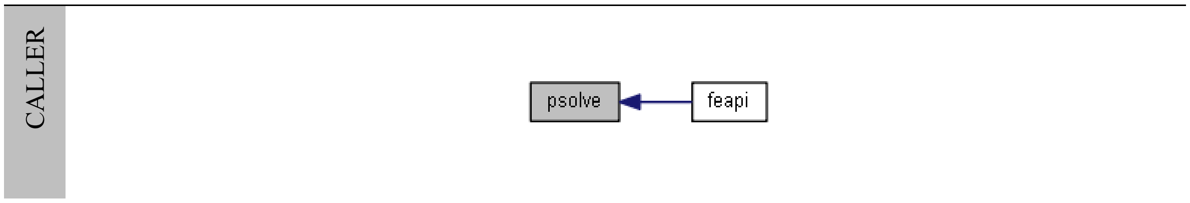


Q. psolve.f90

```
! This sub-routine is the primary FEAPI solution driver.
SUBROUTINE psolve()
USE precision
USE interfaces
USE data_feapi
USE data_mgmt
IMPLICIT NONE
! :
! :
RETURN
END SUBROUTINE psolve
```

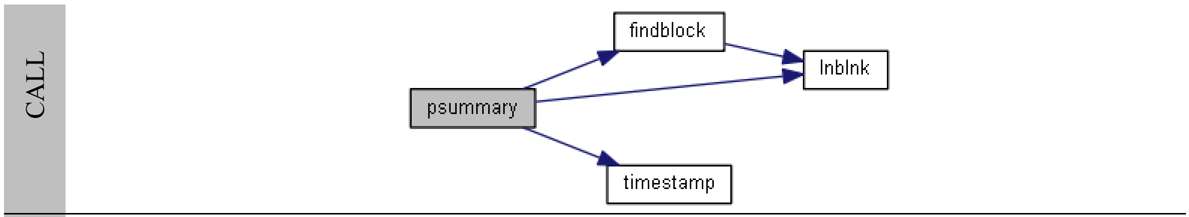
CALL

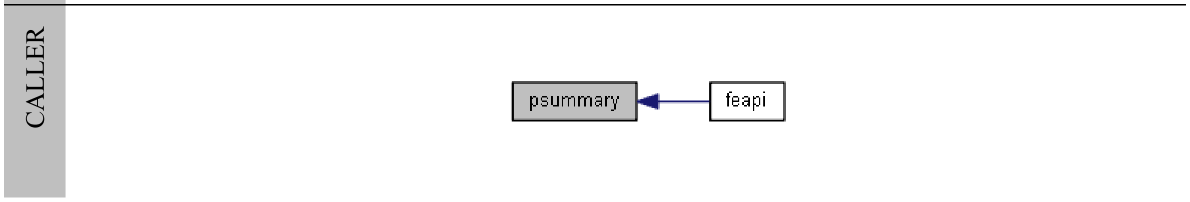




R. psummary.f90

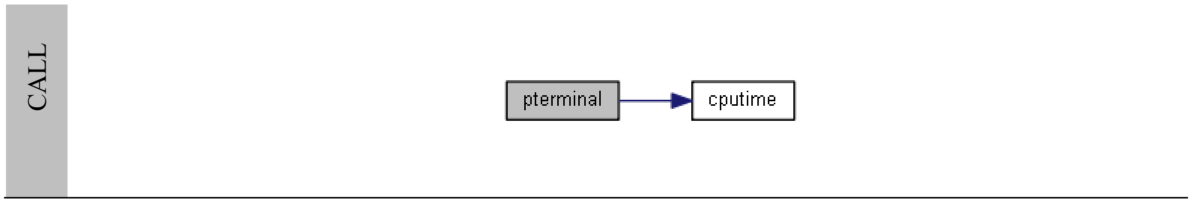
```
! This sub-routine posts FEAPI program summary.
SUBROUTINE psummary()
USE interfaces
USE data_mgmt
USE data_post
USE data_feapi
USE data_domain
USE data_transient
USE data_structural
IMPLICIT NONE
! :
! :
RETURN
END SUBROUTINE psummary
```



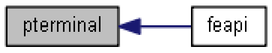


S. pterminal.f90

```
! FEAPI program terminal.
SUBROUTINE pterminal()
USE interfaces
USE data_feapi
IMPLICIT NONE
! :
! :
RETURN
END SUBROUTINE pterminal
```



CALLER

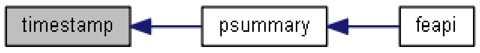


T. timestamp.f90

```
! This function the current timestamp.
FUNCTION timestamp()
IMPLICIT NONE
! - Arguments.
! Timestamp. Example: '20 April 2012 4:20:01.234 PM'
CHARACTER(LEN = 40)   :: timestamp
! :
! :
RETURN
END FUNCTION timestamp
```

CALL

CALLER

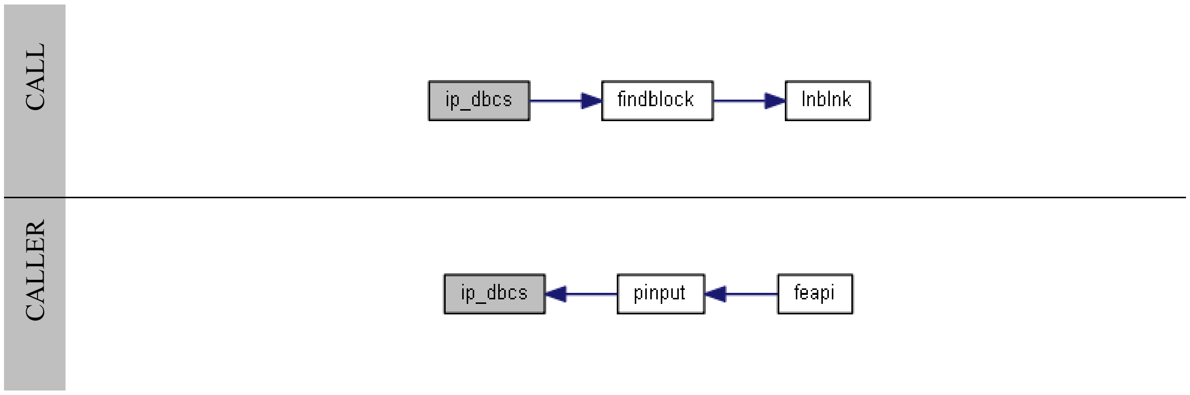


D.2 FEAPI–GiD

D.2.1 Preprocessor

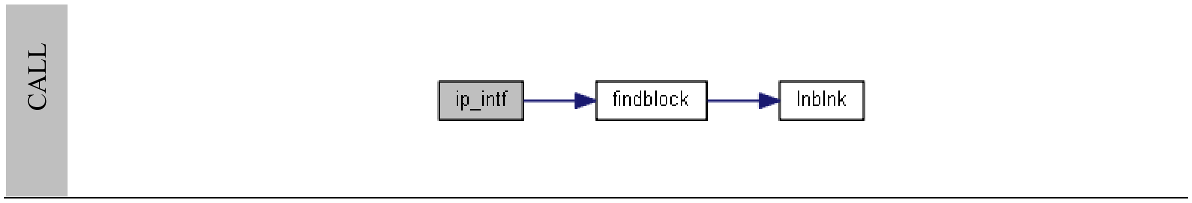
A. ip_dbcs.f90

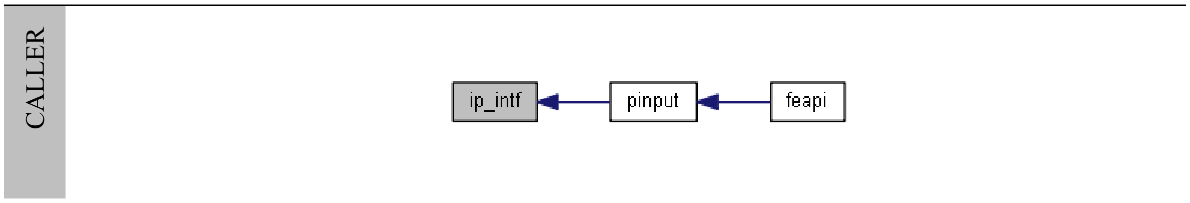
```
! This sub-routine reads FEAPI domain boundary conditions.
SUBROUTINE ip_dbcs(db)
USE interfaces
USE data_feapi
USE data_domain
IMPLICIT NONE
! - Arguments.
INTEGER, INTENT(IN)  :: db ! Domain block.
! :
! :
RETURN
END SUBROUTINE ip_dbcs
```



