Lehrstuhl für Angewandte Mechanik
Technische Universität München

Lukas Scheucher, B.Sc.

Development and Analysis of Multi Preconditioned FETI Methods

Term Paper

19. 09. 2016

Supervisor:
Michael Leistner, Dipl. Ing.
Prof. Dr. Daniel J. Rixen

Contents

List of Figures

Nomenclature

Representation of scalars. tensors and other quantities

q, Q	Scalar quantity
$\vec{q}$	Continuous vectorial quantity
Q	Continuous matrix quantity
$\mathbf{q}$	Discrete vectorial quantity
$\mathbf{Q}$	Discrete matrix quantity

Operators and symbols

$(\cdot)^T$	Transpose of a tensor
$(\cdot)^{-1}$	Inverse of a tensor or mapping
$(\cdot)^+$	Pseudoinverse of a tensor or mapping
$\dot{(\cdot)}, \ddot{(\cdot)}$	First and second time derivative at a fixed reference position
det	Determinant
div	Spatial divergence operator
∂	Virtual quantity

FE space discretization

nele	Number of elements
nnod	Number of nodes
ndim	Number of spatial dimensions
ndof	Number of degrees of freedom
N_{k}	FE shape function of node k
$\vec{\xi}$	Position in FE parameter space
$\vec{\mathbf{X}}$	Discrete nodal positions in reference configuration
$\vec{\mathbf{x}}$	Discrete nodal positions in current configuration
$\vec{\mathbf{D}}$	Discrete nodal displacements in reference configuration
$\vec{\mathbf{d}}$	Discrete nodal displacements in current configuration
$\vec{\mathbf{X}}$	Discrete nodal positions in reference configuration
h	Characteristic element size
p	Polynomial degree of finite element interpolation

Domain decomposition

Ω_s	Domain with index s
Γ_s	Interface with index s
$(\cdot)^{(s)}$	Quantity related to domain s

FETI-Method

General

$(\cdot)^{(s)}$	substructure local quantity
$(\cdot)^{[s]}$	assembled substructure local quantity
λ	Lagrange multiplier (can be interpreted as interface force)
$\boldsymbol{\lambda}$	Vector of Lagrange multipliers
$\boldsymbol{\lambda}_{\mathbf{N}}$	Vector of Lagrange multipliers in the natural subspace
$\boldsymbol{\lambda}_{\mathbf{C}}$	Vector of Lagrange multipliers in the auxiliary(coarse) space
$\boldsymbol{\lambda}_{\mathbf{F}}$	Vector of Lagrange multipliers in the conjugate gradient iteration space
$\boldsymbol{\alpha}$	Generalized coordinates of rigid body modes
f	External force (one component)
$\mathbf{f}$	Vector of external forces (dof-based)
$\mathbf{d}$	Interface displacements
r	Displacement gap(residuum)
$\mathbf{r}$	Vector of residua
$\mathbf{t}$	Trace operator(3.7)
$\mathbf{B}$	Boolean assembly operator (3.7)
$\mathbf{R}$	Rigid body modes
$\mathbf{S}$	Local Schur complement (3.14)
$\mathbf{F}$	Local Dual Schur complement
$\mathbf{G}$	Constraint matrix
$\mathbf{A}$	Scaling matrix
$\mathbf{C}$	Auxiliary Coarse space
$\mathbf{P}$	Projector to natural coarse space
$\mathbf{P}_c$	Projector to auxiliary coarse space
$\mathcal{K}$	Condition number

FETI-2

$\mathbf{q}$	Search direction in natural subspace
$\mathbf{z}$	Search direction in conjugate gradient iteration space

FETI-S

$\mathbf{Q}$	Block of search directions in natural subspace
$\mathbf{W}$	Search directions in natural subspace
$\mathbf{Z}$	Block of search directions in conjugate gradient iteration space

FETI-AS

τ	Adaptivity criterion
t_i^s	Convergence Indicator
ρ	Contraction factor

Abbreviations

DD	Domain Decomposition
BDD	Balanced Domain Decomposition
FETI	Finite Element Tearing and Interconnecting
FETI-1	One-level FETI algorithm
FETI-2	Two-level FETI algorithm
FETI-S	Simultaneous FETI algorithm
FETI-B	Block FETI algorithm
FETI-AS	Adaptive Simultaneous FETI algorithm
FETI-FAS	Fast Adaptive Simultaneous FETI algorithm
FE	Finite element
FEM	Finite element method
GP	Gauss point
IBVP	Initial boundary value problem
LM	Lagrange multiplier
NC	Non-conforming
ODE	Ordinary differential equation
PMTPE	Principle of minimum of total energy
PVW	Principle of virtual work

Chapter 1

Introduction

1.1 Motivation

The field of high performance computing has seen a significant paradigm shift over the past 10-15 years. While Moore's law [14] is still valid to this day, the way progress is achieved has drastically changed. Single-core performance has been stagnating for years, which gave rise to the advent of multi-core processors. But as the architectures are changing, so do the algorithms. Amdahl's law[19] poses harsh requirements on efficient parallel algorithms. In order to reach satisfying scalability, inter-processor communication, as well as serial algorithm parts, have to be kept at an absolute minimum. An intuitive way of approaching that goal is domain decomposition(DD). In particular, the Finite Element Tearing and Interconnection (FETI) or the Balanced Domain Decomposition (BDD) are well-established methods for mechanical problems. The basic idea of DD methods is to split the domain into, ideally, equal-sized subdomains(substructures). Local Problems can then be solved for each substructure simultaneously on multiple cores sequentially, before an iterative technique is required to connect the domains together by finding the common interface unknowns.
Generally, primal and dual DD algorithms can be differentiated by the type of interface quantity that are used for the connection. Primal DD methods use interface displacements, whereas dual DD methods solve for the interface forces. A profound overview over these methods can be found in [8].
Unfortunately, classical DD shows very bad behaviour for cases with irregular-shaped substructures as well as cases with jumps in the material parameters along, as well as across the interface. For a general purpose algorithm, useful for real engineering problems, this has to be addressed. Several approaches have been proposed, based on scaling of the operators [25] or by solving for critical eigenmodes on the interface [21, 23].
For FETI-type solvers, a promising algorithm, the Simultaneous FETI (FETI-S), has been proposed in [16] for two sub-domains and was generalized for an arbitrary number of sub-domains in [17]. An efficient implementation of this algorithm, as well as the Block FETI method (FETI-B) has recently been described [9].
A recent publication [22] has proposed a new, Adaptive Multi Preconditioned Conjugate Gradient Algorithm (AMPCG) for cases where the preconditioner is built as a sum of contributions (like in FETI). A first application to the Balancing Domain Decomposition method has shown promising results.

1.2 Objective

This thesis aims at a profound analysis of the before mentioned, existing FETI algorithms. Their derivations will be described and theoretical aspects are going to be discussed. Intensive numerical assessments will highlight the potentials and weak points of each method and will be used as basis for the final conclusions.
Additionally, the before mentioned AMPCG algorithm will be applied on the FETI method and compared to the existing algorithms.
Finally, an outlook regarding further developments of the FETI methods will be drawn, and promising ideas will be discussed.

Chapter 2

Governing equations and finite element formulations

The following section provides a short summary of the most important equations of solid mechanics in the finite element context. It thus serves as a short preparation for Chapter 3, which will will enhanced the basic formulation by a domain decomposition approach. A comprehensive declaration of all terms and expressions used in this thesis can also be found in the nomenclature. Moreover, since domain decomposition typically involves some kind of constraint enforcement, the most important methods in a finite element context are highlighted.

2.1 Solid mechanics

The classical displacement based problem in linear continuum mechanics can be described by a set of three governing equations. The balance equation(BE) describes the relationship between the stress resultant and the external forces, the constitutive equation(CE) provides information about the material behaviour and finally, the kinematic equation (KE) links displacement and strains. The basic equations can be very descriptively linked in the Tonti diagram[24], as provided in Figure 2.1.

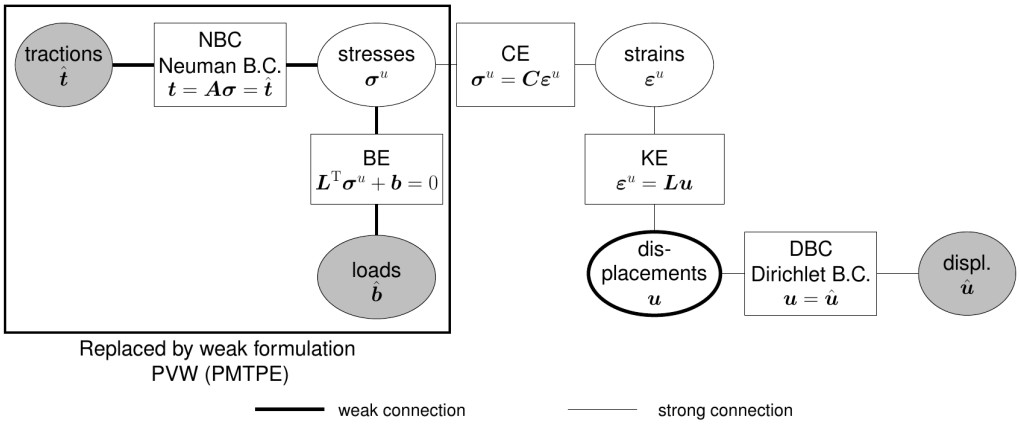


Figure 2.1: *Tonti diagramm for a linear, small displacement elasticity problem. It classifies variables and equations of the solid mechanics PDEs and helps to derive an appropriate finite element formulation. A classic, displacement based formulation is depicted here.*

In index-notation, the system of equations to be solved reads

$$\frac{\partial}{\partial x_i}\sigma_{ij} + \bar{X}_j = 0 \qquad in\ \Omega_0 \tag{2.1}$$

$$t_j = n_i \sigma_{ij} = \bar{t}_j \qquad \textit{on Neumann boundary} \tag{2.2}$$

$$\sigma_{ij} = \sigma_{ji} \tag{2.3}$$

$$\sigma_{ij} = C_{ijkl} \epsilon_{kl} \tag{2.4}$$

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}\right) \tag{2.5}$$

$$u_j = \bar{u}_j \qquad \textit{on Dirichlet boundary} \tag{2.6}$$

A system of equations is derived, that can, except for special, very simple cases, not be solved analytically. The finite element method now entails two key ideas. Firstly, instead of a continuous domain, considerations are focused on a finite number of discrete elements("finite elements"). Each Element defines its own function space for the primary variables via the so called shape functions. It is obvious that the thereby introduced limitation of the solution function space will generally not contain the solution to the original problem. Therefore, as a second step, selected governing equations are "weakened", i.e. they are multiplied by a weighting function and only enforced in an integral sense. For the formulation, presented in Figure 2.1 the weak form can be formulated as

$$\int_\Omega \underbrace{(\nabla \cdot \boldsymbol{\sigma}^u + \hat{\mathbf{b}})}_{\vec{R}_{\mathrm{BE}}} \cdot \underbrace{\vec{w}}_{\substack{\text{weighting}\\ \text{function}}} \, d\Omega + \int_{\Gamma_\sigma} \underbrace{(\nabla \cdot \boldsymbol{\sigma}^u + \hat{\mathbf{b}})}_{\vec{R}_{\mathrm{FBC}}} \cdot \underbrace{\vec{w}}_{\substack{\text{weighting}\\ \text{function}}} \, d\Gamma = 0 \, . \tag{2.7}$$

Gauss divergence divergence is subsequently applied for further derivations.

Principle of Virtual Work Generally, the weight function $\vec{w}$ is arbitrary, and can be chosen freely. If, however, one makes the particular choice of

$$w_{\mathrm{i}} = \partial u_{\mathrm{i}} \tag{2.8}$$

work expressions are obtained for the individual terms and the Principle of Virtual Work (PVW) is derived as (in matrix notation)

$$\int_\Omega \partial \boldsymbol{\epsilon}^{u\mathrm{T}} \boldsymbol{\sigma}^u \, d\Omega = \int_\Omega \partial u^{\mathrm{T}} \, d\Omega + \int_{\Gamma_\sigma} \partial u^{\mathrm{T}} \mathbf{t} \, d\Gamma \, . \tag{2.9}$$

The PVW is valid for arbitrary materials, also for dissipative ones that have no potential. The finite element formulation is finally derived expressing all terms of the PVW exclusively in the primary variables. To do so, the strong connections of the Tonti diagram are taken advantage of.

2.2 Space discretization

The FEM in solid mechanics is based upon the idea of finding a numerical solution to Equation (**??**) at discrete points, commonly referred to as nodes. Connected nodes form elements, into which the problem domain is partitioned into: The primary variable on element e is then typically approximated by interpolation functions (shape functions).
In this case, the displacement field has been used as the only primary variable, and a local interpolation approach, i.e., the shape functions only have local support, has been utilized.
Therefore:

$$\Omega_0 \approx \cup_{\mathrm{e}=1}^{nele} \Omega_0^{(e)} \, , \tag{2.10}$$

$$u(\vec{X}, t) \approx u_{\mathrm{h}}^{(e)}(\vec{X}, t) = \sum_{\mathrm{k}=1}^{nnod^{(e)}} N_{\mathrm{k}}(\vec{X}) d_{\mathrm{k}}(t) \, . \tag{2.11}$$

$$\vec{x}(\vec{X},t) \approx \vec{x}_{\mathrm{h}}^{(e)}(\vec{X},t) = \sum_{\mathrm{k}=1}^{nnod^{(e)}} N_{\mathrm{k}}(\vec{X}) x_{\mathrm{k}}(t)\ . \tag{2.12}$$

2.3 Methods of constraint enforcement

From a mathematical point of view, coupling problems are basically optimization problems with equality constraints. A problem of this kind can generally be formulated as:

$$\min_{x} f(\vec{x})\ , \tag{2.13}$$

$$\text{subjected to } g(\vec{x}) = 0\ . \tag{2.14}$$

Numerous methods have been proposed to solve this kind of problems. Among the most common ones are the Lagrange multiplier approach and the penalty method. This thesis focuses solemnly on the first on, nevertheless a quick introduction into the alternatives shall be provided, so that the pros and cons of each method can be discussed.

Lagrange multiplier approach

The Lagrange multiplier method demands the existence of continuous partial derivatives for both: $\vec{f}$ and $\vec{g}$. It introduces a new variable λ, called Lagrange multiplier, and considers the Lagrange function defined as

$$\Lambda(\vec{x},\vec{\lambda}) = f(\vec{x}) + \lambda \cdot g(\vec{x})\ . \tag{2.15}$$

Solutions of the original problem are stationary points of the Lagrange function. However, not all stationary points yield a solution of the original problem. The method of Lagrange multipliers thus gives a necessary condition. Sufficient conditions can be obtained by surveying the Hessian matrix. Details can be found in any standard mathematical literature concerning optimization theory. The main advantage of the Lagrange multiplier approach lies in its exact fulfilment of the constraints. Moreover, it is a single step method, meaning that no iterative procedure is necessary to fulfil the boundary conditions.
The Lagrange multiplier approach does, however, come with the disadvantage of introducing additional unknowns, as well as worsening the overall system of equations constitution.

Penalty approach

A penalty method replaces the constraint problem by a series of unconstrained problems, such that their solutions ideally converge to the solution of the original problem. Basically the series of unconstrained minimization problems can be written as:

$$\min \Phi(\vec{x}) = f(\vec{x}) + \sigma_{\mathrm{k}} \sum_{\mathrm{i}\in\mathrm{I}} g(c_{\mathrm{i}}(\vec{x})),\ \text{where} \tag{2.16}$$

$$g(c_{\mathrm{i}}(\vec{x})) = \min(0, ci(x))^2\ . \tag{2.17}$$

They main advantage of the penalty method lies in the fact, that it does not introduce any new unknowns, and thus maintains the positive definiteness of the system matrix. On the other hand, the constraints are not fulfilled exactly. Regarding the choice of σ_{k} one has to weight carefully between a more accurate constraint fulfilment and the overall systems constitution since a greater σ_k value might enforce the constraints stronger, but can also lead to numerical instabilities.

