Imperial College of Science, Technology and Medicine
Department of Mechanical Engineering

Parallel Harmonic Balance Method for Analysis of Nonlinear Mechanical Systems

Jiří Blahoš

October 2022

Submitted in part fulfilment of the requirements for the degree of PhD.

Originality statement

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Copyright declaration

Blahoš Jiří

October 2022

Abstract

Mechanical vibration analysis and modelling are essential tools used in the design of various mechanical components and structures. In the case of turbine engine design specifically, the ability to accurately predict vibration of various parts is crucial to ensure their safe operation while maintaining efficiency. As the designs become increasingly complex and margins for errors get smaller, high fidelity numerical vibration models are necessary for their analysis. Research of parallel algorithms has progressed significantly in the last decades, thanks to the exponential growth of the world's available computational resources. This work explores the possibilities for parallel implementations for solving large scale nonlinear vibration problems. A C++ code using MPI was developed to validate these implementations in practice. The harmonic balance method is used in combination with finite elements discretisation and applied to an elastic body with the Green-Lagrange nonlinear model for large deformations. A parameter continuation scheme using a predictor-corrector approach is included to compute frequency response functions. A Newton-Raphson solver is used to solve the bordered nonlinear system of equations in the frequency domain. Three different parallel algorithms for solving the linearised problem in each Newton iteration are analysed - a sparse direct solver (using MUMPS library), GMRES (using PETSc library) and an inhouse implementation of FETI. The performance of the solvers is analysed using beam testcases and a fan blade geometry. Scalability of MUMPS and the FETI solver is assessed. Full nonlinear frequency response functions with turning points are also computed. Use of artificial coarse space and preconditioning in FETI is discussed as it greatly impacts convergence properties of the solver. The presented parallel linear solvers show promising scalability results and an ability to solve nonlinear systems of several million degrees of freedom.

Keywords: nonlinear vibration analysis, distributed geometric nonlinearity, harmonic balance, parameter continuation, frequency response function, parallel solver, MUMPS, FETI, GMRES, domain decomposition, large scale model

Acknowledgements

Throughout the writing of this dissertation and during the years of my PhD, I have been supported and guided by the following people which I want to thank.

First and foremost I want to thank my parents for their love and for guiding and supporting me through my entire life. I also want to thank all other family members for their love and encouragement in difficult times.

I want to express my gratitude to my supervisor Dr Loïc Salles. His support, experience and knowledge were invaluable for my navigation through many aspects of the entire PhD study.

I am also thankful to all my colleagues in the VUTC lab at Imperial College London for their professional help. Special thanks goes to Dr Alessandra Vizzaccaro and Dr Fadi El Haddad for their numerous help during my study.

I also want to thank the EXPERTISE [172] training network for providing funding and connections with many people and institutions working in this field.

Special thanks is dedicated to Professor Daniel J. Rixen from Technical University of Munich for his time and kindness. His knowledge significantly contributed to this work.

I am grateful to IT4Innovations for opportunity for secondment and providing their computational resources. Special thanks goes to Dr Tomáš Brzobohatý for his time and shared knowledge.

There are no accidents.

Master Oogway

Contents

Originality statement **i**

Copyright declaration **iii**

Abstract **v**

Acknowledgements **vii**

1 Introduction **1**

1.1 Motivation 1

1.2 EXPERTISE network 4

1.3 Objectives 4

1.4 Outline 5

2 Physics and discretisation **7**

2.1 State of the art 7

2.2 Problem physics 8

2.2.1 Linear deformations 9

2.2.2 Nonlinear deformations 11

2.2.3 Boundary conditions 12

2.3 Finite elements 12

2.3.1 Boundary conditions 17

2.3.2 Evaluating the mesh integrals 17

2.3.3 C tensor, symmetries and Voigt notation 22

2.4 Conclusion 25

3 Vibration modelling **26**

3.1 State of the art 26

3.1.1 Harmonic balance and continuation 27

3.1.2 Reduced order modelling 28

3.1.3 Linear and nonlinear solvers 29

3.2 Harmonic Balance Method 32

3.3 Newton-Raphson 36

3.4 Alternating Frequency Time (AFT) 38

3.5 Boundary conditions 40

3.6 Continuation 41

3.7 Linear analysis and modes 45

3.8 Conclusion 48

4 FETI for HBM **50**

4.1 State of the art 50

4.1.1 Parallel linear solvers 51

4.1.2 Domain decomposition methods 51

4.1.3 FETI method 52

4.1.4 Parallel HBM 54

4.2 Domain decomposition of the nonlinear HBM problem 55

4.3 Treatment of corners 60

4.4 Linearised problem - Newton step 61

4.5 Solving the linearised problem 66

4.6 Preconditioning 71

4.7 Choice of artificial coarse space 73

4.8 Continuation 75

4.9 Conclusion 77

5 Code **79**

5.1 Structure 80

5.1.1 Input and output 83

5.2 Linear algebra 85

5.2.1 DOF ordering 88

5.2.2 Global solvers 89

5.2.3 Notes about MUMPS 91

5.3 FETI implementation 93

5.3.1 Primal and dual maps 93

5.3.2 Application of the dual problem operator 96

5.3.3 Projectors 98

5.3.4 Convergence 103

5.4 Conclusion 104

6 Results **105**

6.1 Hardware overview 106

6.2 Testcase overview 106

6.2.1 Note on parameters 107

6.3 Global solvers results 109

6.3.1 Code verification 109

6.3.2 Scalability 112

6.3.3 Blade FRC 118

6.3.4 GMRES performance on global system 121

6.4 FETI solver results 127

6.4.1 Coarse space and preconditioner effect 127

6.4.2 Scalability 137

6.4.3 Projected dual problem equations 153

6.4.4 Frequency Response Curve 154

6.5 Conclusion 161

7 Conclusion **163**

7.1 Research work conclusion 163

7.1.1 Global system with direct solver 163

7.1.2 Global system with GMRES 163

7.1.3 FETI 164

7.2 Scientific contributions 164

7.3 Future work 165

Bibliography **166**

A Cantilever beam response study **182**

A.0.1 Discussion 191

List of Tables

5.1 List of used 3rd party libraries 82

6.1 Salomon supercomputer information 106

6.2 Karolina supercomputer information 107

6.3 Clamped-clamped beam testcase parameters 109

6.4 Clamped-clamped beam testcase numerical parameters 110

6.5 Cantilever beam verification testcase parameters 111

6.6 Clamped-clamped beam testcase numerical parameters 111

6.7 Clamped-clamped beam scaled testcase parameters 113

6.8 Small scale scalability testcase parameters 113

6.9 MUMPS scalability runtime 116

6.10 Blade FRC testcase parameters 118

6.11 Blade FRCs computation times 120

6.12 First weak scalability test dimensions 138

6.13 Second weak scalability test dimensions 147

6.14 First strong scalability test dimensions 148

6.15 FETI FRC testcase numerical parameters 155

A.1 Cantilever beam continuation numerical parameters 183

List of Figures

1.1 Diagram of analysis for a bladed disk 3

2.1 Mesh example 13
2.2 HEXA20 element 18

3.1 Pendulum vibration 27
3.2 FRC example 29
3.3 Nonzero stiffness matrix pattern example 31
3.4 Periodic harmonic motion example 33
3.5 Diagram of AFT for the nonlinear force vector 39
3.6 Diagram of AFT for the nonlinear Jacobian matrix 40
3.7 Example of nonlinear FRC for clamped-clamped beam 42
3.8 Diagram of one continuation step 43
3.9 Clamped-clamped beam testcase 45
3.10 Linear FRC for different damping values 47
3.11 Beam linear FRC example 48

4.1 Types of FETI 53
4.2 Espreso (IT4I) scalability 54

4.3 FETI diagram 58

4.4 Effect of matrix B and B^T 59

4.5 Multiple domains interfacing in a corner with fully redundant constraints. . 61

4.6 Translation and rotation of neighbouring domains 75

4.7 Construction of interface rotations for artificial coarse space 76

5.1 Code flow diagram 80

5.2 Code abstractions and implementations 81

5.3 CSR matrix storage example 85

5.4 Example of index distribution (nonoverlapping) 86

5.5 Example of index distribution (nonoverlapping) 86

5.6 Global matrix distribution 87

5.7 Example of imports between distributions 88

5.8 DOF ordering scheme 89

5.9 Mesh distribution and global map 90

5.10 Mesh distribution and FETI primal maps 94

5.11 Dual variable distribution in FETI 95

5.12 Steps in applying the $\mathcal{F}$ operator 97

5.13 FETI natural coarse space required communication 100

5.14 FETI dual vector nonzero pattern overlap 100

5.15 FETI artificial coarse space required communication 102

6.1 Clamped-clamped beam testcase 107

6.2 Cantilever beam testcase 107

6.3 Blade testcase 108

6.4 Clamped-clamped beam verification testcase FRC 110

6.5 Cantilever beam first mode 111

6.6 Cantilever beam verification testcase FRC 111

6.7 Small scale scalability - runtime, setup 1 114

6.8 Small scale scalability - efficiency, setup 1 114

6.9 Small scale scalability - runtime, setup 2 115

6.10 Small scale scalability - efficiency, setup 2 115

6.11 MUMPS scalability runtime 117

6.12 MUMPS scalability speed-up 118

6.13 Blade FRCs for various forcing amplitudes 119

6.14 Blade FRCs for H0123 and H012345 120

6.15 Blade FRCs for H0123 and H012345 (detail) 121

6.16 Scaled clamped-clamped beam nonlinear FRC 122

6.17 Global system GMRES solve iteration count 123

6.18 Global system GMRES solve residuals for 128 MPIs 124

6.19 Global system GMRES solve runtimes 124

6.20 Global system GMRES nonlinear solve GMRES residuals 125

6.21 Global system GMRES nonlinear solve GMRES residuals 125

6.22 Global system GMRES nonlinear solve Newton residuals 126

6.23 Global system GMRES nonlinear solve runtimes 126

6.24 FETI coarse space use comparison 129

6.25 FETI coarse space use comparison 130

6.26 FETI coarse space use comparison 131

6.27 FETI coarse space use comparison 132

6.28 FETI coarse space use comparison 132

6.29 FETI coarse space use comparison 134

6.30 FETI coarse space use comparison 135

6.31 FETI coarse space use comparison 136

6.32 FETI coarse space use comparison 136

6.33 Scalability results - timing 140

6.34 Scalability results - GMRES convergence 141

6.35 Scalability results - timing 142

6.36 Scalability results - GMRES convergence 143

6.37 Scalability results - timing 144

6.38 Scalability results - GMRES convergence 145

6.39 Scalability results - GMRES tolerance comparison 146

6.40 Scalability results - decomposition mesh comparison 147

6.41 Scalability results - decomposition comparison 148

6.42 Scalability results - timing 149

6.43 Scalability results - speed-up 150

6.44 Scalability results - GMRES iteration count 151

6.45 Scalability results - domain connectivity 152

6.46 FETI comparison of dual problem variants 154

6.47 FETI FRCs for H0123 GMRES tolerance comparison 156

6.48 FETI FRC amplitudes comparison 157

6.49 FETI FRC harm 012345 detail 158

6.50 Direct solver reference for FRC harm 012345 158

6.51 FETI FRC GMRES iterations comparison 160

A.1 Cantilever beam testcase 182

A.2 Cantilever beam testcase excitation 183

A.3 Cantilever beam FRC 184

A.4 Cantilever beam FRC - zoomed 184

A.5 Cantilever beam FRC - zoomed 185

A.6 Cantilever beam FRC - zoomed 185

A.7 Cantilever beam FRC - high harmonics count 186

A.8 Cantilever beam modes 186

A.9 Cantilever beam FRC mode 2 H012 187

A.10 Cantilever beam FRC mode 2 H0123 188

A.11 Cantilever beam FRC mode 20 H0123 189

A.12 Cantilever beam FRC mode 20 H012345 190

Nomenclature

α, α_i	Vector of coefficients for null space vectors $\mathcal{R}$ in FETI (global, for domain *i*)
β	Vector of coefficients for FETI dual solution λ_1 on the artificial coarse space $\mathcal{G}_c$
ϵ	Small strain tensor
Γ_D	Part of boundary of Ω_0 where Dirichlet condition is prescribed
λ	Vector of Lagrange multipliers (dual variables) in FETI
λ, μ	Elasticity tensor constants for isotropic materials (only in section 2.3.3)
λ_0, λ_{12}, λ_1, λ_2	Parts of $\Delta\lambda$ solution in FETI dual problem
$\mathbb{R}$	Set of real numbers
$\mathcal{A}$, $\mathcal{A}_i$	FETI domain matrices (global, for domain *i*)
$\mathcal{A}^+$, $\mathcal{A}_i^+$	Pseudoinverse of $\mathcal{A}$, $\mathcal{A}_i$
$\mathcal{D}$	First part of right hand side vector in linearised dual problem equations in FETI
$\mathcal{E}$	Second part of right hand side vector in linearised dual problem equations in FETI
$\mathcal{F}$	FETI dual operator
$\mathcal{F}'$	Preconditioner for the FETI dual operator $\mathcal{F}$
$\mathcal{G}$, $\mathcal{G}_i$	Artificial coarse space vectors in dual variables in FETI (global, for domain *i*)
$\mathcal{G}$, $\mathcal{G}_i$	Null space vectors in dual variables in FETI (global, for domain *i*)
$\mathcal{L}_i$	FETI domain matrix left null space basis vectors in columns for domain *i*
$\mathcal{N}$	Number of domain in a mesh domain decomposition

$\mathcal{P}$	FETI orthogonal projector
$\mathcal{P}_c$	FETI conjugate projector
$\mathcal{Q}$	FETI orthogonal projector
$\mathcal{R}$, $\mathcal{R}_i$	FETI domain matrix right null space basis vectors in columns (global, for domain i)
$\mathcal{S}_i^{bb}$	Schur complement of $\mathcal{A}_i$ on the domain boundary
ν	Poisson's ratio
ω	Base frequency of vibration
Ω_0, Ω	Closed connected space on which the problem is defined in undeformed, deformed coordinate system
π	$3.14159265\ldots$
ρ_0, ρ	Material density in undeformed, deformed coordinate systems
σ	Small stress tensor
τ	Prediction vector in continuation procedure
$\tilde{F}$	Vector of external forces in frequency domain
$\tilde{F}_{int}^{nl}$	Vector of nonlinear part of internal forces in frequency domain
$\tilde{u}$	Vector of harmonic (Fourier) coefficients for displacements in frequency domain
$\tilde{u}_0$, $\tilde{u}^c$, $\tilde{u}^s$	Constant, cosine and sine components of $\tilde{u}$
$\tilde{u}_i$	Vector of harmonic (Fourier) coefficients for displacements in frequency domain on domain i (in parts discussing FETI)
a^c, a^s	Amplitudes of external force vector for its base frequency cosine and sine components
B, B_i	Matrix of domain interface constraints in FETI (global, for domain i)
$b_{k,n}^c$, $b_{k,n}^s$	Values of cosine and sine functions sampled at discrete time points
B_i^b	Boundary (interface) portion (column wise) of a B_i matrix

C	Elasticity tensor (4th order)
D	Damping matrix
dot	Standard vector dot product
E	Green-Lagrange strain tensor
E_Y	Young's modulus
F	Vector of external forces in time domain
f_0, f	External force field in undeformed, deformed coordinate systems
F_D	Deformation gradient
F_{int}	Vector of internal forces in time domain
F_{int}^{nl}	Vector of nonlinear part of internal forces in time domain
G	General set of nonlinear algebraic equations, later used specifically for the HBM system
g	Additional constraint function in correction stage of continuation procedure
h	Used as a limit variable in certain equations
H, H_i	Function for HBM equations of motion in FETI (global, for domain i)
H_c	FETI domain interface constraint function
h_t	Length of time step in alternating frequency time procedure time sampling
K	Stiffness matrix (linear part)
K^{nl}	Stiffness matrix (nonlinear part)
L	Lagrangian for FETI formulation
M	Mass matrix
N^r, N	Scalar finite element shape function in reference, actual element coordinates
N_t	Number of discrete time points in alternating frequency time procedure time sampling

O	A zero vector or matrix of appropriate (fitting) size
$R(t)$	Residual of discrete equations of motion in time domain
r_M, r_K	Rayleigh damping coefficients for mass and stiffness matrix respectively
r_T, r_R	Translational and rotational parts of rigid body modes of domain linear stiffness matrix K in FETI
S	Second Piola-Kirchhoff stress tensor
s	Size of prediction step in continuation procedure
T	One time period of vibration
t_n	Discrete time values in alternating frequency time procedure time sampling
u	Vector of discretised displacements (from section 2.3 onward)
u, U	Continuous displacement vector function
v	Weak formulation test function, finite element shape function
w	Weight coefficient for finite element Gauss quadrature rules
x	A generic variable
x^0, x	Undeformed, deformed spatial coordinate systems (in section 2.2)
Z	Dynamic stiffness matrix in HBM

Chapter 1

Introduction

1.1 Motivation

Vibration is a physical phenomenon naturally occurring in all rotatory mechanical systems as well as many other constructions or components experiencing periodic loading (bridges, buildings). When not properly controlled, excessive vibration can cause catastrophic failures leading to significant financial cost as well as serious health hazards. Even smaller magnitude vibration can cause damage to mechanical components over a long time span due to high cycle fatigue. Modelling and predicting occurrence of any and all vibration in industrial machines and constructions is therefore a necessary step in their engineering process.

Designs of new aero engines in aerospace industry are required to meet many different and sometimes conflicting criteria. Emphasis is placed on high fuel efficiency to reduce the operating cost as much as possible. Other aspects include complexity and cost of both manufacturing and subsequent maintenance, durability or compatibility with other components of the aircraft. In recent decades, a growing emphasis is also placed on the ecological friendliness of the engines. Finally, all these requirements need to be coupled with high robustness and reliability of the engines to ensure the maximum possible level of safety during their operation. Combining this entire set of requirements efficiently into the final design demands use of the most advanced mathematical modelling tools as well as the state of the art computing capabilities.

Analysis of vibration as a scientific field and its use in turbine engineering has many decades of history [30, 129]. Many approaches and models have been since developed. Harmonic balance method (HBM) is a common tool modelling approach for steady state harmonic motion, which makes it a popular method for vibration analysis [180]. While explicit time marching schemes have been successfully used for modelling nonlinear vibration [188], HBM was shown to be a more efficient tool for obtaining steady state responses [158]. Parameter continuation techniques and bifurcation detection algorithms have been developed to acquire proper understanding of character of the vibration across a range of operating frequencies [3]. Accurate models of various nonlinearities (such as large deformations or contact) are necessary to correctly predict vibration responses of complex industrial components [111]. Model order reduction techniques are often used to reduce the size of the analysed system to reduce computational cost [150]. Experimental measurement of vibration is then important to validate any new proposed modelling technique [42]. Generally, a complete vibrational analysis of a turbine and its components is a complex procedure with many steps and employing various algorithms [162]. A diagram of vibrational analysis of a bladed disk can be seen in 1.1.

While reduced order models are very useful vibration analysis tools, they inherently introduce certain assumptions into the model, which might reduce its accuracy or scope of applicability. Additionally, their validation relies on accurate full model results. This means that modelling of full nonlinear vibration systems remains an essential component of the analysis. Thanks to rapid development of microchip technology and utilisation of massively parallel computer systems in the recent decades large scale computer simulations can be now executed in previously unthinkable times. Combining the state of the art vibration analysis tools with modern parallel programming techniques can lead to significant improvements in terms of accuracy and level of detail of the models. This level of detail is necessary for further refinement of turbine designs while maintaining high safety and reliability standards. Such demand requires the ability to process meshes with millions of nodes. Models of this size are commonly used in industry for linear modal analysis and computation of linear vibrational responses [74]. However, the computational requirements are much higher for models with distributed nonlinearities. While some attempts to solve large scale nonlinear mechanical vibration problems with HBM have been made [113], the number of degrees of freedom is limited to hundreds of thousands at most. This work aims to push this limit further.

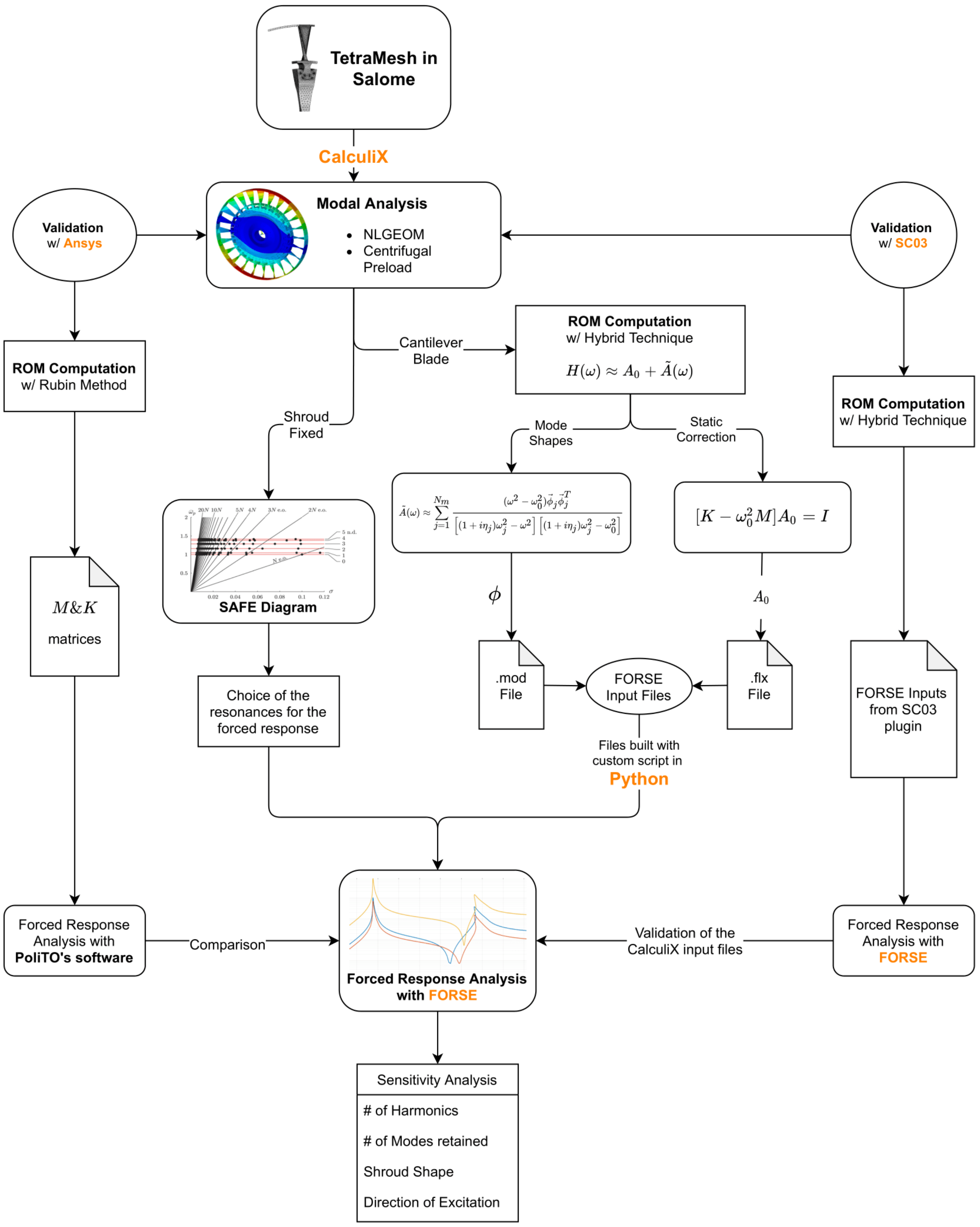

TetraMesh in Salome
CalculiX
Modal Analysis
NLGEOM
Centrifugal Preload
Validation w/ Ansys
Validation w/ SC03
Cantilever Blade
ROM Computation w/ Hybrid Technique
$H(\omega) \approx A_0 + \tilde{A}(\omega)$
ROM Computation w/ Rubin Method
Shroud Fixed
Mode Shapes
Static Correction
ROM Computation w/ Hybrid Technique
$\tilde{A}(\omega) \approx \sum_{j=1}^{N_m} \frac{(\omega^2 - \omega_0^2)\vec{\phi}_j\vec{\phi}_j^T}{\left[(1+i\eta_j)\omega_j^2 - \omega^2\right]\left[(1+i\eta_j)\omega_j^2 - \omega_0^2\right]}$
$[K - \omega_0^2 M]A_0 = I$
SAFE Diagram
$M\&K$ matrices
ϕ
A_0
.mod File
FORSE Input Files
.flx File
FORSE Inputs from SC03 plugin
Choice of the resonances for the forced response
Files built with custom script in Python
Forced Response Analysis with PoliTO's software
Comparison
Forced Response Analysis with FORSE
Validation of the CalculiX input files
Forced Response Analysis with FORSE
Sensitivity Analysis
# of Harmonics
# of Modes retained
Shroud Shape
Direction of Excitation

Figure 1.1: Diagram of analysis for a bladed disk taken from [162].

1.2 EXPERTISE network

The project presented here was a part of a European Training Network (ETN) called EXPERTISE (models, EXperiments and high PERformance computing for Turbine mechanical Integrity and Structural dynamics in Europe) [172]. The goal of this network was to connect researchers across Europe in a joint effort towards developing advanced tools for dynamics analysis of large-scale models of turbine components. An important aspect of this network was collaboration and training of students in various engineering areas related to nonlinear structural dynamics of turbomachinery and high performance computing (HPC). Four work packages (WPs) were established employing together 15 Ph.D. students. WP1 research focused on understanding of the physics of friction contacts and developing advanced contact models. WP2 work was aimed at experimental identification of contact interfaces. WP3, where this particular project was listed, had a goal to develop efficient software analysis tool for highly refined finite element models for nonlinear vibration problems. WP4 then focused on the computing problematic, especially on data dependency, task parallelism and high level abstractions of I/O operations in parallel computing.

The collaboration within the EXPERTISE network contributed greatly to the results presented in this work. In particular, extensive discussions took place with the IT4Innovations research institute in the Czech Republic and the Technical University Munich. Their knowledge regarding the domain decomposition methods and parallelisation techniques were invaluable for this work.

1.3 Objectives

Recent development in the field of parallel computing and algorithms opens new possibilities for large scale modelling of nonlinear mechanical vibration problems. Various divide and conquer algorithms have been successfully used to obtain significant computational speed-ups in related fields, such as static and transient structural analysis, acoustics or electromagnetism. The main objective of this work is to explore options to efficiently parallelise solving process of full large scale nonlinear vibration models in mechanics for distributed memory architectures.

A parallel C++ code was developed as a part of this project. This code implements algorithms described in this work so that they can be tested and evaluated on the current top of the line computer hardware. The code takes advantage of several existing software libraries for parallel computing. They offer efficient implementations of various complex functionalities required for numerical modelling, such as meshing, discretisation, interprocess communication, linear algebra or linear solvers. The code is used to compute nonlinear vibration responses on several testcases. The goal is to study the capabilities of the proposed parallelisation techniques as well as reveal directions where further research is required.

FETI method, a parallel algorithm for solving linear systems of equations based on domain decomposition approach, has been shown to be very efficient for large scale systems. This work applies the method to solve large scale vibration systems to study its suitability for this type of problems. The formulation of the method is adapted to better fit the harmonic balance equations and its performance is analysed in depth.

1.4 Outline

The content of this thesis is divided into seven chapters, each covering one aspect of the studied problematic. Following the introduction, the second chapter lays out the underlying physical and discrete models used for this work. First, the state of the art in the fields of continuum mechanics and finite elements is overviewed. Next the equations of motion for a continuous solid are established in both their strong and weak form, including definition of the nonlinearities. Boundary conditions are also addressed. Finally, the finite element scheme used to discretise the equations of motion is described.

The third chapter introduces the harmonic balance method (HBM) as the tool used to model vibration. Again, the state of the art regarding this method and topics closely related to it is first overviewed. Application of HBM on the discrete equations of motion obtained from the second chapter is then described in detail. The equations are transformed into their frequency domain form. A standard Newton-Raphson method is then introduced to solve those nonlinear equations. The alternating frequency time and parameter continuation procedures are described to allow for computing a nonlinear vibrational response over a range of frequencies. The boundary conditions are addressed again and one section is dedicated to linear modal analysis.

The fourth chapter presents the FETI method and its application to the nonlinear vibrational system arising from HBM. State of the art of this method as well as of parallel solvers in general is first discussed. Next, the application of the FETI method approach to solving the nonlinear HBM equations is described in detail. The FETI form of the HBM equations is established. The algorithm is adapted to fit the indefinite and nonsymmetric character of the problem. Preconditioning technique is also discussed and an artificial coarse space is introduced to improve convergence of the iterative solver. Finally, the extended bordered system is defined for a parameter continuation procedure.

The fifth chapter covers implementation of the theory and algorithms that were presented in the previous three chapters. Basic characteristics of the developed code are laid out. The chapter describes the structure of the code, 3rd party libraries that were used and discusses their properties. Special focus is placed on the parallelisation aspect of the code, highlighting its implementation details. In particular, implementation of the FETI method is discussed extensively.

The sixth chapter provides practical results that were achieved with the implemented code.

First, overview of used hardware and testcases is provided. Comparisons of results from the code to results of other researchers are then presented for verification purposes. Next, the chapter presents results for three different parallel solvers - MUMPS, PETSc GMRES and an inhouse FETI implementation. Single frequency responses as well as responses for full frequency ranges are computed using large nonlinear vibration models. The solvers are analysed in terms of their convergence properties, memory efficiency and parallel scalability. A discussion is provided for each result.

Lastly, the results of this work are discussed and conclusions are presented. Various performance aspects of each solver are assessed. Scientific contributions of the work are suggested, as well as an outlook into possible future research of the topic.

Chapter 2

Physics and discretisation

This chapter provides an overview of the basic theoretical background in mathematics and physics used in mechanical analysis that is relevant for this work. First, the mathematical framework describing the physics of motion of elastic bodies undergoing deformation is laid out. This provides the connecting point with the physical phenomenon of solid mechanics and mechanical vibration in particular. Second, discretisation of the established equations of motion using a well known finite element technique is described. This transfers the physical motion from the realm of partial differential equations defined on a space continuum into the realm of ordinary differential equations defined for a set of spatially discrete variables.

2.1 State of the art

When modelling vibration (or any general motion) of continuous solids, their underlying material model needs to be defined first. There are many types and ways to classify materials and their behaviour. A fundamental level study of material physics can be found in [149]. Hartley et al. [88] provided an overview regarding the topic of material deformation and its modelling. This work uses the isotropic elastic material model. This means no plastic deformations are assumed and the material properties don't vary with the direction of examination. Use of this model is valid for many engineering applications. Its linear version can be mathematically described as:

$$\begin{aligned} div(\sigma) + \rho_0 f_0 &= \rho_0 \ddot{u} \\ \sigma &= C\epsilon \\ \epsilon &= \frac{1}{2}(\nabla u + (\nabla u)^T) \end{aligned} \tag{2.1}$$

The above equations are discussed in detail in Section 2.2.1. Overview of the topic of linear elasticity can be found in [82, 167]. However, the linear model is not sufficient in many modelling scenarios as real vibrating structures often exhibit nonlinear behaviour, especially when the deformation becomes larger. Therefore, nonlinear elastic models need to be used instead. A comprehensive study on behaviour and modelling of nonlinear elastic materials can be found in [32].

Besides material properties, a common source of nonlinearity in mechanics is contact. A general overview of contact modelling for solids is in [145]. Research of techniques to model contact has been of utmost importance in turbomachinery engineering. A turbine engine consists of many moving parts which are in various forms of contact with one another. Accurate modelling of those contact points is necessary to obtain accurate vibration response of the entire structure. Examples of studies regarding contact in bladed disks and how it affects their vibration can be found in [42, 68, 141, 143]. Rizvi et al. [154] provided a survey of dynamics experiencing dry friction damping.

When modelling solids it is often not feasible to model their physics analytically, especially in cases of complex geometries or boundary conditions. A discretisation scheme is used instead. The most common discretisation technique used in this field is the finite element method (FE or FEM). Its principle stands on weak formulation of the motion equations by introducing a space of test functions and then discretising this set of test functions along with discretisation of the solid into a mesh of elements and nodes. The weak form of the equations of motion with v being the test function looks as follows:

$$\int_{\Omega_0} div(\sigma) v d\Omega_0 + \rho_0 \int_{\Omega_0} f_0 v d\Omega_0 = \rho_0 \int_{\Omega_0} \ddot{u} v d\Omega_0 \tag{2.2}$$

See Sections 2.2.1 and 2.3 for details. Use of this method dates as far back as the 1940s [91] and it has since been used in many fields for numerical modelling of partial differential equations. The theory of this method as well as its practical use and computer implementations have been described in a wide range of literature [19, 20, 38, 119, 168].

2.2 Problem physics

To model vibration, one first needs to establish the physical framework describing a motion of a solid body. The most basic model based on continuum mechanics for solids is the so called linear elasticity model. This model assumes that the body deformation is linearly proportional to the magnitude of the deforming force. This assumption only holds up to a certain point in terms of magnitude of the deformation. When the deformation is too large, a more accurate model needs to be used to account for the nonlinearity. Both linear and nonlinear elastic models used for this work are described in the following sections.

2.2.1 Linear deformations

Assume a body covering an area (closed connected space) $\Omega^0 \subset \mathbb{R}^3$, which is in a rest (undeformed) state, and is described by a coordinate system x^0 in $\mathbb{R}^3$. When deformed, the body covers a new area $\bar{\Omega}$, described by a new coordinate system $x : \mathbb{R}^3 \to \mathbb{R}^3$ which maps points of the body from their undeformed location x^0 to their deformed location. The displacement field $u : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^3$ is then defined as:

$$u(x^0, t) = x(x^0, t) - x^0 \tag{2.3}$$

with t representing the time variable.

Spatial derivative of u with respect to the undeformed coordinate system is denoted as:

$$\nabla u = \frac{\partial u}{\partial x^0} = \begin{bmatrix} \frac{\partial u_1}{\partial x_1^0} & \frac{\partial u_1}{\partial x_2^0} & \frac{\partial u_1}{\partial x_3^0} \\ \frac{\partial u_2}{\partial x_1^0} & \frac{\partial u_2}{\partial x_2^0} & \frac{\partial u_2}{\partial x_3^0} \\ \frac{\partial u_3}{\partial x_1^0} & \frac{\partial u_3}{\partial x_2^0} & \frac{\partial u_3}{\partial x_3^0} \end{bmatrix} \tag{2.4}$$

with u_1, u_2, u_3 and x_1^0, x_2^0, x_3^0 denoting (only in this instance) first, second and third components of u and x^0 respectively. Deformation gradient is defined as:

$$F_D = \frac{\partial x}{\partial x^0} = I + \nabla u \tag{2.5}$$

When using the linear model, which is suitable as long as the deformations are small, the strain tensor $\epsilon : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^{3\times 3}$ can be defined as:

$$\epsilon = \frac{1}{2}(\nabla u + (\nabla u)^T) \tag{2.6}$$

with the arguments x^0 and t in ϵ and u omitted for better readability.

A variation of this strain is then:

$$\delta\epsilon = \frac{1}{2}(\nabla \delta u + (\nabla \delta u)^T) \tag{2.7}$$

Cauchy stress σ is used, and is coupled linearly with the strain tensor as:

$$\sigma = C\epsilon \tag{2.8}$$

where C is a fourth order tensor. In this work, it is assumed to be a tensor with constant coefficients. Its form will be discussed in more detail later (see Section 2.3.3).

The governing partial differential equation of linear elasticity is:

$$div(\sigma) + \rho_0 f_0 = \rho_0 \ddot{u} \tag{2.9}$$

with $f_0 = f_0(x^0, t) : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^3$ being the external body forces acting on the body (in the undeformed state). ρ_0 denotes the material density and is assumed to be constant in both space and time. The double dotted $\ddot{u}$ represents the second derivative of u with respect to the time variable.

As a prerequisite to the following finite element analysis, the equations of motion need to be transformed to their weak form [19]. This is done by multiplying them by a test function $v = \delta u$ and integrating over Ω_0, which yields:

$$\int_{\Omega_0} div(\sigma) v d\Omega_0 + \rho_0 \int_{\Omega_0} f v d\Omega_0 = \rho_0 \int_{\Omega_0} \ddot{u} v d\Omega_0 \tag{2.10}$$

with multiplication between functions on $\mathbb{R}^3$ (such as fv or $\ddot{u}v$) representing the standard vector dot product. Applying Green's identity to the integral involving stress σ and rearranging terms, the above yields:

$$\rho_0 \int_{\Omega_0} \ddot{u} v d\Omega_0 + \int_{\Omega_0} \sigma : \nabla v d\Omega_0 - \int_{\partial\Omega_0} v(\sigma n) dS = \rho_0 \int_{\Omega_0} f v d\Omega_0 \tag{2.11}$$

The : operator stands for double dot product between 2 second order tensors, i.e. $A : B = \sum_{i,j} A_{i,j} B_{i,j}$. The stress tensor σ is symmetric (this is discussed in more detail in Section 2.3.3), which implies that:

$$\begin{aligned} \sigma : \nabla v &= \sigma : sym(\nabla v) \\ &= \sigma : \frac{1}{2}(\nabla v + (\nabla v)^T) \\ &= \sigma : \delta\epsilon \end{aligned} \tag{2.12}$$

Assuming zero surface tension, i.e. eliminating the boundary integral, the final weak form for a linear elastic problem yields:

$$\rho_0 \int_{\Omega_0} \ddot{u} v d\Omega_0 + \int_{\Omega_0} \sigma : \delta\epsilon d\Omega_0 = \rho_0 \int_{\Omega_0} f v d\Omega_0 \tag{2.13}$$

The above represents the weak form of (2.9). The weak form is a more suitable formulation for solving of practical problems, as solving their strong form is in general not feasible.

2.2.2 Nonlinear deformations

When deformations are large, linear strain is no longer a suitable model. Instead, the Green-Lagrange strain tensor is used in this work to account for this nonlinearity:

$$E = \frac{1}{2}(\nabla u + (\nabla u)^T + (\nabla u)^T \nabla u) \tag{2.14}$$

A variation of the strain δE with respect to a particular test function v can also be determined:

$$\begin{aligned}
\delta E &= \lim_{h\to 0} \frac{E(u+hv) - E(u)}{h} \\
&= \frac{1}{2}\lim_{h\to 0} \frac{\nabla u + h\nabla v + (\nabla u + h\nabla v)^T + (\nabla u + h\nabla v)^T(\nabla u + h\nabla v) - \nabla u - (\nabla u)^T - (\nabla u)^T\nabla u}{h} \\
&= \frac{1}{2}\lim_{h\to 0} \frac{h\left[\nabla v + (\nabla v)^T + (\nabla v)^T\nabla u + (\nabla u)^T\nabla v\right] + h^2(\nabla v)^T\nabla v}{h} \\
&= \frac{1}{2}\left[\nabla v + (\nabla v)^T + (\nabla v)^T\nabla u + (\nabla u)^T\nabla v\right] + \underbrace{\frac{1}{2}\lim_{h\to 0} h(\nabla v)^T\nabla v}_{=0} \\
&= \frac{1}{2}\left[\nabla v + (\nabla v)^T + (\nabla v)^T\nabla u + (\nabla u)^T\nabla v\right]
\end{aligned} \tag{2.15}$$

The weak form from the linear case (2.13) still holds, but only in the deformed state:

$$\rho \int_\Omega \ddot{u} v d\Omega + \int_\Omega \sigma : \delta\epsilon d\Omega = \rho \int_\Omega f v d\Omega \tag{2.16}$$

with f and ρ representing the same quantities as f_0 and ρ_0 but in the deformed state. However, it can be shown that this can be rewritten into:

$$\rho_0 \int_{\Omega_0} \ddot{u} v d\Omega_0 + \int_{\Omega_0} S : \delta E d\Omega_0 = \rho_0 \int_{\Omega_0} f_0 v d\Omega_0 \tag{2.17}$$

where S stands for the second Piola-Kirchhoff stress. This formulation is entirely in the undeformed body state. Details of this transformation can be seen in [20]. Furthermore, assuming the material to be of the St Venant-Kirchhoff type, i.e. assuming small strains (not deformations), the stress strain relationship can be kept linear as in 2.8:

$$S = CE \tag{2.18}$$

2.2.3 Boundary conditions

There are three most commonly used types of boundary conditions that can be imposed on the boundary $\partial\Omega_0$ to further restrict a solution to a partial differential equation - Dirichlet, Neumann and Robin boundary condition.

The Dirichlet boundary condition prescribes specific values for the solution, i.e.:

$$u = a(x^0) \text{ on } \Gamma_D \subset \partial\Omega_0 \tag{2.19}$$

The Neumann condition prescribes value for the derivative of the solution in the direction of the surface's outer normal n:

$$\frac{\partial u}{\partial n} = (\nabla u)n = b(x^0) \text{ on } \Gamma_N \subset \partial\Omega_0 \tag{2.20}$$

However, in elasticity equations, this condition takes the form of prescribed traction in the normal direction:

$$\sigma n = b(x^0) \text{ on } \Gamma_N \subset \partial\Omega_0 \tag{2.21}$$

Note that this still involves derivative of u, given the equations (2.6) and (2.8). This version of the Neumann condition matches the terms obtained in the boundary integral in (2.11).

The last variant is the so called Robin boundary condition, which is a linear combination of the previous two:

$$\alpha u + \beta\frac{\partial u}{\partial n} = c(x^0) \text{ on } \Gamma_R \subset \partial\Omega_0 \tag{2.22}$$

This work will only employ the homogeneous Dirichlet boundary condition, i.e.: $u = 0$ on a given Γ_D. For the rest of the boundary, zero surface traction is assumed, in correspondence with the elimination of the boundary integral in (2.11). The particularities of enforcing the Dirichlet boundary condition will be discussed in later sections.

2.3 Finite elements

From this section forward, a new notation is adopted. The continuous vector field of displacements u will be denoted as U. u will instead represent the vector of displacements for a discrete set of nodes on a mesh.

Assume a discretisation of Ω_0 into a mesh of 3D elements labelled e, i.e. $\Omega_0 = \cup_i e_i$. These elements don't overlap and they are each determined by a set of nodes. For each

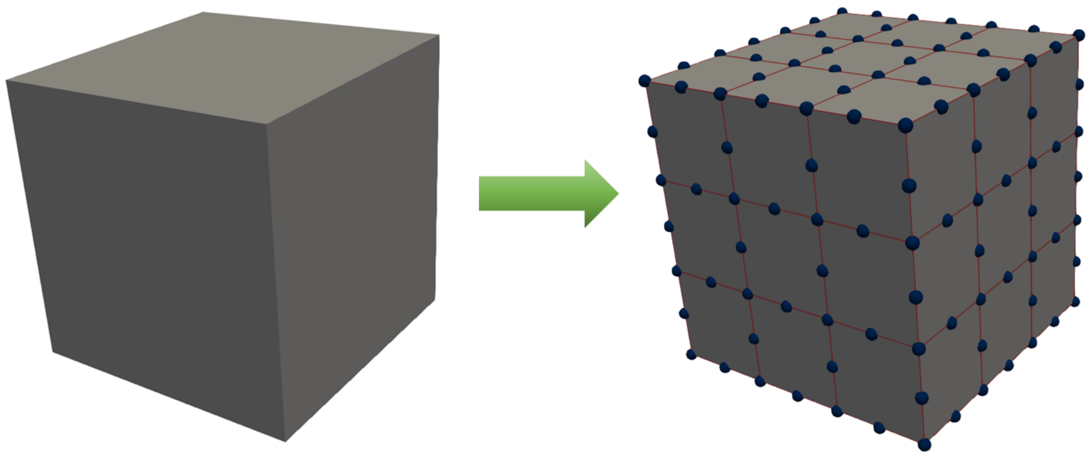


Figure 2.1: Example of a regularly meshed cube solid

of these nodes and each of the xyz coordinates, a single test function (whose restriction on a particular element is called a shape function for that element) v is assumed. The standard Lagrange interpolating polynomials are used in this work. This means that each of these functions is nonzero at exactly 1 of the 3 coordinates only, has value 1 at exactly one node and 0 at all the other ones. They are also zero everywhere outside the elements that coincide with the function's corresponding node. Let's index these functions v_i for i from 1 to $3 \times N_{nodes}$, or from 1 to N_{dof}, as there are 3 degrees of freedom assumed per node, one per each coordinate axis. The weak form equation (2.17) becomes a set of equations corresponding to the set of the test functions v.

The continuous solution field U is then expressed as a linear combination of these test functions:

$$U = \sum_i u_i v_i \tag{2.23}$$

The vector of the solution coefficients u_i is u. This vector is still a function of time, i.e. $u : \mathbb{R} \rightarrow \mathbb{R}^{N_{dof}}$. Let's have a closer look at the individual terms in (2.17). Starting with the inertial term:

$$\begin{aligned} I_1^j = \rho_0 \int_{\Omega_0} \ddot{U} v_j d\Omega_0 &= \rho_0 \int_{\Omega_0} \sum_i \left(\ddot{u}_i v_i\right) v_j d\Omega_0 \\ &= \rho_0 \sum_i \ddot{u}_i \int_{\Omega_0} v_i v_j d\Omega_0 \\ &= \rho_0 \sum_i \ddot{u}_i \sum_k \int_{e_k} v_i v_j d\Omega_0 \end{aligned} \tag{2.24}$$

it is not difficult to realise that this expression collectively over all values of j represents a matrix vector multiplication:

$$\begin{aligned}\begin{pmatrix} I_1^1 \\ I_1^2 \\ \vdots \\ I_1^{N_{dof}} \end{pmatrix} &= \rho_0 \begin{pmatrix} \sum_i \ddot{u}_i \sum_k \int_{e_k} v_i v_1 d\Omega_0 \\ \sum_i \ddot{u}_i \sum_k \int_{e_k} v_i v_2 d\Omega_0 \\ \vdots \\ \sum_i \ddot{u}_i \sum_k \int_{e_k} v_i v_{N_{dof}} d\Omega_0 \end{pmatrix} \\ &= M\ddot{u}\end{aligned} \tag{2.25}$$

where:

$$M = \rho_0 \sum_k \begin{bmatrix} \int_{e_k} v_1 v_1 d\Omega_0 & \int_{e_k} v_2 v_1 d\Omega_0 & \cdots & \int_{e_k} v_{N_{dof}} v_1 d\Omega_0 \\ \int_{e_k} v_1 v_2 d\Omega_0 & \int_{e_k} v_2 v_2 d\Omega_0 & & \int_{e_k} v_{N_{dof}} v_2 d\Omega_0 \\ \vdots & & \ddots & \vdots \\ \int_{e_k} v_1 v_{N_{dof}} d\Omega_0 & \int_{e_k} v_2 v_{N_{dof}} d\Omega_0 & \cdots & \int_{e_k} v_{N_{dof}} v_{N_{dof}} d\Omega_0 \end{bmatrix} \tag{2.26}$$

M is the mass matrix of the discretised system.

For the external forces term, one obtains:

$$I_2^j = \rho_0 \int_{\Omega_0} f_0 v_j d\Omega_0 = \rho_0 \sum_k \int_{e_k} f_0 v_j d\Omega_0 \tag{2.27}$$

This can be also summarised into the vector of external forces:

$$F = \begin{pmatrix} I_2^1 \\ I_2^2 \\ \vdots \\ I_2^{N_{dof}} \end{pmatrix} \tag{2.28}$$

Note that this vector is still a function of time.

Lastly, the internal forces:

$$\begin{aligned} I_3^j &= \int_{\Omega_0} S(E) : \delta E^j d\Omega_0 \\ &= \int_{\Omega_0} (CE) : \delta E^j d\Omega_0 \end{aligned} \tag{2.29}$$

with δE^j standing for the strain variation with respect to test function v_j. This integral over the entire range of j indices forms the vector of the internal forces inside the body. It is nonlinear in u, i.e.:

$$F_{int}(u) = \begin{pmatrix} I_3^1 \\ I_3^2 \\ \vdots \\ I_3^{N_{dof}} \end{pmatrix} \tag{2.30}$$

In the future sections, the derivative of this vector $\nabla_u F_{int}(u)$ will also be important. Let's for convenience denote P_L and P_R the left and right term around the C tensor in the integral (2.29):

$$\begin{aligned} P_L(u) = E(U) &= \frac{1}{2}\left[(\nabla U)^T \nabla U + \nabla U^T + \nabla U\right] \\ &= \frac{1}{2}\left[(\sum_i u_i (\nabla v_i)^T)(\sum_i u_i \nabla v_i) + \sum_i u_i (\nabla v^T + \nabla v)\right] \\ P_R^j(u) = \delta E^j(u) &= \frac{1}{2}\left[(\nabla U)^T \nabla v_j + (\nabla v_j)^T \nabla U + (\nabla v_j)^T + \nabla v_j\right] \\ &= \frac{1}{2}\left[\sum_i u_i (\nabla v_i)^T v_j + (\nabla v_j)^T \sum_i u_i \nabla v_i + (\nabla v_j)^T + \nabla v_j\right] \end{aligned} \tag{2.31}$$

The derivatives with respect to a particular element of u then yield:

$$\begin{aligned} \frac{\partial P_L(u)}{\partial u_k} &= \frac{1}{2}\left[(\nabla U)^T \nabla v_k + (\nabla v_k)^T \nabla U + (\nabla v_k)^T + \nabla v_k\right] \\ \frac{\partial P_R^j(u)}{\partial u_k} &= \frac{1}{2}\left[(\nabla v_k)^T \nabla v_j + (\nabla v_j)^T \nabla v_k\right] \end{aligned} \tag{2.32}$$

Note that $\frac{\partial P_L(u)}{\partial u_k} = P_R^k(u) = \delta E^k(u)$. The derivative of the integral (2.29) with a particular index j with respect to u_k is then:

$$\begin{aligned} \frac{\partial I_3^j}{\partial u_k} &= \int_{\Omega_0} \left[(CP_L) : \frac{\partial P_R^j}{\partial u_k} + \left(C \frac{\partial P_L}{\partial u_k}\right) : P_R^j\right] d\Omega_0 \\ &= \int_{\Omega_0} \left[\frac{1}{2}(CE) : ((\nabla v_k)^T \nabla v_j + (\nabla v_j)^T \nabla v_k) + \left(C\delta E^k\right) : \delta E^j\right] d\Omega_0 \end{aligned} \tag{2.33}$$

The first term in the sum inside the integral can be rearranged:

$$\begin{aligned}(CE) : ((\nabla v_k)^T \nabla v_j + (\nabla v_j)^T \nabla v_k) &= Tr((CE)^T (\nabla v_k)^T \nabla v_j) + Tr((CE)((\nabla v_j)^T \nabla v_k)^T) \\ &= Tr((CE)(\nabla v_k)^T \nabla v_j) + Tr((CE)(\nabla v_k)^T \nabla v_j) \\ &= 2Tr((CE)(\nabla v_k)^T \nabla v_j) \\ &= 2(\nabla v_k (CE)) : \nabla v_j\end{aligned} \tag{2.34}$$

This gives a new expression for (2.33):

$$\frac{\partial I_3^j}{\partial u_k} = \int_{\Omega_0} \left[(\nabla v_k (CE)) : \nabla v_j + \left(C\delta E^k \right) : \delta E^j \right] d\Omega_0 \tag{2.35}$$

It can be shown that the above expression is symmetric in terms of indices j and k. Furthermore, the term $\left(C\delta E^k\right) : \delta E^j$ contains, among others, term $\left(C((\nabla v_k)^T + \nabla v_k)\right) : ((\nabla v_j)^T + \nabla v_j)$ which is independent of u. It is the only constant part of this expression and can be, similarly to the mass matrix above, expressed as a stiffness matrix K. The entire derivative of the internal forces is then:

$$\nabla F_{int}(u) = \begin{bmatrix} \frac{\partial I_3^1}{\partial u_1} & \frac{\partial I_3^1}{\partial u_2} & \cdots & \frac{\partial I_3^1}{\partial u_{N_{dofn}}} \\ \frac{\partial I_3^2}{\partial u_1} & \frac{\partial I_3^2}{\partial u_2} & & \frac{\partial I_3^2}{\partial u_{N_{dofn}}} \\ \vdots & & \ddots & \vdots \\ \frac{\partial I_3^{N_{dofn}}}{\partial u_1} & \frac{\partial I_3^{N_{dofn}}}{\partial u_2} & \cdots & \frac{\partial I_3^{N_{dofn}}}{\partial u_{N_{dofn}}} \end{bmatrix} = K + \nabla F_{int}^{nl}(u) = K + K^{nl}(u) \tag{2.36}$$

with $K^{nl} = \nabla F_{int}^{nl}$ for convenience. Similarly, the internal forces vector F_{int} can be split into its linear and nonlinear part:

$$F_{int}(u) = Ku + F_{int}^{nl}(u) \tag{2.37}$$

The entire weak form (2.17), combined with the discretised displacement field (2.23), can be then expressed as a set of nonlinear differential equations with u and its time derivatives as the unknown function:

$$M\ddot{u} + F_{int}(u) = F(t) \tag{2.38}$$

Rayleigh damping is used to introduce motion damping into the system. The damping matrix D therefore takes the form:

$$D = r_M M + r_K K \tag{2.39}$$

With r_M and r_K being the Rayleigh damping coefficients.

The final form of the motion equations with the damping term included looks as follows:

$$M\ddot{u} + D\dot{u} + F_{int}(u) = F(t) \tag{2.40}$$

2.3.1 Boundary conditions

As mentioned previously, the only boundary condition used in this work is the homogeneous Dirichlet boundary condition, i.e. $u(t) = 0$ for $t >= 0$. This implies that also $\dot{u}(t) = 0$ and $\ddot{u}(t) = 0$. No forcing is imposed on the Dirichlet boundary degrees of freedom as those dofs are supposed to remain stationary, i.e. $F_i(t) = 0$ for i representing a Dirichlet boundary dof.

Starting with the equations of motion without damping (2.38), the traditional way to impose a zero Dirichlet boundary condition is to zero out the corresponding row and column in the M and K matrices, and inserting zero into the F_{int}^{nl} vector. Then, 1 is inserted into the diagonal of K for the given degree of freedom. This turns (2.38) into:

$$\begin{array}{cc} & \begin{array}{ccccc} & & \text{i-th column} & & \\ & & \downarrow & & \end{array} \\ & \\ \text{i-th row} \rightarrow & \begin{bmatrix} K' & \cdots & O & \cdots & K' \\ \vdots & & O & & \vdots \\ O & O & 1 & O & O \\ \vdots & & O & & \vdots \\ K' & \cdots & O & \cdots & K' \end{bmatrix} \begin{pmatrix} u' \\ \vdots \\ u_i \\ \vdots \\ u' \end{pmatrix} = \begin{pmatrix} F' \\ \vdots \\ 0 \\ \vdots \\ F' \end{pmatrix} \end{array} \tag{2.41}$$

with K', u' and F' symbolically representing all other parts of the given matrix/vector. This expression achieves $u_i = 0$ with u_i being a particular Dirichlet boundary degree of freedom.

When assembling the damping matrix D as described in (2.39), the mass matrix is used as is, i.e. including the zeroed out rows and columns. The stiffness matrix is used as is with the exception that the diagonal 1's are replaced by zeros as well. The derivative of nonlinear forces ∇F_{int}^{nl} is treated the same way, the Dirichlet rows and columns are zeroed out, including the diagonal elements.

2.3.2 Evaluating the mesh integrals

In order to assemble the discretised equations motion as defined in (2.38), the integrals I_1^j, I_2^j and I_3^j (described in (2.24), (2.27) and (2.29)), as well as derivatives of I_1^j and I_3^j (described in (2.25) and (2.35)), need to be evaluated. First, as it was hinted before, since the domain is decomposed into a set of non-overlapping elements that completely cover it, the integrals can be computed per-element, meaning $\int_{\Omega_0} = \sum_k \int_{e_k}$, with e_k standing for the mesh elements. The individual elements are commonly simple geometrical shapes. Tetrahedrons and hexahedrons are among the most used elements used in FE modelling. However, computing the integrals on the elements analytically is still not practical, especially for nonlinear terms. Instead, a numerical integration scheme is used.

In this work, Gauss quadrature rules are used. These rules define a value of an integral of a function over a volume by a sum of function values in given points, multiplied by certain coefficients:

$$\int_e f(x) \approx \sum_i w_i f(x_i) \tag{2.42}$$

The coefficients w are typically called weights. The positions of the points x_i and values of their corresponding weights w_i are determined in such a way so that the approximation above provides exact results for polynomials up to a certain order. This order determines the order of the quadrature rule. The higher the order the more accurate the approximation is but also the more points it contains.

Quadrature rules are defined on so called reference elements. A reference element is a uniquely defined element located in a reference coordinate axes x^r. An example of a 20 node hexahedron (HEXA20) reference element is in Figure 2.2.

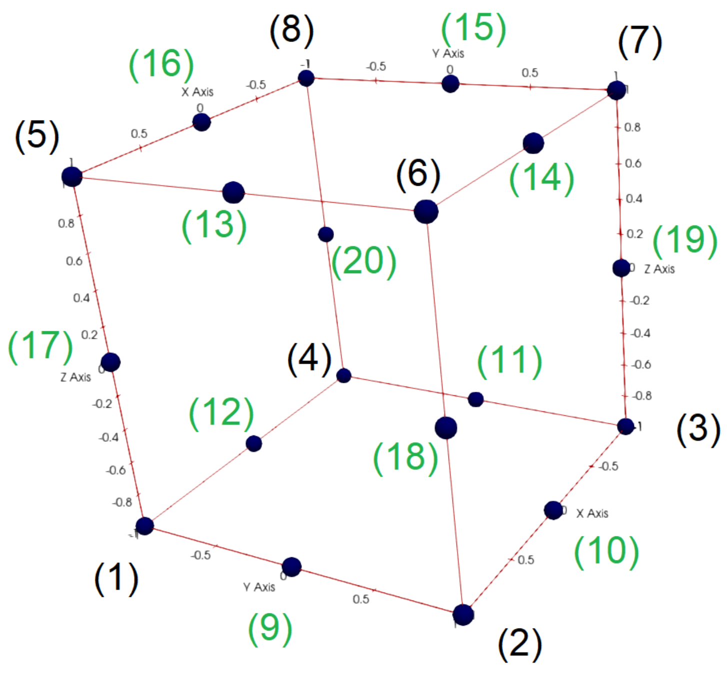


Figure 2.2: HEXA20 reference element with its conventional node numbering. First numbered are the 8 vertex nodes (black labels) and then the 12 mid-edge nodes (green labels).

Vertices of this element are located at coordinate values of ± 1. This fixes the element's size and positioning in the coordinate space, which also fixes the quadrature points x_i^r. The weight values are independent of the element shape and position.

$$\int_{e^r} f(x^r) \approx \sum_i w_i f(x_i^r) \tag{2.43}$$

Next, a set of reference shape functions in the reference coordinates $N^r(x^r)$ is defined for a particular reference element. These reference shape functions are defined as scalar functions, i.e. $N^r : \mathbb{R}^3 \rightarrow \mathbb{R}$. For the HEXA20 reference element, the reference shape functions are as follows:

$$
\begin{aligned}
N_1^r(x^r) &= \frac{1}{8}(1-x_1^r)(1-x_2^r)(1-x_3^r)(-2-x_1^r-x_2^r-x_3^r)\\
N_2^r(x^r) &= \frac{1}{8}(1+x_1^r)(1-x_2^r)(1-x_3^r)(-2+x_1^r-x_2^r-x_3^r)\\
N_3^r(x^r) &= \frac{1}{8}(1+x_1^r)(1+x_2^r)(1-x_3^r)(-2+x_1^r+x_2^r-x_3^r)\\
N_4^r(x^r) &= \frac{1}{8}(1-x_1^r)(1+x_2^r)(1-x_3^r)(-2-x_1^r+x_2^r-x_3^r)\\
N_5^r(x^r) &= \frac{1}{8}(1-x_1^r)(1-x_2^r)(1+x_3^r)(-2-x_1^r-x_2^r+x_3^r)\\
N_6^r(x^r) &= \frac{1}{8}(1+x_1^r)(1-x_2^r)(1+x_3^r)(-2+x_1^r-x_2^r+x_3^r)\\
N_7^r(x^r) &= \frac{1}{8}(1+x_1^r)(1+x_2^r)(1+x_3^r)(-2+x_1^r+x_2^r+x_3^r)\\
N_8^r(x^r) &= \frac{1}{8}(1-x_1^r)(1+x_2^r)(1+x_3^r)(-2-x_1^r+x_2^r+x_3^r)\\
N_9^r(x^r) &= \frac{1}{4}(1-(x_1^r)^2)(1-x_2^r)(1-x_3^r)\\
N_{10}^r(x^r) &= \frac{1}{4}(1+x_1^r)(1-(x_2^r)^2)(1-x_3^r)\\
N_{11}^r(x^r) &= \frac{1}{4}(1-(x_1^r)^2)(1+x_2^r)(1-x_3^r)\\
N_{12}^r(x^r) &= \frac{1}{4}(1-x_1^r)(1-(x_2^r)^2)(1-x_3^r)\\
N_{13}^r(x^r) &= \frac{1}{4}(1-x_1^r)(1-x_2^r)(1-(x_3^r)^2)\\
N_{14}^r(x^r) &= \frac{1}{4}(1+x_1^r)(1-x_2^r)(1-(x_3^r)^2)\\
N_{15}^r(x^r) &= \frac{1}{4}(1+x_1^r)(1+x_2^r)(1-(x_3^r)^2)\\
N_{16}^r(x^r) &= \frac{1}{4}(1-x_1^r)(1+x_2^r)(1-(x_3^r)^2)\\
N_{17}^r(x^r) &= \frac{1}{4}(1-(x_1^r)^2)(1-x_2^r)(1+x_3^r)\\
N_{18}^r(x^r) &= \frac{1}{4}(1+x_1^r)(1-(x_2^r)^2)(1+x_3^r)\\
N_{19}^r(x^r) &= \frac{1}{4}(1-(x_1^r)^2)(1+x_2^r)(1+x_3^r)\\
N_{20}^r(x^r) &= \frac{1}{4}(1-x_1^r)(1-(x_2^r)^2)(1+x_3^r)
\end{aligned}
\tag{2.44}
$$

The above functions can be collectively written into a vector N^r as:

$$
N^r = \begin{pmatrix} N_1^r \\ N_2^r \\ \vdots \end{pmatrix}
\tag{2.45}
$$

Let's define nodal coordinates of an actual mesh element as:

$$Y = \begin{pmatrix} Y_1^1 & Y_2^1 & Y_3^1 \\ Y_1^2 & Y_2^2 & Y_3^2 \\ Y_1^3 & Y_2^3 & Y_3^3 \\ Y_1^4 & Y_2^4 & Y_3^4 \\ Y_1^5 & Y_2^5 & Y_3^5 \\ \vdots & & \end{pmatrix} \tag{2.46}$$

Then the spatial coordinates x within bounds of the element e can be expressed using the reference shape functions N^r and the element nodal coordinates as:

$$x = Y^T N^r(x^r) = g(x^r) \tag{2.47}$$

where $g : \mathbb{R}^3 \to \mathbb{R}^3$ is a function mapping the reference coordinates to the actual coordinates for the particular element. One can then define shape functions for elements in their actual coordinates, using the reference shape functions:

$$N_i(x) = N_i^r(x^r) = N_i^r(g^{-1}(x)) \tag{2.48}$$

These shape functions (all of them collectively denoted N) are essentially the same as N^r, only stretched to fit the actual element e. Using the established relationships between the reference and actual coordinates and their corresponding shape functions, an integral over element e can be expressed as:

$$\begin{aligned} \int_e f(N_j(x))de &= \int_e f(N_j^r(g^{-1}(x))de \\ &= \int_{e^r} f(N_j^r(g^{-1}(g(x^r)))|det((\nabla g)(x^r))|de^r \\ &= \int_{e^r} f(N_j^r(x^r))|detJ|de^r \\ &\approx \sum_i w_i f(N_j^r(x_i^r))|detJ|de^r \end{aligned} \tag{2.49}$$

using the substitution $x = g(x^r)$ on the second line, with $J = (\nabla g)(x^r)$, and the Gauss quadrature approximation as defined in (2.43) on the third line. This relationship shows that integral of any function of shape functions on the actual element can be rewritten as an integral on the reference element, using the reference shape functions instead. Lastly, the test functions v used two sections back can be defined using the scalar shape functions N. As mentioned before, each function v has only one of its coordinates nonzero. Therefore:

$$v_i(x) = \begin{pmatrix} N_j(x) \\ O \\ O \end{pmatrix} \text{ or } \begin{pmatrix} O \\ N_j(x) \\ O \end{pmatrix} \text{ or } \begin{pmatrix} O \\ O \\ N_j(x) \end{pmatrix} \tag{2.50}$$

The different indexing i and j is used to emphasize that there is a different number of v and N functions.

The relationship between integration on an element e and computing Gauss quadrature on the corresponding reference element established by (2.49) can be extended to functions of several shape functions as well. Considering that the shape functions v used in the finite element discretisation described in Section 2.3 are merely an extension of shape functions N into 3 dimensions, this integration relationship can be applied to functions of v's as well. Therefore, the integrals I_1, I_2 and I_3, as well as their derivatives, can be computed in per element fashion using the Gauss quadrature procedure on reference elements as described above. For each element of the mesh, one only needs to know its coordinates, as those define the transforming function g. Additionally, for nonlinear terms one also needs to know values of displacements on the element nodes.

It should be noted that the matrices and vectors in (2.38) might be different when the Gauss quadrature integration is performed, as it is only an approximation. The accuracy of the integration depends on the quadrature order and exact expression that is being integrated. No differentiation in notation is made in this work to separate the exact and approximated matrices.

2.3.3 C tensor, symmetries and Voigt notation

The C tensor describing the stress-strain relationship (2.8) and (2.18) can have many forms, depending on the required material properties. For this work, the material is assumed to be isotropic. This means that the entire tensor can be described using only 2 scalar constants:

$$C_{ijkl} = \lambda \delta_{ij}\delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) \tag{2.51}$$

with δ being the Kronecker's delta. These two constants can be expressed in terms of Young's modulus E_Y and Poisson's ration ν:

$$\begin{aligned} \lambda &= \frac{E_Y \nu}{(1+\nu)(1-2\nu)} \\ \mu &= \frac{E_Y}{2(1+\nu)} \end{aligned} \tag{2.52}$$

This form of C introduces many symmetries. Both stresses - σ and S - are symmetric because of this, as well as the product $C\delta E^k$ in (2.35).

In (2.35) it was stated that this term is symmetric in terms of swapping indices j and k. This symmetry is proven here in general case for square real matrices A, X and Y, where A is symmetric but X and Y are arbitrary:

$$\begin{aligned}(XA) : Y &= \sum_{i,k} \left[\sum_{j} \left(X_{ij} A_{jk} \right) Y_{ik} \right] \\ &= \sum_{i,j,k} \left(X_{ij} A_{jk} Y_{ik} \right) \\ &= \sum_{i,j,k} \left(Y_{ik} A_{kj} X_{ij} \right) \text{ ... using symmetry of } A \\ &= \sum_{i,j} \left[\sum_{k} \left(Y_{ik} A_{kj} \right) X_{ij} \right] = (YA) : X\end{aligned} \tag{2.53}$$

which directly proves that $(\nabla v_k(CE)) : \nabla v_j = (\nabla v_j(CE)) : \nabla v_k$ in 2.35 since $S = CE$ is symmetric. Using this observation combined with a property of the double dot product $A : B = A : sym(B)$ for symmetric A, one can show that:

$$\begin{aligned}(\nabla v_k(CE)) : \nabla v_j &= (sym(\nabla v_k)(CE)) : sym(\nabla v_j) \\ &= (\delta\epsilon^k(CE)) : \delta\epsilon^j\end{aligned} \tag{2.54}$$

with $\delta\epsilon^k$ and $\delta\epsilon^j$ being variations of the small strain tensor as defined in (2.7) for functions v_k and v_j respectively. The second term $(C\delta E^k) : \delta E^j$ in (2.35) is thanks to the symmetries of C also symmetric in j and k indices.

Because of the existing symmetries, the tensor notation in the finite element terms is typically simplified by using Voigt notation. This notation reduces the order of tensors, turning second order tensors (matrices) into first order tensors (vectors) and fourth order tensors into matrices. A general 3×3 matrix A is turned into a 6 element vector $\bar{A}$:

$$A = \begin{bmatrix} A_{xx} & A_{xy} & A_{xz} \\ A_{yx} & A_{yy} & A_{yz} \\ A_{zx} & A_{zy} & A_{zz} \end{bmatrix} \rightarrow \bar{A} = \begin{pmatrix} A_{xx} \\ A_{yy} \\ A_{zz} \\ \frac{1}{2}\left(A_{xy} + A_{yx}\right) \\ \frac{1}{2}\left(A_{xz} + A_{zx}\right) \\ \frac{1}{2}\left(A_{yz} + A_{zy}\right) \end{pmatrix} \tag{2.55}$$

For example, the term $\delta\epsilon$ in Voigt notation yields:

$$\delta\bar{\epsilon} = \begin{pmatrix} \frac{\partial v_x}{\partial x} \\ \frac{\partial v_y}{\partial y} \\ \frac{\partial v_z}{\partial z} \\ \frac{1}{2}\left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x}\right) \\ \frac{1}{2}\left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x}\right) \\ \frac{1}{2}\left(\frac{\partial v_y}{\partial z} + \frac{\partial v_z}{\partial y}\right) \end{pmatrix} \tag{2.56}$$

The idea is to put the diagonal values of the matrix first, followed by sums of the off-diagonal terms divided by 2. The variation of the Green-Lagrange strain δE is vectorised into $\delta\bar{E}$ in the same way. Note that each shape function only has one of its coordinates nonzero, as established in (2.50). This means that in the bottom 3 rows of (2.56), only one of the two terms will be nonzero for any shape function.

The C tensor becomes a 6×6 matrix in Voigt notation:

$$\bar{C} = \frac{E_Y}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \tag{2.57}$$

Using this notation, the product $(C\delta E^k) : \delta E^j$ can be expressed in Voigt notation as:

$$(C\delta E^k) : \delta E^j = (\delta\bar{E}^k)^T \bar{C} \delta\bar{E}^j \tag{2.58}$$

This form involves only matrix vector multiplication. This vectorised notation allows for stacking multiple $\delta\bar{E}$ vectors into a matrix. As described in Section 2.3.2, the mesh integrals can be evaluated on the per element basis. One can therefore evaluate a complete contribution of a particular element into the global system matrices by stacking values of $\delta\bar{E}$ for all shape functions of that element (for all degrees of freedom of that element) into a matrix and evaluate:

$$\begin{bmatrix} (\delta\bar{E}^1)^T \\ (\delta\bar{E}^2)^T \\ \vdots \\ (\delta\bar{E}^N)^T \end{bmatrix} \bar{C} \begin{bmatrix} \delta\bar{E}^1 & \delta\bar{E}^2 & \cdots & \delta\bar{E}^N \end{bmatrix} \tag{2.59}$$

with the indexing $1, 2, \ldots, N$ going over all shape functions related to all degrees of freedom of the element. This method of computation can be much faster than computing the terms individually. The term $(\delta\epsilon^k(CE)) : \delta\epsilon^j$ can also be computed for the entire element using one set of matrix multiplications with Voigt notation. The exact procedure can be found in detail in [20].

2.4 Conclusion

This chapter first covers the basics of continuum mechanics that are relevant for this work. The equations of motion for continuous solids are established in their strong and subsequently weak form. Both a linear model and a nonlinear model of deformation are formulated. The source of nonlinearity in this work arises from the use of the Green-Lagrange strain tensor. Overview of the most common types of boundary conditions in solid mechanics is provided.

Next, the finite element discretisation scheme is applied on the weak formulation. Numerical evaluation of individual terms of the motion equations in their weak form is discussed in detail. Standard Lagrange polynomials are used combined with a Gauss quadrature rule. A way of enforcing the homogeneous Dirichlet boundary condition in the discrete equations of motion is described. The damping term is added using the Rayleigh damping model. The result is a system of nonlinear ordinary differential equations with time being the independent variable.