B. ip_intf.f90

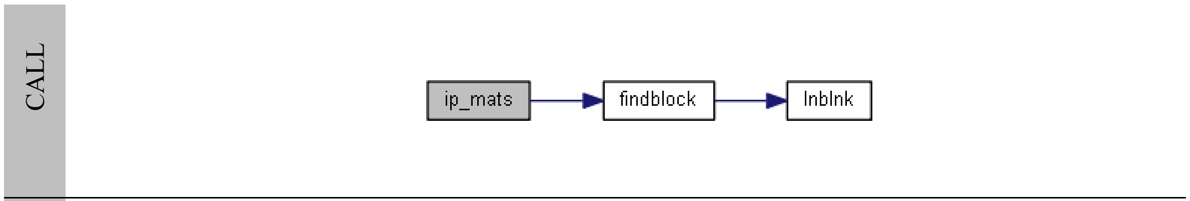
```
! This sub-routine reads FEAPI domain interface variables.
SUBROUTINE ip_intf(db)
USE interfaces
USE data_feapi
USE data_domain
IMPLICIT NONE
! - Arguments.
INTEGER, INTENT(IN)  :: db ! Domain block.
! :
! :
RETURN
END SUBROUTINE ip_intf
```

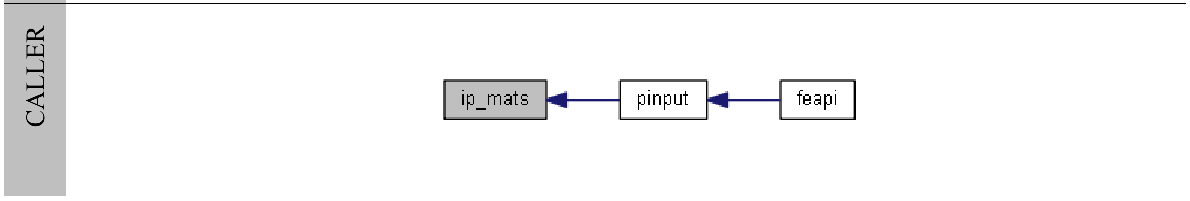




C. ip_mats.f90

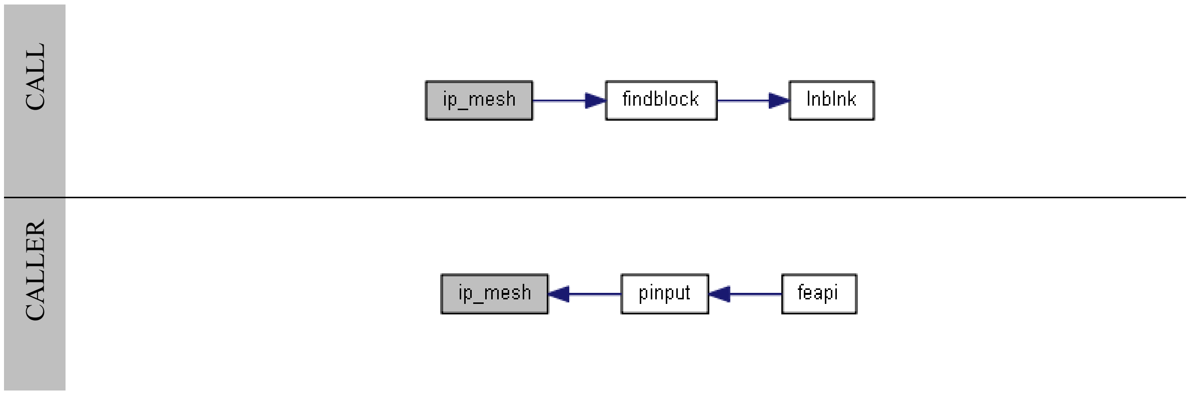
```
! This sub-routine reads FEAPI material variables.
SUBROUTINE ip_mats(db)
USE interfaces
USE data_feapi
USE data_domain
USE data_material
IMPLICIT NONE
! - Arguments.
INTEGER, INTENT(IN)   :: db ! Domain block.
! :
! :
RETURN
END SUBROUTINE ip_mats
```





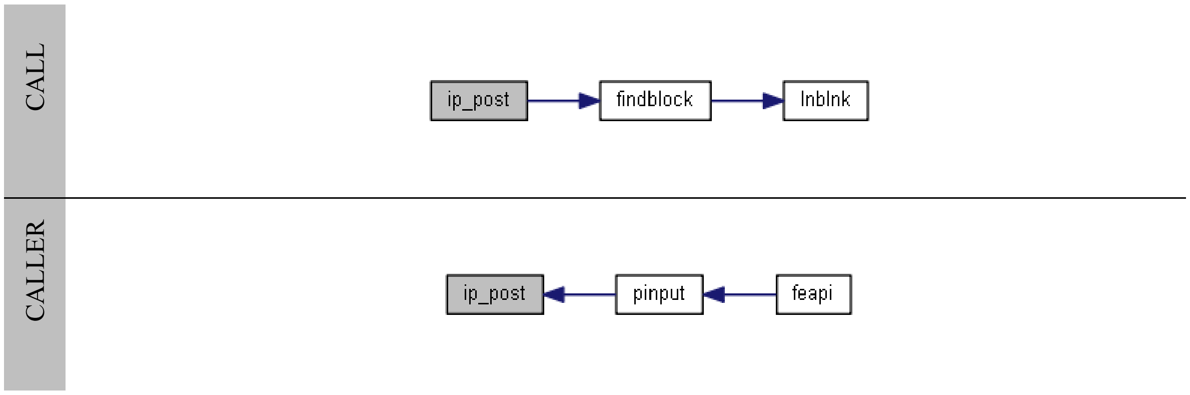
D. ip_mesh.f90

```
! This This sub-routine reads FEAPI domain variables.
SUBROUTINE ip_mesh(db)
USE interfaces
USE data_feapi
USE data_domain
IMPLICIT NONE
! - Arguments.
INTEGER, INTENT(IN)   :: db ! Domain block.
! :
! :
RETURN
END SUBROUTINE ip_mesh
```



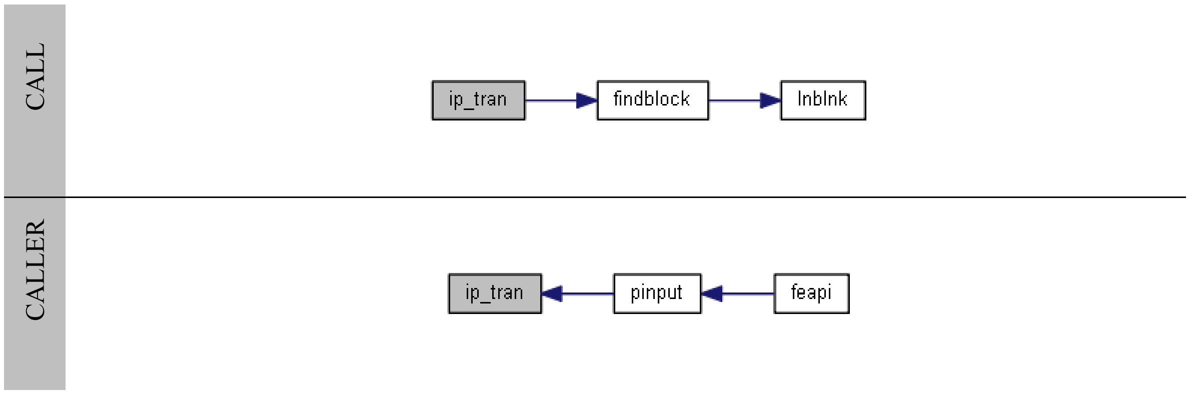
E. ip_post.f90

```
! This sub-routine reads FEAPI post variables.
SUBROUTINE ip_post(db)
USE interfaces
USE data_feapi
USE data_post
IMPLICIT NONE
! - Arguments.
INTEGER, INTENT(IN)  :: db ! Domain block.
! :
! :
RETURN
END SUBROUTINE ip_post
```



F. ip_tran.f90

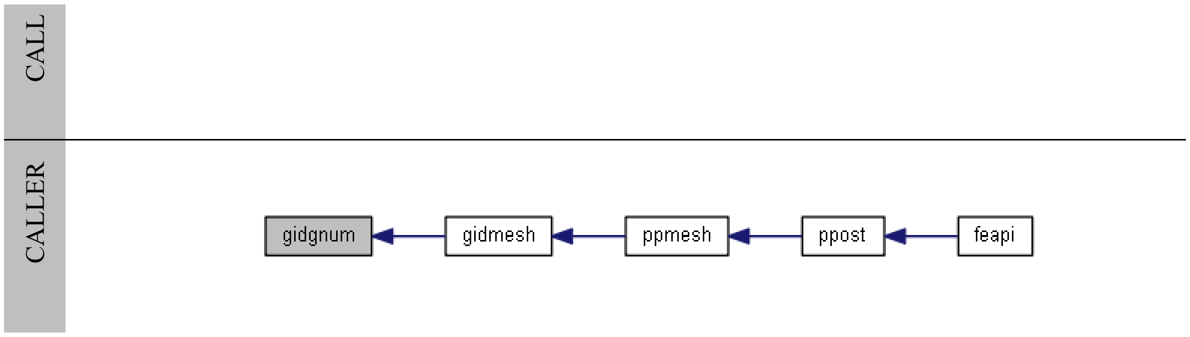
```
! This sub-routine reads FEAPI transient analysis options.
SUBROUTINE ip_tran(db)
USE interfaces
USE data_feapi
USE data_transient
IMPLICIT NONE
! - Arguments.
INTEGER, INTENT(IN)  :: db ! Domain block.
! :
! :
RETURN
END SUBROUTINE ip_tran
```