Augmented Lagrange

The Augmented Lagrange method has similarities to the Penalty method. It replaces the constraint optimization problem by a series of unconstrained problems and adds a penalty term to the objective. Moreover, the Augmented Lagrange Method adds yet another term, that mimics a Lagrange multiplier.
Starting from the problem defined in Equation (2.13), the approach uses the following unconstrained objective:

$$\min \boldsymbol{\phi}_{\mathrm{k}}(\vec{x}) = f(\vec{x}) + \frac{\mu_{\mathrm{k}}}{2} \sum_{i \in I} g(c_i)(\vec{x})^2 - \sum_{i \in I} \lambda_i g(c_i)(\vec{x}) \, . \tag{2.18}$$

After each iteration, not only μ_k, but also the variable λ is updated according to the rule

$$\lambda_i \leftarrow \lambda_i - \mu_k c_i(\vec{x}_k) \, , \tag{2.19}$$

where $\vec{x}_k$ denotes the solution to the unconstrained problem at the k-th step.
The variable λ can be regarded as an estimate of the Lagrange multiplier, as introduced in Section **??**.
Contrary to the Penalty approach, $\mu \to \infty$ is not required in order to solve the original, unconstrained problem, thanks to the presence of the Lagrange multiplier term. The method is therefore especially favourable due to both, an exact fulfilment of the boundary conditions and no introduction of new Unknowns. It also avoids the ill conditioning of the standard quadratic penalty method.

Nitsche method

The Nitsche Method is closely related to the development of discontinuous Galerkin methods. It has originally been developed as a simple approach to handle Dirichlet boundary conditions, but its real strength lies in the generality with which interface problems can be handled: arbitrary degree of polynomial approximations, arbitrary(shape regular meshes), even different physical models on either side are possible [10].
A full introduction is beyond the scope of this thesis, thorough investigations are provided in [15] and [10].

Chapter 3

The FETI formulation

3.1 Problem Setup

The reference problem for all further considerations is described in Figure 3.1. For the sake of simplicity, a two domain setup shall be considered here, a generalization to an arbitrary number of sub-domains can, however easily be derived.

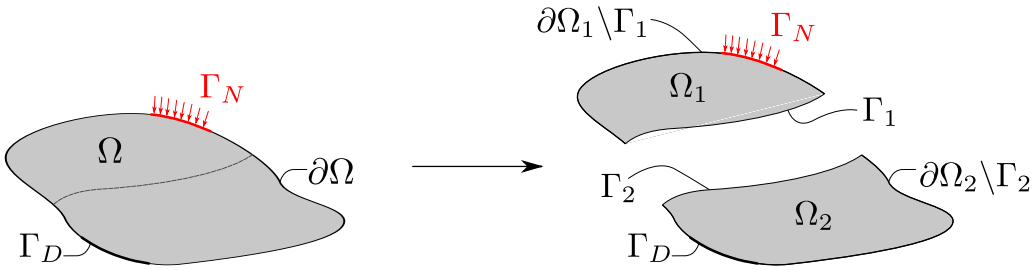


Figure 3.1: *Reference problem. The simple case of two substructures is considered here, a generalization to an arbirtray number of substructures can, however, be easily achieved. The domain Ω is split into the subdomains Ω_1 and Ω_2 by the interface Γ. The aim of all DD methods is to distribute those domains to different proccessing nodes and solve the problems as locally as possible. Of course, some interprocessor communication can not be avoided, since information(e.g. the boundary conditions) has to be propagated from one domain to the other. Looking at this example it shall also be noted, that substructuring may introduce so-called "floating substructures", which means substructures can have rigid body modes, although the overall system does not.*

3.2 General

The simple problem of linear elasticity can be formulated in Ω as:

$$\mathbf{Ku} = \mathbf{f} \tag{3.1}$$

The FETI-method tears the domain apart, introducing two sub-domains. Thus, a compatibility condition has to be imposed, linking them together. The FETI method enforces this compatibility by the Lagrange multiplier approach(see Section 2.3). One can thus write:

$$\mathbf{K}^{(1)}\mathbf{u}^{(1)} = \mathbf{f}^{(1)} \tag{3.2}$$

$$\mathbf{K}^{(2)}\mathbf{u}^{(2)} = \mathbf{f}^{(2)} \tag{3.3}$$

$$(\lambda, \mathbf{u^{(2)}} - \mathbf{u^{(1)}}) = 0 \tag{3.4}$$

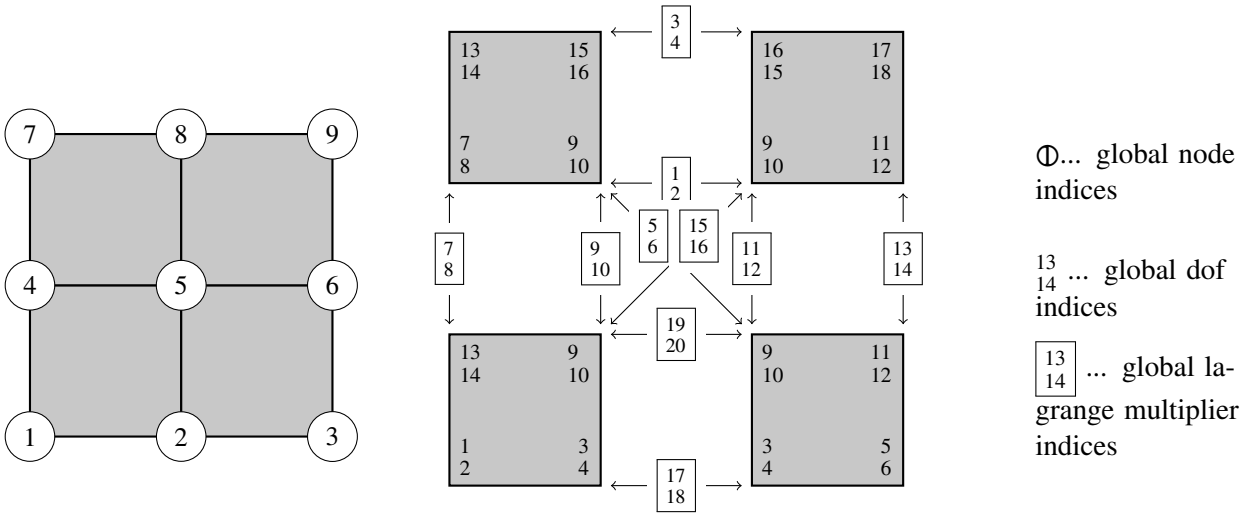


Figure 3.2: *Basic discretized problem. The setup consists of four substructures, each of which is built up of just one quadrilateral element. Albeit very simple, this example can be used to explain all relevant operators and problems concerning the FETI method. The element/node/dof/Lagrange-multiplier numbering scheme is consistent with the Matlab code developed for this thesis [11]. Its is very crucial to note the difference between the global numbering scheme and the substructure-based numbering scheme.*

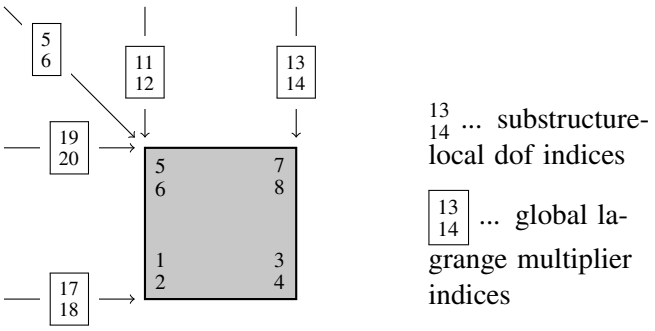


Figure 3.3: *Detailed view of the bottom right substructure. Substructure-local dof indices are printed here, since they are used to create the assembly operator* $\mathbf{B}$*. For this particular example, the assembly and trace operator of the bottom right substructure are given in Equation* (3.7).

In contrast to e.g. the mortar method, the FETI approach, enforces Equation (3.4) not in an integral, but in a point-wise sense. This does not introduce errors, since only conforming meshes are considered here.
For an arbitrary number of sub-domains Equations (3.2)-(3.4) can be equivalently formulated as

$$\mathbf{K}^{(s)}\mathbf{u}^{(s)} = \mathbf{f}^{(s)} + \mathbf{t}^{(s)^T}\mathbf{B}^{(s)^T}\lambda \tag{3.5}$$

$$\sum_s \mathbf{t}^{(s)}\mathbf{B}^{(s)}\mathbf{u}^{(s)} = 0 \tag{3.6}$$

where $\mathbf{B}^{(s)}$ is a substructure-based, boolean assembly operator, who's rows correspond to the global Lagrange multipliers, and who's columns relate to the substructure-based interface-dofs. A negative sign is used, when the Lagrange multiplier connects to a substructure with a smaller ID, otherwise a positive sign is entered. This reflects the condition from Equation (3.4).
The trace operators $\mathbf{t}^{(s)}$, map the columns of $\mathbf{B}^{(s)}$ to a substructure-based total dof numbering scheme.
For the basic example as described in Figure 3.2 the operators $\mathbf{B}$ and $\mathbf{t}$ for the bottom left substructure(isolated in Figure 3.3) are given as

$$B^{(3)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \\ 16 \\ 17 \\ 18 \\ 19 \\ 20 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}, \quad \mathbf{t}^{(3)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \tag{3.7}$$

The basic idea of FETI now is to reduce the substructure domain problems to a connected interface problem. This can be achieved very elegantly, by rearranging the basic equations (3.5)-(3.6) and was first described in [7]. One should note, that a solution to Equation (3.5) can only exist if $(\mathbf{f}^{(s)} + \mathbf{t}^{(s)^T}\mathbf{B}^{(s)^T}\lambda)$ is orthogonal to the null-space of $\mathbf{K}$.

3.3 Interface reduction

As mentioned FETI is a reduction technique that solves for the interface unknowns only. After the interface solution has been obtained, the global solution will be recovered.
This section is denoted to an step by step explanation of the reduction and subsequent recovery process. We will end up with a system of Equations in the interface unknowns(ans possibly some rigid body modes) only. The process of solving that system will be discussed in Section 5.

We begin considerations by a reformulation of Equation(3.5) as:

$$\mathbf{u}^{(s)} = \mathbf{K}^{(s)^{-1}}(\mathbf{f}^{(s)} + \mathbf{t}^{(s)^T}\mathbf{B}^{(s)^T}\lambda) \tag{3.8}$$

However, as will be explained in Section 3.5, for static problems it is required to search for λ in the so-called natural subspace, which is is the rigid body mode free space. Thus,

$$\mathbf{u}^{(s)} = \mathbf{K}^{(s)^{-1}}(\mathbf{f}^{(s)} + \mathbf{t}^{(s)^T}\mathbf{B}^{(s)^T}\lambda) + \mathbf{R}^{(s)}\boldsymbol{\alpha}^{(s)} \tag{3.9}$$

where $\mathbf{R}^{(s)}$ denotes the nullspace of $\mathbf{K}^{(s)}$ and $\boldsymbol{\alpha}^{(s)}$ are the generalized coordinates for the interpolation. It should be noted that for the iterative solution process that will follow, a practical implementation will involve a matrix factorization anyway. The nullspace thus comes as a by-product for free.

Inserting Equation (3.9) into Equation (3.6) gives

$$\sum_s \mathbf{B}^{(s)}\mathbf{t}^{(s)}\left(\mathbf{K}^{(s)^{-1}}(\mathbf{f}^{(s)} + \mathbf{t}^{(s)^T}\mathbf{B}^{(s)^T}\lambda) + \mathbf{R}\boldsymbol{\alpha}^{(s)}\right) = 0 \text{ or} \tag{3.10}$$

$$\underbrace{\underbrace{(\sum_s \mathbf{B}^{(s)}\underbrace{\mathbf{t}^{(s)}\mathbf{K}^{(s)^{-1}}\mathbf{t}^{(s)^T}}_{\mathbf{F}^{(s)}}\mathbf{B}^{(s)^T})}_{\mathbf{F}^{[s]}}\lambda + \sum_s \underbrace{\mathbf{B}^{(s)}\mathbf{t}^{(s)}\mathbf{R}^{(s)}}_{\mathbf{G}^{(s)}}\boldsymbol{\alpha}^{(s)}}_{\mathbf{F}} = \underbrace{\sum_s \mathbf{B}^{(s)}\mathbf{t}^{(s)}\mathbf{K}^{(s)^{-1}}\mathbf{f}^{(s)}}_{-\mathbf{d}} \tag{3.11}$$

where the abbreviations $\mathbf{F}^{(s)}$ for the local dual schur complement, $\mathbf{F}^{[s]}$ for the assembled local dual schur complement, $\mathbf{G}$ for the rigid body mode space and $\mathbf{d}$ for the condensed interface displacements have been introduced.

So far, the system is under-determined, since we only have two equations, but three unknowns. However, a third equation can be derived by looking at Equation 3.5. If $\mathbf{K}^{(s)}$ is singular, a solution can only exist if $(\mathbf{f}^{(s)} + \mathbf{t}^{(s)^T}\mathbf{B}^{(s)^T}\lambda)$ has no component in the Nullspace $\mathbf{R}$:

$$\mathbf{R}^{(s)^T}(\mathbf{f}^{(s)} + \mathbf{t}^{(s)^T}\mathbf{B}^{(s)^T}\lambda) = 0 \text{ or} \tag{3.12}$$

$$\underbrace{\mathbf{R}^{(s)^T}\mathbf{f}^{(s)}}_{\mathbf{e}^{(s)^T}} = \underbrace{\mathbf{R}^{(s)^T}\mathbf{t}^{(s)^T}\mathbf{B}^{(s)^T}}_{\mathbf{G}^{(s)^T}}\lambda \tag{3.13}$$

where the abbreviation $\mathbf{e}$ for the components of the external forces in the rigid body mode space has been established.
Moreover, we can introduce

$$\mathbf{S}^{(s)} = \mathbf{K}_{bb}{}^{(s)} - \mathbf{K}_{bi}{}^{(s)}\mathbf{K}_{ii}{}^{(s)^+}\mathbf{K}_{ib}{}^{(s)} \tag{3.14}$$

$$\mathbf{S}^{[s]} = \mathbf{B}^T\left(\mathbf{S}^{(s)}\right)\mathbf{B} \tag{3.15}$$

$$\mathbf{S} = \sum_s \mathbf{S}^{[s]} \tag{3.16}$$

as the local schur operator, assembled local schur operator and global schur operator respectively.
Throughout this thesis $(\cdot)^{(s)}$ will be used for substructure local quantities and $(\cdot)^{[s]} = \sum_s \mathbf{B}^{(s)}\mathbf{t}^{(s)}(\cdot)^{(s)}\mathbf{t}^{(s)^T}\mathbf{B}^{(s)^T}$ denotes the assembled local quantity.

3.4 Matrix notation

Putting all previously derived Equations together, one can finally write the basic FETI Problem in matrix notation as:

$$\begin{bmatrix} \mathbf{F} & \mathbf{G} \\ \mathbf{G}^T & \mathbf{0} \end{bmatrix}\begin{bmatrix} \lambda \\ \alpha \end{bmatrix} = \begin{bmatrix} \mathbf{d} \\ \mathbf{e} \end{bmatrix} \tag{3.17}$$

Where

$$\mathbf{e} = \left[\ldots, \mathbf{f}^{(s)^T}\mathbf{R}^{(s)}, \ldots\right]^T \tag{3.18}$$

$$\mathbf{G} = \left[\cdots, \mathbf{B}^{(s)}\mathbf{t}^{(s)}\mathbf{R}^{(s)}, \cdots\right] \tag{3.19}$$

$$\mathbf{F} = \sum_s \mathbf{B}^{(s)}\mathbf{F}^{(s)}\mathbf{B}^{(s)T} \tag{3.20}$$

$$\mathbf{d} = -\sum_s \mathbf{B}^{(s)}\mathbf{t}^{(s)}\mathbf{K}^{(s)-1}\mathbf{f}^{(s)} \tag{3.21}$$

and λ are the unknown interface forces, $\boldsymbol{\alpha}$ the rigid body mode components respectively.