Chapter 3

Vibration modelling

Following the established theory in Chapter 2, this chapter introduces the harmonic balance method (HBM) on top of it. This method eliminates the remaining continuous variable in the system of equations (2.40) - time - and establishes a set of nonlinear algebraic equations to be solved. HBM is a well established tool used to model nonlinear vibrational motion. Closely related to HBM are the alternating frequency time procedure (AFT) and the continuation algorithm, which together provide a tool set to obtain a nonlinear vibration response for a range of frequencies. The way of handling of boundary conditions for the problem in its HBM form is also discussed.

3.1 State of the art

Mechanical vibration is a repetitive back and forth motion of a body (or its parts) around its equilibrium position. Such motion occurs when the body is displaced or deformed from this equilibrium position and forces exist that try to return it back to it. A classical high school example of a vibrational motion is in Figure 3.1. Such motion is often, even though not necessarily, periodic. A general overview of the problematic of vibration analysis can be found in [72, 92, 175]. In many real situations, vibration is damped, meaning the amplitude of the motion is decreasing over time. Such damping occurs naturally due to factors like air friction or friction between multiple moving bodies. Often times damping is a desired factor and is introduced into the system with intent to reduce the amount of unwanted vibration. An overview of damping techniques can be found in [12]. The problematic of identifying damping coefficients from experimental measurements is discussed in [146].

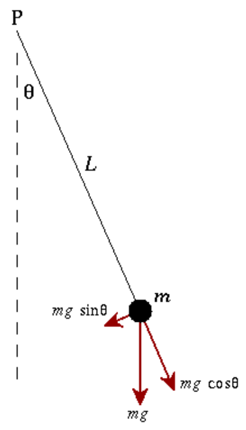


Figure 3.1: A simple example of vibrational motion - a suspended pendulum offset from its equilibrium position and propelled by gravity to swing back and forth.

3.1.1 Harmonic balance and continuation

Harmonic balance method (HBM) is a well established and widely used tool for modelling vibration. The basic idea of the method is an assumption that the solution to nonlinear equations of motion is periodic and can be approximated by a truncated Fourier series of sinusoidal functions:

$$u(t) = \tilde{u}_0 + \sum_{k=1}^{m} \tilde{u}_k^c cos(k\omega t) + \tilde{u}_k^s sin(k\omega t) \tag{3.1}$$

For details on the above equation see Section 3.2. The presence of higher harmonics ($k > 1$) is necessary to account for nonlinearities of the system. The method can be viewed as a Galerkin procedure in the time coordinate, using the *sin* and *cos* functions as shape functions, similarly to what the finite element method does in space coordinates. The use of Fourier series for approximating solutions to nonlinear mechanics problems can be traced back to [112]. In the 1960s the method was used in its current form for solving nonlinear periodic systems [180, 179]. For modern comprehensive overview of the HBM method and its applications in nonlinear vibration problems one can refer to [111]. Cameron and Griffin [29] described an alternating frequency time procedure to numerically evaluate nonlinear terms in frequency domain. Fast Fourier transform (FFT) can be used in HBM to speed up evaluation of the Fourier coefficients [31].

Typically, one is interested in vibration response to a range of frequencies $\langle \omega_{min}, \omega_{max} \rangle$ rather than a single point. A set of solutions for a frequency range is called the frequency response curve (FRC). An example of an FRC is in Figure 3.2. A continuation technique needs to be used for nonlinear problems in order to capture all existing solutions, as there can be more than one for each frequency point. An overview of continuation techniques can be found in [3]. Calvetti and Reichel [28] discussed solving large continuation problems and identifying singularities in the Jacobian matrix. Such singularities can indicate presence of bifurcations. Padmanabhan and Rajendra [133] proposed a method to follow period-doubling bifurcations. Beran and Carlson [16] used domain decomposition methods to compute Hopf bifurcation points. Chan and Saad [33] detailed the use of bordered systems that arise from the continuation method. Xie et al. [192] described tracking of bifurcations and detecting boundaries of stability. Lahmam et al. [115] proposed higher order predictor-corrector algorithms. Damil and Potier-Ferry [43] introduced a method called asymptotic numerical method (ANM) as an alternative to the predictor-corrector approach. This method uses a perturbation expansion technique to compute solutions along the FRC without need for a corrector. Cochelin et al. [36] further studied the ANM method. Kessab et al. [99] used ANM for plastic nonlinearities. Zahrouni et al. [87] proposed adaptations of ANM to handle contact problems. Several other alternatives to continuation method have been used. Incremental harmonic balance method was used in [34, 171]. Khang et al. [130] used shooting method to compute nonlinear vibrations of piecewise-linear systems.

Many software tools implementing continuation techniques have been developed. In 1986 Doedel [50] introduced AUTO, a continuation and bifurcation software for ordinary differential equations, written in Fortran. It supports both OpenMP and MPI parallelism. Several Matlab tools used for numerical continuation and vibration analysis exist. Dhooge et al. [49] developed MATCONT, a Matlab package for numerical bifurcation analysis of ODEs. MANLAB is a Matlab package for interactive continuation and bifurcation analysis of non linear systems of equations [37, 80, 81]. Dankowicz and Schilder [44] introduced Continuation Core and Toolboxes (COCO), a Matlab-based development platform that provides a large amount of standard functionality required for investigating bifurcation problems and implementing toolboxes for new types of problems. NLvib is another Matlab tool which is used for nonlinear vibration analysis [111]. Košata et al. [108] developed HarmonicBalance.jl in language Julia for nonlinear dynamics modelling using harmonic balance. It also supports parallelisation. LOCA is a numerical continuation package in the Trilinos library, implemented in C++ and supporting both OpenMP and MPI parallelism (through other Trilinos packages) [173].

Detroux et al. [48] demonstrated use of HBM to study vibration response of a nonlinear tuned vibration absorber. Aguirre [11] used HBM to perform dynamic analysis of turbomachinery blades. Salles et al. [157] used HBM with dual time stepping algorithms for contact interfaces with fretting wear. Petrov [142] analysed vibration of bladed disks using cyclic symmetry. Seinturier [166] detailed industrial practices for computing forced response of bladed disks for aero-mechanical optimisation. Guskov et al. [83] employed HBM to analyse dynamic properties of Jeffcott rotor systems subject to unbalances.

3.1.2 Reduced order modelling

Reduced order modelling (ROM) techniques are commonly used in vibration analysis to reduce the number of degrees of freedom of the analysed system. Comparison of various techniques can be found in [150]. One of the most popular ROM techniques is the

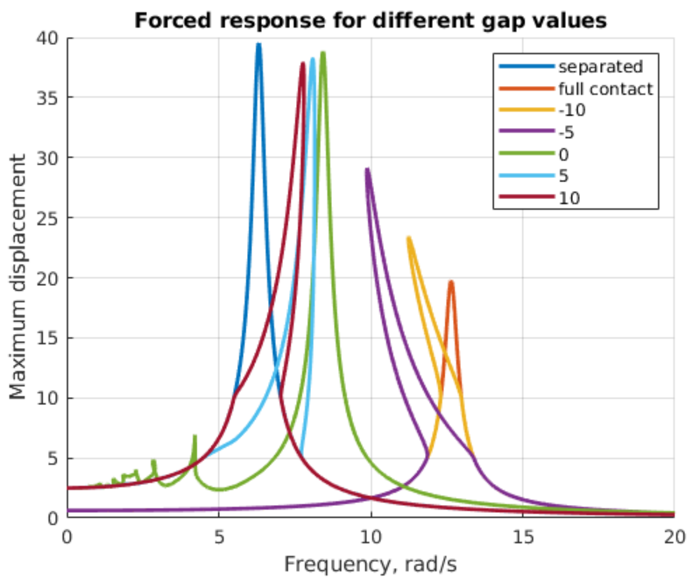


Figure 3.2: An example of an FRC. Amplitude of nonlinear vibration computed over a range frequencies. This response is for a single dof spring-mass oscillator with a wall contact at various gaps.

Craig-Bampton model [41] which uses a small selected set of normal modes to transform the system into modal coordinates. This transformation results in using only a small set of modal amplitudes as new unknowns. Kim and Lee [100] enhanced this method by compensating for the error caused by the residual modes and [24] further improved the method. Application of model order reduction techniques for geometrically nonlinear structures can be found in [178, 185]. Kang et al. [95] and Kim and Kang [103] studied a hyper-reduction method for rotating component (such as propeller blades) forced vibration analysis. Kim et al. [102] performed structural analysis based on reduced order modelling for gas turbine blades, using a transient time model. They employed both full order models and reduced ones in different stages of the analysis, using parallel direct sparse solver (PARDISO) for the linear system of equations.

3.1.3 Linear and nonlinear solvers

When discretising a PDE system of any sort using the FE method (or any other discretisation method), one eventually encounters a set of linear or nonlinear algebraic equations to solve. The nonlinear system arising from the harmonic balance method can be written as follows:

$$Z(\omega)\tilde{u} + \tilde{F}^{nl}_{int}(\tilde{u},\omega) = \tilde{F} \tag{3.2}$$

See again Section 3.2 for details. Solving such systems is often the most time consuming part of the entire computing process. Extensive research has therefore been conducted regarding nonlinear and linear numerical solvers. These two types of solvers often operate in a hierarchical manner. An outside nonlinear solver runs in loops, solving a linear system repeatedly, until a good solution approximation is found. This is the case for example of the Newton-Raphson method, possibly the most commonly used nonlinear solver. History of this solver can be found in [25]. Many variants of this method have been developed over the years. Noor and Waseem [131] proposed a variant with cubic convergence rate. Knoll and Keyes [107] discussed Jacobian free Newton-Krylov methods, meaning no need for assembling the Jacobian matrix explicitly. A generalisation of the method for nonsmooth functions was studied in [147]. Other nonlinear solvers have been proposed, such as Adomian decomposition [8] or homotopy perturbation [75]. An overview of methods for solving nonlinear systems can be found in [7, 132].

Linear solvers can be categorised into two groups - direct and iterative. The base distinction is that a direct solver provides after a series of calculations the final solution to the system, without any intermediate steps. Iterative solvers on the other hand begin with an initial solution estimate which they gradually improve upon in a series of iterations, until convergence criteria are met. Stopping an iterative solver prematurely can still provide a reasonable solution estimate, which is not the case with direct solvers. On the other hand, iterative solvers commonly struggle more with ill conditioned matrices [177].

When solving a linear system, it is important to consider properties of the system matrix. Matrices coming from finite element discretisations typically have sparse pattern, meaning proportionally only a small percentage of their elements are nonzero. An example of such matrix is in Figure 3.3. Such matrices are usually stored in special formats that reduce amount of required storage space to its minimum [144]. A solver working with such matrix should ideally preserve this sparsity as much as possible. Other important matrix properties are symmetry, definiteness and the condition number. Laub [116] covered topics regarding analysis of these matrix properties.

Direct linear solvers typically rely on a form of LU (or Cholesky for symmetric matrices) factorisation of the matrix. An overview of direct solvers for sparse matrices is in [45]. One of the challenges when factorising a sparse matrix is to maintain the sparsity of the factors. In theory, factors of a sparse matrix are dense. However, a good reordering of the matrix rows and columns can be found to achieve at least a certain level of sparsity. The most popular software packages implementing such reordering techniques are Scotch [140], Metis [97] or Parmetis [96]. Many software packages for direct solving of linear systems have been developed, such as SuperLU [47] and its parallel version [118], PARDISO [2, 163, 164], which was later integrated into the IntelMKL library [1], and MUMPS [4, 5], which was used in this work.

Overview of iterative linear solvers can be found in [77, 155]. A prominent category of linear solvers are the so called Krylov subspace solvers. These solvers generate a sequence of growing Krylov spaces on which the solution is approximated. Krylov spaces are generated iteratively by multiplying the current residual by the system matrix to obtain a new basis vector which extends the approximation space [182]. In exact arithmetic, such solvers are guaranteed to find the exact solution in at most n iterations, where n is the rank of the system matrix. However, in practice a good approximation can be found much earlier. Since Krylov methods don't require any factorisation, no reordering of the matrix is necessary, and their memory requirements are generally much lower for large systems.

Among the most popular Krylov subspace methods are the conjugate gradient method (CG) [90] and the generalised minimal residual method (GMRES) [156]. CG is used for symmetric positive definite matrices, while GMRES can be used for general matrices. However, GMRES requires more memory as the entire Krylov space basis needs to be stored. It also requires a more computationally expensive orthogonalisation procedure. Convergence analysis of GMRES can be found in [78]. Many other Krylov subspace iterative methods exist. The biconjugate gradient method (BiCG) [69] and its stabilised version (BiCGStab) [183] are modifications of CG for nonsymmetric matrices. MINRES [134] is a conjugate gradient type method for symmetric but indefinite matrices. CGS [169] is a modification of BiCG that doesn't require multiplications by transpose of the matrix. Ghai et al. [73] provided a comparison of Krylov subspace methods for large-scale linear systems.

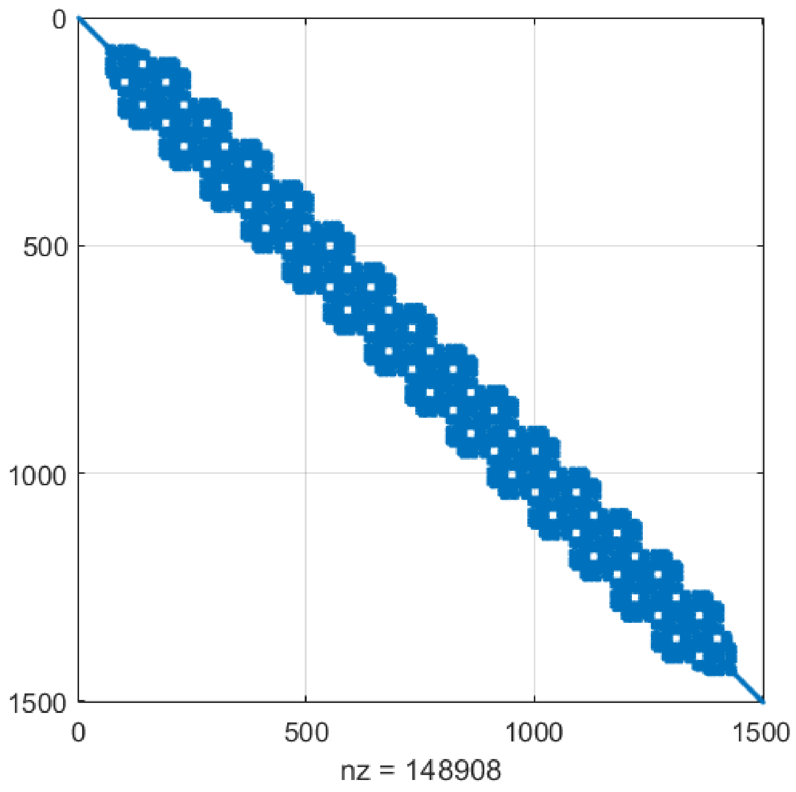


Figure 3.3: An example of the nonzero value (blue) pattern of a stiffness matrix for FE discretisation of a clamped-clamped beam. This particular matrix has an average of 99 nonzero values per row, while its size is 1503 $\times$ 1503. This means the sparsity of this matrix is around 0.93.

An important part of solving a linear system is preconditioning of the system matrix. This essentially means finding an approximation of the inverse of the matrix which will reduce the condition number of the system, making it easier to solve. In case of an iterative solver, preconditioning can radically reduce the number of iterations required to reach the solution. In direct solver, the matrix rows and columns can be scaled and reordered in order to reduce impacts of numerical errors. Bertaccini and Durastante [17] and Benzi [14] provided an overview of preconditioning options for large linear systems. Modified variants of direct solvers can be used to compute fast incomplete factorisation of the matrix that can be used as a preconditioner for an iterative solver. Arany [6] and Kaasschieter [94] described use of incomplete Cholesky factorisation as preconditioning for CG. Mittal and Al-Kurdi [128] studied the use of incomplete LU factorisation to precondition GMRES. Désiré and Atenekeng Kahou [187] studied use of Schwarz domain decomposition method as GMRES preconditioning.

3.2 Harmonic Balance Method

The previous sections showed how the analytical partial differential equations of elasticity, including the Green-Lagrange model for large deformations, are discretised into a system of ordinary nonlinear differential equations (2.40), where the unknown displacement $u(t)$ is a vector function of time. When provided with initial conditions for $u(0)$ and $\dot{u}(0)$, such equations can be solved by a time integration scheme. By discretising the time variable into a set of time points, one can compute the solution u at next time point using the knowledge of solution (and its first derivative) in the previous time points by applying methods like Euler or Runge-Kutta iterative schemes.

Vibration is a type of motion that exhibits a certain repetitive pattern, typically many times over. In this work the assumption is made that the vibrational motion in question is periodic with a single time period T. In many scenarios, it is not important at which time point the motion starts and what are the absolute values on the time axis. Having the information about the motion over any window of one time period provides enough information. It is not important at which time point this period window is anchored, as seen in Figure 3.4.

There are 2 types of vibrations. The first is free vibration, meaning there is no external force present. In the motion equations (2.40), this would mean $F = O$. If the damping term is present, such vibration diminishes over time due to dissipation of energy. The motion is therefore technically not periodic as the amplitude decreases with time. If no damping is present, the motion keeps repeating periodically with the same amplitude. The frequency of the motion is determined by the physical parameters of the system. The second type is forced vibration. This occurs when a periodic external force (excitation force) actuates the body, keeping it in motion even when damping is present. Ignoring any possible initial transitory motion when the force action begins, in steady state when the motion becomes periodic, its period matches the period of the excitation force. Any other components of the motion will be eliminated by the damping. Even without any damping, the body is assumed to react only with the frequency of the excitation force.

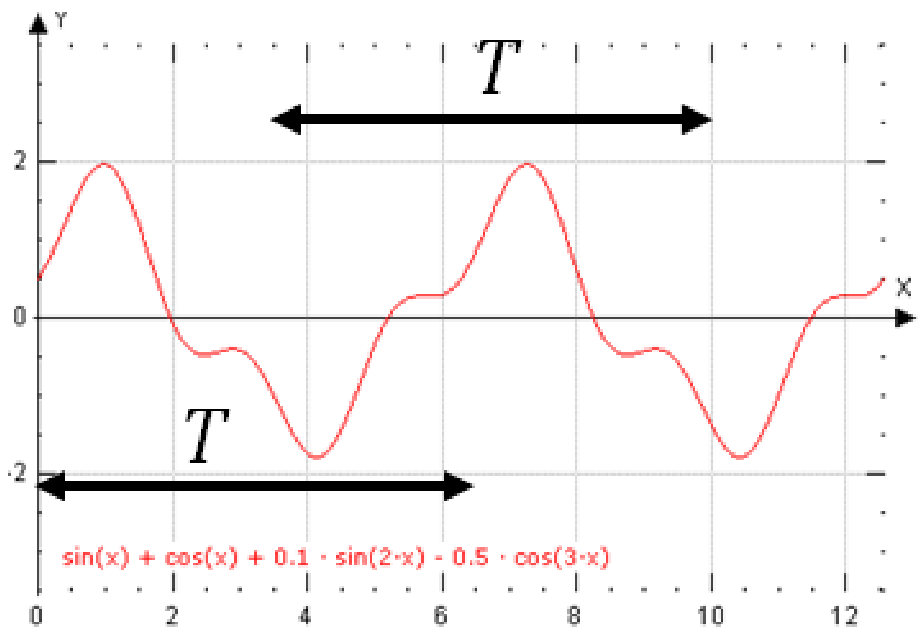


Figure 3.4: Example of a periodic harmonic motion, with 2 possible time windows stretching over one period of the motion marked. The character of the motion is fully described within both of those windows.

The most fundamental way to describe a periodic motion is by using sine and cosine waves. A motion with a sinusoidal shape is called harmonic motion. Periodic motion with sinusoidal shape appears in many physical phenomena, such as vibrating strings in musical instruments, a pendulum released from a slightly offset position, and others. Furthermore, by a series of sine and cosine functions with growing frequency, one can approximate other periodic functions. This is the basis of Fourier series and Fourier analysis. The sine and cosine functions additionally have the convenient property of being each other's derivatives (ignoring the sign).

Harmonic balance method (HBM) is an alternative method to time integration techniques that can be used to model vibrational motion. It is based on the ideas and assumptions discussed above. The excitation force $F(t)$ is assumed to have a sinusoidal shape, i.e.:

$$F(t) = a^c cos(\omega t) + a^s sin(\omega t) \tag{3.3}$$

with ω being the excitation frequency and a^c and a^s being vectors of coefficients, independent of time.

Given the frequency of motion ω, the period of motion is:

$$T = \frac{2\pi}{\omega} \tag{3.4}$$

In case of a linear system (using the linear strain formulation in 2.6, meaning F_{int}^{nl} in 2.37 is zero), the body will respond purely on that same frequency, meaning the solution can be expressed as:

$$u(t) = \tilde{u}^c cos(\omega t) + \tilde{u}^s sin(\omega t) \tag{3.5}$$

with $\tilde{u}^c$ and $\tilde{u}^s$ being again vectors of constant coefficients. In nonlinear case, an excitation at frequency ω can invoke motion not only at that frequency, but also at its integer multiples. The solution formula needs to be therefore expanded to:

$$u(t) = \tilde{u}_0 + \sum_{k=1}^{m} \tilde{u}_k^c cos(k\omega t) + \tilde{u}_k^s sin(k\omega t) \tag{3.6}$$

The above can be understood as a finite Fourier series approximation of the solution, as the solution in this form might not be the exact solution. The nonlinearity can induce motion that could only be captured exactly by using an infinite Fourier series, meaning $k \to \infty$. The $\tilde{u}$ coefficients correspond to different harmonic components of the solution. The coefficient $\tilde{u}_0$ represents the 0th harmonic part, or the constant term. In general, the more of the higher harmonic components are used for the solution, the more accurate the obtained solution will be. The newly introduced coefficients $\tilde{u}$ can be collectively expressed as a vector:

$$\tilde{u}(t) = \begin{pmatrix} \tilde{u}_0 \\ \tilde{u}_1^c \\ \tilde{u}_1^s \\ \vdots \\ \tilde{u}_m^c \\ \tilde{u}_m^s \end{pmatrix} \tag{3.7}$$

Not all harmonic parts (shortly harmonics) need to be necessarily present in the system. Based on the available information about the nature of the problem at hand, one can selectively use only specific harmonics that are expected to be the most relevant ones (responding with highest amplitudes) for approximating the solution in time. The excitation vector is then expressed using the same harmonics that were used for the solution:

$$F(t) = a_0 + \sum_{k=1}^{m} a_k^c cos(k\omega t) + a_k^s sin(k\omega t) \tag{3.8}$$

However, F will only have nonzero coefficients at first harmonic, i.e. the a_1^c and a_1^s values. Having the solution in form (3.6), one can also express its first and second derivatives:

$$\dot{u}(t) = \omega \sum_{k=1}^{m} k \left[-\tilde{u}_k^c sin(k\omega t) + \tilde{u}_k^s cos(k\omega t) \right] \tag{3.9}$$

$$\ddot{u}(t) = -\omega^2 \sum_{k=1}^{m} k^2 \left[\tilde{u}_k^c cos(k\omega t) + \tilde{u}_k^s sin(k\omega t)\right] \tag{3.10}$$

As mentioned previously, the solution in this form is only an approximation. A deviation from the actual solution can therefore be expressed in a form of residual:

$$R(t) = M\ddot{u} + D\dot{u} + F_{int}(u) - F(t) \tag{3.11}$$

Having defined the residual, it is natural to aim to minimise it. One way to do that is to require projections of the residual on the same harmonic functions used to approximate the solution to be zero, i.e.:

$$\begin{aligned} &\frac{2}{T}\int_0^T R(t)\cdot 1 \quad && dt = 0 && \\ &\frac{2}{T}\int_0^T R(t)\cos(k\omega t) \quad && dt = 0, && \text{for } k = 1..m \\ &\frac{2}{T}\int_0^T R(t)\sin(k\omega t) \quad && dt = 0, && \text{for } k = 1..m \end{aligned} \tag{3.12}$$

As before, note that not all harmonics, meaning not all values of k, need to be present necessarily. The above projections using the integral $\int_0^T dt$ are essentially a dot product in space of functions defined on $\langle 0, T\rangle$. As mentioned previously, one period is enough to entirely determine the shape of the solution. This procedure is analogous to the weak formulation of the partial differential equations in 2.3. The solution there was also expressed as a sum of coefficients multiplied by the base shape functions. The equations of motion were then, like here, projected onto the same shape functions, resulting in a set of equations with the solution coefficients as unknowns. Here, the newly introduced unknowns are the harmonic coefficients in vector $\tilde{u}$. The integral in the projections over time eliminates time as a variable in the system. Therefore, by evaluating the integrals in (3.12), a new system of purely algebraic equations is obtained. By eliminating time, the transition is made from time domain, where the vector $u(t)$ is the unknown, to frequency domain, with unknown being the vector of coefficients $\tilde{u}$. The matrices and vectors in (3.11) are by evaluating the projections transformed into frequency domain. It is useful to note here that the harmonic functions $cos(k\omega t)$ and $sin(k\omega t)$, as well as the constant term function, are orthogonal to one another under the dot product defined by the projection integral $\int_0^T dt$.

The excitation vector can be transformed analytically, resulting in the excitation vector in frequency domain $\tilde{F}$:

$$\tilde{F} = \begin{pmatrix} 2a_0 \\ a_1^c \\ a_1^s \\ \vdots \\ a_m^c \\ a_m^s \end{pmatrix} \tag{3.13}$$

As mentioned previously, only the first harmonic coefficients a_1^c and a_1^s will be considered nonzero in the following text.

Another portion of the residual $R(t)$ that can be transformed analytically is the linear part $M\ddot{u} + D\dot{u} + Ku$. The resulting term in frequency domain will be $Z(\omega)\tilde{u}$, where:

$$Z(\omega) = \begin{bmatrix} 2K & O & O & O & O & \cdots & O & O \\ O & K-\omega^2 M & -\omega D & O & O & & O & O \\ O & \omega D & K-\omega^2 M & O & O & & O & O \\ O & O & O & K-(2\omega)^2 M & -2\omega D & & O & O \\ O & O & O & 2\omega D & K-(2\omega)^2 M & & O & O \\ \vdots & & & & & \ddots & & \vdots \\ O & O & O & O & O & & K-(m\omega)^2 M & -m\omega D \\ O & O & O & O & O & \cdots & m\omega D & K-(m\omega)^2 M \end{bmatrix} \tag{3.14}$$

The remaining part of $R(t)$ to be handled is $F_{int}^{nl}(u)$, which will transform into $\tilde{F}_{int}^{nl}(\tilde{u}, \omega)$. This transformation cannot be performed analytically in general and will be addressed later. The transformation of the equations of motion from (2.40), using (3.12), results in equations of motion in frequency domain:

$$Z(\omega)\tilde{u} + \tilde{F}_{int}^{nl}(\tilde{u}, \omega) = \tilde{F} \tag{3.15}$$

These equations are of purely algebraic character, as there is no continuous variable present in them. The spatial dimensions have been eliminated by the finite element discretisation and time has been removed by the projections of the residual on the harmonic functions. The following sections will focus on how to solve this nonlinear system of equations.

3.3 Newton-Raphson

A common tool used to solve general systems of nonlinear algebraic equations is the so called Newton-Raphson (shortly Newton's) method [98]. Assume a general vector of unknowns $x \in \mathbb{R}^n$. A set of nonlinear equations can be then written as a vector function $G : \mathbb{R}^n \to \mathbb{R}^n$:

$$G(x) = 0 \tag{3.16}$$

Given a solution guess x^k, the Newton's method computes a solution step Δx^k:

$$\nabla G(x^k)\Delta x^k = -G(x^k); \tag{3.17}$$

The solution is then updated by adding the step:

$$x^{k+1} = x^k + \Delta x^k \tag{3.18}$$

This process repeats until the solution to (3.16) is found, or the maximum number of iterations is reached. An initial solution guess x^0 needs to be provided for the method to start. For the lack of a better alternative, zero vector is commonly used. Acceptance of the solution guess x^k as a solution can be decided by various criteria. One can demand the norm $\|G(x^k\|$ (the error in the residual) being under a given tolerance. A relative residual error $\frac{\|G(x^k)\|}{\|G(x^0)\|}$ can also be measured. A stagnation of the solution guesses can also be checked, i.e. checking the size of the step $\|\Delta x^k\|$. Convergence of Newton's iterations to a solution is not guaranteed and it depends on the solved equations (especially on the character of the nonlinearity) as well as the initial guess. However, Newton's method generally shows good convergence quality when the initial guess is picked near the actual solution. In such cases, the method has been proven to have quadratic rate of convergence [98].

Given the general formula for the Newton's iteration method (3.17), its application on the HBM equations as defined in 3.15 yields the following:

$$\begin{aligned} \nabla_{\tilde{u}} \left[Z(\omega)\tilde{u} + \tilde{F}^{nl}_{int}(\tilde{u},\omega) - \tilde{F} \right] \Delta\tilde{u}^k = \\ \left[Z(\omega) + \nabla_{\tilde{u}}\tilde{F}^{nl}_{int}(\tilde{u},\omega) \right] \Delta\tilde{u}^k = \tilde{F} - Z(\omega)\tilde{u}^k - \tilde{F}^{nl}_{int}(\tilde{u}^k,\omega) \end{aligned} \tag{3.19}$$

The above system can be solved by any available serial or parallel linear solver that accepts as inputs the system matrix and the right hand side vector. For given parameter ω one can this way obtain the nonlinear vibrational response of the system for that frequency. In this work two such readily available solvers have been tested - MUMPS (parallel direct sparse solver) and GMRES. Details about how they were used can be found in Chapter 5 and results obtained in Chapter 6. Details about about evaluating the nonlinear terms in the equations and computing a full nonlinear response curve for a range of frequencies are discussed in the following sections.

3.4 Alternating Frequency Time (AFT)

The evaluation of the projection integrals (3.12) for the nonlinear term F_{int}^{nl} of the residual $R(t)$ in Section 3.2 was skipped. This is because calculating these integrals analytically is either impossible or impractical. Therefore, a numerical approximation is called upon once more. A method called AFT is used for this. AFT stands for alternating frequency time procedure [29]. It samples the time along the period T to numerically approximate value of the integrals $\int_0^T$. For example, projection on the k-th harmonic cosine function can be approximated as follows:

$$\begin{aligned}\frac{2}{T}\int_0^T F_{int}^{nl}(u(t))cos(k\omega t)dt &\approx \frac{2}{N_t}\sum_{n=0}^{N_t-1} F_{int}^{nl}(u(nh_t))cos(k\omega nh_t)\\ &= \frac{2}{N_t}\sum_{n=0}^{N_t-1} F_{int,n}^{nl}cos(k\omega nh_t)\end{aligned} \tag{3.20}$$

given that the time period T is divided equally into N_t steps of size $h_t = \frac{T}{N_t}$. The discrete time steps are $t_n = nh_t$. This operation can be collectively along all used harmonic functions seen as discrete Fourier transform of $F_{int,n}^{nl}$ into $\tilde{F}_{int}^{nl}$. A diagram of this procedure is in Figure 3.5. To obtain values of $u(t)$ for the discrete time points t_n, equation (3.6) is used:

$$u(t_n) = \tilde{u}_0 + \sum_{k=1}^{m} \tilde{u}_k^c cos(k\omega t_n) + \tilde{u}_k^s sin(k\omega t_n) \tag{3.21}$$

It can be noted that:

$$\omega t_n = n\omega h_t = n\omega\frac{T}{N_t} = n\omega\frac{2\pi}{\omega N_t} = n\frac{2\pi}{N_t} \tag{3.22}$$

which shows that the discretised values of $cos(k\omega t_n)$ and $sin(k\omega t_n)$ are in fact not ω dependent. They can therefore be defined universally as:

$$\begin{aligned} b_{k,n}^c = cos(k\omega t_n) &= cos\left(kn\frac{2\pi}{N_t}\right)\\ b_{k,n}^s = sin(k\omega t_n) &= sin\left(kn\frac{2\pi}{N_t}\right)\end{aligned} \tag{3.23}$$

This means that as long as F_{int}^{nl} only depends on u and not $\dot{u}$ or $\ddot{u}$, the transformation integrals such as in (3.20) will only depends on $\tilde{u}$ and not ω. A simplification can therefore be made by removing ω as a parameter:

$$\tilde{F}_{int}^{nl}(\tilde{u},\omega) \rightarrow \tilde{F}_{int}^{nl}(\tilde{u}) \tag{3.24}$$

Using this numerical integration procedure, the vector of nonlinear forces in frequency domain $\tilde{F}^{nl}_{int}$ can be computed as:

$$\tilde{F}^{nl}_{int}(\tilde{u}) = \frac{2}{N_t} \sum_{n=0}^{N_t-1} \begin{pmatrix} 1 \cdot F^{nl}_{int,n} \\ b^c_{1,n} \cdot F^{nl}_{int,n} \\ b^s_{1,n} \cdot F^{nl}_{int,n} \\ \vdots \\ b^c_{m,n} \cdot F^{nl}_{int,n} \\ b^s_{m,n} \cdot F^{nl}_{int,n} \end{pmatrix} \tag{3.25}$$

An equality sign rather that the $\approx$ sign is used as this is the actual definition of the nonlinear forces vector which will be used in computations. It can be seen how the AFT got its name. Starting from a vector in frequency domain $\tilde{u}$, it transfers this vector to time domain as shown in (3.21). In time domain, the nonlinear vectors $F^{nl}_{int,n}$ are computed for each time point. After that, these vectors are multiplied by the b functions values and summed to obtain a vector in frequency domain.

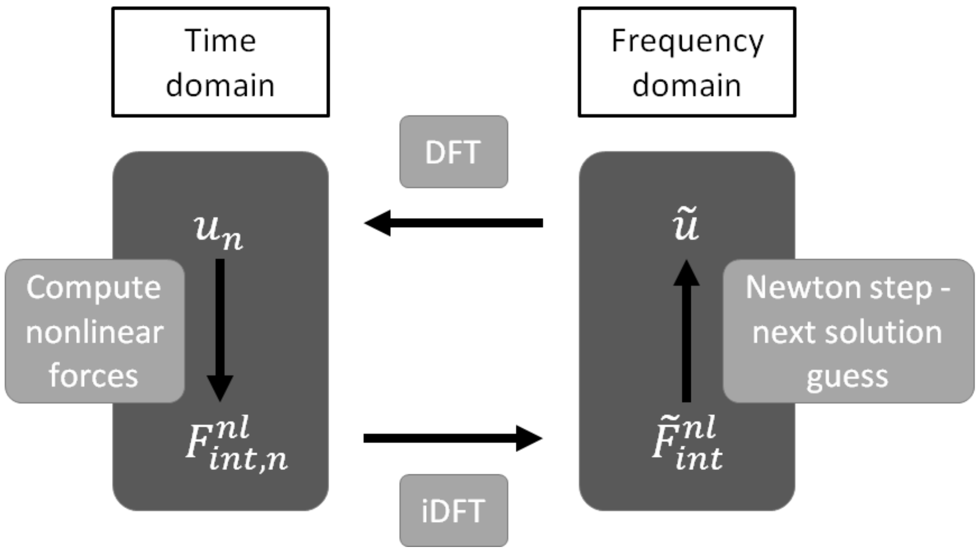


Figure 3.5: Diagram of AFT procedure to evaluate nonlinear forces vector $\tilde{F}^{nl}_{int}$. DFT and iDFT stand for discrete Fourier transform and its inverse respectively.

The matrix of derivatives $\nabla_{\tilde{u}}\tilde{F}^{nl}_{int}$ also needs to be evaluated in order to assemble the complete Jacobian matrix for Newton's iterations as described in (3.19). Diagram of this evaluation is in Figure 3.6. Mathematically it yields the following:

$$\nabla_{\tilde{u}}\tilde{F}^{nl}_{int}(\tilde{u}) = \frac{2}{N_t} \sum_{n=0}^{N_t-1} \begin{bmatrix} 1 \cdot K^{nl}_n & b^c_{1,n} \cdot K^{nl}_n & b^s_{1,n} \cdot K^{nl}_n & \cdots & b^c_{m,n} \cdot K^{nl}_n & b^s_{m,n} \cdot K^{nl}_n \\ b^c_{1,n} \cdot K^{nl}_n & b^c_{1,n}b^c_{1,n} \cdot K^{nl}_n & b^c_{1,n}b^s_{1,n} \cdot K^{nl}_n & & b^c_{1,n}b^c_{m,n} \cdot K^{nl}_n & b^c_{1,n}b^s_{m,n} \cdot K^{nl}_n \\ b^s_{1,n} \cdot K^{nl}_n & b^s_{1,n}b^c_{1,n} \cdot K^{nl}_n & b^s_{1,n}b^s_{1,n} \cdot K^{nl}_n & & b^s_{1,n}b^c_{m,n} \cdot K^{nl}_n & b^s_{1,n}b^s_{m,n} \cdot K^{nl}_n \\ \vdots & & & \ddots & & \vdots \\ b^c_{m,n} \cdot K^{nl}_n & b^c_{m,n}b^c_{1,n} \cdot K^{nl}_n & b^c_{m,n}b^s_{1,n} \cdot K^{nl}_n & & b^c_{m,n}b^c_{m,n} \cdot K^{nl}_n & b^c_{m,n}b^s_{m,n} \cdot K^{nl}_n \\ b^s_{m,n} \cdot K^{nl}_n & b^s_{m,n}b^c_{1,n} \cdot K^{nl}_n & b^s_{m,n}b^s_{1,n} \cdot K^{nl}_n & \cdots & b^s_{m,n}b^c_{m,n} \cdot K^{nl}_n & b^s_{m,n}b^s_{m,n} \cdot K^{nl}_n \end{bmatrix} \tag{3.26}$$

where $K_n^{nl} = K^{nl}(u(t_n))$, using the notation established by (2.36).

The equation above shows that in order to numerically evaluate $\nabla_{\tilde{u}} \tilde{F}_{int}^{nl}$, one needs to evaluate the gradient of the forces $\nabla_u F_{int}^{nl}$ in time domain for each of the discrete time steps t_n. The subsequent transfer into the frequency domain gradient is a straightforward transformation using the harmonic basis functions b and summing over the time points.

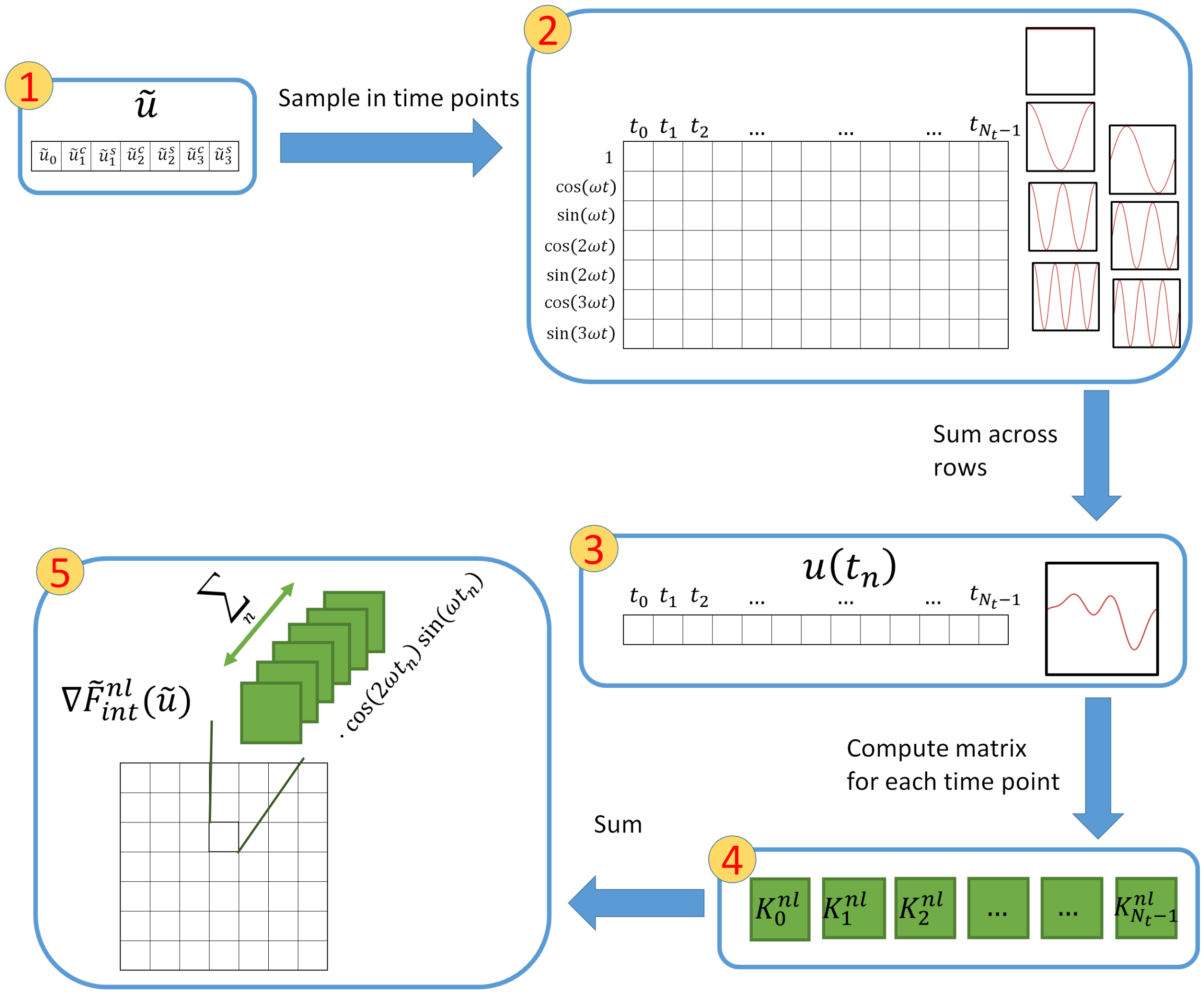

1
$\tilde{u}$
$\tilde{u}_0$ $\tilde{u}_1^c$ $\tilde{u}_1^s$ $\tilde{u}_2^c$ $\tilde{u}_2^s$ $\tilde{u}_3^c$ $\tilde{u}_3^s$
Sample in time points
2
t_0 t_1 t_2 ... ... t_{N_t-1}
1
$\cos(\omega t)$
$\sin(\omega t)$
$\cos(2\omega t)$
$\sin(2\omega t)$
$\cos(3\omega t)$
$\sin(3\omega t)$
Sum across rows
3
$u(t_n)$
t_0 t_1 t_2 ... ... t_{N_t-1}
Compute matrix for each time point
4
K_0^{nl} K_1^{nl} K_2^{nl} ... ... $K_{N_t-1}^{nl}$
Sum
5
$\nabla \tilde{F}_{int}^{nl}(\tilde{u})$
$\sum_n$
$\cdot\cos(2\omega t_n)\sin(\omega t_n)$

Figure 3.6: Diagram of AFT procedure to evaluate nonlinear Jacobian matrix $\nabla_{\tilde{u}} \tilde{F}$.

3.5 Boundary conditions

In order to achieve $u_i(t) = 0$ in the HBM system all the harmonic coefficients related to that time domain degree of freedom need to be 0, i.e.:

$$\begin{aligned} u_i = 0 \implies \tilde{u}_{i,0} &= 0 \\ \tilde{u}_{i,k}^c &= 0 \text{ for all } k \\ \tilde{u}_{i,k}^s &= 0 \text{ for all } k \end{aligned} \tag{3.27}$$

The way of imposing this requirement is analogous to how it is imposed in time domain 2.3.1. The corresponding positions in the right hand side vector in (3.19) are zeroed out, as well as the Jacobian matrix rows and columns. In fact, if one performs this modification on the time domain matrices and vectors as described in 2.3.1, this will automatically create the desired matrix and vector form in the HBM system. The stiffness matrix on the diagonal blocks in $Z(\omega)$ will provide the required 1 on the main diagonal for all solution coefficients on all harmonics and zeroed out appropriate rows and columns. The mass and damping matrices will also have the required rows and columns zeroed out. The nonlinear vector $\tilde{F}^{nl}_{int}$ is assembled from the nonlinear vectors in time domain as shown in (3.25). Similarly, the nonlinear part of the Jacobian matrix $\nabla_{\tilde{u}}\tilde{F}^{nl}_{int}$ is assembled from nonlinear Jacobian matrices in time domain (3.26). Therefore, both $\tilde{F}^{nl}_{int}$ and $\nabla_{\tilde{u}}\tilde{F}^{nl}_{int}$ will have required rows and columns zeroed out.

3.6 Continuation

In order to obtain a set of solutions for a range of frequencies, typically called frequency response curve (FRC), a continuation algorithm needs to be employed. This means following the solution curve, using an already computed solution as an initial guess to compute the next one. There are many types of continuation algorithms. The one employed in this work is a predictor-corrector approach. For prediction, either naive, tangent or secant predictors are used. For corrector the so called pseudo arc-length formula is used. This corrector is a linearisation of another commonly used corrector called the arc-length corrector. Diagram describing one step of a predictor-corrector continuation is in Figure 3.8. Example of a nonlinear FRC can be seen in Figure 3.7.

The goal is to obtain a set of solutions for a range of ω:

$$\{(\tilde{u},\omega)\} \text{ for } \omega \in \langle \omega_{min}, \omega_{max} \rangle \tag{3.28}$$

These points can be described as solutions to the following equation:

$$G(\tilde{u},\omega) = Z(\omega)\tilde{u} - \tilde{F}^{nl}_{int}(\tilde{u},\omega) - \tilde{F} \tag{3.29}$$

Note that ω is now also a variable of the HBM system. A tangent $\tau = \left(\begin{smallmatrix}\tau_{\tilde{u}}\\ \tau_{\omega}\end{smallmatrix}\right)$ of the solution curve at point $(\tilde{u},\omega)$ is implicitly defined as:

$$[\nabla_{\tilde{u}}G(\tilde{u},\omega)\nabla_{\omega}G(\tilde{u},\omega)] \begin{pmatrix}\tau_{\tilde{u}}\\ \tau_{\omega}\end{pmatrix} = O \tag{3.30}$$

In order to simplify the equation above, it is assumed $\tau_{\omega} = 1$. This assumption will allow to compute the $\tilde{u}$ part of the tangent vector at the cost of one solve of the linear system

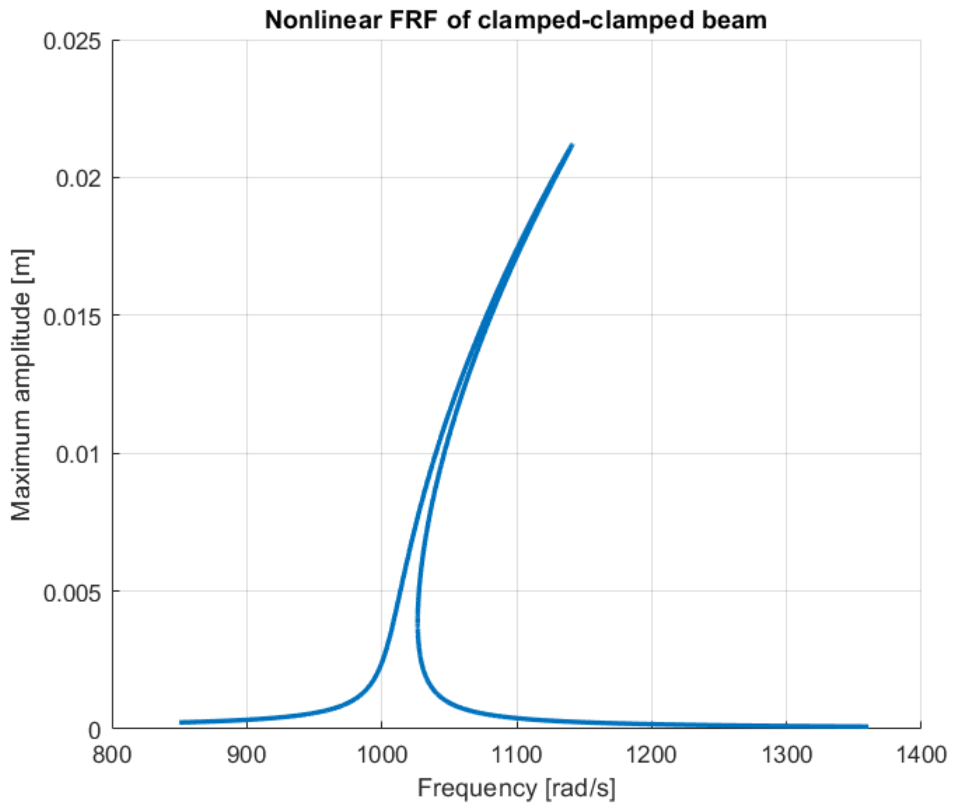


Figure 3.7: Example of nonlinear FRC for clamped-clamped beam for a range of frequencies. It can be seen that for certain frequencies multiple solutions exist.

of equations, same as in Newton iterations. The assumption is reasonable, as it is rare to encounter the case where the ω part of the tangent vector would be 0 (i.e. the tangent vector would be exactly vertical). The new formula to compute the tangent is then:

$$\nabla_{\tilde{u}} G(\tilde{u}, \omega) \tau_{\tilde{u}} = -\nabla_{\omega} G(\tilde{u}, \omega) \tag{3.31}$$

Having computed the tangent τ, one can obtain a reasonable guess for the next solution $(\tilde{u}^p_{i+1}, \omega^p_{i+1})$ on the solution curve, using an already computed solution $(\tilde{u}_i, \omega_i)$ as a starting point:

$$(\tilde{u}^p_{i+1}, \omega^p_{i+1}) = (\tilde{u}_i, \omega_i) + \frac{s}{\|\tau\|}(\tau_{\tilde{u}}, \tau_{\omega}) \tag{3.32}$$

with s being the size of the step one wishes to perform. This is called the prediction part of the continuation algorithm. After the predicted solution $(\tilde{u}^p_{i+1}, \omega^p_{i+1})$ is obtained, it is used as the initial guess for the Newton iterative procedure, which identifies an actual solution on the solution curve in the vicinity of the predicted point. This part is called the correction. The step size parameter s can be adjusted throughout the tracking of the curve depending on how the correction procedure manages to converge.

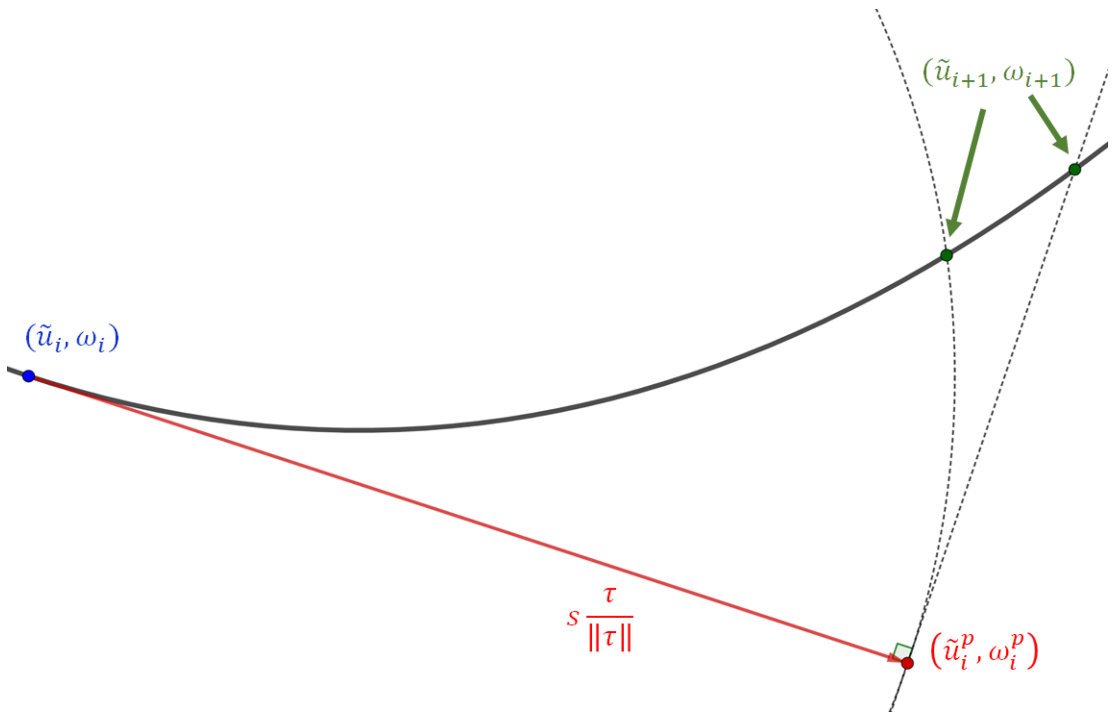

$(\tilde{u}_{i+1}, \omega_{i+1})$
$(\tilde{u}_i, \omega_i)$
$s\frac{\tau}{\|\tau\|}$
$(\tilde{u}_i^p, \omega_i^p)$

Figure 3.8: Diagram of one continuation step. Blue point - already computed solution, red arrow - prediction step using tangent predictor, red point - next solution prediction, blue points - next possible solutions, obtained with arc-length and pseudo arc-length corrector constraints (from left to right).

Since ω was added as a variable, another equation also needs to be added to properly determine solutions to (3.29):

$$g(\tilde{u}, \omega) = 0 \tag{3.33}$$

where $g(\tilde{u}, \omega) : \mathbb{R}^{N+1} \to \mathbb{R}$.

The pseudo arc-length corrector approach uses g to enforce orthogonality between the search space of the Newton iterations and the prediction vector:

$$g(\tilde{u}, \omega) = dot((\tilde{u}, \omega) - (\tilde{u}^p_{i+1}, \omega^p_{i+1}), \tau) \tag{3.34}$$

Its derivative yields:

$$\nabla g = (\nabla_{\tilde{u}} g, \nabla_{\omega} g) = (\tau_{\tilde{u}}, \tau_{\omega}) \tag{3.35}$$

For comparison, the arc-length corrector (which is not used in this work) would yield:

$$g(\tilde{u}, \omega) = \|(\tilde{u}, \omega) - (\tilde{u}^p_{i+1}, \omega^p_{i+1})\| - s \tag{3.36}$$

Newton step with the Jacobian matrix extended by the added variable ω and the equation $g = 0$ is then defined as follows:

$$\begin{bmatrix} \nabla_{\tilde{u}} G(\tilde{u}^k, \omega^k) & \nabla_{\omega} G(\tilde{u}^k, \omega^k) \\ \nabla_{\tilde{u}} g(\tilde{u}^k, \omega^k) & \nabla_{\omega} g(\tilde{u}^k, \omega^k) \end{bmatrix} \begin{pmatrix} \Delta \tilde{u}^k \\ \Delta \omega^k \end{pmatrix} = - \begin{pmatrix} G(\tilde{u}^k, \omega^k) \\ g(\tilde{u}^k, \omega^k) \end{pmatrix} \tag{3.37}$$

One possible way to solve this extended linear system is to separate the extra column and row of the matrix related to the ∇_{ω} derivative and the g function:

$$\begin{aligned} x_1 = - & \quad \nabla_{\tilde{u}} G(\tilde{u}^k, \omega^k)^{-1} G(\tilde{u}^k, \omega^k) \\ x_2 = & \quad \nabla_{\tilde{u}} G(\tilde{u}^k, \omega^k)^{-1} \nabla_{\omega} G(\tilde{u}^k, \omega^k) \end{aligned} \tag{3.38}$$

The variables x_1 and x_2 are temporary variables which are both computed by solving with the matrix $\nabla_{\tilde{u}} G(\tilde{u}^k, \omega^k)$. The newton step in $\tilde{u}$ and ω are then computed as follows:

$$\begin{aligned} \Delta \omega^k &= \frac{-g(\tilde{u}^k, \omega^k) - dot\left(\nabla_{\tilde{u}} g(\tilde{u}^k, \omega^k), x_1\right)}{\nabla_{\omega} g(\tilde{u}^k, \omega^k) - dot\left(\nabla_{\tilde{u}} g(\tilde{u}^k, \omega^k), x_2\right)} \\ \Delta \tilde{u}^k &= x_1 - \Delta \omega^k x_2 \end{aligned} \tag{3.39}$$

This algorithm is a result of applying Gauss elimination of the last row the matrix from (3.37), then solving for the last variable which is $\Delta \omega^k$ and then substituting it into the remaining rows to solve for $\Delta \tilde{u}^k$. It is commonly called the bordered solve algorithm, as it performs Gauss elimination around the added border row and column of the matrix. The main benefit of this approach is that it allows for solving linear systems with matrix $\nabla_{\tilde{u}} G(\tilde{u}^k, \omega^k)$ rather than the extended one. That means that any parallel algorithm that can efficiently invert this matrix can still be used as is for the extended system when performing continuation. The drawback is that two solves (for two different right hand side vectors but same matrix) need to be performed for each Newton iteration (to obtain the temporary variables x_1 and x_2).

As mentioned previously, prediction directions other than the tangent are possible. Another popular option is the secant predictor, which is computed as a vector passing through the last 2 known solutions:

$$\tau^s = (\tilde{u}_i, \omega_i) - (\tilde{u}_{i-1}, \omega_{i-1}) \tag{3.40}$$

The obvious advantage of this predictor is its extremely low computational cost (a subtraction of two vectors instead of solving a linear system). The drawback can be worse accuracy of the predicted solution. An even simpler prediction method is the so called naive or natural prediction. It performs a step purely in frequency, keeping the previous solution as the predicted one:

$$\tau^n = (O, \pm 1) \tag{3.41}$$

this predictor might be used for simplicity in situations when the Newton solver doesn't have any issues finding the right solution and there is no need to differentiate between multiple possible solutions for one frequency. This predictor is not capable of handling FRC turning points (points where the curve changes direction in terms of ω).

3.7 Linear analysis and modes

Let's for a moment (the scope of this section) limit the equations of motion posed in (2.40) to their linear form, i.e.:

$$M\ddot{u} + D\dot{u} + Ku = F(t) \tag{3.42}$$

This means that when HBM is applied, the corresponding equations in frequency domain (3.15) are reduced to:

$$Z(\omega)\tilde{u} = \tilde{F} \tag{3.43}$$

Since Z is a block diagonal matrix, an excitation on a particular harmonic will only cause response on that same harmonic. Assuming only an excitation on the first harmonic, the problem can be solved using only the first harmonic and the Z matrix will therefore be only:

$$Z(\omega) = \begin{bmatrix} K - \omega^2 M & -\omega D \\ \omega D & K - \omega^2 M \end{bmatrix} \tag{3.44}$$

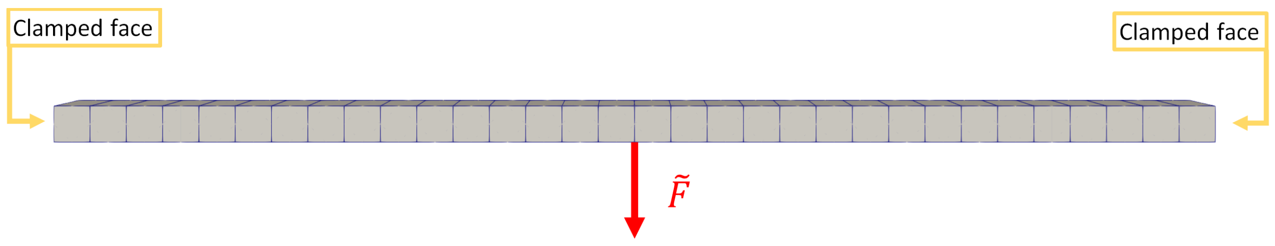


Figure 3.9: Computational testcase - a clamped-clamped beam with square profile and excitation force in the middle.

Let's use this simple linear HBM model to obtain FRC for a clamped-clamped beam excited in the middle (Figure 3.9). Without the need to go into specific values of various parameters, the general shape of the FRC will look similarly to that in Figure 3.11. Only one solution exists for each frequency ω and the amplitude of vibration is significantly higher around certain frequencies, called resonant or natural frequencies. These frequencies are typically computed by solving a following generalised eigenvalue problem:

$$Kv = \omega^2 Mv \tag{3.45}$$

with v being the corresponding eigenvector. These eigenvectors are the so called free vibration mode shapes (modes). One way of deriving them is to solve the following initial value problem:

$$\begin{aligned} Ku(t) + M\ddot{u}(t) &= 0 \\ u(0) &= u_0 \\ \dot{u}(0) &= 0 \end{aligned} \tag{3.46}$$

Just like in HBM, a harmonic solution is assumed:

$$\begin{aligned} u(t) &= Acos(\omega t) + Bsin(\omega t) \\ \dot{u}(t) &= -\omega Asin(\omega t) + \omega Bcos(\omega t) \\ \ddot{u}(t) &= -\omega^2 Acos(\omega t) - \omega^2 Bsin(\omega t) = -\omega^2 u(t) \end{aligned} \tag{3.47}$$

Taking into account the initial conditions, it turns out that $A = u_0$ and $B = O$. The actual solution is therefore in form:

$$u(t) = u_0 cos(\omega t) \tag{3.48}$$

Since the assumption (3.47) was made about the form of the solution, its validity needs to be verified by plugging it back into the original equation (3.46). This gives:

$$\left(Ku_0 - \omega^2 Mu_0\right) cos(\omega t) = 0 \tag{3.49}$$

Since this must be true for any time $t \geq 0$, it follows that:

$$Ku_0 - \omega^2 Mu_0 = 0 \tag{3.50}$$

Therefore, in order for (3.48) to be a valid solution, the pair (u_0, ω) needs to satisfy the above equation, which is the same as (3.45). This shows a property of the mode shapes. Without any damping and external forcing, the system will remain in the same periodic motion indefinitely, without gaining or loosing any energy.

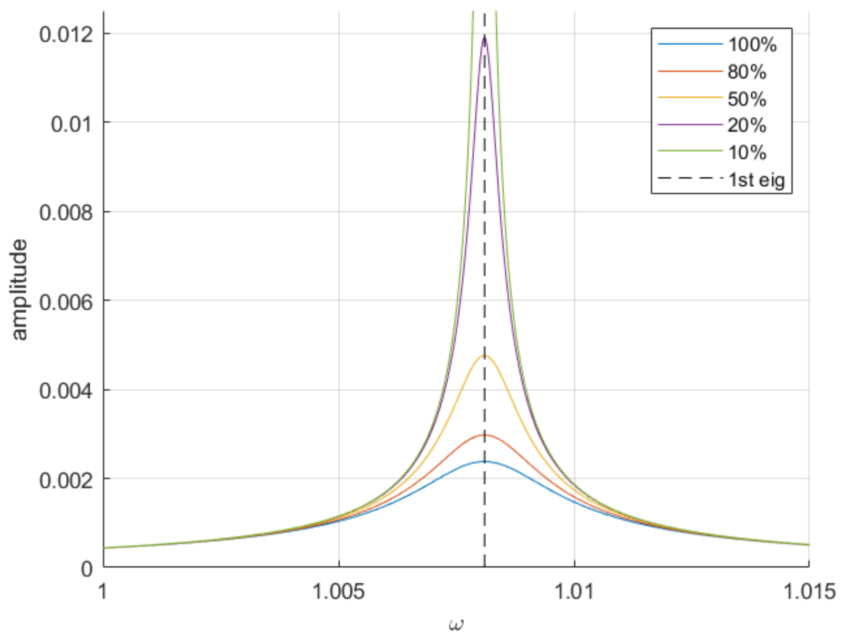


Figure 3.10: Demonstration of how reducing the damping increases (with inverse proportionality) the amplitude of the response. The legend describes the amount of damping with respect to a reference value.

One can further notice that without any damping present, the Z matrix looks as follows:

$$Z(\omega) = \begin{bmatrix} K - \omega^2 M & O \\ O & K - \omega^2 M \end{bmatrix} \tag{3.51}$$

The main diagonal blocks correspond to the expression in (3.50). Therefore, a mode shape v belongs to null space of such Z assembled for the corresponding resonance frequency ω_v. More precisely:

$$\begin{bmatrix} K - \omega_v^2 M & O \\ O & K - \omega_v^2 M \end{bmatrix} \begin{pmatrix} v \\ v \end{pmatrix} = \begin{pmatrix} O \\ O \end{pmatrix} \tag{3.52}$$

This means the matrix Z is singular for the resonant frequency. Therefore, a solution to (3.43) with no damping present might not exist for this frequency, depending on the character of the excitation. It is a well known property of matrices that $Im(A) \perp null(A^T)$ [13]. Taking advantage of the symmetry of Z, it follows that also $Im(A) \perp null(A)$. The cos and sin portions of the excitation vector $\tilde{F} \in Im(A)$ must therefore be orthogonal to the given mode shape vector in order for (3.43) to have a solution.

The presence of this singularity can be observed even with damping present. As the damping decreases, the amplitude of vibration at the resonant frequency increases with inverse proportionality. This can be seen in Figure 3.10. However, the presence of damping does eliminate this singularity, as long as the damping matrix doesn't have the given mode shape as its null vector as well.

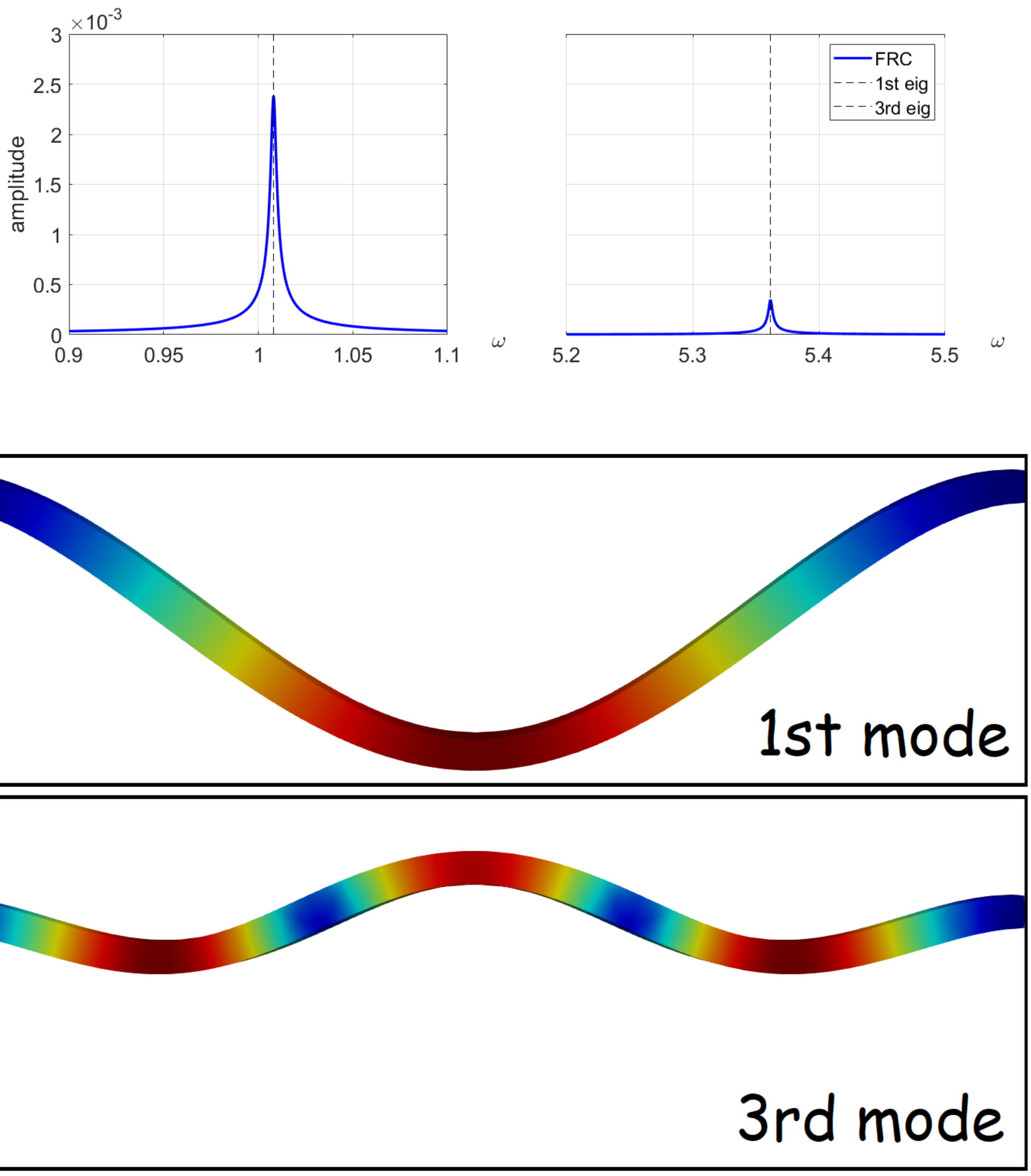


Figure 3.11: Example of a linear FRC for a clamped-clamped beam, with 2 responding modes highlighted.

3.8 Conclusion

This chapter builds on top of the theory laid out in the second chapter by introducing the harmonic balance method. First, state of the art of nonlinear vibration analysis and HBM in particular is overviewed. HBM is a well established method within the field as it allows modelling of steady state vibration without the need for expensive time stepping procedures. Coupling HBM with other tools such as parameter continuation, nonlinear models and efficient linear solvers allows for solving of complex nonlinear vibration analysis problems. State of the art of numerical solvers is also discussed as these are necessary components of computational modelling.

The discrete nonlinear equations of motion are then used as an entry point into the HBM formulation for steady state periodic motion. The frequency domain form of the equations of motion is derived. The AFT procedure using numerical integration for evaluation of nonlinear terms is explained. The nonlinear HBM equations are then used in context of a predictor-corrector type parameter continuation. For this, the equations are extended into the so called bordered system by adding the frequency as another variable and introducing an additional constraint equation. Naive, tangent and secant predictors are described. Standard Newton-Raphson nonlinear solver is used for the correction stage. This provides a sufficient toolset for computing nonlinear FRCs of a vibrating structure. In addition, a brief summary of linear modal analysis is provided as it will be useful in the following chapter.

Chapter 4

FETI for HBM

As discussed in the previous chapter, the HBM equation as shown in (3.19) can be fed into an existing linear solver of choice and be solved. The bordered system used for the continuation procedure can easily be built around this. This is one of the approaches that is also explored in this work. Another approach was developing an in house domain decomposition parallel linear solver based on the FETI method. This solver uses the existing theory of the FETI method and adapts it specifically for the nonlinear HBM problem. The mathematics of this solver is described in this chapter.

FETI, which stands for finite element tearing and interconnecting, is a well established domain decomposition method for solving systems of linear equations. It is designed in a way that allows for efficient parallel implementation of the algorithm. It was first introduced in [67] as a way to solve static linear elasticity problems. Since then it has been successfully used to solve various systems with billions of degrees of freedom.

This chapter first provides the state of the art of parallel linear solvers and the FETI method in particular. Next, the basic theory of the method and its fitting onto a nonlinear HBM problem is described. The HBM equations are derived in a domain decomposition manner. This form of the equations is then linearised for the Newton-Raphson solver. Preconditioning of the FETI form of the equations is discussed and an artificial coarse space is introduced for better convergence. Lastly, the equations are extended by the frequency variable to be used in the parameter continuation procedure.

4.1 State of the art

FETI is an algorithm that is well suited for applications on massively parallel computer architectures. This section therefore provides the state of the art overview not only for FETI but also for parallel linear solvers in general. A special section is also dedicated to research specifically in the area of parallelising the HBM equations or equations of similar structure.

4.1.1 Parallel linear solvers

Parallelism is an essential part of modern linear solvers. Parallel computing is a relatively young field which rose into prominence with the development of multiprocessor computer architectures. Current top of the line supercomputers have number of processing cores in the order of millions [53]. To utilise efficiently such architectures, parallel implementations of the linear solvers (and other related algorithms) had to be developed. A parallel algorithm utilises a set of threads or tasks that are capable of running independently of each other, each solving part of the whole problem. Common way to measure parallel efficiency is by assessing an algorithm's scalability [114]. However, other more general measures can be employed [194], taking into account multiple aspects such as reliability, cost efficiency, development time and energy consumption [186]. Many software libraries have been developed providing parallel implementations of various linear solvers, such as previously mentioned MUMPS, SuperLU or PARDISO. Packages offering extensive linear algebra functionalities in parallel have been developed, such as PETSc [10], Trilinos [173] and many others. Kim et al. [101] proposed a domain-wise parallel direct solver for large scale structural analysis.

Efficient handling of FE meshes is an important part of any large numerical model. Meca et al. [126] proposed an approach for parallel loading and preprocessing of unstructured meshes stored in sequential files. Gharbi et al. [57] presented a parallel mesh generation method that is well suited for domain decomposition methods.