D.2.2 Postprocessor

A. gidgnum.f90

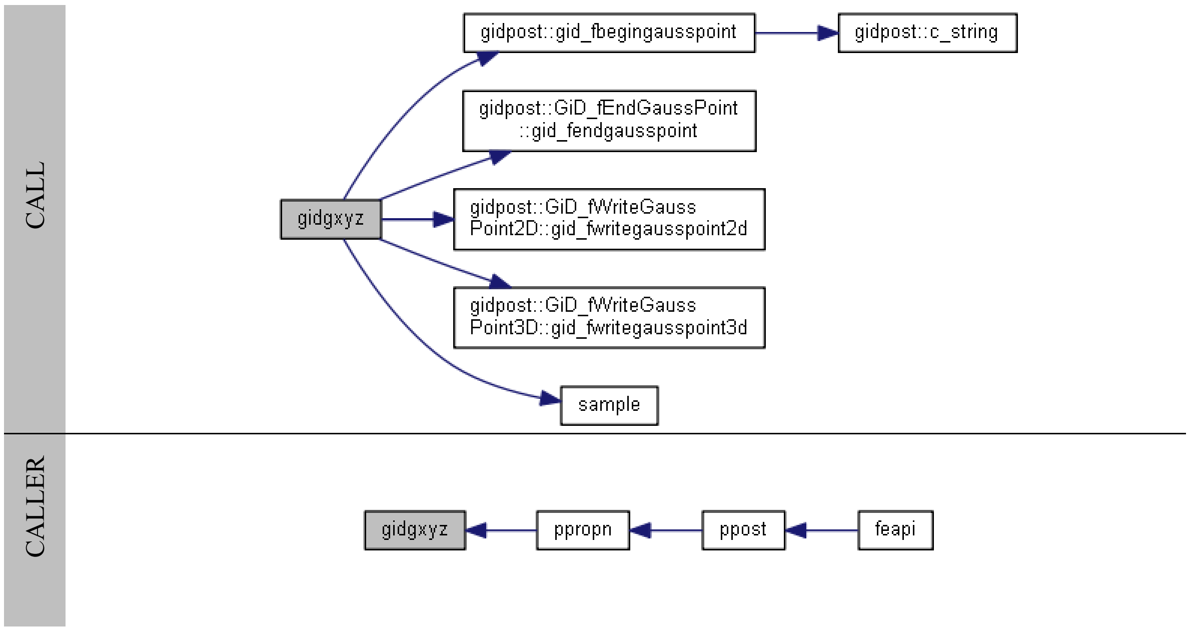
```
! This sub-routine transforms FEAPI (local) element node numbering
! to GiD (local) element node numbering.
SUBROUTINE gidgnum(etype,nod,g_num,gnum)
IMPLICIT NONE
! - Arguments.
CHARACTER(LEN = *),  INTENT(IN)     :: etype ! Element type.
INTEGER,             INTENT(IN)     :: nod ! Nodes per element.
INTEGER,             INTENT(IN)     :: g_num(:) ! FEAPI element connectivity.
INTEGER,             INTENT(INOUT)  :: gnum(:) ! GiD element connectivity.
! :
! :
RETURN
END SUBROUTINE gidgnum
```



B. gidgxyz.f90

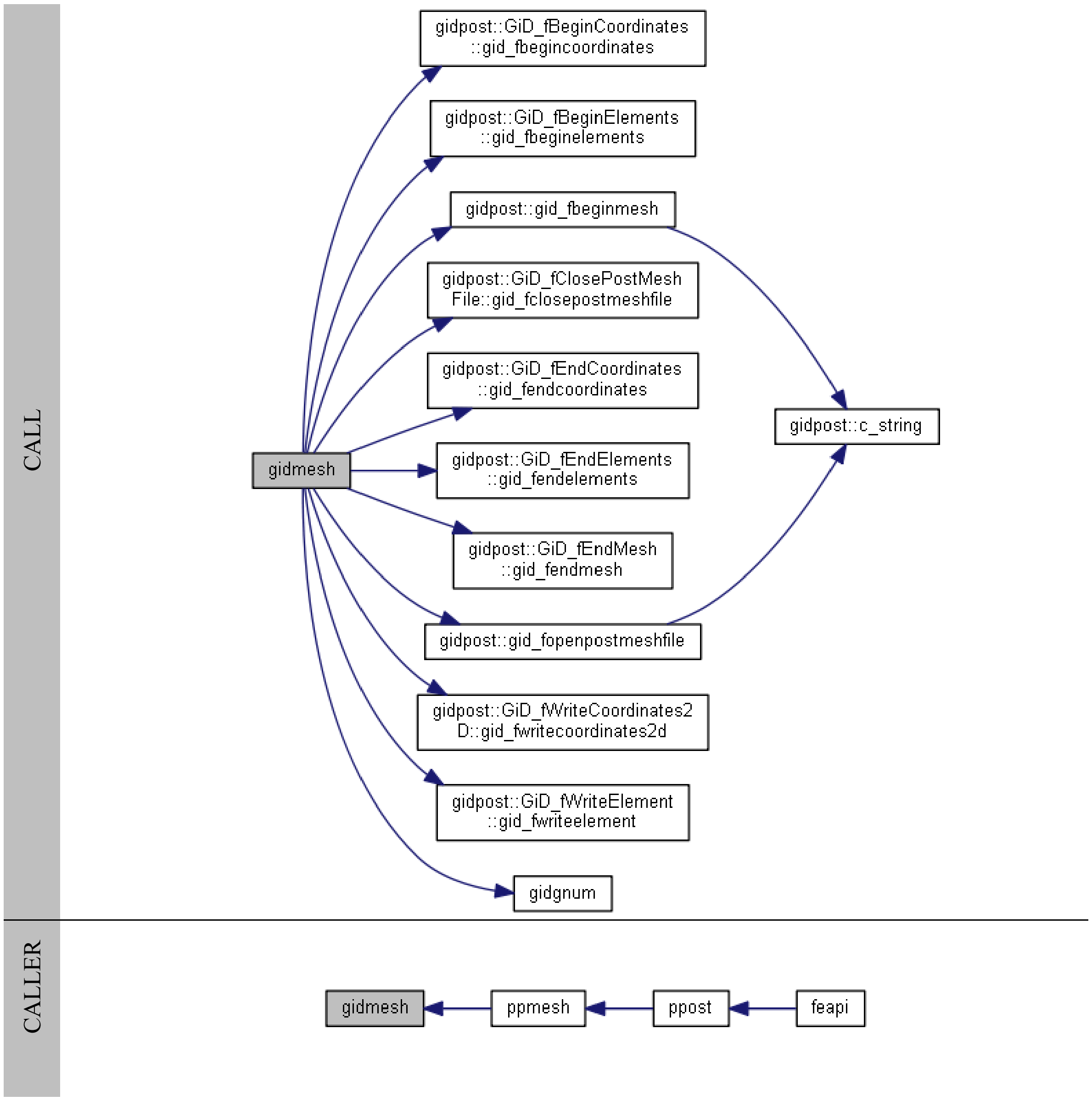
```
! This sub-routine posts Gauss integration point coordinates to
! GiD post result file.
SUBROUTINE gidgxyz(db)
USE precision
USE interfaces
USE data_feapi
USE data_domain
USE GiDPost
USE data_post
IMPLICIT NONE
! - Arguments.
```

```
INTEGER, INTENT(IN)  :: db ! Domain block.
! :
! :
RETURN
END SUBROUTINE gidgxyz
```


CALL
gidgxyz
gidpost::gid_fbegingausspoint
gidpost::c_string
gidpost::GiD_fEndGaussPoint ::gid_fendgausspoint
gidpost::GiD_fWriteGauss Point2D::gid_fwritegausspoint2d
gidpost::GiD_fWriteGauss Point3D::gid_fwritegausspoint3d
sample
CALLER
gidgxyz
ppropn
ppost
feapi