3.5 The natural subspace

As was explained in (3.12), classical FETI formulation encounters a problem for static calculations if the local stiffness matrix is singular. In that case, we required

$$\mathbf{R}^{(s)T}(\mathbf{f}^{(s)} + \mathbf{t}^{(s)T}\mathbf{B}^{(s)T}\lambda) = \mathbf{0} \tag{3.22}$$

where $\mathbf{B}$ is the substructure based boolean assembly operator, introduced in Equation (3.5), consisting of $0, +1, -1$, defined such that the continuity conditions for each Lagrange multiplier is ensured(see Equation (3.4)) We had solved this problem by prescribing the

$$\mathbf{G}^T\boldsymbol{\Delta}_i = 0 \tag{3.23}$$

Inserting Equation (3.31) one can thus write

$$\mathbf{G}^T(\mathbf{d} - \mathbf{F}\lambda_i - \mathbf{G}\boldsymbol{\alpha}_i) = 0 \tag{3.24}$$

which can easily be reformulated to

$$\mathbf{G}^T \underbrace{(\mathbf{d} - \mathbf{F}\lambda_i)}_{\mathbf{r}_i} - \mathbf{G}^T\mathbf{G}\boldsymbol{\alpha}_i = 0 \tag{3.25}$$

Solved for $\boldsymbol{\alpha}_i$ this gives

$$\boldsymbol{\alpha}_i = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\mathbf{r}_i \tag{3.26}$$

Inserting Equation (3.26) into Equation (3.31) gives

$$\boldsymbol{\Delta}_i = \mathbf{r}_i - \mathbf{G}(\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\mathbf{r}_i \tag{3.27}$$

$$= \underbrace{(\mathbf{I} - \mathbf{G}(\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T)}_{\mathbf{P}}\mathbf{r}_i \tag{3.28}$$

where by looking at the equation, $\mathbf{P}$ can be interpreted as a projection operator, that removes all rigid body components from $\mathbf{r}_i$.
As first described in [5], and further investigated in [16] it is often advantageous to choose $\boldsymbol{\Delta}_i$ to be not simply orthogonal, but A-orthogonal to a matrix A:

$$\mathbf{G}^T\mathbf{A}\boldsymbol{\Delta}_i = 0 \tag{3.29}$$

which leads to the following expression for the natural subspace projector

$$\mathbf{P}^T = \mathbf{I} - \mathbf{A}\mathbf{G}(\mathbf{G}^T\mathbf{A}\mathbf{G})^{-1}\mathbf{G}^T \tag{3.30}$$

Choices for A The equation for the natural subspace projector above has introduced the matrix $\mathbf{A}$. It has been shown in[16] that this matrix plays the role of a preconditioner for the coarse grid displacements. Ideally, the preconditioner used for the Conjugate Gradient iterations should be taken for $\mathbf{A}$.
The Equations in the previous chapter show that the Lagrange multipliers are corrected by a step of $\mathbf{AG}$ at every Conjugate Gradient iteration. From a dimensional point of view, one can thus argue that since $\mathbf{G}$ represents restrictions on the displacement modes, $\mathbf{A}$ should represent a stiffness matrix. Particularly, since the product represents interface forces due to rigid body modes, the ideal choice for $\mathbf{A}$ is the preconditioner.
However, choosing the preconditioner typically introduces a significant fill-in in the inversion of $\mathbf{GAG}^T$.
The bottom line is that in most cases simply choosing $\mathbf{A} = \mathbf{I}$ is the more efficient option. During this thesis, this choice has been retained.

3.6 Residuum formulation

The natural solver choice for the FETI problem is the Conjugate Gradient method. This chapter aims at formulating the CG algorithm specifically for the FETI problem. Some alternations to the classical CG notation will be done to ensure parallel efficiency of the method.

The CG algorithm consists of three main points. Firstly new search directions are created, these search directions are then updated through orthogonalization to the previous one, and finally an optimal step length is determined by minimizing the residual error along this direction. In practice, Orthogonalization is performed with regards to all previous search directions due to the finite precision of the machine.
The first question is therefore, how to derive the search directions. Since FETI operates on interface forces as primary variables, we need an update for the force vector.
For any given approximation λ to the interface force solution one can calculate the resulting gap between the substructures as

$$\boldsymbol{\Delta}_i = \sum_s \mathbf{B}^{(s)}\mathbf{u}^{(s)} = \mathbf{d} - \mathbf{F}\lambda_i - \mathbf{G}\boldsymbol{\alpha} = \mathbf{0} \tag{3.31}$$

The gap, projected to the natural subspace can thus be written as

$$\mathbf{r}_i = \mathbf{P}^T(\mathbf{d} - \mathbf{F}\lambda_{\mathbf{N}i}) \tag{3.32}$$

where the rigid body mode component vanished due to the very definition of the natural subspace projector $\mathbf{P}$.
The idea now it to estimate the interface forces associated with displacement of that value by a preconditioning step:

$$\mathbf{z}_i = \mathbf{S}\mathbf{r}_i \tag{3.33}$$

Since the local schur operator has full rank, one again need a projection to the natural subspace to eliminate all rigid body mode components.

$$\mathbf{w}_i = \mathbf{P}\mathbf{z}_i \tag{3.34}$$

The dual schur operator $\mathbf{F}$ is used to calculate the interface displacements related to this forces as

$$\mathbf{q}_i = \mathbf{F}\mathbf{w}_i \tag{3.35}$$

After full orthogonalization of the search direction a line-search is performed to minimize the energy expression. The Lagrange multiplier is then updated as

$$\delta_i = \mathbf{q}_i{}^T \mathbf{w}_i \tag{3.36}$$
$$\gamma_i = \mathbf{r}_i{}^T \mathbf{z}_i \tag{3.37}$$
$$\lambda_{\mathbf{F}i+1} = \lambda_{\mathbf{F}i} + (\gamma_i/\delta_i)\mathbf{w}_i \tag{3.38}$$

The resulting gap to to the updated force does not need to be explicitly computed, but instead it can be updated too as

$$\mathbf{r}_{i+1} = \mathbf{r}_i - (\gamma_i/\delta_i)\mathbf{P}^T\mathbf{q}_i \tag{3.39}$$

The process is then repeated until satisfying convergence has been obtained. The full solution for lambda is then recovered as

$$\lambda = \lambda_0 + \lambda_F \tag{3.40}$$

The displacements can finally be recovered by Equation (3.8).

3.7 Preconditioners

As will be explained in Chapter 4, iterative solution techniques typically use preconditioners to accelerate convergence. As can be seen from the formulation above preconditioner in the context of FETI is an approximation to the operator

$$\sum_s \mathbf{B}^{(s)}\mathbf{K}^{(s)-1}\mathbf{B}^{(s)T} \tag{3.41}$$

The natural choice for this problem in the FETI formulation can be identified as

$$\mathbf{F}^{-1} = \sum_s \mathbf{B}^{(s)}\mathbf{t}^{(s)}\mathbf{S}^{(s)}\mathbf{t}^{(s)T}\mathbf{B}^{(s)T} \tag{3.42}$$

where the schur complement $\mathbf{S}$ is defined as

$$\mathbf{S}^{(s)} = \mathbf{K}^{(s)}_{bb} - \mathbf{K}^{(s)}_{bi}\mathbf{K}^{(s)+}_{ii}\mathbf{K}^{(s)}_{ib} \tag{3.43}$$

This version of the preconditioner is typically denoted as Dirichlet preconditioner in the literature.
The pseudo inversion of $\mathbf{K}^{(s)+}$ is costly and introduces a lot of non-zero entries in the preconditioner matrix. Therefore [7] proposed to neglect the second term in Equation (3.43), which results in the o called lumped preconditioner. One can summarize

$$\mathbf{F}_D^{-1} = \sum_s \mathbf{B}^{(s)}\mathbf{t}^{(s)}\left(\mathbf{K}^{(s)}_{bb} - \mathbf{K}^{(s)}_{bi}\mathbf{K}^{(s)+}_{ii}\mathbf{K}^{(s)}_{ib}\right)\mathbf{t}^{(s)T}\mathbf{B}^{(s)T} \quad \text{... Dirichlet Preconditioner} \tag{3.44}$$
$$\mathbf{F}_L^{-1} = \sum_s \mathbf{B}^{(s)}\mathbf{t}^{(s)}\mathbf{K}^{(s)+}_{ii}\mathbf{t}^{(s)T}\mathbf{B}^{(s)T} \quad \text{... Lumped Preconditioner} \tag{3.45}$$

However, this definition of the preconditioner has been shown to be not consistent in [18] for the case where crosspoints are present. A Crosspoint is a point, of the mesh where more than two substructures meet. An example would be node 5 in Figure 3.2 where all four substructures meet. It is obvious that our definition of the Lagrange multipliers introduces a redundancy in the Equations, since one could simply drop the four diagonal Lagrange multipliers, and still the problem would be fully determined. The identifications and removal of redundant Lagrange multipliers, however, is cumbersome and inefficient.
From a mechanical point of view, the problem arises when updating the interface displacements in the Conjugate

Gradient. If one dof carries more than one Lagrange multiplier, than its displacement is multiply updated and the update steps typically contradict on another in a non-converged situation.
A simple and effective approach to eliminates this problem is Multiplicity Scaling. It can be formulated as

$$\mathbf{F}_D^{-1} = \sum_s \mathbf{D}^{(s)}\mathbf{B}^{(s)}\mathbf{t}^{(s)}\mathbf{S}^{(s)}{\mathbf{t}^{(s)}}^T{\mathbf{B}^{(s)}}^T\mathbf{D}^{(s)} \quad \text{... Dirichlet Preconditioner} \tag{3.46}$$

$$\mathbf{F}_L^{-1} = \sum_s \mathbf{D}^{(s)}\mathbf{B}^{(s)}\mathbf{t}^{(s)}{\mathbf{K}_{ii}^{(s)}}^+{\mathbf{t}^{(s)}}^T{\mathbf{B}^{(s)}}^T\mathbf{D}^{(s)} \quad \text{... Lumped Preconditioner} \tag{3.47}$$

where $\mathbf{D}^{(s)}$ is a substructure based diagonal scaling matrix, that holds the multiplicity of the corresponding interface dofs. As multiplicity of a dof we define the number of Lagrange multiplier related to that dof. This multiplicity can be derived very elegantly as

$$\mathbf{D} = \frac{1}{2}\mathbf{B}^T\mathbf{B} \tag{3.48}$$

$$\mathbf{B} = \sum_s \mathbf{B}^{(s)}\mathbf{t}^{(s)} \tag{3.49}$$

Another inconsistency, when it comes to preconditioners are heterogeneities across substructure interfaces. Similarly to the considerations above, a scaling approach can be defined too, according to

$$\mathbf{F}_D^{-1} = \sum_s \boldsymbol{\beta}^{(s)}\mathbf{B}^{(s)}\mathbf{t}^{(s)}\mathbf{S}^{(s)}{\mathbf{t}^{(s)}}^T{\mathbf{B}^{(s)}}^T\boldsymbol{\beta}^{(s)} \quad \text{... Dirichlet Preconditioner} \tag{3.50}$$

$$\mathbf{F}_L^{-1} = \sum_s \boldsymbol{\beta}^{(s)}\mathbf{B}^{(s)}\mathbf{t}^{(s)}{\mathbf{K}_{ii}^{(s)}}^+{\mathbf{t}^{(s)}}^T{\mathbf{B}^{(s)}}^T\boldsymbol{\beta}^{(s)} \quad \text{... Lumped Preconditioner} \tag{3.51}$$

$$\beta^{(s),i} = diag(\mathbf{K}_{bb}^{(s),i})\left[\sum_{r:\Gamma_I^i \subset \{\Gamma_I \cap \Gamma_I^{(r)}\}} diag(\mathbf{K}_{bb}^{(r),i})\right]^{-1} \tag{3.52}$$

Rixen and Farhat introduced this approach as "Superlumped Smoothed Preconditioner". In the literature the more descriptive term "K-scaling" is often used.
Both scaling approaches introduced here are appealing since they do not introduce a significant extra cost, while drastically improving the solver performance, especially for the case with high heterogeneities across the interface. The policy in this thesis, that can also be recommended with clear considence as a standard approach, therefore is to always use both of these scaling types combined. Exceptions(see Chapter 6) will be explicitly mentioned.

3.8 Scalability

The main goal of a DD algorithm is scalability. Generally, one can distinguish numerical scalability and parallel scalability. In iterative DD methods as FETI, numerical scalability is given if the condition number after the preconditioning step grows weakly with the ratio of the sub-domain-size h to the overall size H.
For unpreconditioned interface problems [13] have shown, that the condition number grows asymptotically

$$\mathcal{K} = \mathcal{O}\left(\frac{H}{h}\right) \tag{3.53}$$

which makes algorithms like that useless for real engineering problems. The authors in [13] have proven mathematically, that using sub-domain-based Dirichlet operators as pre-conditioners results in a condition number of the natural coarse problem of

$$\mathcal{K} = \mathcal{O}\left(1 - log^{\beta}(\frac{H}{h})\right) \tag{3.54}$$

It is important to note, that the numerical scalability of the FETI-1 method can be traced back to the inherent problem of the natural coarse space. This gives rise to a problem, when it comes to structural dynamics calculations. As [3] writes, and as it was shown in Section 3.5 the natural coarse problem was introduced due to a potential singularity of the substructure stiffness matrices.
This singularity, however, its no longer present in dynamic calculation, since an implicit time integrator typically leads to a system of equations that is governed by substructure matrices of the form

$$\mathbf{A}^{(s)} = \mathbf{M}^{(s)} + \zeta \Delta t^2 \mathbf{K}^{(s)} \tag{3.55}$$

Since $\mathbf{M}^{(s)}$ is positive definite and $\mathbf{K}$ is positive semi definite, any linear combination of the two is also positive definite and thus invertible. However, as Farhat[3] wrote, loosing the projector $\mathbf{P}$ means loosing its mechanism of propagating the error globally, which was the basis of the scalability considerations performed in [3], see also Section 3.8.
Without the natural coarse problem,however, the conditioning properties are thrown back to Equation (3.53) A solution to this problem is the introduction of an auxiliary coarse problem, which will leads to the FETI-2 methods described in Section 5.2.

Chapter 4

Iterative Solution techniques

It has already been outlined, that the goal of FETI methods is parallelism. The approach was to solve local domain-wise problems on different computing nodes. A serial interface problem then remains to connect the local problems. It thus comes natural, that the solution process of this problem should also be parallelized, otherwise scalability will be quite poorly due to Amdahl's law.
Direct solver thus can not be used as they are inherently serial. We chose to use iterative solver instead, which will be discussed in this chapter.

4.1 Theory

For the purpose of this introduction, and to be consistent with the notation from [22], considerations will be generalized to the linear system

$$\mathbf{Ax} = \mathbf{b} \tag{4.1}$$

As described in [12][20], the convergence properties of an iterative method applied to a system of type (4.1) depend on the condition number of the matrix $\mathbf{A}$.
We will see in Chapter 6, that substructuring often leads to very ill-conditioned problems. One therefore typically solves the related problem

$$\mathbf{HAx} = \mathbf{Hb} \tag{4.2}$$

instead.

The matrix $\mathbf{H}$ is thereby chosen as an approximation to the inverse of $\mathbf{A}$, which significantly improves the condition number of the new iteration matrix $\mathbf{HA}$.
The crucial part therefore is, of course, the choice of an appropriate preconditioner. A simple, but not very effective approach could be the inverse of the diagonal of $\mathbf{A}$. Other types are deflation, augmentation as well as balancing. For the FETI method, the natural choice, however, is projection. The reason for this is explained in [7].
The idea is simple. One assumes that the bad condition number of the matrix is caused by just a little fraction of the full eigenvalue spectrum. One therefore tries to identify these "bad" eigenmodes and create a so called "auxiliary coarse space(deflation space)" with those modes. The problem is then solved directly within this small subspace, before the iterative scheme continues in the remaining space, orthogonal to the deflation space. If the deflation space is chosen right, the convergence properties are significantly improved.

4.2 The Conjugate Gradient algorithm

The most prominent iterative solver is the Conjugate Gradient method. We will see, that the choice of CG comes natural with the FETI formulation. This section is denoted to a formulation of several special variants of CG that we will use in Chapter 5. Considerations are limited to Preconditioned Conjugate Gradient methods.