4.1.2 Domain decomposition methods

Domain decomposition methods are a group of linear solvers that do not very well fit into the direct/iterative division. They are often called hybrid solvers as they utilise both a direct and an iterative solver in different parts of their solve process. They can also be viewed as preconditioners to iterative solvers. Domain decomposition solvers significantly gained in popularity with the arrival of parallel computing, as their formulation and structure is suitable for efficient parallelisation. An overview of domain decomposition methods can be found in [51, 55]. They can be divided into two categories - overlapping and nonoverlapping, depending on the overlap of the domains the methods create. The most popular overlapping method is the Schwarz method [165]. Several variants of this method have been developed over the years, such as the multiplicative Schwarz method [15], additive Schwarz [56] or restricted additive Schwarz method [86] which has been successfully used to solve problems of elasticity and Stokes systems of hundreds of millions of degrees of freedom. Nonoverlapping domain decomposition methods include among others the balancing domain decomposition (BDD) and the finite element tearing and interconnecting method (FETI), which was used in this work. The theory of BDD can be found in [121]. An advanced variant - balancing domain decomposition with constraints (BDDC) - was studied in [122, 123]. This method was also used for solving linear elasticity problems for hundreds of millions of degrees of freedom [9].

4.1.3 FETI method

FETI, which is one of the primary focuses of this work, was first proposed in [67] as a method to solve linear elastic problems. Convergence properties of the method were discussed in [65]. The basic idea of the method is to assume a separate problem defined for each domain with free boundary on domain interfaces and then enforce domain continuity via constraints using the Lagrange multiplier technique. Farhat and Mandel [64] applied FETI to 4th order plate bending problems. Cho et al [35] applied FETI for large scale nonlinear static structural multibody analysis, using a co-rotational formulation. FETI-DP (dual primal) was later developed. This method keeps exact domain continuity at certain parts of the mesh [61, 124]. Klawonn and Rheinbach [105] discussed an inexact FETI-DP variant. Lumba and Datta [120] applied FETI-DP as a part of their rotor aeroelastic solver X3D. Galliet and Cochelin [70] used FETI in combination with ANM to compute solutions of static nonlinear deformations with parametrised loading force. Total-FETI method which also uses Lagrange multipliers to enforce Dirichlet boundary condition was presented in [54] and its parallel implementation can be found in [110]. A hybrid FETI variant, which is a combination of classical FETI and FETI-DP was studied in [26, 106]. Paraschos and Vouvakis [135] proposed a FETI-DOP (dual overlapping primal) variant, using an auxiliary primal coarse problem that is fully overlapping. A diagram highlighting differences between some of the above mentioned FETI variants is in Figure 4.1. Recent massively parallel implementations of the FETI method, utilising the state of the art computing hardware, can be found in [85, 176]. Example of scalability results of FETI for billions of degrees of freedom can be seen in Figure 4.2. Cermak et al. [58] provided an insight into FETI preconditioning. Vašatová et al. [181] studied parallel assembly of the natural coarse space matrix.

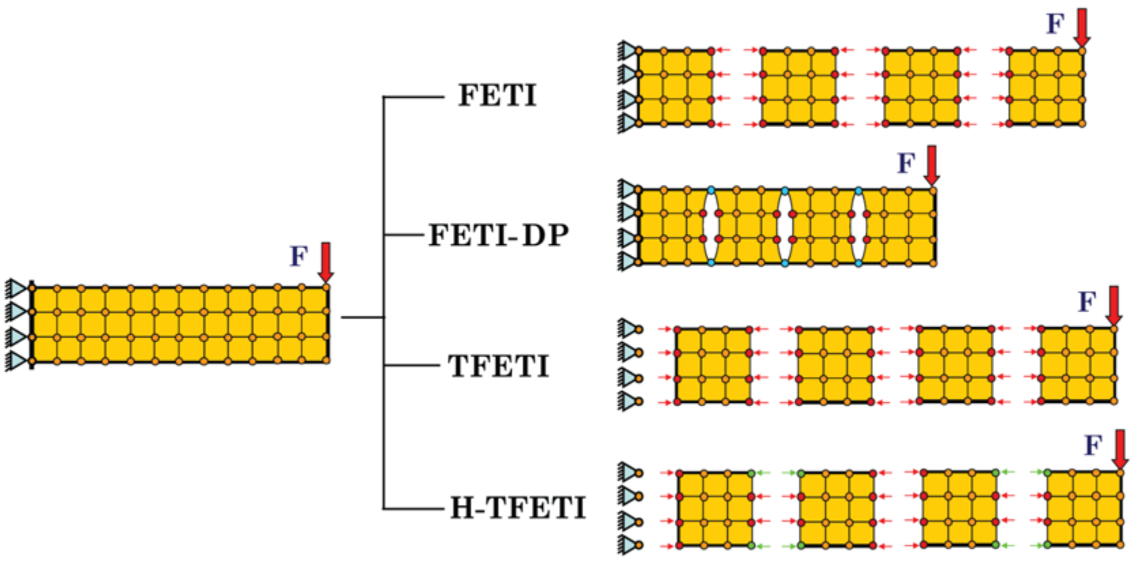


Figure 4.1: Diagram showing basic differences between some of the most prevalent variants of the FETI method [125]. The original FETI method from [65] is on top. Below that is FETI-DP which keeps domains connected by their corners by keeping the corner primal variables in the dual formulation [61]. Total-FETI (TFETI) method is similar to the standard FETI method but it also disconnects the Dirichlet boundary condition and adds Lagrange multipliers to enforce it [54]. This means all the domains are equal in terms of their set of rigid body modes. The final diagram shows the hybrid total-FETI (HTFETI) which combines TFETI and FETI-DP approaches [106].

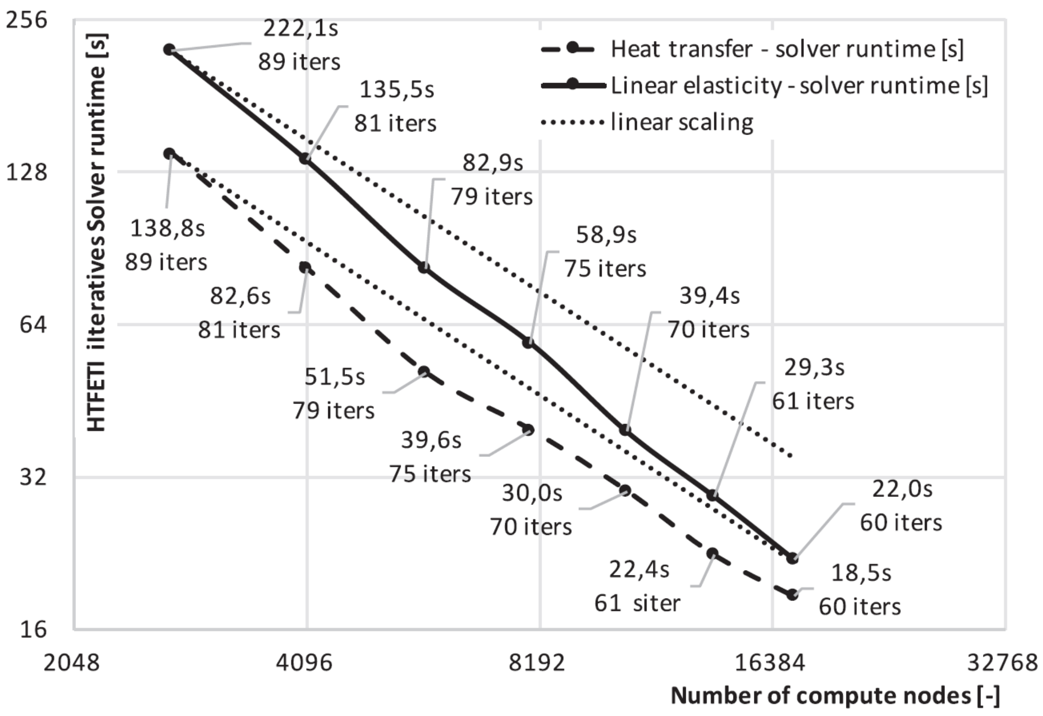


Figure 4.2: Results of the strong scalability test for heat transfer and linear elasticity problems with 20 and 11 billion unknowns respectively. Obtained with ESPRESO solver running on Titan supercomputer. Figure taken from [85]. The results demonstrate superlinear scaling of the solver by staying below the linear scaling marker line. This means that the solver scales better with increasing the number of CPUs than what is commonly called the ideal scalability. This does not mean that better scaling cannot be applied, but it is a good benchmark for the quality of scalability. It assumes that doubling the number of computational resources will half the required computational time.

One more class of linear solvers that should be mentioned are multigrid solvers. The basic idea behind multigrid solvers is to solve the system at hand multiple times on several coarse grids in a sequence in order to accelerate computations of different wavelengths of the solution. Multigrid solvers can also be used as preconditioners for other solvers. An overview of multigrid solvers can be found in [191]. Their parallel properties were studied in [18]. Many multigrid software packages have been developed, such as BoomerAMG [89] [59], ML [173] or AMGCL [46]. Blatt et al. [23] demonstrated use of a multigrid solver to solve Laplace and Poisson problems with around 1.34×10^{11} dofs.

4.1.4 Parallel HBM

Finally, the intersection of the topics discussed in previous sections and chapters is reviewed, meaning parallel solvers for nonlinear HBM equations, as this is the topic of this work. The HBM matrix is generally nonsymmetric and indefinite, meaning algorithms such as FETI, which in their original form assumed symmetric positive definite matrix, need to be modified. Cai and Widlund [27] studied domain decomposition algorithms for indefinite elliptic problems. Xu and Cai [193] studied preconditioning of GMRES for indefinite or nonsymmetric problems. Farhat et al. [60] proposed a variant of FETI called FETI-DPH for solving acoustic scattering problems. The algebraic equations from these problems resemble those of linear HBM equations. Farhat et al. [62] solved large scale linear mechanical vibration and nonlinear transient problems with FETI-DP. Farhat et al. [63] applied FETI-DP for shell vibration problems. Corral and Crespo Vaquerizo [40] used HBM to solve Navier-Stokes equations for periodic flows with a parallel multigrid solver. Sanghavi [161] studied FETI methods for acoustic problems with porous materials, which involves solving the Helmholtz equation. Kuether and Steyer [113] solved nonlinear HBM problems in mechanics with predictor-corrector continuation with GMRES preconditioned by inverting diagonal blocks of the system matrix in parallel. The largest testcase presented has 900k degrees of freedom. Ju and Zhu [93] computed FRCs for nonlinear mechanical vibration problems using HBM with shared memory parallelisation with up to 8 cores. Kononenko [109] analysed linear mechanical vibration response using a parallel simulation suite called ACE3P. Patil and Datta [137, 138] solved nonlinear HBM problems related to periodic rotor dynamics, using a parallel skyline solver and a direct sparse solver (MUMPS).

Lot of research on solving HBM equations has been done in modelling of electrical currents and electromagnetic fields. Garcia Bedoy Torres [71] developed a Python code for shared memory parallelisation of HBM for circuit simulations. Copeland and Langer [39] followed [193] and applied a domain decomposition technique to solve nonlinear eddy current problems using HBM. Dong and Li [52] solved nonlinear HBM equations for circuit simulations. Parallelisation is done by (partially) neglecting off diagonal coupling between different harmonics of HBM and solving a set of separate linear systems, one for each harmonic, in parallel. Li and Dong [117] further developed this technique. Yao et al. [195] applied FETI-DP to solve 3D nonlinear dynamic electromagnetic problems. It also assumed decoupling of different harmonics.

It can be seen that HBM is a popular tool for analysing vibration in various scientific fields. However, research around its parallel implementation for distributed nonlinearity in mechanical vibration problems seems to be limited in terms of maximum problem size. Several specifics of this field need to be accounted for. A predictor-corrector frequency continuation needs to be employed to obtain a full response curve. The distributed nonlinearity creates coupling between dofs of all harmonics within one physical degree of freedom. This coupling cannot be generally neglected. The system Jacobian matrix is also indefinite and nonsymmetric. Research specifically targeting the topic of efficient parallel implementations of HBM for mechanical vibration is therefore necessary.

4.2 Domain decomposition of the nonlinear HBM problem

FETI is a non-overlapping type of domain decomposition algorithm, meaning the mesh is decomposed into domains that don't overlap in volume, they only share common faces and nodes at the interfaces (see Figure 4.1 for illustration). Each domain covers an internally connected volume (meaning there are no separate 'islands' within one domain) and all domains together cover the entire mesh. The method can be seen as solving a constrained optimisation problem using the Lagrange multiplier technique.

Assuming a decomposition into $\mathcal{N}$ domains, the motion equations (3.15) are first assumed separately for each domain (indexed with i):

$$Z_i(\omega)\tilde{u}_i + \tilde{F}_{int}^{nl}(\tilde{u}_i) = \tilde{F}_i \tag{4.1}$$

with $\tilde{u}_i$ being the solution vector on all nodes of domain i, meaning these vectors will have overlaps on interfaces between neighbouring domains. The solution vector across all domains can be written into one vector as:

$$\tilde{u} = \begin{pmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \vdots \\ \tilde{u}_{\mathcal{N}} \end{pmatrix} \tag{4.2}$$

Note that this vector is not the same in terms of its content as the $\tilde{u}$ vector in the HBM sections, as this form of the solution vector contains multiple variables for the same dof for domain interface nodes. The notation is the same since it should be clear from the context which version of the vector is being used. No Dirichlet boundary condition is enforced in these domain equations of motion. The domains are treated as completely free and independent of each other. This means that solving (4.1) on each domain would not provide solution for the global problem (3.15). The Dirichlet boundaries, as well as connectivity of neighbouring domains, are added via additional constraint equation:

$$\sum_i B_i\tilde{u}_i = O \tag{4.3}$$

The structure of the B_i matrices will be discussed in a moment. First, a Lagrangian can be established:

$$L(\tilde{u}_i, \lambda) = f(\tilde{u}_i) - \lambda^T \left(\sum_i B_i\tilde{u}_i \right) \tag{4.4}$$

with scalar function f here being such that its gradient yields (4.1), meaning:

$$\nabla_{\tilde{u}_i} f(\tilde{u}_i) = Z_i(\omega)\tilde{u}_i + \tilde{F}_{int}^{nl}(\tilde{u}_i) - \tilde{F}_i \tag{4.5}$$

The argument $\tilde{u}_i$ means that the domain solution vectors are from all domains inserted as parameters into the function, i.e. $f(\tilde{u}_i) = f(\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_\mathcal{N})$. A stationary point $(\tilde{u}_i, \lambda)$ of this Lagrangian is then defined by equating its derivatives to zero, i.e.:

$$\begin{aligned} \frac{\partial L}{\partial \tilde{u}_i}(\tilde{u}_i, \lambda) &= O \\ \frac{\partial L}{\partial \lambda}(\tilde{u}_i, \lambda) &= O \end{aligned} \tag{4.6}$$

Evaluating the $\frac{\partial L}{\partial \tilde{u}_i}$ derivative yields:

$$Z_i(\omega)\tilde{u}_i + \tilde{F}_{int}^{nl}(\tilde{u}_i) = \tilde{F}_i + B_i^T \lambda \tag{4.7}$$

providing domain motion equations. The derivative $\frac{\partial L}{\partial \lambda}$ yields the constraint equation as already established in (4.3). Solving (4.7) together with (4.3) with unknowns being $\tilde{u}$ and λ is equivalent to solving the global HBM equations from (3.15) for the global solution vector $\tilde{u}$. The λ part of the FETI solution provides information about forces between domains on their interfaces. The displacement values in $\tilde{u}$ are called the primal variables, and variables in λ are called the dual variables. This Lagrange multiplier procedure is the basic principle of FETI and the reason for its name. The mesh is first torn into separate domains and then connected back together using the Lagrange multipliers λ. An illustration of this can be seen in Figure 4.3.

The domains are also separated from their Dirichlet boundaries, which are then also enforced by the constraint equations (4.3). This approach is known as total-FETI. This variant used for static analysis is described in [110]. This is different to the original FETI algorithm proposed in [67], which kept the Dirichlet boundary condition incorporated in a classical way into the domain matrices. The advantage of the total-FETI approach is that all domains are treated in the same way, regardless of presence of any Dirichlet boundary.

The structure of the B_i matrices that prescribe the constraint equations can be described as follows:

$$B_i = \begin{bmatrix} O & B_i^b \end{bmatrix} \tag{4.8}$$

with:

$$B_i^b = \begin{bmatrix} \pm 1 & O & \cdots & O \\ O & \pm 1 & \cdots & O \\ \vdots & & & \\ O & O & \cdots & \pm 1 \end{bmatrix} \tag{4.9}$$

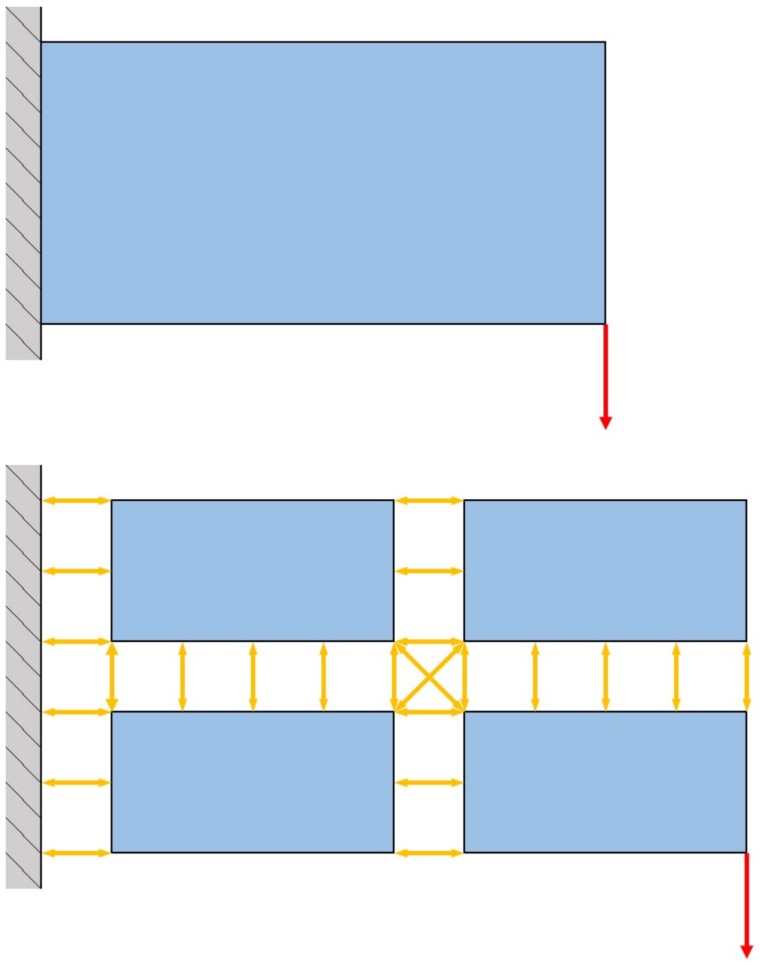


Figure 4.3: Diagram of total FETI domain decomposition. The blue rectangles represent individual domains that are connected together by the Lagrange multipliers on their interfaces (yellow arrows). The Dirichlet boundary condition is also enforced by them.

B_i^b is the boundary part of the B_i matrix, meaning those columns that correspond to boundary degrees of freedom in $\tilde{u}_i$. The ± 1 values are decided in a way so that the corresponding boundary dof gets multiplied by 1 on one domain and -1 on the other one. This ensures the continuity between domains by enforcing $\tilde{u}_i - \tilde{u}_j = O$ for boundary degrees of freedom. The B_i matrix additionally also enforces the Dirichlet boundary condition by including 1 for the corresponding dofs.

Matrices B_i can be also interpreted as a restriction of dofs of domain i to its interface. An interface of a domain is considered to be the set of primal dofs that are connected to dofs on other domains via the Lagrange multipliers λ. An interface between two domains then refers to the primal variables that are connected between those two domains, or the corresponding λ variables that connect them, depending on the context. Two domains are considered to be neighbours if an interface exists between them. In the opposite direction, multiplication by B_i^T inserts the interface dual variables to their appropriate positions in the domain primal variable vector. Since domain variables get mapped by B_i only to interface of

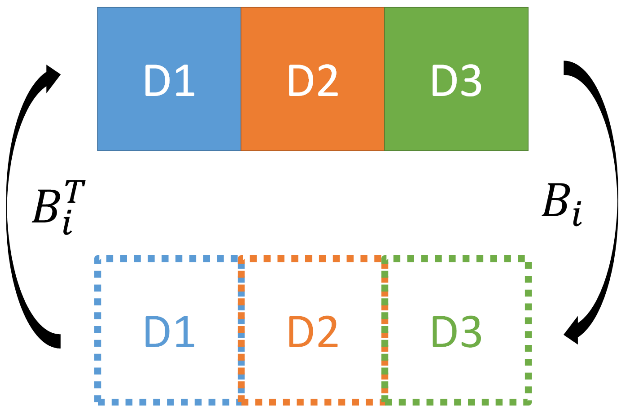


Figure 4.4: Effect of applying matrices B_i and B_i^T. No direct connection exists between domains 1 and 3 through the B matrices application, since they don't share an interface together. A connection between domain primal variables and interface dual variables exists only between neighbouring domains, those that share a common interface.

that domain and nowhere else, inversely, only the interface variables related to a particular domain will get mapped to that domain by B_i^T. This local relationship between domains and their interfaces is illustrated in Figure 4.4 and it will prove useful in the implementation of the FETI solver.

Before going further, few functions are introduced to simplify notation. Let's wrap the FETI domain equations of motion (4.7) into a function H_i:

$$H_i(\tilde{u}_i, \omega, \lambda) = Z_i(\omega)\tilde{u}_i + \tilde{F}_{int}^{nl}(\tilde{u}_i) - \tilde{F}_i - B_i^T\lambda = O \tag{4.10}$$

including the frequency ω as a variable also. This will be used later when the continuation procedure is applied. The derivative of H_i by $\tilde{u}_i$ is then:

$$\begin{aligned} \mathcal{A}_i(\tilde{u}_i, \omega, \lambda) &= \nabla_{\tilde{u}_i} H_i(\tilde{u}_i, \omega, \lambda) \\ &= Z_i(\omega) + \nabla_{\tilde{u}_i}\tilde{F}_{int}^{nl}(\tilde{u}_i) \end{aligned} \tag{4.11}$$

and its derivative by λ:

$$\nabla_\lambda H_i(\tilde{u}_i, \omega, \lambda) = -B_i^T \tag{4.12}$$

The domain connectivity equation (4.3) is wrapped into an H_c function:

$$H_c(\tilde{u}) = \sum_i B_i \tilde{u}_i = O \tag{4.13}$$

with its derivative by $\tilde{u}_i$ being:

$$\nabla_{\tilde{u}_i} H_c(\tilde{u}) = B_i \tag{4.14}$$

The derivative of H_c by λ is O.

Functions H_i over all domains can be grouped into one function H:

$$H(\tilde{u}, \omega, \lambda) = \begin{pmatrix} H_1(\tilde{u}_1, \omega, \lambda) \\ H_2(\tilde{u}_2, \omega, \lambda) \\ \vdots \\ H_{\mathcal{N}}(\tilde{u}_{\mathcal{N}}, \omega, \lambda) \end{pmatrix} = O \tag{4.15}$$

and its derivative by $\tilde{u}$:

$$\mathcal{A}(\tilde{u}, \omega, \lambda) = \nabla_{\tilde{u}} H(\tilde{u}, \omega, \lambda) = \begin{bmatrix} \mathcal{A}_1 & O & \cdots & O \\ O & \mathcal{A}_2 & \cdots & O \\ \vdots & & \ddots & \vdots \\ O & O & \cdots & \mathcal{A}_{\mathcal{N}} \end{bmatrix} \tag{4.16}$$

Using the notation established above, the complete FETI formulation of the nonlinear HBM equations of motion can be expressed as:

$$\begin{pmatrix} H(\tilde{u}_i, \omega, \lambda) \\ H_c(\tilde{u}_i) \end{pmatrix} = \begin{pmatrix} O \\ O \end{pmatrix} \tag{4.17}$$

4.3 Treatment of corners

Depending on the mesh decomposition, multiple (more than 2) domains can meet in certain locations of the mesh. There are multiple ways to prescribe the domain continuity constraint (4.3) for such degrees of freedom. One can pick one domain and connect all the other domains to it. Or run a 'chain' of connections from the first domain to the last. In principle, any variant that binds all the present domains together will work. Another way is to introduce an additional degree of freedom for the corner dofs and bind all dofs to that extra dof. This method is called a localised version of Lagrange multipliers and can be seen for example in [136], among other possible variants. For this work, a fully redundant set of Lagrange multipliers is used for all corners between multiple domains. This means that all domains are connected to all other domains. The choice of this method is based on [153], which states that this choice is necessary to be able to construct efficient preconditioners. It also proves convenient from the implementation standpoint, as all domains are treated equally. An example of such fully redundant domain connection can be seen in Figure 4.5. This approach of fully redundant connections is not applied to the Dirichlet boundary degrees of freedom. A degree of freedom that has a Dirichlet boundary condition imposed onto itself does not anymore connect to degrees of freedom of other domains connected in that point.

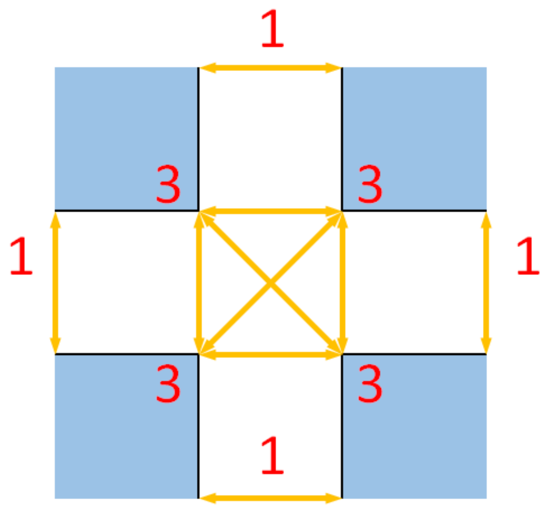


Figure 4.5: Multiple domains (blue areas) interfacing in a corner with fully redundant constraints. The red numbers represent number of Lagrange multipliers related to the corresponding primal degree of freedom. These numbers can be used for scaling the ± 1 values in the B matrices.

Additionally, a scaling of the values in the B matrices is introduced for the corner degrees of freedom. One can observe that for the corner degrees of freedom, the B_i matrix as defined in (4.8) and (4.9) will have multiple nonzero values in the corresponding column. The values are scaled in such a way that $(B_i^b)^T B_i^b = I$, with I being an identity matrix of an appropriate size. This means dividing the values in B_i for the corner degrees of freedom by the factor of $\sqrt{m}$, where m represents the number of connections for the given dof, i.e. the number of nonzero values in the particular column. The values of m are also highlighted in Figure 4.5. This scaling is discussed for instance in [58]. It can also be viewed as a form of preconditioning. Since there are multiple connections in the corner, their significance can be lowered to match the 'strength' of the connections between only 2 domains.

4.4 Linearised problem - Newton step

Considering ω to be a fixed parameter for a moment, Newton step formula for equations (4.17) can be formed analogously to (3.19). Denoting the Newton step as $(\Delta\tilde{u}, \Delta\lambda)$, it can be computed by solving:

$$\begin{bmatrix} \mathcal{A} & \nabla_\lambda H \\ \nabla_{\tilde{u}_i} H_c & O \end{bmatrix} \begin{pmatrix} \Delta\tilde{u} \\ \Delta\lambda \end{pmatrix} = - \begin{pmatrix} H \\ H_c \end{pmatrix} \tag{4.18}$$

omitting the function parameters for simpler notation. Expanding individual terms in the above yields:

$$\begin{bmatrix} \mathcal{A}_1 & O & \cdots & O & -B_1^T \\ O & \mathcal{A}_2 & \cdots & O & -B_2^T \\ \vdots & & \ddots & \vdots & \vdots \\ O & O & \cdots & \mathcal{A}_\mathcal{N} & -B_\mathcal{N}^T \\ B_1 & B_2 & \cdots & B_\mathcal{N} & O \end{bmatrix} \begin{pmatrix} \Delta\tilde{u}_1 \\ \Delta\tilde{u}_2 \\ \vdots \\ \Delta\tilde{u}_\mathcal{N} \\ \Delta\lambda \end{pmatrix} = - \begin{pmatrix} H_1 \\ H_2 \\ \vdots \\ H_\mathcal{N} \\ H_c \end{pmatrix} \tag{4.19}$$

By defining B to be the domain connectivity matrix covering all domains:

$$B^T = \begin{bmatrix} B_1^T \\ B_2^T \\ \vdots \\ B_\mathcal{N}^T \end{bmatrix} \tag{4.20}$$

the Newton step (4.18) can be rewritten as:

$$\begin{bmatrix} \mathcal{A} & -B^T \\ B & O \end{bmatrix} \begin{pmatrix} \Delta\tilde{u} \\ \Delta\lambda \end{pmatrix} = - \begin{pmatrix} H \\ H_c \end{pmatrix} \tag{4.21}$$

The above formula represents one iteration of Newton method where the steps $\Delta\tilde{u}$ and $\Delta\lambda$ are used to update the solution guess $\tilde{u}$ and λ. In FETI, this linearised problem is solved by first eliminating the primal variables from the equations by expressing:

$$\Delta\tilde{u} = \mathcal{A}^+ \left(-H + B^T\Delta\lambda\right) + \mathcal{R}\alpha \tag{4.22}$$

or in terms of one domain:

$$\Delta\tilde{u}_i = \mathcal{A}_i^+ \left(-H_i + B_i^T\Delta\lambda\right) + \mathcal{R}_i\alpha_i \tag{4.23}$$

where $\mathcal{A}_i^+$ is a pseudoinverse of $\mathcal{A}_i$ and:

$$\mathcal{A}^+ = \begin{bmatrix} \mathcal{A}_1^+ & O & \cdots & O \\ O & \mathcal{A}_2^+ & \cdots & O \\ \vdots & & \ddots & \vdots \\ O & O & \cdots & \mathcal{A}_\mathcal{N}^+ \end{bmatrix} \tag{4.24}$$

Note the presence of matrix $\mathcal{R}$ which stands for null space (a base of it in columns) of $\mathcal{A}$, as the domain matrices can possibly be singular. This is because the Dirichlet boundary condition is not included in the domain matrices and some domains might be without any Dirichlet boundary whatsoever, so the domains are left 'floating' without being attached anywhere. One has to therefore account for it by adding the null space to the solution. The null space is defined on each domain separately as:

$$\mathcal{A}_i \mathcal{R}_i = O \tag{4.25}$$

The domain null space bases $\mathcal{R}_i$ are then stacked diagonally into one matrix $\mathcal{R}$ across all domains:

$$\mathcal{R} = \begin{bmatrix} \mathcal{R}_1 & O & \cdots & O \\ O & \mathcal{R}_2 & \cdots & O \\ \vdots & & \ddots & \vdots \\ O & O & \cdots & \mathcal{R}_{\mathcal{N}} \end{bmatrix} \tag{4.26}$$

Additional vector of unknowns α is introduced which represents the vector of coefficients that multiply the null space vectors:

$$\alpha = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{\mathcal{N}} \end{pmatrix} \tag{4.27}$$

The step in primal solution $\Delta\tilde{u}$ as expressed in (4.22) is substituted into the second row of equations from (4.21) to obtain:

$$\begin{gathered} B\left[\mathcal{A}^+\left(-H + B^T\Delta\lambda\right) + \mathcal{R}\alpha\right] = -H_c \\ B\mathcal{A}^+B^T\Delta\lambda + B\mathcal{R}\alpha = B\mathcal{A}^+H - H_c \end{gathered} \tag{4.28}$$

Let's simplify the notation in (4.28) by denoting:

$$\mathcal{F} = B\mathcal{A}^+B^T = \sum_i B_i\mathcal{A}_i^+B_i^T \tag{4.29}$$

and:

$$\begin{aligned} \mathcal{G} = B\mathcal{R} &= [B_1\mathcal{R}_1, B_2\mathcal{R}_2, \cdots, B_{\mathcal{N}}\mathcal{R}_{\mathcal{N}}] \\ &= [\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_{\mathcal{N}}] \end{aligned} \tag{4.30}$$

and:

$$\mathcal{D} = B\mathcal{A}^{+}H - H_c = \sum_i \left(B_i\mathcal{A}_i^{+}H_i\right) - H_c \tag{4.31}$$

Equation (4.28) then yields:

$$\mathcal{F}\Delta\lambda + \mathcal{G}\alpha = \mathcal{D} \tag{4.32}$$

This formulation is called the dual problem, as it includes only the dual variables λ and the primal variables $\tilde{u}$ have been eliminated. The α vector is only temporary for the linear solve, it doesn't propagate across Newton iterations. Since a new set of variables α has been introduced, additional set of equations is also required to fully determine the whole system. Before applying the pseudoinverse $\mathcal{A}^{+}$ in (4.21), the domain equation reads:

$$\mathcal{A}_i\Delta\tilde{u} = -H_i + B_i^T\Delta\lambda \tag{4.33}$$

The right hand side of the above system must naturally belong to image of $\mathcal{A}_i$. A general linear algebra rule states that a matrix image is orthogonal to the null space of its transpose [13]. This implies the requirement:

$$\mathcal{L}_i^T\left(-H_i + B_i^T\Delta\lambda\right) = O \tag{4.34}$$

with $\mathcal{L}_i$ being the left null space $\mathcal{A}_i$, i.e. $\mathcal{A}_i^T\mathcal{L}_i = O$. The equation above is called the compatibility equation, as it enforces the compatibility of the right hand side in (4.33). An argument follows showing that $\mathcal{L}_i$ can be replaced with $\mathcal{R}_i$ as they are equal.

First, in a linear case $\mathcal{A}_i = Z_i$. It was discussed before (see Section 3.7) that for the nonzero harmonic blocks, the presence of damping removes mode shapes from the null space of the matrix. An assumption is made that no other null space vectors exist for these blocks. This is not a proof, but it has been verified by various testcases. The zero harmonic block is formed by the stiffness matrix K, which is symmetric and therefore has the same left and right null space. In particular, the basis of null space of K in 3 dimensions consists of 6 vectors - 3 translations along 3 coordinate axes, and 3 rotations around those same axes [26, 66]. Let's denote these two sets of 3 vectors r_T and r_R, i.e.:

$$null(K) = null(K^T) = \begin{bmatrix} r_T & r_R \end{bmatrix} \tag{4.35}$$

Taking all the above into consideration, it is assumed that for linear problems it holds that $null(Z_i) = null(Z_i^T)$, with the only null space being the null space of the matrix K in the 0th harmonic block (if present).

Second, in nonlinear case, the nonlinear term $\nabla_{\tilde{u}}\tilde{F}^{nl}_{int}$ is added to the Z matrix. The block structure of this term is shown in (3.26). Each block (in terms of harmonic coefficients) consists of a sum of several matrices K^{nl} multiplied by some coefficients. An assumption is made here that all these matrices, as well as their sum, have the same null space, regardless of what displacement they are evaluated for (as long as it's not zero). Specifically, the null space of any K^{nl} is the translation part of null space of the linear K. The rotation vectors are no longer present [66, 189, 190].

$$null(K^{nl}) = null({K^{nl}}^T) = \begin{bmatrix} r_T \end{bmatrix} \tag{4.36}$$

Based on the above, it is concluded that the null space of the complete domain matrix as defined in (4.11) is either both translation vectors r_T and rotation vectors r_R in a linear case or only the translation vectors r_T for a nonlinear case. In both cases, this null space is only present for the 0th harmonic part. If 0th harmonic is not used, no null space is present. Furthermore, because of (4.35) and (4.36), it is assumed that $null(\mathcal{A}_i) = null(\mathcal{A}_i^T)$. One can therefore replace the left null space with the right null space in (4.34), obtaining the compatibility equation in form:

$$\mathcal{R}_i^T \left(-H_i + B_i^T \Delta\lambda \right) = O \tag{4.37}$$

After rearranging:

$$\mathcal{G}_i^T \Delta\lambda = \mathcal{R}_i^T B_i^T \Delta\lambda = \mathcal{R}_i^T H_i = \mathcal{E}_i \tag{4.38}$$

Collectively for all domains, the compatibility equation reads:

$$\mathcal{G}^T \Delta\lambda = \begin{bmatrix} \mathcal{G}_1 \\ \mathcal{G}_2 \\ \vdots \\ \mathcal{G}_{\mathcal{N}} \end{bmatrix} \Delta\lambda = \begin{pmatrix} \mathcal{E}_1 \\ \mathcal{E}_2 \\ \vdots \\ \mathcal{E}_{\mathcal{N}} \end{pmatrix} = \mathcal{E} \tag{4.39}$$

Complete dual problem including the compatibility equation is then:

$$\begin{bmatrix} \mathcal{F} & \mathcal{G} \\ \mathcal{G}^T & O \end{bmatrix} \begin{pmatrix} \Delta\lambda \\ \alpha \end{pmatrix} = \begin{pmatrix} \mathcal{D} \\ \mathcal{E} \end{pmatrix} \tag{4.40}$$

This completes the transition of the linearised problem in Newton step from being defined in both primal and dual variables in (4.21) into its dual form.

4.5 Solving the linearised problem

The dual problem from (4.40) can be solved in many ways. The most simple one would be to assemble the problem matrix and use any available linear solver. However, this would not take advantage of the properties of the system and would be gravely inefficient. The standard way to solve the dual problem coming from the FETI method, adapted to the HBM structure of the domain matrices $\mathcal{A}_i$, is described in the following paragraphs. The entire process (including preconditioning which is discussed afterwards) is also summarised in Algorithm 1 at the end of this section.

The dual solution $\Delta\lambda$ is first split into 2 orthogonal components:

$$\Delta\lambda = \lambda_0 + \mathcal{P}\lambda_{12} \tag{4.41}$$

where $\lambda_0 \in Im(\mathcal{G})$ and $\mathcal{P}\lambda_{12} \in null(\mathcal{G}^T)$. The component λ_{12} is allowed to be in any space and the orthogonality to λ_0 is ensured by applying an orthogonal projection $\mathcal{P}$ in the form:

$$\mathcal{P} = I - \mathcal{G}(\mathcal{G}^T\mathcal{G})^{-1}\mathcal{G}^T \tag{4.42}$$

This projection is orthogonal to a projection:

$$\mathcal{Q} = \mathcal{G}(\mathcal{G}^T\mathcal{G})^{-1}\mathcal{G}^T \tag{4.43}$$

One can easily verify that $\mathcal{P}\mathcal{Q} = O$ and $\mathcal{P}\mathcal{G} = O$.

Since $\lambda_0 \in Im(\mathcal{G})$ and the other component $\mathcal{P}\lambda_{12} \perp Im(\mathcal{G})$, it follows that:

$$\begin{aligned}\mathcal{Q}\Delta\lambda &= \mathcal{Q}(\lambda_0 + \mathcal{P}\lambda_{12}) \\ &= \mathcal{Q}\lambda_0 \\ &= \lambda_0\end{aligned} \tag{4.44}$$

In other words, if the complete solution $\Delta\lambda$ is known, then the λ_0 component can be obtained by orthogonally projecting that solution onto $Im(\mathcal{G})$. This observation can be used in combination with the second line of equations in (4.40):

$$\begin{aligned}\lambda_0 &= \mathcal{Q}\Delta\lambda \\ &= \mathcal{G}(\mathcal{G}^T\mathcal{G})^{-1}\mathcal{G}^T\Delta\lambda \\ &= \mathcal{G}(\mathcal{G}^T\mathcal{G})^{-1}\mathcal{E}\end{aligned} \tag{4.45}$$

The image of $\mathcal{G}$ is called the natural coarse space of the problem in dual variables. It is created from null spaces of individual domains that were transferred into dual variables by the B_i matrices. Number of columns in $\mathcal{G}$ is determined by number of null space vectors across all domains. As discussed previously, the only null space comes from the 0th harmonic part of $\mathcal{A}_i$ and can have at most 6 vectors per one domain. This means that the matrix $\mathcal{G}^T\mathcal{G}$ inside the $\mathcal{P}$ and $\mathcal{Q}$ is relatively small compared to the size of the dual problem. Because of that, computing the natural coarse space solution component λ_0 in the way described in (4.45) requires inverting only this small matrix, makings this computation relatively low cost.

It remains to compute λ_{12} and α. The next step is to require that the residual of the first equation in (4.40) to be orthogonal to columns of $\mathcal{G}$. This is a type of a weak solve, similar in principle to the residual projection in (3.12):

$$\begin{aligned} \mathcal{G}^T(\mathcal{F}\Delta\lambda + \mathcal{G}\alpha - \mathcal{D}) &= O \\ \mathcal{G}^T\mathcal{F}\Delta\lambda + \mathcal{G}^T\mathcal{G}\alpha &= \mathcal{G}^T\mathcal{D} \end{aligned} \tag{4.46}$$

Rearranging to express α yields:

$$\begin{aligned} \mathcal{G}^T\mathcal{G}\alpha &= \mathcal{G}^T(\mathcal{D} - \mathcal{F}\Delta\lambda) \\ \alpha &= (\mathcal{G}^T\mathcal{G})^{-1}\mathcal{G}^T(\mathcal{D} - \mathcal{F}\Delta\lambda) \end{aligned} \tag{4.47}$$

hence obtaining an expression to compute α, provided that $\Delta\lambda$ is known, which at this point it is not. However, the α variable can now be eliminated by plugging the above expression for it back into the first line of (4.40) to obtain:

$$\mathcal{F}\Delta\lambda + \mathcal{G}(\mathcal{G}^T\mathcal{G})^{-1}\mathcal{G}^T(\mathcal{D} - \mathcal{F}\Delta\lambda) = \mathcal{D} \tag{4.48}$$

and after reorganising the terms:

$$\begin{aligned} (I - \mathcal{G}(\mathcal{G}^T\mathcal{G})^{-1}\mathcal{G}^T)\mathcal{F}\Delta\lambda &= (I - \mathcal{G}(\mathcal{G}^T\mathcal{G})^{-1}\mathcal{G}^T)\mathcal{D} \\ \mathcal{P}\mathcal{F}\Delta\lambda &= \mathcal{P}\mathcal{D} \end{aligned} \tag{4.49}$$

Using the fact that the λ_0 component of $\Delta\lambda$ is already known and noting the presence of the $\mathcal{P}$ projector on both sides the above can be rearranged into:

$$\mathcal{P}\mathcal{F}\mathcal{P}\lambda_{12} = \mathcal{P}(\mathcal{D} - \mathcal{F}\lambda_0) \tag{4.50}$$

Obtaining a system to be solved for unknown λ_{12}. Once λ_{12} is computed, the full dual solution $\Delta\lambda$ can be obtained, followed by computing α using (4.47). This completely solves the dual problem.

The reason why the dual problem is solved in this way has already been hinted. By solving first for λ_0, one can relatively quickly compute the low frequency part of the solution (frequency in spatial dimensions). The null space vectors of domain matrices $\mathcal{A}_i$ can be seen as eigenvectors attached to the lowest possible eigenvalue - 0. They can also be interpreted as vibration modes of (3.45) with $\omega = 0$ (with $K + K^{nl}$ instead of just K for nonlinear case). One can therefore see that the vibration frequency of the modes in time and spatial frequency of their deformations are related. See the modes in Figure 3.11 for comparison. The higher frequency mode also has more 'wavy' shape. The solution λ_0 therefore provides a coarse approximation (hence the naming coarse space) of the solution in dual variables across all domains. It is the rough estimate without the fine details.

When solving (4.50) for the other solution component λ_{12}, an iterative Krylov space solver is typically used. This is because an iterative solver doesn't require explicit assembly of the problem matrix, in this case $\mathcal{F}$. Only its multiplication with a vector needs to be implemented. Studies of FETI convergence show [151, 170] that Krylov space solvers typically struggle with establishing the 'global shape' of the solution, meaning finding the high wavelength solution components. On the other hand, they are fast to smooth out local low wavelength errors. Because of this property, having the coarse component λ_0 already computed removes this component from the solution. Solving for λ_0 in advance can also be seen as establishing the general global shape of the solution across all domains. The iterative solver then needs to only find the 'fine detail' solution λ_{12} for which the convergence will be much faster. This is what fundamentally makes FETI a powerful algorithm. The coarse solution λ_0 is first efficiently computed by inverting a small sized problem. Then, the fine solution part λ_{12} is again efficiently computed by the iterative solver thanks to the knowledge of λ_0.

At this point, the discussion about FETI could be finished. The algorithm described above is sufficient to solve the dual problem. However, as discussed previously, the only null space present in $\mathcal{G}$ is on the 0th harmonic portion of the frequency domain vectors. This means that also the natural coarse space solution only has nonzero values on the 0th harmonic positions. For other harmonics, no coarse space is effectively present, so the iterative solver will have to compute full solution for those harmonics, which will likely slow down the convergence. It is therefore natural to ask for another coarse space for the nonzero harmonics.

One can add an additional, so called artificial, coarse space $\mathcal{G}_c$ that is not mandatory but can improve convergence of the iterative solver used to solve (4.50). Such additional artificial coarse space is commonly used in the FETI2 method [76]. This coarse space is required to be orthogonal to the natural coarse space $\mathcal{G}$, i.e. $\mathcal{G}_c^T \mathcal{G} = O$. Discussion on how to achieve this orthogonality follows below in Section 4.7. The construction of this coarse space is analogous to the construction of the natural coarse space $\mathcal{G}$. Starting from domain vectors in primal variables $\mathcal{R}_{c,i}$ on each domain, one can assemble $\mathcal{G}_c$ as:

$$\begin{aligned}\mathcal{G}_c = B\mathcal{R}_c &= [B_1\mathcal{R}_{c,1}, B_2\mathcal{R}_{c,2}, \cdots, B_{\mathcal{N}}\mathcal{R}_{c,\mathcal{N}}] \\ &= [\mathcal{G}_{c,1}, \mathcal{G}_{c,2}, \cdots, \mathcal{G}_{c,\mathcal{N}}]\end{aligned} \tag{4.51}$$

Unlike the vectors in $\mathcal{R}_i$, which form the null space of $\mathcal{A}_i$, the vectors in $\mathcal{R}_{c,i}$ can be generally any vectors. One can then assume a further split of the solution λ_{12}:

$$\lambda_{12} = \lambda_2 + \mathcal{G}_c\beta \tag{4.52}$$

with $\mathcal{G}_c\beta$ representing the solution part on space generated by columns of $\mathcal{G}_c$. Next, the same weak solution procedure is applied with $\mathcal{G}_c$ to (4.50) as was done with $\mathcal{G}$ in (4.46):

$$\mathcal{G}_c^T\mathcal{P}\mathcal{F}\mathcal{P}(\lambda_2 + \mathcal{G}_c\beta) = \mathcal{G}_c^T\mathcal{P}(\mathcal{D} - \mathcal{F}\lambda_0) \tag{4.53}$$

Since $\mathcal{G}_c^T\mathcal{G} = O$, it follows that $\mathcal{G}_c^T\mathcal{P} = \mathcal{G}_c^T$ and also $\mathcal{P}\mathcal{G}_c = \mathcal{G}_c$, so the above can be simplified:

$$\mathcal{G}_c^T\mathcal{F}\mathcal{P}(\lambda_2 + \mathcal{G}_c\beta) = \mathcal{G}_c^T(\mathcal{D} - \mathcal{F}\lambda_0) \tag{4.54}$$

Expressing β:

$$\begin{aligned}\mathcal{G}_c^T\mathcal{F}\mathcal{G}_c\beta &= \mathcal{G}_c^T(\mathcal{D} - \mathcal{F}(\lambda_0 + \mathcal{P}\lambda_2)) \\ \beta &= (\mathcal{G}_c^T\mathcal{F}\mathcal{G}_c)^{-1}\mathcal{G}_c^T(\mathcal{D} - \mathcal{F}(\lambda_0 + \mathcal{P}\lambda_2))\end{aligned} \tag{4.55}$$

Substituting for β in (4.50) yields:

$$\mathcal{P}\mathcal{F}\mathcal{P}(\lambda_2 + \mathcal{G}_c(\mathcal{G}_c^T\mathcal{F}\mathcal{G}_c)^{-1}\mathcal{G}_c^T(\mathcal{D} - \mathcal{F}(\lambda_0 + \mathcal{P}\lambda_2))) = \mathcal{P}(\mathcal{D} - \mathcal{F}\lambda_0) \tag{4.56}$$

After rearrangement:

$$\begin{aligned}\mathcal{P}(\mathcal{F} - \mathcal{F}\mathcal{P}\mathcal{G}_c(\mathcal{G}_c^T\mathcal{F}\mathcal{G}_c)^{-1}\mathcal{G}_c^T\mathcal{F})\mathcal{P}\lambda_2 &= \mathcal{P}(\mathcal{D} - \mathcal{F}\lambda_0) - \mathcal{P}\mathcal{F}\mathcal{P}\mathcal{G}_c(\mathcal{G}_c^T\mathcal{F}\mathcal{G}_c)^{-1}\mathcal{G}_c^T(\mathcal{D} - \mathcal{F}\lambda_0) \\ \mathcal{P}(I - \mathcal{F}\mathcal{G}_c(\mathcal{G}_c^T\mathcal{F}\mathcal{G}_c)^{-1}\mathcal{G}_c^T)\mathcal{F}\mathcal{P}\lambda_2 &= \mathcal{P}(I - \mathcal{F}\mathcal{G}_c(\mathcal{G}_c^T\mathcal{F}\mathcal{G}_c)^{-1}\mathcal{G}_c^T)(\mathcal{D} - \mathcal{F}\lambda_0)\end{aligned} \tag{4.57}$$

Let's denote:

$$\mathcal{P}_c = I - \mathcal{F}\mathcal{G}_c(\mathcal{G}_c^T\mathcal{F}\mathcal{G}_c)^{-1}\mathcal{G}_c^T \tag{4.58}$$

It can be seen that this is a projection analogous to (4.42), involving $\mathcal{G}_c$ this time. However, this projection is not orthogonal like $\mathcal{P}$ as it includes the FETI operator $\mathcal{F}$. It is called a conjugate projection. The equation (4.57) then yields:

$$\mathcal{P}\mathcal{P}_c\mathcal{F}\mathcal{P}\lambda_2 = \mathcal{P}\mathcal{P}_c(\mathcal{D} - \mathcal{F}\lambda_0) \tag{4.59}$$

The equation (4.59), which is the equation (4.50) additionally projected by $\mathcal{P}_c$, is then solved with an iterative solver to obtain λ_2. After that, the last part of the $\Delta\lambda$ solution can be obtained:

$$\lambda_1 = \mathcal{G}_c\beta \tag{4.60}$$

with β being determined from (4.55). The reason to write λ_1 as $\mathcal{G}_c\beta$ was to emphasize the fact that λ_1 exists in the space spanned by columns of $\mathcal{G}_c$. The complete solution to the dual problem is then simply:

$$\Delta\lambda = \lambda_0 + \mathcal{P}(\lambda_1 + \lambda_2) \tag{4.61}$$

As before, once $\Delta\lambda$ is known, α can be computed using (4.47). This provides a complete solution to the dual problem stated in (4.40). Once the dual problem is solved, the primal solution step $\Delta\tilde{u}_i$ can be computed using (4.22). This completely solves one Newton iteration, providing the next solution guess $(\tilde{u}_i, \lambda)$.

As a final note, FETI for static elastic problems standardly uses the conjugate gradient method (CG) as the iterative solver to solve (4.50) or (4.59) [67, 76]. This is because $\mathcal{F}$ in those problems is symmetric positive definite. This is not the case here as the domain matrices $\mathcal{A}_i$ are generally indefinite and nonsymmetric. This property translates to the dual system matrix $\mathcal{F}$. Therefore, GMRES is used in this work instead of CG.

Algorithm 1 Newton step with FETI

```
1: Input: $\mathcal{A}_i$, $B_i$, $H_i$, $H_c$

2: Factorise $\mathcal{A}_i$
3: $\mathcal{F} = \sum_i B_i \mathcal{A}_i^+ B_i^T$    ▷ Dual problem operator
4: $\mathcal{D} = \sum_i \left(B_i \mathcal{A}_i^+ H_i\right) - H_c$    ▷ Dual problem right hand side

5: $\mathcal{R}_i = null\left(\mathcal{A}_i\right)$    ▷ Compute domain matrix null space
6: $\mathcal{G} = [\mathcal{B}_1\mathcal{R}_1, \mathcal{B}_2\mathcal{R}_2, \ldots, \mathcal{B}_\mathcal{N}\mathcal{R}_\mathcal{N}]$
7: Assemble and factorise $\mathcal{G}^T\mathcal{G}$
8: $\mathcal{P} = I - \mathcal{G}\left(\mathcal{G}^T\mathcal{G}\right)^{-1}\mathcal{G}^T$    ▷ Orthogonal projection

9: $\mathcal{R}_{c,i}$ = CreateArtificialCS()    ▷ Establish artificial coarse space
10: $\mathcal{G}_c = [\mathcal{B}_1\mathcal{R}_{c,1}, \mathcal{B}_2\mathcal{R}_{c,2}, \ldots, \mathcal{B}_\mathcal{N}\mathcal{R}_{c,\mathcal{N}}]$
11: Assemble and factorise $\mathcal{G}_c^T\mathcal{F}\mathcal{G}_c$
12: $\mathcal{P} = I - \mathcal{F}\mathcal{G}_c\left(\mathcal{G}_c^T\mathcal{F}\mathcal{G}_c\right)^{-1}\mathcal{G}_c^T$    ▷ Conjugate projection

13: $\mathcal{E} = \left((\mathcal{R}_1^T H_1)^T, (\mathcal{R}_2^T H_2)^T, \ldots, (\mathcal{R}_\mathcal{N}^T H_\mathcal{N})^T\right)^T$

14: if isPreconditionerDirichlet then
15:     $\mathcal{S}_i^{bb} = \mathcal{A}_i^{bb} - \mathcal{A}_i^{bi}\left(\mathcal{A}_i^{ii}\right)^{-1}\mathcal{A}_i^{ib}$    ▷ Compute domain matrix Schur complement
16:     $\mathcal{S}_i = \begin{bmatrix} O & O \\ O & \mathcal{S}_i^{bb} \end{bmatrix}$
17:     $\mathcal{F}' = \sum_i B_i \mathcal{S}_i B_i^T$
18: else if isPreconditionerLumped then
19:     $\mathcal{F}' = \sum_i B_i \mathcal{A}_i B_i^T$
20: end if

21: $\lambda_0 = \mathcal{G}\left(\mathcal{G}^T\mathcal{G}\right)^{-1}\mathcal{E}$    ▷ Compute solution on natural coarse space

22: GMRES_Op = $\mathcal{P}\mathcal{P}_c^T\mathcal{F}'\mathcal{P}\mathcal{P}_c\mathcal{F}\mathcal{P}$
23: GMRES_Rhs = $\mathcal{P}\mathcal{P}_c^T\mathcal{F}'\mathcal{P}\mathcal{P}_c(\mathcal{D} - \mathcal{F}\lambda_0)$
24: $\lambda_2$ = GMRES(GMRES_Op, GMRES_Rhs)    ▷ Solve the projected dual problem

25: $\beta = (\mathcal{G}_c^T\mathcal{F}\mathcal{G}_c)^{-1}\mathcal{G}_c^T(\mathcal{D} - \mathcal{F}(\lambda_0 + \mathcal{P}\lambda_2))$
26: $\lambda_1 = \mathcal{G}_c\beta$    ▷ Compute solution on artificial coarse space
27: $\Delta\lambda = \lambda_0 + \mathcal{P}\left(\lambda_1 + \lambda_2\right)$    ▷ Compute complete dual solution

28: $\alpha = (\mathcal{G}^T\mathcal{G})^{-1}\mathcal{G}^T(\mathcal{D} - \mathcal{F}\Delta\lambda)$
29: $\Delta\tilde{u}_i = \mathcal{A}_i^+\left(-H_i + B_i^T\Delta\lambda\right) + \mathcal{R}_i\alpha_i$    ▷ Compute primal solution

30: Output: $\Delta\tilde{u}_i$, $\Delta\lambda$
```

4.6 Preconditioning

Preconditioning is a common way to improve performance of Krylov based iterative solvers. Their goal is to transform the linear system to lower the condition number of the matrix, which can significantly improve the rate of convergence of the solver [139]. An ideal preconditioner would be the inverse of the system matrix, which is however not efficient to compute. A preconditioner is therefore in some sense an approximation of the matrix inverse, such that it is computationally cheap to obtain.

In FETI, a preconditioner operator $\mathcal{F}'$ can be used to improve convergence when solving the projected dual problem (4.59). There are 2 preconditioners that are commonly used in FETI for static elasticity problems [58] to improve convergence of the projected dual problem solver. These preconditioners are also explored in this work.

The first option is the so called lumped preconditioner. This preconditioner takes form of:

$$\mathcal{F}' = \sum_i B_i \mathcal{A}_i B_i^T \tag{4.62}$$

The lumped preconditioner can be interpreted as approximating the inverse of a sum by sum of inverses, since $\mathcal{F} = \sum_i B_i A^+ B_i^T$. The second preconditioner is called Dirichlet and it reads:

$$\mathcal{F}' = \sum_i B_i \mathcal{S}_i B_i^T \tag{4.63}$$

where $\mathcal{S}_i = \begin{bmatrix} O & O \\ O & \mathcal{S}_i^{bb} \end{bmatrix}$. The $\mathcal{S}_i^{bb}$ block represents the Schur complement of the domain matrix $\mathcal{A}_i$ on the domain boundary degrees of freedom, meaning:

$$\mathcal{S}_i^{bb} = \mathcal{A}_i^{bb} - \mathcal{A}_i^{bi} (\mathcal{A}_i^{ii})^{-1} \mathcal{A}_i^{ib} \tag{4.64}$$

where b and i superscripts indicate boundary and internal primal domain degrees of freedom respectively, for rows and columns (in that order). The block division of $\mathcal{S}_i$ is only symbolic, as in practice the structure will depend on order of dofs on the domain.

It should be noted that the mathematical expressions for both of the preconditioners follow the pattern of the expression for $\mathcal{F}$. They are both computed as a sum of partial results computed on each domain. This is important for efficient parallelism. While the lumped preconditioner is computationally cheaper to obtain, the results in [58] show that the Dirichlet preconditioner achieves better reduction of number of iterations in the iterative solver. Both preconditioners will be assessed in later sections.

Given the operator form of the projected dual problem, the preconditioner $\mathcal{F}'$ can simply be applied from the left:

$$\mathcal{F}' \mathcal{P} \mathcal{P}_c \mathcal{F} \mathcal{P} \lambda_2 = \mathcal{F}' \mathcal{P} \mathcal{P}_c (\mathcal{D} - \mathcal{F} \lambda_0) \tag{4.65}$$

However, it was observed experimentally (see results in Section 6.4.3) that adding a second set of projections $\mathcal{P}\mathcal{P}_c^T$ after the preconditioner improves convergence of GMRES:

$$\mathcal{P}\mathcal{P}_c^T\mathcal{F}'\mathcal{P}\mathcal{P}_c\mathcal{F}\mathcal{P}\lambda_2 = \mathcal{P}\mathcal{P}_c^T\mathcal{F}'\mathcal{P}\mathcal{P}_c(\mathcal{D} - \mathcal{F}\lambda_0) \tag{4.66}$$

This version of the projected dual problem was used for all results in this work (except when no preconditioner was used). This means that another set of $\mathcal{P}$ and $\mathcal{P}_c$ projections needs to be performed, with the latter transposed. The form of the equations above was reached during discussions of this topic with professor Rixen [152]. The optimal form of applying the preconditioner to the projected dual problem requires a deeper mathematical analysis. This was not possible to conduct during the time scope of this project. Therefore, the equation (4.66) is presented as is without a thorough explanation. Alternatively, the simpler version of the equations (4.65) is also valid and can be successfully used.

4.7 Choice of artificial coarse space

So far, nothing has been said about the form of the artificial coarse space $\mathcal{G}_c$. The only requirement set for it is that $\mathcal{G}_c^T\mathcal{G} = O$. It should be noted that this requirement is not strictly necessary. One can achieve this orthogonality by first projecting any arbitrary $\mathcal{G}_c$ by $\mathcal{P}$, i.e. picking:

$$\bar{\mathcal{G}}_c = \mathcal{P}\mathcal{G}_c \tag{4.67}$$

then:

$$\bar{\mathcal{G}}_c^T\mathcal{G} = \mathcal{G}_c^T\mathcal{P}^T\mathcal{G} = \mathcal{G}_c^T(\mathcal{G} - \mathcal{G}) = O \tag{4.68}$$

However, this approach might cause a problem in terms of parallelisation of the algorithm. It is therefore better to pick $\mathcal{G}_c$ such that it already satisfies the orthogonality condition and whose vectors are well localised, i.e. spanning only several neighbouring interfaces in the mesh.

As it was previously argued (see Sections 3.7 and 4.4), the null space of the domain matrices $\mathcal{A}$ is restricted only to the 0th harmonic, if that is present in the system. The B_i matrices do not change this, so all the vectors in $\mathcal{G}$ have nonzero values only at their 0th harmonic positions. This means that any vector in $\mathcal{G}_c$ that is zero in its 0th harmonic will trivially satisfy the orthogonality.

A natural idea is to use the same vectors for the artificial coarse space that are used for the natural coarse space, i.e. expand the null space of the 0th harmonic block of the domain matrix to other harmonics, as those are loosely speaking the 'coarsest vectors available'. Since the vectors in $\mathcal{G}_c$ do not need to be in any relation to the actual null space of the 0th block, one can pick the entire 6 vectors that form the null space of the linear stiffness matrix K, regardless of the problem being possibly nonlinear. Denoting:

$$r_i = null(K_i) = \begin{bmatrix} r_{T,i} & r_{R,i} \end{bmatrix} \tag{4.69}$$

with $r_{T,i}$ and $r_{R,i}$ being the translation and rotation parts respectively. The entire artificial coarse space in primal variables can then look like this:

$$\mathcal{R}_{c,i} = \begin{array}{c} \begin{bmatrix} O & O & \cdots & O & O \\ r_i & O & \cdots & O & O \\ O & r_i & \cdots & O & O \\ \vdots & & & & \\ O & O & \cdots & r_i & O \\ O & O & \cdots & O & r_i \end{bmatrix} \end{array} \begin{array}{c} \text{0th harmonic - no artificial CS} \\ \text{1st harmonic cosine part} \\ \text{1st harmonic sine part} \\ \\ \text{m-th harmonic cosine part} \\ \text{m-th harmonic sine part} \end{array} \tag{4.70}$$

$\mathcal{G}_c$ is then computed as shown in (4.51).

This choice of $\mathcal{G}_c$ is used in this work for several reasons. First, this coarse space is computationally cheap to assemble and can be evaluated independently on each domain. Second, no better options were discovered during this research. Adding several lowest frequency modes of the individual domains was tested but with no observable convergence improvements. This was done by computing the free vibration modes on each domain using the domain K_i and M_i matrices (see (3.50)). Another artificial coarse space that was tested is a coarse space proposed in [60]. This wave-based coarse space attempts to recognise the specific character of the Helmholtz equation which in its structure resembles the HBM problem. However, difficulties implementing this coarse space were encountered and this option was also dropped. This is not a definitive conclusion about any of the unsuccessfully tested options. A more thorough research and tuning of those options could reveal that these options are viable candidates. This work is nevertheless limited to the artificial coarse space described in (4.70).

Additionally, $\mathcal{G}_c$ can also include vectors that are nonzero on the 0th harmonic. Here one must be careful to satisfy the orthogonality condition. The idea used in this work is to use the rotation part of K null space once the system becomes nonlinear and the rotations are no longer part of K^{nl} null space. However, these rotations cannot be used in the form in which they appear in the null space of K. While they are orthogonal to translations on the same domain, they are not orthogonal to translations of the neighbouring domains. This can be seen in Figure 4.6. A way to resolve this issue for a regular hexahedral mesh is to construct separate rotation vectors for each interface between 2 domains. These rotations

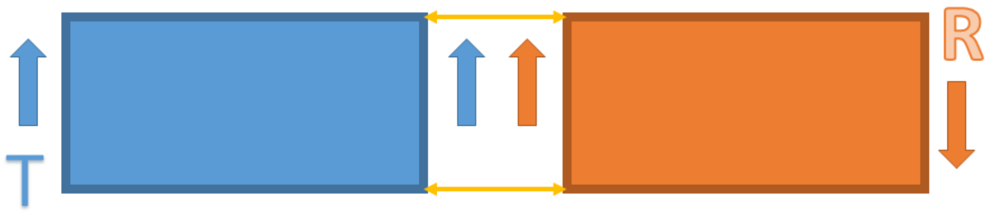


Figure 4.6: Translation and rotation of neighbouring domains. The right boundary of the blue domain and the left boundary of the orange domain correspond to the same degrees of freedom in dual variables λ.

are constructed around centres of the given interfaces, and restricted only to those interfaces. i.e. the vectors r_i are zeroed out everywhere outside the interface dofs. Furthermore, the rotations also have to be zeroed out at the edges of the interfaces when there is a corner between more than 2 domains. These modified rotations are shown in Figure 4.7. If a domain interface is too small (in terms of number of elements it covers in any direction) and the rotations need to be restricted because of the corners so much that not even width of a single element would be covered by it, that rotation vector is omitted from the coarse space completely.

4.8 Continuation

The continuation algorithm as presented in Section 3.6 requires additional attention when combined with FETI. This is because the set of unknowns in the HBM problem was extended by adding the Lagrange multipliers λ. When frequency ω is assumed as another unknown, the complete set of variables required to describe a solution point to (4.7) on an FRC is $(\tilde{u}, \lambda, \omega)$. This means that an FRC also exists in the λ variables and so those need to be followed by the continuation loop as well. Assuming a prediction direction $\tau = (\tau_{\tilde{u}_i}, \tau_\lambda, \tau_\omega)$ and a predicted solution guess $(\tilde{u}_i^p, \lambda^p, \omega^p)$, the pseudo arc-length constraint function g is analogous to the one defined in (3.34):

$$g(\tilde{u}_i, \lambda, \omega) = dot((\tilde{u}_i, \lambda, \omega) - (\tilde{u}_i^p, \lambda^p, \omega^p), \tau) \tag{4.71}$$

the lower index i identifying index of the Newton iteration was dropped to not be confused with domain index for $\tilde{u}_i$. Combining the standard newton step using the FETI algorithm as defined in (4.21) with the pseudo arc-length correction extension as done in (3.37), one obtains:

$$\begin{bmatrix} \mathcal{A} & -B^T & \nabla_\omega H \\ B & O & \nabla_\omega H_c \\ \nabla_{\tilde{u}} g & \nabla_\lambda g & \nabla_\omega g \end{bmatrix} \begin{pmatrix} \Delta\tilde{u} \\ \Delta\lambda \\ \Delta\omega \end{pmatrix} = - \begin{pmatrix} H \\ H_c \\ g \end{pmatrix} \tag{4.72}$$

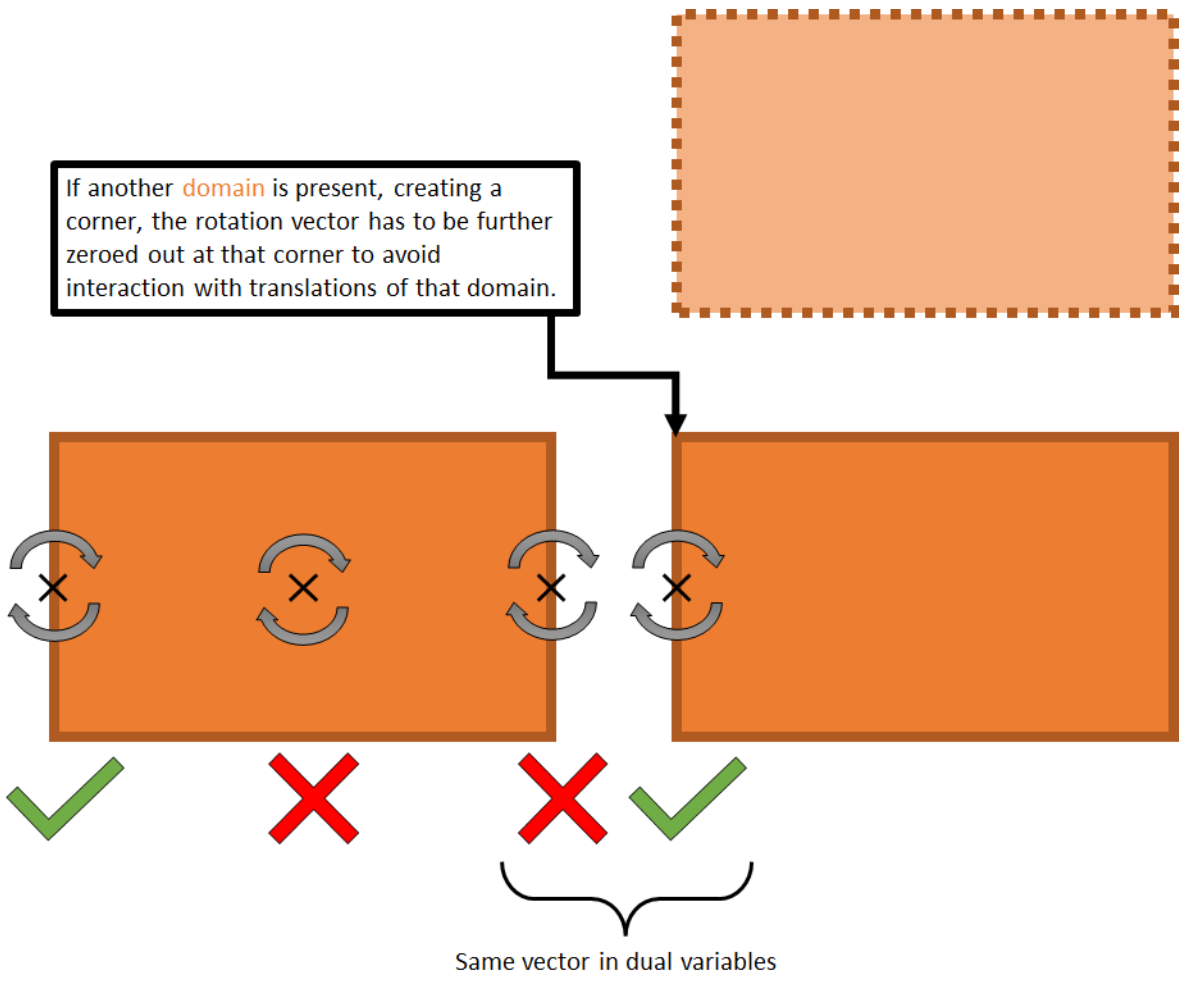

If another domain is present, creating a corner, the rotation vector has to be further zeroed out at that corner to avoid interaction with translations of that domain.
Same vector in dual variables

Figure 4.7: Construction of domain interface rotations for artificial coarse space on 0th harmonic for a regular mesh. Instead of rotating around the centre of the domain, for each individual interface between 2 domains, a rotation of that interface is constructed around its centre. The rotation is restricted only to dofs of that interface and has zeros everywhere else in the domain. Zeroing out at the edges of the interfaces is further performed if there is a corner between multiple domains. This ensures orthogonality of all such rotations to all translation vectors in $\mathcal{G}$.

with the function parameters as well as the upper k index omitted for notation simplification purposes. Noting that $\nabla_\omega H_c = O$ as H_c does not depend on ω and using the pseudo arc-length constraint function g defined in (4.71), the above simplifies to:

$$\begin{bmatrix} \mathcal{A} & -B^T & \nabla_\omega H \\ B & O & O \\ \tau_{\tilde{u}} & \tau_{\tilde{\lambda}} & \tau_\omega \end{bmatrix} \begin{pmatrix} \Delta\tilde{u} \\ \Delta\lambda \\ \Delta\omega \end{pmatrix} = - \begin{pmatrix} H \\ H_c \\ g \end{pmatrix} \tag{4.73}$$

Applying the same bordered solve algorithm as in (3.38) and (3.39), one obtains:

$$\begin{bmatrix} \mathcal{A} & -B^T \\ B & O \end{bmatrix} \begin{pmatrix} x_{1,\tilde{u}} \\ x_{1,\lambda} \end{pmatrix} = - \begin{pmatrix} H \\ H_c \end{pmatrix}$$
$$\begin{bmatrix} \mathcal{A} & -B^T \\ B & O \end{bmatrix} \begin{pmatrix} x_{2,\tilde{u}} \\ x_{2,\lambda} \end{pmatrix} = \begin{pmatrix} \nabla_\omega H \\ O \end{pmatrix} \tag{4.74}$$

Note that both of the temporary variables x_1 and x_2 can be obtained by solving the standard linear system with the FETI structure as defined in (4.21). After that, the Newton steps can be obtained:

$$\begin{aligned} \Delta\omega &= \frac{-g - dot((\tau_{\tilde{u}}, \tau_\lambda), x_1)}{\tau_\omega - dot((\tau_{\tilde{u}}, \tau_\lambda), x_2)} \\ (\Delta\tilde{u}, \Delta\lambda) &= x_1 - \Delta\omega x_2 \end{aligned} \tag{4.75}$$

In case of tangent predictor, same approach as in (3.31) can be applied, but using the FETI solver. This would yield a linear system:

$$\begin{bmatrix} \mathcal{A} & -B^T \\ B & O \end{bmatrix} \begin{pmatrix} \tau_{\tilde{u}} \\ \tau_\lambda \end{pmatrix} = \begin{pmatrix} -\nabla_\omega H \\ O \end{pmatrix} \tag{4.76}$$

with the matrix $\mathcal{A}$ and vector H evaluated for an already computed solution $(\tilde{u}_i, \lambda, \omega)$ and $(\tau_{\tilde{u}}, \tau_\lambda)$ being parts of the tangential vector in $\tilde{u}_i$ and λ at that solution point.