C. gidmesh.f90

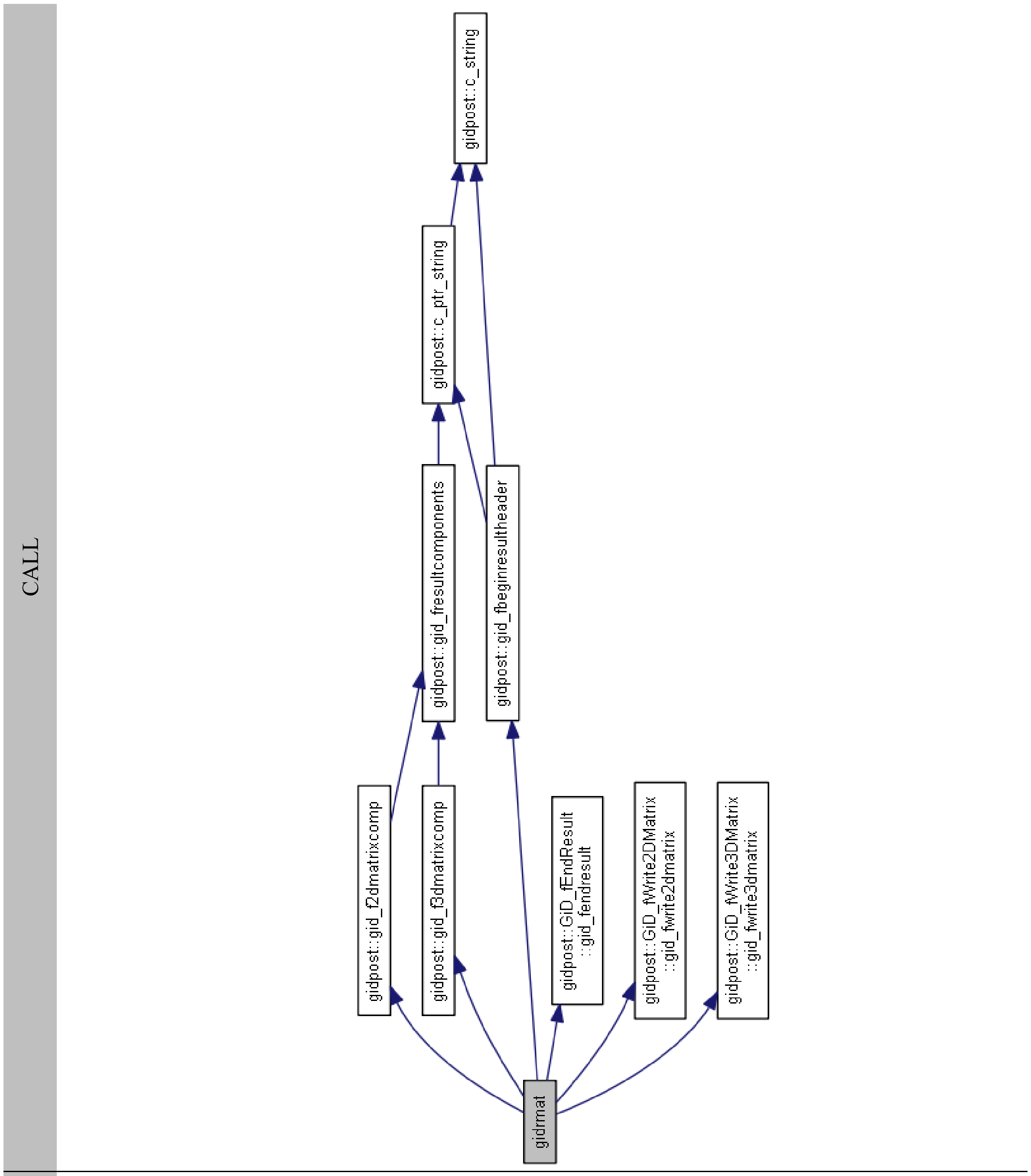
```
! This sub-routine posts FEAPI GiD post mesh file.
SUBROUTINE gidmesh(db)
USE precision
USE interfaces
USE data_feapi
USE data_domain
USE GiDPost
USE data_post
IMPLICIT NONE
! - Arguments.
INTEGER, INTENT(IN)  :: db ! Domain block.
! :
! :
RETURN
END SUBROUTINE gidmesh
```

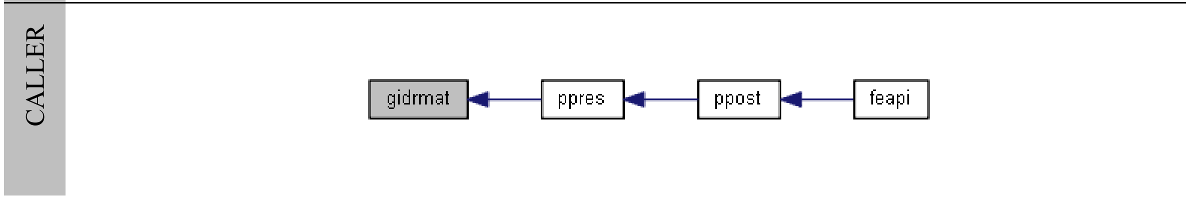

CALL
gidpost::GiD_fBeginCoordinates ::gid_fbegincoordinates
gidpost::GiD_fBeginElements ::gid_fbeginelements
gidpost::gid_fbeginmesh
gidpost::GiD_fClosePostMesh File::gid_fclosepostmeshfile
gidpost::GiD_fEndCoordinates ::gid_fendcoordinates
gidpost::c_string
gidmesh
gidpost::GiD_fEndElements ::gid_fendelements
gidpost::GiD_fEndMesh ::gid_fendmesh
gidpost::gid_fopenpostmeshfile
gidpost::GiD_fWriteCoordinates2 D::gid_fwritecoordinates2d
gidpost::GiD_fWriteElement ::gid_fwriteelement
gidgnum
CALLER
gidmesh
ppmesh
ppost
feapi

D. gidrmat.f90

```
! This sub-routine posts FEAPI matrix results
! (results over Gauss integration points).
SUBROUTINE gidrmat(db,rt,rn,rmat)
USE GiDPost
USE data_post
USE data_feapi
USE data_domain
IMPLICIT NONE
! - Arguments.
INTEGER,             INTENT(IN)  :: db ! Domain block.
REAL(sdp),           INTENT(IN)  :: rt ! Result time.
CHARACTER(LEN =  *), INTENT(IN)  :: rn ! Result name.
REAL(sdp),           INTENT(IN)  :: rmat(:,:,:) ! Result vector.
! :
```

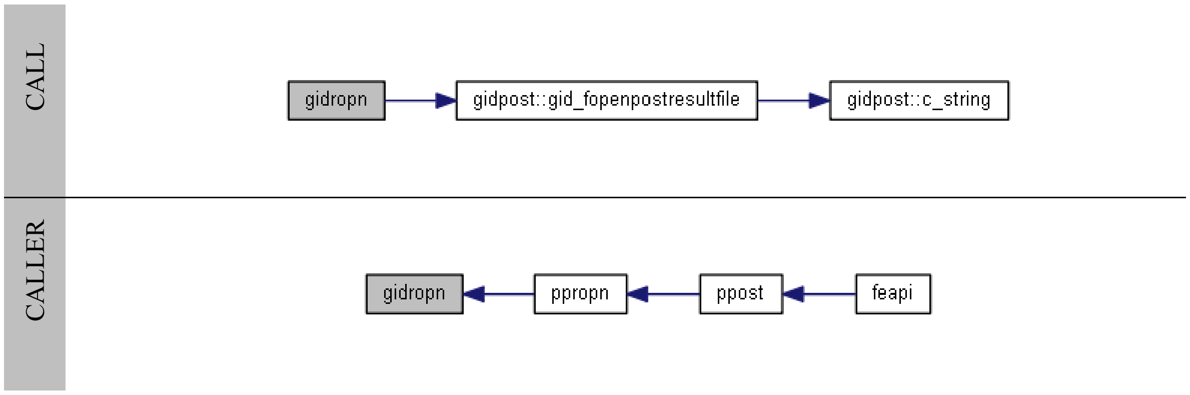
```
! :
RETURN
END SUBROUTINE gidrmat
```


CALL
gidpost::c_string
gidpost::c_ptr_string
gidpost::gid_fresultcomponents
gidpost::gid_fbeginresultheader
gidpost::gid_f2dmatrixcomp
gidpost::gid_f3dmatrixcomp
gidpost::GiD_fEndResult ::gid_fendresult
gidpost::GiD_fWrite2DMatrix ::gid_fwrite2dmatrix
gidpost::GiD_fWrite3DMatrix ::gid_fwrite3dmatrix
gidrmat