4.2.1 The Projected Preconditioned Conjugate Gradient algorithm

The PPCG algorithm introduces both a left preconditioner $\mathbf{H}$ as well as a right preconditioner $\mathbf{\Pi}$, where the latter one can be interpreted as an A-orthogonal projection to a subspace defined by U:

$$\mathbf{\Pi} = \mathbf{I} - \mathbf{U}(\mathbf{U}^T\mathbf{A}\mathbf{U})\mathbf{U}^T\mathbf{A} \tag{4.3}$$

The derivation of the projector equation was outlined in Section 3.5.
The general form of a PPCG algorithm is depicted in Figure 4.1

$\mathbf{x}_0 = \Pi\mathbf{x}_{00} + \mathbf{U}(\mathbf{U}^T\mathbf{A}\mathbf{U})^+\mathbf{U}^T\mathbf{b}$

$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$

$\mathbf{z}_0 = \mathbf{H}\mathbf{r}_0$

$\mathbf{p}_0 = \mathbf{\Pi}\mathbf{z}_0$

While not converged

- $\mathbf{q}_i = \mathbf{A}\mathbf{p}_i$
- $\alpha_i = \frac{\langle \mathbf{r}_i \mathbf{z}_i \rangle}{\langle \mathbf{q}_i \mathbf{p}_i \rangle}$
- $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{p}_i$
- $\mathbf{r}_{i+1} = \mathbf{r}_i - \alpha_i \mathbf{q}_i$
- $\mathbf{z}_{i+1} = \mathbf{H}\mathbf{r}_{i+1}$
- $\mathbf{z}_{i+1} = \mathbf{H}\mathbf{r}_{i+1}$
- For $j = 1, ..., i$
  - $\frac{\langle \mathbf{z}_{i+1}, \mathbf{A}\mathbf{p}_j \rangle}{\langle \mathbf{p}_j, \mathbf{A}\mathbf{p}_j \rangle}$
  - $\mathbf{p}_{i+1} = \mathbf{\Pi}\mathbf{z}_{i+1} - \beta_i \mathbf{p}_i$

Figure 4.1: *Structogram of the Projected Preconditioned Conjugate Gradient. Compared to the PCG algorithm, the solution is searched for in a subspace, orthogonal to a chosen space* $\mathbf{U}$*. The idea is to put "bad modes" into this space, to improve convergence.*

From the formulation above it is obvious, that the choice of the coarse space is essential. Naturally, further augmenting the space $\mathbf{U}$ leads to better convergence. In the limit, if all vectors are incorporated into $\mathbf{U}$ a direct solution method is retrieved, and convergence is achieved in one step. However, building the projector $\mathbf{\Pi}$ entails the factorization of $\mathbf{U}^\mathsf{T}\mathbf{A}\mathbf{U}$. A reasonable compromise should thus be found.
A useful tool for convergence studies are error estimators. In [12] the following a posterior one is introduced for the PCG algorithm:

$$\frac{\|\mathbf{x}_* - \mathbf{x}_i\|_A}{\|\mathbf{x}_* - \mathbf{x}_0\|_A} = 2\left[\frac{\sqrt{\lambda_{\mathbf{HA},max}/\lambda_{\mathbf{HA},min}} - 1}{\sqrt{\lambda_{\mathbf{HA},max}/\lambda_{\mathbf{HA},min}} + 1}\right]^i \tag{4.4}$$

Here, $\mathbf{x}_*$ denotes the final, converged solution, $\lambda_{\mathbf{HA},max}$ and $\lambda_{\mathbf{HA},min}$ denote the maximal and minimal value of the Preconditioned matrix $\mathbf{A}$ respectively.
Equivalently an estimate for the PPCG algorithm can be derived as

$$\frac{\|\mathbf{x}_* - \mathbf{x}_i\|_A}{\|\mathbf{x}_* - \mathbf{x}_0\|_A} = 2\left[\frac{\sqrt{\lambda_{\mathbf{HA\Pi},max}/\lambda_{\mathbf{HA\Pi},min}} - 1}{\sqrt{\lambda_{\mathbf{HA\Pi},max}/\lambda_{\mathbf{HA\Pi},min}} + 1}\right]^i \tag{4.5}$$

The function described by this formula is visualized in Figure 4.2. The figure shows, that the convergence properties are extremely sensible to the condition number of the projected, preconditioned operator. The proposed strategy in the PPCG algorithm thus is to identify all extreme, isolated eigenvalues an use their corresponding eigenvectors to built the coarse space. The remaining iteration matrix will thus show a lower condition number, which should be reflected higher convergence rates.

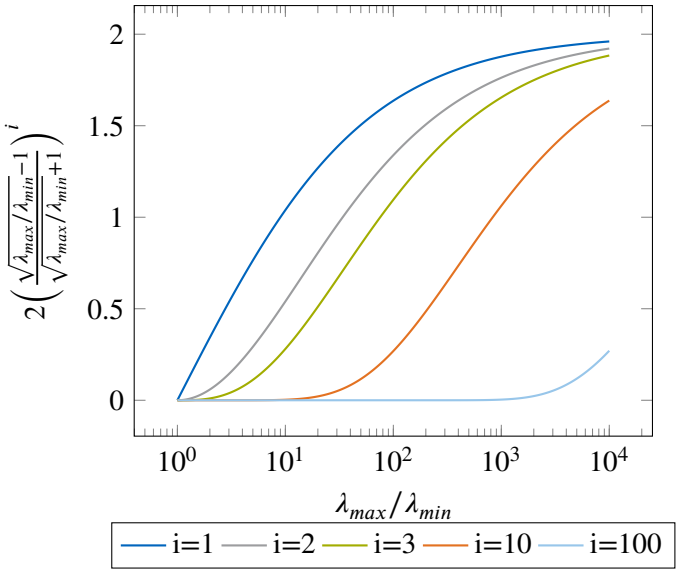


Figure 4.2: *A posteriori error estimation for the PCG algorithm according to Equation (4.4). As expected, a condition number of 1 leads to a direct solution. The graph shows that an increasing eigenvalue ratio, drastically worsens performance.*

4.2.2 The Multi Preconditioned Conjugate Gradient algorithm

The main idea of the Multi Preconditioned Conjugate Gradient algorithm is intuitive. It is designed for problems, where the preconditioner can be written as a sum of local contributions. The approach is straightforward. Instead of adding all local contributions together, each local direction is solved for independently. Therefore, in every iteration the algorithm searches for an optimal step in a space of dimension # *substructures*. Of course, the cost of one iteration is significantly higher than for FETI-2, but the hope is, that the overall number of iterations can be significant reduced, especially for problems with very local phenomena like jumps in the material coefficients. The MMPCG algorithm is summarized in Figure 4.3.

The Adaptive Multi Preconditioned Conjugate Gradient Algorithm

The previous section has introduced the Multi Preconditioned Conjugate Gradient method. Therein, each iteration will give significantly better results than a simple PCG one, thanks to the increased iteration space. However, with a growing number of substructures, solving in the growing search space can become quite expensive.

$\mathbf{x}_0 = \Pi\mathbf{x}_{00} + \mathbf{U}(\mathbf{U}^T\mathbf{A}\mathbf{U})^+\mathbf{U}^T\mathbf{b}$

$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$

$\mathbf{Z}_0 = (\sum_{s=1}^{N} \mathbf{H}^{(s)})\mathbf{r}_0$

$\mathbf{P}_0 = \mathbf{\Pi}\mathbf{Z}_0$

While not converged

- $\mathbf{Q}_i = \mathbf{A}\mathbf{P}_i$
- $\mathbf{\Delta}_i = \mathbf{Q}_i{}^T\mathbf{P}_i,\ \boldsymbol{\gamma}_i = \mathbf{P}_i{}^T\mathbf{r}_i,\ \boldsymbol{\alpha}_i = \mathbf{\Delta}_i{}^+\boldsymbol{\gamma}_i$
- $\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{P}_i\boldsymbol{\alpha}_i$
- $\mathbf{r}_{i+1} = \mathbf{r}_i - \mathbf{Q}_i\boldsymbol{\alpha}_i$
- $\mathbf{Z}_{i+1} = [\mathbf{H}^1\mathbf{r}_{i+1}|...|\mathbf{H}^N\mathbf{r}_{i+1}]$
- For $j = 1, ..., i$
  - $\boldsymbol{\beta}_{i,j} = \frac{\langle \mathbf{z}_{i+1}, \mathbf{A}\mathbf{p}_j \rangle}{\langle \mathbf{p}_j, \mathbf{A}\mathbf{p}_j \rangle}$
  - $\mathbf{P}_{i+1} = \mathbf{\Pi}\mathbf{Z}_{i+1} - \mathbf{P}_i\boldsymbol{\beta}_{i,j}$

Figure 4.3: *Structogram of the Multi Preconditioned Projected Conjugate Gradient. The main difference compared to the PPCG algorithm as described in Figure 4.1 is that a whole set of search directions is used in each iteration.*

Remembering that MMPCG was derived as a method for problems, where the preconditioner can be written as a sum of contributions, it seems rational to assume that some local contributions are less important than others. The idea of Adaptive MMPCG therefore is to identify the important directions, built the search space with them, and to sum the not-so-important directions simply up, like in PCG.

To do so, a method must be derived that determines the importance of a search direction. Of course the cost of this calculation must be kept low to get an effective improvement over MPPCG.
An approach to do so, was first described in [21] and further investigated in [22].

First of all, an a priori error estimate should be derived [1].
We know that

$$\mathbf{x}_* = \mathbf{x}_0 + \sum_{j=0}^{n-n_0-1} \alpha_i\mathbf{p}_j = \mathbf{x}_i + \sum_{j=i}^{n-n_0-1} \alpha_j\mathbf{p}_j \tag{4.6}$$

where $\mathbf{x}_*$ is the actual solution. Since, by construction, search directions are $\mathbf{A}$-orthogonal one can write

$$\|\mathbf{d}_i\|_\mathbf{A}^2 = \|\mathbf{x}_* - \mathbf{x}_i\|_\mathbf{A}^2 = \sum_{j=i}^{n-n_0-1} \alpha_j^2 \|\mathbf{p}_i\|_\mathbf{A}^2 \tag{4.7}$$

Simple subtraction finally gives:

$$\|\mathbf{d}_i\|_\mathbf{A}^2 = \|\mathbf{d}_{i-1}\|_\mathbf{A}^2 - \alpha_{i-1}^2 \|\mathbf{p}_{i-1}\|_\mathbf{A}^2 \tag{4.8}$$

The authors in [1] now use this to derive the a-posteriori error estimate as

$$\frac{\|\mathbf{d}_{i-1}\|_{\mathbf{A}}^2}{\|\mathbf{d}_i\|_{\mathbf{A}}^2} = 1 + \frac{\|\alpha_{i-1}\mathbf{p}_{i-2}\|}{\|\mathbf{d}_i\|_{\mathbf{A}}^2} = 1 + \frac{\|\alpha_{i-1}\mathbf{p}_{i-2}\|}{\|\mathbf{r}_i\|_{\mathbf{H}}^2}\frac{\|\mathbf{r}_i\|_{\mathbf{H}}^2}{\|\mathbf{d}_i\|_{\mathbf{A}}^2} = 1 + \frac{\|\alpha_{i-1}\mathbf{p}_{i-2}\|}{\|\mathbf{r}_i\|_{\mathbf{H}}^2}\underbrace{\frac{\|\mathbf{d}_i\|_{\mathbf{AHA}}^2}{\|\mathbf{d}_i\|_{\mathbf{A}}^2}}_{<\lambda_{\mathbf{HA},min}} \tag{4.9}$$

where the last term is related to a Rayleigh quotient for $\mathbf{HA}$.

The a posteriori error estimate can therefore be formulated as

$$\frac{\|\mathbf{d}_i\|_{\mathbf{A}}^2}{\|\mathbf{d}_{i-1}\|_{\mathbf{A}}^2} \leq \left(1 + \lambda_{\mathbf{HA},min}\frac{\|\alpha_{i-1}\mathbf{p}_{i-2}\|}{\|\mathbf{r}_i\|_{\mathbf{H}}^2}\right)^{-1} \tag{4.10}$$

One can now show that from this point, it suffices to show

$$\frac{\|\alpha_{i-1}\mathbf{p}_{i-2}\|}{\|\mathbf{r}_i\|_{\mathbf{H}}^2} \geq \tau \; with \; \tau := \frac{1-\rho^2}{\lambda_{min}\rho^2} \geq 0 \tag{4.11}$$

to guarantee convergence.

An adaptive MPCG algorithm now entails an evaluation of Condition (4.11) at each iteration. Firstly a direction derived as simple sum of all substructure directions is used in every iteration for the search space. Only if the τ-condition is fulfilled for a substructure, the direction is used as additional new search direction in an MPCG sense.

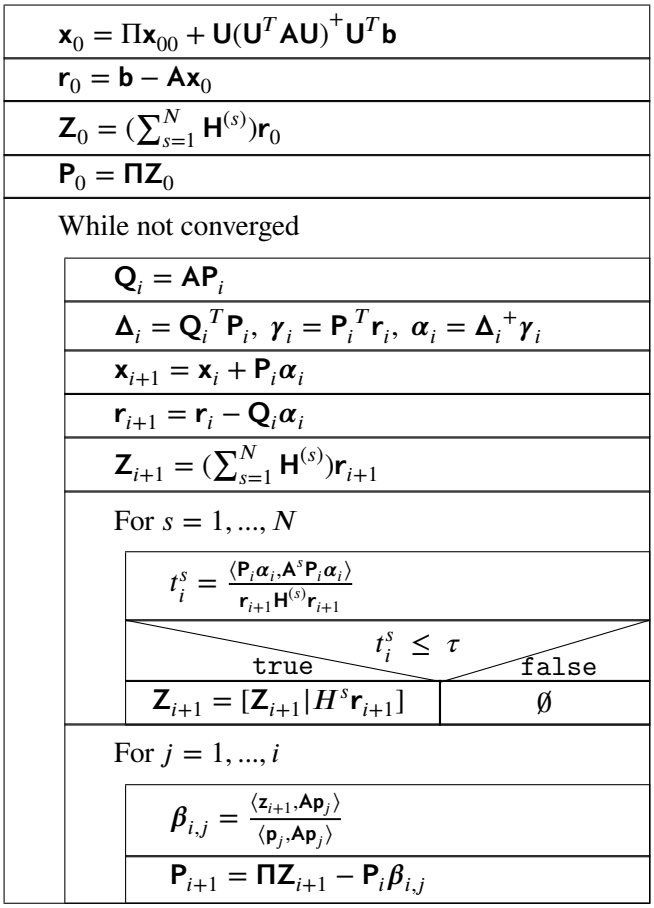


Figure 4.4: *Structogram of the Adaptive Multi Preconditioned Conjugate Gradient algorithm. The algorithm is based on the MPPCG algorithm described in Figure 4.3. While the MMPCG algorithms uses one search direction for each summand of the preconditioner in each iteration, the AMPPCG approach consists in performing a so-called τ-test on each of these directions first. Only directions that have been determined as being important are used to build up the search space, the less important ones are simply summed up in a PCG-like manner.*

Chapter 5

FETI Solvers

The basic system of Equations(3.17) to be solved has already been derived in Chapter 3.

The natural solver choice for the FETI problem is the Conjugate Gradient method. It will be shown that FETI fits neatly into the CG scheme described in Chapter 4.
This thesis is restricted to static problems, where Section 3.5 already explained that rigid body modes cause some problems. The solution will be the introduction of the so-called "natural subspace"
Finally, a projection operator into this subspace has been derived.
Combining this knowledge, with the PCG algorithm introduced in Section 4.2.1 leads to the so-called one-level FETI method(FETI-1).
The projection into the natural subspace however, is also used in all other FETI algorithms.

5.1 One-level FETI(FETI-1)

This section aims at formulating the basic PCG algorithm specifically for the FETI problem as introduced in Chapter 3. The name refers to the one-level of deflation(projection to the natural subspace).
Some alternations to the classical CG notation will be done to ensure parallel efficiency of the method.

The PCG algorithm consists of three main points. First, new search directions are created, these search directions are then updated through orthogonalization to the previous ones, and finally an optimal step length is determined by minimizing the residual error along this direction. In practice, orthogonalization is performed with regards to all previous search directions due to the finite precision of the machine.
The first question is therefore, how to derive the search directions. Since FETI operates on interface forces as primary variables, we need an update for the force vector.

The FETI-1 algorithm is outlined in Figure 5.1. The reader may note the relation to Figure 4.3.

5.2 Two-level FETI(FETI-2)

The so-called FETI-2 algorithm can be classified as an Projected (Preconditioned) Conjugate Gradient algorithm just like FETI-1.
As mentioned in Section 3.8, the main difference lies in the introduction of an auxiliary coarse space.
Albeit derived as necessity for heterogeneous structural dynamics problems, a suitable auxiliary coarse space has also proven very useful for static FETI calculations. Therefore, every FETI-1 method, enhanced by an auxiliary coarse space of any kind shall now be called a FETI-2 method, referring to the two levels of projection.
One particular example of the general advantage of an auxiliary coarse space was first described in [3] an further analysed in [6].