4.9 Conclusion

This chapter provides the necessary theoretical background for the FETI method and its application on the nonlinear HBM equations. State of the art of parallel linear solvers in general is first overviewed. Many existing linear solvers have been parallelised and many new solvers have been developed specifically to benefit from the new massively parallel computer platforms. This opened possibilities for solving numerical problems at a completely new scale. State of the art of the FETI method in particular is then reviewed. The method has become a popular domain decomposition linear solver since its introduction in 1991 and it has since proven to be an efficient solver. In the realm of static elasticity problems it achieved impressive scalability on problems reaching sizes of billions of degrees of freedom.

Following the state of the art overview, the formulation of the FETI algorithm for the HBM equations is introduced. It follows the well established theory of the FETI method used in static elasticity problems. The mesh is decomposed into domains and interface continuity is enforced using Lagrange multipliers. The dual equations are derived using the standard procedure of eliminating the primal variables and introducing the domain rigid body modes (null space of domain matrices). The total-FETI approach is used. This variant also detaches all domains from their Dirichlet boundary. An argument is made about the nature of the domain matrices null space. The standard orthogonal projection on the natural coarse space is performed to obtain the portion of a solution on the coarse space. Then, the additional artificial coarse space is introduced to improve convergence for the nonzero harmonic components of the solution. This coarse space can be extended to also further improve convergence on the 0th harmonic. The properties of the artificial coarse space and its possible forms are discussed. GMRES is the iterative solver chosen to solve the projected dual equations for the fine portion of the dual solution.

Other essential aspects of the solver such as treatment of corners between multiple domains and preconditioning are addressed. The FETI formulation of the linearised HBM equations inside a Newton iteration is put into context with the bordered system used in the continuation procedure.

Chapter 5

Code

A large portion of this work was dedicated to developing a code in which the theory presented above could be properly tested. Early in the study process, a decision was made to develop a custom code from scratch instead of building upon an existing library. There are libraries that could have served as a good starting point, such as the ESPRESO library from IT4I [84], which has proven to be very efficient in its FETI algorithm implementation. However, a new custom code was developed because it offered more flexibility and opportunity to tailor it more around the HBM problem structure, as well as a good opportunity for personal skill development of the author. The code is publicly available at Gitlab [21].

The code was developed in C++ under the Linux environment, using extensively the object oriented programming principles. It has been given working name ParHBM, standing simply for parallel harmonic balance method. Emphasis was put on code modularity so that various parts can potentially be replaced by different implementations. While the code is in house developed from scratch, it does employ many 3rd party libraries to handle many tasks such as linear algebra, solving linear systems, graphical output and meshing. All of the used libraries are well established open source packages with good support. The focus of the development was therefore mostly on the harmonic balance method, continuation and the FETI algorithm.

For parallelism, the message passing interface (MPI) was used. This is a common tool used for creating massively parallel applications, i.e. applications capable of running on distributed memory architectures. MPI spawns a desired number of processes (ranks) that all execute the same code on a distributed memory platform. Each MPI process is bound to a CPU on one of the computing nodes. This is in contrast to libraries such as OpenMP, which only support multithreading on a single processor and memory system (shared memory). MPI hides plenty of the communication code required for inter processor communication and offers a set of functions for various types of data exchange between processes. It is also possible to combine MPI and OpenMP to take advantage of the shared memory between several CPUs on one processor as well as to allow communication between different processors. However, this option was decided against in this work for simplicity reasons.

The following chapter provides an insight into the code global structure. It presents the main functionalities of the code, shows interactions between key modules and briefly describes IO. Several parallel implementation features are highlighted, especially related to the FETI algorithm.

5.1 Structure

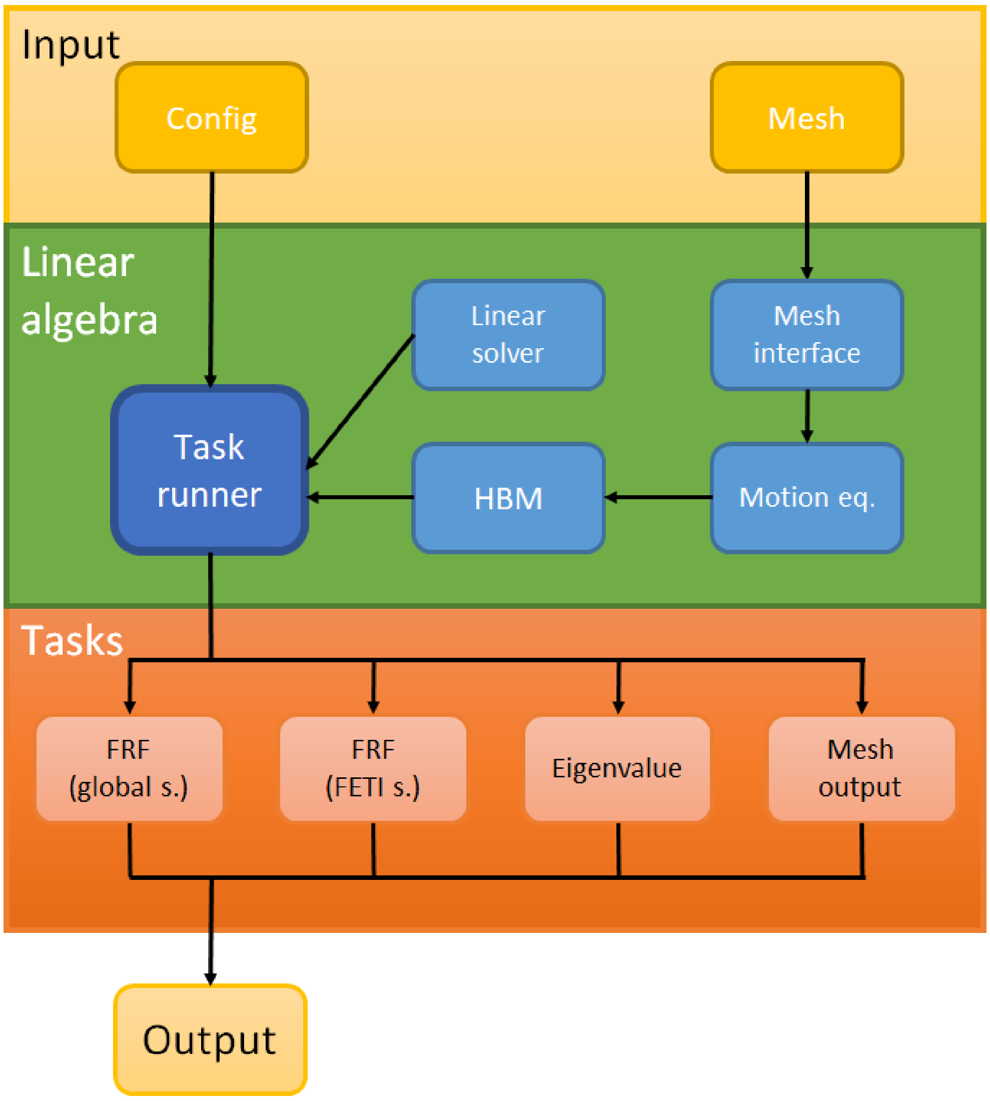


Figure 5.1: Diagram of the general flow of the code run. The code takes a configuration file and a mesh as an input. Task runner acts as a main class that executes selected tasks while employing the key modules such as HBM class, FE class, mesh interface and linear solvers. The linear algebra framework is employed throughout the most of the code.

The structure of the code is designed around the object oriented principle. Several classes represent key elements of the code. Their overview and general interaction can be seen in Figure 5.1. The HBM class handles assembling of the harmonic problem matrices and vectors in accordance to (3.15), as well as their FETI domain versions as in (4.7). The abstract linear solver class represents a common interface to various linear solvers. The

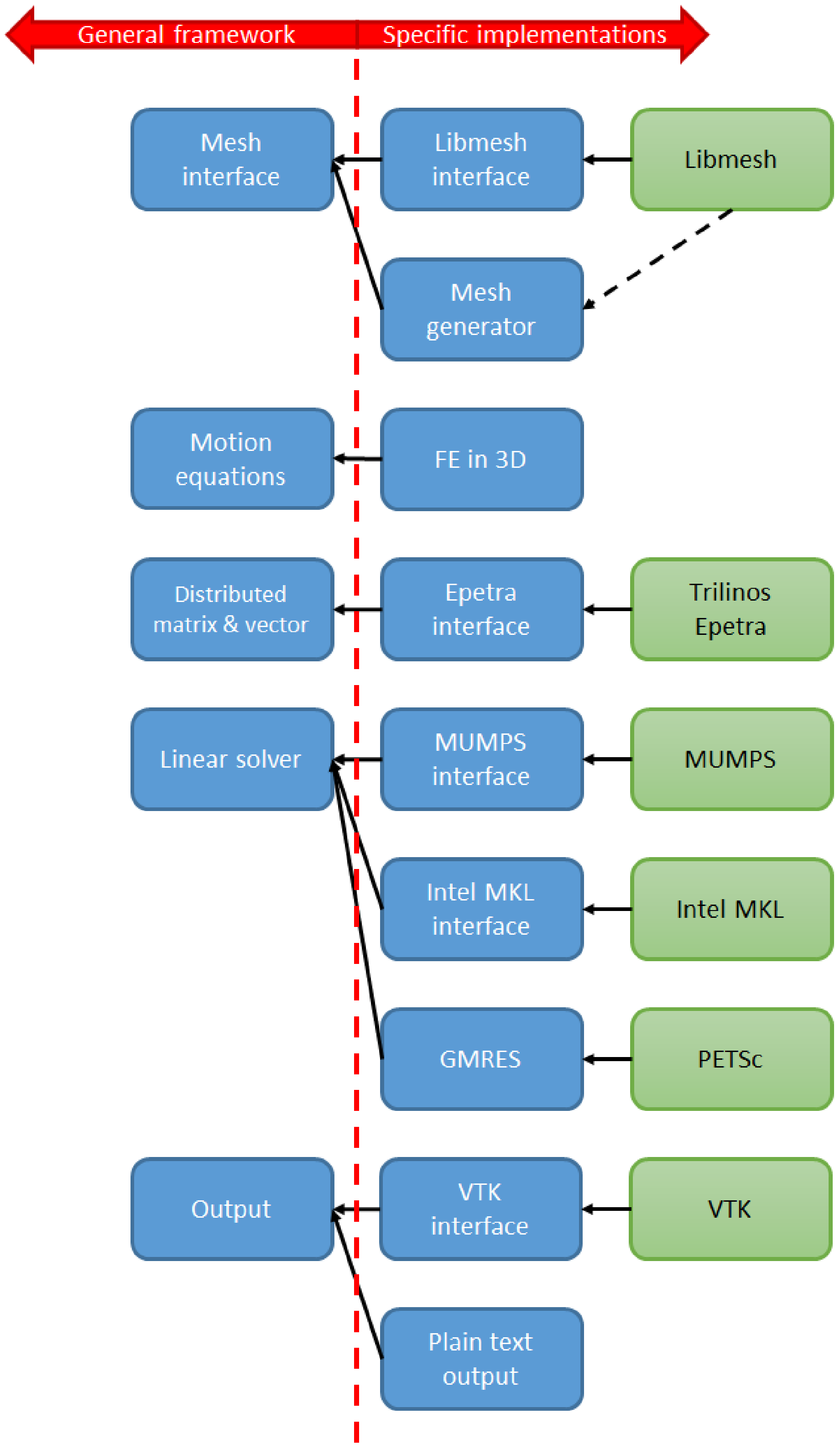


Figure 5.2: Diagram of some of the key abstract elements of the code and their implementations and 3rd party libraries used.

motion equation class provides an interface for equations of motion in time domain as they are written in (2.40). Various physical systems can yield such discrete motion equations. However, the code currently only provides an implementation for 3D finite elements dis-

List of used 3rd party libraries	
MUMPS [5] [4]	Parallel direct sparse solver
Intel MKLPDSS [1]	Parallel direct sparse solver
Eigen [79]	Linear algebra package
libMesh [104]	Parallel mesh processing package
Spectra [148]	Large scale eigenvalue library
Armadillo [159] [160]	Linear algebra package
Trilinos (Epetra) [173]	Parallel linear algebra package
PETSc [10]	Portable, Extensible Toolkit for Scientific Computation

Table 5.1: List of used 3rd party libraries

cretisation as described in 2.3. A general mesh interface is also present to provide a unified access to mesh functionalities such as obtaining nodal and element data. Lastly, a linear algebra interface is used throughout the most of the code. This interface provides access to the basic linear algebra classes such as matrices and vectors and functionalities around them. The linear algebra interface also handles distributed parallelism, i.e. the matrices and vectors can exist among multiple MPI ranks. This will be discussed in more detail later.

The general interfaces described above have each their implementation, possibly multiple. The overview of the most important parts of the code represented by a general interface and their implementations can be seen in Figure 5.2. This design is a typical feature of an object oriented code and it allows for simple additions or replacement of various implementations of the same interface as well as for unified handling of different 3rd party libraries. The list of all used 3rd party libraries is in Table 5.1.

The linear solver interface currently supports 3 different solvers that are sourced by 3 different 3rd party packages (Intel MKL, MUMPS and GMRES from PETSc). Switching between these solvers is then a matter of changing one configuration option value. However, due to technical issues with IntelMKL regarding compilation only the GMRES and MUMPS solvers were used for this work.

The mesh interface provides an access to the problem mesh. The mesh is spatially distributed into domains among MPI ranks. The number of domains is equal to number of MPI ranks, with each domain assigned to one MPI rank. The interface has 2 different implementations. First is a wrapper around the libMesh library. This library is capable of loading and parsing common mesh file formats (such as gmsh, unv, exd and others) and partitioning (using Metis or Parmetis) the mesh among MPI ranks. It further provides information about the mesh distribution, nodal and element mapping to the mesh domains. For each element of the mesh, it provides the finite element values such as coordinates of the Gauss integration points, values of the Jacobian determinants, the shape functions as well as their derivatives. It supports linear as well as quadratic finite elements and various quadrature rules for numerical integration. A drawback of this library is that its loading and distribution part scales badly with number of MPI ranks and mesh size. Therefore, an additional mesh interface implementation was created that generates a regular hexahedral mesh directly in the code. While this only limits the available meshes to shapes such as beams, cubes and plates, the advantage is good scalability as no file loading has to be performed, and most of the mesh generation can be performed in parallel. This mesh generator still uses some of the libMesh library features such as obtaining values required for numerical integration over elements (this is expressed by the dashed line in 5.2). The VTK library is used for visualisation and postprocessing of computed results.

The central class in the code is the task runner. It interacts with all the other core classes in the code and uses them to execute tasks demanded by the configuration. Three tasks are available: FRC computation (using either the global (3.15) or FETI (4.17) form of the HBM equations), computation of eigenfrequencies of a linear system (3.45) using the eigenvalue library Spectra and output of the distributed mesh into VTK files. Pseudocode of the continuation loop can be seen in Algorithm 2. Among other things, the algorithm also shows how the continuation step size was adapted. In case of FETI, the continuation is additionally performed in the λ variables as well. In case of the tangent predictor variant, the tangent vector computed by equation (3.31) or (4.76) can point in either direction. Therefore, the tangent from the previous continuation step is stored and cosine between those two tangents is computed. If this cosine is negative, meaning the angle between the vectors is larger than 90°, the current tangent is reversed. This step is based on the assumption that the tangent vector of the curve would not change direction so dramatically between 2 continuation steps.

5.1.1 Input and output

Input of the code is all done through a single configuration file. This file uses a flat structure consisting of a set of keys and corresponding values, which can range from single numerical values to lists of strings. On code start, each MPI rank loads the configuration file, parses it as stores all the provided key-value pairs. Each of the existing keywords also have its predefined default value that is used if no value is provided in the configuration file. Additionally, any key-value pair can be provided directly in the terminal when executing the code as a command line argument, overriding the configuration file.

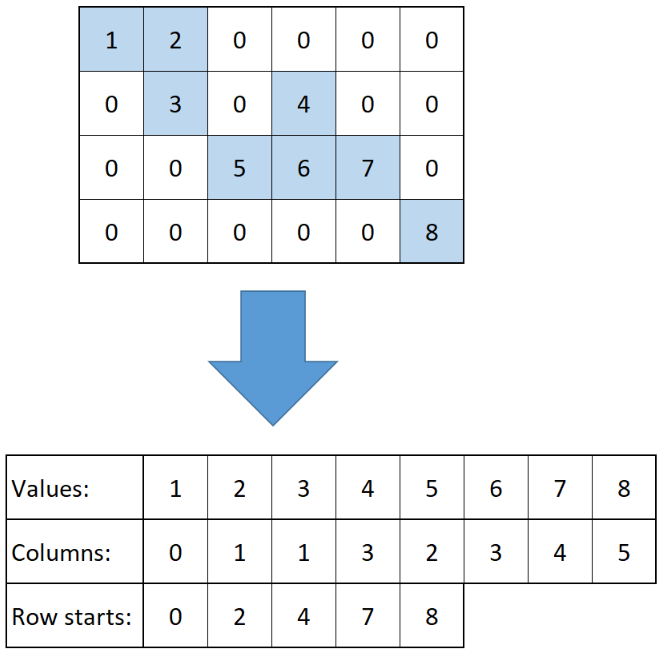
Output of the code can be created in two formats as illustrated in Figure 5.1: plain text and VTK files. The text output simply lists values for all degrees of freedom for all frequencies of an FRC. One file is created per MPI rank, with the local dofs values. Additionally, only output for selected mesh nodes can be created. The values are written in the csv format. The VTK output creates the set of vtu files, one per domain. This file stores the data in a VTK unstructured grid format. It provides the information of the mesh partition for the given MPI rank, together with the values of solutions for all nodes of that partition. Additionally, one pvtu file is created that holds information about all the vtu partitions, so the entire solution can be loaded in a visualisation tool through this one file.

Algorithm 2 Continuation loop

```
1: Input: Mesh, external forces, physical parameters
2: Input: Start and end frequencies $\omega_s$ and $\omega_e$
3: $\omega_{min} = min(\omega_s, \omega_e)$
4: $\omega_{max} = max(\omega_s, \omega_e)$
5: $s = s_s$                                                    ▷ Starting continuation step
6: $\omega = \omega_s$                                          ▷ Starting continuation frequency
7: $\tilde{u} = NewtonSolve(\omega)$                            ▷ Solve for starting frequency, see (3.17)
8: $(\tilde{u}_p, \omega_p) = (O, 0)$                           ▷ Storage for previous solution and frequency
9: while $\omega \in \langle \omega_{min}, \omega_{max} \rangle$ do
10:     $(\tau_{\tilde{u}}, \tau_\omega) = PredDir(\tilde{u}, \omega, \tilde{u}_p, \omega_p)$   ▷ Prediction direction, see (3.31), (3.40), (3.41)
11:     $(\tilde{u}^p, \omega^p) = \frac{s}{\|\tau\|}(\tau_{\tilde{u}}, \tau_\omega)$   ▷ Prediction step, see (3.32)
12:     if $\omega^p \notin \langle \omega_{min}, \omega_{max} \rangle$ then
13:         break                                               ▷ End loop if out of frequency range
14:     end if
15:     $(\tilde{u}_{new}, \omega_{new}) = NewtonSolve(\tilde{u}^p, \omega^p, \tau_{\tilde{u}}, \tau_\omega)$   ▷ Correction step, see (3.37)
16:     if NewtonConverged then
17:         $(\tilde{u}_p, \omega_p) = (\tilde{u}, \omega)$     ▷ Update previous solution
18:         $(\tilde{u}, \omega) = (\tilde{u}_{new}, \omega_{new})$   ▷ Update current solution
19:         if $newtonIt < maxNewtonIt$ then                    ▷ Did Newton converge in less than maximum number of iterations?
20:             $s = s * sUpCoeff$                              ▷ Increase $s$ by a coefficient $> 1$
21:         else                                                ▷ Keep the step size the same
22:         end if
23:     else
24:         $s = s * sDownCoeff$                                ▷ Decrease $s$ by a coefficient $< 1$
25:         if $s < s_{min}$ then
26:             break                                           ▷ End loop if even minimum step size failed
27:         end if
28:     end if
29: end while
```

5.2 Linear algebra

A fundamental building block of the code is an efficient linear algebra implementation. As mentioned before, this is provided by the Epetra package, which is a part of the Trilinos library. Epetra provides efficient large scale distributed implementations of sparse matrices and vectors, and operations on them. It also employs the object oriented principle, making the library code easily understandable and well structured.



1	2	0	0	0	0
0	3	0	4	0	0
0	0	5	6	7	0
0	0	0	0	0	8

Values:	1	2	3	4	5	6	7	8
Columns:	0	1	1	3	2	3	4	5
Row starts:	0	2	4	7	8			

Figure 5.3: Example of a matrix stored in the CSR format.

Epetra uses a compressed sparse row (CSR) matrix format. Rather than storing the entirety of the matrix in a larger array, only the nonzero values are stored in a list, along with their column indices. An additional list of indices is stored to mark starts of individual rows in the value and column arrays. This storage system is efficient for matrices who don't have many nonzero values, compared to their total number of elements, which is commonly the case with matrices coming from the finite element discretisation. Furthermore, the row start indices array reduces the total memory consumption even more, as it is not necessary to store row indices for all nonzero values. An example of such storage is in Figure 5.3.

To achieve parallelism, a matrix or a vector is distributed among the MPI ranks. It essentially exists on all ranks simultaneously. Each rank then contains part of the data. A particular distribution is defined by distribution maps. These are maps defining a distribution of indices among MPI ranks, within a given MPI communicator, meaning among given set of MPI ranks. A set of indices is split among MPI ranks, so that each rank only contains a subset of those indices. This distribution can be nonoverlapping, meaning the subsets are

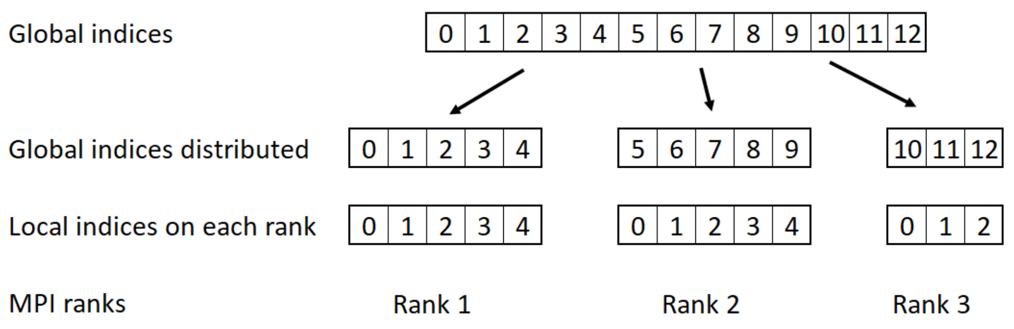


Figure 5.4: Example of index distribution among MPI ranks. This distribution has no overlaps between ranks.

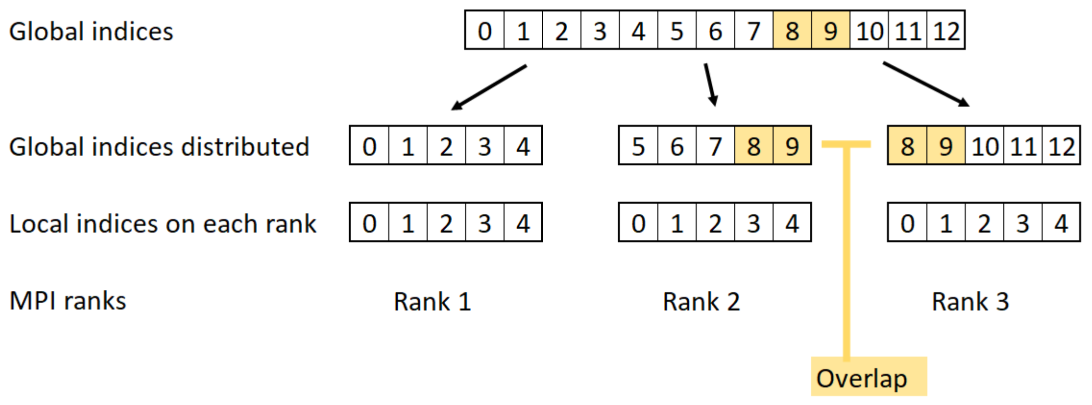


Figure 5.5: Example of index distribution among MPI ranks. This distribution has an overlap between ranks 2 and 3 - indices 8 and 9 are present on both ranks.

mutually exclusive, or overlapping, meaning some of the subsets contain common indices. An example of a nonoverlapping distribution map is in Figure 5.4 and of an overlapping distribution map in Figure 5.5. The distribution maps contain the original global indices (a portion of them), but they also establish new local indexing on each MPI rank. It should be noted that the distribution maps don't necessarily have to be contiguous, i.e. for example rank 1 can contain indices 0 and 2 and rank 2 can contain indices 1 and 3.

A distributed matrix is built using two distributed maps - one for row distribution and one for column distribution. The row distribution map determines the actual distribution of the matrix data among the MPI ranks and it has to be nonoverlapping. A matrix is distributed uniquely along its rows. The column map determines the range of column indices that contain any nonzero values on a given rank. It doesn't determine an actual distribution of data among MPI ranks, but it is essential for operations such as matrix-vector multiplication. This map can be (and often is) overlapping. An example of a matrix distributed among MPI ranks, with its row and column maps, is in Figure 5.6. A distributed vector is built using only one distribution map - the row distribution, as it only contains one

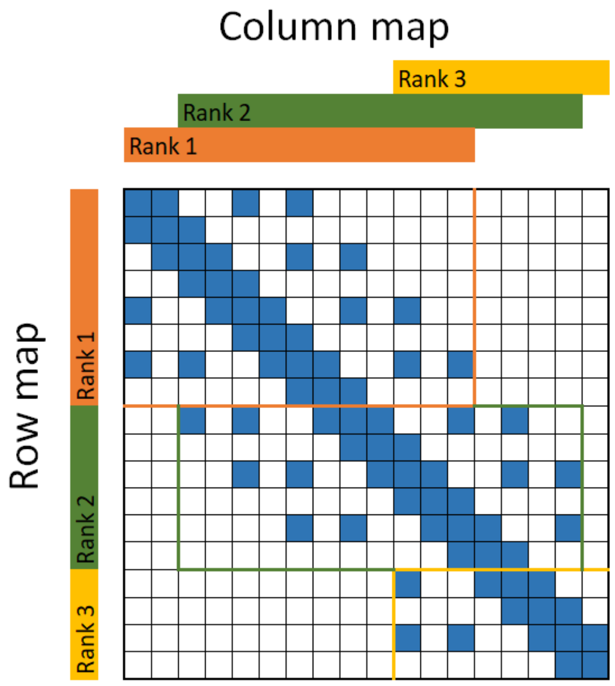


Figure 5.6: Example of a distribution of a matrix assembled globally over all MPI ranks. Blue fields represent nonzero values. The matrix is uniquely distributed by rows, defined by row maps. Column maps can overlap in order to cover all columns related to the given set of rows.

column. However, in the case of vectors, this row distribution map can also be overlapping.

Another important element of the distribution framework are imports. Import is a transfer of values from a matrix/vector of one distribution to the same matrix/vector (in terms of dimensions) on the same MPI communicator but with different distribution. In this code, only imports of vectors are performed. This is useful in various situations when vector values need to be redistributed among MPI ranks in different way, so each MPI rank has access to values that it requires for its further computation tasks. For example, a matrix-vector multiplication has to import the multiplied vector from its map into the column map of the multiplying matrix, so that each rank has all the vector values necessary to compute the local part of the result. An example of possible imports is in Figure 5.7. Naturally, imports involve communication between MPI ranks, so they should be used carefully. The Epetra matrix and vector classes also allow for adding values to nonlocal positions, meaning adding values from a rank to row indices that are not present in the distribution map for that rank. This of course also implies communication, but it is a necessary thing when assembling the finite element matrices.

The functionalities above - the distributed matrix and vector class, the distribution maps and imports - are all provided by the Epetra package. They are, however, hidden behind an interface that allows for potential replacement by another linear algebra library, without having to modify the rest of the code.

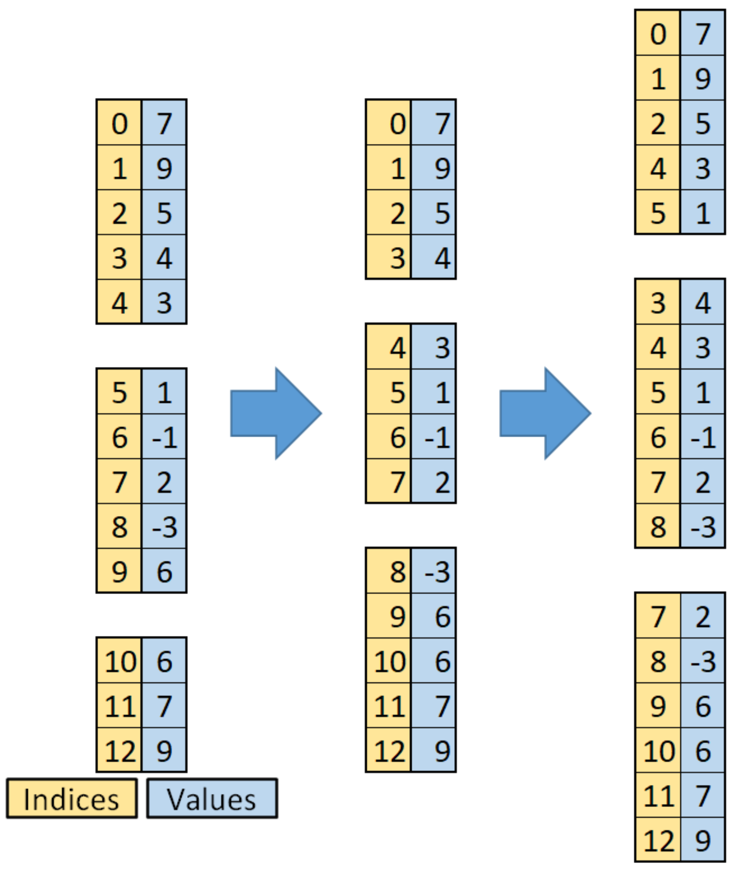


Figure 5.7: Example of two different imports between vectors of same size but with different distributions. A vector distributed uniquely (no overlaps) has its values imported into another vector with a different unique distribution. Then the values are imported again, this time into a vector with overlapping and noncontiguous distribution.

5.2.1 DOF ordering

Throughout Chapters 3 and 4, ordering degrees of freedom in vectors and matrices is done by harmonics. First come all dofs related to harmonic 0, then all dofs related to harmonic 1 cosine part, then sine part, etc. This is done intentionally in order to simplify the structure of the vectors and matrices, since individual harmonics are of primary interest when describing HBM. Using this ordering, the dynamic stiffness matrix $Z(\omega)$ can be conveniently expressed using the harmonic block structure as seen in (3.14). For implementation of spatially parallelised HBM the preferred ordering is different. Ordering used in this work is illustrated by Figure 5.8.

The idea of this ordering is that dofs are primarily sorted by nodes. This means that the distribution of dofs follows the spatial distribution of the mesh into domains. Within the scope of one node dofs are sorted first by their dimension and then harmonic coefficient. All dofs related to a particular node are present on that node's MPI rank. This eliminates any need for MPI communication when transforming solution and forces vectors from frequency to time domain and back using the AFT procedure.

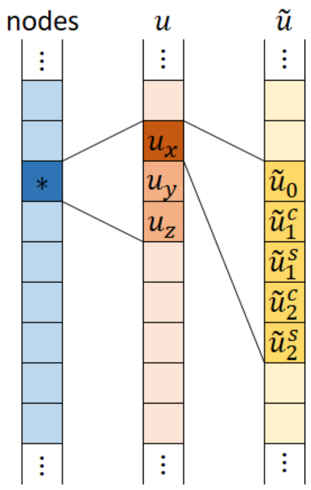


Figure 5.8: Hierarchy of orderings applied to degrees of freedom. First, dofs are sorted by their corresponding nodes. Then, within each group of dofs for one node, dofs are sorted by their physical dimension. Lastly, within each group for one node and dimension, dofs are sorted by harmonics.

This ordering is applied to all maps, non overlapping and overlapping, as well as to both primal and dual variable vectors and matrices.

5.2.2 Global solvers

As stated previously, one way to compute an FRC is by solving the global system of equations. A solver in this case operates on the global set of linear equations and uses matrices and vectors distributed globally among all MPI ranks. All distribution maps used for the global system of equations use the global communicator MPI_COMM_WORLD. The HBM system matrices and vectors, as written in (3.15), are assembled globally for the entire mesh. The mesh is distributed among MPI ranks, with each rank owning an exclusive subset of the mesh elements and nodes. Since the degrees of freedom are connected to mesh nodes, a global unique distribution map is established based on the nodal distribution of the mesh. This map is used as the row map for the Jacobian matrix in the Newton solver and all the vectors as well. Dirichlet boundary condition for this global system is enforced by zeroing out corresponding rows and columns and putting 1 on the diagonal as described in Section 3.5. An example of a mesh distribution and corresponding dof maps is in Figure 5.9.

Additionally to the unique map, another map is required for certain tasks. In order to assemble the nonlinear part of the Jacobian matrix $\nabla_{\tilde{u}}\tilde{F}_{int}^{nl}$ and the nonlinear force vector $\tilde{F}_{int}^{nl}$, one needs displacement values on all nodes of all elements on the given rank, so that the element contributions can be computed. Therefore, values of the solution vector $\tilde{u}$ that is kept in the non overlapping distribution have to be imported into another vector which is distributed in an overlapping distribution. This overlapping distribution includes all nodes in contact with elements that are local to the given rank. This distribution is the second distribution illustrated in Figure 5.9. Importing from the non overlapping map into the

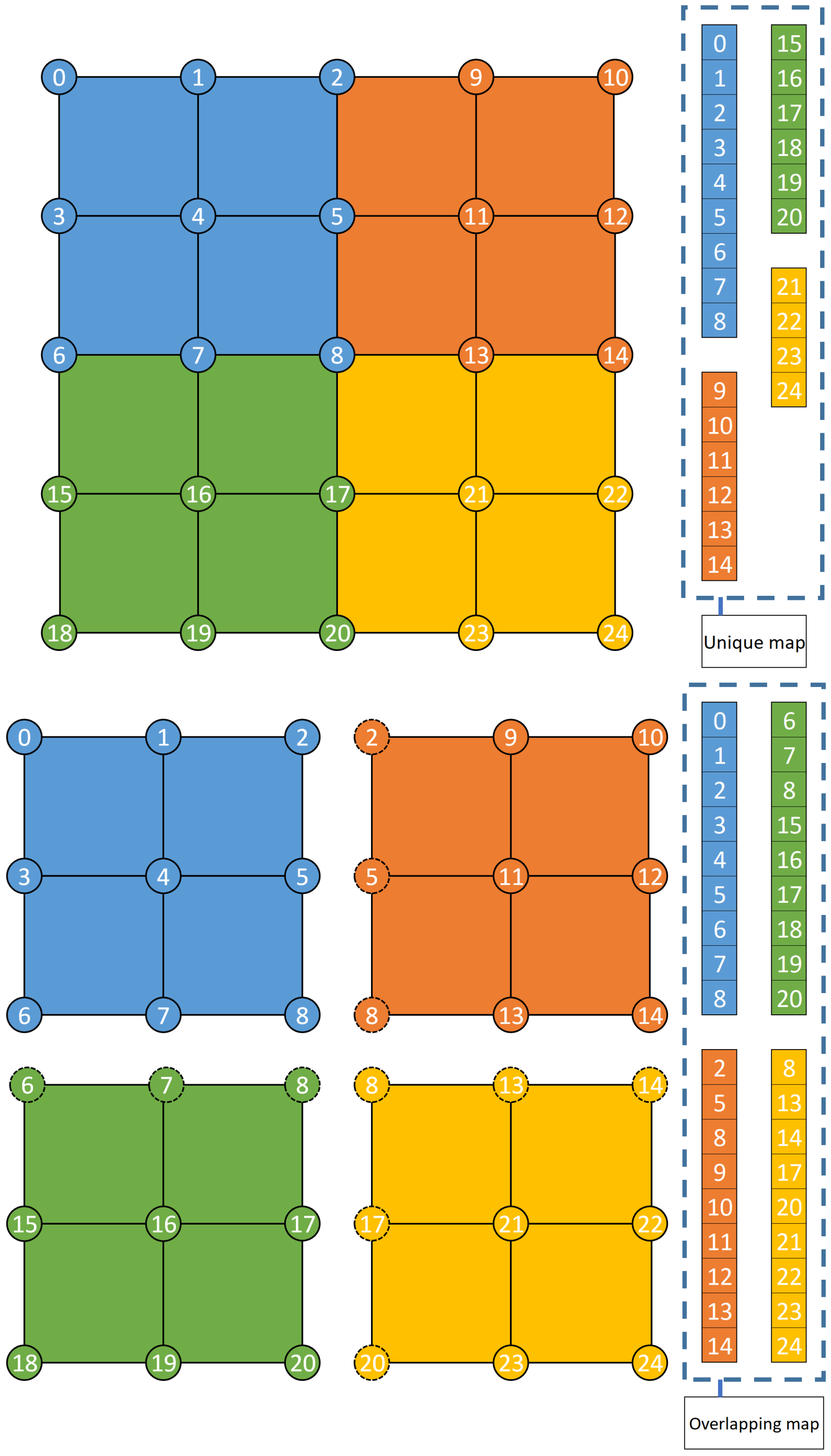


Figure 5.9: Example of a mesh distribution and 2 maps generated from it. Each colour represents one MPI rank. The first map is a nonoverlapping defining unique distribution of nodes among MPI ranks. The second is an overlapping map which includes domain boundary nodes in all those ranks whose domains are in contact with that node.

overlapping map involves communication. However, this communication only happens among neighbouring domains of the mesh, so its performance impact is limited.

The code currently includes 3 different global linear system solvers, as illustrated in Figure 5.2 - MUMPS, Intel MKLPDSS and PETSc GMRES. All these solvers are capable of solving a system of linear equations in parallel, and they all use a distribution approach similar to the one used by the Epetra package. It is therefore straightforward to feed the distributed matrix into the solvers. The MKL solver uses exactly the same CSR format that's described in Figure 5.3. MUMPS uses the same format for the values and row indices arrays, but it also requires full array of column indices (same as row indices array), so this array has to be specially created for this solver. The PETSc GMRES solver doesn't require the system matrix assembled globally (except when certain preconditioners are used) but in the current code implementation the matrix is still assembled.

Convergence

Convergence of the newton loop solving a global HBM system as described in (3.15) is determined based on the relative residual norm:

$$\frac{\|Z(\omega)\tilde{u} + \tilde{F}^{nl}_{int}(\tilde{u},\omega) - \tilde{F}\|}{\|\tilde{F}\|} \leq tol \tag{5.1}$$

This criterion is used also in pseudo arc-length continuation loop when ω is also a variable in the system. Satisfaction of the added constraint equation g in (3.34) is not considered for convergence. In case of using MUMPS as the linear solver for individual Newton iterations, the relative residuals after solving the linear systems (3.38) are not controlled. In case of the GMRES solver, the tolerance for this residual is set in configuration and observed. Additionally, the maximum number of GMRES iterations is also restricted by a configuration value. In neither case is the linear solver relative residual used as a convergence criterion. The computed Newton step is accepted regardless of the final relative residual of the linear system.

5.2.3 Notes about MUMPS

The MUMPS solver is used as the main direct solver of linear systems in the code. It is used as a global solver when solving the global HBM system, as well as a solver for domain matrices $\mathcal{A}$ and matrices inside projectors $\mathcal{P}$ and $\mathcal{P}_c$ in the FETI solver. Two versions of the library were used in the code for the time span of this work. First, the version 5.1.2 was linked and used for for the earlier computations. Later, an update to version 5.4.1 was made.

MUMPS operates in 3 stages. The first stage is analysis. In this step, the solver analyses the matrix structure in terms of its nonzero pattern. It only requires to be provided with indices of nonzero values. In ParHBM, this step needs to be invoked in many cases only once throughout the entire continuation run. This is because nonzero patterns of certain matrices (such as the global system Jacobian matrix) don't change. Even when the Jacobian matrix happens to be linear for certain iterations of Newton solve, the solver is still provided with all potentially nonzero indices. Even values that happen to be zero for the linear case are provided as explicit zeros into the solver.

The second stage is factorisation. This stage computes the actual LU factorisation of the matrix. The matrix values have to be provided for this step. Factorisation needs to be performed every time when the matrix values change, i.e. at the beginning of each Newton iteration. However, it doesn't need to be executed every time a solve is required with a different right hand side. This is beneficial for the corrector step of the continuation algorithm which uses the bordered matrix solve approach and therefore 2 solves with 2 different right hand sides are required as seen in (3.38). The factorisation stage is the most computationally expensive part.

The last stage of the MUMPS solver is the solve stage. This step computes the solution of the system, provided a right hand side vector, performing the forward and backward elimination using the factors L and U computed during the factorisation phase. As mentioned before, it can be executed many times for various rhs vectors as long as the matrix is unchanged. It also allows for multi right hand side (rhs) solves, which is more efficient than solving for each rhs vector separately in a loop. This is beneficial in case of solving (3.38). The solve phase has significantly lower computational requirements that the factorisation phase.

MUMPS offers several other features, such as computing Schur complement, which was used for the Dirichlet preconditioner in FETI, or solving a rank deficient system by detecting null pivots of the matrix. This means it offers multiplication by the matrix pseudoinverse, which is used for domain matrices in FETI when applying $\mathcal{A}_i^+$. MUMPS can also solve a system with a transposed matrix, using the initialised and factorised original matrix, allowing for application of $\mathcal{P}_c^T$ required in (4.66) without needing for separate assembly of $(\mathcal{G}_c^T \mathcal{F} \mathcal{G}_c)^T$ and $\mathcal{A}_i^T$.

The older version (5.1.2) used in this work comes with a limitation of only allowing centralised right hand side vectors, which means that the right hand side vector of a linear system needs to be collected from all ranks to rank 0 before solving the system. This obviously creates a communication bottleneck. This has been changed in version 5.2.1 when support for distributed right hand side vectors was added. However, the solver interface in the code was not updated to support this new feature, so even with the newer MUMPS version the rhs was still gathered on rank 0. This bottleneck impacts the MUMPS solver whenever a matrix distributed over multiple MPI ranks is factorised. The domain matrix solves in FETI won't be affected by this since those MUMPS instances operate each on a single rank.

The solution vector is distributed inside MUMPS. The distribution map is decided by the package during the factorisation phase and provided to the user. Since the MUMPS interface in ParHBM returns solution in the same distribution as the right hand side vector and the matrix rows, an import of the solution needs to be made after every solve from the MUMPS distribution to the unique row distribution. The amount of communication required for this depends on how different those two distributions are.

An important part of the FETI solver is null space detection on domain matrices $\mathcal{A}_i$. MUMPS is capable of detecting null space of matrices and solving a linear system with rank deficient matrix and returning one possible solution, essentially providing a pseudoinverse functionality. The null space detection in MUMPS is controlled by a threshold parameter for null pivot detection. A pivot encountered in factorisation is considered to be null if the infinite norm of its row/column is smaller than $tol \times \|A_{pre}\|$ where tol is a configurable parameter and A_{pre} is the system matrix, preprocessed before factorisation:

$$A_{pre} = PD_rAQ_cD_cP^T \tag{5.2}$$

where P is a permutation matrix, Q_c is a column permutation and D_r and D_c are diagonal matrices for row and column scaling. This null pivot detection has to be enabled on the domain matrix solvers to get the pseudoinverse functionality. However, the null space found by MUMPS is not used in the code. As it was discussed in Section 4.4, the assumption is made that null space is present only on the 0th harmonic block, and has a specific size. In linear case it contains 6 vectors (3 translations and 3 rotations) and in nonlinear case only the 3 translations. Therefore, these vectors are assembled by the code directly, in a way that makes them mutually orthogonal, and normalised. It is then checked whether they are truly null space vectors by multiplying them by the domain matrix and checking the error from zero vector. The code also checks that the number of null space vectors is equal to number of detected null pivots by MUMPS on each domain.

5.3 FETI implementation

The FETI solver is an in house implementation, following the theory laid out in Chapter 4. The projected dual problem (4.59) or (4.66) is solved using the GMRES PETSc solver, the same that is used for the global system. The FETI domain matrices $\mathcal{A}_i$ and matrices inside the projectors $\mathcal{P}$ and $\mathcal{P}_c$ are factorised by MUMPS. This approach allowed for experiments with various parts of the algorithm and adaptation of the solver for specifically solving nonlinear HBM problems.

5.3.1 Primal and dual maps

The FETI solver uses the same domain distribution as the global solvers. It's only the treatment of this distribution that is different. While in case of the global system dofs

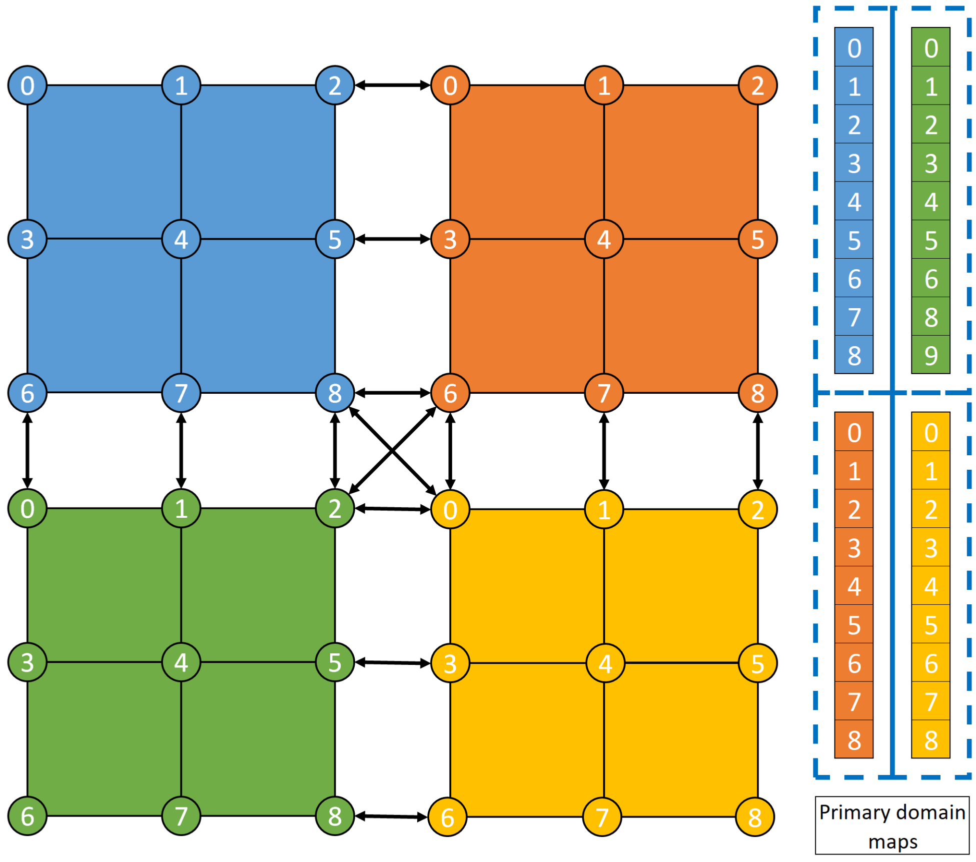


Figure 5.10: Example of a mesh distribution and its interpretation by the FETI algorithm. The distribution itself is the same, but the domain boundary nodes and dofs are assumed to exist separately on all domains that they are in contact with. The primal variable domain maps are shown on the right. Compare that to the overlapping map in Figure 5.9.

are indexed globally among all MPI ranks, the FETI distribution of the primal variables is localised to each MPI rank and therefore domain. Each domain owns all nodes that are connected with its elements, including the boundary nodes. This means that the nodes and dofs are redundant on domain interfaces. Each domain then indexes these dofs locally by its own numbering starting from 0. Each domain therefore creates its own local domain map for primal dofs, which exists on a local communicator MPI_COMM_SELF. An example of such distribution is in Figure 5.10. The domain maps, while having different indexing, can be bijectively mapped onto the overlapping global map which is shown in Figure 5.9. This relationship can be used to transfer FETI primal solution on domains into a global solution vector so it can be compared with solutions from global solvers. First, domain solutions are transferred from domain maps into the overlapping global map. Then, the overlapping map is restricted to the unique global map, meaning only one value is selected for each interface degree of freedom. This choice doesn't have any impact on the solution as long as the domain connectivity equation 4.3 is satisfied within a good enough tolerance. However, in terms of the FETI algorithm, the primal variables only exist in their local domain maps on each MPI rank. No concept of a global solution vector is required.

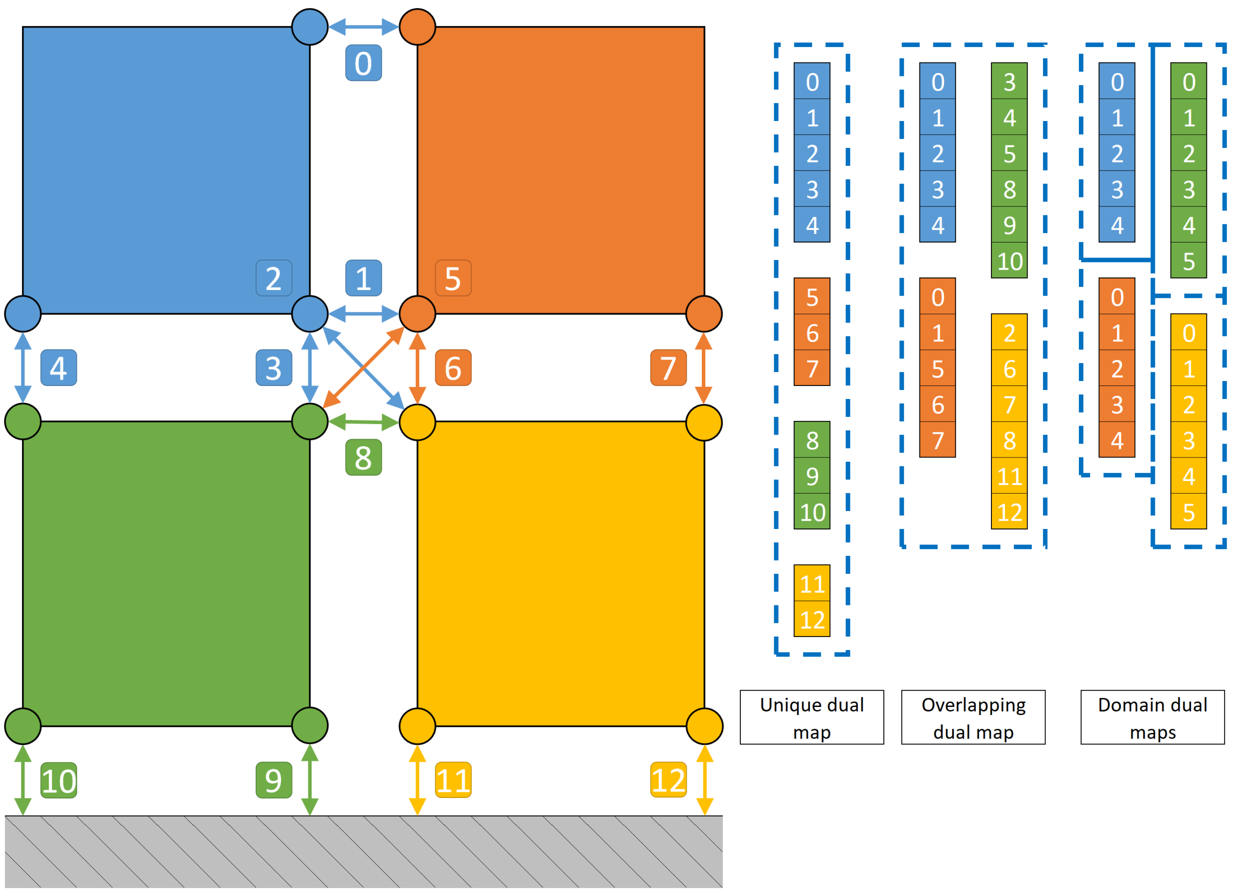


Figure 5.11: Example of a mesh distribution and its interpretation by the FETI algorithm. Highlighting the Lagrange multipliers λ between domains and their distribution among MPI ranks. A possible unique distribution is on the left. An overlapping distribution where each rank owns all lambdas connected with its domain is in the middle. The right map represents the same distribution as the middle map, but localised for each domain, analogously to the localised primal variables in Figure 5.10.

The situation becomes different with the dual variables λ. These variables exist on domain interfaces only. Since the GMRES solver that solves the projected dual problem requires a unique distribution of unknowns among MPI ranks, such distribution needs to be established for λ. This global unique distribution should be well balanced in terms of sizes on individual ranks, which is achieved by a custom implemented balancing algorithm. This algorithm exchanges ownership of dual variables between neighbouring domains, moving it from domains with more variables to domains with less. This exchange is performed a fixed number of times. A requirement is placed on the distribution that each rank only contains those dual variables that belong to one of its interfaces. Besides the unique map, an overlapping map needs to be created that includes all lambdas on each rank that are connected to the corresponding domain. This map is required for importing values of λ from a unique dual vector when domain primal variables need to be computed, using the equation (4.23):

$$\Delta\tilde{u}_i = \mathcal{A}_i^+ \left(-H_i + B_i^T \Delta\lambda\right) + \mathcal{R}_i \alpha_i \tag{5.3}$$

As it was illustrated in Figure 4.4, to perform multiplication of $B_i^T \lambda$ for domain i, only values of λ shared between processor i and its neighbours are required. This is the purpose of the overlapping dual map. Additionally, a domain dual map on a local communicator is created on each rank that mirrors the overlapping dual map, but establishes local indexing on each domain, in same manner as the primal domain maps mirror the global overlapping primal map. All these 3 different dual maps are illustrated in Figure 5.11.

5.3.2 Application of the dual problem operator

One of the most common operations when solving a linear system using the FETI method is multiplication by the $\mathcal{F}$ matrix. It is present multiple times in the projected dual problem equations (4.59) and (4.66), once directly and then in the $\mathcal{P}_c$ projector. This means the multiplication by $\mathcal{F}$ needs to be performed twice or three times in each iteration of GMRES. The mathematical formula for it is simple. However, its explicit assembly is unnecessary and would be inefficient. Instead, it is treated as an operator, and only its multiplication effect on a vector is implemented.

The GMRES solver operates on vectors in unique dual variable map. Therefore, the implementation of the $\mathcal{F}$ operator needs to accept and also return vectors in this distribution. However, internally, all 3 dual maps that were described above are used. A diagram explaining the implementation of the $\mathcal{F}$ operator is in Figure 5.12.

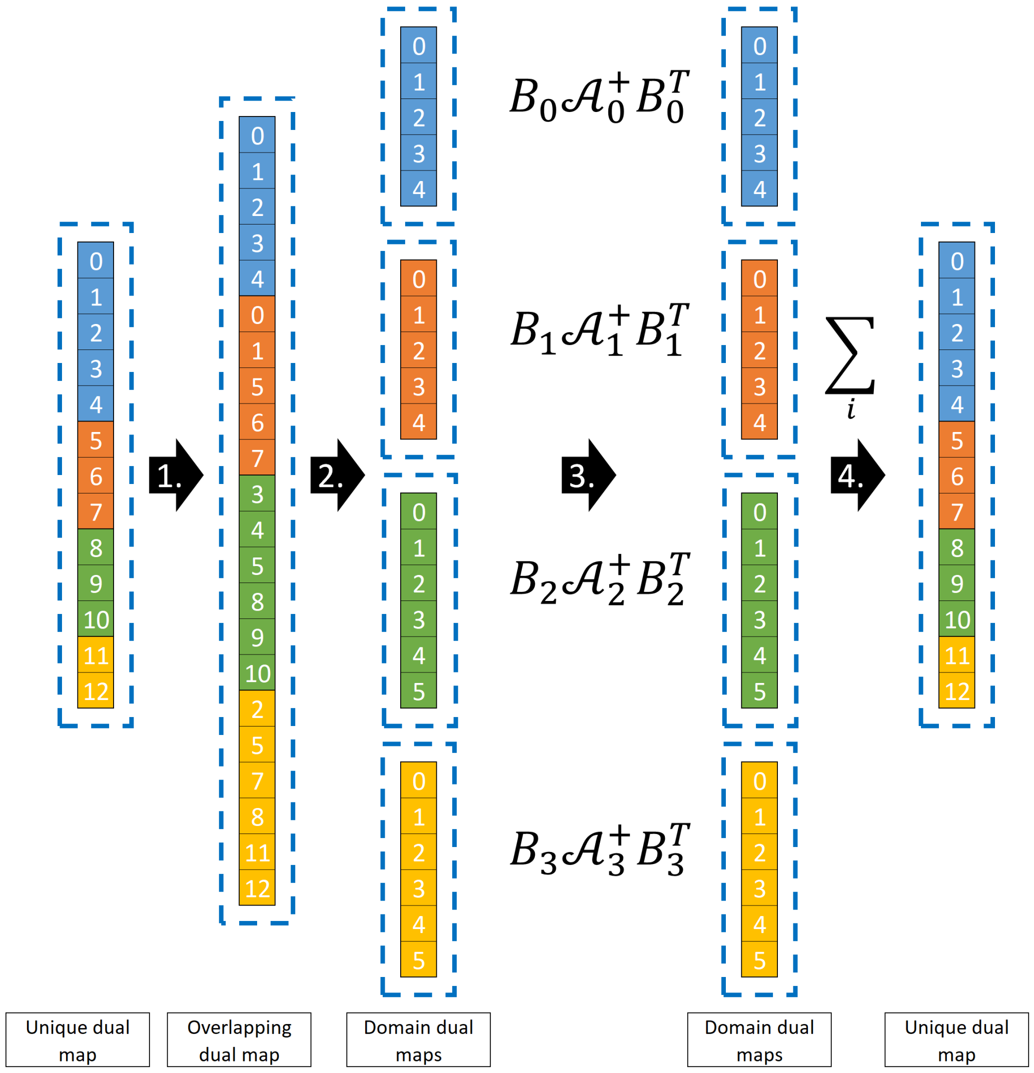


Figure 5.12: The process of $\mathcal{F}$ application. MPI communication between neighbouring domains is performed in steps 1 and 4. The other steps are performed completely independently on each rank.

The input dual vector first needs to be imported from the unique map into the overlapping map, in order to have all necessary dual variables on each rank. This import involves MPI communication but only among neighbouring domains. Next, the values are transferred from overlapping map into local domain dual maps. This is because the B_i matrices are built using the primal and dual domain maps, meaning each B_i matrix exists completely locally on its MPI rank. The row map of the B_i matrix is the dual domain map and the column map is the primal domain map. The transfer of values from the overlapping to domain maps involves no MPI communication. The vectors in domain dual variables are then multiplied by B_i^T on each domain in parallel.

The result of multiplication by B_i^T is a vector in primal domain map. This vector is then multiplied by $\mathcal{A}_i^+$. This multiplication is done by solving a linear system with $\mathcal{A}_i$ as the system matrix. This operation is performed by MUMPS. The domain matrix has the domain primal map for its both row and column map, so it exists entirely locally on each rank. A separate instance of MUMPS solver is created on each rank for this matrix, using the MPI_COMM_SELF communicator. It is factorised once at the beginning of a Newton iteration, and then the solve phase is called with null pivot detection enabled. The result of $\mathcal{A}_i^+$ multiplication is then multiplied by B_i^T, providing a vector in dual domain map.

The final part is to sum contributions of dual vectors from all domains into one vector in the unique dual map. The linear algebra framework offers insertions of values into positions that are on another rank. An option exists to sum multiple values that are inserted into the same position from multiple different ranks. Since the global unique distribution of the domain dual variables is known on each rank, the values are simply inserted into their appropriate positions and the unique distribution vector is assembled, summing all contribution in the process. This step again involves MPI communication between ranks of neighbouring domains.

5.3.3 Projectors

Same as with the $\mathcal{F}$ operator, the input and output of both the $\mathcal{P}$ and $\mathcal{P}_c$ projectors are in unique dual variables. Both of these projectors are also not explicitly assembled, rather their application on a vector is implemented.

The $\mathcal{P}$ projector is defined only by the natural coarse space vectors in $\mathcal{G}$. Provided an input x in the unique dual map, one has to compute $(I - \mathcal{G}(\mathcal{G}^T\mathcal{G})^{-1}\mathcal{G}^T)x$, following the definition in (4.42). Multiplying the outer brackets yields $x - \mathcal{G}(\mathcal{G}^T\mathcal{G})^{-1}\mathcal{G}^T x$. The first step is therefore to compute:

$$x^1 = \mathcal{G}^T x \tag{5.4}$$

This involves computing a dot product of x with each column in $\mathcal{G}$. The vectors of $\mathcal{G}$ exist locally on domains. Following the definition in (4.30), $\mathcal{G}_i$ is assembled in domain dual map of the i-th domain. $\mathcal{G}_i$ only has nonzero values on positions related to interface surrounding its domain, and zeros everywhere else. To evaluate dot product of x with columns of $\mathcal{G}_i$, only values of x on interface of domain i are relevant. This means the x vector needs to be imported from the dual unique map to the overlapping map, which involves only communication between neighbouring domains. The result x^1 is then stored in parallel, each domain containing results of $\mathcal{G}_i^T x$. A unique global map, called the natural coarse space map, is created for this vector, with its size on each rank corresponding to number of columns in $\mathcal{G}_i$ for that rank. Creation of this map involves global communication of column counts of $\mathcal{G}_i$ between all domains to establish the global indexing. However, this communication only exchanges a single integer and is therefore cheap.

The next step is to compute:

$$x^2 = (\mathcal{G}^T\mathcal{G})^{-1}x^1 \tag{5.5}$$

For this, the matrix $\mathcal{G}^T\mathcal{G}$ is explicitly assembled in parallel. Its row map is the natural coarse space map used to store x^1. It's block structure per domain can be expressed as:

$$\mathcal{G}^T\mathcal{G} = \begin{array}{c} \begin{bmatrix} \mathcal{G}_1^T\mathcal{G}_1 & \mathcal{G}_1^T\mathcal{G}_2 & \cdots & \mathcal{G}_1^T\mathcal{G}_\mathcal{N} \\ \mathcal{G}_2^T\mathcal{G}_1 & \mathcal{G}_2^T\mathcal{G}_2 & \cdots & \mathcal{G}_2^T\mathcal{G}_\mathcal{N} \\ \vdots & & & \vdots \\ \mathcal{G}_\mathcal{N}^T\mathcal{G}_1 & \mathcal{G}_\mathcal{N}^T\mathcal{G}_2 & \cdots & \mathcal{G}_\mathcal{N}^T\mathcal{G}_\mathcal{N} \end{bmatrix} \end{array} \begin{array}{l} \text{Domain 1} \\ \text{Domain 2} \\ \\ \text{Domain } \mathcal{N} \end{array} \tag{5.6}$$

However, not all blocks of $\mathcal{G}_i^T\mathcal{G}_j$ will have nonzero values. Naturally, the diagonal blocks $\mathcal{G}_i^T\mathcal{G}_i$ will be nonzero. These products can be evaluated independently on each domain. Besides the diagonal blocks, a block $\mathcal{G}_i^T\mathcal{G}_j$ will have nonzero values only when domains i and j share a common interface. This means that again only communication of values between neighbouring domains is required. The extent of such communication is illustrated in Figure 5.13. Furthermore, only parts of $\mathcal{G}_i$ vectors need to be communicated to each neighbouring domain, as only a part of the entire domain interface is shared with each neighbour. This is shown in Figure 5.14. This fact reduces the amount of data to be communicated between neighbours to the necessary minimum. Using this communication scheme, a rank i receives all necessary values of $\mathcal{G}_j$, where j marks all neighbouring domains, and assembles its part of the $\mathcal{G}^T\mathcal{G}$ matrix, meaning all nonzero $\mathcal{G}_j^T\mathcal{G}_j$ blocks, independently of other ranks. The matrix is then passed into and factorised by MUMPS, which allows for solving $x^2 = (\mathcal{G}^T\mathcal{G})^{-1}x^1$ in parallel. The resulting vector x^2 is distributed in the same way as x^1.

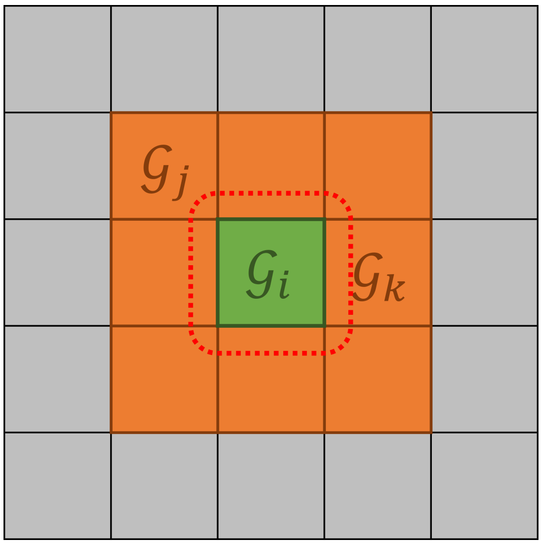


Figure 5.13: Diagram showing domains that need to communicate with the green domain, coloured in orange, when assembling $\mathcal{G}^T\mathcal{G}$. Communication is required between neighbouring domains because they share parts of their interfaces, where the values of $\mathcal{G}$ exist on each domain.

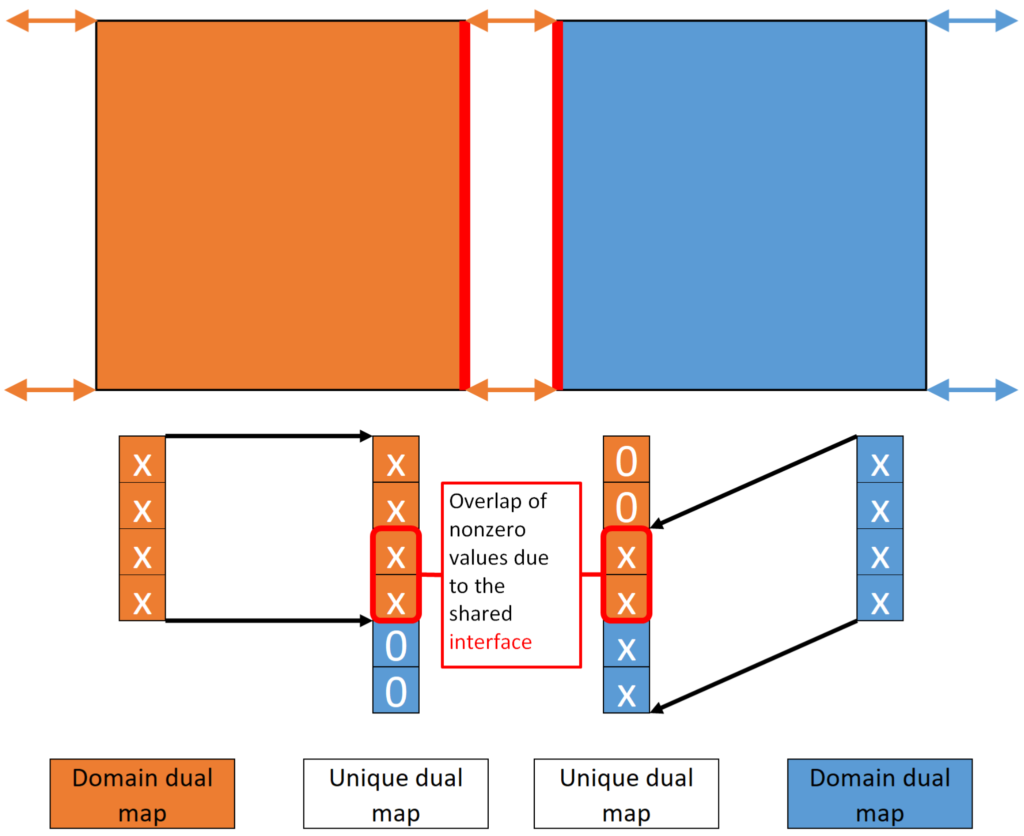


Figure 5.14: Two vectors in dual variables in global unique dual map, each coming from a domain dual vector for different domain. An overlap in their nonzero pattern (positions marked with X) is caused by the shared interface between the two domains.

The final part is to compute:

$$x^3 = \mathcal{G}x^2 \tag{5.7}$$

This implies multiplying columns of $\mathcal{G}$ by values in x^2 and then summing the results into one vector. The multiplication part is entirely parallel as the distribution of values in x^2 is already matching the distribution of columns in $\mathcal{G}$. A single vector x^3 is then created with dual unique distribution, and the result vectors are summed into it. This also involves communication among neighbouring domains.

After performing the above steps, the product $\mathcal{P}x$ is then computed simply as $x - x^3$. The required communication is among domain neighbours plus communication required by MUMPS when factorising and solving with the $\mathcal{G}^T\mathcal{G}$ matrix. It should also be noted that this matrix is very small compared to the size of the entire dual problem, having at most 6 rows and columns per domain, corresponding to the 6 possible null space vectors of $\mathcal{A}$.

Application of $\mathcal{P}_c$ is in many ways similar to the application of $\mathcal{P}$. Computing the dot products $x^1 = \mathcal{G}_c^T x$ is done in the same way. The columns of $\mathcal{G}_c$ are distributed analogously to $\mathcal{G}$ - each domain owns vectors on its interface $\mathcal{G}_{c,i}$. An artificial coarse space unique map is established to store x^1, based on the distribution of columns of $\mathcal{G}_c$.

The second step - assembly of $\mathcal{G}_c^T \mathcal{F} \mathcal{G}_c$ - requires more attention due to the presence of the $\mathcal{F}$ operator. Application of $\mathcal{F}$ to vectors of $\mathcal{G}_{c,i}$ of a particular domain i extends the nonzero pattern of the result vectors to interfaces of neighbouring domains. This is because domain matrices of neighbouring domains $\mathcal{A}_j$ couple dofs between various interfaces. Illustration of this can be seen in Figure 5.15.