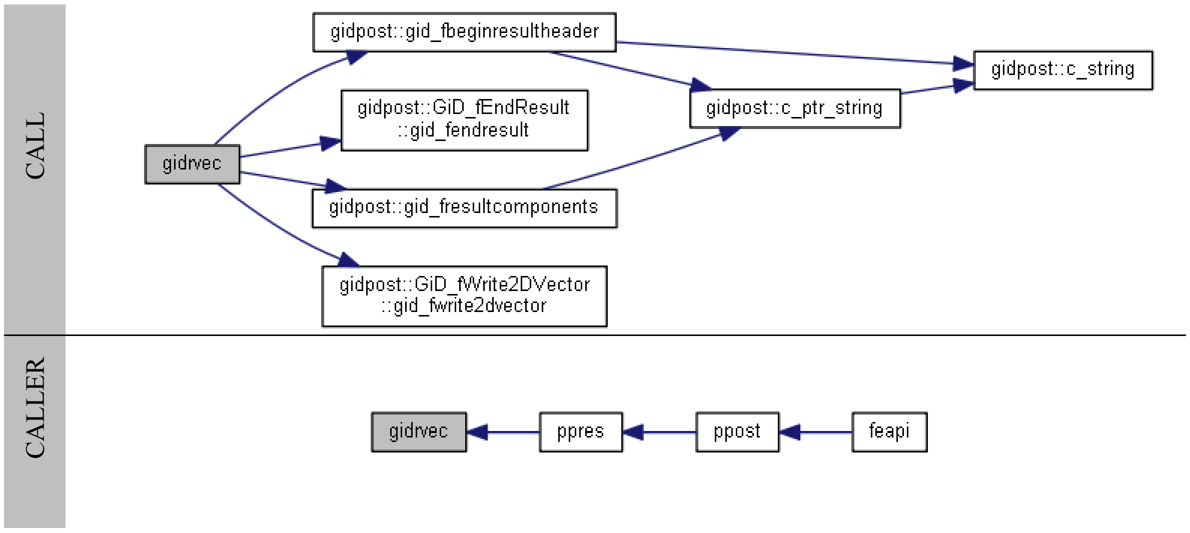
E. gidropn.f90

```
! This sub-routine opens a FEAPI-GiD post result file.
SUBROUTINE gidropn(db)
USE GiDPost
USE data_feapi
USE data_post
IMPLICIT NONE! - Arguments.
INTEGER, INTENT(IN)   :: db ! Domain block.
! :
! :
RETURN
END SUBROUTINE gidropn
```



F. gidrvec.f90

```
! This sub-routine posts FEAPI vector results (results over nodes).
SUBROUTINE gidrvec(db,rt,rn,rvec)
USE precision
USE GiDPost
USE data_post
USE data_feapi
USE data_domain
IMPLICIT NONE
! - Arguments.
INTEGER,                INTENT(IN)   :: db ! Domain block.
REAL(sdp),              INTENT(IN)   :: rt ! Result time.
CHARACTER(LEN =  *),    INTENT(IN)   :: rn ! Result name.
REAL(sdp),              INTENT(IN)   :: rvec(0:) ! Result vector.
! :
! :
RETURN
END SUBROUTINE gidrvec
```

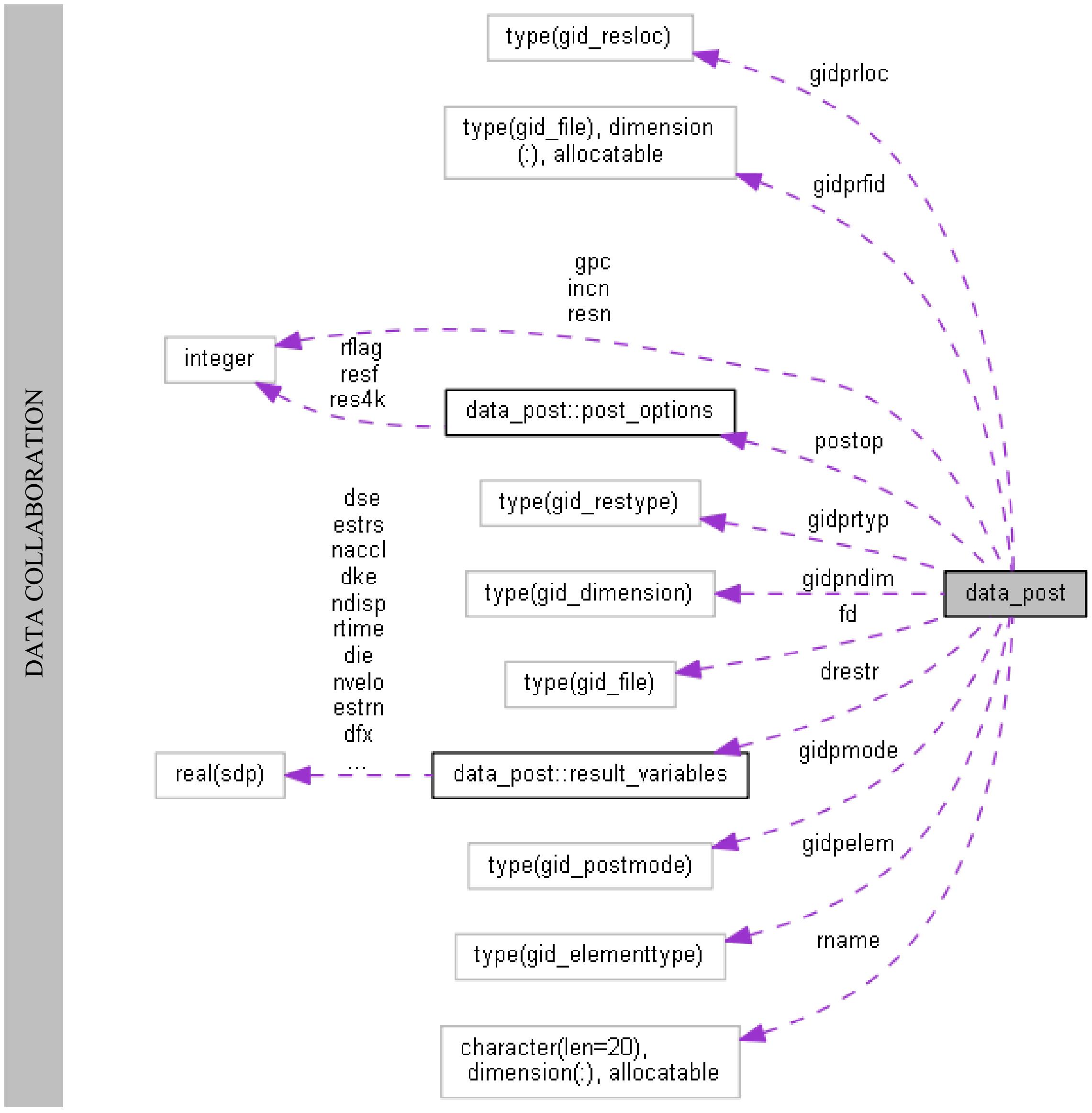

CALL
gidrvec
gidpost::gid_fbeginresultheader
gidpost::GiD_fEndResult ::gid_fendresult
gidpost::gid_fresultcomponents
gidpost::GiD_fWrite2DVector ::gid_fwrite2dvector
gidpost::c_ptr_string
gidpost::c_string
CALLER
gidrvec
ppres
ppost
feapi

D.2.3 GiDPost

A. data_post.f90

```
! This module contains FEAPI post variables
MODULE data_post.f90
USE precision
USE GiDPost
IMPLICIT NONE

! Data type for domain result variables.
TYPE :: result_variables
REAL(sdp),  ALLOCATABLE :: rtime(:) ! Result time.
REAL(sdp),  ALLOCATABLE :: ndisp(:,:) ! Nodal displacements.
REAL(sdp),  ALLOCATABLE :: nvelo(:,:) ! Nodal velocities.
REAL(sdp),  ALLOCATABLE :: naccl(:,:) ! Nodal accelerations.
REAL(sdp),  ALLOCATABLE :: estrs(:,:,:,:) ! Element stresses.
REAL(sdp),  ALLOCATABLE :: estrn(:,:,:,:) ! Element strains.
REAL(sdp),  ALLOCATABLE :: dke(:) ! Domain kinetic energy.
REAL(sdp),  ALLOCATABLE :: dse(:) ! Domain stiffness energy.
REAL(sdp),  ALLOCATABLE :: dfx(:) ! Domain external work.
REAL(sdp),  ALLOCATABLE :: die(:) ! Domain interface energy.
END TYPE result_variables

! Data type for FEAPI post options.
TYPE :: post_options
INTEGER :: resf ! Result frequency.
INTEGER, ALLOCATABLE :: rflag(:) ! Result flag.
INTEGER :: res4k ! Result keyword (4. Stress).
END TYPE post_options

! Public attributes
INTEGER :: resn = 9 ! Number of FEAPI results.
INTEGER :: incn ! Include nodes.
INTEGER :: gpc ! Gauss point coordinates.
CHARACTER(LEN = 20), ALLOCATABLE :: rname(:) ! Result names.

! Public attributes (DDT = Derived data type)
TYPE(GiD_PostMode) :: gidpmode
! DDT for GiD spatial dimension.
```

```
TYPE(GiD_Dimension) :: gidpndim
! DDT for GiD element type.
TYPE(GiD_ElementType) :: gidpelem
! DDT for result type.
TYPE(GiD_ResType) :: gidprtyp
! DDT for result location.
TYPE(GiD_ResLoc) :: gidprloc
! DDT for GiD file unit identifier.
TYPE(GiD_File) :: fd
! DDT for GiD post.res file unit identifier.
TYPE(GiD_File), ALLOCATABLE :: gidprfid(:)
! DDT for GiDPost options.
TYPE(post_options), ALLOCATABLE :: postop(:)
! DDT for domain result storage.
TYPE(result_variables), ALLOCATABLE :: drestr(:)
END MODULE data_post
```