(1) Calculate equilibrium solution with regards to rigid body modes	$\lambda_{\mathbf{N}0} = \mathbf{AG}(\mathbf{G}^T\mathbf{AG})^{-1}\mathbf{e}$
(2) Projector to natural subspace	$\mathbf{P} = \mathbf{I} - \mathbf{AG}(\mathbf{G}^T\mathbf{AG})^{-1}\mathbf{G}^T$
(3) Preconditioner	$\mathbf{S} = \sum_s \mathbf{S}^{[s]}$
(4) Calculate residual(gap) in natural subspace	$\mathbf{r}_0 = \mathbf{P}^T(\mathbf{d} - \mathbf{F}\lambda_{\mathbf{N}0})$
(5) Calculate resulting forces (preconditioning)	$\mathbf{z} = \mathbf{S}\mathbf{r}_0$,
(6) Remove rigid body-components of forces	$\mathbf{w}_0 = \mathbf{P}\mathbf{z}_0$
(7) Initialize	$\lambda_{\mathbf{F}0} = \mathbf{0}, i = 0$
(8) While not converged	$\sqrt{\mathbf{r}_i^T\mathbf{z}_i} > \epsilon$
(9) Compute gap-change due to forces $\mathbf{w}_i$	$\mathbf{q} = \mathbf{F}\mathbf{w}_i$
(10) Auxiliary parameter	$\delta_i = \mathbf{q}_i{}^T\mathbf{w}_i$
(11) Auxiliary parameter	$\gamma_i = \mathbf{r}_i^T\mathbf{z}_i$
(12) Step in the new direction	$\lambda_{\mathbf{F}i+1} = \lambda_{\mathbf{F}i} + (\gamma_i/\delta_i)\mathbf{w}_i$
(13) Gap after force step	$\mathbf{r}_{i+1} = \mathbf{r}_i - (\gamma_i/\delta_i)\mathbf{P}^T\mathbf{q}_i$
(14) Calculate resulting forces (preconditioning)	$\mathbf{z}_{i+1} = \mathbf{S}r_{i+1}$
(15) Remove rigid body-components of forces	$\mathbf{w}_{i+1} = \mathbf{P}\mathbf{z}_{i+1}$
(16) Loop over previous iterations	for: $0 \le j \le i$
(17) Compute factor	$\phi_{i,j} = \mathbf{q}_j^T\mathbf{w}_{i+1}$
(18) Orhorgonalize to direction j	$\mathbf{w}_{i+1} \leftarrow \mathbf{w}_{i+1} - (\phi_{i,j}/\delta_j)\mathbf{w}_j$
(19) Increase iteration counter	$i \leftarrow i + 1$
(20) Compute total interface forces	$\lambda = \lambda_{\mathbf{N}0} + \lambda_{\mathbf{F}i}$

Figure 5.1: *Structogram of the FETI-1 algorithm. It can be classified as a Projected Preconditioned Conjugate Gradient algorithm as described in Section 4.2.1. The coarse* $\mathbf{R}$ *grid is built upon the rigid body modes. The solution in the coarse space is searched for directly, the Conjugate Gradient then gives the natural subspace component. The solution is finally obtained as a sum of those two.*

The basic FETI-1 algorithm shows bad convergence due to limited error propagation, especially for heterogeneities, see Chapter 6. This is also reflected in the very broad eigenvalue distribution of FETI-1 (see Figure 6.2).
A possible solution is the introduction of an auxiliary coarse space, that holds critical modes. The idea is to identify the bad eigenmodes, or an approximation thereof, and to then solve the problem in a subspace (the so-called coarse space) orthogonal to these eigenmodes:

$$\mathbf{C}^T\mathbf{\Delta}_i = \mathbf{0} \tag{5.1}$$

which is completely analogous to Equation (3.23).
The coarse space projector can thus be derived as

$$\mathbf{P}_c = \mathbf{I} - \mathbf{C}(\mathbf{C}^T\mathbf{F}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{F} \tag{5.2}$$

The FETI-2 algorithm is described in Figure 5.2.

(1) Calculate equilibrium solution with regards to rigid body modes	$\lambda_{\mathbf{N}0} = \mathbf{AG}(\mathbf{G}^T\mathbf{AG})^{-1}\mathbf{e}$
(2) Projector to natural subspace	$\mathbf{P} = \mathbf{I} - \mathbf{AG}(\mathbf{G}^T\mathbf{AG})^{-1}\mathbf{G}^T$
(3) Preconditioner	$\mathbf{S} = \sum_s \mathbf{S}^{[s]}$
(4) Coarse space equilibrium solution	$\lambda_{\mathbf{C}0} = \mathbf{C}(\mathbf{C}^T\mathbf{FC})^{-1}\mathbf{C}^T(\mathbf{d} - \mathbf{F}\lambda_{\mathbf{N}0})$
(5) Coarse space projector	$\mathbf{P}_c = \mathbf{I} - \mathbf{C}(\mathbf{C}^T\mathbf{FC})^{-1}\mathbf{C}^T\mathbf{F}$
(6) Calculate residual(gap) in natural subspace	$\mathbf{r}_0 = \mathbf{P}^T(\mathbf{d} - \mathbf{F}\lambda_{\mathbf{N}0})$
(7) Calculate resulting forces (preconditioning)	$\mathbf{z} = \mathbf{S}\mathbf{r}_0$
(8) Remove rigid body-components of forces	$\mathbf{w}_0 = \mathbf{P}\mathbf{z}_0$
(9) Initialize	$\lambda_{\mathbf{F}0} = \mathbf{0}, i = 0$
(10) While not converged	$\sqrt{\mathbf{r}_i^T\mathbf{z}_i} > \epsilon$
(11) Remove coarse space components of gap-change due to forces $\mathbf{w}_i$	$\mathbf{q} = \mathbf{P}_c{}^T\mathbf{F}\mathbf{w}_i$
(12) Auxiliary parameter	$\delta_i = \mathbf{q}_i{}^T\mathbf{w}_i$
(13) Auxiliary parameter	$\gamma_i = \mathbf{r}_i^T\mathbf{z}_i$
(14) Step in the new direction	$\lambda_{\mathbf{F}i+1} = \lambda_{\mathbf{F}i} + (\gamma_i/\delta_i)\mathbf{w}_i$
(15) Gap after force step	$\mathbf{r}_{i+1} = \mathbf{r}_i - (\gamma_i/\delta_i)\mathbf{P}^T\mathbf{q}_i$
(16) Calculate resulting forces (preconditioning)	$\mathbf{z}_{i+1} = \mathbf{S}r_{i+1}$
(17) Remove rigid body-components of forces	$\mathbf{w}_{i+1} = \mathbf{P}\mathbf{z}_{i+1}$
(18) Loop over previous iterations	for: $0 \leq j \leq i$
(19) Compute factor	$\phi_{i,j} = \mathbf{q}_j^T\mathbf{w}_{i+1}$
(20) Orhorgonalize to direction j	$\mathbf{w}_{i+1} \leftarrow \mathbf{w}_{i+1} - (\phi_{i,j}/\delta_j)\mathbf{w}_j$
(21) Increase iteration counter	$i \leftarrow i + 1$
(22) Compute total interface forces	$\lambda = \lambda_{\mathbf{N}0} + \lambda_{\mathbf{C}0} + \mathbf{P}_c\lambda_{\mathbf{F}i}$

Figure 5.2: *FETI-2 algorithm. Differences to the classical FETI-1 algorithm are underlined red.*

5.2.1 Generalized eigenvalues in the overlap(Geneo)

A mathematically appealing approach for the definition of the auxiliary coarse space $\mathbf{C}$ in a FETI-2 method are the so-called Geneo methods (Generalized eigenvalues on the interface).
The idea is to identify the "bad" eigenmodes, corresponding to the "bad" eigenvalues in the spectrum of the system matrix (see Figure 6.2) and to build up the auxiliary coarse space with them. The coarse problem is solved for directly, before the CG iterations themselves are then performed in a subspace orthogonal to these modes, the spectrum of the projected system matrix can thus be significantly improved. Extensive analysis of this methods[22][21][23] has shown their remarkable robustness, especially for the generally quite challenging FETI problems described in Chapter 6. However, although mathematically found and robust, the eigenvalue problems on the interface introduce a substantial computational overhead and cannot be parallelized well. What is more increasing the size of the auxiliary coarse grid degenerates the parallelizability of the algorithm further. Thus other methods like the family of Multi Preconditioned Conjugate Gradient solvers are generally more efficient for engineering problems.

5.3 Simultaneous FETI(FETI-S)

The FETI-1 and FETI-2 algorithm use one search direction at each iteration for the whole interface problem. For the specific case, where the preconditioner is built as a sum of local contributions, however, we have already introduced the appealing approach of Multi Preconditioned (Projected) Conjugate Gradient in Section 4.2.2. This is exactly the case in FETI, in fact, the local preconditioners $\mathbf{F}$ have to be built anyway, so no additional work is introduced here.
The application of the MPPCG concept to the FETI problem is called simultaneous FETI, or FETI-S, and was first described in [16] for two sub-domains, and consequently generalized to an arbitrary number of sub-domains in [17]. The basic idea is to exploit the additive structure of the preconditioner(see Equation (3.20)). One can thus precondition each substructure independently, which results in s search directions that can be searched for.

$$\mathbf{Z}_i = \left[\cdots, \mathbf{B}^{(s)}\mathbf{t}^{(s)}\mathbf{S}^{(s)}\mathbf{t}^{(s)^T}\mathbf{B}^{(s)T}\mathbf{r}_i, \cdots\right] \tag{5.3}$$

The crucial advantage of this method is that, although at each iteration, on minimizes with respect to s search directions, the cost of each iteration is not increasing proportionally, as explained in Section 5.3.1. Particularly, the number of Neumann solves on the substructures is not increased.

$$\mathbf{S} = \sum_s \mathbf{B}^{(s)}\mathbf{t}^{(s)}\mathbf{S}^{(s)}\mathbf{t}^{(s)^T}\mathbf{B}^{(s)T} \tag{5.4}$$

Additionally, the scaling approaches as explained in Section 3.7 are applied to the preconditioner. The efficiency of the FETI-S algorithm was substantially improved in [9] by introducing an important trick, that localized the application of the FETI operator. Moreover, the paper also provides an efficient implementation of the Conjugate Gradient algorithm, that all FETI algorithms provided in this paper are based on. Generally, the FETI-S algorithm can be classified as a Multi Preconditioned Conjugate Gradient Algorithm (MPPCG)[2].
It is noteworthy, that since MPPCG algorithms do not minimize in a Krylov subspace, a heuristic bound for the number of iterations can not be justified. However, extensive numerical analysis have certified a very robust behaviour.
A detailed description of the FETI-S algorithm is provided in Figure 5.3

5.3.1 Computational cost of FETI-S

The idea of FETI-S seems appealing. But of course, the increased number of search directions does not come without a price. Efficiency considerations have been carried out by [9] and this chapter aims at pointing out the most important aspects. First of all, we note that the communication patterns of the algorithm do not change compared to FETI-1, they only involve more data.
Secondly, we obviously see that the most important part of any FETI algorithm are the local Neumann and Dirichlet solves. Compared to the FETI-1 algorithm, the preconditioner now has to be applied all columns of $\mathbf{W}$ instead of just a simple vector. The overhead of this is low.
The paper also introduced a clever method to improve the efficiency of this step by exploiting the locality of $\mathbf{Z}$:

$$\mathbf{Q}_{i+1} = \mathbf{F}\mathbf{W}_{i+1} = \mathbf{F}\mathbf{P}\mathbf{Z}_{i+1} - \sum_{j=0}^{i} \mathbf{Q}_j {\Delta_j}^{+} \boldsymbol{\phi}_{i,j} \tag{5.5}$$

$$\tag{5.6}$$

Inserting the very definition of the natural subspace projector as defined in Equation (**??**) one can write

$$\mathbf{Q}_{i+1} = (\mathbf{F}\mathbf{Z}_{i+1} - \mathbf{F}\mathbf{A}\mathbf{G}(\mathbf{G}^T\mathbf{A}\mathbf{G})^{+}\mathbf{G}^T\mathbf{Z}_{i+1}) - \sum_{j=0}^{i} \mathbf{Q}_j {\Delta_j}^{+} \boldsymbol{\phi}_{i,j} \tag{5.7}$$

$$\tag{5.8}$$

Thanks to this reformulation, only localized Neumann problems need to be solved and the efficiency is greatly improved.
Furthermore, the multiple Neumann problems can typically be solved simultaneously on different processors since parallelism was the motivation for these algorithms in the first place.
Another source of numerical overhead is the (pseudo-)inversion of $\mathbf{\Delta}_i$, which is typically dense. However, its size is *numsubstruct* $\times$ *numsubstruct* and thus relatively small compared to the interface-based operators.

(1) Calculate equilibrium solution with regards to rigid body modes	$\lambda_{\mathbf{N}0} = \mathbf{AG}(\mathbf{G}^T\mathbf{AG})^{-1}\mathbf{e}$
(2) Projector to natural subspace	$\mathbf{P} = \mathbf{I} - \mathbf{AG}(\mathbf{G}^T\mathbf{AG})^{-1}\mathbf{G}^T$
(3) Calculate residual(gap) in natural subspace	$\mathbf{r}_0 = \mathbf{P}^T(\mathbf{d} - \mathbf{F}\lambda_0)$
(4) Calculate resulting forces for each substructure (preconditioning)	$\mathbf{Z}_0 = \left[\cdots, \mathbf{S}^{[s]}\mathbf{r}_0, \cdots\right]$
(5) Rigid body-components of forces	$\mathbf{W}_0 = \mathbf{PZ}_0$
(6) Initialize	$\lambda_{\mathbf{F}0} = 0, i = 0$
(7) While not converged	$\sqrt{\mathbf{r}^T\mathbf{Z1}} > \epsilon$
(8) Compute gap-changes due to forces $\mathbf{W}_i$	$\mathbf{Q}_i = \mathbf{FW}_i$
(9) Auxiliary variable	$\mathbf{\Delta}_i = \mathbf{Q}_i{}^T\mathbf{W}_i$
(10) Auxiliary variable	$\boldsymbol{\gamma}_i = \mathbf{Z}_i{}^T\mathbf{r}_i$
(11) Step in new direction	$\lambda_{\mathbf{F}i+1} = \lambda_{\mathbf{F}i} + \mathbf{W}_i\mathbf{\Delta}_i{}^+\boldsymbol{\gamma}_i$
(12) Gap change due to force step	$\mathbf{r}_{i+1} = \mathbf{r}_i - \mathbf{P}^T\mathbf{Q}_i\mathbf{\Delta}_i^+\boldsymbol{\gamma}_i$
(13) Calculate resulting forces for each substructure (preconditioning)	$\mathbf{Z}_{i+1} = \left[\cdots, \mathbf{S}^{[s]}\mathbf{r}_{i+1}, \cdots\right]$
(14) Remove rigid body-components of forces	$\mathbf{W}_{i+1} = \mathbf{PZ}_{i+1}$
(15) Loop over previous iterations	for: $0 \leq j \leq i$
(16) Compute othogonalization factor	$\boldsymbol{\phi}_{i,j} = \mathbf{Q}_j^T\mathbf{W}_{i+1}$
(17) Orhtogonalize to direction j	$\mathbf{W}_{i+1} \leftarrow \mathbf{W}_{i+1} - \mathbf{W}_j\mathbf{\Delta}_j{}^+\boldsymbol{\phi}_{i,j}$
(18) Increase iteration counter	$i \leftarrow i + 1$
(19) Compute total interface forces	$\lambda = \lambda_{\mathbf{N}0} + \lambda_{\mathbf{F}0}$

Figure 5.3: *The simultaneous FETI algorithm is an implementation of the Multi Preconditioned Conjugate Gradient as described in Section 4.2.2. Each substructure contributes its own search direction that is solved for independently of the others.*

5.4 Adaptive Simultaneous FETI(FETI-AS)

As explained before, the FETI-S solver is just an application of the MPPCG concept, described in Section 4.2.2 on the FETI problem. As already mentioned, while the FETI-S method shows very good convergence rates, its iterations become more expensive with an increasing number of subdomains. AS will be pointed out in the

following sections, typically some search directions are more important than others. It thus seems intuitively to try to identify the really important search directions, and restrict iterations on them. This is exactly what the Adaptive Multi Preconditioned Conjugate Gradient algorithm 4.2.2 does. Applied on the FETI problem, an Adaptive Simultaneous FETI solver can thus be found as described in Figure 5.4.