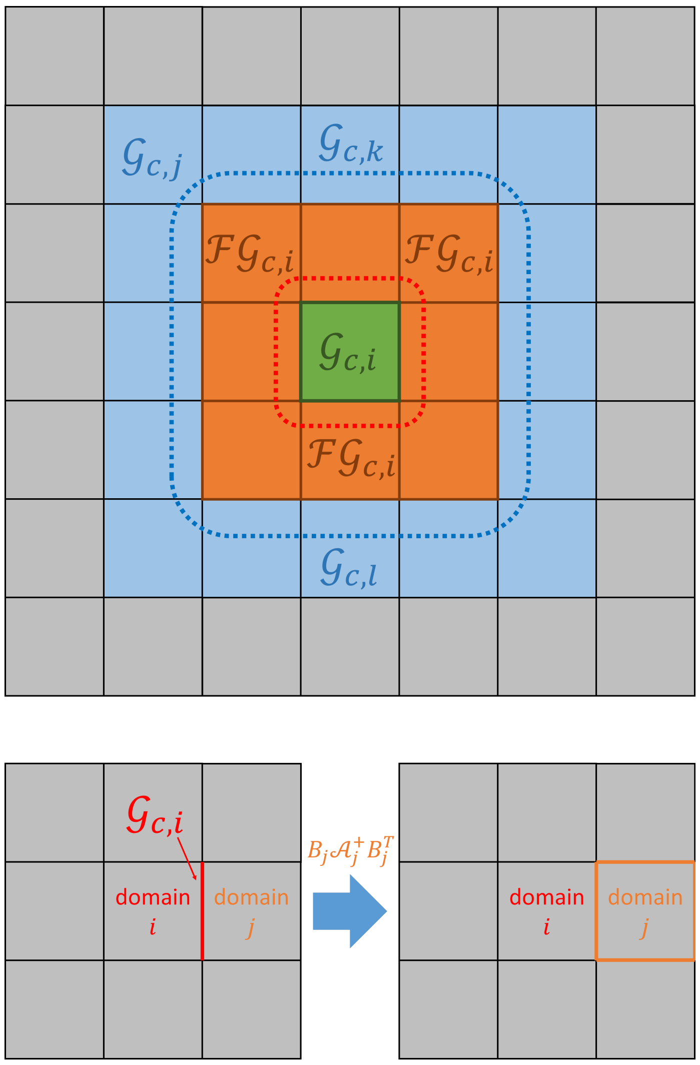


Figure 5.15: Diagram (top) showing domains that need to communicate with the green domain, coloured in orange and blue, when assembling $\mathcal{G}_c^T \mathcal{F} \mathcal{G}$. Each domain needs to communicate with its neighbours and also neighbours of its neighbours. This is because the $\mathcal{F}$ operator extends the nonzero pattern of a $\mathcal{G}_c$ vector of domain i to its neighbours (bottom).

Therefore, more blocks in $\mathcal{G}_c^T \mathcal{F} \mathcal{G}_c$ will be nonzero. Each domain needs to communicate with its neighbours and also neighbours of its neighbours in this step. The matrix is assembled in parallel in the same way as $\mathcal{G}^T \mathcal{G}$, using the unique artificial coarse space map for its row map. The matrix is generally larger than $\mathcal{G}^T \mathcal{G}$, especially when using multiple harmonics, since $\mathcal{G}_{c,i}$ contains 6 vectors for each harmonic coefficient. However, it is still much smaller than the entire dual problem.

The solve $x^2 = (\mathcal{G}_c^T \mathcal{F} \mathcal{G}_c)^{-1} x^1$ is then performed. Next, the multiplication and sum of columns of $\mathcal{G}_c$ is performed in the same way as in case of $\mathcal{P}$. Lastly, the $\mathcal{F}$ operator is applied. The mechanics of that step have already been described previously.

In summary, application of both $\mathcal{P}$ and $\mathcal{P}_c$ projectors is very similar. For each domain, they both require communication only between limited number of domains that are nearby, without need for any global communication. Therefore their application should be scalable to a certain degree. The $\mathcal{P}_c$ projector requires farther reaching communication because of the presence of the $\mathcal{F}$ operator. Application of $\mathcal{P}_c^T$ required after the preconditioner $\mathcal{F}'$ can be done by using the MUMPS feature to solve with transpose of the matrix. The transposed projector yields $\mathcal{P}_c^T = I - \mathcal{G}_c(\mathcal{G}_c^T\mathcal{F}\mathcal{G}_c)^{-T}\mathcal{G}_c^T\mathcal{F}^T$ and the transposed $\mathcal{F}$ operator $\mathcal{F}^T = B\mathcal{A}^{+T}B^T$. Applying $\mathcal{F}^T$ means simply solving the domain matrices $\mathcal{A}_i$ with the transposed flag in MUMPS. The same is then done for the transpose of the $\mathcal{G}_c^T\mathcal{F}\mathcal{G}_c$ matrix. Finally, it should be noted that computing α in (4.47), λ_0 in (4.45), β in (4.55) and λ_1 in (4.60) involves operations that have all been described above.

5.3.4 Convergence

In FETI, same approach is used with regards to convergence checks as in case of the global solvers. Only the relative residual of the nonlinear problem as defined in 4.7 is checked during the decision making about accepting a point as a solution on the FRC curve. The pseudo arc-length constraint function residual is not checked. The solver checks:

$$\frac{\|Z_i(\omega)\tilde{u}_i + \tilde{F}_{int}^{nl}(\tilde{u}_i,\omega) - B_i^T\lambda - \tilde{F}_i\|}{\|\tilde{F}_i\|} \leq tol \tag{5.8}$$

for a given solution candidate $(\tilde{u}_i, \lambda, \omega)$. The norm $\|\cdot\|$ stands for norm over all domains, meaning:

$$\|x_i\| \text{ expands into } \sqrt{\sum_i \|x_i\|^2} \tag{5.9}$$

Additionally, when using FETI it is important to also verify the domain connectivity condition (4.3). The code displays the error of $\|\sum_i B_i\Delta\tilde{u}_i - H_c\|$ at the end of each FETI solve as well as $\|\sum_i B_i\tilde{u}_i\|$ at the end of each Newton loop. However, these errors are not part of the convergence criteria of the Newton solver, they are only displayed to the user for verification. Same is true for relative residual of the actual FETI linear solve.

The GMRES solver used to solve the projected dual problem is given a tolerance parameter for its relative residual, meaning it checks:

$$\frac{\|\mathcal{P}\mathcal{P}_c^T\mathcal{F}'\mathcal{P}\mathcal{P}_c\mathcal{F}\mathcal{P}\lambda_2 - \mathcal{P}\mathcal{P}_c^T\mathcal{F}'\mathcal{P}\mathcal{P}_c(\mathcal{D} - \mathcal{F}\lambda_0)\|}{\|\mathcal{P}\mathcal{P}_c^T\mathcal{F}'\mathcal{P}\mathcal{P}_c(\mathcal{D} - \mathcal{F}\lambda_0)\|} \leq tol \tag{5.10}$$

When this tolerance is reached, the solver stops. However, even if this tolerance is not reached and the solver stops at iteration limit, the computed solution estimate is still used as a valid λ_2 solution and the code continues without any difference.

5.4 Conclusion

The chapter provides an overview of the code that has been developed during this work [21], highlighting several of its key characteristics. The general organisation of the code and its parallelisation model are briefly described. This provides an insight into the various algorithmic components that are required to solve a nonlinear vibration problem, and how they can be logically organised. A pseudocode of the continuation loop is shown, as this is the central part of the code.

Parallel linear algebra framework is next described. Organisation of dofs, implementation and distribution of global system matrices and vectors and communication between MPI ranks is covered. Criteria of convergence for global system are established. Technical specifics of used 3rd party solver packages (PETSc GMRES and MUMPS) and their incorporation into the code are described.

The implementation of the FETI solver is subsequently covered. Emphasis is placed on distribution of both primal and dual variables and their exchange between domains, as this is a crucial aspect in terms of scalability of the solver. Concept of domain, global overlapping and unique dual variable distribution maps is described. Another important topic with regards to parallelism is the communication pattern used for assembly and application of both coarse space projectors $\mathcal{P}$ and $\mathcal{P}_c$ and the dual FETI operator $\mathcal{F}$. Convergence criteria for the FETI solver are established. These criteria are analogous to the ones for the global solvers, with the additional check of domain connectivity error.

Chapter 6

Results

The following chapter presents various results obtained with the code and methodology described in the previous chapters. First, an overview of the testing hardware specifications is presented. Since parallelism is an essential topic of this work, the code was executed on several state of the art supercomputers available at the time. These supercomputers by far exceed computational capabilities of any laptop, desktop computer or workstation. Their relevant technical parameters are presented to provide performance context for the obtained results.

Next, used testcases are presented. Several geometries and forcing configurations were used to assess the code's capabilities. In addition, various number of elements, domains and used harmonics were often tried in order to observe changes in runtime and convergence properties. The code was tested mostly on academic examples, meaning beams and cubes. However, a turbine engine blade geometry was also tested.

After the introduction into the hardware and testcases, the results themselves follow, separated into two parts. First part shows the results for the global solvers, namely MUMPS and PETSc GMRES. The framework for global solvers was developed first as it was simpler, considering the solvers themselves (meaning the matrix factorisation in MUMPS and the GMRES algorithm in PETSc) are 3rd party libraries. Results from the MUMPS solver were verified against several results from other researches. This validated the implementation of the finite element and harmonic balance equations. Scalability of the MUMPS solver was also assessed and a full FRC for a blade geometry was computed, highlighting the importance of the proper choice of harmonics. While the MUMPS solver is capable of handling problems over a million degrees of freedom in total size, it shows its limits in terms efficiency as the problem size grows. The Intel MKL PDSS solver was not tested as there were issues with compiling the code with Intel libraries.

The second part of the results is dedicated to the FETI solver. This solver is still in its prototyping phase and many improvements and optimisations could be made in the future. Nevertheless, it does seem to surpass the direct solver in its capability of handling larger problems. Similar tests to the direct solver were performed, meaning computation of a complete FRC and scalability assessment. Furthermore, additional tests related to particularities of the FETI algorithm were made, such as studying the effect of the artificial coarse space, number of GMRES iterations in various parts of the FRC and others.

6.1 Hardware overview

An overview of the hardware used to obtain the results for this work is presented. The performance benefits of using distributed memory parallelism with libraries such as MPI can only be fully realised when the code runs on dozens and more CPUs. This requires a specialised hardware which by far exceeds performance of any personal computer - a computer cluster or a supercomputer. A supercomputer consists of hundreds or thousands of processors which each contains multiple computing cores. Such computers allow for solving of linear systems with millions and even billions of degrees of freedom, as long as the solver scales with increasing number of MPI ranks. At the time of writing, the largest supercomputer in the world is Fugaku in Riken, Japan, with 7,630,848 computing cores and theoretical peak performance of 537,212 TFlop/s [53]. The unit Flop/s stands for the number of floating point operations per second the system can perform.

Two different supercomputers were used to run tests of ParHBM. Both are located at IT4Innovations, a research institute in Ostrava, Czech Republic. Salomon is the older supercomputer at the site and has already been decommissioned. It was used for the early tests with the global system solved by MUMPS. Karolina, which was installed as a replacement for Salomon, was used for later tests. The key parameters of both systems are enumerated in Tables 6.1 and 6.2.

SALOMON	
Location	Ostrava, Czechia
Installed in	2015
Peak performance [TFlop/s]	2,011
Compute node:	2x12 cores Intel Haswell, 2.5 GHz
RAM per node [GB]	128
Interconnect	Infiniband FDR 56 Gb/s

Table 6.1: Salomon supercomputer information

6.2 Testcase overview

Various aspects of the code's performance were assessed using a set of testcases. Their geometry, physical parameters, external forcing and boundary conditions are described below. Number of elements in their meshes, number of used domains and used harmonics will be specified in the following results sections, as different tests used different configurations, depending on the purpose of the test.

KAROLINA	
Location	Ostrava, Czechia
Installed in	2021
Peak performance [TFlop/s]	15,690
Compute node:	2x64 cores AMD 7H12, 2.6 GHz
RAM per node [GB]	256
Interconnect	Infiniband HDR 200 Gb/s

Table 6.2: Karolina supercomputer information

The first two testcases are a clamped-clamped and a cantilever beam. The clamped-clamped beam is the same as the one discussed in Section 3.7. Beams are commonly used as academic testcases in vibration analysis because of their simple geometry. Many analytical results have been derived for beams which allows for verification of new numerical methods. The clamped-clamped beam has a square profile, and is excited by a single nodal force in the middle, perpendicular to the beam's length. The diagram of the testcase is in Figure 6.1. The cantilever beam is flat and is excited at its free end by a single nodal force in the middle. The diagram is in Figure 6.2. In all cases, the beams were meshed with a regular HEXA20 mesh.

The last testcase is a geometry of a turbine engine fan blade inside a sector of a bladed disk. This geometry is not from any actual commercial blade design, but it has the general shape of a fan blade, so it can be considered as an industrial geometry testcase. The diagram of the testcase is in Figure 6.3. The blade is meshed purely by HEXA20 elements.

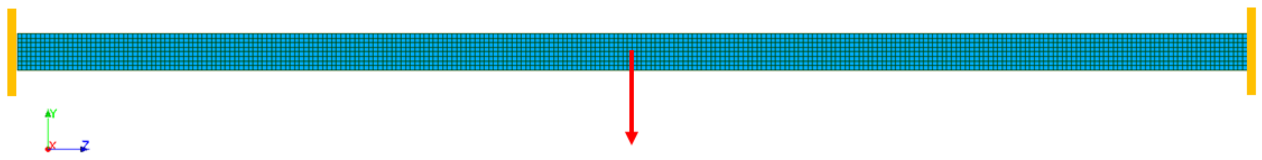


Figure 6.1: Diagram of the clamped-clamped beam testcase. A single nodal force on first harmonic (cos) is applied in the middle of the beam.

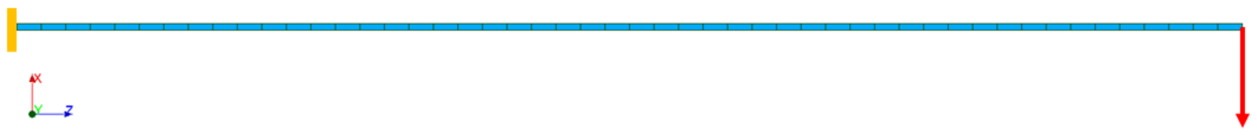


Figure 6.2: Diagram of the cantilever beam testcase. A single nodal force on first harmonic (cos) is applied at the free end of the beam, in the middle.

6.2.1 Note on parameters

Physical and numerical parameters for each testcase are shown individually with each result. A Gauss quadrature rule of order 4 was used universally for all tests for assembling

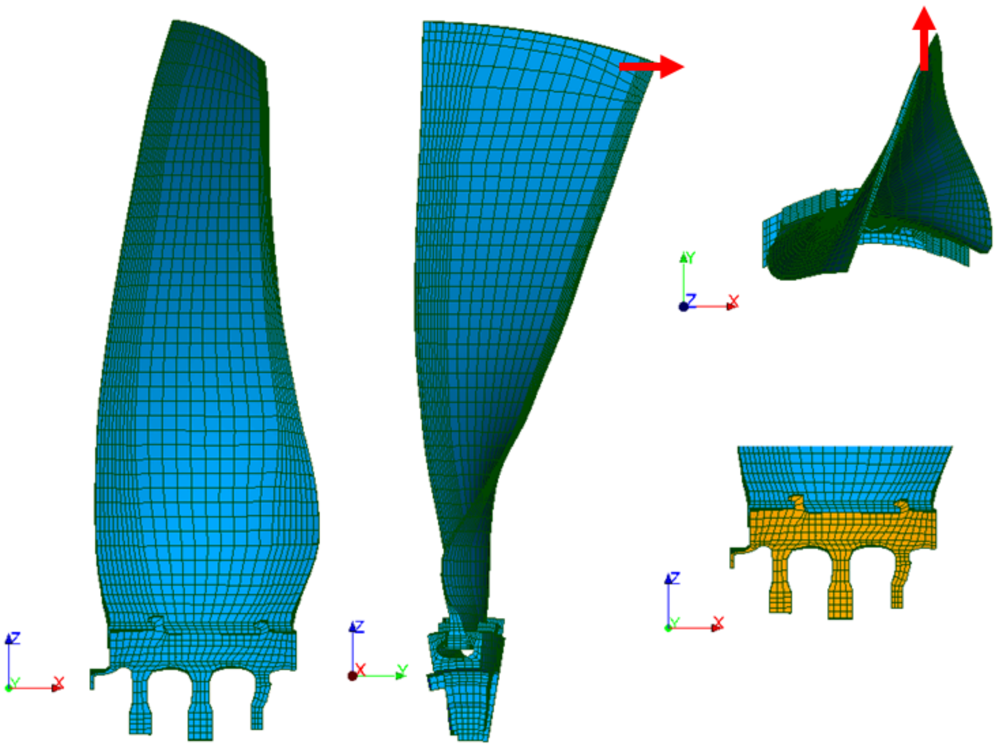


Figure 6.3: Diagram of the fan blade testcase. A single nodal force on first harmonic (cos) is applied at the tip of the blade. Zero Dirichlet boundary condition is imposed at the sides of the disk sector. The contact place between the blade and the disk is removed by fixing the contact nodes together since no contact model was developed for this work.

both K and M matrices as well as the nonlinear force vector F_{int}^{nl}. The size of the continuation step during a continuation run was adapted by 2 fixed coefficients (for up and down scaling) in accordance with Algorithm 2.

For some tests, values of certain numerical parameters (such as number of AFT points, size of continuation step, maximum number of Newton iterations in the corrector) are unknown due to a loss of data at one point during the research. However, the absence of these parameters does not affect the key conclusions drawn from the results of the particular tests. The number of AFT points was always set high enough to properly sample the motion in time (at least 6 points per period of the highest harmonic). For all the scalability tests the number of AFT points is known. The size of the continuation step would only influence the time spent on the computation of the FRCs, as long as it is kept within reasonable bounds and the procedure doesn't skip whole portions of the FRC. None of the testcases dealing with computing a full FRC is used for drawing conclusions about the time efficiency of the code.

6.3 Global solvers results

First, solvers of the global HBM system were tested. As mentioned earlier, implementation of these solvers was simpler and therefore done first. One global HBM Jacobian matrix is assembled for each Newton iteration and is then passed to either MUMPS or PETSc GMRES solver. The results from the global solvers were used to validate the implementation of the nonlinear HBM method in parallel, so it can be in return later used for verification of the FETI solver results. The performance of these solvers was assessed as well. The GMRES solver in PETSc was specifically tested to have a comparison with FETI, which also uses the same GMRES solver to solve its dual problem. Many of the results for the global solvers that are presented here have already been published in [22].

6.3.1 Code verification

Verification of the code was done using both the clamped-clamped and the cantilever beams. In both cases a nonlinear FRC was computed around a specific resonance frequency and it was compared to results obtained by other researchers. At the time, only native continuation loop was implemented in the code, meaning the code only made simple steps in frequency and running Newton with fixed frequency, using previous solution as initial guess. This means that the results obtained for these tests lack turning points. Both verification testcases were computed on Salomon supercomputer and using the MUMPS solver.

The clamped-clamped beam FRC was computed around its 1st resonance frequency. The mesh was $2 \times 2 \times 15$ HEXA20 elements. A forward and backward frequency sweeps were run to obtain as much of the FRC as possible. The computed FRC was compared to an FRC computed by a reduced order modelling technique developed independently by another researcher at Imperial College [184]. The results can be seen in Figure 6.4. Parameters of this testcase are in Table 6.3 and Table 6.4.

Dimensions $[m]$	$0.03 \times 0.03 \times 1$
Density $[kg/m^3]$	7800
Young's modulus $[N/m^2]$	2.1×10^{11}
Poisson's ratio []	0.3
Damping matrix	$D = 3 \times M$
Excitation force amplitude $[N]$	200

Table 6.3: Clamped-clamped beam testcase parameters.

The cantilever beam FRC was also computed around its first resonance frequency. The corresponding linear mode can be seen in Figure 6.6. This verification testcase was designed to match with the testcase in [174]. This work presents a nonlinear FRC for a cantilever beam for various centrifugal loadings, including zero loading, which is the result used for comparison. Their FRC was computed using a numerical time integration rather than

Newton tolerance []	9.9×10^{-7}
Max Newton steps	4
AFT point count	32
Continuation step init	5
Continuation step min	10^{-5}
Continuation step max	5
Continuation step scale up	1.1
Continuation step scale down	0.4

Table 6.4: Clamped-clamped beam testcase numerical parameters.

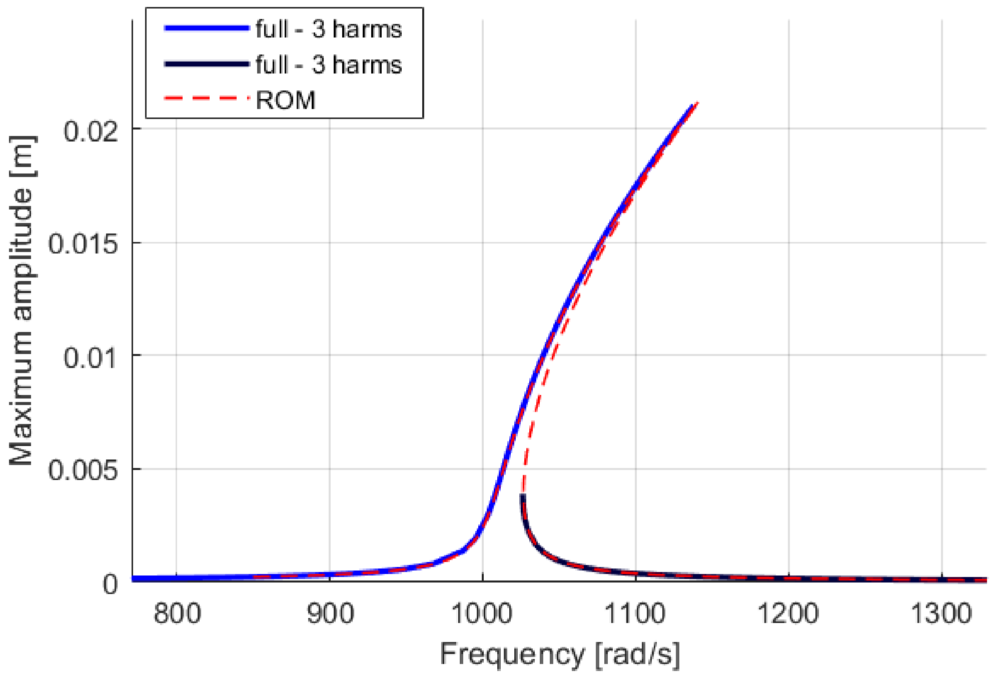


Figure 6.4: FRC for the clamped-clamped beam geometry obtained from ParHBM (blue and black) and FRC obtained for the same testcase using a ROM model.

HBM. All available physical parameters were copied from this paper. Their list is in Table 6.5. Poisson's ratio value was not found in the paper and was therefore chosen to be 0.3, as the authors refer to titanium alloy being the modelled material. Comparison of result FRCs can be seen in Figure 6.5. Data extracted from circle series in top left plot in Figure 6 in [174] were used. Additionally, results from the same ROM code as in the previous testcase were also computed for comparison. Forward and backward sweeps in frequency using harmonics 0123 were computed, as well as forward sweep using harmonics 01234567. However, a different mesh was used as the paper uses beam elements which are not available in ParHBM. The cantilever beam was meshed by $1 \times 8 \times 40$ HEXA20 elements. The ROM model used the same mesh.

Dimensions $[m]$	$0.005 \times 0.1 \times 1$
Density $[kg/m^3]$	4400
Young's modulus $[N/m^2]$	1.04×10^{11}
Poisson's ratio []	0.3
Damping matrix	$D = 0.2467 \times M$
Excitation force amplitude $[N]$	2
Normalising frequency f_0 [rad/s]	24.8945

Table 6.5: Cantilever beam verification testcase parameters.

Newton tolerance	5.2×10^{-4}
Max Newton steps	unknown
AFT point count	unknown
Continuation step parameters	unknown

Table 6.6: Clamped-clamped beam testcase numerical parameters.

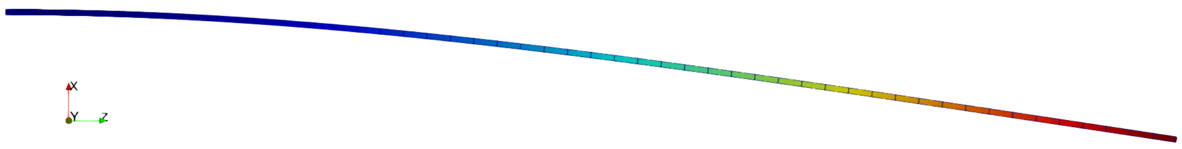


Figure 6.5: First linear mode of the cantilever beam.

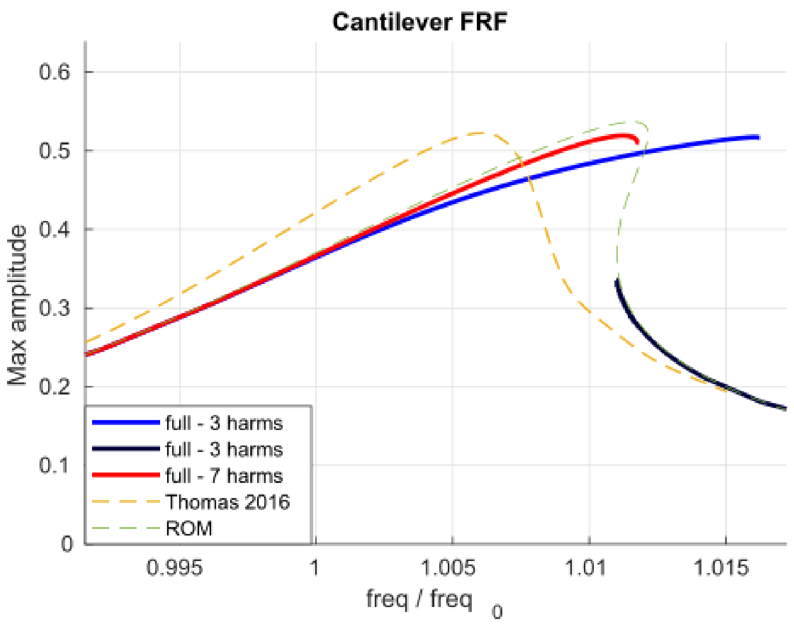


Figure 6.6: FRC comparison for the cantilever beam verification testcase. Yellow and green dashed lines represent results obtained in [174] and by the ROM technique respectively. Full lines represent FRCs computed by ParHBM.

Discussion

The two presented testcases provide a verification of the implementation implementation of the ParHBM code. Results obtained from the ROM model using the same meshes are nearly identical to those from ParHBM. There are larger differences with the result from [174], however, this can be attributed to different meshing and also use of a different computation technique. This gives a reasonable level of confidence in the results obtained from the developed code in terms of it correctly evaluating and solving the HBM terms and equations as described in previous chapters. More thorough testing would however be needed before any serious industrial use of this code. Additionally, note that these results do not prove the validity of the code in a sense that the mathematical model used for this work is sufficient to accurately reflect real life vibrational responses. This problematic is beyond the scope of this research.

6.3.2 Scalability

Scalability is one of the most important properties of any parallel solver. It is a measure of how well the solver maintains its efficiency when the amount of used computing power (meaning CPUs) increases. There are two types of scalability measures - strong and weak. Strong scalability assumes constant total problem size, i.e. the same mesh is used every time, and increasing number of CPUs. In ideal scenario, the total runtime of the solver decreases in direct proportion to the number of used CPUs. The second (weak) type of scalability assumes constant problem size (number of mesh elements) per CPU. The total mesh size therefore grows proportionally to number of MPIs used. In this case, the runtime is in ideal scenario expected to remain constant.

Small scale testcase

The first scalability test was a strong scalability analysis performed on a smaller scale mesh and it focuses on scalability of the AFT procedure. The clamped-clamped beam testcase as presented in Figure 6.1 was used. The parameters of this testcase are different from the one in Table 6.3 and can be found in Table 6.7. This modified version of the testcase was designed later in the research process by scaling the previous testacase. The physical units (time, length and mass) were scaled in order to move the natural frequency close to 1, as well as bring the numeric values of Young's modulus closer to 1 (in terms of orders of magnitude) in order to achieve similar magnitudes of numerical values in K and M matrices. This scaling was an attempt to improve conditioning of the HBM matrices. Karolina was used as hardware for this test and the newer version of MUMPS (5.4.1) was employed.

Two different setups were used in terms of number of nodes, used harmonics, number of AFT time integration points and used MPIs. They are both described in Table 6.8.

For both cases one nonlinear solve was performed using Newton tolerance 5×10^{-6} and requiring 4 Newton iterations. The frequency was picked as 0.98. As seen in Table 6.8, the scalability measurements start from 1 MPI, meaning that this testcase can be used as a

Dimensions [m]	$0.003 \times 0.003 \times 0.1$
Density [kg/m^3]	39
Young's modulus [N/m^2]	10.5
Poisson's ratio []	0.3
Damping matrix	$D = 3 \times 10^{-3} \times M$
Excitation force amplitude [N]	1×10^{-10}
Elements per domain	$2 \times 2 \times 4$

Table 6.7: Clamped-clamped beam scaled testcase parameters.

	Setup 1	Setup 2
Mesh elements	$2 \times 2 \times 512$	$2 \times 2 \times 32$
Harmonics	0123	0123456789
AFT points	32	128
MPIs	$1, 2, 4, 8, 16, 32, 64, 128, 256$	$1, 2, 4, 8, 16, 32, 64, 128$
Freq. domain dofs	$323, 001$	$55, 917$

Table 6.8: Small scale scalability testcase parameters.

comparison benchmark with standard serial software implementations of HBM and AFT procedures.

The results are presented in the following figures. Figure 6.7 shows the runtime of the code and its distribution among the most important tasks for the first setup. Figure 6.8 then the corresponding time efficiency of those same tasks in relation to the 1 MPI case. Efficiency of 100% represents ideal strong scalability. Figures 6.9 and 6.10 then show the same data for the second setup.

Discussion

Figures 6.7 and 6.9 show that the majority of the runtime is spent on assembly of the Jacobian matrix. Especially for the second setup, which has a larger number of AFT points, this task takes over 96% of the total runtime in the 1 MPI (i.e. serial) run. The AFT procedure for the Jacobian matrix dominates the runtime percentage for almost all MPI cases in both setups. Only for the highest MPI counts there is another task with a percentually significant time cost - matrix factorisation in the linear solver. It should be noted that the implementation of the AFT procedure is most likely not very well optimised and the runtimes presented here could be improved.

While the AFT procedure for the Jacobian matrix is the most time consuming task in both testcases, it is also the most scalable task, as can be seen in the efficiency Figures 6.8 and 6.10. Same is true for the assembly of the residual vector assembly which uses the same AFT procedure but applied only to a vector instead of a matrix. On the other hand, the scalability of the linear solver tasks (analysis, factorisation and solve) quickly deteriorates

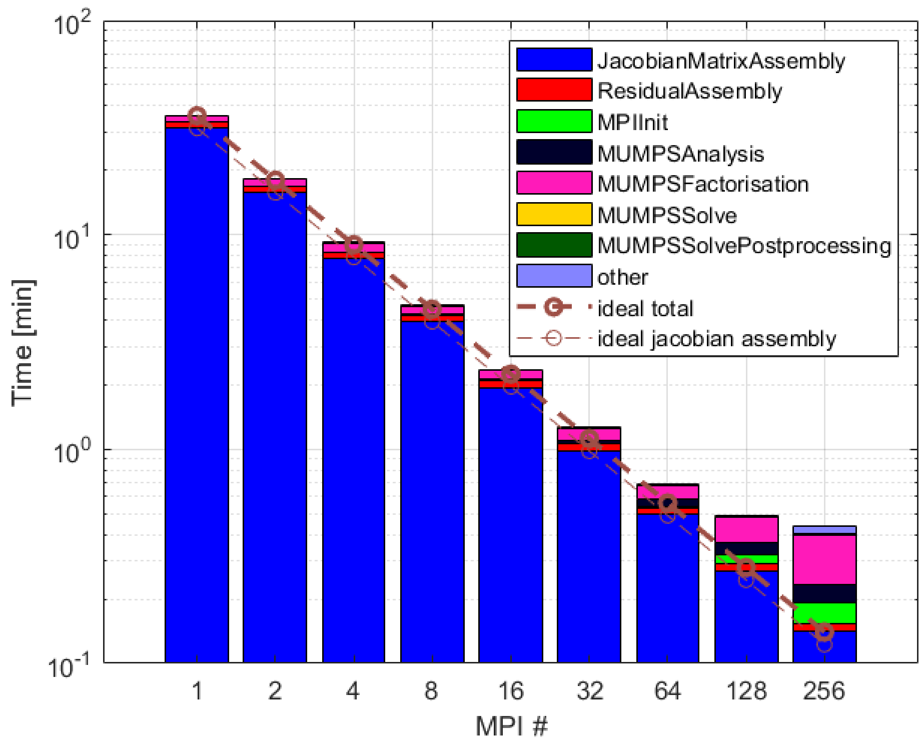


Figure 6.7: Code runtime for the first setup of the small scalability test. The blue colour represents the most expensive task of all - the assembly of the Jacobian matrix. Y axis is in logarithmic scale.

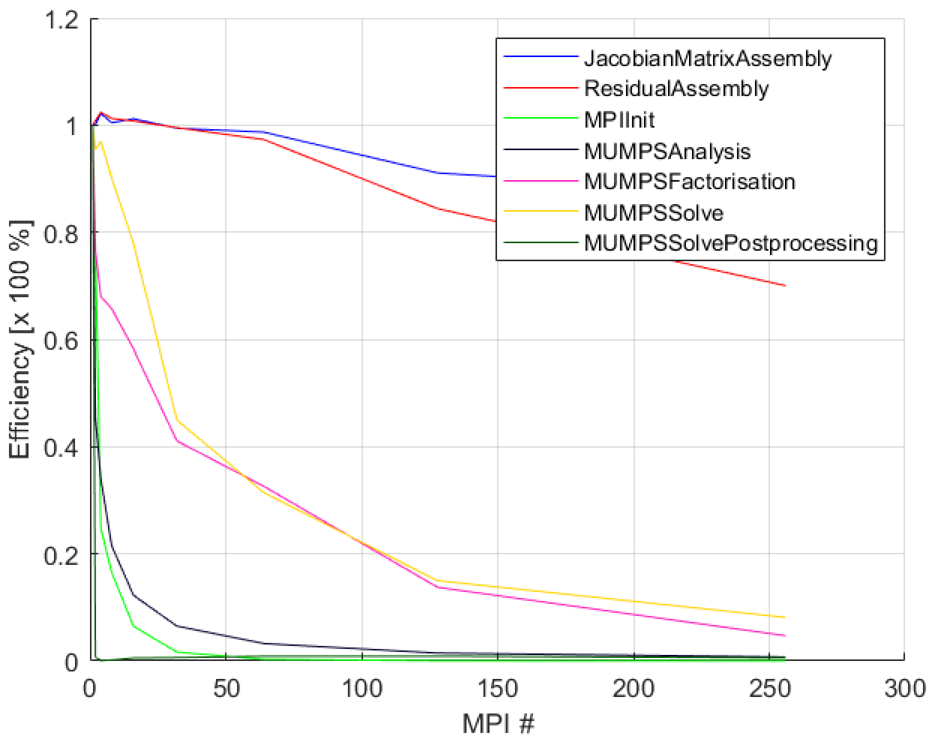


Figure 6.8: Code efficiency analysis for the first setup of the small scalability test.

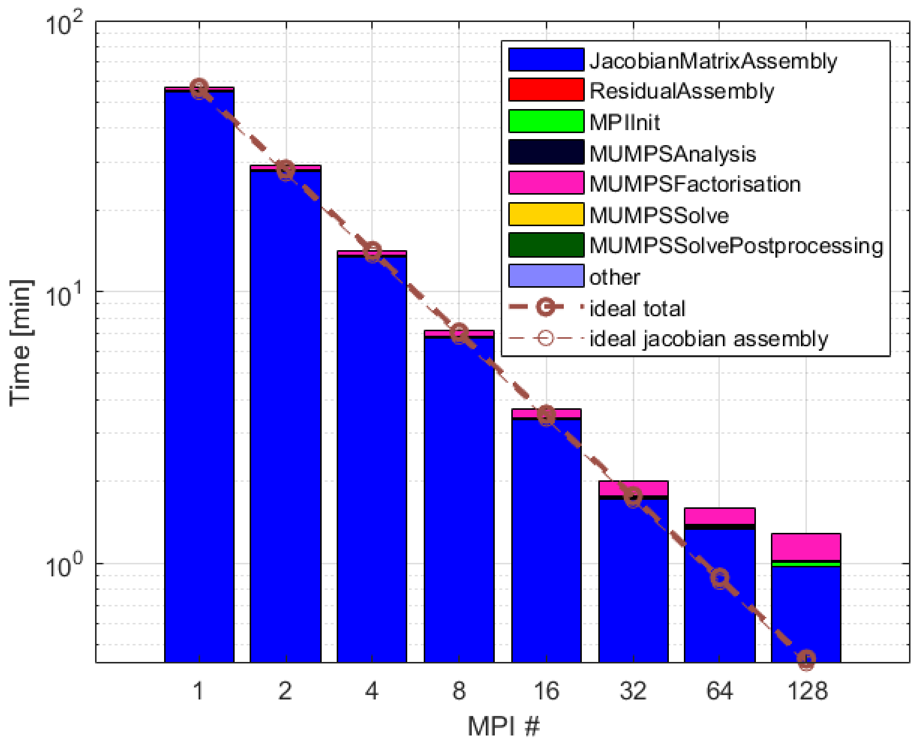


Figure 6.9: Code runtime for the second setup of the small scalability test. The blue colour represents the most expensive task - the assembly of the Jacobian matrix. Y axis is in logarithmic scale.

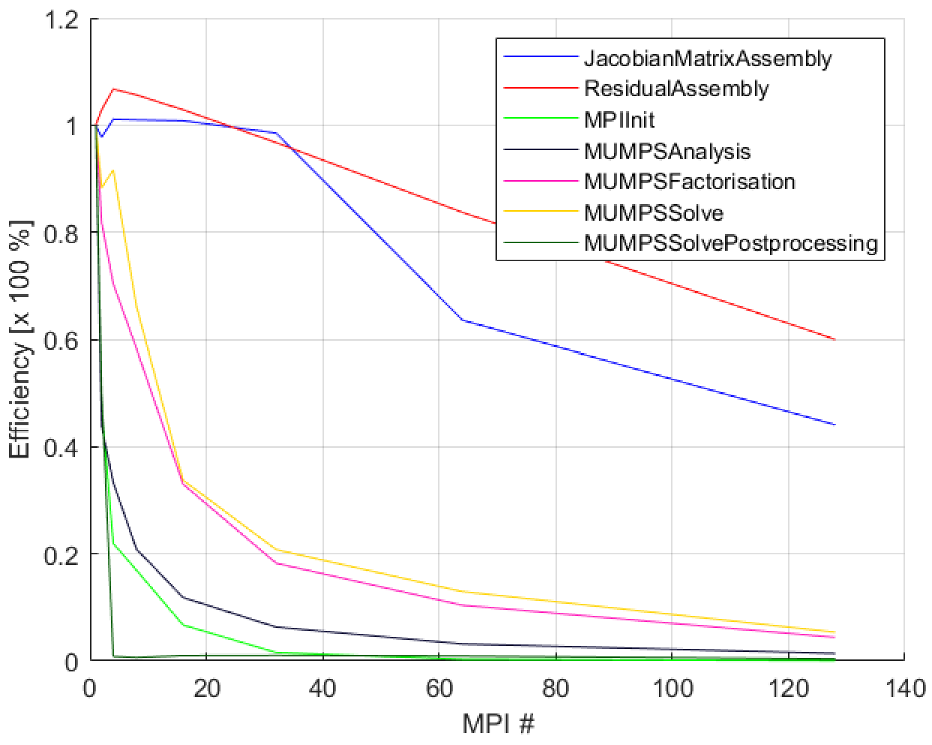


Figure 6.10: Code efficiency analysis for the second setup of the small scalability test.

with rising MPI count. This will be further explored in the following section which analyses the scalability on a larger scale mesh.

Large scale testcase

The scalability of solving the global system for a large scale problem was assessed by performing a strong scalability test on the MUMPS solver. Version 5.1.2 was used for this test. The GMRES solver used for the global system was not tested for its lack of performance, which is shown later in Section 6.3.4. This test was run on Salomon using the clamped-clamped beam testcase, with parameters being the same as in the validation testcase (see Table 6.3). The beam mesh was regular HEXA20 mesh with $8 \times 8 \times 266$ elements. This corresponds to 81,621 nodes. A nonlinear problem for 1 frequency point (980 rad/s) was solved, requiring 4 Newton iterations to reach required tolerance. Readers can refer to Figure 6.4 for its approximate location on the FRC. Harmonics 0123 were again used, like in the validation test, meaning the total size of the linear system was 1,714,041 dofs. The AFT procedure used 32 time points. Mesh decomposition was performed by libMesh library which internally uses Parmetis. The list of MPIs used was 4, 8, 16, 32, 64, 128, 256, 384. The resulting runtimes can be seen in Figure 6.11 and Table 6.9. Various parts of the code were timed separately for more detailed evaluation of the solver. Each of the MUMPS' phases (analysis, factorisation and solve) were timed individually, as well as preprocessing and postprocessing required before and after the solve phase. Preprocessing involves for instance communicating the entire right hand side vector to one rank. Postprocessing includes redistributing the solution from the MUMPS distribution to the code distribution. The same data interpreted as achieved speed-up with respect to the lowest MPI count are shown in Figure 6.12.

MPI	MUMPS factorisation	MUMPS analysis	Jacobian matrix assembly	Total time
4	4,158	137	710	5,279
8	2,286	89	357	2,877
16	1,706	71	177	2,035
32	982	53	92	1,173
64	862	47	47	986
128	636	66	25	753
256	460	168	14	678
384	416	181	11	664

Table 6.9: MUMPS strong scalability runtime. All times are in seconds.

Discussion

The direct solver was shown to provide a significant parallel speed-up, from approximately 1.5 hours for 4 MPIs to 0.18 hours for 384 MPIs. However, the efficiency is still lacking as can be seen in the speed-up plot. The most time consuming part of the code is the matrix

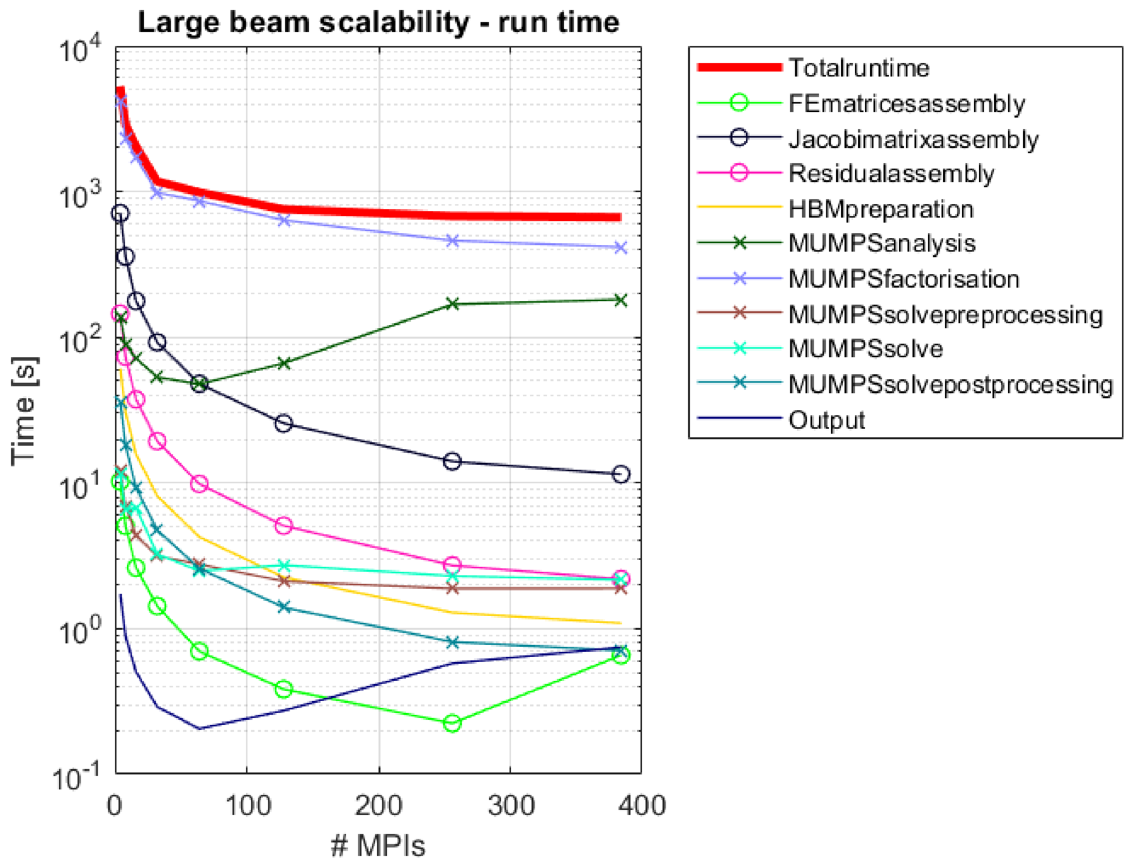


Figure 6.11: MUMPS global solver strong scalability results - runtime. Various parts of the code were timed separately.

factorisation, which is understandable as it is the main part of solving the linear system. This part seems to not scale very well and therefore damages significantly the scalability of the whole solver. The analysis phase of MUMPS also seems to scale badly especially for the higher end of the MPI counts. The Jacobian matrix assembly, which includes the AFT procedure, is another time consuming part. This however seems to scale quite well, which confirms the previously shown results on the smaller scale meshes. These observations lead to the conclusion that the solver is the most critical part of the whole HBM code when it comes to scalability for large scale meshes.

It is possible that this testcase was actually too small for MUMPS in terms of size per domain for the higher numbers of MPIs. However, an issue was observed with MUMPS regarding memory consumption. The analysis and factorisation phase require large amount of RAM for their operation. The scalability test presented in this section was run on 24 nodes at Salomon, meaning 768 cores. However, only 384 cores were used at max. When running the case on all 768 cores (using 768 MPIs), the solver ran out of memory. This means that the memory requirements were higher than 4096 GB that were in total available on those nodes. The assembled Jacobian matrix only occupied a small portion of this amount.

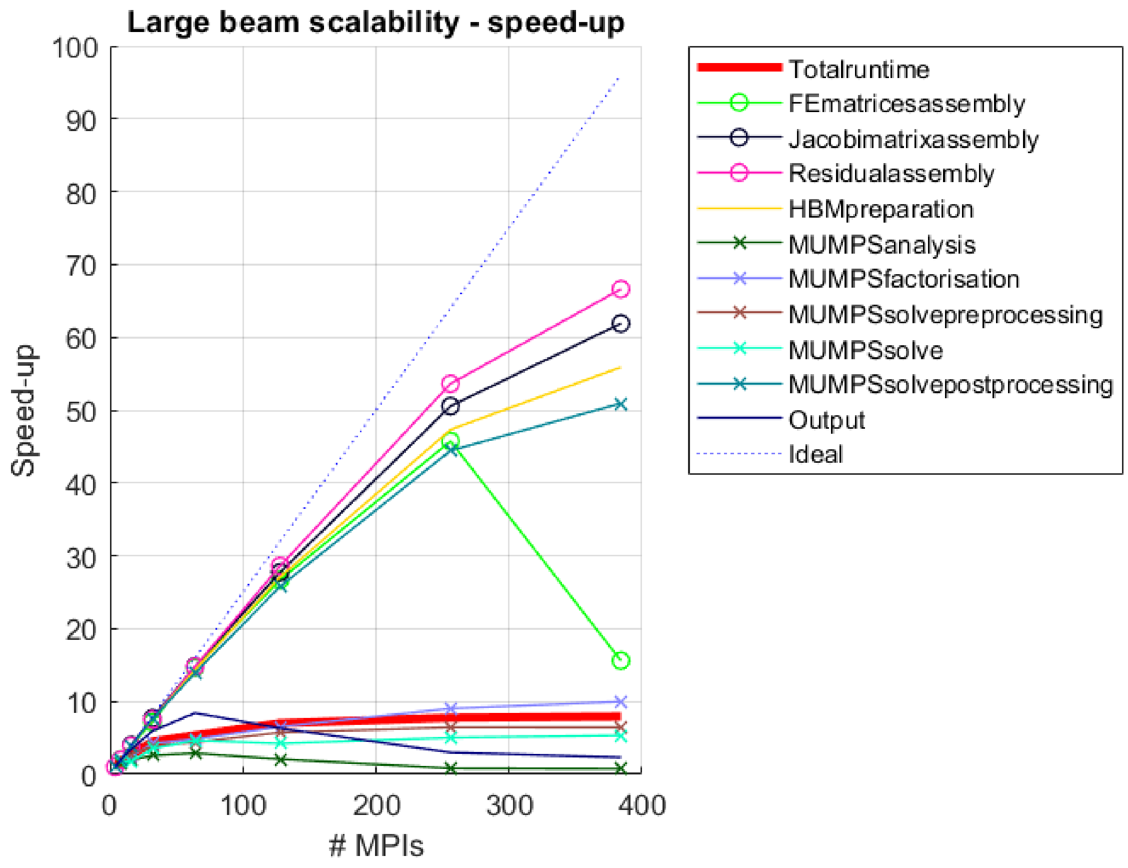


Figure 6.12: MUMPS global solver strong scalability results - speed-up. Various parts of the code were timed separately.

6.3.3 Blade FRC

The global system together with MUMPS was also used for computing a nonlinear FRC for the fan blade testcase. FRC was computed around the first mode of the blade. The parameters of this test can be seen in Table 6.10. This testcase was run on 2 nodes of Salomon, meaning 48 MPIs were used.

Density $[kg/m^3]$	4429
Young's modulus $[N/m^2]$	1.15×10^{11}
Poisson's ratio []	0.32
Damping matrix	$D = 0.3981004 \times M$
Mesh elements	7,722
Mesh nodes	41,021
Newton tolerance	between 8.6×10^{-4} and 4.3×10^{-3}
AFT point count	20 (H0123), unknwon (H012345)
Continuation step size params	unknown

Table 6.10: Blade FRC testcase parameters.

Excitation force was applied at one node at the tip of the blade as seen in Figure 6.3. The testcase consists of two parts. In first part, the FRC was computed for various amplitudes of the excitation force, using harmonics 0123 in all cases. For this part, an older version of ParHBM was used which only used naive prediction and did not have the ability to turn around at turning points. The FRCs were therefore computed in two separate sweeps - forward and backward, to obtain as much of the FRC as possible. The resulting curves are in Figure 6.13. Total code runtimes for each FRC are in Table 6.11.

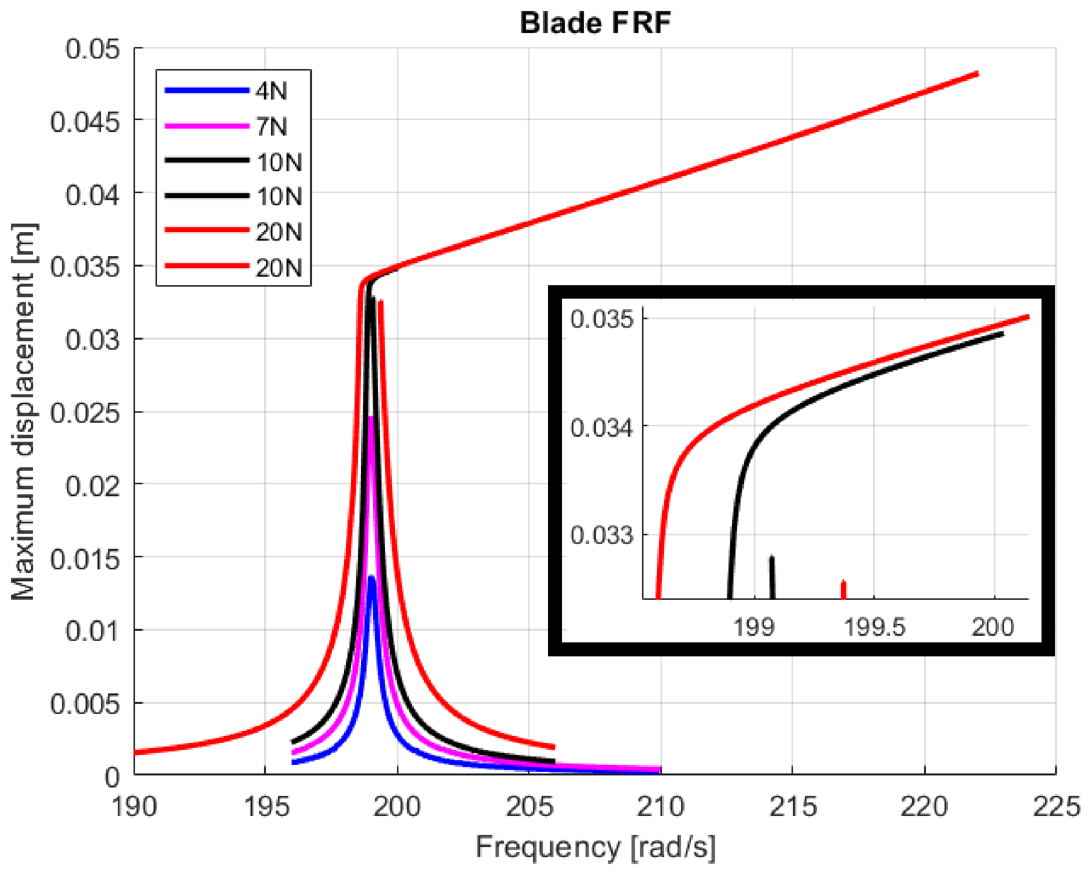


Figure 6.13: Blade testcase nonlinear FRC for various forcing amplitudes, using harmonics 0123.

Following these results, another FRC computation was performed, using the largest 20N force amplitude, and comparing harmonics 0123 and 012345. This computation was performed using a newer code version where turning points were already supported. The computed FRCs can be seen in Figures 6.14 and 6.15.

Discussion

The fan blade FRC test was performed to demonstrate that the solver can work with a geometry resembling a realistic industrial shape, as the other testcases are simple beams. The size of this mesh is still smaller than typical sizes of meshes used in actual industrial computations. However, even on this size the computation of an entire FRC around one resonance frequency took multiple hours to complete. This shows the time scale required for such computations and why efficient parallelisation techniques are required. In this

Curve	**Total time [h]**	**# points**
4N	4.73	146
7N	6.62	147
10N (f)	14.23	120
10N (b)	6.32	85
20N (f)	106.73	906
20N (b)	6.42	80

Table 6.11: Computation times for the blade FRCs. The (f) and (b) marks indicate forward and backward frequency sweep. Note that these times depend on the number of AFT points as well as the size and the method of adjustment of the continuation step.

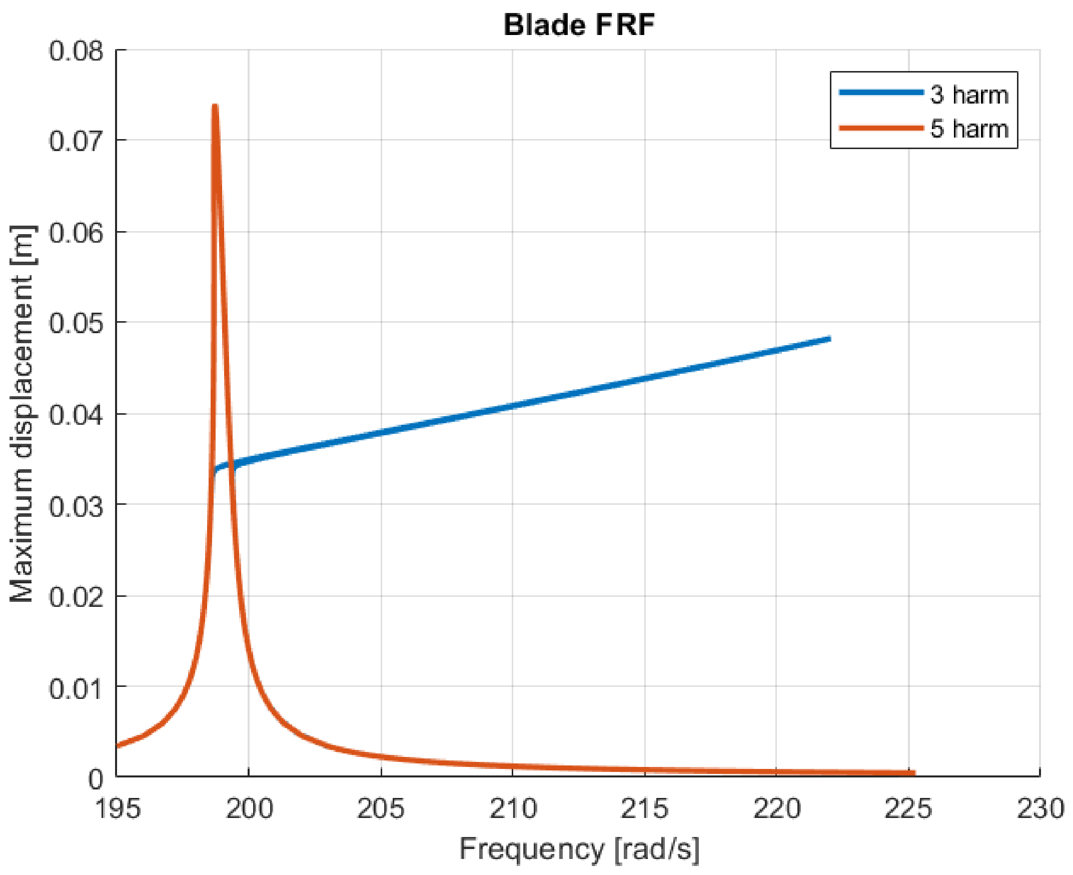


Figure 6.14: Blade testcase nonlinear FRCs for 20N forcing. Comparison of harmonics 0123 and 012345 solutions.

testcase, only 48 MPIs were used, since the previous scalability test showed that the benefits of using more MPIs deteriorate rapidly after this point.

An interesting observation can be made regarding the number of harmonics used to model the vibration. As seen in Figure 6.14, using only harmonics 0123 leads to a sudden stiffening effect of the response, which is no longer present when harmonics 4 and 5 are added. The response computed using harmonics 012345 has a shape that is expected physically, as structures of this type typically manifest a softening effect. Clearly, harmonics 0123 are not

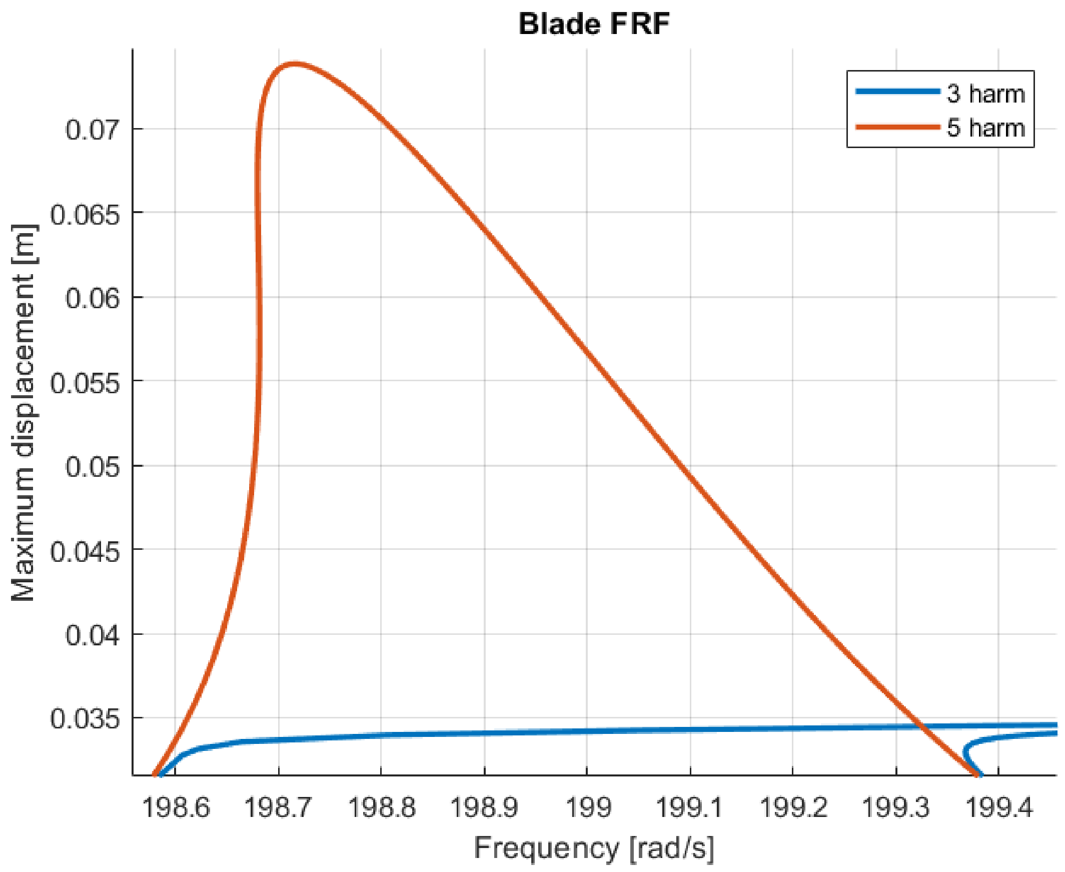


Figure 6.15: Blade testcase nonlinear FRCs for 20N forcing - detail. Comparison of harmonics 0123 and 012345 solutions.

enough in this case, probably due to harmonics 4 and 5 being significant components of the nonlinear motion as the amplitude increases. Since they are not included, that higher harmonic motion component is locked and this creates an artificial stiffening effect. This shows the importance of selecting the right set of harmonics to accurately capture the character of nonlinear vibration. An additionally testcase study further elaborating on this problematic can be found in the appendix A.

6.3.4 GMRES performance on global system

All the results presented to this point were computed using the MUMPS solver. However, since GMRES was also added to ParHBM in order to solve the projected dual problem in FETI, it was also tested on the global problem to see how an iterative Krylov space solver would perform. For this test, the clamped-clamped beam testcase was used. A linear and nonlinear HBM problem for 1 frequency point were solved on Karolina, using increasing number of MPIs. Number of elements per domain was fixed, so this test can be considered to be a weak scalability test. Numbers of GMRES iterations were measured for each Newton iteration, as well as the total runtime of the code. The scaled version of beam testcase parameters from Table 6.7 was used. 32 time points were used for AFT in all tests. The first resonance frequency of this beam is around 1.003, depending on the exact used mesh. The meshes were generated by the ParHBM's regular mesh generator, decomposing the mesh into domains purely along the long axis of the beam (Z axis). Numbers of used MPI ranks were 8, 16, 32, 64 and 128. The frequency picked for this test was 0.98. Its position on the beam FRC can be seen in Figure 6.16.

PETSc offers various preconditioners which can be enabled simply by setting a flag in the code. Several of such preconditioners available for GMRES were tested. However, none of them was tuned in any way for the HBM problem. Only the flag for the particular preconditioner was switched on and all possible parameters were left to their default values determined by PETSc. Therefore the results obtained are only rough estimates and the solver performance could probably be improved by custom tailoring the preconditioners.

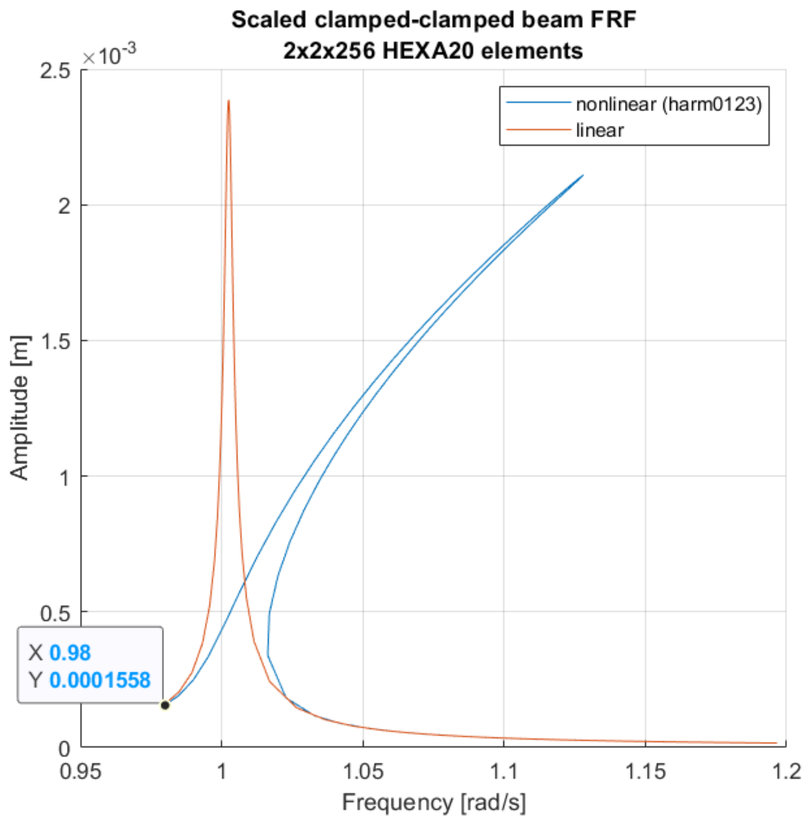


Figure 6.16: Nonlinear FRC of the scaled clamped-clamped beam, measured in its middle along the Y axis (axis of excitation force). This particular FRC was computed for 2x2x256 HEXA20 elements. It can differ slightly for different mesh sizes. The highlighted frequency point 0.98 is where the tests for single frequency were run.

First, the linear test was run, meaning that only harmonic 1 was used. GMRES was used without preconditioning, and then with block Jacobi, geometric algebraic multigrid (GAMG) and additive Schwarz method (ASM) preconditioners. Required number of GMRES iterations for each variant can be seen in Figure 6.17. The number was capped at 1600 to avoid GMRES running for too long, as the solver slows down with iterations because of the growing space used for direction orthogonalisation. For the largest case (128 MPIs) the achieved relative residual norm is in Figure 6.18. Tolerance of 10^{-8} was required as a stopping criteria, but was not reached in many cases. The total runtime of the code is in Figure 6.19. For reference, the same problem was also solved using the MUMPS solver.

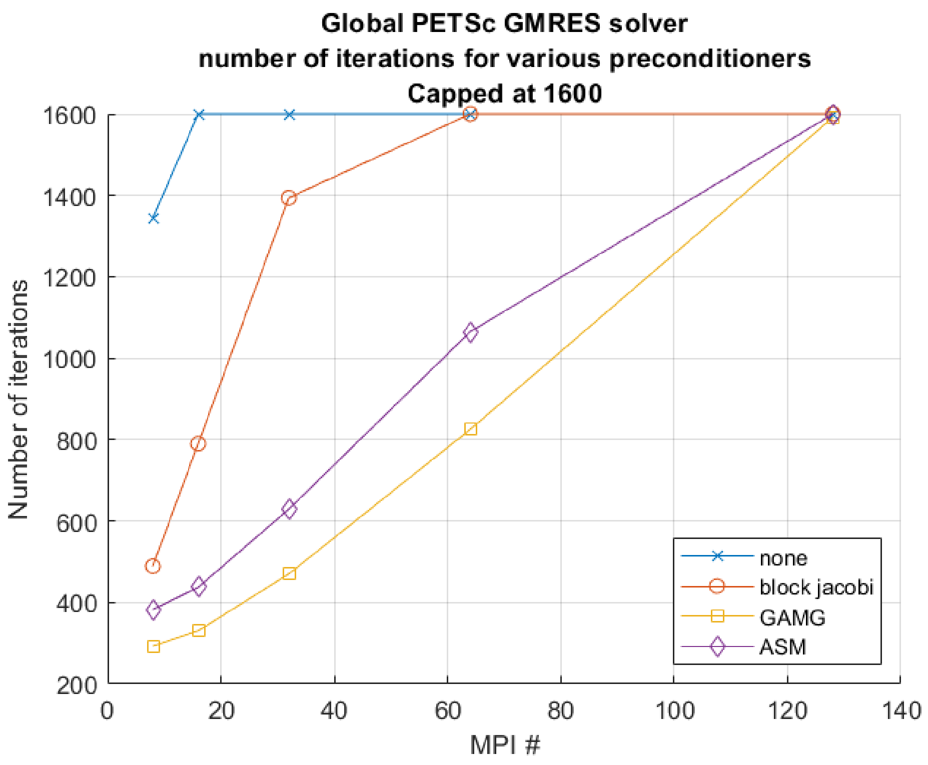


Figure 6.17: Number of GMRES iterations performed by the GMRES solver for the global linear HBM problem for various MPI count.

For the nonlinear test, harmonics 0123 were used. The maximum number of GMRES iterations was extended to 3200. Aside from GMRES without preconditioning only the GAMG preconditioner was tested as this one performed the best in terms of convergence in the linear testcase. The maximum number of Newton iterations was set to 4. The requested relative residual tolerance for the linear solver in each Newton iteration was again set to 10^{-8}. The relative residual tolerance for convergence of Newton solver was set to 5×10^{-6}. In all cases, all 4 Newton iterations were required. Figures 6.20 and 6.21 show envelopes of relative GMRES residuals over all Newton iterations, meaning for each GMRES iteration the maximum and minimum relative residual across Newton iterations is plotted. It can be seen that for larger number of MPIs GMRES struggled to converge to a reasonable tolerance. Final reached residuals in Newton loop after the 4 iterations are shown in Figure 6.22. MUMPS results are again added for comparison. Total runtimes of the code are in Figure 6.23.

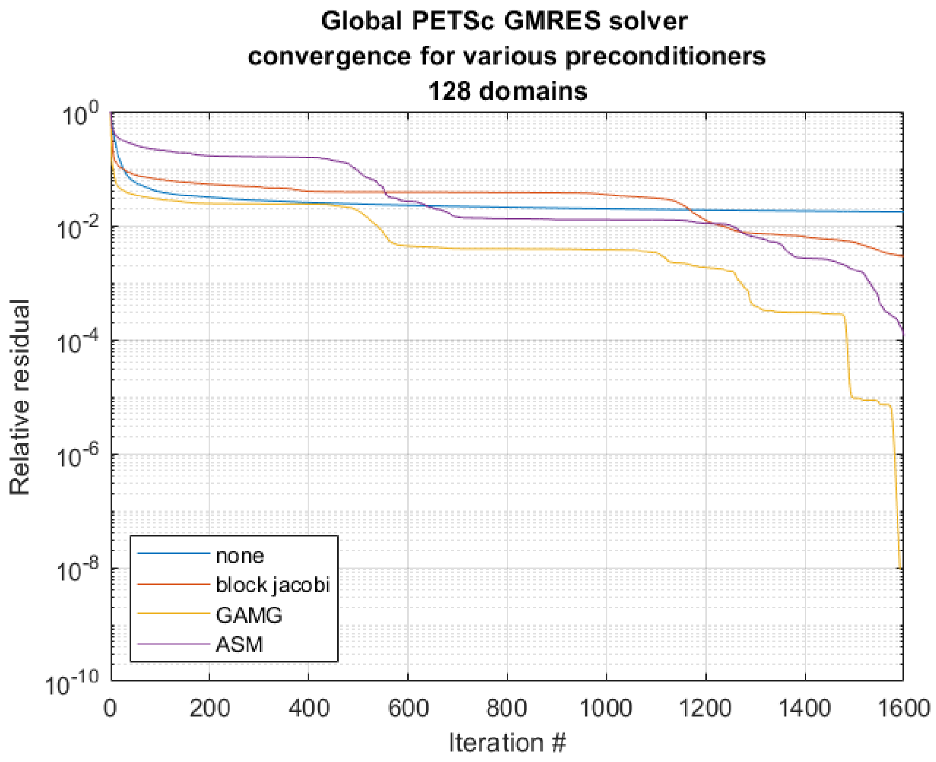


Figure 6.18: GMRES convergence curves for the global linear HBM problem for 128 domains/MPIs.