B. GiDPost.f90

Refer: http://www.gidhome.com/component/manual/gidpost/introduction

C. GiD_hdf5.lib

Refer: http://www.gidhome.com/component/manual/gidpost/introduction

D. GiD_post.lib

Refer: http://www.gidhome.com/component/manual/gidpost/introduction

E. GiD_zlib.lib

Refer: http://www.gidhome.com/component/manual/gidpost/introduction

D.3 MGMT

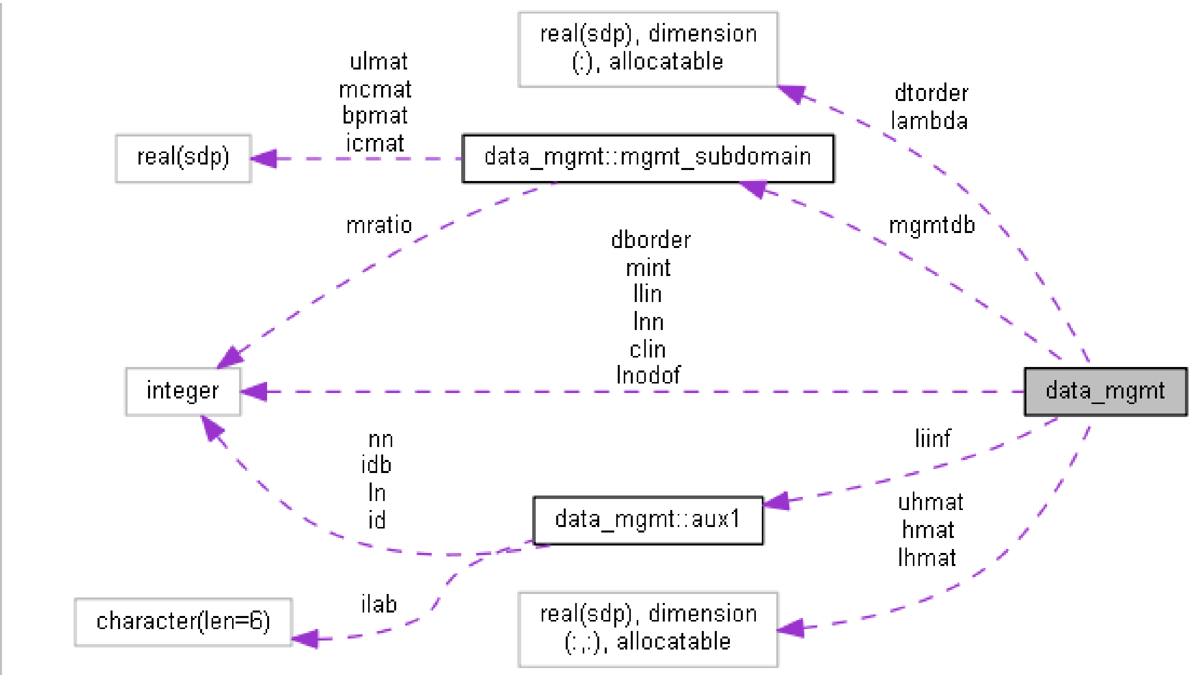
D.3.1 Modules

A. data_mgmt.f90

```
! This module contains MGMT variables.
MODULE data_mgmt
USE precision
IMPLICIT NONE
! Auxiliary data type for domain interface info.
TYPE :: aux1
       ! Interface domain block.
       INTEGER :: idb
       ! Interface ID.
       INTEGER :: id
       ! Interface label.
       CHARACTER(LEN = 6)   :: ilab
       ! Number of nodes on a particular interface.
       INTEGER :: nn
       ! List of nodes on a particular interface.
       INTEGER, ALLOCATABLE :: ln(:)
END TYPE aux1
! Total number of 'Master' ( = Mortar) interfaces.
INTEGER :: mint
! Total number of nodes associated with the interface of
INTEGER :: lnn
! List of Lagrange interface nodes.
INTEGER, ALLOCATABLE :: llin(:,:)
! (Clone) List of Lagrange interface nodes.
INTEGER, ALLOCATABLE :: clin(:,:)
! Degrees of freedom (per node) associated with the Lagrange multipliers.
INTEGER :: lnodof
! Lagrange interface info.
TYPE(aux1), ALLOCATABLE :: liinf(:)
! Sub-domain time-step hierarchy (descending).
REAL(sdp),  ALLOCATABLE :: dtorder(:)
! Sub-domain hierarchy.
INTEGER,  ALLOCATABLE :: dborder(:)
! Array of Lagrange multipliers.
REAL(sdp),  ALLOCATABLE :: lambda(:)
! Interface condensation matrix.
REAL(sdp),  ALLOCATABLE :: hmat(:,:)
! Lower triangular decomposition of the interface condensation matrix.
REAL(sdp),  ALLOCATABLE :: lhmat(:,:)
! Upper triangular decomposition of the interface condensation matrix.
REAL(sdp),  ALLOCATABLE :: uhmat(:,:)
! Data type for sub-domain specific MGMT variables.
TYPE :: mgmt_subdomain
       ! Sub-domain time-step ratio with respect to parent sub-domain.
       INTEGER :: mratio
       ! Boolean projection matrix.
       REAL(sdp),  ALLOCATABLE :: bpmat(:,:)
       ! Multi constraint interface matrix.
       REAL(sdp),  ALLOCATABLE :: mcmat(:,:)
       ! Interface connectivity matrix.
       REAL(sdp),  ALLOCATABLE :: icmat(:,:)
       ! Unit load response matrix.
       REAL(sdp),  ALLOCATABLE :: ulmat(:,:,:,:)
```

```
END TYPE mgmt_subdomain
! Derived data type for sub-domain specific MGMT variables.
TYPE(mgmt_subdomain), ALLOCATABLE :: mgmtdb(:)
END MODULE data_mgmt
```

DATA COLLABORATION

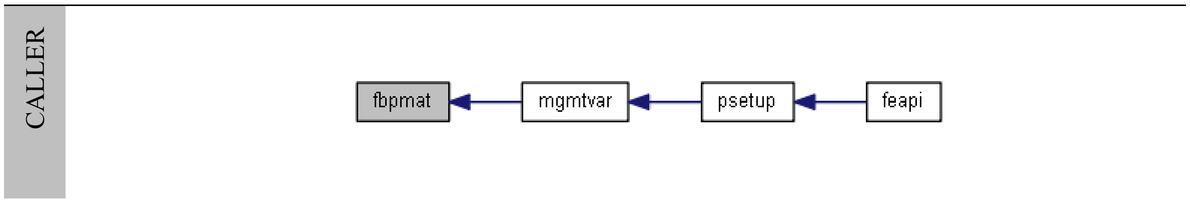


D.3.2 Library

A. fbpmat.f90

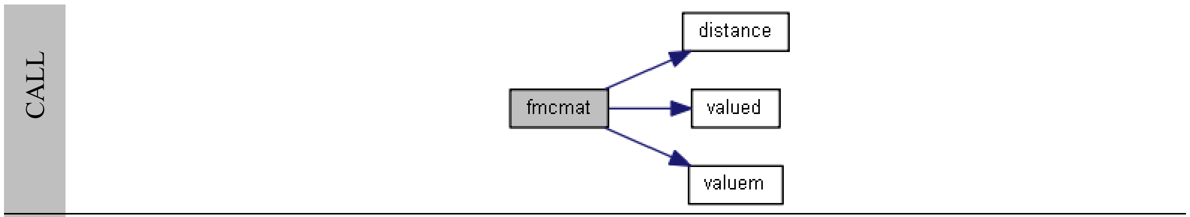
```
! This sub-routine forms the Boolean projection matrix (unbiased).
SUBROUTINE fbpmat(lnodes,gnf,mat)
USE precision
IMPLICIT NONE
! - Arguments.
! List of domain interface nodes.
INTEGER,    INTENT(IN)     :: lnodes(:)
! Domain nodal freedom matrix.
INTEGER,    INTENT(IN)     :: gnf(:,:)
! Boolean projection matrix (unbiased).
REAL(sdp),  INTENT(INOUT)  :: mat(:,0:)
! :
! :
RETURN
END SUBROUTINE fbpmat
```

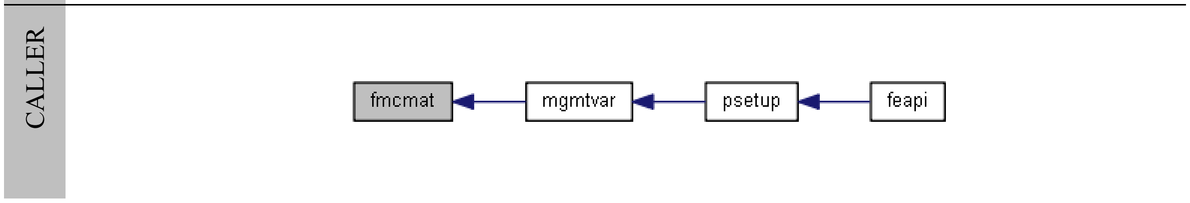
CALL



B. fmcmat.f90

```
! This sub-routine computes the multi constraint matrix
! using mortar finite elements (Trapezoidal Rule).
SUBROUTINE fmcmat(db)
USE precision
USE interfaces
USE data_mgmt
USE data_domain
IMPLICIT NONE
! - Arguments.
! Domain block.
INTEGER, INTENT(IN)   :: db
! :
! :
RETURN
END SUBROUTINE fmcmat
```