(1) Calculate equilibrium solution with regards to rigid body modes	$\lambda_{\mathbf{N}0} = \mathbf{AG}(\mathbf{G}^T\mathbf{AG})^{-1}\mathbf{e}$
(2) Projector to natural subspace	$\mathbf{P} = \mathbf{I} - \mathbf{AG}(\mathbf{G}^T\mathbf{AG})^{-1}\mathbf{G}^T$
(3) Calculate residual(gap) in natural subspace	$\mathbf{r}_0 = \mathbf{P}^T(\mathbf{d} - \mathbf{F}\lambda_0)$
(4) Preconditioning	$\mathbf{Z}_0 = \mathbf{S}\mathbf{r}_0$
(5) Rigid body-components of forces	$\mathbf{W}_0 = \mathbf{P}\mathbf{Z}_0$
(6) Initialize	$\lambda_{\mathbf{F}0} = 0, i = 0$
While not converged	$\sqrt{\mathbf{r}^T\mathbf{Z}\mathbf{1}} > \epsilon$
(7) Compute gap-changes due to forces $\mathbf{W}_i$	$\mathbf{Q}_i = \mathbf{F}\mathbf{W}_i$
(8) Auxiliary parameter	$\mathbf{\Delta}_i = \mathbf{Q}_i{}^T\mathbf{W}_i$
(9) Auxiliary parameter	$\boldsymbol{\gamma}_i = \mathbf{Z}_i{}^T\mathbf{r}_i$
(10) Auxiliary parameter	$\boldsymbol{\alpha}_i = \mathbf{\Delta}_i{}^+\boldsymbol{\gamma}_i$
(11) Step in new direction	$\lambda_{\mathbf{F}i+1} = \lambda_{\mathbf{F}i} + \mathbf{W}_i\boldsymbol{\alpha}_i$
(12) Gap change due to force step	$\mathbf{r}_{i+1} = \mathbf{r}_i - \mathbf{P}^T\mathbf{Q}_i\boldsymbol{\alpha}_i$
(13) New	$\mathbf{Z}_{i+1} = \mathbf{S}\mathbf{r}_{i+1}$
Loop over all substructures	$for\ s = 1, ..., N\ do$
(14) Compute convergence indicator	$t_i^s = \frac{\langle \mathbf{W}_i\boldsymbol{\alpha}_i, \mathbf{F}^{[s]}\mathbf{W}_i\boldsymbol{\alpha}_i\rangle}{\mathbf{r}_{i+1}{}^T\mathbf{S}^{[s]}\mathbf{r}_{i+1}}$
(15) τ – test	$t_i^s < \tau?$
`true`	`false`
(16) Add direction to search space $\mathbf{Z}_{i+1} = [Z_{i+1}\|\mathbf{S}^{[s]}\mathbf{r}]$	*(17) Do nothing* $\emptyset$
(18) Remove rigid body-components of forces	$\mathbf{W}_{i+1} = \mathbf{P}\mathbf{Z}_{i+1}$
Loop over previous iterations	for: $0 \leq j \leq i$
(19) Compute orthogonalization factor	$\boldsymbol{\phi}_{i,j} = \mathbf{Q}_j^T\mathbf{W}_{i+1}$
(20) Orthogonalize to direction j	$\mathbf{W}_{i+1} \leftarrow \mathbf{W}_{i+1} - \mathbf{W}_j\mathbf{\Delta}_j{}^+\boldsymbol{\phi}_{i,j}$
(21) Increase iteration counter	$i \leftarrow i + 1$
(22) Compute total interface forces	$\lambda = \lambda_{\mathbf{N}0} + \lambda_{\mathbf{F}0}$

Figure 5.4: *Structogram of the Adaptive Simultaneous FETI algorithm. The algorithm is quite analogue to the FETI-S one. The differences are marked in red. This version shows the implementation of a local τ criterion. Compared to the often used global one, the local criterion allows to iterate on an arbitrarily reduced number of search directions*

As explained in Section 4.2.2 the appropriate choice of τ can become quite delicate in this algorithm. We have already introduced an estimation formula based on a prescribed contraction factor ρ and the minimum eigenvalue of the system matrix. However, all the analyses carried out during this thesis showed that the optimal choice for both τ and ρ is problem-dependent.
In the following, a quick analysis on this issue has been conducted. The setup is the same as in Figure 5.8 and results are provided in Figure 5.6. It can be shown that the necessary number of search directions can be significantly reduced, without increasing the overall iteration count.
We therefore prefer a direct choice of tau over the calculation with the contraction factor. If implemented so, the Adaptive Simultaneous FETI algorithm hardly introduces any additional work in the τ-test, while providing remarkable potential for increasing the efficiency. We therefore suggest that this algorithm should, in principle, always be favoured to simple FETI-S. If in doubt, one better chooses τ bigger, since in the limit $\tau \to \infty$ the FETI-S algorithm is retrieved.

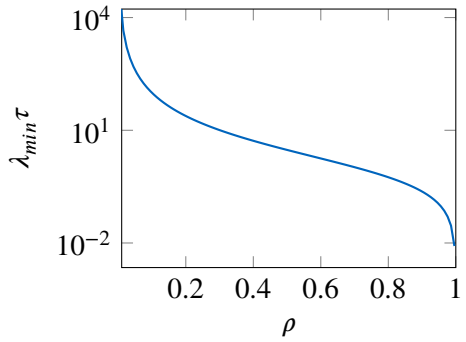


Figure 5.5: *Relation between the contraction factor and τ according to the formula $\frac{1-\tau^2}{\lambda_{min}\tau^2}$. In can be observed, that for a wide range of ρ τ does not change dramatically*

5.4.1 Fast Adaptive Simultaneous FETI(FETI-FAS)

The previous section has highlighted the potential superiority of the FETI-AS solver compared to the FETI-S one. Many search directions seem to be not that important, so neglecting them does not really increase the total iteration count much.
Looking at Figure 5.6 one can also note two things. Firstly, it seems that with progressing iteration, more and more search directions can be neglected, while it does make sense to use all search directions at first.
Secondly, the optimal choice of τ is crucial and, unfortunately, it is problem-dependent. This problem can be avoided by prescribing a contraction factor ρ according to Equation (4.11), this does, however, introduce an eigenvalue problem and it is often difficult to draw a clear line between the smallest eigenvalue and eigenvalues that are numerically zero, due to the projection.
This section is thus dedicated to a thorough analysis of these mechanisms. Specifically, new approaches will be introduced for the selection of important search directions. Since they avoid the expensive τ-test, they will be denoted as Fast Adaptive Simultaneous FETI schemes(FETI-FAS) in this thesis.
Numerical analysis will finally test their robustness and performance compared to FETI-AS and conclusions will be drawn in Chapter 7.

The problem with FETI-1/FETI-2 For the following considerations, the setup already described in Figure 5.8 will be discussed. Iteration numbers for this problem have already been provided in Figure **??**. It seems that the more the algorithm is shifted towards the classical FETI-1 approach, the more rapidly the total iteration count increases. Indeed, the same problem solved with FETI-1 takes 147 iterations.
To analyse this behaviour the development of the convergence indicator $t^{(s)}$ for the substructures has been visu-

alized in Figure 5.13. It can bee seen, that FETI-S is much more capable of improving the substructure solutions uniformly, compared to FETI-1. The reason is, that in FETI-1, all substructure directions are simply summed up. So few contributions dominate the sum. More over, the same force value, relating to a Lagrange multiplier, is used for both sides of the interface in FETI-1, which can lead to significant errors.
Noting that the calculation of $t^{(s)}$ does not introduce a significant overhead, one could think of several ways to improve the convergence:

1. Build blocks of search directions according to their contribution factor
2. Select search directions which showed a decreasing contribution factor
3. Select the search directions with lowest $t^{(s)}$

The results for all three approaches are visualized in Figure 5.14. Due to the results of that figure, scheme 2 has been favoured for the rest of this thesis. If not mentioned otherwise FETI-FAS refers to that scheme from now on

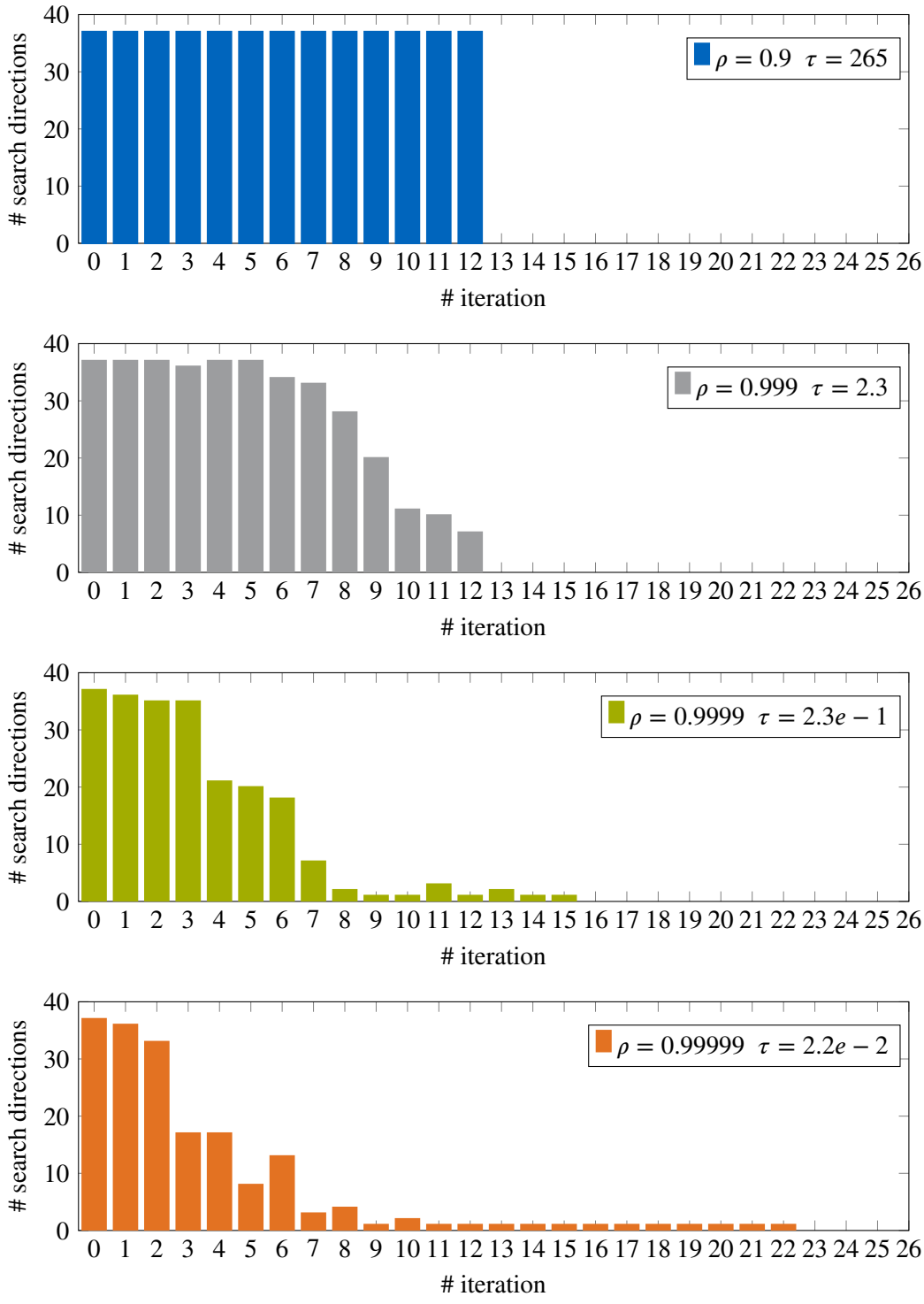


Figure 5.6: *Study for the optimal choice of τ. The problem described in Figure 5.9 is considered here. Four different choices for ρ or τ respectively were tested. The plots visualized the number of search directions during each iteration of the FETI-AS solver. Obviously, the total number of iterations is given by the number of bars in each plot. The figure shows, that an appropriate choice of the parameter τ is crucial in order to exploit the full potential of the FETI-AS Solver. In the top figure, τ has obviously been chosen too big, thus the number of search directions is not significantly reduced. In the bottom figure, τ has been chosen too small, thus the total number of iterations is highly increased, making the advantage of reduced search directions to naught.*

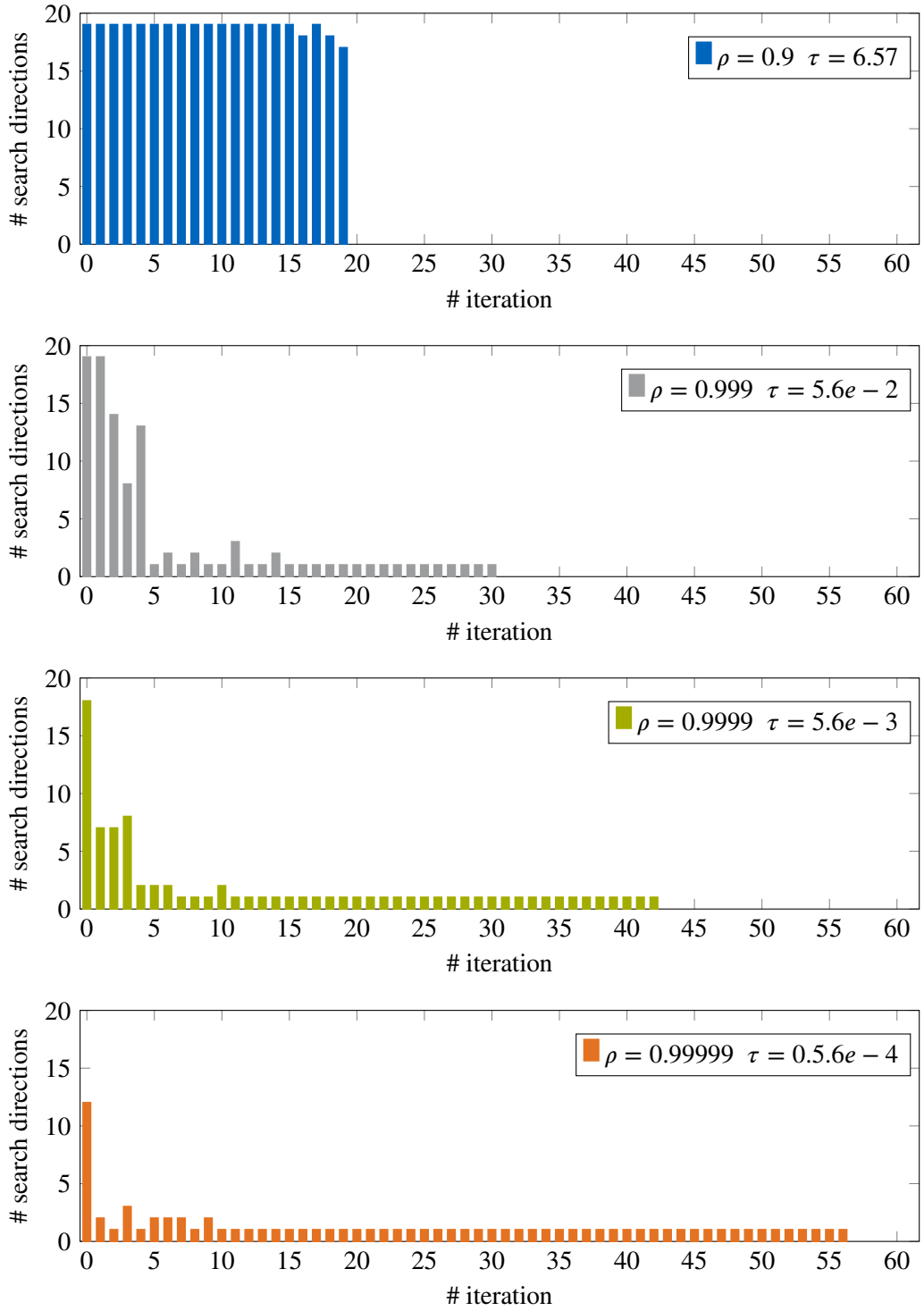


Figure 5.7: *Another study for the optimal choice of τ. In contrast to the setup in Figure 5.7, the domain was split only in half the number of substructures an the domain ratio was changed from 1:1 to 1:3. What is more, a softer material was used. What this aims at is a demonstration that even the contraction factor ρ is not problem independent. This of course constitutes a huge draw back on the generalizability of the FETI-AS method.*

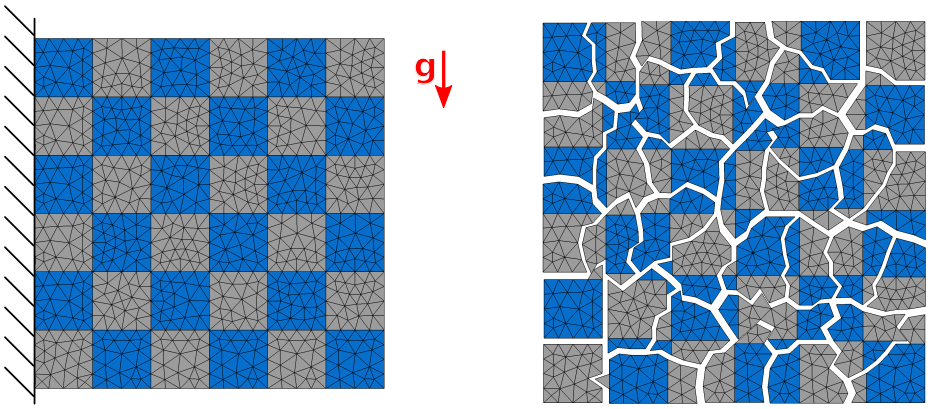


Figure 5.8: *Setup for the visualization of search directions. A square-shaped domain is used. The bearing is statically determined on the left face. The whole structure is subjected to gravity. The two materials (blue and grey) are aligned in an checkerboard pattern. Their stiffness ratio is 1:100. It will be explained in Section 6.3, why heterogeneous problems are of special interest.*