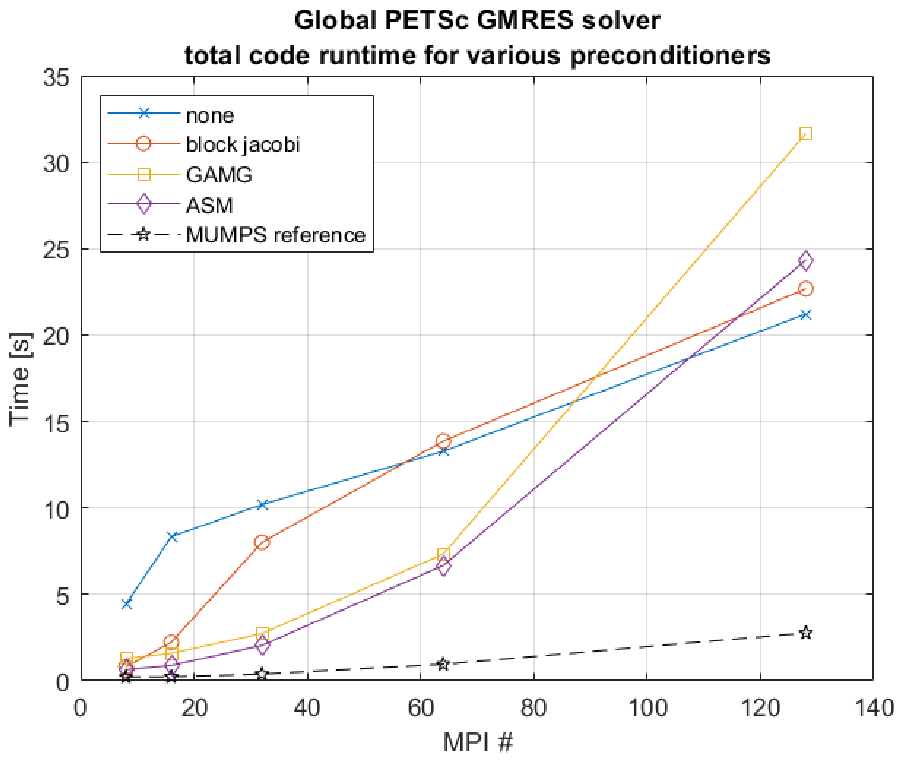


Figure 6.19: Total ParHBM runtime for global linear HBM problem. Comparison of GMRES solver with various preconditioners and MUMPS solver.

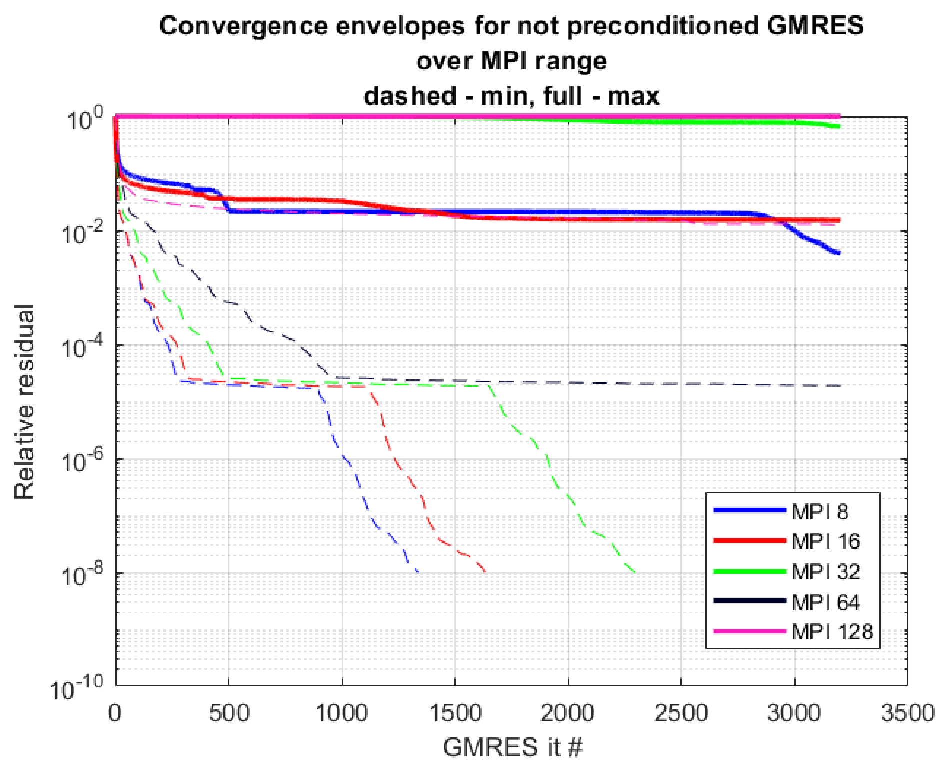


Figure 6.20: Envelopes of relative GMRES residuals across Newton iterations for varying MPI size for non-preconditioned GMRES for the nonlinear global system testcase.

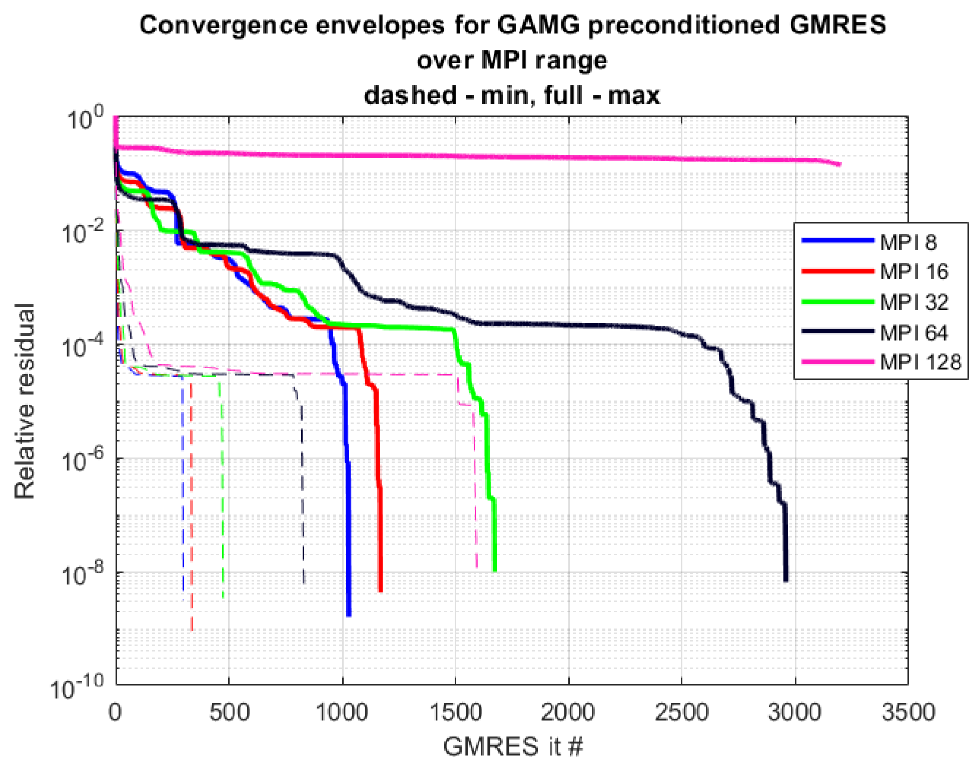


Figure 6.21: Envelopes of relative GMRES residuals across Newton iterations for varying MPI size for GAMG preconditioned GMRES for the nonlinear global system testcase.

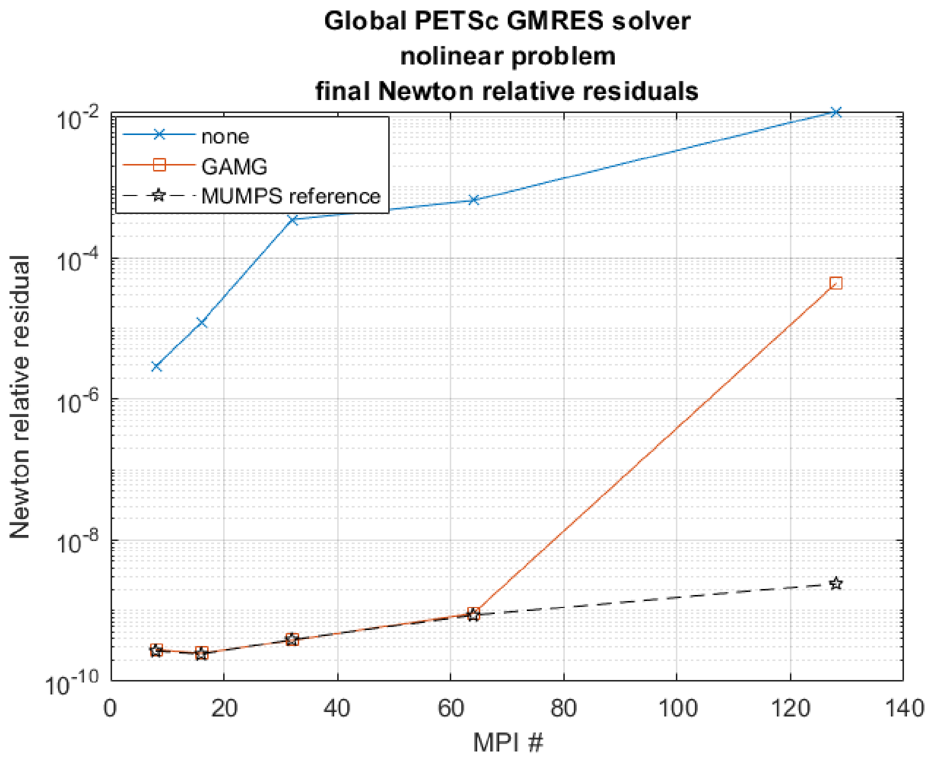


Figure 6.22: Global nonlinear HBM problem. Final relative residuals reached by Newton loop after 4 iterations for non-preconditioned GMRES, GAMG preconditioned GMRES and MUMPS for varying MPI size.

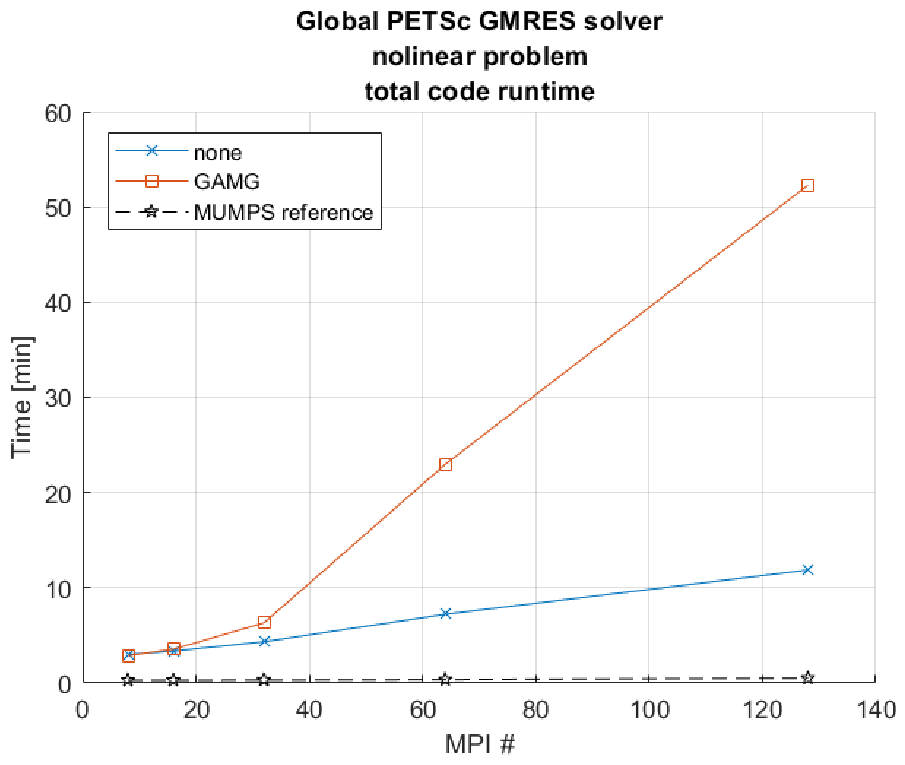


Figure 6.23: Total ParHBM runtime for global nonlinear HBM problem. Comparison non-preconditioned GMRES, GAMG preconditioned GMRES and MUMPS.

Discussion

It can be seen on both the linear and the nonlinear testcase that the GMRES solver on the global HBM system struggles to reach any reasonable tolerance in terms of relative residual. The only version that manages to reach the desired tolerance is GMRES preconditioned with GAMG. However, in all cases, the number of required GMRES iterations grows rapidly with number MPIs. In the nonlinear case this inevitably affects convergence of the outside Newton solver. When comparing with the MUMPS reference results, it is clear that MUMPS is outperforming GMRES in both time performance and Newton convergence. As noted before, the preconditioning of the GMRES solver was done in the simple way of switching appropriate flags. It is possible that with a more sophisticated and tuned preconditioning the performance could improve. The FETI algorithm is actually sometimes described as a preconditioner for Krylov based iterative solvers.

6.4 FETI solver results

The last portion of results is dedicated to the FETI solver. There are many aspects of the solver to be studied as the algorithm has a certain level of complexity. The results can also be heavily impacted by specifics in the code implementation, especially in handling the communication of data between MPI ranks. The key point of interest is the ability of the solver to solve large systems and its scalability. Additionally, a study of the effect of preconditioners and coarse space is presented. A case of full FRC computation using this solver is also included. All results in this section were computed on Karolina supercomputer. The newer version of MUMPS (5.4.1) was used.

6.4.1 Coarse space and preconditioner effect

Before assessing the performance of the solver it is important to compare various options available in terms of its configuration. As it was discussed in section 4.5, it is possible to solve the dual problem using only the natural coarse space $\mathcal{G}$, meaning solving the equation (4.50). However, it was argued that adding the artificial coarse space would be beneficial for convergence rate. This coarse space can be added only for nonzero harmonics, or it can also have vectors on the 0th harmonic, as it was discussed in section 4.7. The interface rotations were used in that case. In section 4.6 two possible preconditioners were introduced - the lumped and Dirichlet preconditioner. A comparison is made here between using one of those preconditioners (and neither) and using the artificial coarse space $\mathcal{G}_c$ in various forms (and not using it at all).

For this comparison, a testcase similar to the one specified in Table 6.7 is used, in 2 versions. A nonlinear problem is solved, using harmonics 0123 and solving for frequency 0.98 (see Figure 6.16). Number of domains is 32 and 128 and number of elements per domain is $2 \times 2 \times 4$ and $2 \times 2 \times 8$ respectively. Tolerance for relative GMRES residual was set to 10^{-5}.

Tolerance in the outside Newton solver was set to 5×10^{-6}. 32 time points were used in AFT for all cases. In the first smaller testcase, the maximum number of GMRES iterations was set large enough that it was never reached. In the larger testcase, it was limited to 800.

For the smaller testcase, the GMRES convergence curves for all Newton iterations are in 3 following figures, split by used artificial coarse space type. Figure 6.24 shows data for case when no artificial coarse space was used, comparing all 3 preconditioner options (none, lumped and Dirichlet). Figure 6.25 shows the same comparison for artificial coarse space used only on nonzero harmonics. Figure 6.26 then shows data for artificial coarse space used on all harmonics. Figures 6.27 and 6.28 then show summary of total GMRES iterations performed and total time spent in the GMRES solver for all 9 tested variants. In the time graph, times spent on preparing the artificial coarse space projector $\mathcal{P}_c$ and preparing the Schur complement for the Dirichlet preconditioner are also included. Newton solver converged successfully in all 9 variants.

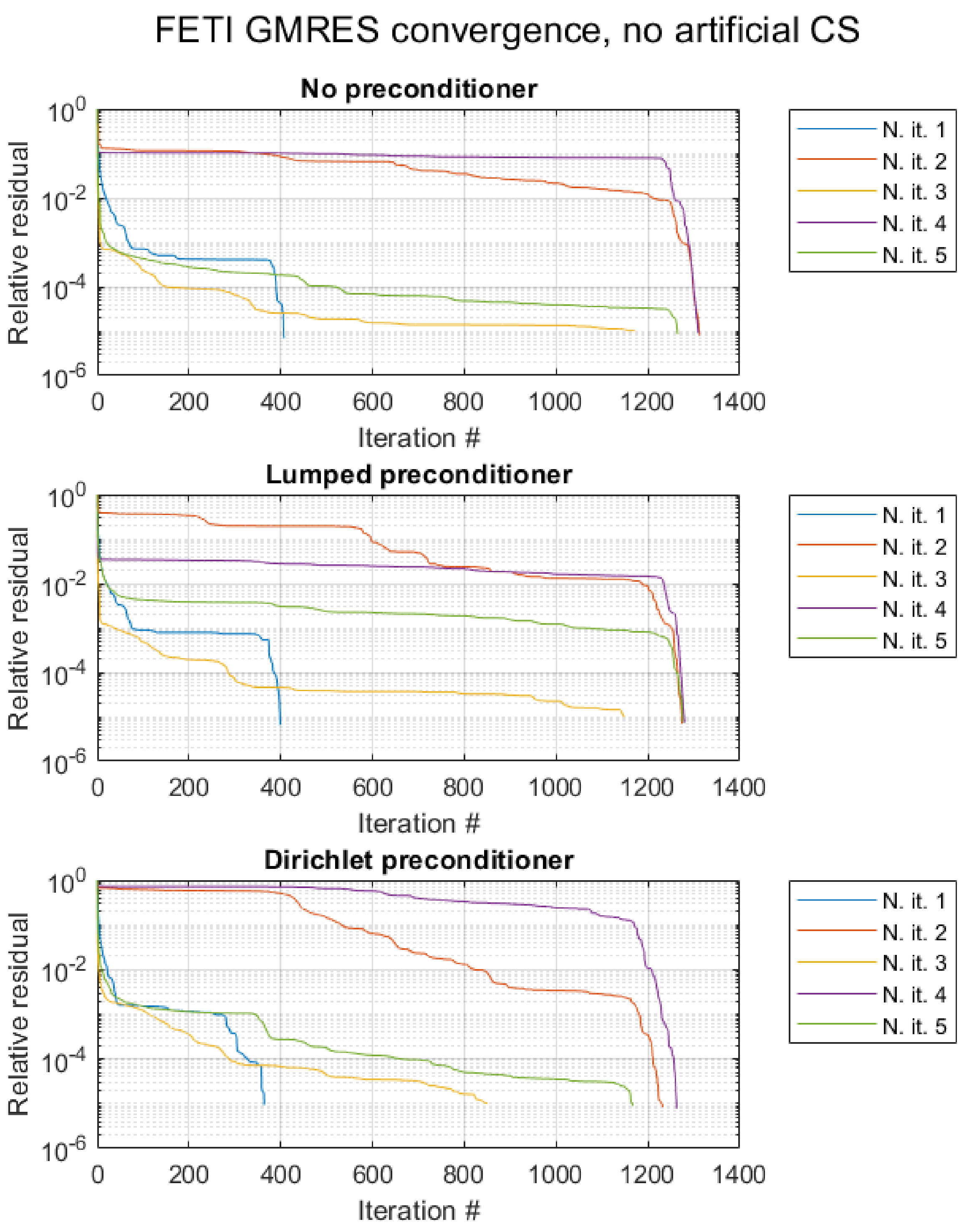


Figure 6.24: GMRES convergence in FETI when no artificial coarse space is used, 32 domain testcase. Comparison for different preconditioner variants. Convergence for each Newton iteration for 1 frequency point solve is shown.

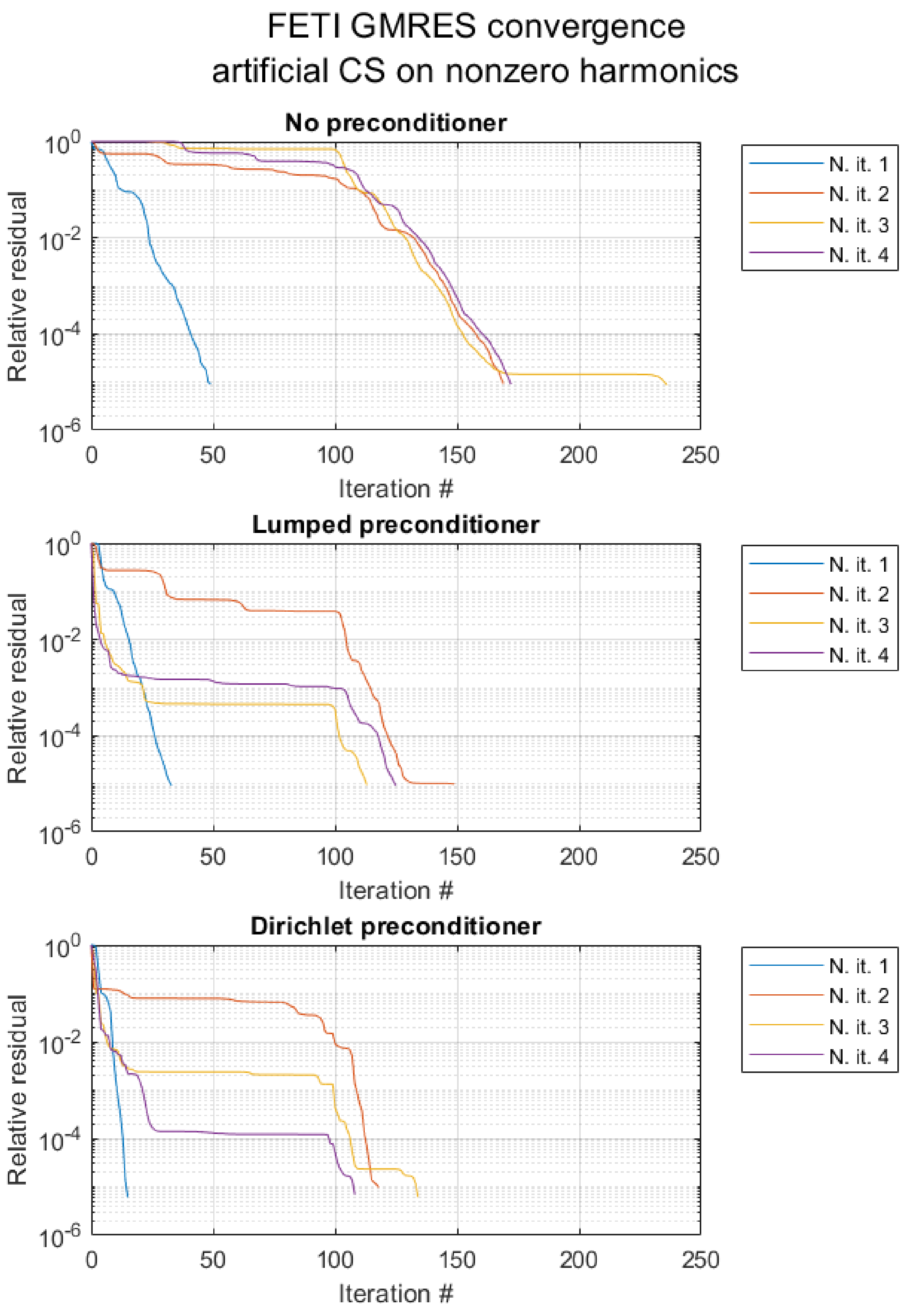


Figure 6.25: GMRES convergence in FETI when artificial coarse space on nonzero harmonics is used, 32 domain testcase. Comparison for different preconditioner variants. Convergence for each Newton iteration for 1 frequency point solve is shown.

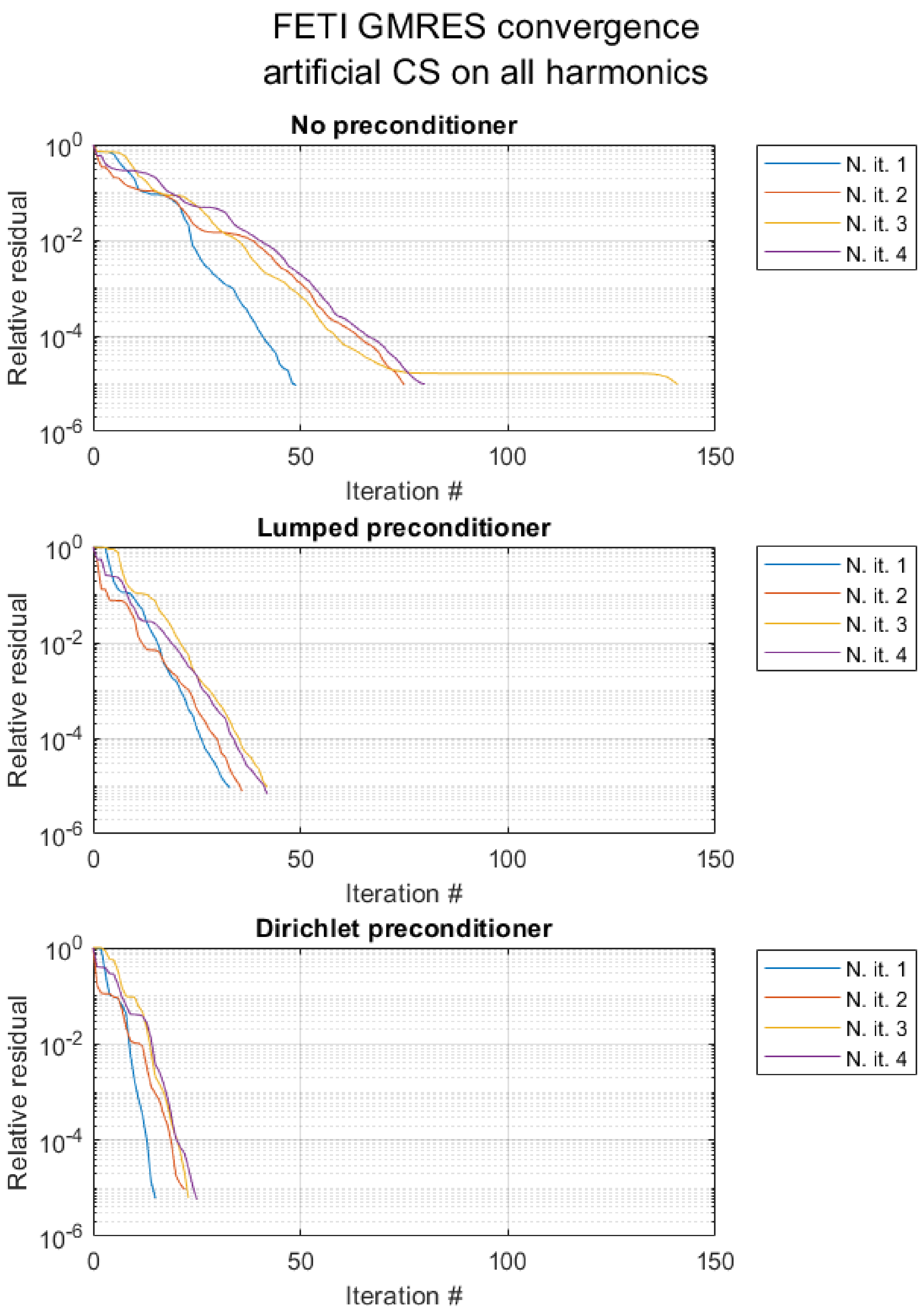


Figure 6.26: GMRES convergence in FETI when artificial coarse space on all harmonics is used, 32 domain testcase. Comparison for different preconditioner variants. Convergence for each Newton iteration for 1 frequency point solve is shown.

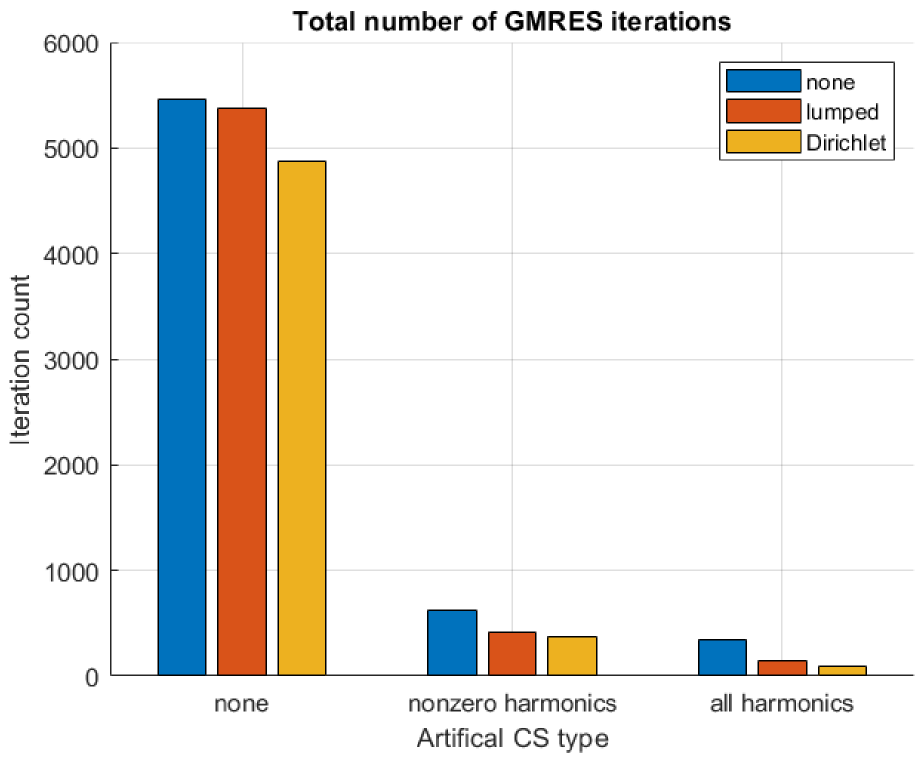


Figure 6.27: GMRES convergence total iteration count summary in FETI, 32 domain testcase.

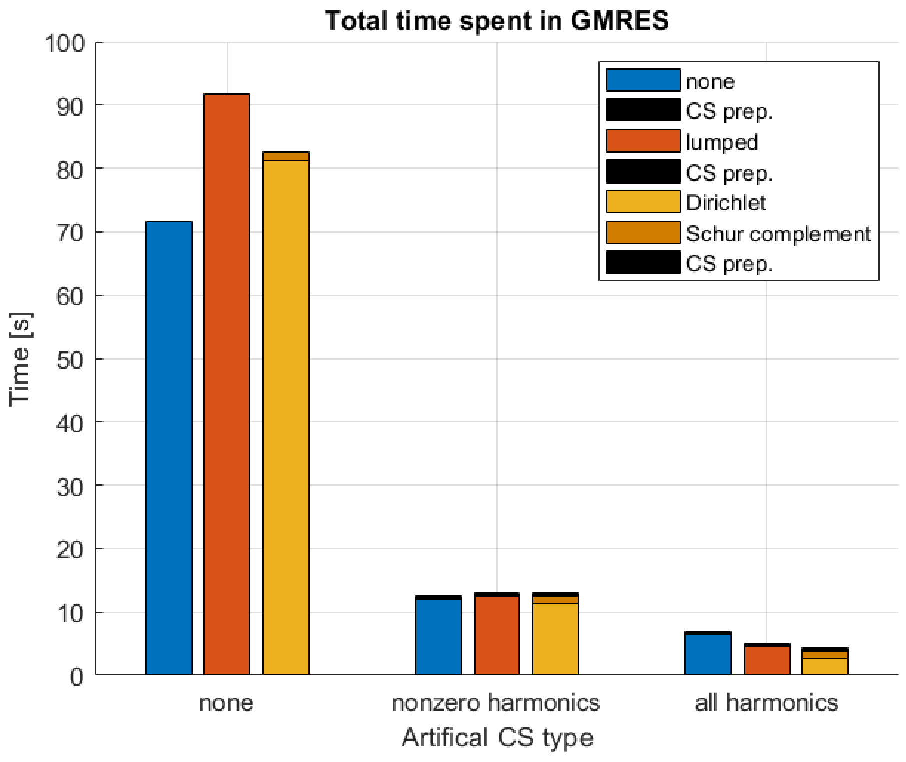


Figure 6.28: GMRES convergence total solver time summary in FETI, 32 domain testcase. Time required to prepare artificial coarse space and Schur complement also shown.

Next follow the same data for the larger testcase. In this one, the option of no artificial coarse space was dropped as it clearly shows to be significantly worse than the other two. Comparisons of GMRES convergence curves are in Figures 6.29 for nonzero harmonics artificial coarse space and 6.30 for artificial coarse space on all harmonics. Newton solver failed to converge for cases with artificial coarse space on nonzero harmonics with both lumped and Dirichlet preconditioner. In all other cases Newton converged. Summary of total GMRES iterations and total GMRES time are in Figures 6.31 and 6.32.

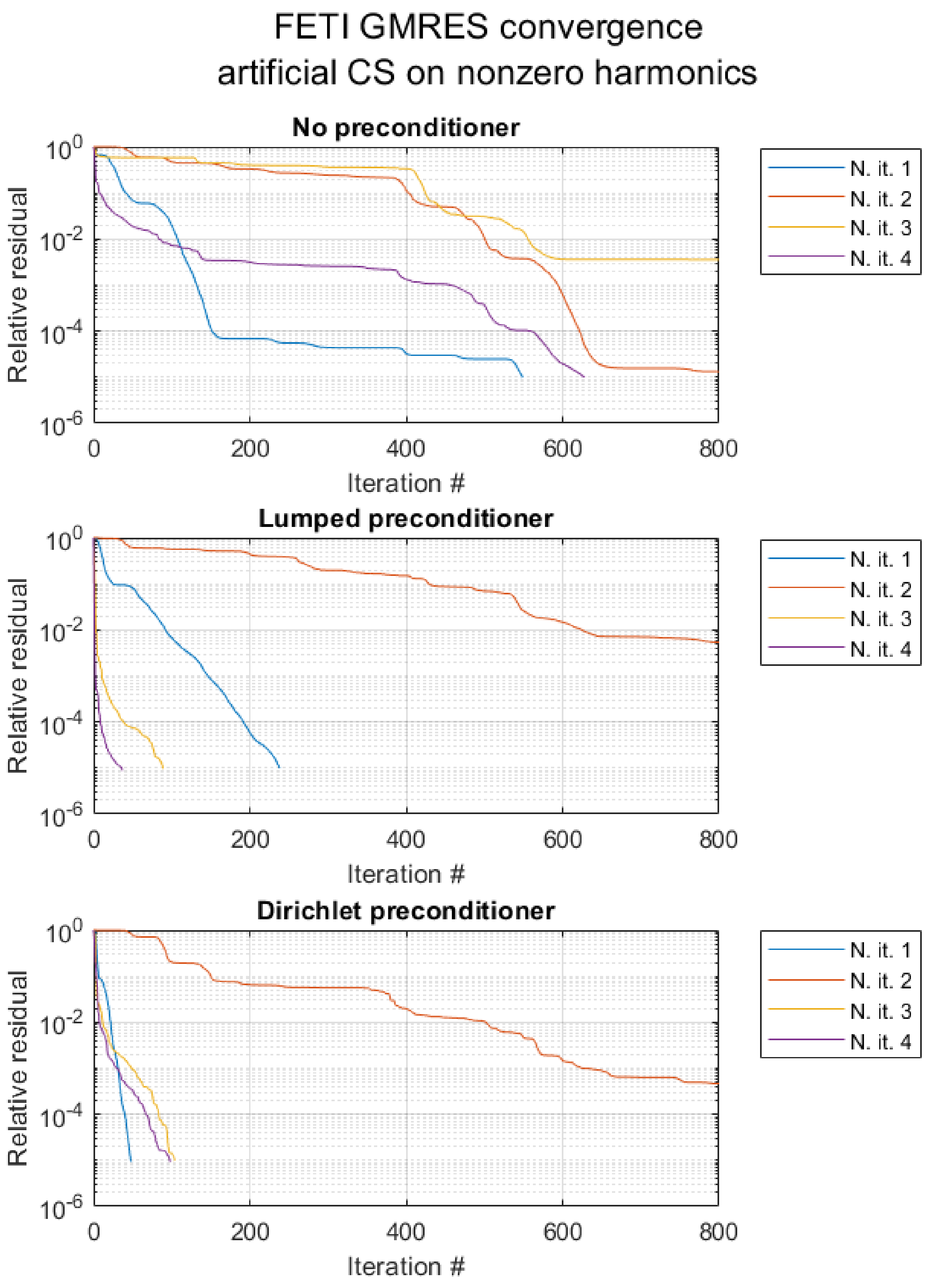


Figure 6.29: GMRES convergence in FETI when artificial coarse space on nonzero harmonics is used, 128 domain testcase. Comparison for different preconditioner variants. Convergence for each Newton iteration for 1 frequency point solve is shown. For Dirichlet preconditioner, GMRES performed $[48, 800, 104, 98]$ iterations. Newton solver failed to converge for lumped and Dirichlet preconditioner variants, but it converged for no preconditioner case.

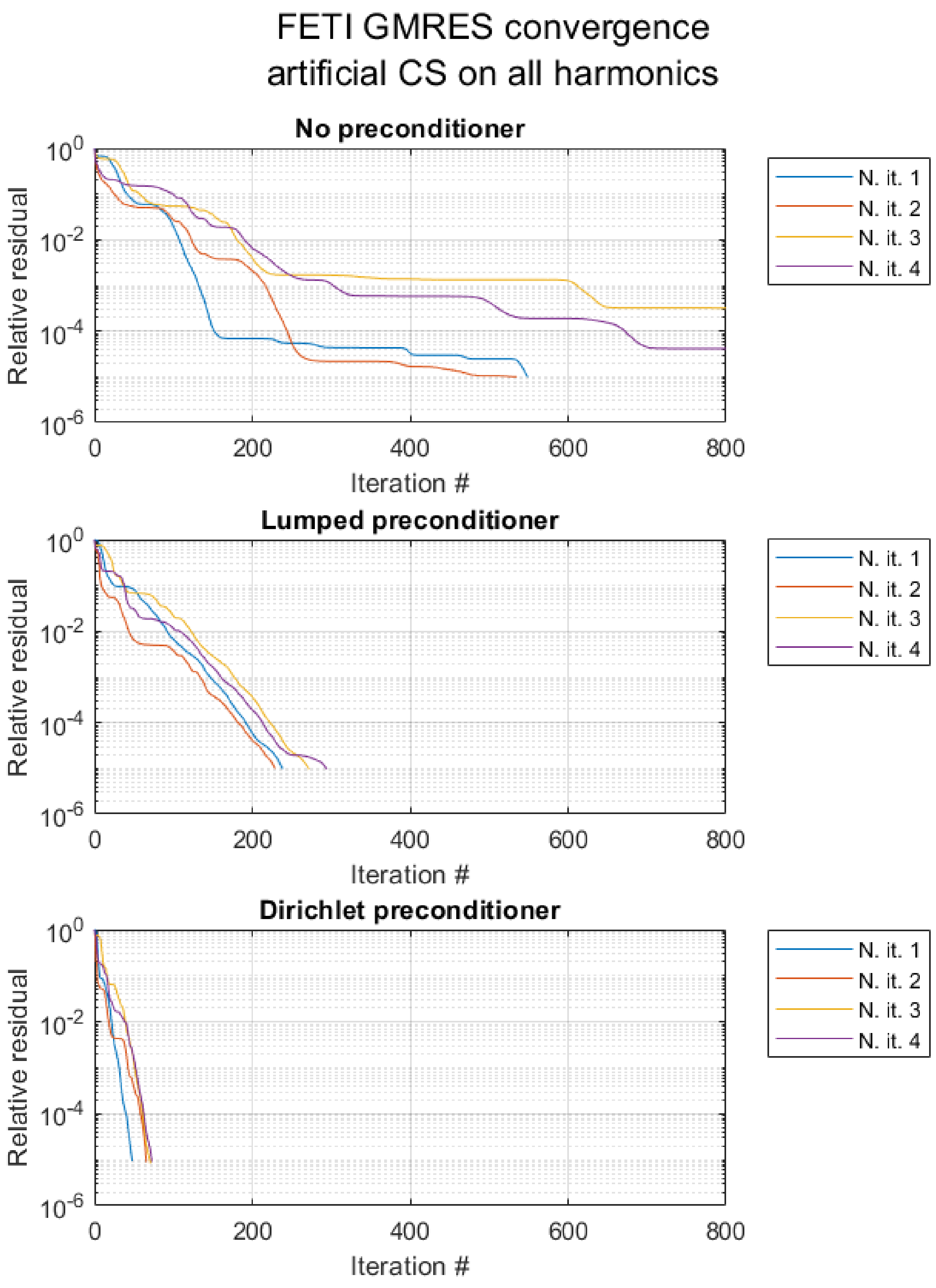


Figure 6.30: GMRES convergence in FETI when artificial coarse space on all harmonics is used, 128 domain testcase. Comparison for different preconditioner variants. Convergence for each Newton iteration for 1 frequency point solve is shown. For Dirichlet preconditioner, GMRES performed [48, 66, 71, 73] iterations.

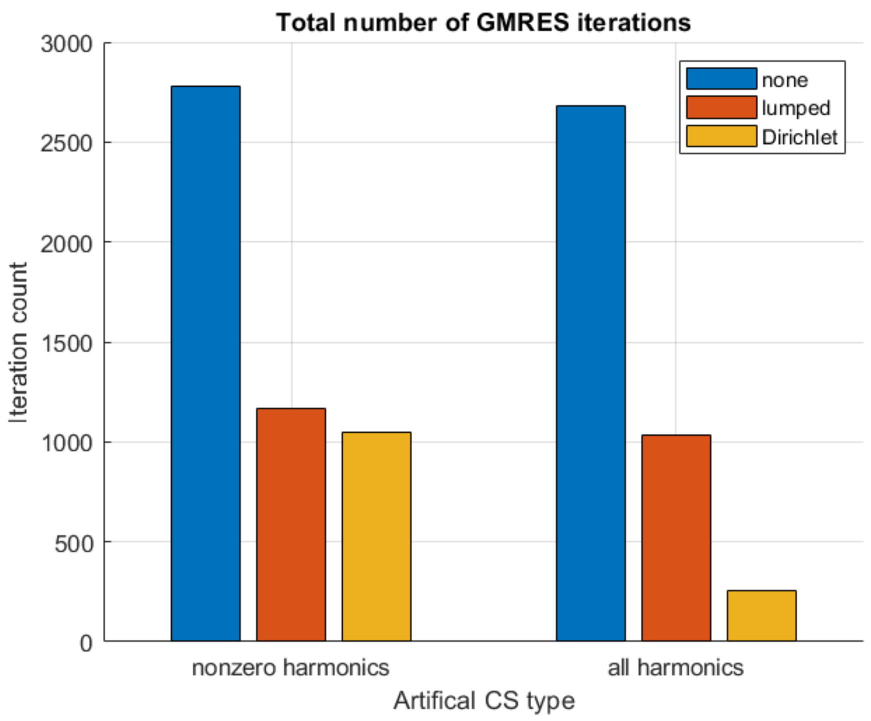


Figure 6.31: GMRES convergence total iteration count summary in FETI, 128 domain testcase.

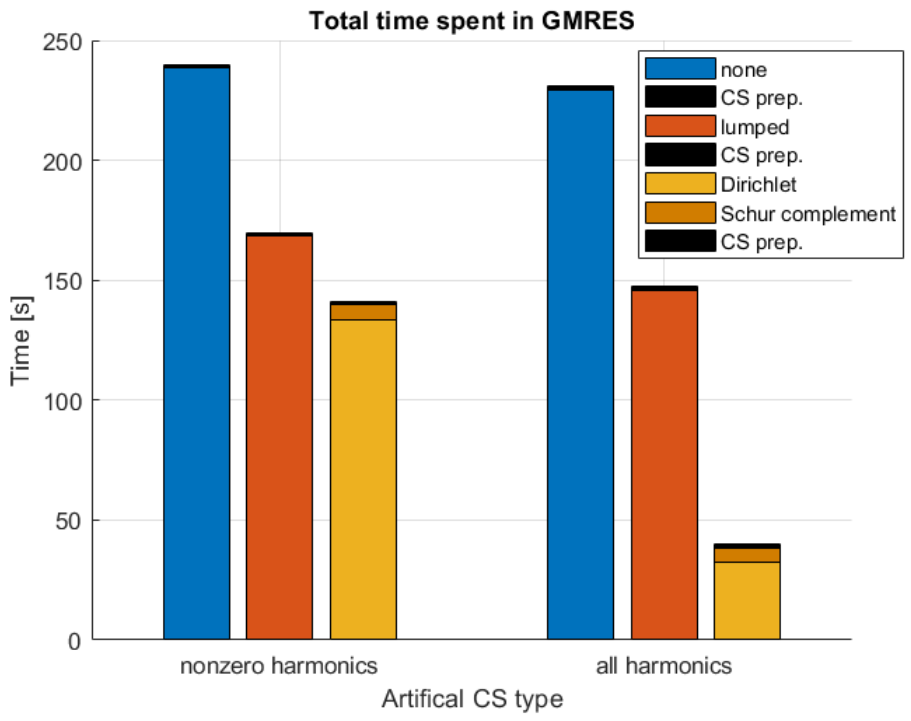


Figure 6.32: GMRES convergence total solver time summary in FETI, 128 domain testcase. Time required to prepare artificial coarse space and Schur complement also shown.

Discussion

First and the most apparent conclusion from this test is that the solver performs very poorly when no artificial coarse space is used (Figure 6.24). This variant was therefore removed from consideration for the larger testcase. This second larger testcase then shows a more pronounced comparison of the remaining options. In case of artificial coarse space on nonzero harmonics only (Figure 6.29), Newton solver did not converge for both preconditioners while it did converge without a preconditioner. This might be related to poor performance of the second Newton iteration in those cases. Convergence rate of Newton iterations 2-4 can be seen to be much worse than convergence of the 1st iteration already for the smaller mesh (Figure 6.25). This issue apparently became even more pronounced and the 2nd Newton iteration did not reach the required convergence, possibly ruining the Newton solution guess for the next iterations. Situation improved significantly when coarse space on all harmonics was used (Figure 6.30). Presence of the additional coarse space on the 0th harmonic reduces the number of iterations from 2nd Newton iteration further, bringing numbers of all iterations close together. The first Newton iteration is not affected as in that iteration the problem is linear since zero initial guess is used. Null space of the domain matrices therefore also includes the rotation vectors, so those are not used for the artificial coarse space.

The use of the interface rotations for the 0th harmonic artificial coarse space greatly improves GMRES convergence. However, it can still be seen that the number of GMRES iterations required generally grows slightly with each Newton iteration (Figure 6.30). The number of GMRES iterations is also significantly larger across all variants for the larger mesh testcase compared to the smaller one. It is possible that the artificial coarse space choice for this work is not optimal and a better null space vectors exists that would prevent GMRES iterations from growing with the problem size and Newton iterations. Note also that the design of the 0th harmonic artificial coarse space (interface rotations) as presented in 4.7, has its limitations. First, it might be difficult to identify individual domain interfaces on irregular meshes. Second, the rotations need to be zeroed out at corners with more than 2 domains to preserve the $\mathcal{G}_c^T \mathcal{G} = O$ orthogonality condition. This means that the rotations in certain decompositions won't cover the entire interface but only its interior, which could reduce effectiveness of such coarse space.

Finally, it seems that the Dirichlet preconditioner is the most effective in reducing the number of GMRES iterations. When coupled with the complete artificial coarse space it outperforms all other options (Figures 6.30, 6.32), even despite the extra cost of calculating Schur complement of domain matrices. Therefore, this combination was used in all further tests.

6.4.2 Scalability

A series of scalability tests for the FETI algorithm follows. By the time of computing these results the amount of computation resources available at IT4I was limited due to energy supply issues. All results in this section were obtained using 2 nodes on Karolina, meaning 256 cores. The setup of the testcase is mostly the same as the scaled clamped-clamped beam test in Table 6.7. The only difference is the number of elements per domain. Harmonics 0123 were used. Newton solve of nonlinear HBM problem for 1 frequency point (0.98) was performed. The Newton solver performed 4 iterations and 32 AFT points were used in all cases. Both weak and strong scalability were assessed, including timing of various parts of the FETI algorithm to identify potential bottlenecks. Results for the same testcases for the MUMPS solver were computed as well for reference, so the following tests also further illustrate the scalability of that solver.

Weak scalability

The first weak scalability testcase was designed to be as large as possible within the available RAM. The mesh was again decomposed purely along the Z axis. This type of decomposition minimises sizes of domain interfaces, which consequently minimises memory requirements on the coarse space projector matrices. The number of elements per domain was $2 \times 2 \times 54$. HEXA20 elements were used again. The list of MPIs and corresponding problem sizes are in Table 6.12.

MPI	Total elements	Total nodes	Total dofs (primal)
4	864	6,501	136,521
8	1,728	12,981	272,601
16	3,456	25,941	544,761
32	6,912	51,861	1,089,081
64	13,824	103,701	2,177,721
96	20,736	155,541	3,266,361
128	27,648	207,381	4,355,001
192	41,472	311,061	6,532,281
256	55,296	414,741	8,709,561

Table 6.12: First weak scalability test dimensions. Total dofs are counted in global sense, meaning dofs overlapping at interfaces are all counted as one.

The testcase was run with 3 different configurations for the GMRES tolerance, as this value influences the number of iterations the solver performs. First, the test was run with GMRES tolerance 10^{-5}. The time results for this test are in Figure 6.33. Several parts were timed separately: time spent in GMRES, time spent on processing of domain matrices (this includes their analysis and factorisation by MUMPS and computation of the Schur complement for the Dirichlet preconditioner), assembly of domain matrices (this includes the AFT procedure for evaluation of the nonlinear terms) and preparation of the $\mathcal{P}$ and $\mathcal{P}_c$ projectors. For reference, time results for the MUMPS solver for the same problem are added. MUMPS was only able to handle the problem up to 192 MPIs. For the 256 MPI case, the solver ran out of RAM. A more detailed analysis of the GMRES solve is then provided, showing individual times of $\mathcal{P}$, $\mathcal{P}_c$ and $\mathcal{F}$ operators application in each GMRES iteration.

Figure 6.34 shows the progress of GMRES convergence for each Newton iteration for the largest number of MPIs (256).

As the results show, the number of GMRES iterations grows significantly with number of domains, and so does the time spent in GMRES. Therefore, the same test was run with more GMRES tolerance configurations. One with the tolerance set to 10^{-3} and second with the same tolerance which is adaptively changed between Newton iterations. This adaptation takes the form of:

$$tol^i = tol^0 \times max\left(1, \frac{\|\tilde{F}\|}{\|H^i\|}\right) \tag{6.1}$$

with tol^i being the GMRES tolerance for the i-th Newton iteration and tol^0 being the baseline tolerance (in this case 10^{-3}). H^i represents the residual of the Newton solver in its i-th iteration. This approach loosens the tolerance when the Newton approaches the solution, making the relative residual of GMRES being based on the Newton residual rather than the residual of the internal linear problem of the particular iteration. These two tests start from MPI count 32.

Time results for the fixed 10^{-3} tolerance are in Figure 6.35. MUMPS results are added again for reference. GMRES convergence for the largest MPI count is in Figure 6.36. The same results for the adaptive 10^{-3} tolerance are then in Figures 6.37 and 6.38. It can be seen that GMRES stops at different tolerance for different Newton iterations.

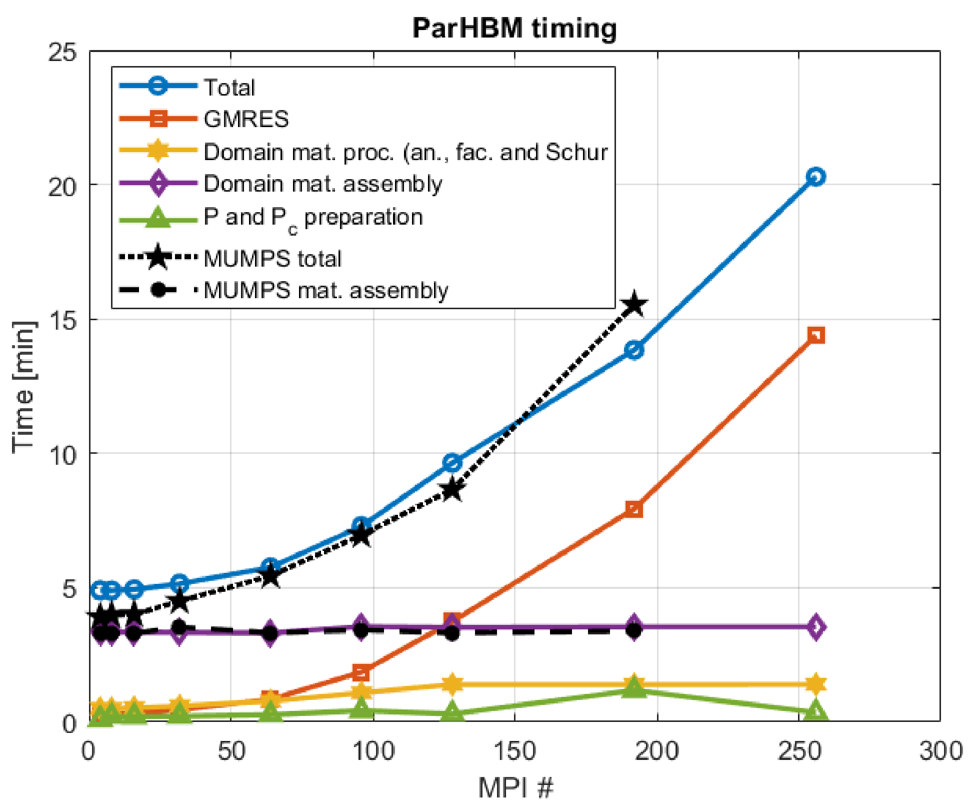
It should be noted that tolerance for the Newton solver was in all cases set to 5×10^{-6} and the solver converged within that tolerance in all cases in 4 iterations. The error of the domain connectivity equation $\sum_i B_i\tilde{u}_i$ was also measured in relative proportion to the magnitude of the solution vector:

$$\text{connectivity error} = \frac{\| \sum_i B_i\tilde{u}_i\|_\infty}{\|\tilde{u}_i\|_\infty} \tag{6.2}$$

with $\|\tilde{u}_i\|_\infty$ computed collectively across all domains:

$$\|\tilde{u}_i\|_\infty \text{ expands into } \max_i \|\tilde{u}_i\|_\infty \tag{6.3}$$

Comparison of total runtimes for all 3 tolerance variants and comparison of reached connectivity errors are in Figure 6.39.



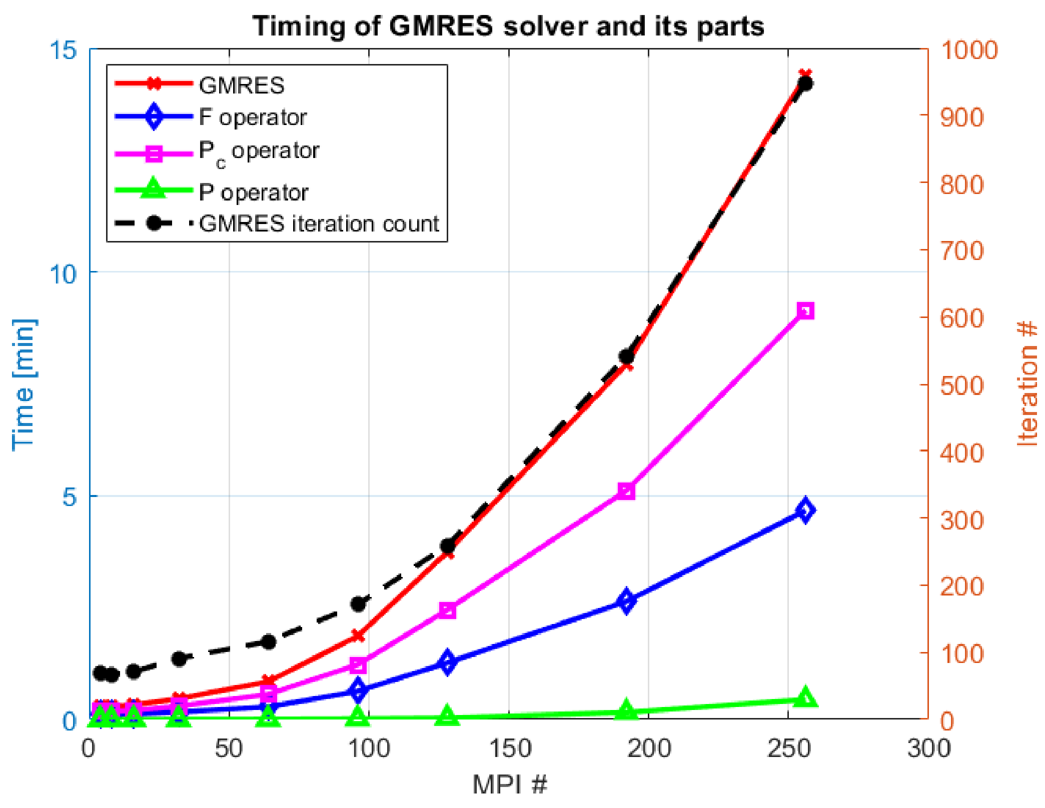


Figure 6.33: Total runtime of ParHBM for weak scalability test with clamped-clamped beam (harmonics 0123) and times of most significant parts of the code. Variant with 10^{-5} GMRES tolerance. MUMPS solver times for the same problem added for reference.

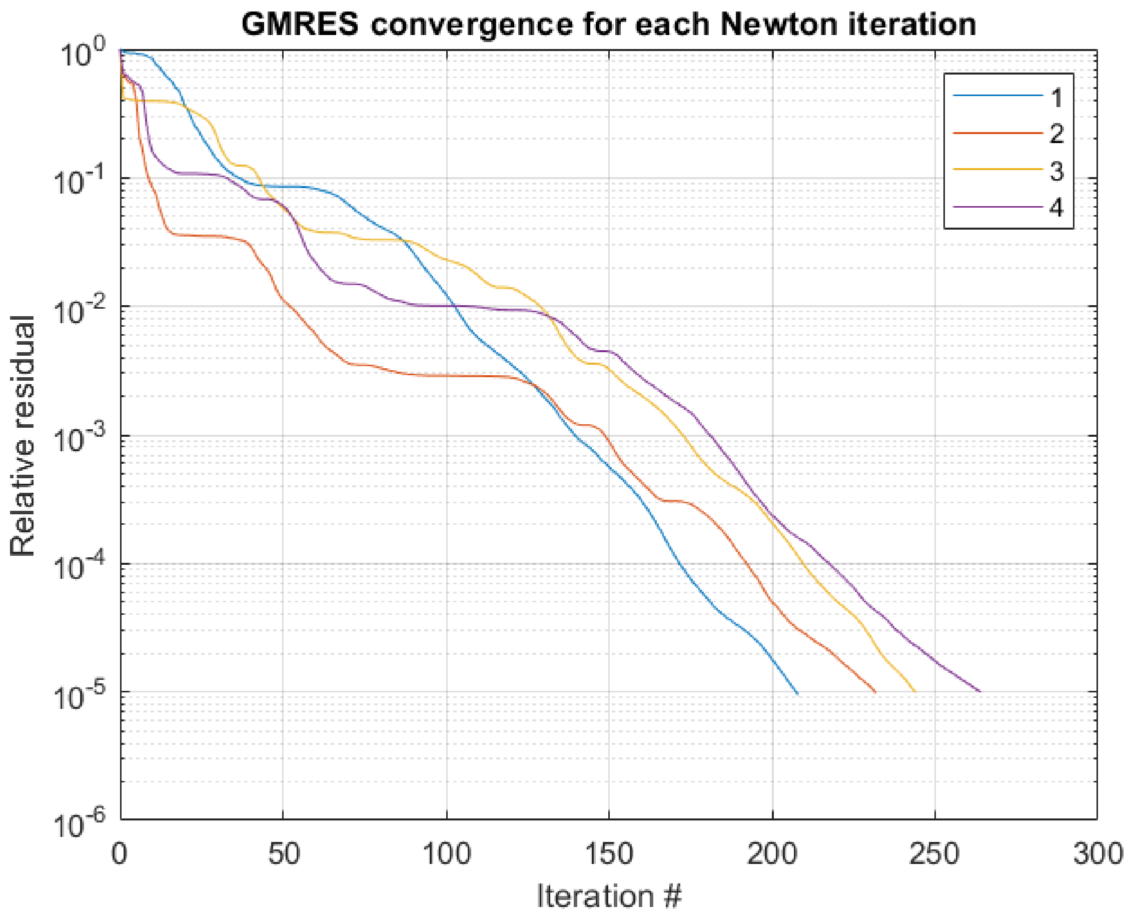
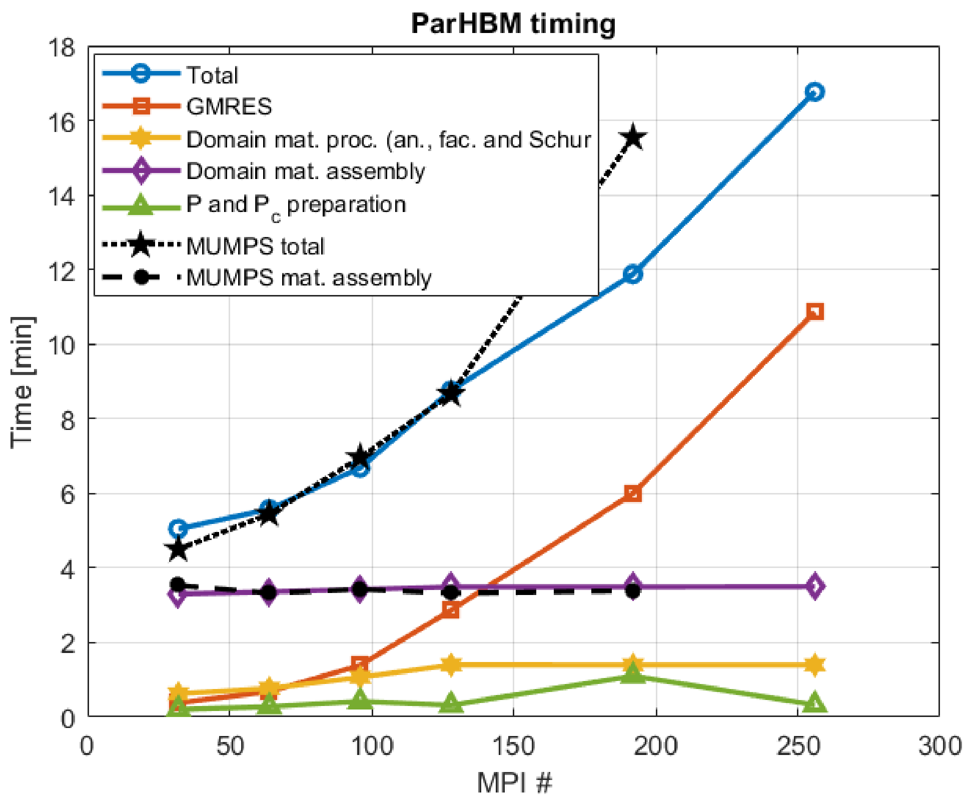


Figure 6.34: GMRES convergence for weak scalability test with clamped-clamped beam (harmonics 0123) for the largest case - 256 MPIs. Variant with 10^{-5} GMRES tolerance.



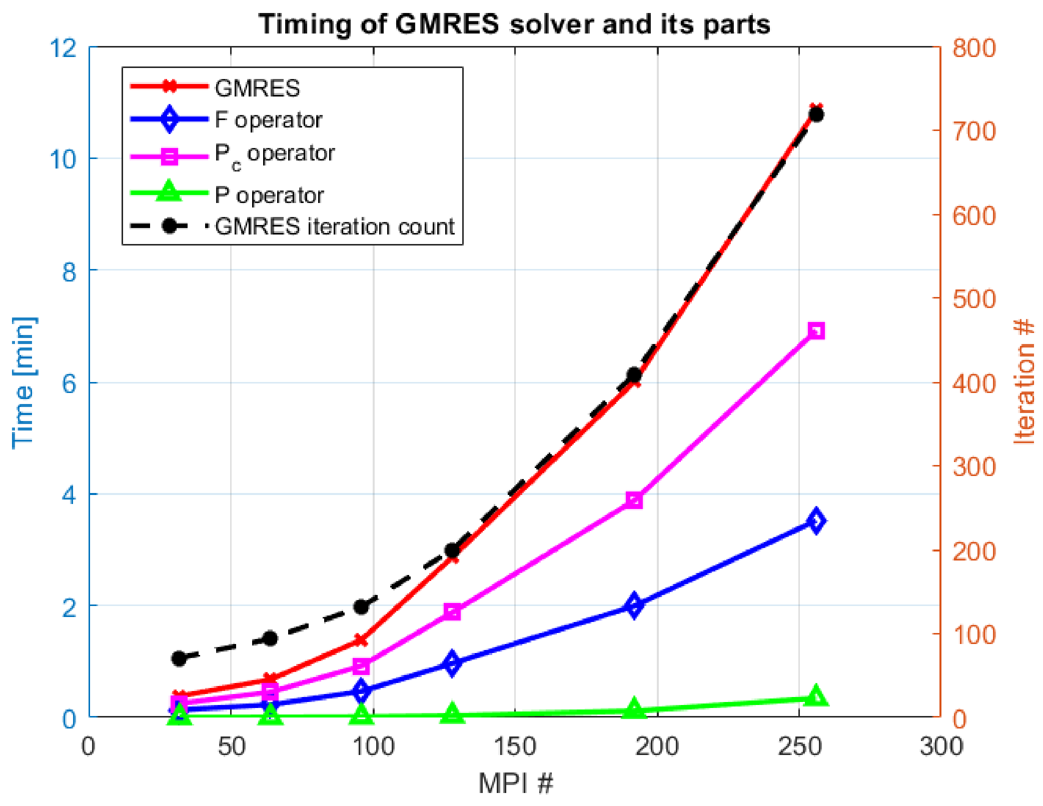


Figure 6.35: Total runtime of ParHBM for weak scalability test with clamped-clamped beam (harmonics 0123) and times of most significant parts of the code. Variant with 10^{-3} GMRES tolerance. MUMPS solver time for the same problem added for reference.

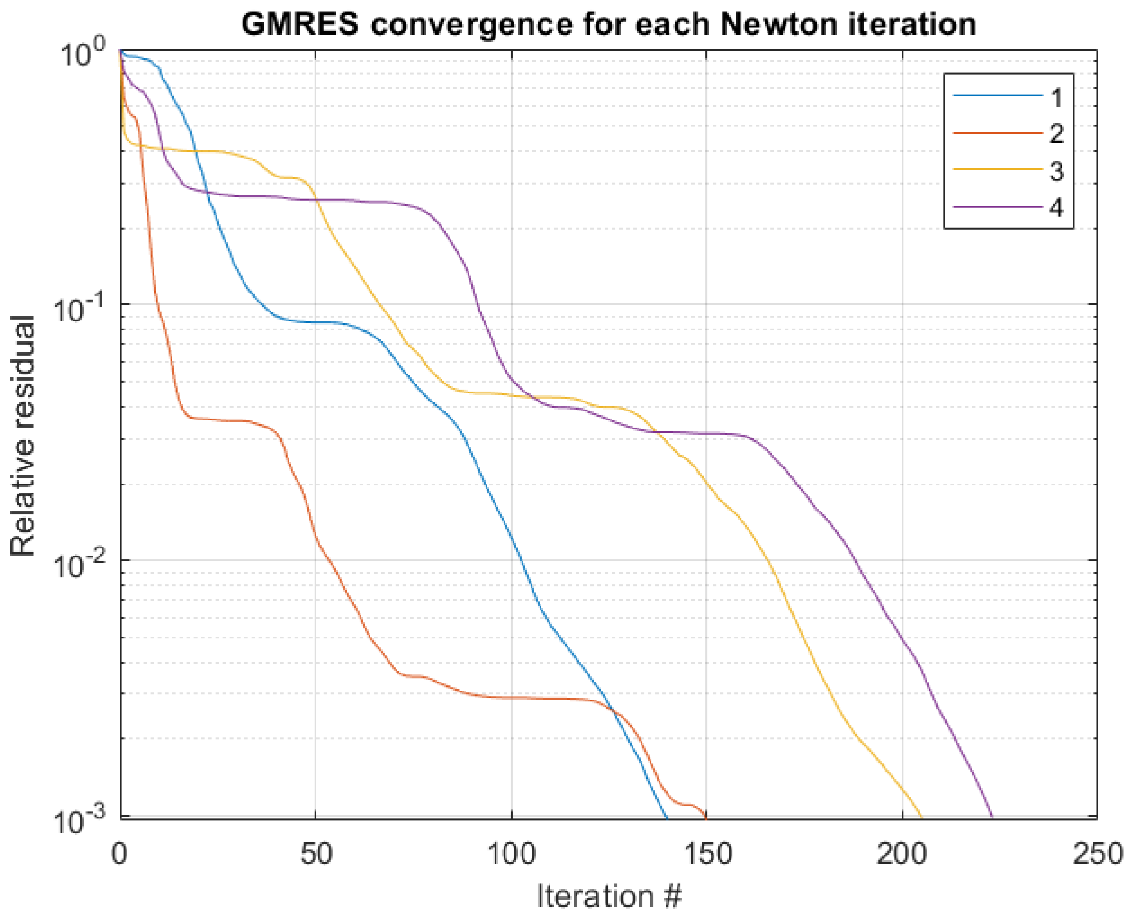
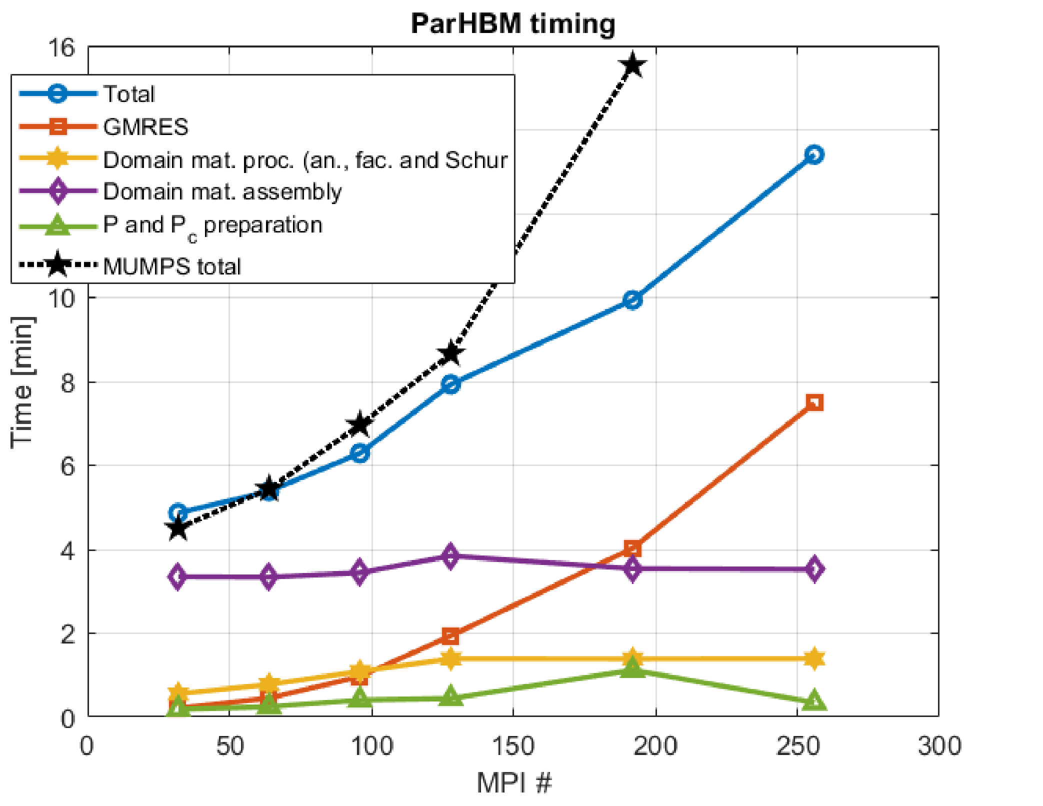


Figure 6.36: GMRES convergence for weak scalability test with clamped-clamped beam (harmonics 0123) for the largest case - 256 MPIs. Variant with 10^{-3} GMRES tolerance.