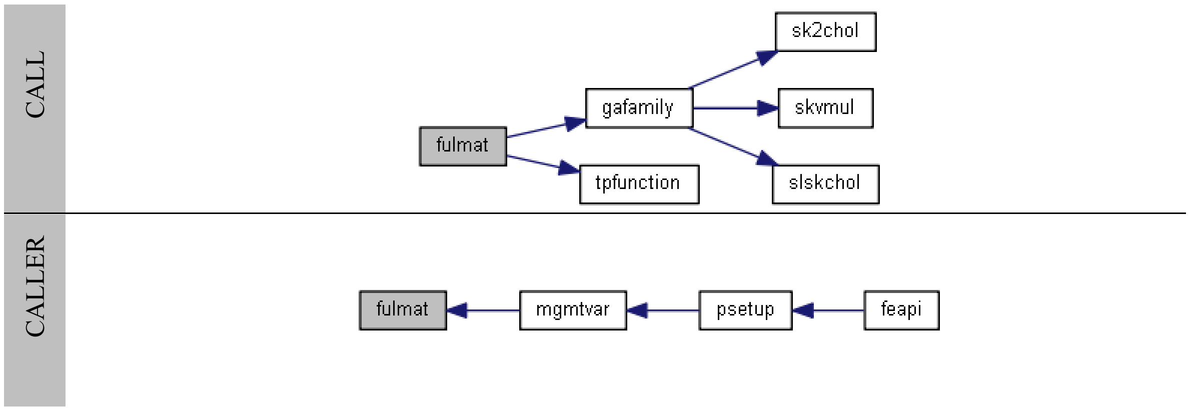




C. fulmat.f90

```
! This sub-routine computes the sub-domain unit load response matrix.
SUBROUTINE fulmat(db)
USE precision
USE interfaces
USE data_mgmt
USE data_domain
USE data_transient
USE data_structural
IMPLICIT NONE
! - Arguments.
! Domain block.
INTEGER, INTENT(IN)   :: db

! :
! :
RETURN
```

```
END SUBROUTINE fulmat
```



D. immat.f90

```
! This sub-routine computes the interface element mass matrix.
SUBROUTINE immat(imm,fun1,fun2,ndof,nodof)
USE precision
IMPLICIT NONE
! - Arguments.
! Shape functions (Interface of Lagrange multipliers).
REAL(sdp),  INTENT(IN)  :: fun1(:)
! Shape functions (Domain interface).
REAL(sdp),  INTENT(IN)  :: fun2(:)
! Number of Degrees of freedom per node.
INTEGER,    INTENT(IN)  :: nodof
! Number of Degrees of freedom per element.
INTEGER,    INTENT(IN)  :: ndof
! Interface element mass matrix.
REAL(sdp),  INTENT(OUT) :: imm(:,:)
! :
! :
RETURN
END SUBROUTINE immat
```

E. invmap.f90

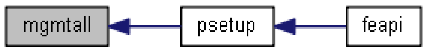
```
! This sub-routine performs the inverse mapping of global coordinates
! to local coordinates (Bilinear isoparametric quadrilateral elements only).
SUBROUTINE invmap(pointsl,pointsg,coord)
USE precision
IMPLICIT NONE
! - Arguments.
! Element coordinates.
REAL(sdp),  INTENT(IN)  :: coord(:,:)
! Global points and coordinates.
REAL(sdp),  INTENT(IN)  :: pointsg(:,:)
! Local points and coordinates.
REAL(sdp),  ALLOCATABLE,  INTENT(OUT) :: pointsl(:,:)
! :
! :
RETURN
END SUBROUTINE invmap
```

F. mgmtall.f90

```
! This sub-routine is used to allocate MGMT variables.
SUBROUTINE mgmtall()
USE precision
USE data_mgmt
USE data_feapi
USE data_domain
IMPLICIT NONE
! :
! :
RETURN
END SUBROUTINE mgmtall
```

CALL

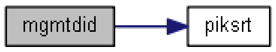
CALLER



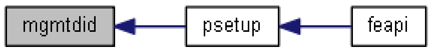
G. mgmtdid.f90

```
! This sub-routine is used to setup MGMT sub-domain hierarchy.
SUBROUTINE mgmtdid()
USE interfaces
USE data_mgmt
USE data_feapi
USE data_transient
IMPLICIT NONE
! :
! :
RETURN
END SUBROUTINE mgmtdid
```

CALL



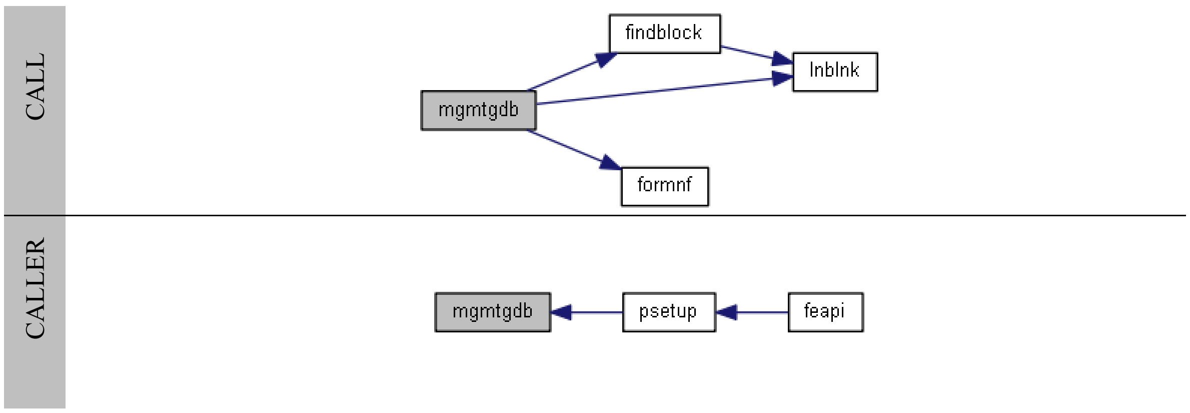
CALLER



H. mgmtgdb.f90

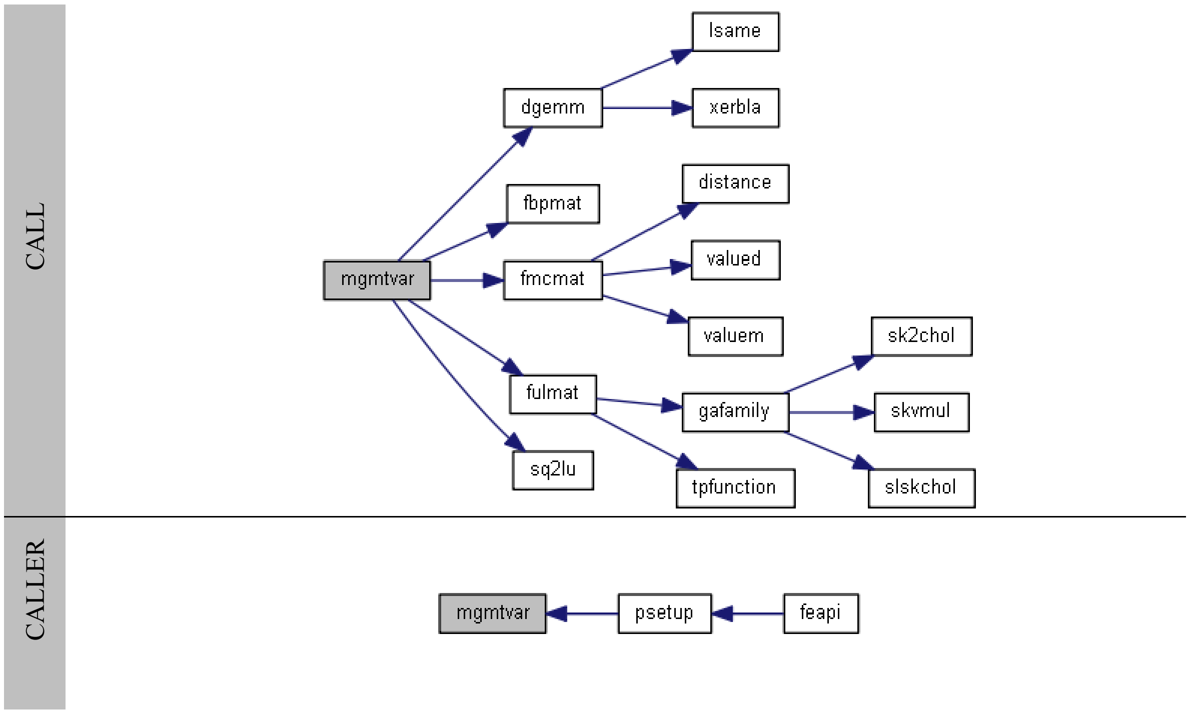
```
! This sub-routine forms the MGMT global domain block.
```

```
SUBROUTINE mgmtgdb()
USE interfaces
USE data_mgmt
USE data_post
USE data_feapi
USE data_domain
USE data_transient
USE data_structural
IMPLICIT NONE
! :
! :
RETURN
END SUBROUTINE mgmtgdb
```