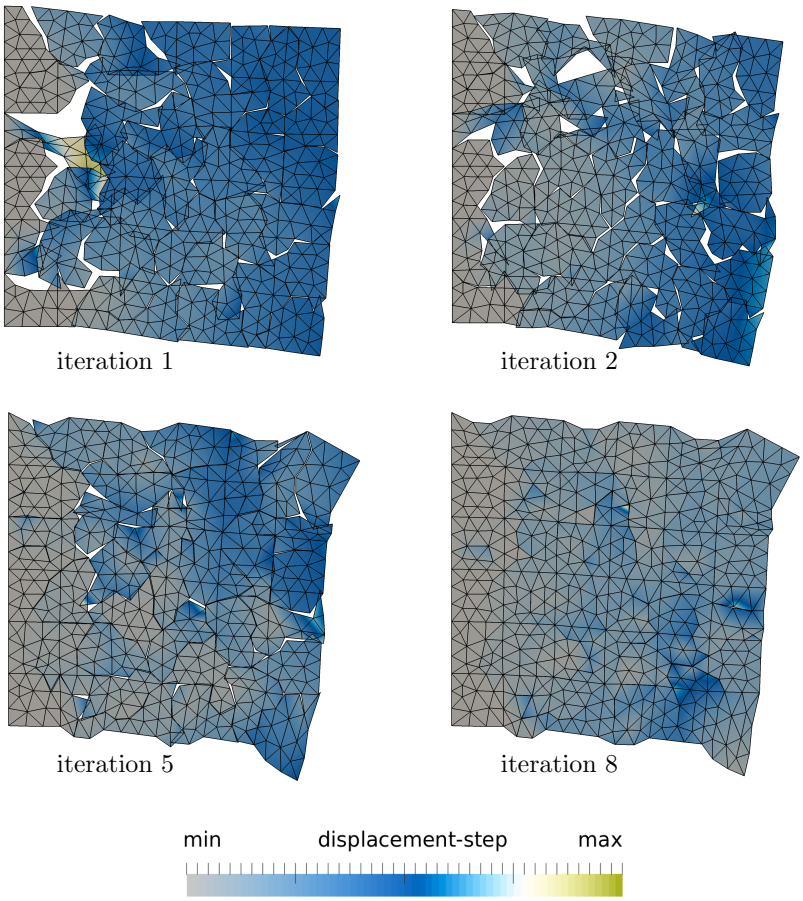


Figure 5.9: *Visualization of the search directions in an Simultaneous FETI simulation of the problem setup described in Figure 5.8. All figures are warped by differently scaled displacement vectors for better visualization. The color corresponds to the step that the algorithm takes in that iteration. The figures show two things. Firstly, the algorithm is obviously very effective in capturing the largest errors. Secondly, with an increasing iteration number, the steps are effectively concentrated on very small areas. This is very well reflected in Figure 5.6, where the number of search directions, that the FETI-AS solver chooses to work on, significantly reduces over the iterations and explains why FETI-AS shows hardly any slowdown in the convergence compared to FETI-S.*

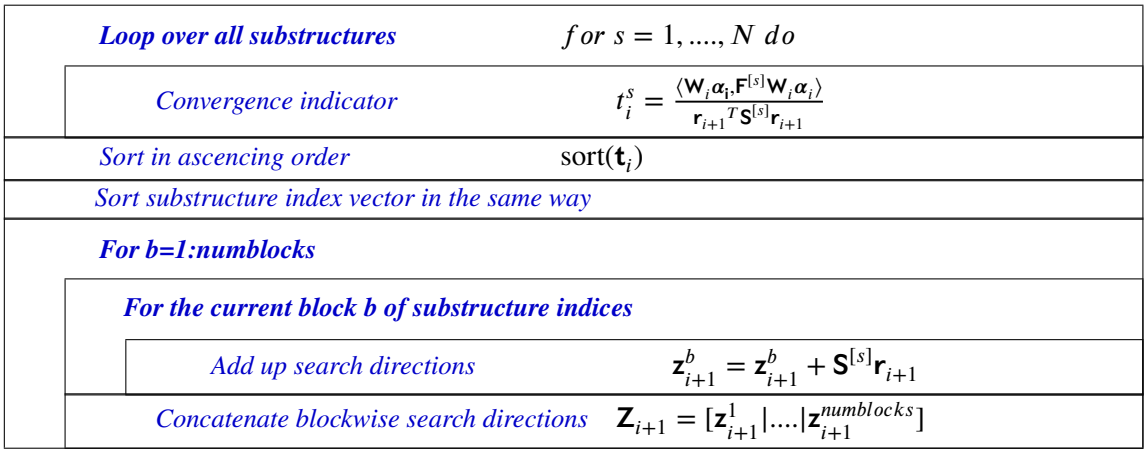


Figure 5.10: *Scheme 1 for the FETI-FAS approach as described in Section 5.4.1. The substructure convergence indicator as sorted in ascending order in each iteration. The substructure indices are then grouped block-wise, according to their convergence indicator. Each block gives one summed up search direction. The search space is then created with these search directions.*

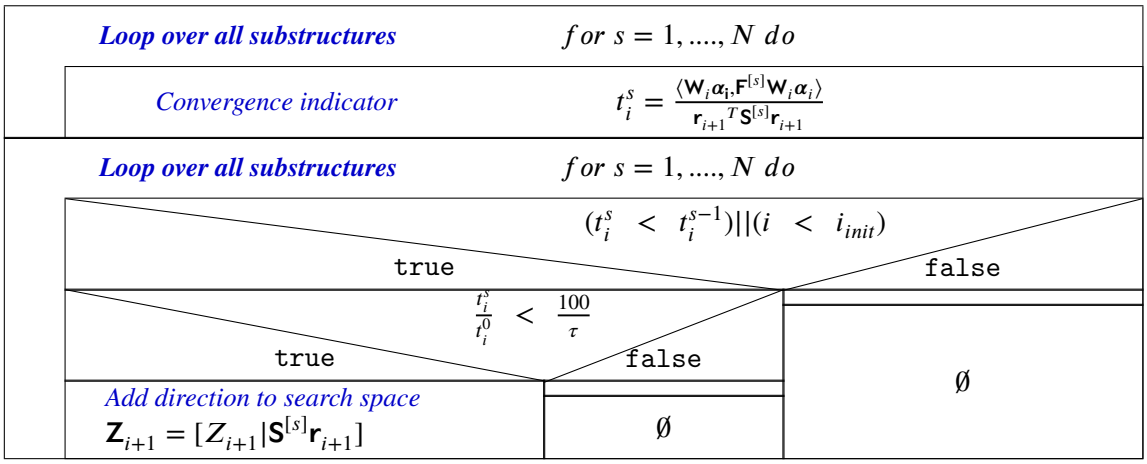


Figure 5.11: *Scheme 2 for the FETI-FAS approach as described in Section 5.4.1. The substructure convergence factors are compared to the previous iterations. If it decreased than the search directions is incorporated into the search space. For a specified number of iterations at the beginning, all search directions are used. In order to avoid the incorporation of search directions that relate to already converged substructures due to numerical fluctuations, search directions are no longer incorporated into the search space, if their convergence factor has converged up to a factor of 100 to the simulation tolerance. For all calculations during this thesis, this barrier has shown to be practical.*

Loop over all substructures	$for\ s = 1, ..., N\ do$
Convergence indicator	$t_i^s = \frac{\langle \mathbf{W}_i \boldsymbol{\alpha}_i, \mathbf{F}^{[s]} \mathbf{W}_i \boldsymbol{\alpha}_i \rangle}{\mathbf{r}_{i+1}{}^T \mathbf{S}^{[s]} \mathbf{r}_{i+1}}$
Sort in ascencing order	$\mathrm{sort}(\mathbf{t}_i)$
Sort substructure index vector in the same way	
For s= indices of $numdir$ ***lowest*** t_i^s	
Concatenate blockwise search directions	$\mathbf{Z}_{i+1} = [\mathbf{Z}_{i+1}^1 \vert \mathbf{S}^{[s]} \mathbf{r}_{i+1}]$

Figure 5.12: *Scheme 3 for the FETI-FAS approach as described in Section 5.4.1. The substructure convergence indicator as sorted in ascending order in each iteration. Only the directions with lowest t_i^s are incorporated into the search space.*

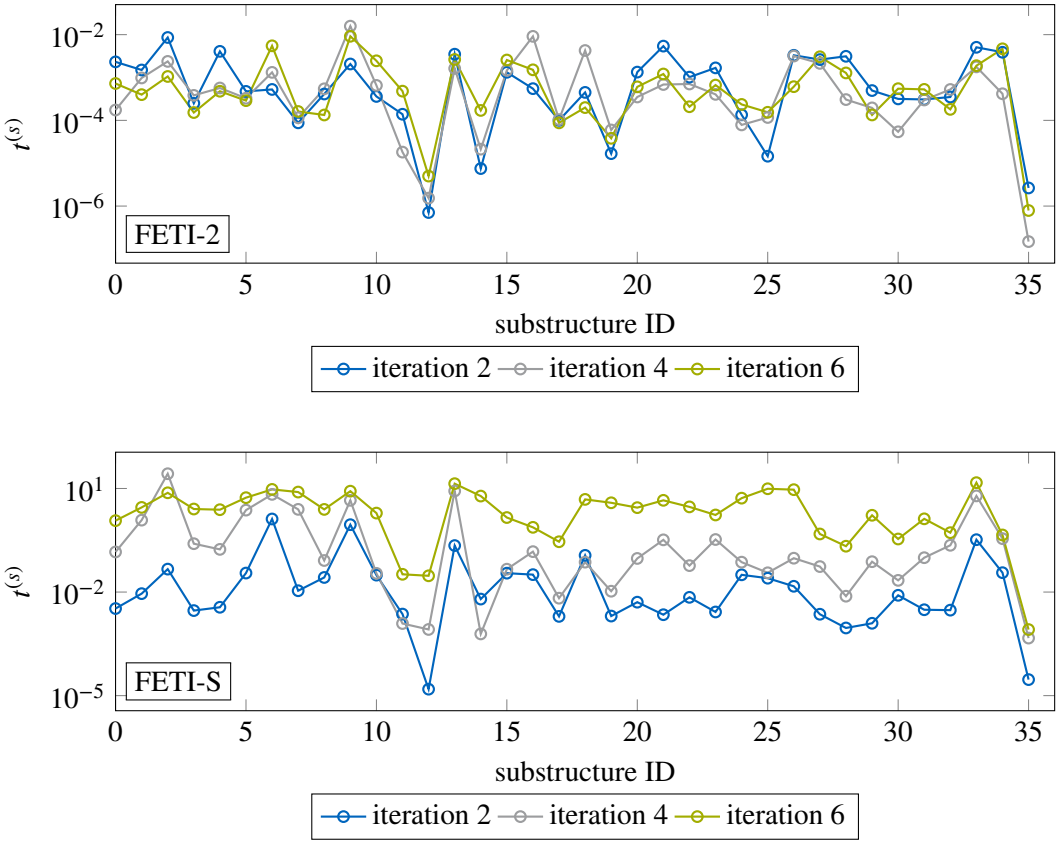


Figure 5.13: *Plot of the substructure convergence indicator $t^{(s)}$ for FETI-2 and FETI-S. Of course, $t^{(s)}$ is irrelevant for those solvers. However, they give a pretty good interpretation on how the choice of τ influences the converge properties of a FETI-AS simulations. Every FETI-AS search space includes the direction, created by the simple sum of preconditioners, as can be seen in Figure 5.4. Some directions are then additionally added according to the so called τ-test(step (15) in Figure 5.4). If all directions are added, a classical FETI-S is retrieved.*

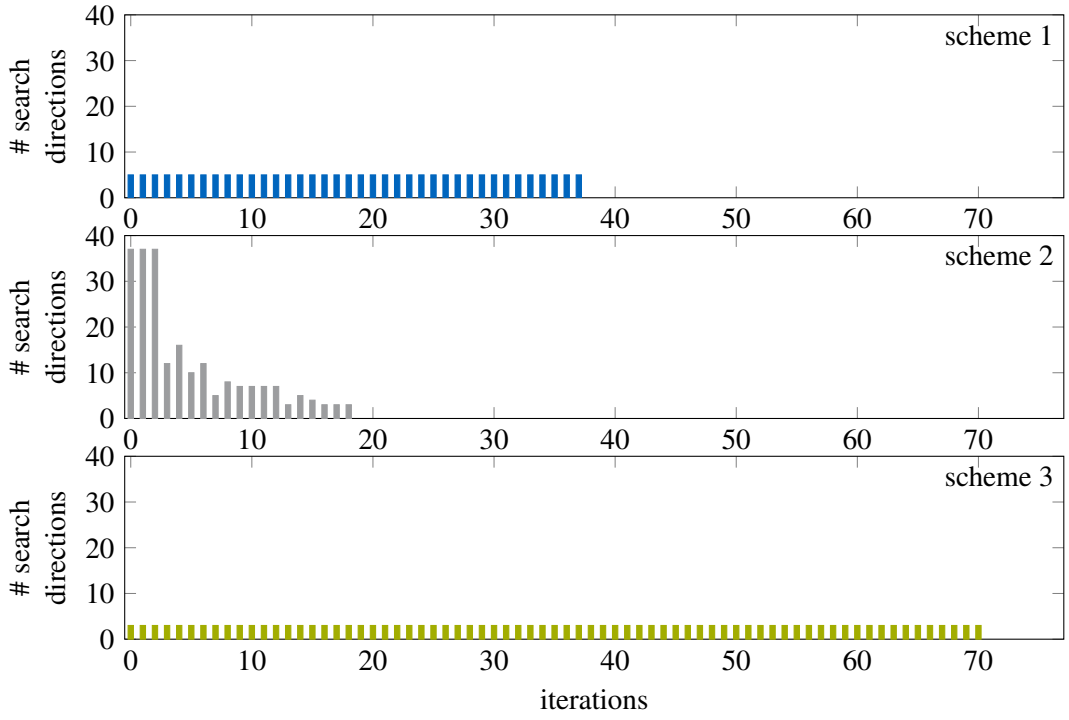


Figure 5.14: *For scheme 1 two search directions, built by the sum of the the lower and upper half of search directions sorted by t_s^i respectively, were used. Scheme 2 was implemented just as described in Section 5.4.1.For scheme 3, the four search directions with lowest $t^{(s)}$ have been used in each iteration.*

Chapter 6

Numerical Assessment

The previous chapter has introduced the FETI method as an efficient implementation of the domain decomposition approach for the purpose of parallelization. However, as any DD method, FETI struggles with some conditioning problems, especially as the number of sub-domains increases.
Chapter 5 introduced several variations of the original FETI-method. Some of which were derived exactly for the purpose of handling the numerical difficulties. Since all mentioned methods have been implemented into the FEMAC code[11], the following chapter provides a profound numerical analysis of the presented methods and their capabilities to handle the heterogeneities along and across the interface, different partitioning schemes, irregular interfaces, incompressibility and inclusion problems. Taking into account the computational effort and scalability issues, conclusions will be drawn, and recommendations will be provided in Chapter 7.

6.1 Default parameters

The previous chapter have already introduced numerous parameters and choices to be made for the FETI solvers. If not explicitly specified otherwise, the following parameters have been used for all calculations in this thesis.

Termination criterion	Residual of 10^{-6}
Preconditioner type	Lumped
A-matrix	Unity matrix
Scaling	Multiplicity scaling + K-scaling
ρ for FETI-AS	0.999
FETI-2 coarse grid	Geneo with 6 lowest eigenmodes per substructure

6.2 Conditioning

As explained in Chapter 1, the FETI type of solvers are hybrids between direct and iterative solvers. A large problem is typically divided into a number of substructures large enough such that each sub-problem can be solved for directly in reasonable time. The hybridization is introduced in the formulation when it comes to solving the connecting interface problem. As explained in Section 4.1 and outlined algorithmically in Section 4.2, iterative methods like the Conjugate Gradient algorithm are typically used for this purpose.
As any iterative method, a CG algorithm is quite susceptible to bad condition numbers. It should be noted at this point, that the conditioning is a property of the problem formulation, not the algorithm itself. Of course, the absolute value of $\mathcal{K}$ depends on the norm used, it should therefore be used only for relative statements.

6.2.1 Theory

We consider a linear system of equations

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{6.1}$$

We know, that if $det(\mathbf{A}) = 0$, in other words A is singular, this system does not have a solution.
If, on the other hand, $det(\mathbf{A}) \neq 0$, the convergence of an iterative scheme is determined by the condition number defined as $\mathcal{K} = \|\mathbf{A}\| \, \|\mathbf{A}^{-1}\|$.
Every iterative algorithm basically works by creating an update on the current solution by considering the residual defined as

$$\mathbf{r}^k = \mathbf{b} - \mathbf{A}\mathbf{x}^k \tag{6.2}$$

The accuracy of the solution is usually defined in terms of the residual as

$$\epsilon = \|\mathbf{r}\| = \|\mathbf{b} - \mathbf{A}\mathbf{x}^k\| \tag{6.3}$$

It his well known that the error will decrease at minimum by a factor of $2\left(\frac{\sqrt{\mathcal{K}}-1}{\sqrt{\mathcal{K}}+1}\right)^i$ in k steps.