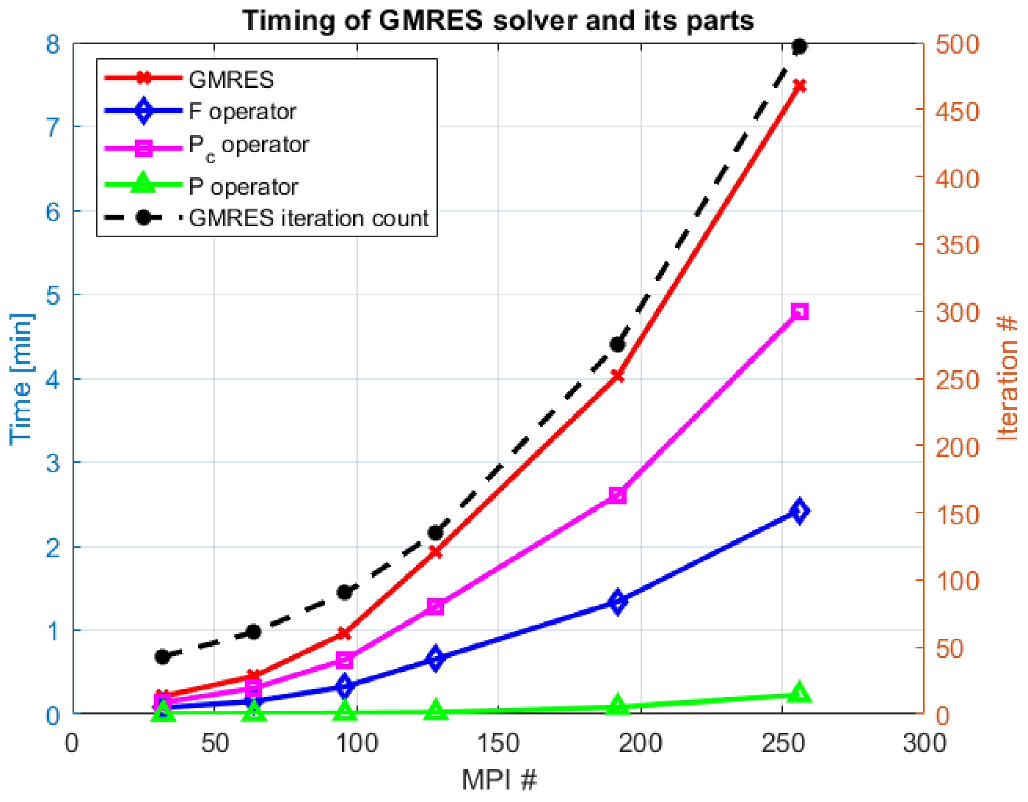


Figure 6.37: Total runtime of ParHBM for weak scalability test with clamped-clamped beam (harmonics 0123) and times of most significant parts of the code. Variant with 10^{-3} adaptive GMRES tolerance. MUMPS solver time for the same problem added for reference.

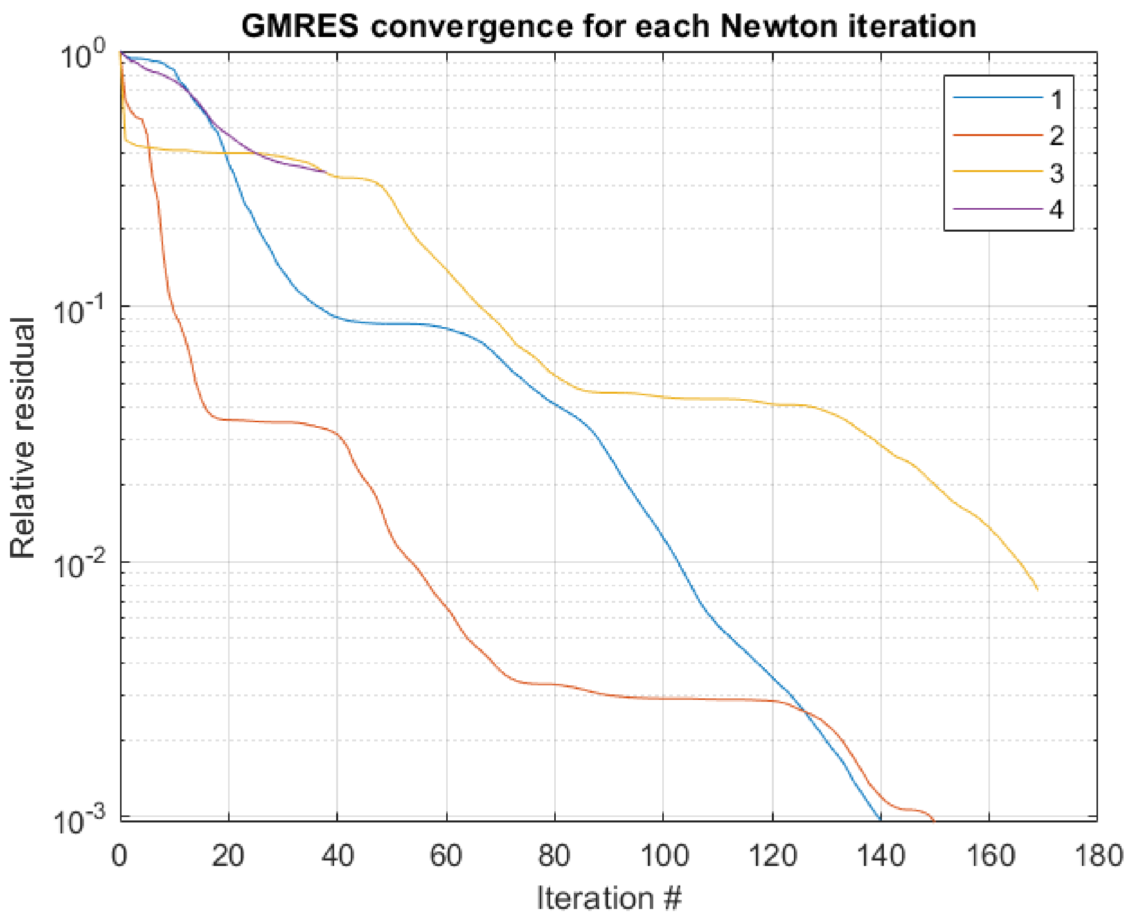
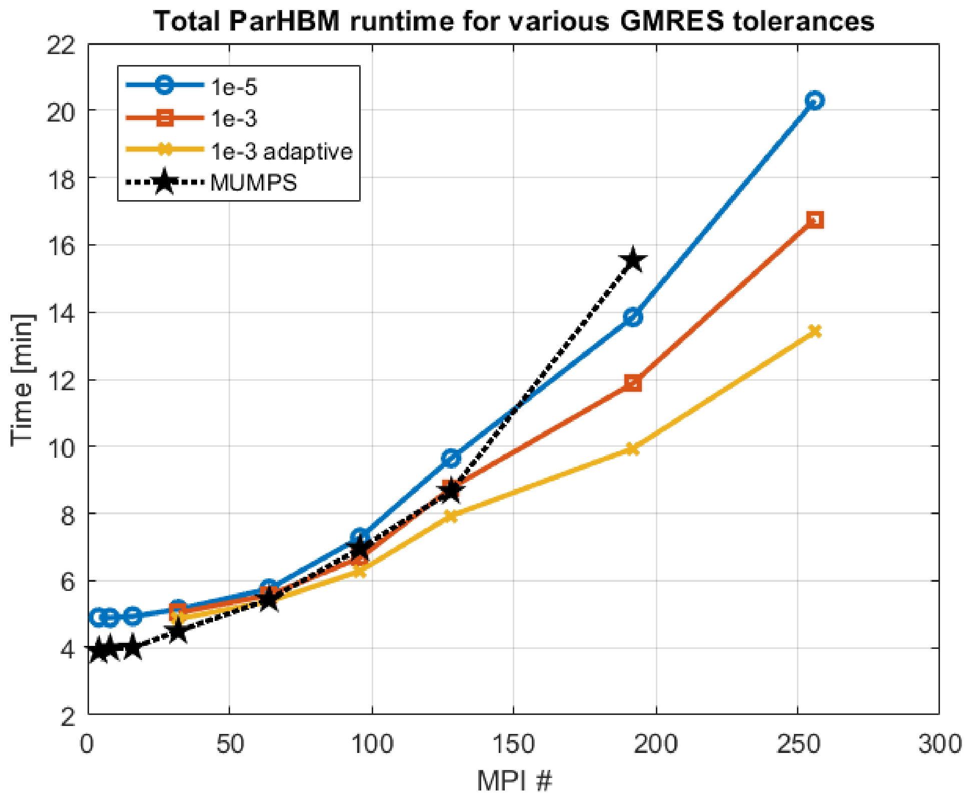


Figure 6.38: GMRES convergence for weak scalability test with clamped-clamped beam (harmonics 0123) for the largest case - 256 MPIs. Variant with 10^{-3} GMRES adaptive tolerance.



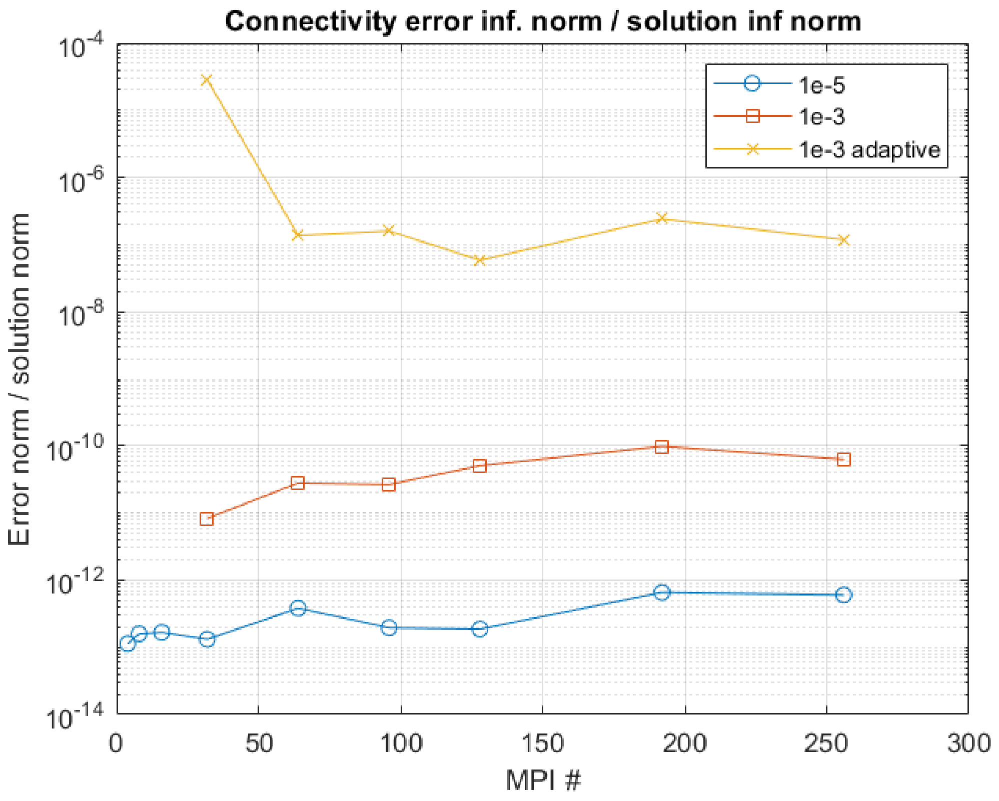


Figure 6.39: Comparison of total ParHBM runtimes (top) and domain connectivity error norm ratio to solution norm (bottom) for various GMRES tolerances. The 'infinite' norm $\|\|_\infty$ was used, meaning taking the largest absolute value of the vector.

The second weak scalability testcase used a much smaller domain size - only 36 elements. The list of MPIs and sizes is in Table 6.13.

MPI	Total elements	Total nodes	Total dofs (primal)
16	576	3,045	63,945
32	1,152	5,957	125,097
64	2,304	11,781	247,401
96	3,456	17,605	369,705
128	4,608	23,429	492,009
192	6,912	35,077	736,617
256	9,216	46,725	981,225

Table 6.13: Second weak scalability test dimensions. Total dofs are counted in global sense, meaning dofs overlapping at interfaces are all counted as one.

This testcase was tested with 2 different domain decomposition strategies. The first is decomposition into $2 \times 2 \times \frac{n}{4}$ domains, where n is the number of used MPIs. The second strategy is the standard pure Z axis decomposition. Example of both decompositions for case of 64 MPIs is in Figure 6.40. The number of elements per domain was adjusted accordingly to keep the overall mesh size the same in both strategies. First strategy ('type 1') used $3 \times 3 \times 4$. elements per domain and the second ('type 2') used $6 \times 6 \times 1$ elements.

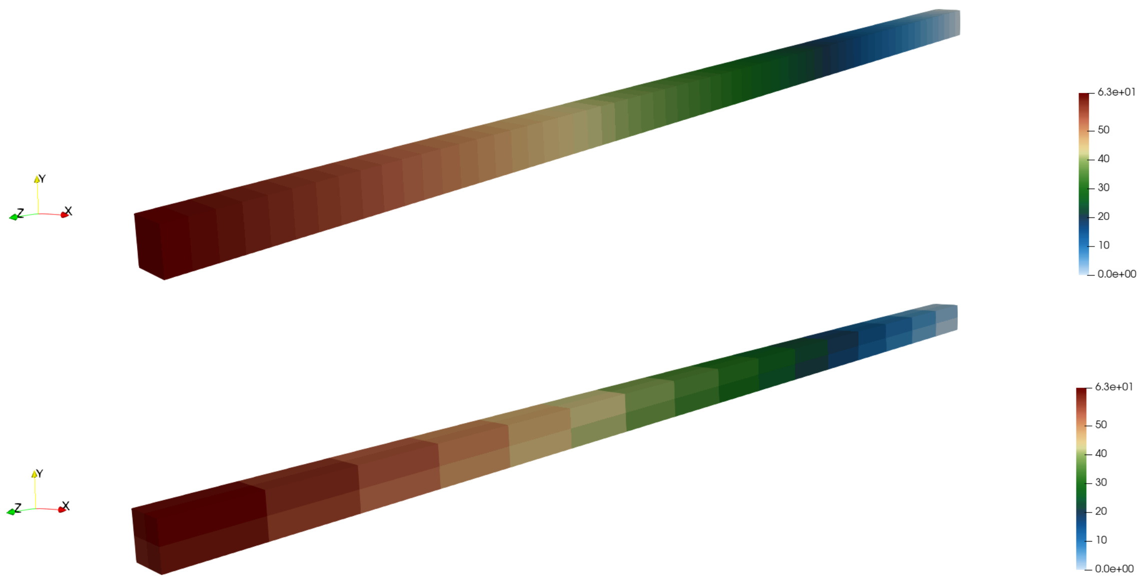


Figure 6.40: Comparison of 2 domain decompositions for case of 64 MPIs. Top - Z axis slicing (type 2), bottom - 2×2 domains along the XY axes, the rest along Z (type 1).

The timing of the code for both decomposition types, as well as MUMPS for the same configurations, can be seen in Figure 6.41. For the first type, the complete MPI range was computed with MUMPS. For the second decomposition type, only the first and last points were computed.

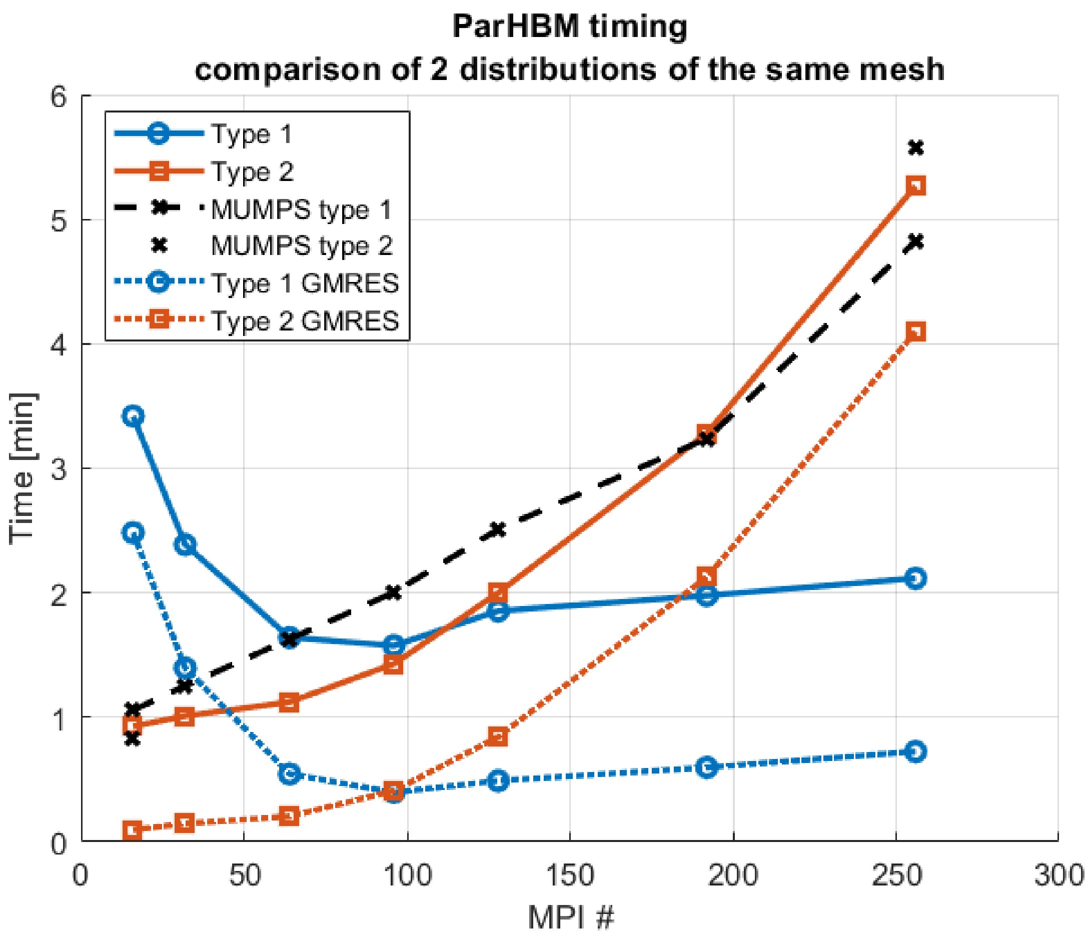


Figure 6.41: FETI weak scalability results. Comparison of total runtimes for 2 different domain decompositions (type 1 and type 2) across a range of MPIs. MUMPS results for both types shown in black. For type 2, only first and last points were computed.

Strong scalability

A single strong scalability test was computed and is presented here. The parameters are the same as in the weak scalability cases except for the mesh size and decomposition. The total size of the mesh was $8 \times 8 \times 256$ HEXA20 elements. This implies 78,561 nodes and 1,649,781 dofs. The list of used MPIs and corresponding decompositions is in Table 6.14. GMRES tolerance was set to 10^{-3}. A nonlinear problem using harmonics 0123 was again solved, taking 4 Newton iterations for all MPI counts.

MPI	domains $\times$ elements count
16	$(1 \times 1 \times 16) \times (8 \times 8 \times 16)$
32	$(1 \times 1 \times 32) \times (8 \times 8 \times 8)$
64	$(1 \times 1 \times 64) \times (8 \times 8 \times 4)$
128	$(2 \times 2 \times 32) \times (4 \times 4 \times 8)$
256	$(2 \times 2 \times 64) \times (4 \times 4 \times 4)$

Table 6.14: Strong scalability test dimensions.

The change in decomposition from purely Z axis based to a decomposition along all 3 axes between 64 and 128 MPIs was based on the observation in the previous test where the runtime was much lower for the latter for higher MPI count. The decomposition for the MUMPS solver was the same for each MPI count.

Total ParHBM runtime was measured for both solvers as well as several individual parts of the algorithms. For the FETI solver, the time spent in GMRES, domain matrix assembly and projectors $\mathcal{P}$ and $\mathcal{P}_c$ preparation was measured. For MUMPS, the global matrix assembly and its factorisation are shown. The time data is presented in Figure 6.42. Figure 6.43 then interprets the same data in terms of a speed-up, highlighting the ideal speed-up based on the increase of MPI count relative to the lowest count of 16. Figure 6.44 shows number of required GMRES iterations for each MPI count and each Newton iteration. Lastly, Figure 6.45 shows domain connectivity errors related to the FETI primal solution in terms of the $\|\|_\infty$ norm.

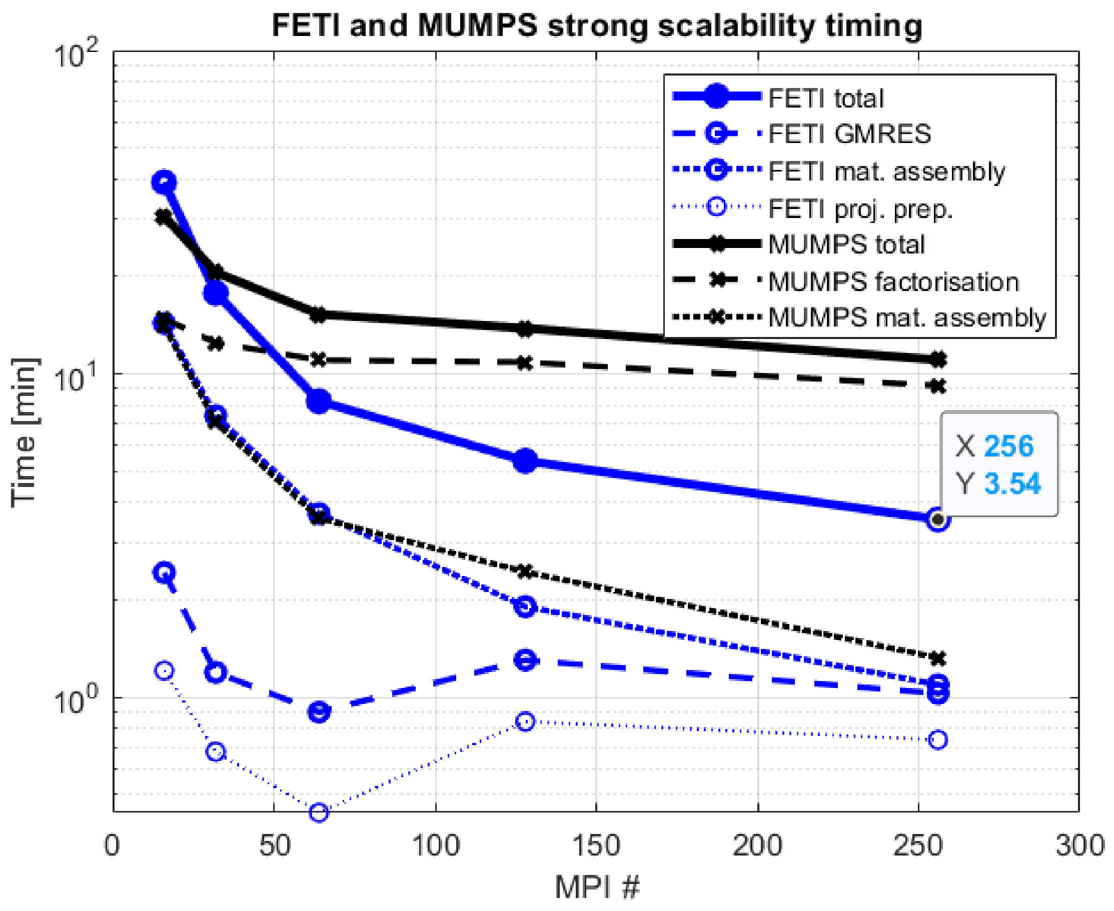


Figure 6.42: FETI strong scalability results. Total runtimes (thick lines) of ParHBM for both the FETI and MUMPS solvers. Several key algorithm components highlighted by dashed lines.

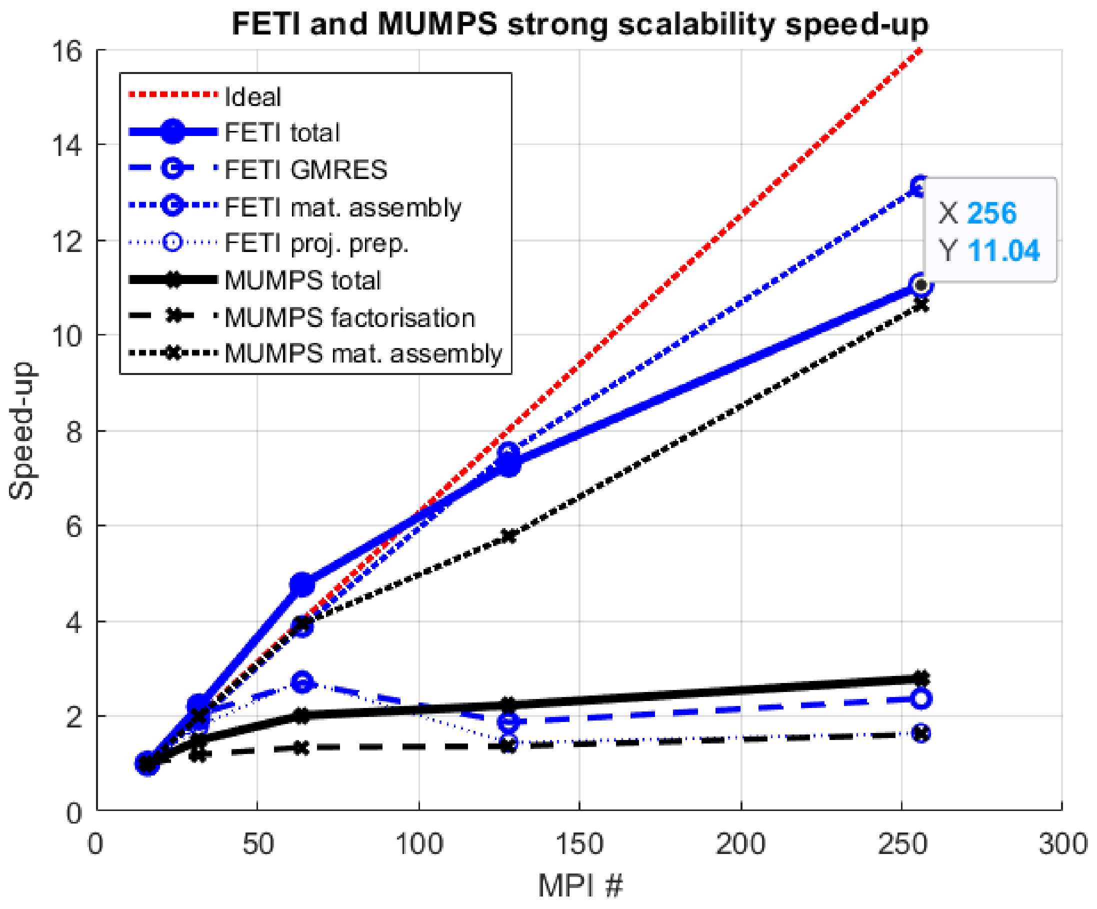


Figure 6.43: FETI strong scalability results. Achieved speed-up (thick lines) of ParHBM for both the FETI and MUMPS solvers, compared to the ideal speed-up (red line). Several key algorithm components highlighted by dashed lines.

Discussion

The total runtime of a code in case of weak scalability is expected to remain constant along the increasing number of MPIs. This is very clearly not the case of either of the solvers in the first presented test. Both MUMPS and FETI show significant increase in time with increasing number of MPIs (Figures 6.33, 6.35, 6.37). In case of the FETI solver it's the time spent in GMRES that plays the biggest part in this increase in total runtime. It can be seen that this increase in GMRES time correlates strongly with the number of GMRES iterations required to reach the required tolerance. While other parts of the FETI algorithm also don't show perfect scalability, the growing number of GMRES iterations is the key component damaging the overall solver performance.

The comparison of various GMRES tolerances provides several interesting insights. First, decreasing this tolerance can significantly reduce the overall number of GMRES iterations required (Figures 6.33, 6.35, 6.37). Even when the GMRES tolerance is as high as 10^{-3}, the outside Newton solver still successfully converges within a tolerance of 5×10^{-6}. It should be noted that only part of the dual problem solution is solved for with the GMRES solve. A part of the solution is obtained by solving directly on the natural and artificial coarse spaces.

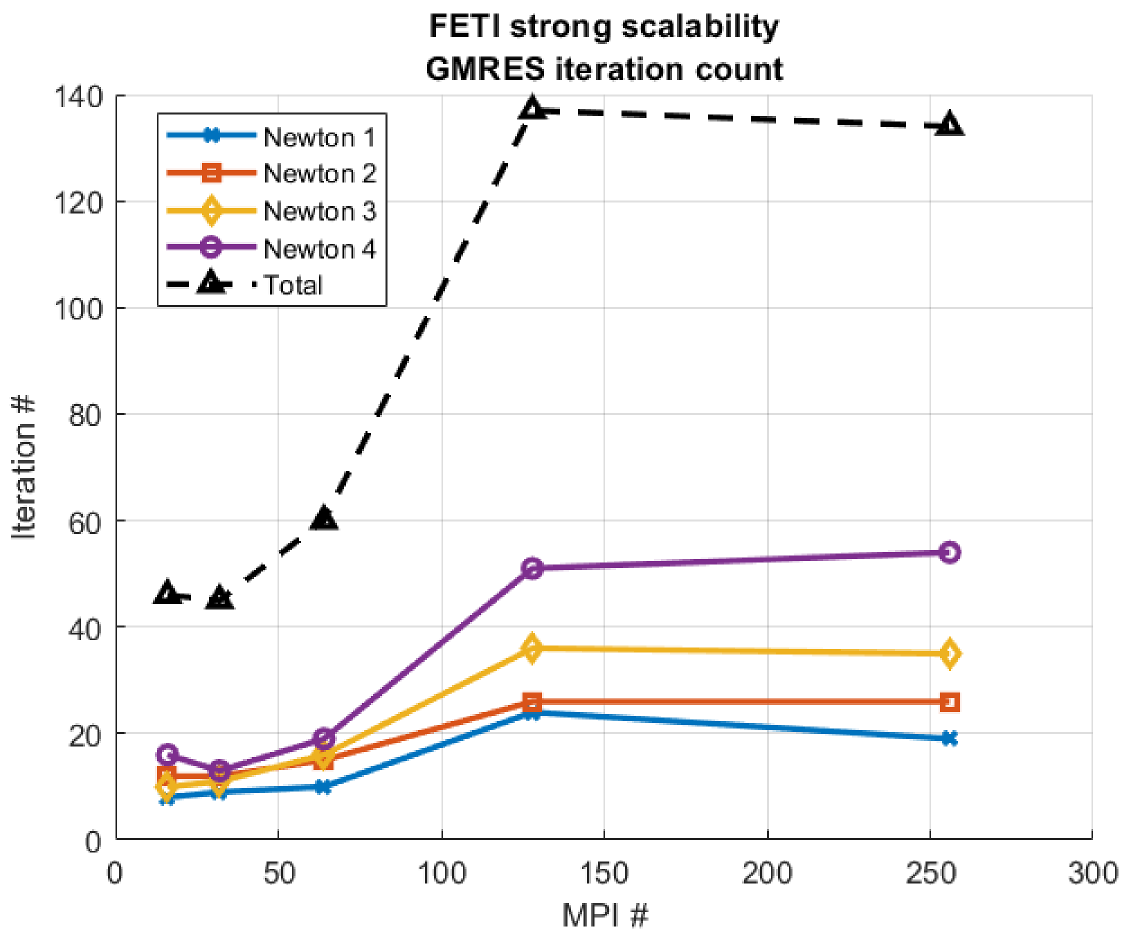


Figure 6.44: FETI strong scalability results. Number of GMRES iterations for each Newton iteration and their total count over the entire Newton solve.

The loosening of the GMRES tolerance however directly correlates with deterioration of the domain continuity of primal domain solutions (Figure 6.39). It is therefore a trade off between the two. Importantly, loosening the GMRES tolerance doesn't significantly improve the scalability of the solver, only the absolute value of the runtime for each MPI count.

When comparing GMRES convergence curves in Figures 6.34 and 6.36 two things can be concluded. First, in both cases, the number of GMRES iterations required increases with each Newton iteration. Second, while the looser tolerance 10^{-3} reduces the number of iterations, it also deteriorates the individual convergence curves. Comparing 3rd and 4th Newton iterations for both cases, it can be seen that in the 10^{-5} case, the solver reached tolerance of 10^{-3} in well under 200 iterations while it took more than 200 iterations to reach the same tolerance in the 10^{-3} case where it was the terminating tolerance. Both of these observations lead to the idea of the adaptive tolerance that was described above. While this approach further reduces the number of required GMRES iterations, it also worsens the domain continuity error even more. It can be concluded that the choice of the GMRES tolerance is clearly an impactful factor and more research in that area should be conducted in the future.

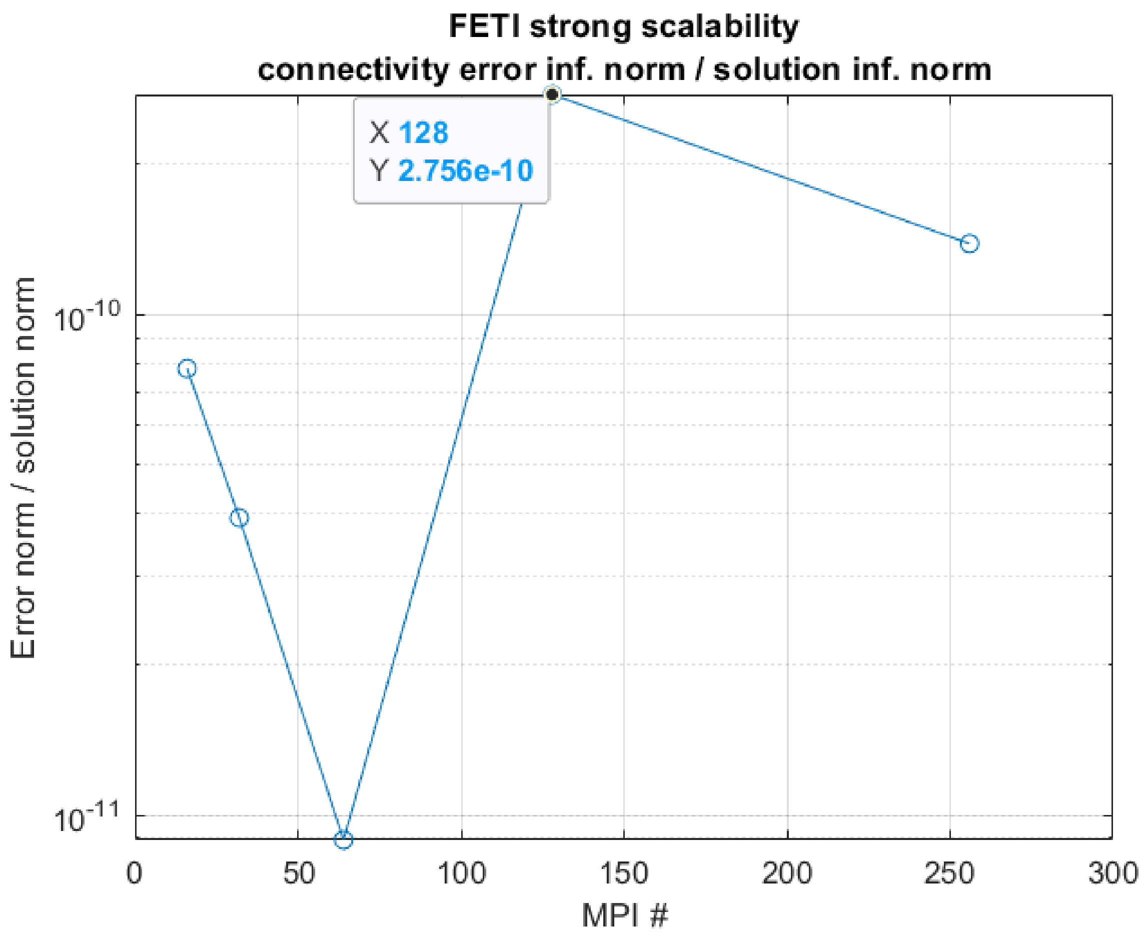


Figure 6.45: FETI strong scalability results. Domain connectivity divided by the solution norm in infinite norms.

The second weak scalability testcase was designed to test the FETI solver with smaller domains. Scalability of this testcase (Figure 6.41) is much better when considering the better parts of the curves for each decomposition. It can be concluded that the decomposition strongly affects the scalability of the solver. This agrees with conclusions in [65] which analysed influence of domain aspect ration on FETI convergence. It can be also observed that the MUMPS solver seems to not be affected in any significant way by the decomposition type. As in the first testcase, the dominant factor influencing the scalability of FETI is time spent in GMRES.

The last testcase shows the strong scalability speed-up efficiency of the FETI solver to be around 69% for the largest number of domains (Figure 6.43). The biggest slowing factor seems to be again GMRES. The change of decomposition between 64 and 128 MPIs appears to impact the total timing, however, as was shown previously in the weak scalability tests, keeping the decomposition purely along the Z axis could lead to even worse results. While the number of GMRES iterations increased from 64 to 128 MPIs, it remained approximately the same when going from 128 to 256 MPIs (Figure 6.44). A larger testcase is required to better assess the scalability of the solver in the future.

All the tests are appended with results computed with the MUMPS solver as well. It can be seen that the FETI solver generally matches and in some cases even outperforms MUMPS. MUMPS seems to be more efficient with smaller number of domains. Furthermore, in the first weak scalability testcase, MUMPS was not able to solve the largest MPI problem due to the lack of memory. As it was discussed in the MUMPS scalability testcase for the global system, the solver is quite memory intensive. However, this is by no means a conclusive evaluation of the MUMPS library. As mentioned previously, MUMPS was used with only MPI and not OpenMP, and the solver might perform better with different domain decomposition and domain sizes.

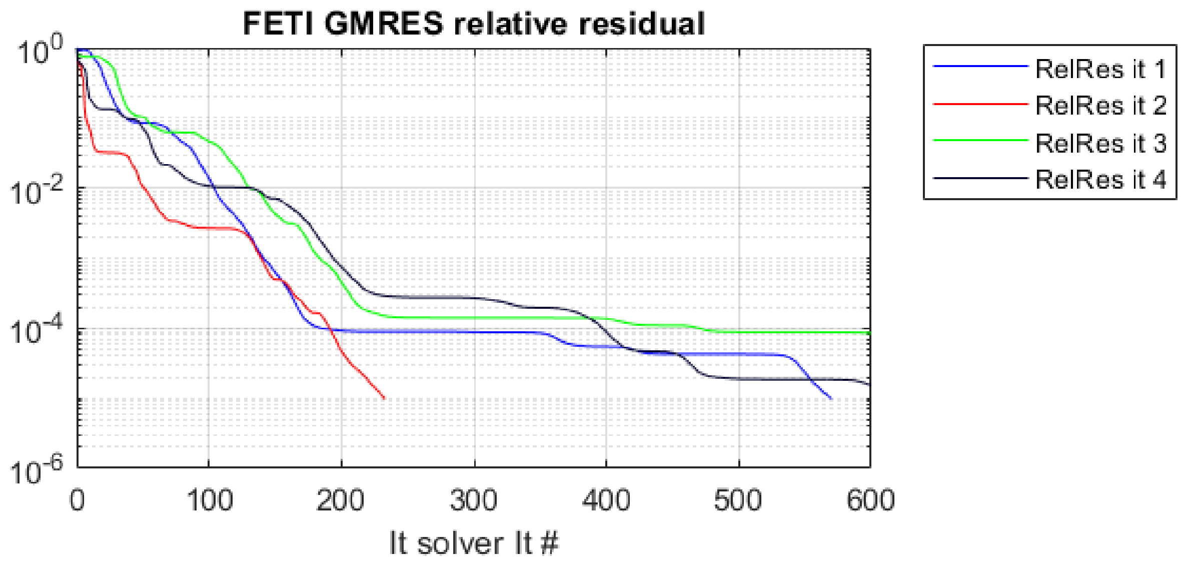
6.4.3 Projected dual problem equations

Section 4.6 introduced two possible forms of the projected dual problem - (4.65) and (4.66). A comparison of these two variants is shown in this section. The 256 MPI case from the largest weak scalability testcase was used for this (see Table 6.12). GMRES tolerance was set to 10^{-5}. Number of AFT points was again 32. Newton solver converged in both cases in 4 iterations within relative tolerance of 5×10^{-6}. Comparison of GMRES convergence for all Newton iterations can be seen in Figure 6.46. The domain continuity error was of same magnitude in both cases.

Discussion

As seen from this result, both versions of the dual problem equations converge to correct solution with good enough connectivity error. Both can therefore be used in principle. The advantage of (4.65) is that it requires only one application of $\mathcal{P}$ and $\mathcal{P}_\rfloor$ while (4.65) applies each twice. However, this testcase demonstrates that the version without the second set of projections suffers from GMRES stagnation sooner than the second version. The second version can also experience such stagnations, but they generally happen later (on lower residual magnitudes). This is why it was used for all results computed in this work.

It should be noted that the application of the preconditioner and projectors are usually integrated directly inside the Krylov space solver. However, for this work, the operator approach was used as it allows for use of a standard GMRES library without any needs for modifications. The form of the projected dual problem equations was discussed with professor Rixen [152], whose work on FETI is referenced several times in this work [153], [72], [61]. However, this discussion hasn't reached its definitive conclusion as of time of this work and it might be that a better version of the dual problem equations exist.



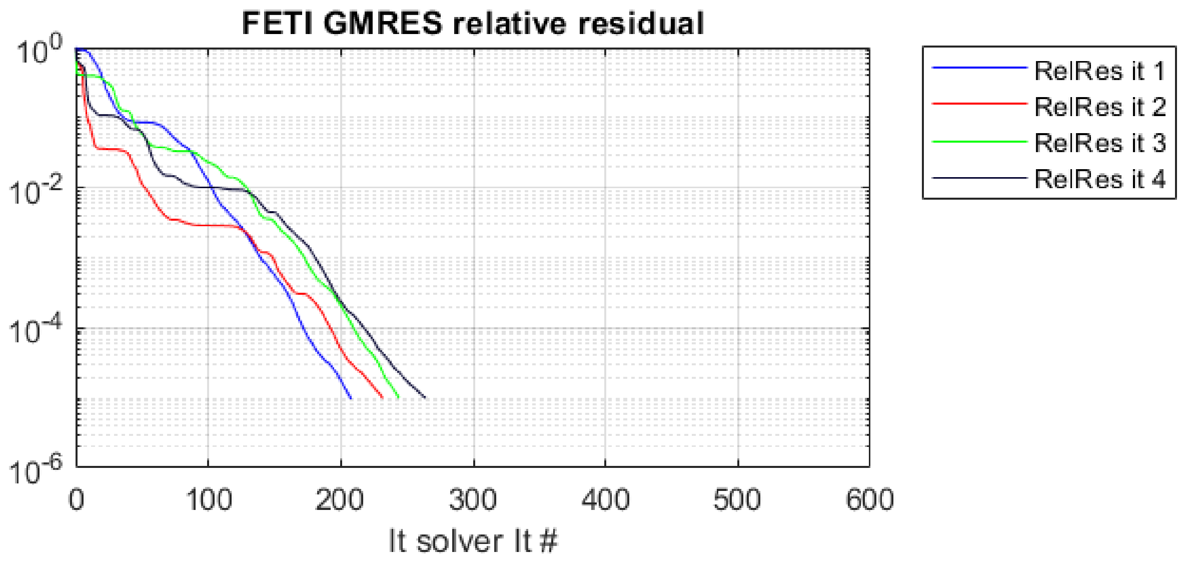


Figure 6.46: Comparison of GMRES convergence when solving (4.65) (top) and (4.66) (bottom).

6.4.4 Frequency Response Curve

The previous sections on FETI results all worked with one fixed frequency point. The capability of the solver to compute a full FRC of a structure is analysed here. It is not a performance analysis as a relatively small testcase was used. The focus is placed on the shape of the computed FRC and number of GMRES iterations required along the curve.

The testcase setup is again the same as in Table 6.7. Two different sets of harmonics are compared - 0123 and 012345. Additionally, smaller damping values were used - the reference damping ($3 \times 10^{-3} \times M$) and half and quarter of that value. The total mesh size was $2 \times 2 \times 64$ elements, decomposed into 32 domains along the Z axis. Secant predictor was used. Different GMRES tolerances had to be set for different cases. For harmonics 0123 case with full and half damping 10^{-4} was enough. However, for the quarter damping case the tolerance had to be reduced to 10^{-5} otherwise the solver wouldn't converge near the turning point. This can be seen in Figure 6.47. For harmonics 012345 tolerance of 10^{-6} was set for all 3 damping values, as even 10^{-5} was not enough at some turning points. Domain connectivity error is not shown for this test as the testcase is quite small and it was consistently good in all cases. Numerical parameters common for all tests in this section can be seen in Table 6.15.

Computed FRCs for all combinations of harmonics and damping are in Figure 6.48. The response was calculated in the same node where the excitation force is applied, and in the same direction. It can be seen that responses for full and half damping are identical for both harmonic sets. A difference can only be seen for the quarter damping. The FRC for harmonics 012345 was computed in two sweeps, forward and backward. This was because of time limitation of 1 hour per job on Karolina at the time of computing these results. The FRC is still not complete as the two sweeps did not have enough time to connect. Detail of the FRC for this testcase can be seen in Figure 6.49. For reference, the FRC for this testcase (H012345, quarter damping) was also computed by the direct solver (MUMPS). This solver is more efficient for smaller meshes and the its code also allows for resuming a continuation procedure from the end of a previous code run. A full FRC was computed with this solver and the result can be seen in Figure 6.50.

Max Newton steps	4
AFT point count	32
Continuation step init	5×10^{-3}
Continuation step min	10^{-5}
Continuation step max	5×10^{-2}
Continuation step scale up	1.2
Continuation step scale down	0.5

Table 6.15: FETI FRC testcase numerical parameters. Applies to both harmonic cases and all 3 damping cases.

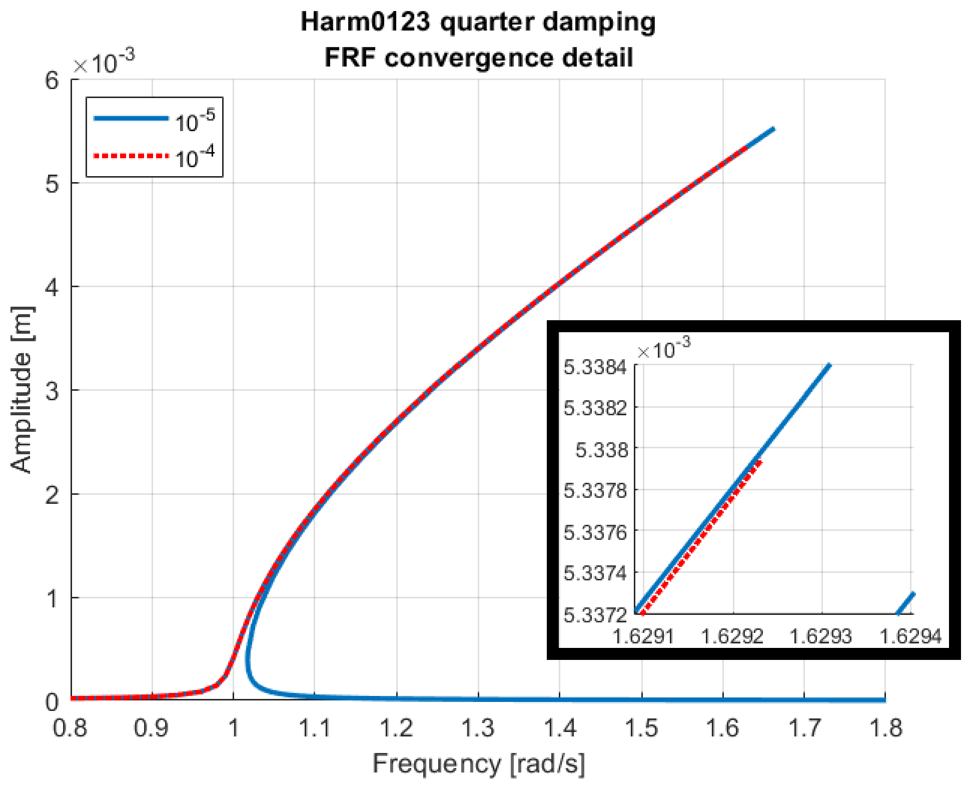


Figure 6.47: Detail of FRC for harmonics 0123 with quarter damping. Comparison of computed FRCs for 2 different GMRES tolerances. The looser tolerance case (red) stopped converging when approaching the first turning point.

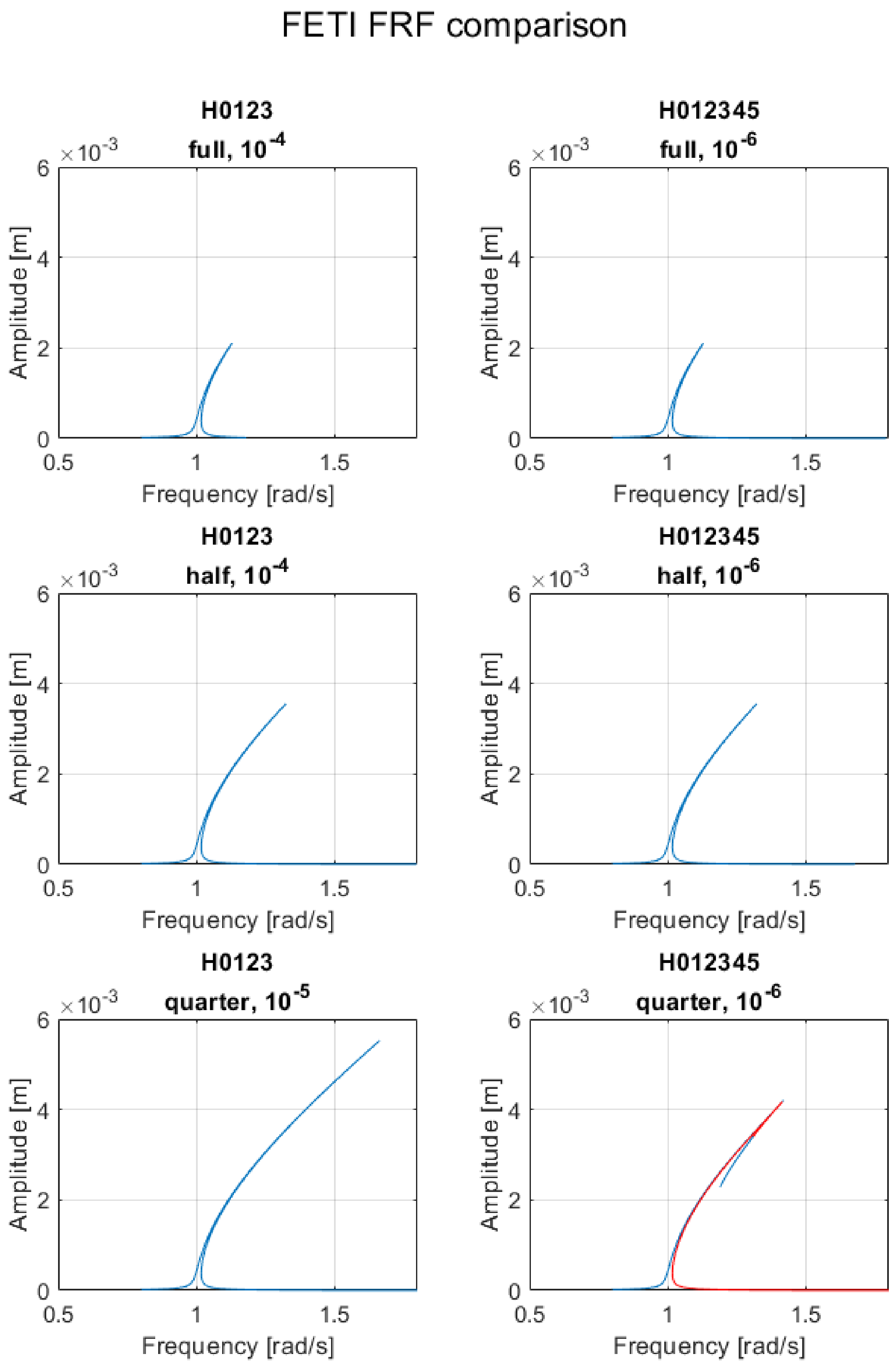


Figure 6.48: FETI FRC results comparison. Harmonics 0123 (left) and 012345 (right) for various damping values. Full damping (top row) is the standard $0.003 \times M$. Results for half and quarter of that value follow below. The number in the title indicates GMRES tolerance. Red line in the bottom right plot is a separate backward sweep.

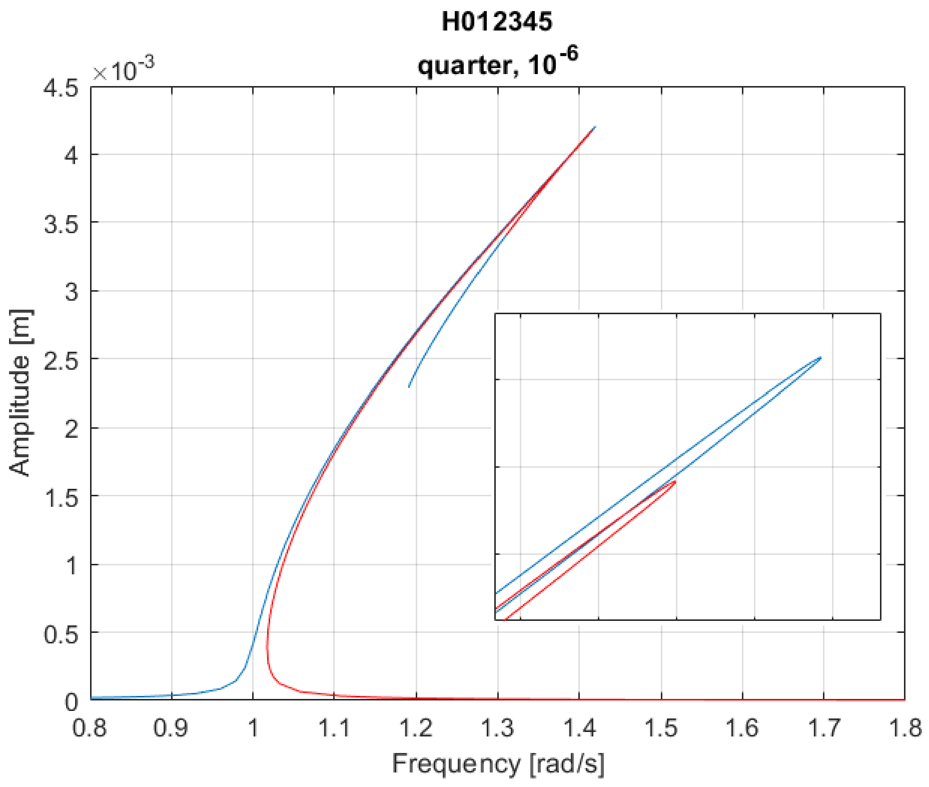


Figure 6.49: Detail of FRC for harmonics 012345 with quarter damping. Red line indicating a separate backward sweep. The two curves are not connected as the FRC is not complete.

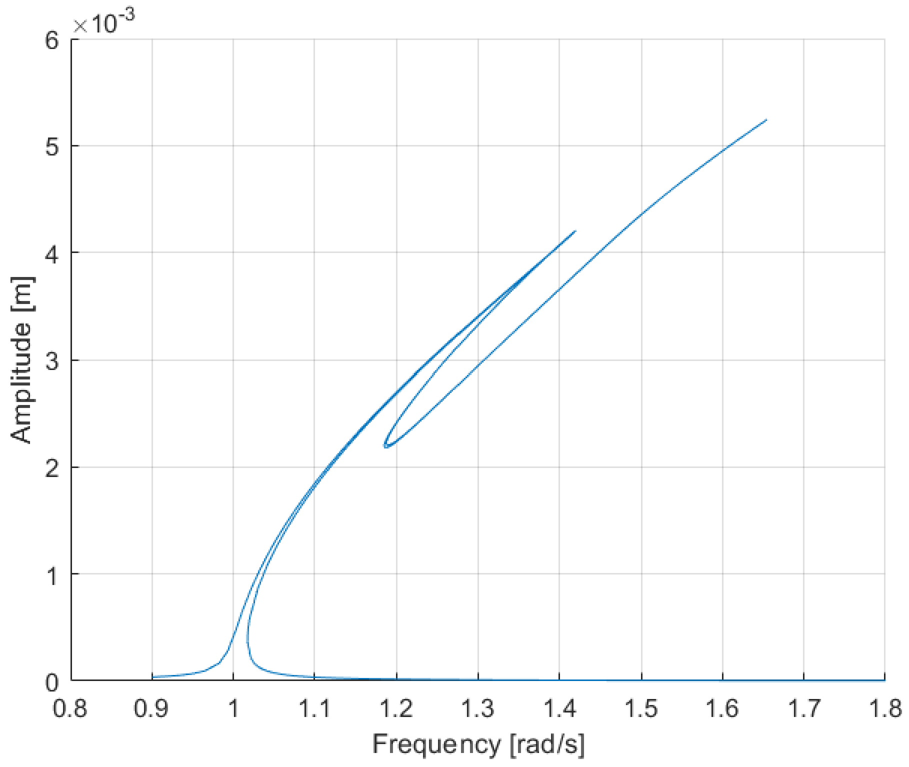


Figure 6.50: Reference full FRC computed with the direct solver (MUMPS) for the H012345 quarter damping testcase.

One of the main points of interest for this test was number of GMRES iterations. Until now, all tests of FETI have been performed on a single frequency point. This test offers an insight into how the number of GMRES iterations varies for different parts of the FRC. A summary of this can be found in Figure 6.51. There are two curves for each variant since there are 2 linear solves required for each Newton step when predictor-corrector algorithm is used, as described in section 4.8. The black dashed lines indicate positions of turning points. Data for the backward sweep for the quarter damping harm012345 case is displayed from left to right, while the direction of the sweep in frequency was from right to left. Only data for converged FRC points are displayed. Number of GMRES iterations was limited to 200.

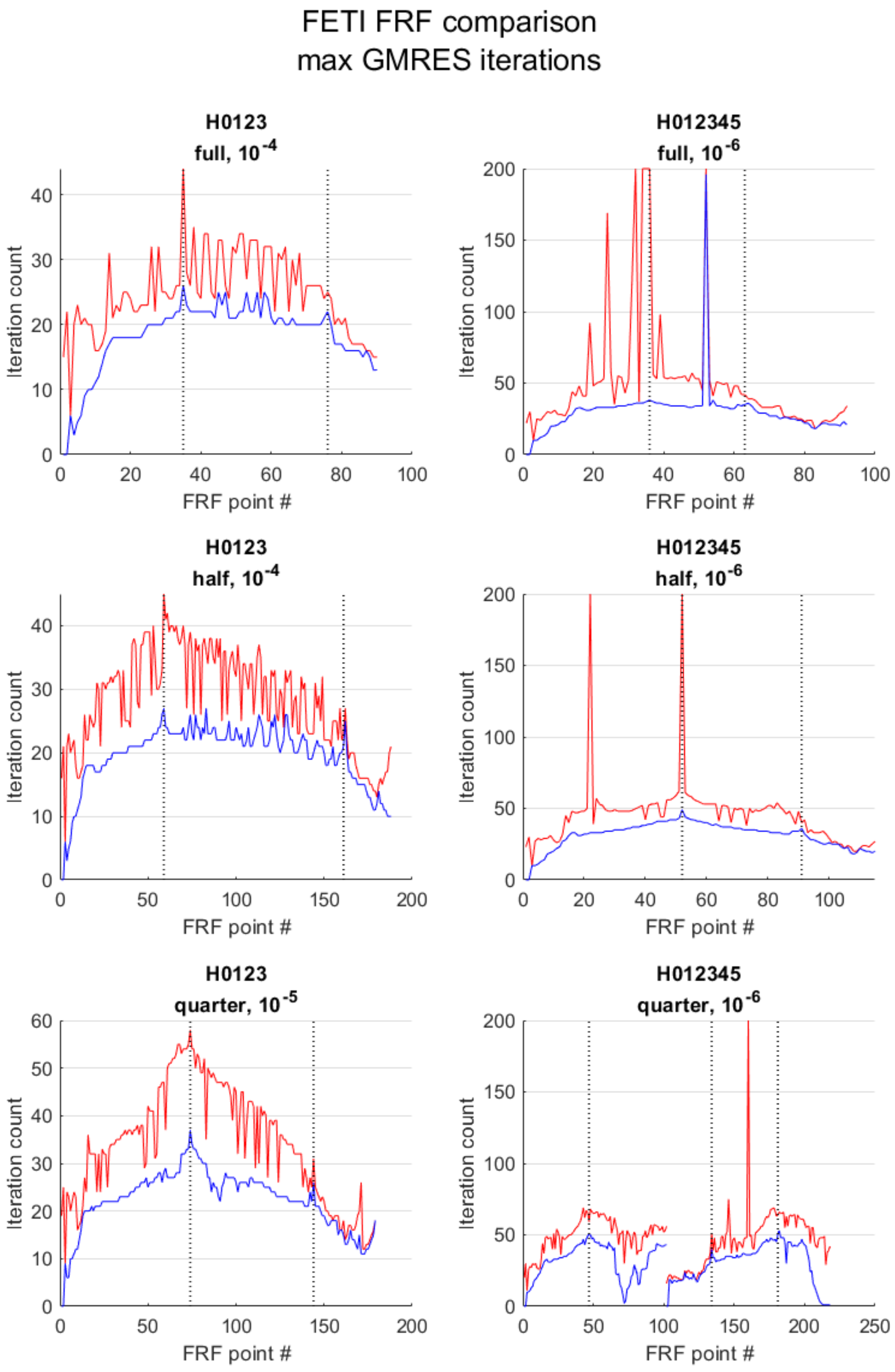


Figure 6.51: FETI FRC results comparison of maximum number of GMRES iterations (across Newton iterations) for each converged FRC point. The vertical dashed lines indicate locations of turning points. Two lines are present for the two solves required for the bordered solve. Red is for x_1 and blue for x_2, using notation from (4.74).

Discussion

This testcase demonstrates the capability of the FETI solver to compute a full nonlinear FRC of a structure. However, tests with much larger meshes are required. Still, several things can be concluded from this test. First, the requirement of stricter GMRES tolerance to properly handle turning points, as illustrated in Figure 6.47. Note that secant predictor was used in this testcase. It is possible that tangent predictor would yield better results in terms of Newton convergence. In Figure 6.51 it can be seen that for several FRC points the GMRES solver stagnated and did not manage to reach the tolerance before reaching maximum iteration count while Newton still converged. This further proves how important the role of GMRES tolerance is. Second, Figure 6.51 shows that the number of GMRES iterations is higher around turning points with the high amplitude (the 'peaks' of the FRCs). This is an important observation as the number of GMRES iterations significantly impacts the overall performance of the FETI solver, as was shown previously.

6.5 Conclusion

A series of results obtained from the created code is presented in this chapter. First, an overview of the employed hardware and list of used testcases and their parameters are laid out.

The first set of results covers solving the global HBM system. First, the code is validated by comparing computed FRCs to those computed for the same testcases independently by other researchers. After that, a strong scalability testcase for the MUMPS global solver on the clamped-clamped beam geometry with problem size of 1.7M dofs is presented. This testcase analyses scalability of various parts of the code, including the three stages of MUMPS solve procedure (analysis, factorisation and solve). It can be seen that the factorisation phase in MUMPS is the most expensive task and that its scalability diminishes with larger numbers of MPIs. This testcase also highlights how memory demanding direct parallel solvers can be. It is however noted that the solver was used without its OpenMP capability. Next, a fan blade geometry testcase is presented, with full nonlinear FRC computed. This demonstrates the capability of the code to operate on real industry-like geometries. This test also highlights the importance of harmonics selection. Finally, a test designed to assess the performance of the GMRES solver on the global system is made. This shows that GMRES equipped with standard preconditioners without any tuning performs poorly for nonlinear HBM problems and is not a viable option for large scale computing.

The second part of the results is dedicated to the FETI solver. First, a study of preconditioning and impact of the artificial coarse space is made. It can be seen that the Dirichlet preconditioner performs better out of the two in terms of both runtime and number of GMRES iterations. This is consistent with results from other studies. The test also demonstrates the importance of the artificial coarse space on both zero and nonzero harmonics.

Three scalability testcases for the FETI solver are presented next. The first weak scalability test pushes the solver to its limits in terms of RAM within the available 2 nodes on Karolina, with the largest case having 8.7 million dofs. The number of GMRES iterations in this testcase grows significantly with the number of domains, which negatively impacts the overall scalability. Even still, the FETI solver runtimes are comparable to those of MUMPS, while being more memory efficient. This testcase also analyses different levels of GMRES tolerances and their impact on the computed primal solution. The second weak scalability testcase uses a smaller mesh, with the largest case having 0.98M dofs. This testcase compares two mesh decompositions. It demonstrates the importance of the decomposition in FETI, as varying the decomposition strongly impacts the number of required GMRES iterations. This is again consistent with other studies on this topic. The first weak scalability testcase would most likely benefit from a better decomposition as well. It can be seen that with good decomposition choice the FETI solver is capable of outperforming the MUMPS solver in terms of runtime. The last testcase assesses strong scalability of the solver using a 1.6M dof system. Mesh decomposition was adjusted for each MPI size based on the previous testcase results. This testcase again shows the impact of GMRES scalability. The FETI solver nevertheless outperforms MUMPS.

Next, a comparison of solving two different versions of the projected dual problem equations is shown, highlighting the number of GMRES iterations and GMRES convergence stagnation. It shows that while the second set of projections implies larger computational cost of each GMRES iteration, it allows GMRES to converge to lower tolerances. The best form of the dual problem formulation remains however an open question.

Finally, a full FRC testcase of a clamped-clamped beam is shown. Three different levels of damping and two sets of harmonics are used for comparison. This testcase again demonstrates the importance of choosing the right tolerance for GMRES. It also highlights an increase in number of GMRES iterations around turning points of the FRCs.

Chapter 7

Conclusion

7.1 Research work conclusion

The present research builds on the well established methodology for nonlinear vibration analysis. It lays out all the necessary components that are necessary to compute full FRCs for 3D geometries with geometric nonlinearity using HBM. Attention is then focused on parallelisation of solving the linear system of equations inside the Newton loop. Three solving options are presented and analysed - solving the global system with a direct parallel solver (MUMPS), solving the global system with an iterative solver (GMRES) and applying a domain decomposition method (FETI). A standalone C++ code was developed to assess performance of these solvers [21].

7.1.1 Global system with direct solver

MUMPS has proven to be capable parallel solver for nonlinear HBM equations. This is not surprising as direct solvers are generally known to ”simply work” for any matrix passed to them. It does not require any parametrisation so it is very user friendly. This made MUMPS an ideal first solver to test. With this solver, a full nonlinear FRC for a fan blade geometry was computed, including successful handling of turning points. The scalability of this solver was assessed. It can be seen that the matrix factorisation phase struggles to scale with increasing number of domains for the used testcase. Memory demands of this solver were also shown to be a limiting factor for the solved problem size. It should be noted however that the solver was used without utilising OpenMP for shared memory parallelism. The solver has proven to be a reliable and robust tool for solving HBM equations across an entire frequency range while providing significant parallel speed-ups, up to a certain point.

7.1.2 Global system with GMRES

GMRES implementation in PETSc was also used to solve the global HBM system. However, it can be seen that this solver struggles to solve the nonlinear problem even in a frequency point with low nonlinearity influence. The number of GMRES iterations required grows significantly with the problem size and runtime is higher than with MUMPS by an order of magnitude. This discrepancy in performance is apparent on problems of sizes around 300k dofs already. This shows that solving the global HBM system with GMRES is not a feasible strategy. Even when coupled with some of the ready to use preconditioners available in PETSc the performance improved only marginally. A more sophisticated preconditioning is therefore necessary.

7.1.3 FETI

The FETI solver results demonstrate several important points. First, it shows that this solver is capable of solving a large nonlinear HBM system. Considering that FETI can be seen as a GMRES preconditioner, it is apparent that GMRES requires this type of sophisticated preconditioning to be able to handle large problems of this kind. MUMPS has also proven to be an essential component of the solver, allowing for parallel factorisations of the coarse space matrices.

The FETI solver in many cases outperforms MUMPS in terms of scalability and runtime. It is also capable of computing full nonlinear FRC including turning points. It was observed that the number of required GMRES iterations increases around turning points. This could lead to scalability issues with larger problems. More tests using larger number of MPIs would be needed in the future to better assess the solver's large scale capabilities.

Second, the results highlight various critical parameters of the solver. The GMRES tolerance can impact the number of iterations and therefore the code runtime in a significant way. It also strongly influences the error of domain connectivity. A strong enough tolerance level is also required to successfully go around turning points on nonlinear FRCs. Another important factor influencing the solver's performance is domain decomposition. The results seem to confirm previous research which suggests that bad aspect ratio of domains deteriorates GMRES convergence.

The exact form of the projected dual problem equations is also important for good GMRES convergence. No definitive conclusion about this topic is made as it requires more thorough mathematical analysis.

Finally, the importance of the artificial coarse space is highlighted. Without the presence of this coarse space on all harmonics GMRES convergence experiences significant stagnations. The artificial coarse space designed for this work is one of many possible ones. Other more fitting coarse spaces could possibly be used for even better convergence properties.

7.2 Scientific contributions

This work demonstrates a successful use of various parallelisation techniques to solve large scale vibration problems with geometric nonlinearity. While the focus of the work is placed on linear solvers, other parts of the code also contribute to the overall speed-up that was achieved. It shows that the current state of the art computing software for linear algebra, meshing, linear solvers and interprocessor communication can be used to compute full nonlinear vibration response of large 3D finite element models. This can allow future researchers to analyse larger systems in more detail and less time, allowing for more efficient design process of various industrial components in turbomachinery and other fields. The ability to compute high fidelity vibration responses is also important for validation of various complex modelling techniques such as reduced order models.

The FETI algorithm adjusted for the HBM problem used in this work is another contribution to the rich research around this method. It shows that this method is capable of efficiently solving nonlinear vibration problems. Both benefits and challenges related to using this method are highlighted. This can be used as an inspiration and benchmark for future research of this algorithm, especially regarding its use for nonsymmetric indefinite matrices.

7.3 Future work

Various topics of this work can be further expanded upon to make the whole process of solving nonlinear vibration problems more efficient and complete.

The physical model chosen for this work was very basic. More physical features can be added in the future, such as different material models, nonlinearities and employing more general boundary conditions. Contact and friction models are especially important for modelling of vibration of multipart moving components. Other physics such as fluid interaction and thermal properties can be modelled.

The numerical model can be extended by various features improving its flexibility and scope of applicability. Different number of harmonics can be used in different parts of the structure. Capability of tracking nonlinear normal modes, detection of bifurcations and stability analysis can be added to the continuation code. Different sources of motion equations can be added, such as 2D models or discrete spring-mass systems.

The FETI algorithm used in this work can be further improved. Optimisations regarding memory management and MPI communication can be made to speed up its performance. Implementing preconditioning and coarse space projection operations directly into code of the iterative solver can also increase its efficiency. Different FETI variants, such as FETI-DP, can be tested. Further research regarding the artificial coarse space can also provide significant convergence improvements.

Due to limited computational resources available at the time of testing, all large scale tests for the FETI solver were obtained with maximum of 256 cores. The scalability results indicate that the solver should be capable of handling much larger systems. Running the code on larger number of cores, preferably at least several thousand, will be necessary to properly assess its capabilities.

As discussed in this work, modelling and analysing vibration of structures such as gas turbines or their components is a complex task involving many steps [74, 162]. The methodology and code developed for this project are only a basic isolated entity to demonstrate capabilities of parallel computing applied to nonlinear vibration problems. The mathematical and software extensions mentioned in previous paragraphs are necessary to make the developed code useful in industrial applications. Additionally, the code needs to be integrated into an existing toolchain that an industrial company uses for its design and development process. Technicalities such as software compatibility, matching inputs and outputs or used libraries and system environments need to be addressed. When all these prerequisites are met, the code could serve as a valuable tool in industry for large scale vibration modelling, tackling complex structures such as whole gas turbines with fine mesh resolution.

Bibliography

[1] *Intel Math Kernel Library. Reference Manual*. Intel Corporation, 2009. Santa Clara, USA. ISBN 630813-054US.

[2] C. Alappat, A. Basermann, A. R. Bishop, H. Fehske, G. Hager, O. Schenk, J. Thies, and G. Wellein. A recursive algebraic coloring technique for hardware-efficient symmetric sparse matrix-vector multiplication. *ACM Trans. Parallel Comput.*, 7(3), June 2020.

[3] E. Allgower and K. Georg. *Introduction to Numerical Continuation Methods*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 2003.

[4] P. Amestoy, A. Buttari, J.-Y. L'Excellent, and T. Mary. Performance and Scalability of the Block Low-Rank Multifrontal Factorization on Multicore Architectures. *ACM Transactions on Mathematical Software*, 45:2:1–2:26, 2019.