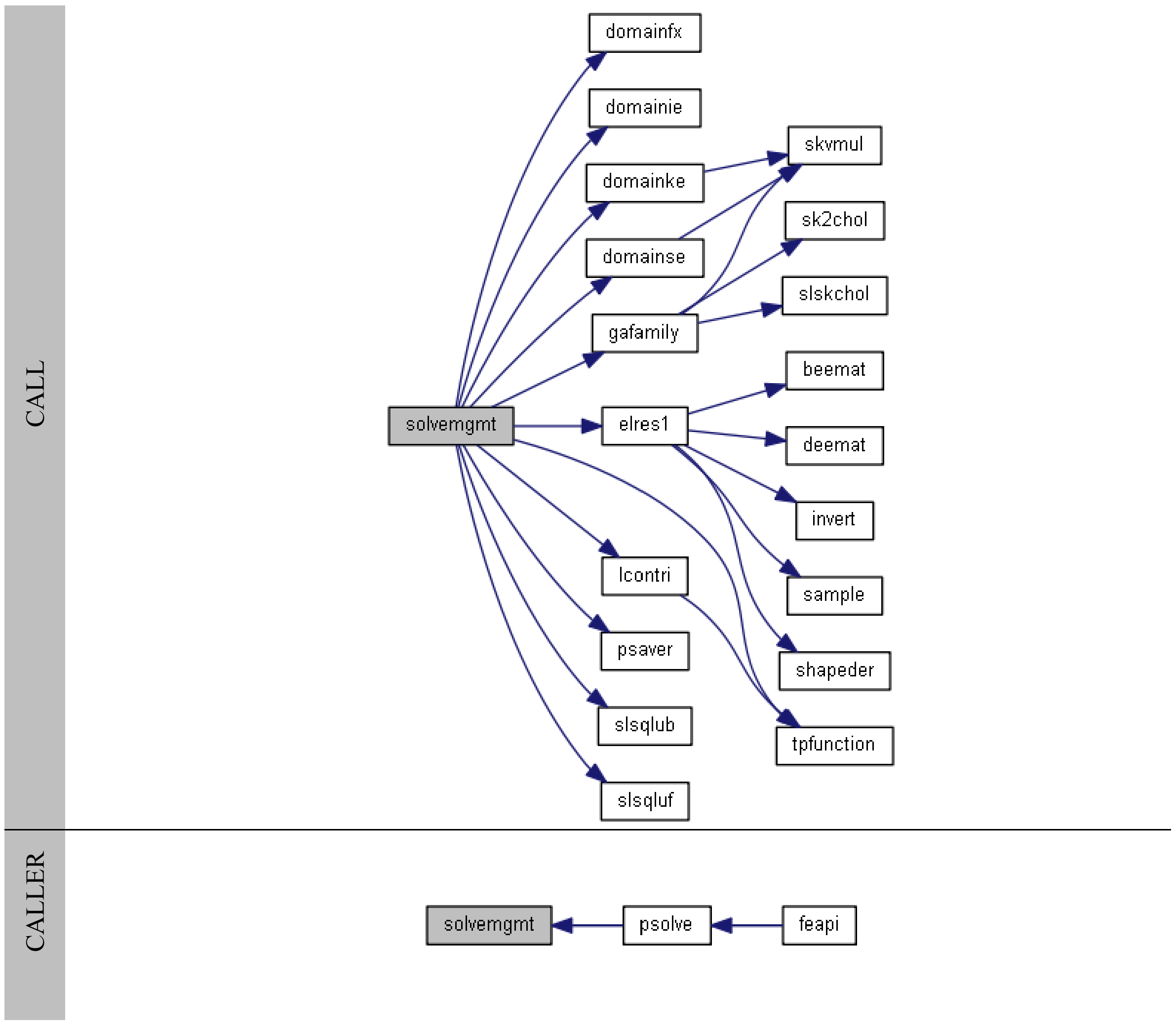
I. mgmtvar.f90

```
! This sub-routine is used to form MGMT variables.
SUBROUTINE mgmtvar()
USE precision
USE interfaces
USE data_mgmt
USE data_feapi
USE data_domain
IMPLICIT NONE
! :
! :
RETURN
END SUBROUTINE mgmtvar
```



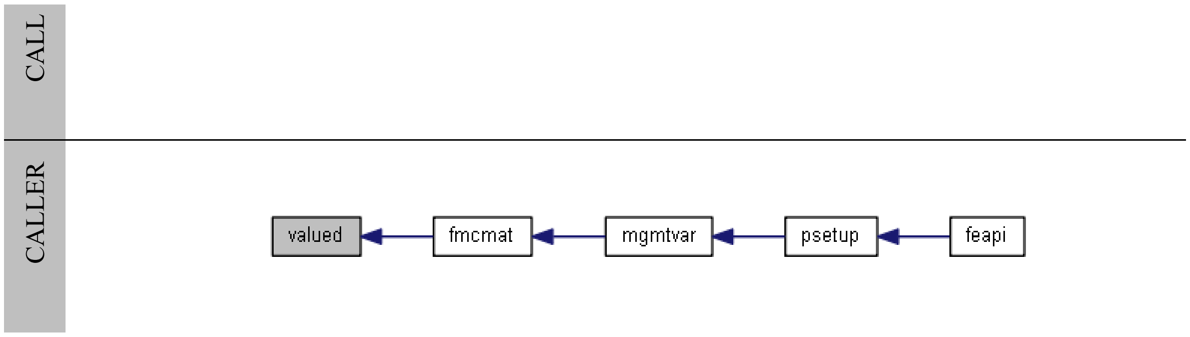
J. solvemgmt.f90

```fortran
! This sub-routine solves coupled MGMT equations.
SUBROUTINE solvemgmt()
USE precision
USE interfaces
USE data_feapi
USE data_domain
USE data_transient
USE data_structural
USE data_post
USE data_mgmt
IMPLICIT NONE
! :
! :
RETURN
END SUBROUTINE solvemgmt
```



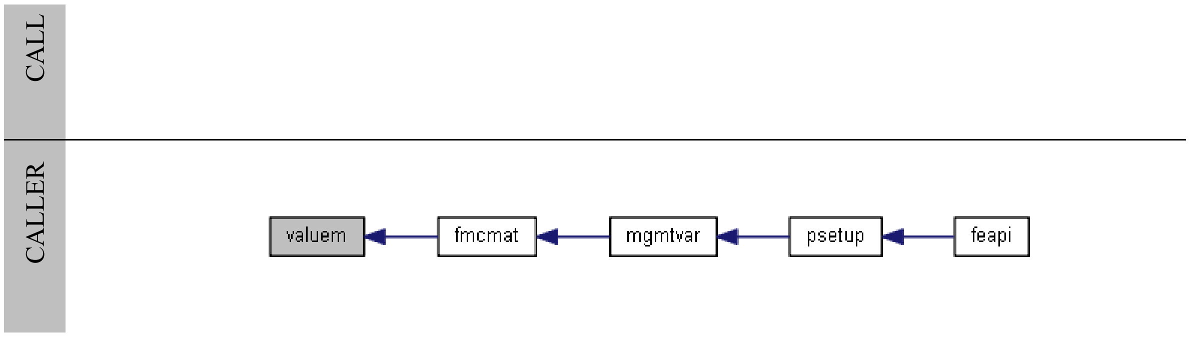
K. valued.f90

```
! This sub-routine returns the value of shape functions on the sun-domain
! interface.
SUBROUTINE valued(x,n,ellen,value)
USE precision
IMPLICIT NONE
! - Arguments
! Location of integration marker over the length.
REAL(sdp),  INTENT(IN)  :: x
! Array of mortar element lengths.
REAL(sdp),  INTENT(IN)  :: ellen(:)
! Shape function locator.
INTEGER,    INTENT(IN)  :: n
! Shape function value.
REAL(sdp),  INTENT(OUT) :: value
! :
! :
RETURN
END SUBROUTINE valued
```



L. valuem.f90

```
! This sub-routine returns the value of shape functions on the mortar
! interface.
SUBROUTINE valuem(x,s,ellen,value)
USE precision
IMPLICIT NONE
! - Arguments
! Location of integration marker over the length.
REAL(sdp),   INTENT(IN)   :: x
! Array of mortar element lengths.
REAL(sdp),   INTENT(IN)   :: ellen(:)
! Shape function locator.
INTEGER,     INTENT(IN)   :: s
! Shape function value.
REAL(sdp),   INTENT(OUT) :: value
! :
! :
RETURN
END SUBROUTINE valuem
```


CALL
CALLER
valuem
fmcmat
mgmtvar
psetup
feapi