[5] P. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

[6] I. Arany. The preconditioned conjugate gradient method with incomplete factorization preconditioners. *Computers & Mathematics with Applications*, 31(4):1–5, 1996. Selected Topics in Numerical Methods.

[7] I. Argyros, S. Hilout, and W. Scientific. *Computational Methods in Nonlinear Analysis: Efficient Algorithms, Fixed Point Theory and Applications*. Computational Methods in Nonlinear Analysis: Efficient Algorithms, Fixed Point Theory, and Applications. World Scientific, 2013.

[8] E. Babolian, J. Biazar, and A. Vahidi. Solution of a system of nonlinear equations by adomian decomposition method. *Applied Mathematics and Computation*, 150:847–854, 03 2004.

[9] S. Badia, A. Martín, and J. Principe. A highly scalable parallel implementation of balancing domain decomposition by constraints. *SIAM Journal on Scientific Computing*, 36:C190–C218, 01 2014.

[10] S. Balay, W. Gropp, L. C. McInnes, and B. F. Smith. Petsc, the portable, extensible toolkit for scientific computation. *Argonne National Laboratory*, 2(17), 1998.

[11] M. Balmaseda Aguirre. *Reduced order models for nonlinear dynamic analysis of rotating structures : Application to turbomachinery blades*. PhD thesis, Université de Lyon, September 2019.

[12] V. K. Bankar and A. S. Aradhye. A Review on Active, Semi-active and Passive Vibration Damping. In *International Journal of Current Engineering and Technology* , 2016.

[13] L. Beilina, E. Karchevskii, and M. Karchevskii. *Numerical Linear Algebra: Theory and Applications*. Springer International Publishing, 2017.

[14] M. Benzi. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics*, 182(2):418–477, 2002.

[15] M. Benzi, A. Frommer, R. Nabben, and D. Szyld. Algebraic theory of multiplicative schwarz methods. *Numerische Mathematik*, 89, 01 2001.

[16] P. Beran and C. Carlson. Domain-decomposition methods for bifurcation analysis. In *35th Aerospace Sciences Meeting and Exhibit*, 1997.

[17] D. Bertaccini and F. Durastante. *Iterative Methods and Preconditioning for Large and Sparse Linear Systems with Applications*. Chapman & Hall/CRC Monographs and Research Notes in Mathematics. CRC Press, 2018.

[18] O. Bessonov. Highly Parallel Multigrid Solvers for Multicore and Manycore Processors. In *Parallel Computing Technologies*, pages 10–20. Springer International Publishing, 2015.

[19] M. Bhatti. *Fundamental Finite Element Analysis and Applications: with Mathematica and Matlab Computations*. Wiley, 2005.

[20] M. Bhatti. *Advanced Topics in Finite Element Analysis of Structures: With Mathematica and MATLAB Computations*. Wiley, 2006.

[21] J. Blahoš. ParHBM - massively parallel C++ code for solvig nonlinear HBM problems. https://gitlab.com/Blahos/ParHBM/.

[22] J. Blahoš, A. Vizzaccaro, L. Salles, and F. Haddad. Parallel harmonic balance method for analysis of nonlinear dynamical systems. volume Volume 11: Structures and Dynamics: Structural Mechanics, Vibration, and Damping; Supercritical CO2 of *Turbo Expo: Power for Land, Sea, and Air*, 09 2020.

[23] M. Blatt, O. Ippisch, and P. Bastian. A massively parallel algebraic multigrid preconditioner based on aggregation for elliptic problems with heterogeneous coefficients. *arXiv*, 09 2012.

[24] S.-H. Boo, J.-H. Kim, and P.-S. Lee. Towards improving the enhanced craig-bampton method. *Computers & Structures*, 196:63–75, 2018.

[25] C. Boyer and U. Merzbach. *A History of Mathematics*. Wiley, 2011.

[26] T. Brzobohatý, M. Jarošová, T. Kozubek, M. Mensík, and A. Markopoulos. Hybrid Total FETI method. In *ECCOMAS 2012 - European Congress on Computational Methods in Applied Sciences and Engineering, e-Book Full Papers*, 01 2012.

[27] X.-C. Cai and O. B. Widlund. Domain decomposition algorithms for indefinite elliptic problems. *SIAM Journal on Scientific and Statistical Computing*, 13(1):243–258, 1992.

[28] D. Calvetti and L. Reichel. Iterative methods for large continuation problems. *Journal of Computational and Applied Mathematics*, 123, 03 2000.

[29] T. M. Cameron and J. H. Griffin. An alternating frequency/time domain method for calculating the steady-state response of nonlinear dynamic systems. *Journal of Applied Mechanics*, 56(1):149–154, 03 1989.

[30] W. Campbell. *Protection of Steam Turbine Disk Wheels from Axial Vibration*. General electric Company, 1924.

[31] A. Cardona, A. Lerusse, and M. Géradin. Fast fourier nonlinear vibration analysis. *Computational Mechanics*, 22:128–142, 1998.

[32] J. Cassels, Y. Fu, Y. Fu, R. Ogden, N. Hitchin, W. Elasticity, and L. M. Society. *Nonlinear Elasticity: Theory and Applications*. Lecture note series / London mathematical society. Cambridge University Press, 2001.

[33] T. F. Chan and Y. Saad. Iterative methods for solving bordered systems with applications to continuation methods. *SIAM Journal on Scientific and Statistical Computing*, 6(2):438–451, 1985.

[34] Y. Cheung, S. Chen, and S. Lau. Application of the incremental harmonic balance method to cubic non-linearity systems. *Journal of Sound and Vibration*, 140(2):273–286, 1990.

[35] H. S. Cho, H. Joo, Y. Lee, M. cheol Gwak, S.-J. Shin, and J. J. ick Yoh. Computational algorithm for nonlinear large-scale/multibody structural analysis based on co-rotational formulation with feti-local method. *Journal of The Korean Society for Aeronautical & Space Sciences*, 44:775–780, 2016.

[36] B. Cochelin, N. Damil, and M. Potier-Ferry. The asymptotic-numerical method: an efficient perturbation technique for nonlinear structural mechanics. *Revue Européenne des Éléments Finis*, 3(2):281–297, 1994.

[37] B. Cochelin and C. Vergez. Manlab, an interactive path following software, 01 2009.

[38] R. D. Cook, D. S. Malkus, M. E. Plesha, and R. J. Witt. *Concepts and Applications of Finite Element Analysis, 4th Edition*. Wiley, 4 edition, October 2001.

[39] D. Copeland and U. Langer. Domain decomposition solvers for nonlinear multiharmonic finite element equations. *Journal of Numerical Mathematics*, 18, 12 2009.

[40] R. Corral and J. Crespo Vaquerizo. Development of an edge-based harmonic balance method for turbomachinery flows. In *Proceedings of the ASME Turbo Expo*, volume 7, 01 2011.

[41] R. R. Craig and M. C. C. Bampton. Coupling of substructures for dynamic analyses. *AIAA Journal*, 6(7):1313–1319, 1968.

[42] C. D, C. Gibert, F. Thouverez, and D. J. Numerical and experimental study of friction damping blade attachments of rotating bladed disks. *International Journal of Rotating Machinery*, 2006, 07 2006.

[43] N. Damil and M. Potier-Ferry. A new method to compute perturbed bifurcations: Application to the buckling of imperfect elastic structures. *International Journal of Engineering Science*, 28:943–957, 1990.

[44] H. Dankowicz and F. Schilder. *Recipes for Continuation*. Computational Science and Engineering. Society for Industrial and Applied Mathematics, 2013.

[45] T. A. Davis, S. Rajamanickam, and W. M. Sid-Lakhdar. A survey of direct methods for sparse linear systems. *Acta Numerica*, 25:383 – 566, 2016.

[46] D. Demidov. AMGCL — A C++ library for efficient solution of large sparse linear systems. *Software Impacts*, 6:100037, 2020.

[47] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Analysis and Applications*, 20(3):720–755, 1999.

[48] T. Detroux, L. Renson, L. Masset, and G. Kerschen. The harmonic balance method for bifurcation analysis of large-scale nonlinear mechanical systems. *Computer Methods in Applied Mechanics and Engineering*, 296:18–38, 11 2015.

[49] A. Dhooge, W. Govaerts, and Y. A. Kuznetsov. Matcont: A matlab package for numerical bifurcation analysis of odes. *ACM Trans. Math. Softw.*, 29(2):141–164, jun 2003.

[50] E. Doedel. Auto: Software for continuation and bifurcation problems in ordinary differential equations. *Applied math. technical report Caltech*, 01 1986.

[51] V. Dolean, P. Jolivet, and F. Nataf. *An introduction to domain decomposition methods - algorithms, theory, and parallel implementation*. SIAM, 2015.

[52] W. Dong and P. Li. A parallel harmonic-balance approach to steady-state and envelope-following simulation of driven and autonomous circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(4):490–501, 2009.

[53] J. Dongarra and P. Luszczek. Top500. In *Encyclopedia of Parallel Computing*, pages 2055–2057. Springer US, 2011.

[54] Z. Dostál, D. Horák, and R. Kučera. Total feti—an easier implementable variant of the feti method for numerical solution of elliptic pde. *Communications in Numerical Methods in Engineering*, 22:1155–1162, 2006.

[55] M. Dryja and O. Widlund. *Towards a Unified Theory of Domain Decomposition Algorithms for Elliptic Problems*. Creative Media Partners, LLC, 2018.

[56] E. Efstathiou and M. Gander. Why restricted additive schwarz converges faster than additive schwarz. *Bit Numerical Mathematics - BIT*, 43:945–959, 12 2003.

[57] Y. El Gharbi, P. Gosselet, A. Parret-Fréaud, and C. Bovet. Hierarchical substructuring and parallel mesh generation of heterogeneous structures for domain decomposition methods. In *14th World Congress in Computational Mechanics and ECCOMAS Congress*, Paris, France, January 2021.

[58] M. Čermák, V. Hapla, J. Kružík, A. Markopoulos, and A. Vašatová. Comparison of different feti preconditioners for elastoplasticity. *Computers & Mathematics with Applications*, 74, 01 2017.

[59] R. Falgout, J. Jones, and U. Yang. The design and implementation of hypre, a library of parallel high performance preconditioners. *Numerical Solution of Partial Differential Equations on Parallel Computers*, pages 267–294, 01 2006.

[60] C. Farhat, P. Avery, R. Tezaur, and J. Li. Feti-dph: A dual-primal domain decomposition method for acoustic scattering. *Journal of Computational Acoustics*, 13(03):499–524, 2005.

[61] C. Farhat, M. Lesoinne, P. LeTallec, K. Pierson, and D. J. Rixen. Feti-dp: a dual–primal unified feti method—part i: a faster alternative to the two-level feti method. *International Journal for Numerical Methods in Engineering*, 50:1523 – 1544, 03 2001.

[62] C. Farhat, M. Lesoinne, and K. Pierson. A scalable substructuring method for static, transient and vibration analyses on massively parallel processors. In *41st Structures, Structural Dynamics, and Materials Conference and Exhibit*, 04 2000.

[63] C. Farhat, J. Li, and P. Avery. A feti-dp method for parallel iterative solution of indefinite and complex-valued solid and shell vibration problems. *International Journal for Numerical Methods in Engineering*, 63:398 – 427, 05 2005.

[64] C. Farhat and J. Mandel. The two-level feti method for static and dynamic plate problems part i: An optimal iterative solver for biharmonic systems. *Computer Methods in Applied Mechanics and Engineering*, 155(1):129–151, 1998.

[65] C. Farhat, J. Mandel, and F. X. Roux. Optimal convergence properties of the feti domain decomposition method. *Computer Methods in Applied Mechanics and Engineering*, 115(3):365–385, 1994.

[66] C. Farhat, K. Pierson, and M. Lesoinne. The second generation feti methods and their application to the parallel solution of large-scale linear and geometrically non-linear structural analysis problems. *Computer Methods in Applied Mechanics and Engineering*, 184(2):333–374, 2000.

[67] C. Farhat and F.-X. Roux. A method of finite element tearing and interconnecting and its parallel solution algorithm. *International Journal for Numerical Methods in Engineering*, 32:1205 – 1227, 10 1991.

[68] C. Firrone and S. Zucca. Modelling friction contacts in structural dynamics and its application to turbine bladed disks. IntechOpen, 09 2011.

[69] R. Fletcher. Conjugate gradient methods for indefinite systems. In *Numerical Analysis*, pages 73–89. Springer Berlin Heidelberg, 1976.

[70] I. Galliet and B. Cochelin. Une version parallèle des MAN par décomposition de domaine. *Revue Européenne des Éléments Finis*, 13(1-2):177–195, April 2012.

[71] J. Garcia Bedoy Torres. Improving harmonic balance performance via parallelization. Master's thesis, ITESO, 2016.

[72] M. Geradin and D. Rixen. *Mechanical Vibrations: Theory and Application to Structural Dynamics*. Wiley, 2015.

[73] A. Ghai, C. Lu, and X. Jiao. A comparison of preconditioned krylov subspace methods for large-scale nonsymmetric linear systems. *Numerical Linear Algebra with Applications*, 26(1):e2215, 2019.

[74] D. Giagopoulos, A. Arailopoulos, I. Zacharakis, and E. Pipili. Finite element model developed and modal analysis of large scale steam turbine rotor: Quantification of uncertainties and model updating. In *UNCECOMP 2017*, pages 32–44, 01 2017.

[75] A. Golbabai and M. Javidi. A new family of iterative methods for solving system of nonlinear algebric equations. *Applied Mathematics and Computation*, pages 1717–1722, 07 2007.

[76] P. Gosselet, D. J. Rixen, F.-X. Roux, and N. Spillane. Simultaneous-FETI and Block-FETI: robust domain decomposition with multiple search directions. *International Journal for Numerical Methods in Engineering*, 104(10):905–927, 2015.

[77] A. Greenbaum. *Iterative Methods for Solving Linear Systems*. Frontiers in Applied Mathematics. Society for Industrial and Applied Mathematics, 1997.

[78] A. Greenbaum and L. N. Trefethen. Gmres/cr and arnoldi/lanczos as matrix approximation problems. *SIAM Journal on Scientific Computing*, 15(2):359–368, 1994.

[79] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[80] L. Guillot, B. Cochelin, and C. Vergez. A generic and efficient taylor series–based continuation method using a quadratic recast of smooth nonlinear systems. *International Journal for numerical methods in Engineering*, 119(4):261–280, 2019.

[81] L. Guillot, B. Cochelin, and C. Vergez. A taylor series-based continuation method for solutions of dynamical systems. *Nonlinear Dynamics*, 98(4):2827–2845, 2019.

[82] M. E. Gurtin. The linear theory of elasticity. In *Linear Theories of Elasticity and Thermoelasticity: Linear and Nonlinear Theories of Rods, Plates, and Shells*, pages 1–295. Springer Berlin Heidelberg, 1973.

[83] M. Guskov, J.-J. Sinou, and F. Thouverez. Multi-dimensional harmonic balance applied to rotor dynamics. *Mechanics Research Communications*, 35:537–545, 2008.

[84] L. Říha, T. Brzobohatý, and A. Markopoulos. Highly scalable feti methods in espreso. *Civil-Comp Proceedings*, 107, 2015. cited By 0.

[85] L. Říha, M. Merta, R. Vavřík, T. Brzobohatý, A. Markopoulos, O. Meca, O. Vysocky, T. Kozubek, and V. Vondrak. A massively parallel and memory-efficient fem toolbox with a hybrid total feti solver with accelerator support. *The International Journal of High Performance Computing Applications*, 33:109434201879845, 09 2018.

[86] R. M. Haferssas, P. Jolivet, and F. Nataf. An Additive Schwarz Method Type Theory for Lions's Algorithm and a Symmetrized Optimized Restricted Additive Schwarz Method. *SIAM Journal on Scientific Computing*, 39(4):A1345 – A1365, February 2017.

[87] Z. Hamid, W. Mtalaa, J. Brunelot, and M. Potier-Ferry. Asymptotic numerical method for strong nonlinearities. *Revue Europeenne des Elements Finis*, 2004:97–118, 04 2012.

[88] P. Hartley, I. Pillinger, and C. Sturgess. *Numerical Modelling of Material Deformation Processes: Research, Development and Applications*. Springer London, 2012.

[89] V. E. Henson and U. M. Yang. Boomeramg: A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 41(1):155–177, 2002. Developments and Trends in Iterative Methods for Large Systems of Equations - in memorium Rudiger Weiss.

[90] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49:409–435, 1952.

[91] A. Hrennikoff. Solution of problems of elasticity by the framework method. *Journal of Applied Mechanics*, 8(4):A169–A175, 12 1941.

[92] D. Inman. *Engineering Vibration*. Pearson Education, 2009.

[93] R. Ju and W. Zhu. An optimized efficient galerkin averaging-incremental harmonic balance method for high-dimensional spatially discretized models of continuous systems based on parallel computing. *Journal of Computational and Nonlinear Dynamics*, 16, 08 2021.

[94] E. Kaasschieter. Preconditioned conjugate gradients for solving singular systems. *Journal of Computational and Applied Mathematics*, 24(1):265–275, 1988.

[95] S.-H. Kang, Y. Kim, H. Cho, and S. Shin. Improved hyper-reduction approach for the forced vibration analysis of rotating components. *Computational Mechanics*, pages 1–14, 2022.

[96] G. Karypis. METIS and ParMETIS. In *Encyclopedia of Parallel Computing*, pages 1117–1124. Springer US, 2011.

[97] G. Karypis and V. Kumar. MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0. http://www.cs.umn.edu/~metis, 2009.

[98] C. Kelley. *Solving Nonlinear Equations with Newton's Method*. Fundamentals of Algorithms. Society for Industrial and Applied Mathematics, 2003.

[99] N. Kessab, B. Braikat, L. Hassane, N. Damil, and M. Potier-Ferry. High order predictor-corrector algorithms for strongly non-linear problems. *Revue de Mécanique Appliquée et Théorique*, 1:587–613, 01 2006.

[100] J.-G. Kim and P.-S. Lee. An enhanced craig–bampton method. *International Journal for Numerical Methods in Engineering*, 103(2):79–93, 2015.

[101] J. H. Kim, C. S. Lee, and S. J. Kim. Development of a high-performance domain-wise parallel direct solver for large-scale structural analysis. In *Proceedings. Seventh International Conference on High Performance Computing and Grid in Asia Pacific Region, 2004.*, pages 267–274, 2004.

[102] Y. Kim, H. Cho, S. Park, H. Kim, and S. Shin. Advanced structural analysis based on reduced-order modeling for gas turbine blade. *AIAA Journal*, 56(8):3369–3373, 2018.

[103] Y. Kim, S.-H. Kang, H. Cho, and S. Shin. Improved nonlinear analysis of a propeller blade based on hyper-reduction. *AIAA Journal*, 60(3):1909–1922, 2022.

[104] B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey. `libMesh`: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations. *Engineering with Computers*, 22(3–4):237–254, 2006.

[105] A. Klawonn and O. Rheinbach. Inexact feti-dp methods. *International Journal for Numerical Methods in Engineering*, 69(2):284–307, 2007.

[106] A. Klawonn and O. Rheinbach. Highly scalable parallel domain decomposition methods with an application to biomechanics. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 90(1):5–32, 2010.

[107] D. Knoll and D. Keyes. Jacobian-free newton–krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004.

[108] J. Košata, J. del Pino, T. L. Heugel, and O. Zilberberg. Harmonicbalance.jl: A julia suite for nonlinear dynamics using harmonic balance, 2022.

[109] O. Kononenko. A massively parallel solver for the mechanical harmonic analysis of accelerator cavities. 2 2015.

[110] T. Kozubek, V. Vondrak, D. Horák, Z. Dostal, V. Hapla, and M. Čermák. Total feti domain decomposition method and its massively parallel implementation. *Advances in Engineering Software*, s 60–61:14–22, 06 2013.

[111] M. Krack and J. Gross. *Harmonic Balance for Nonlinear Vibration Problems*. Mathematical Engineering. Springer International Publishing, 2019.

[112] N. Krylov, N. Bogoliubov, and S. Lefschetz. *Introduction to Non-linear Mechanics*. Annals of mathematics studies. Princeton University Press, 1947.

[113] R. J. Kuether and A. Steyer. Multi-harmonic balance with preconditioned iterative solver. International Modal Analysis Conference (IMAC) XXXIX, 12 2020.

[114] V. Kumar and A. Gupta. Analyzing scalability of parallel algorithms and architectures. *Journal of Parallel and Distributed Computing*, 22(3):379–391, 1994.

[115] H. Lahmam, J. M. Cadou, H. Zahrouni, N. Damil, and M. Potier-Ferry. High-order predictor–corrector algorithms. *International Journal for Numerical Methods in Engineering*, 55(6):685–704, 2002.

[116] A. J. Laub. *Matrix Analysis For Scientists And Engineers*. Society for Industrial and Applied Mathematics, USA, 2004.

[117] P. Li and W. Dong. Parallel preconditioned hierarchical harmonic balance for analog and rf circuit simulation. In *Advances in Analog Circuits*, chapter 5. IntechOpen, 2011.

[118] X. S. Li and J. W. Demmel. SuperLU_DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems. *ACM Trans. Mathematical Software*, 29(2):110–140, June 2003.

[119] G. Liu and S. S. Quek. The finite element method: A practical course: Second edition. *The Finite Element Method: A Practical Course: Second Edition*, 01 2003.

[120] R. Lumba and A. Datta. Development of a Parallel 3D FEA Multibody Solver and Partitioner for Rotorcraft. In *Development of a Parallel 3D FEA Multibody Solver and Partitioner for Rotorcraft*, 01 2022.

[121] J. Mandel and M. Brezina. Balancing domain decomposition: Theory and performance in two and three dimensions. Technical report, Center for Computational Mathematics, University of Colorado at Denver, 1993.

[122] J. Mandel, C. Dohrmann, and R. Tezaur. An algebraic theory for primal and dual substructuring methods by constraints. *Applied Numerical Mathematics*, 54:167–193, 07 2005.

[123] J. Mandel and C. R. Dohrmann. Convergence of a balancing domain decomposition by constraints and energy minimization. *Numerical Linear Algebra with Applications*, 10(7):639–659, 2003.

[124] J. Mandel and R. Tezaur. On the convergence of a dual-primal substructuring method. *Numerische Mathematik*, 88, 05 2000.

[125] A. Markopoulos. Technical report, IT4Innovations National Supercomputing Center, Czech Republic.

[126] O. Meca, L. Říha, and T. Brzobohatý. An Approach for Parallel Loading and Pre-Processing of Unstructured Meshes Stored in Spatially Scattered Fashion. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 749–760, 2019.

[127] M. Mignolet, A. Przekop, S. Rizzi, and S. Spottswood. A review of indirect/non-intrusive reduced order modeling of nonlinear geometric structures. *Journal of Sound and Vibration*, 332:2437–2460, 05 2013.

[128] R. Mittal and A. Al-Kurdi. An efficient method for constructing an ilu preconditioner for solving large sparse nonsymmetric linear systems by the gmres method. *Computers & Mathematics with Applications*, 45(10):1757–1772, 2003.

[129] N. Myklestad. *Fundamentals of Vibration Analysis*. McGraw-Hill, 1956.

[130] V. K. Nguyen, H. Cuong, and M.-T. Nguyen-Thai. Calculation of nonlinear vibrations of piecewise-linear systems using the shooting method. *Vietnam Journal of Mechanics*, 34, 09 2012.

[131] M. Noor and M. Waseem. Some iterative methods for solving a system of nonlinear equations. *Computers & Mathematics with Applications*, 57:101–106, 01 2009.

[132] J. Ortega and W. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 1970.

[133] C. Padmanabhan and R. Singh. Analysis of periodically excited non-linear systems by a parametric continuation technique. *Journal of Sound and Vibration*, 184:35–58, 1995.

[134] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.

[135] G. Paraschos and M. Vouvakis. The dual, overlapping primal feti (feti-dop) domain decomposition. *IEEE Antennas and Propagation Society, AP-S International Symposium (Digest)*, pages 2983–2986, 07 2011.

[136] K. C. Park, C. Felippa, and U. Gumaste. A localized version of the method of lagrange multipliers and its applications. *Computational Mechanics*, 24:476–490, 01 2000.

[137] M. Patil and A. Datta. A scalable time-parallel solution of periodic rotor dynamics in x3d. *Journal of the American Helicopter Society*, 01 2021.

[138] M. Patil and A. Datta. Time-parallel scalable solution of periodic rotor dynamics for large-scale 3d structures. In *AIAA Scitech 2021 Forum*. 2021.

[139] J. W. Pearson and J. Pestana. Preconditioners for krylov subspace methods: An overview. *GAMM-Mitteilungen*, 43(4):e202000015, 2020.

[140] F. Pellegrini and J. Roman. Scotch: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs. In *High-Performance Computing and Networking*, pages 493–498. Springer Berlin Heidelberg, 1996.

[141] E. Petrov. Analytical formulation of friction interface elements for analysis of nonlinear multi-harmonic vibrations of bladed disks. *Journal of Turbomachinery*, 125:364–371, 02 2003.

[142] E. Petrov. A method for use of cyclic symmetry properties in analysis of nonlinear multiharmonic vibrations of bladed disks. *Journal of Turbomachinery*, 126:175–183, 01 2004.

[143] E. Petrov and D. Ewins. Effects of damping and varying contact area at blade-disk joints in forced response analysis of bladed disk assemblies. *Journal of Turbomachinery*, 128:403–410, 04 2006.

[144] S. Pissanetzky. *Sparse Matrix Technology*. Elsevier Science, 2014.

[145] A. Popp and P. Wriggers. *Contact Modeling for Solids and Particles*. Springer, 2018.

[146] S. Pradhan and S. Modak. A Review of Damping Matrix Identification Methods in Structural Dynamics. In *ASME International Mechanical Engineering Congress and Exposition, Proceedings (IMECE)*, volume 12, 11 2012.

[147] L. Qi and J. Sun. A nonsmooth version of newton's method. *Math. Program.*, 58:353–367, 01 1993.

[148] Y. Qiu. Spectra. a library for large scale eigenvalue problems, 2018. https://mloss.org/software/view/689/.

[149] Y. Quere. *Physics of Materials*. CRC Press, 2020.

[150] D. Rixen, B. Seeger, W.-C. Tai, S. Baek, T. Dossogne, M. Allen, R. Kuether, M. Brake, and R. Mayes. A comparison of reduced order modeling techniques used in dynamic substructuring. pages 511–528. Springer, 01 2016.

[151] D. J. Rixen. Domain decomposition solvers (feti), a random walk in history and some current trends, 10 2014. https://cupdf.com/document/domain-decomposition-solvers-feti-domain-decomposition-solvers-feti-a-random.html.

[152] D. J. Rixen. Personal sessions discussing the feti method, 2022.

[153] D. J. Rixen and C. Farhat. A simple and efficient extension of a class of substructure based preconditioners to heterogeneous structural mechanics problems. *International Journal for Numerical Methods in Engineering*, 44:489–516, 1999.

[154] A. Rizvi, C. Smith, R. Rajasekaran, and E. Ken. Dynamics of dry friction damping in gas turbines: Literature survey. *Journal of Vibration and Control*, 22, 01 2014.

[155] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, second edition, 2003.

[156] Y. Saad and M. H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[157] L. Salles, L. Blanc, A. Gouskov, P. Jean, and F. Thouverez. Dual time stepping algorithms with the high order harmonic balance method for contact interfaces with fretting-wear. *Proceedings of the ASME Turbo Expo*, 6, 03 2014.

[158] L. Salles, B. Staples, N. Hoffmann, and C. Schwingshackl. Continuation techniques for analysis of whole aeroengine dynamics with imperfect bifurcations and isolated solutions. *Nonlinear Dynamics*, 86, 11 2016.

[159] C. Sanderson and R. Curtin. Armadillo: A template-based c++ library for linear algebra. *Journal of Open Source Software*, 1:26, 07 2016.

[160] C. Sanderson and R. Curtin. An Adaptive Solver for Systems of Linear Equations. In *2020 14th International Conference on Signal Processing and Communication Systems (ICSPCS)*. IEEE, 12 2020.

[161] C. Sanghavi. *FETI methods for acoustic problems with porous materials*. PhD thesis, Le Mans Université, 09 2020.

[162] S. Scalvini. Nonlinear vibration of bladed discs. Master's thesis, Politecnico di Torino, 2021.

[163] O. Schenk. *Scalable parallel sparse LU factorization methods on shared memory multiprocessors*. PhD thesis, ETH Zurich, 2000.

[164] O. Schenk and K. Gärtner. PARDISO. In *Encyclopedia of Parallel Computing*, pages 1458–1464. Springer US, Boston, MA, 2011.

[165] H. Schwarz. Ueber einige abbildungsaufgaben. *Journal für die reine und angewandte Mathematik*, 1869(70):105–120, 1869.

[166] E. Seinturier. Forced Response Computation for Bladed Disks Industrial Practices and Advanced Methods. World Congress in Mechanism and Machine Science, 2007.

[167] W. Slaughter. *The Linearized Theory of Elasticity*. Birkhäuser Boston, 2002.

[168] I. Smith, D. Griffiths, and L. Margetts. *Programming the Finite Element Method: Fifth Edition*. 06 2015.

[169] P. Sonneveld. Cgs, a fast lanczos-type solver for nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 10(1):36–52, 1989.

[170] N. Spillane and D. J. Rixen. Automatic spectral coarse spaces for robust finite element tearing and interconnecting and balanced domain decomposition algorithms. *International Journal for Numerical Methods in Engineering*, 95, 2013.

[171] K. Sze, S. Chen, and J. Huang. The incremental harmonic balance method for nonlinear vibration of axially moving beams. *Journal of Sound and Vibration*, 281:611–626, 03 2005.

[172] The European Union. models, EXperiments and high PERformance computing for Turbine mechanical Integrity and Structural dynamics in Europe. https://cordis.europa.eu/project/id/721865.

[173] The Trilinos Project Team. The Trilinos Project Website. https://trilinos.github.io.

[174] O. Thomas, A. Sénéchal, and J.-F. Deü. Hardening/softening behavior and reduced order modeling of nonlinear vibrations of rotating cantilever beams. *Nonlinear Dynamics*, 86:1293–1318, 2016.

[175] J. Thomsen. *Vibrations and Stability: Advanced Theory, Analysis, and Tools*. Springer Berlin Heidelberg, 2013.

[176] J. Toivanen, P. Avery, and C. Farhat. A multilevel feti-dp method and its performance for problems with billions of degrees of freedom. *International Journal for Numerical Methods in Engineering*, 116(10-11):661–682, 2018.

[177] J.-C. Tournier, V. Donde, and Z. Li. Comparison of direct and iterative sparse linear solvers for power system applications on parallel computing platforms. *17th Power Systems Computation Conference, PSCC 2011*, 01 2011.

[178] C. Touzé and A. Vizzaccaro. Model order reduction methods for geometrically nonlinear structures: a review of nonlinear techniques. *Nonlinear Dynamics*, accepted, 07 2021.

[179] M. Urabe and A. Reiter. Numerical computation of nonlinear forced oscillations by galerkin's procedure. *Journal of Mathematical Analysis and Applications*, 14:107–140, 1966.

[180] M. Urabe and Wisconsin University Madison Mathematics Research Center. *Galerkin's Procedure for Nonlinear Periodic Systems*. MRC technical summary report. Defense Technical Information Center, 1964.

[181] A. Vašatová, J. Tomčala, R. Sojka, M. Pecha, J. Kružík, D. Horák, V. Hapla, and M. Čermák. Parallel strategies for solving the feti coarse problem in the permon toolbox. In *Programs and Algorithms of Numerical Mathematics 18*, pages 154–163, 06 2017.

[182] H. van der Vorst. *Iterative Krylov Methods for Large Linear Systems*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2003.

[183] H. A. van der Vorst. Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13(2):631–644, 1992.

[184] A. Vizzaccaro. *Reduced Order Modelling of Large Finite Element Structures with Geometric and Contact Nonlinearities: Application to Blade-Casing Interaction in Aircraft Engines*. PhD thesis, Imperial College London, 2 2021.

[185] A. Vizzaccaro, Y. Shen, L. Salles, J. Blahoš, and C. Touzé. Direct computation of nonlinear mapping via normal form for reduced-order models of finite element nonlinear structures. *Computer Methods in Applied Mechanics and Engineering*, 384:113957, 10 2021.

[186] O. Vysocký, J. Zapletal, and L. Říha. A simple framework for energy efficiency evaluation and hardware parameter tuning with modular support for different hpc platforms. In *The Eighth International Conference on Advanced Communications and Computation*, 06 2018.

[187] D. Wakam and A. K. Guy Antoine. Parallel gmres with a multiplicative schwarz preconditioner. *Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées*, Volume 14 - 2011 - Special..., 08 2010.

[188] C. Wang, D. Zhang, Y. Ma, Z. Liang, and J. Hong. Dynamic behavior of aero-engine rotor with fusing design suffering blade off. *Chinese Journal of Aeronautics*, 30(3):918–931, 2017.

[189] F. Wenneker. *Component Mode Synthesis for geometrically nonlinear structures*. PhD thesis, Delft University of Technology, 11 2013.

[190] F. Wenneker and P. Tiso. A substructuring method for geometrically nonlinear structures. In *Dynamics of Coupled Structures, Volume 1*, pages 157–165. Springer International Publishing, 03 2014.

[191] P. Wesseling. *An Introduction to Multigrid Methods*. An Introduction to Multigrid Methods. R.T. Edwards, 2004.

[192] L. Xie, S. Baguet, B. Prabel, and R. Dufour. Bifurcation tracking by harmonic balance method for performance tuning of nonlinear dynamical systems. *Mechanical Systems and Signal Processing*, 88:445–461, 05 2017.

[193] J. Xu and X.-C. Cai. A preconditioned gmres method for nonsymmetric or indefinite problems. *Mathematics of Computation*, 59:311–319, 1992.

[194] X. Yang, J. Du, and Z. Wang. An effective speedup metric for measuring productivity in large-scale parallel computer systems. *The Journal of Supercomputing*, 56:164–181, 05 2011.

[195] W. Yao, J. Jin, and P. T. Krein. A 3d finite element analysis of large scale nonlinear dynamic electromagnetic problems by harmonic balancing and domain decomposition. *International Journal of Numerical Modelling-electronic Networks Devices and Fields*, 29:166–180, 2016.

Appendix A

Cantilever beam response study

In section 6.3.3 a testcase was presented showing FRCs for an industry-like fan blade geometry. It can be seen in Figures 6.14 and 6.15 that the exact choice of harmonics impacts the results greatly. This appendix provides a supporting testcase for a better understanding of how such a difference in results can in principle be explained.

The testcase is a cantilever beam excited at its free end, similar to the one in Table 6.5. This testcase was chosen because it can be analysed more clearly compared to the complex shaped geometry of the blade. It can however be still used to demonstrate the difference that choosing different sets of harmonics makes. The diagram of the testcase is copied here for clarity (see Figure A.1). There are a few minor differences from the original cantilever testcase presented in 6.2. First, the mesh here has $1 \times 1 \times 80$ regular HEXA20 elements. Second, the excitation force at the free end is equally split among the 4 corner nodes but still adding up to the same $2N$ in total (see Figure A.2). Numerical parameters for each testcase are shown in table A.1. Tangent predictor was used in all cases.

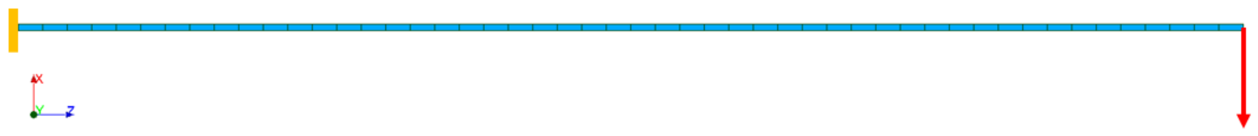


Figure A.1: Diagram of the cantilever beam testcase for the appendix, same as in Figure 6.2.

Direct solver (meaning MUMPS) was used for its better performance for smaller meshes. FRCs were computed around the first resonant frequency using several sets of harmonics - H1, H01, H012, H0123, H012345, H01234567 and H0123456789. The amplitude was measured on a node at the free end of the beam. Full FRCs were computed unlike in the results in 6.3.1 because a newer version of the code was used. All FRCs together can be seen in Figure A.3. The following figures then show zoomed version of this figure. Figure A.4 highlights FRCs for harmonics 1 and 01. Figure A.5 emphasizes differences between

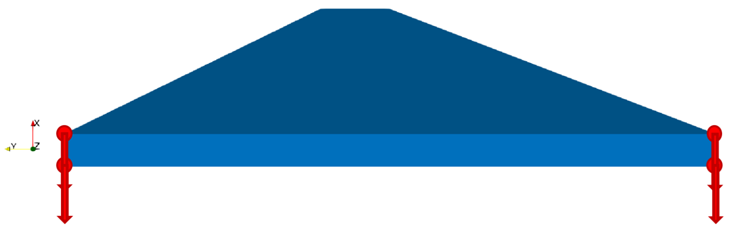


Figure A.2: Diagram of the cantilever beam testcase for the appendix, highlighting the distribution of the excitation force. The total excitation magnitude is $2N$.

Testcase	AFT	Newton tol.	Newton it.	cont. step size	cont. step scaling
H1	24	4×10^{-5}	10	$\langle 10^{-5}, 0.08, 0.08 \rangle$	$0.4, 1.1$
H01	24	5×10^{-4}	24	$\langle 10^{-5}, 0.1, 0.1 \rangle$	$0.4, 1.1$
H012	48	5×10^{-4}	24	$\langle 10^{-5}, 0.1, 0.4 \rangle$	$0.4, 1.1$
H0123	48	5×10^{-4}	24	$\langle 10^{-5}, 0.1, 0.4 \rangle$	$0.4, 1.1$
H012345	64	5×10^{-4}	24	$\langle 10^{-5}, 0.1, 0.4 \rangle$	$0.4, 1.1$
H01234567	96	5×10^{-4}	24	$\langle 10^{-5}, 0.1, 0.4 \rangle$	$0.4, 1.1$
H0123456789	96	5×10^{-4}	24	$\langle 10^{-5}, 0.1, 0.4 \rangle$	$0.4, 1.1$

Table A.1: Numerical parameters for the cantilever beam continuation testcases. Continuation step size is in format $\langle stepmin, stepinit, stepmax \rangle$. Continuation step scaling are constants to scale down and up the continuation step size (see Algorithm 2).

harmonics 012 and both 0123 and 012345. The final Figure A.6 zooms in at the difference between harmonics 0123 and 012345.

The two highest harmonic cases - H01234567 and H0123456789 - were used for verification. Figure A.7 shows comparison of the H012345 case with those two cases. It can be seen that all 3 FRCs are nearly identical.

Besides the standard FRCs projections on modes have also been computed. Specifically, the solutions were projected on modes 2 and 20 (sorted by corresponding eigenfrequency in ascending order). These two modes can be seen in Figure A.8 together with the first mode which is the most excited mode. This is due to the location of the excitation force and the range of analysed frequencies being around the first mode's resonance. Each solution vector in frequency domain $\tilde{u}$ of an FRC was first split into individual harmonics. For each harmonic i the projection on mode k was computed as:

$$q_{i,k} = \frac{\sqrt{\left[(\tilde{u}_i^c)^T M v_k\right]^2 + \left[(\tilde{u}_i^s)^T M v_k\right]^2}}{v_k^T M v_k} \tag{A.1}$$

with $q_{i,k}$ being the modal amplitude for the i-th harmonic of the solution related to mode v_k. The modes v_k were normalised so that their largest element in absolute value is 1. From Figure A.8 in can be seen that in both cases this maximum value is located at the free end of the beam.

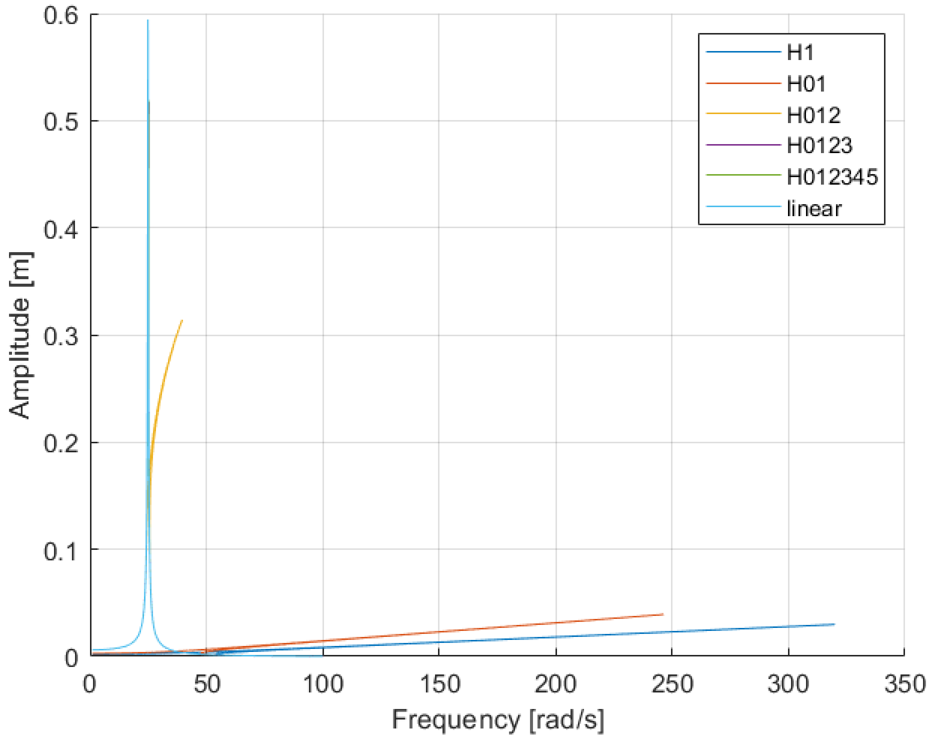


Figure A.3: Cantilever FRCs for various sets of harmonics.

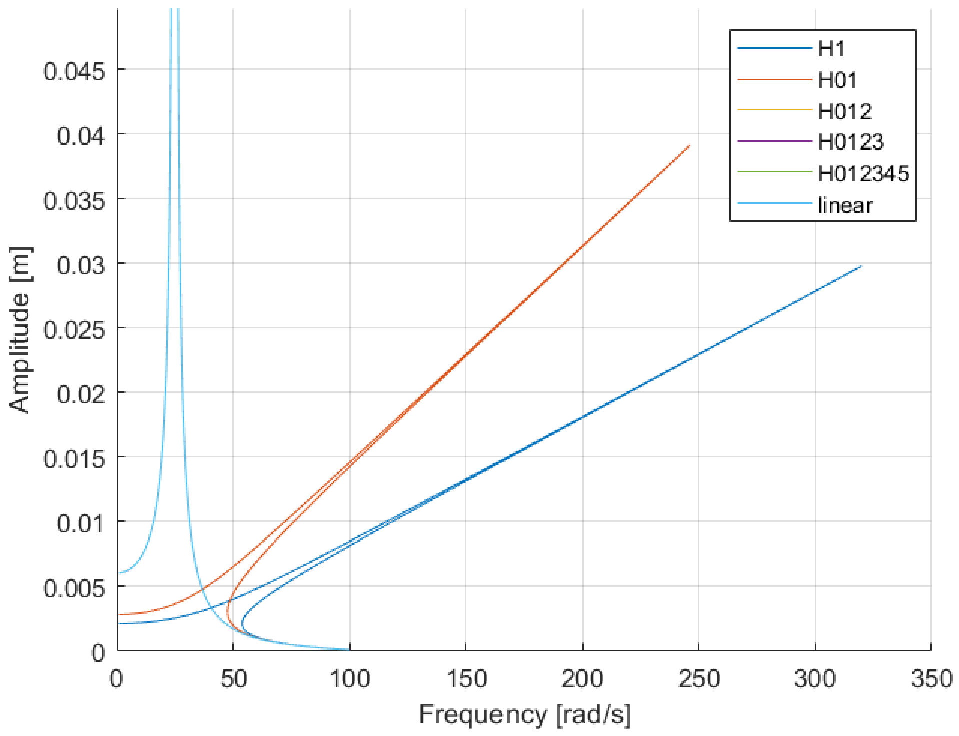


Figure A.4: Cantilever FRCs for various sets of harmonics. Highlight of the results for harmonics 1 and 01.

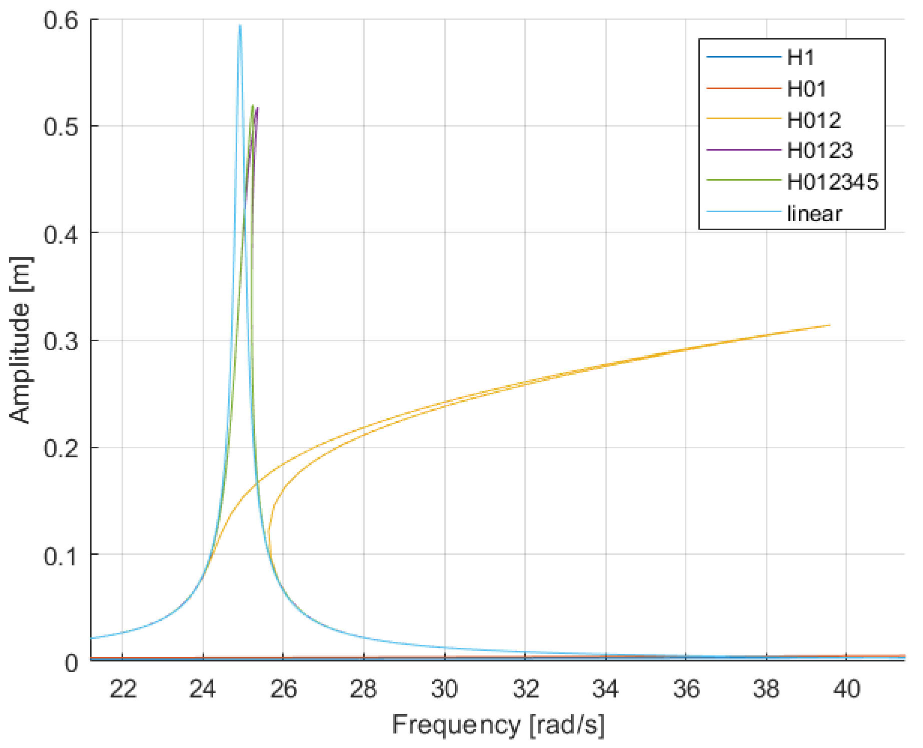


Figure A.5: Cantilever FRCs for various sets of harmonics. Highlight of the difference between harmonics 012 and 0123.

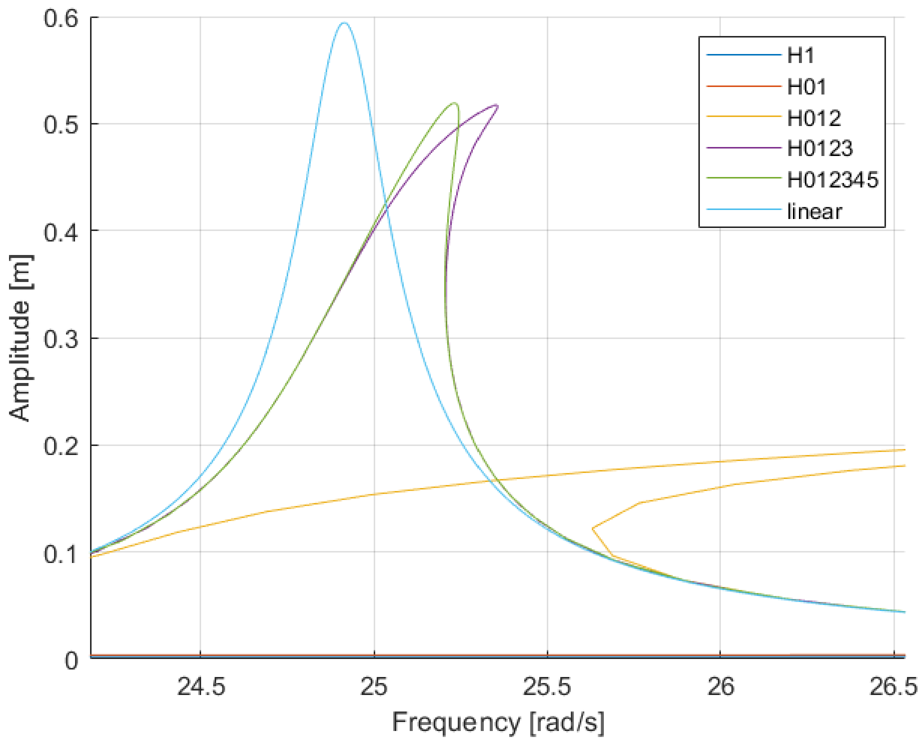


Figure A.6: Cantilever FRCs for various sets of harmonics. Highlight of the difference between harmonics 0123 and 012345.

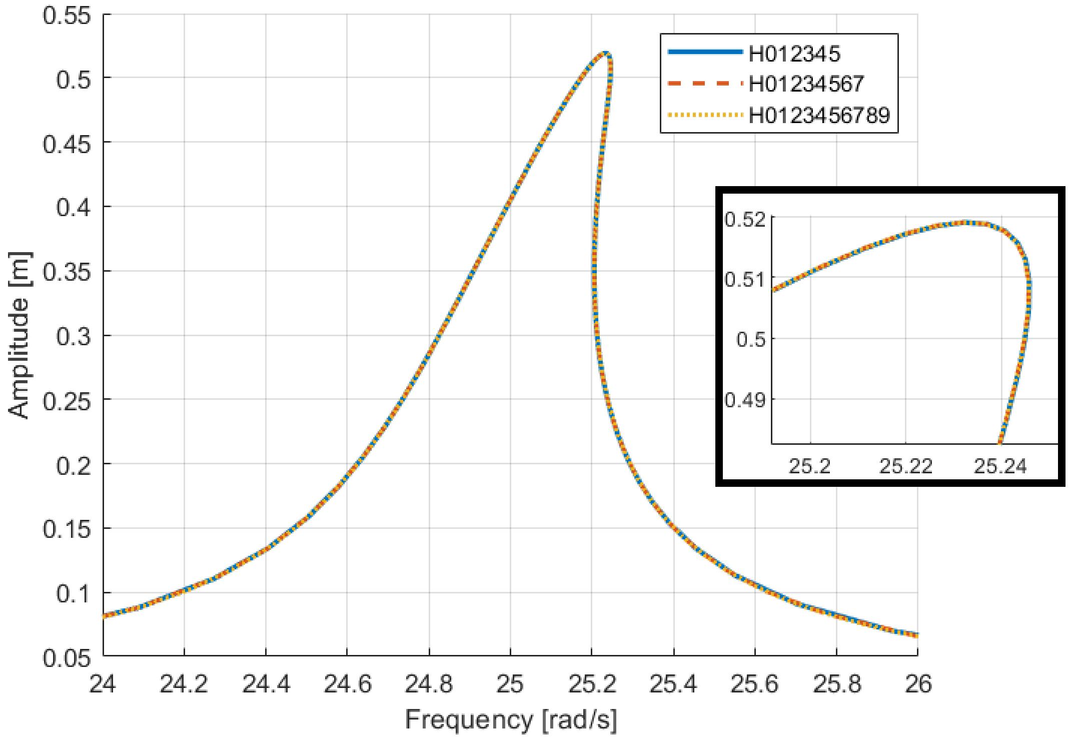


Figure A.7: Cantilever FRCs for the 3 highest harmonic counts - H012345, H01234567 and H0123456789.

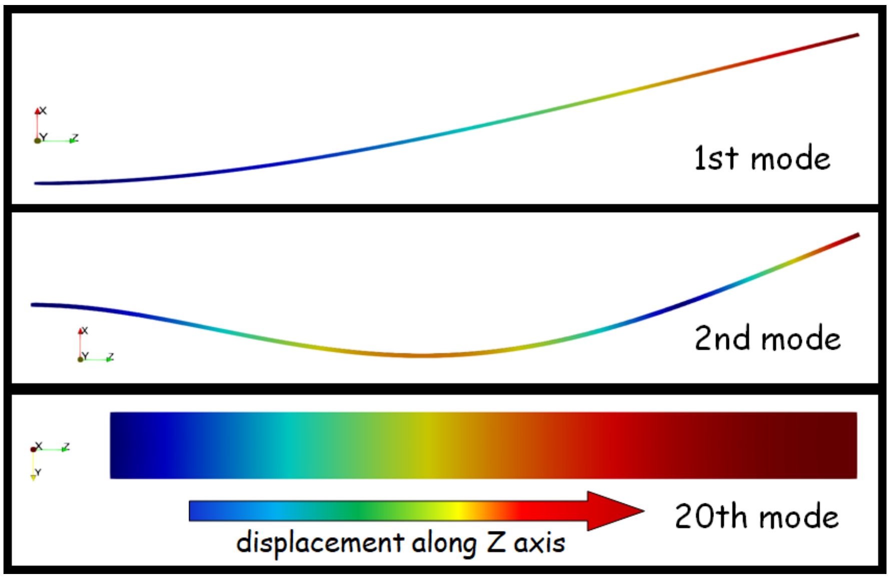


Figure A.8: Cantilever beam modes - 1st, 2nd and 20th.

First, a comparison of projection on mode 2 between harmonic cases 012 and 0123 is shown in Figures A.9 and A.10. The second comparison is on mode 20 between harmonic cases 0123 and 012345 shown in Figures A.11 and A.12. The corresponding $q_{i,k}$ values are plotted for all harmonics. The last figure (Figure A.12) does not have uniform Y axis scaling. It additionally includes a 10% line marking 10% of the maximum amplitude and markers where that threshold is crossed by the FRC on each harmonic.

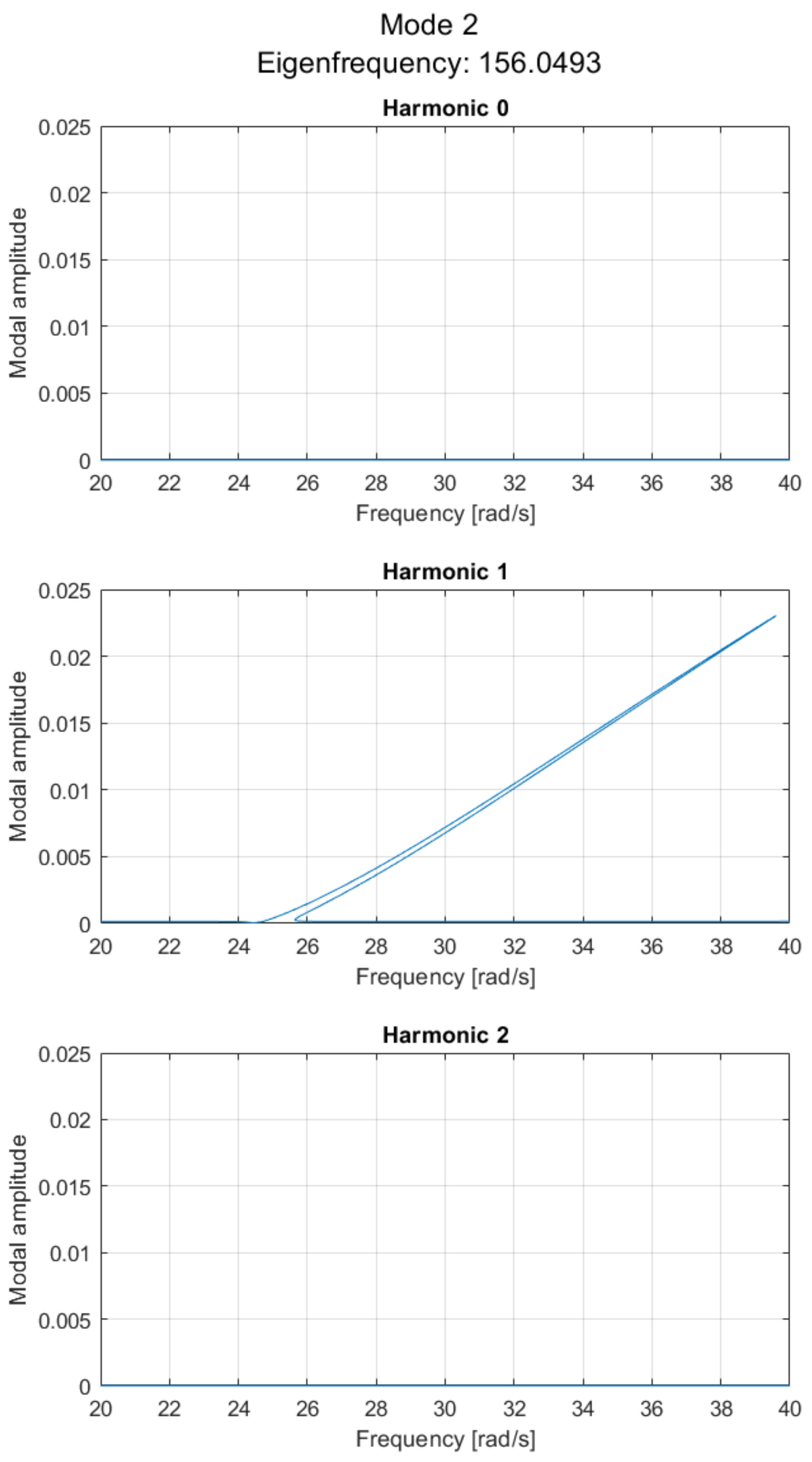


Figure A.9: Modal amplitudes of the 2nd mode for individual harmonics for the H012 testcase.

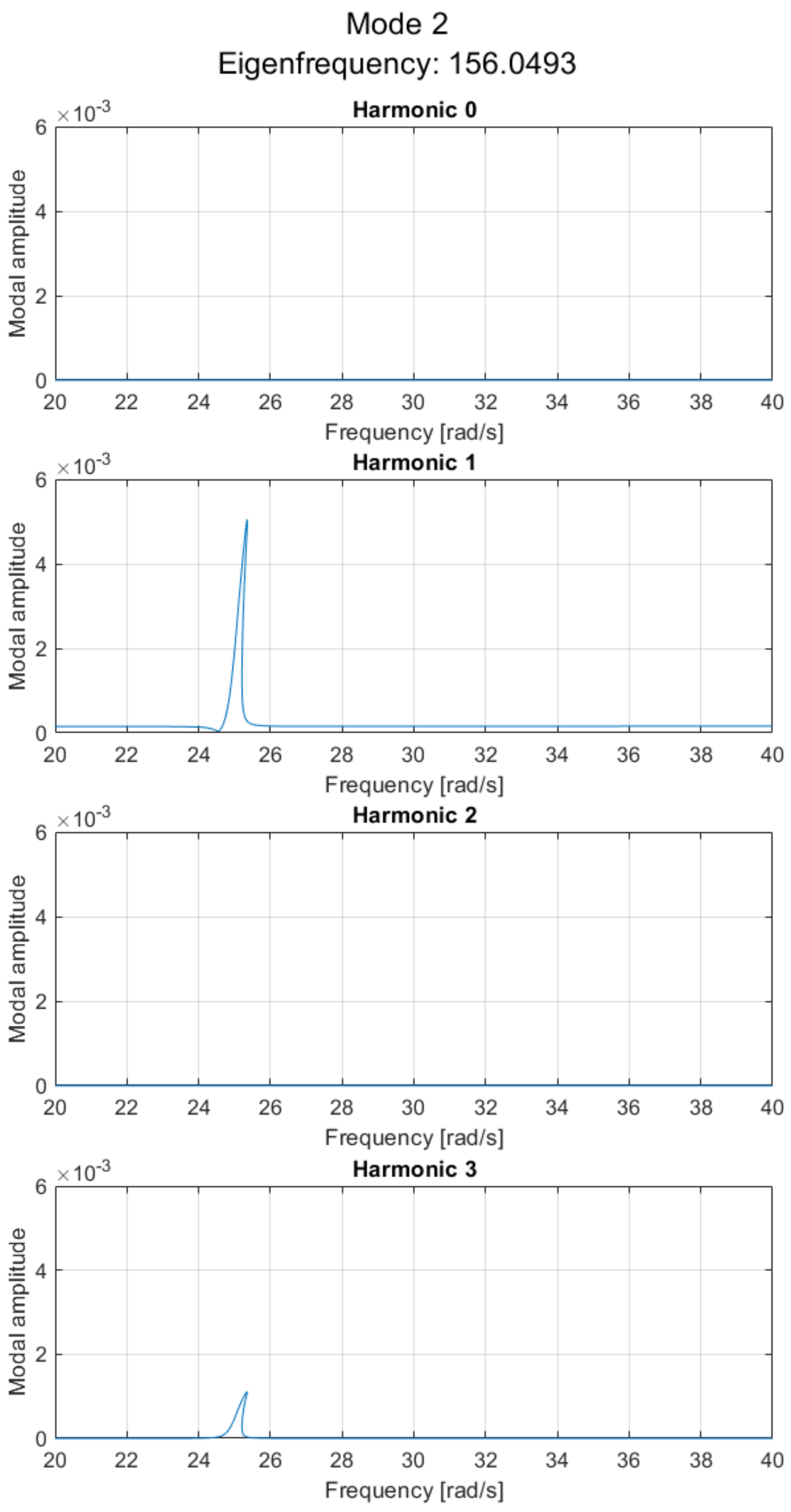


Figure A.10: Modal amplitudes of the 2nd mode for individual harmonics for the H0123 testcase.

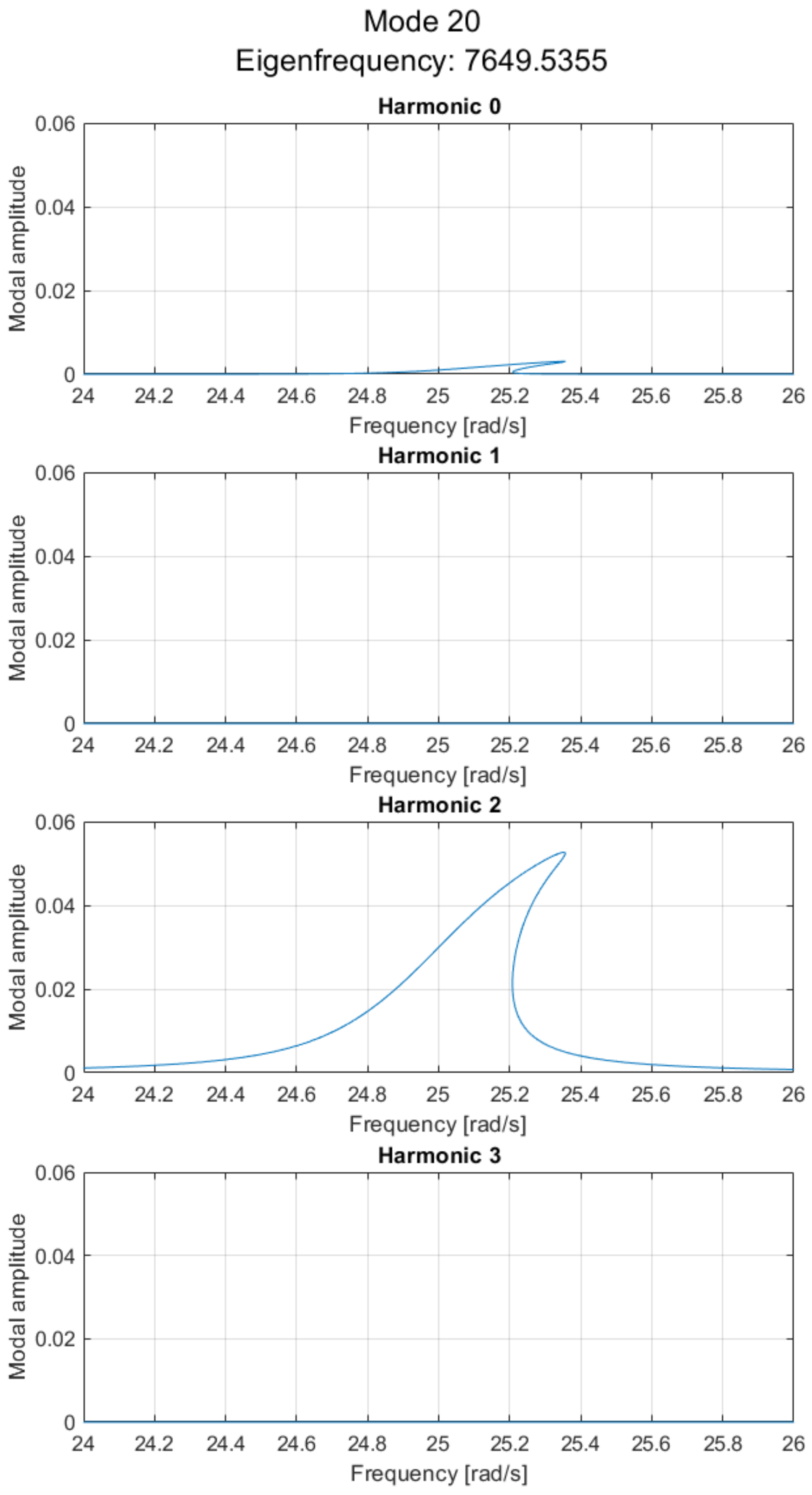


Figure A.11: Modal amplitudes of the 20th mode for individual harmonics for the H0123 testcase.

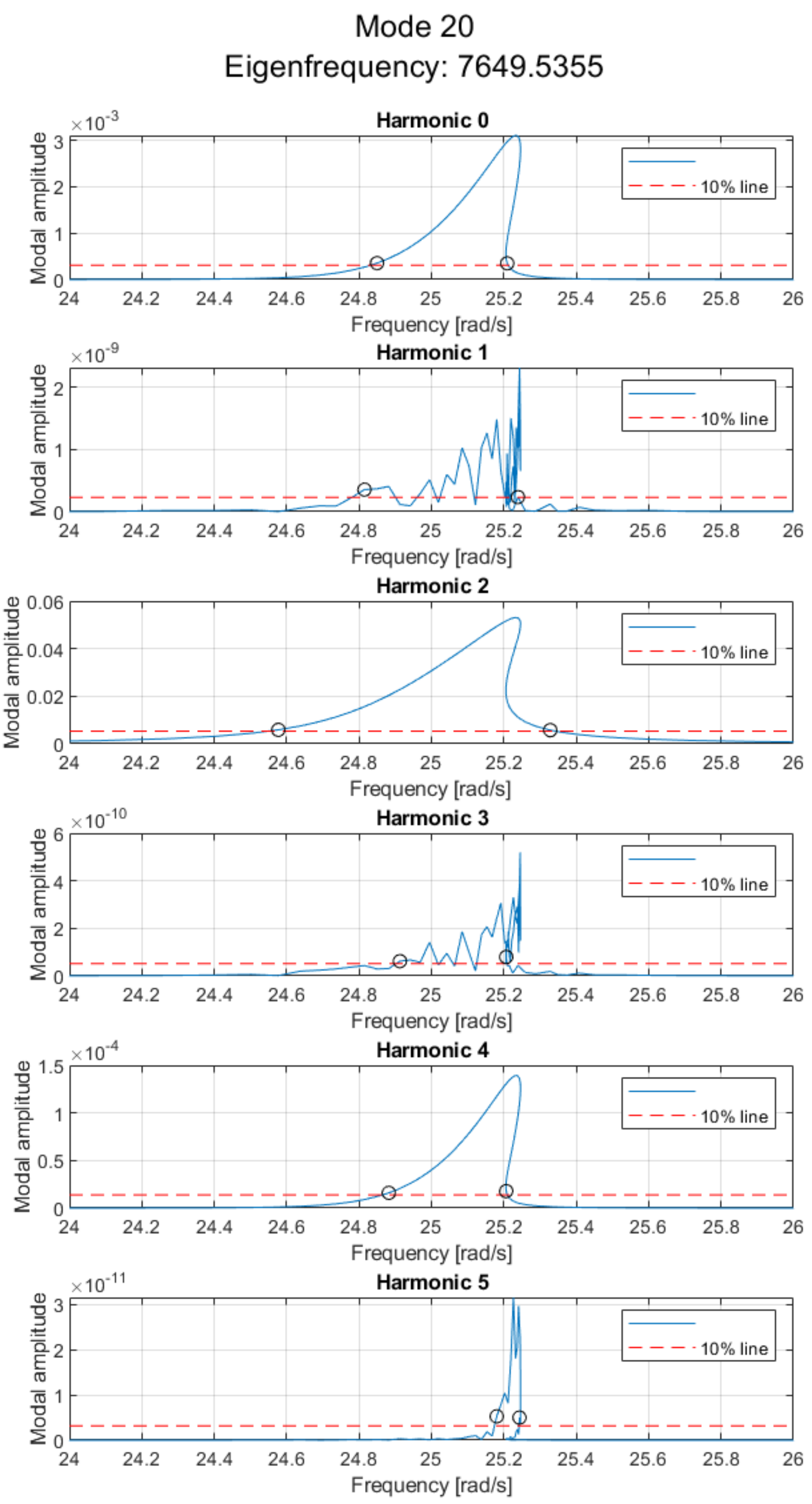


Figure A.12: Modal amplitudes of the 20th mode for individual harmonics for the H012345 testcase. The red line marks the 10% amplitude level with respect to the peak value. The black circles mark where that red line is crossed by the FRC.

A.0.1 Discussion

The first and most straightforward observation that can be made is that the set of used harmonics has significant impact on the shape of the FRCs. In particular, using only harmonic 1 or harmonics 01 yields completely different results from the rest of the cases (Figures A.3 and A.4). It should also be noted that the addition of the 0th harmonic on its own makes a large impact and therefore it was present for all the other cases. It can also be seen that adding more harmonics is only meaningful up to a certain point. As seen in Figure A.7, going beyond the 5th harmonic seems to make no further difference on the FRC. This study therefore focuses on differences between cases up to the H012345 case.

The next case to study is the H012 one. As can be seen in Figure A.5, this FRC already follows what is assumed to be the correct FRC (based on the higher harmonic cases), but only to a certain amplitude. When the amplitude reaches values around 0.1 metres, the H012 FRC starts to strongly deviate, exhibiting a significant stiffening effect. This stiffening effect can be likened to the one observed in the fan blade testcase as seen in Figure 6.14, even though in the fan blade case the curve turns more sharply.

The next point of interest is the difference between FRCs for cases H0123 and H012345. The H0123 case follows the higher harmonic FRC much better compared to the H012 case. The ratio between the frequencies at the FRC peaks for the H0123 and H012345 cases is 1.0047, meaning that there is only 0.47% difference between the curves at their peak values. However, a clear difference in the curves can still be observed visually. Starting at amplitude of around 0.35 metres, the H0123 curves departs to the right, exhibiting again a stronger stiffening effect compared to its H012345 counterpart.

The analysis of modal projections follows next. The presence of the geometric nonlinearity in the model introduces couplings between different modes. The problematic of modal coupling was studied for example in [127, 178]. The 20th mode, which is a longitudinal mode, is quadratically coupled with the first (bending) mode. This coupling can be seen in Figures A.11 and A.12. Figure A.10 shows how significant is the contribution of the 20th mode on the 2nd harmonic. This explains why the H1 and H01 cases, which don't include this, deviate so sharply. Additionally, the importance of quadratic couplings generally becomes significant at lower amplitudes compared to cubic couplings. Figure A.12 then shows that besides the 2nd harmonic the 4th harmonic is also excited, albeit with a much lower amplitude. The circle marks in this plot show that the influence of the 2nd harmonic is a lot broader in terms of frequency range compared to the 4th harmonic which has its influence on the complete FRC more centralised around the resonance frequency.

The second comparison was made on the 2nd mode. This mode, which is the second bending mode, is known to be cubically coupled with the first mode. As mentioned previously, this coupling generally grows in significance later in terms of the amplitude. Figures A.9 and A.10 show projections on this mode for the H012 and H0123 cases respectively. Figure A.10 shows that the 3rd harmonic's contribution for this mode is quite significant, with its amplitude being only about 5$\times$ lower than of the 1st harmonic.

In conclusion, this study shows the importance that various higher harmonics have in relation to coupling between various modes. Neglecting these higher harmonics in the model can introduce artificial stiffening effects to the results as the structure is effectively not allowed to undergo certain types of motion. Noting that this level of complexity is already present in this relatively simple cantilever beam testcase, it not surprising that a more complex geometry like the fan blade exhibits a similar behaviour. The shape and curvature of the blade can lead to many significant coupling terms between various modes that require use of higher harmonics to be captured properly.