6.2.2 Conditioning of the FETI problem

The previous section indicated, that the condition number of the linear system solved plays a crucial role for the convergence properties of the FETI solver. In fact, some variations like the FETI-Geneo approach explicitly exploit that knowledge by removing "bad components" from the iteration space with the goal of improving the condition number. This chapter will introduce several vivid problem setups that lead to badly conditioned operators. One major challenge are heterogeneities of the material properties along, or across the interface.
From an engineering point of view, challenges tend to arise, when the behaviour of a substructure on its own, differs strongly from its behaviour when embedded into the whole system.
Figure 6.1 introduces a standard DD problem setup. It consists of a sandwich-structure-like material distribution that resembles real life composite materials. The beam is divided into 16 regular substructures along the x-axis and is subjected to a load on the right face, while mounted in a statically determined manner on the left. The setup was solved with different FETI solvers and the eigenvalue distribution of the iteration matrix analysed in Figures 6.2 and 6.3.

6.3 Heterogeneities

One of the most challenging problems for domain decomposition methods are changing constitutive parameters(material heterogeneities) along, as well across the interface. The reason for this has been explained in Section 6.2.
From an engineering point of view, problems always occur if substructures on their own show a very different behaviour as compared to them being embedded in the total system. Algorithmically, it does make a large difference whether heterogeneities are along or across the interface. This section therefore is denoted to an extensive analysis of the proposed FETI Solvers with regards to the two types. The material patterns used for this study are provided in Figure 6.5.

6.3.1 Heterogeneities across the interface

A typical example of heterogenities across the interface is described in the top of Figure 6.5. The beam is clamped on the left face and subjected to a uniform downward force on the right face

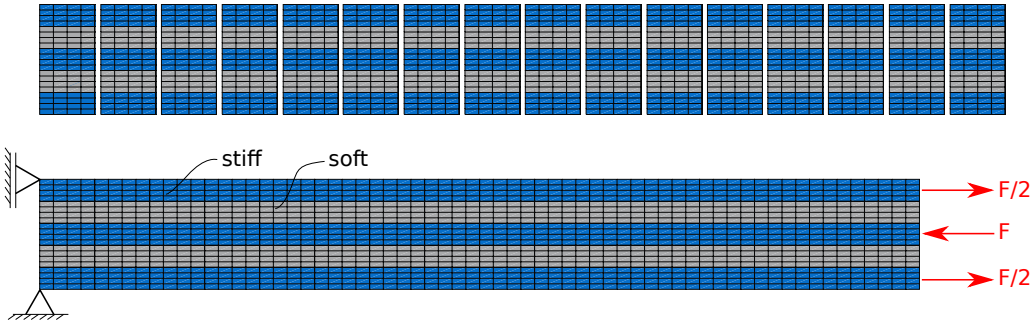


Figure 6.1: *Setup for the eigenvalue analysis regarding heterogeneities along the interface. Note that the forces in this example are self-equilibrated, thus the solution does not contain rigid body modes. For the FETI-1 solver, rigid body components are the only information that is propagated globally. Therefore, calculating this special example leads to the phenomenon depicted in Figure 6.4. The information can only travel one substructure after another during the iterations. The lack of means to transport non rigid-body mode errors globally is one reason, why the FETI-1 algorithm often shows very high iteration counts.*

The reason for the problems with this type of heterogeneity have been explained in Section 3.7.
During the FETI iterations, intermediate solutions for the interface forces are found. Based on this force, a resulting displacement is predicted. In a standard, non scaled FETI-algorithm, this displacement is then applied in equal parts on the two substructure interfaces concerned. If now the substructure on one side of the interface is significantly stiffer, than the other one, this simple 50/50 approach, of course, leads to very wrong results. A possible solution to this problem is K-scaling, which has been derived in [4]. If not mentioned otherwise, this code does always use k-scaling and multiplicity scaling as explained in Section 3.7.
The problem of heterogenities across the interface is visualized in Figure 6.7.

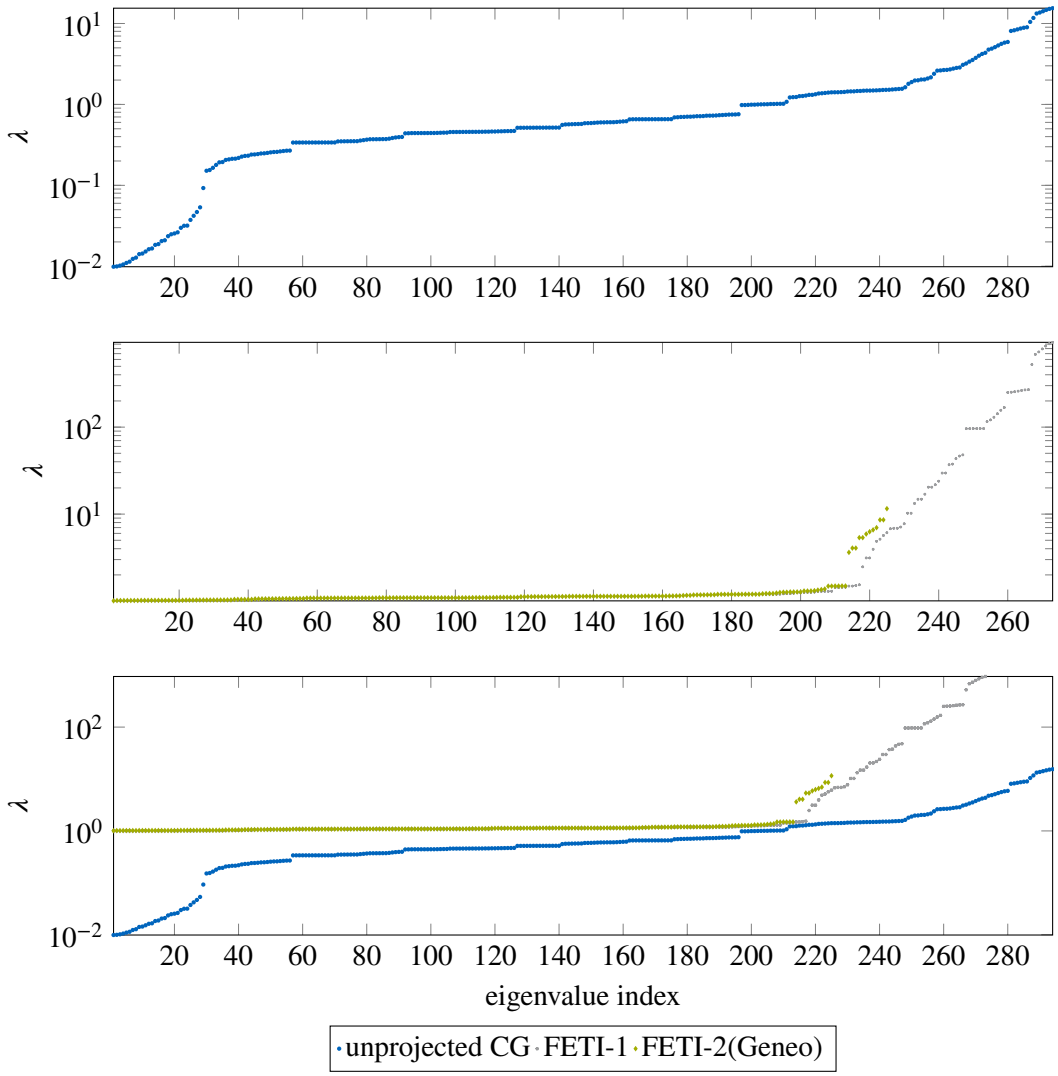


Figure 6.2: *The figures show the eigenvalues of the dual schur operator* **F** *for the different solver types. The top left figure shows the unprojected, unpreconditioned operator. Obviously, the eigenvalue distribution is pretty bad which would, as explained in Section 4.2.1, drastically deteriorate performance. The bottom figure thus shows the dual schur operator after projection to the natural subspace(FETI-1) as well as after the projection into the auxiliary coarse space(FETI-2(GENEO)). Obviously, the projectors catch the extreme, isolated eigenvalues very well. Since the out-projected modes, that are related to the numerically zero eigenvalues in the middle figure have no influence on the following CG algorithm, they have been dropped in the bottom left figure, for better visualization.*
Finally, the bottom figure shows the comparison of the unprojected, unpreconditioned CG with the fully projected ones. Obviously, the condition number has been improved by several orders of magnitudes.

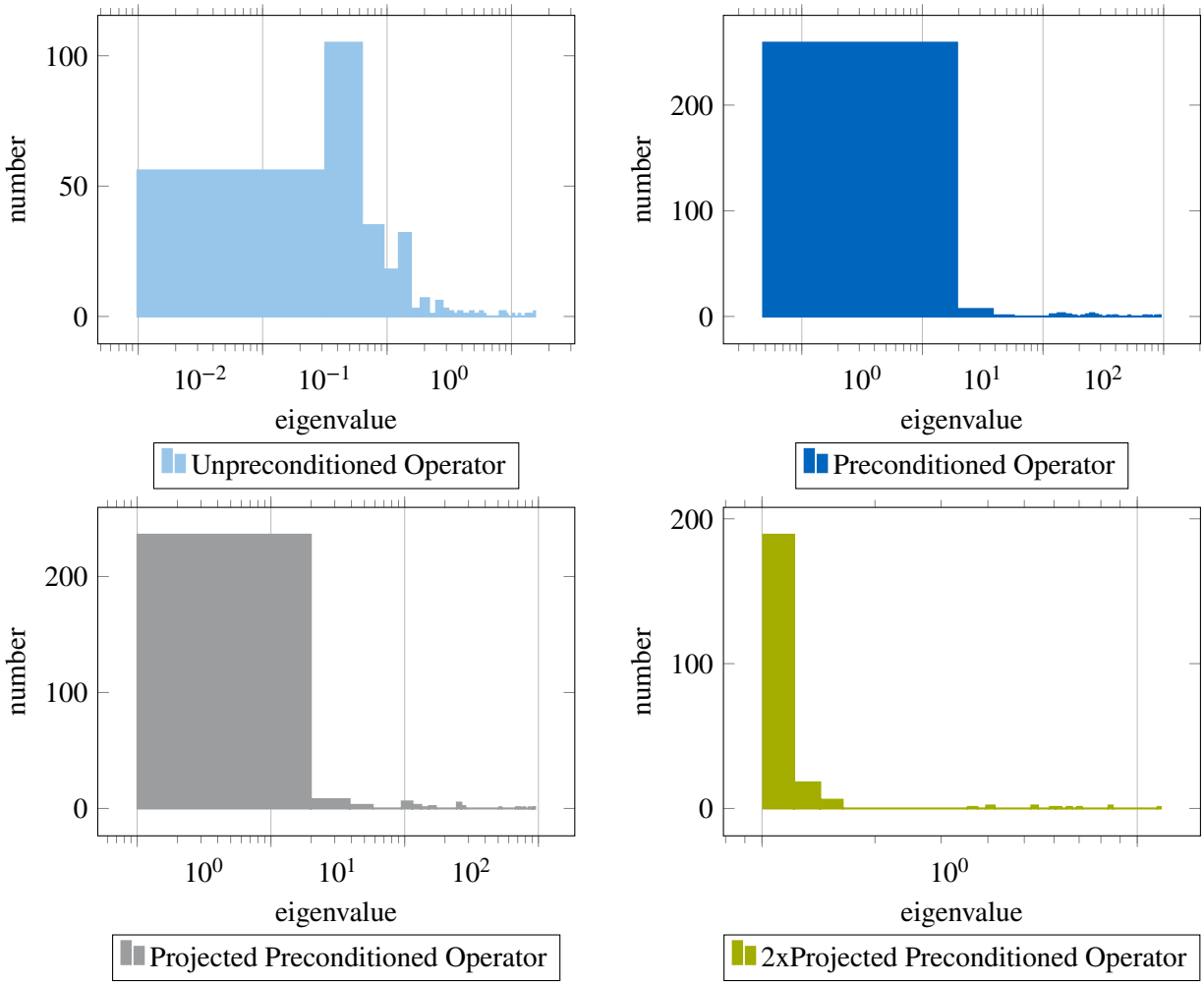


Figure 6.3: *The figure shows the eigenvalues of the raw operator* $\mathbf{A}$*(top left picture), the preconditioned operator* $\mathbf{HA}$*(top right picture), the projected preconditioned operator as in FETI-1* $\mathbf{HA\Pi}$*(bottom left picture) and the 2-level projected, preconditioned operator as in FETI-2* $\mathbf{HA\Pi\Pi}_c$*(bottom right figure), as described in Chapter 4. One clearly sees, that the Projectors do catch bad modes very successfully, thus the condition number is improved.*

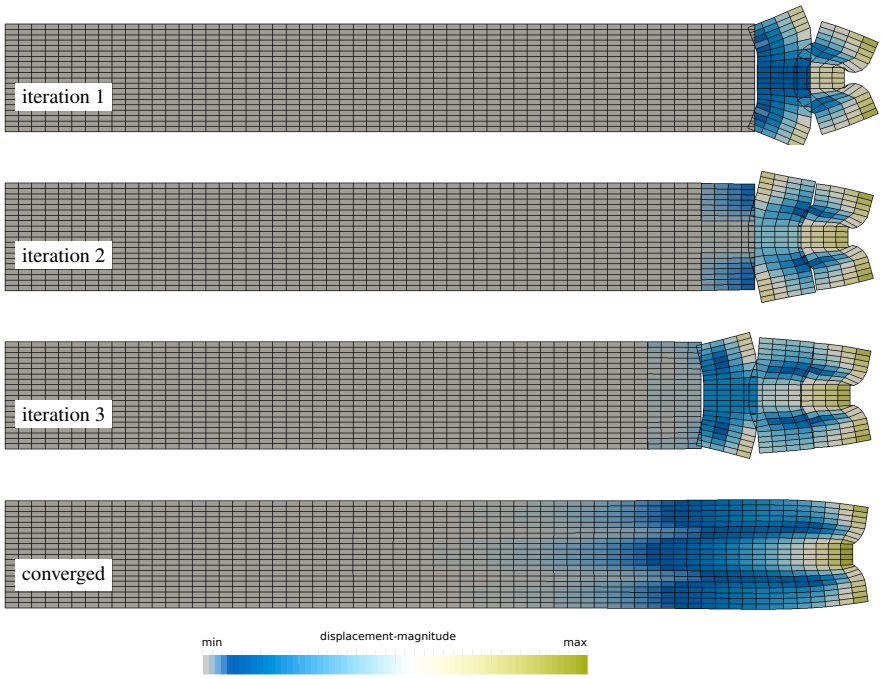
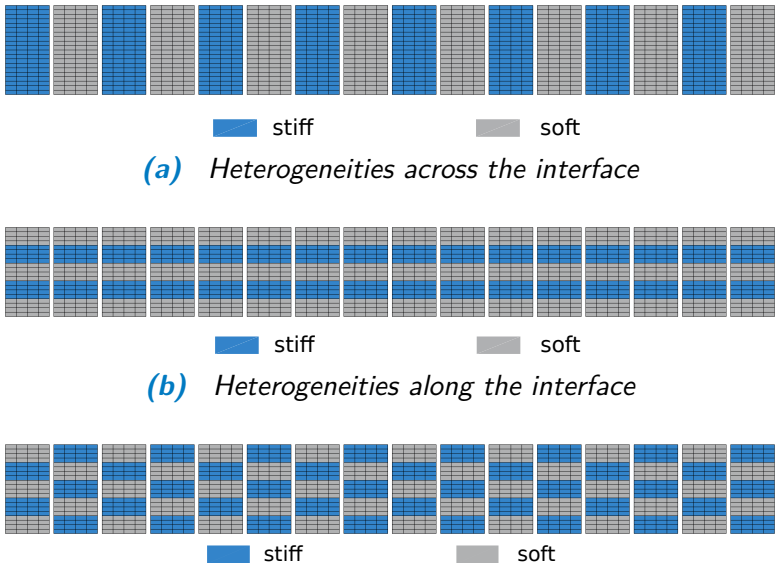


Figure 6.4: *Visualization of the error propagation in FETI-1 for the setup described in Figure 6.1. It can be seen quite clearly, that the error propagation is limited to neighbouring substructures from one iteration to the other. The introduction of an auxiliary coarse grid solves this problem by introducing a mean of propagating the error globally. This example is particularly bad, since the forces are self equilibrated. Thus, FETI-1 has no means of propagating the error globally. For this reasons, information passes only one substructure at a time, which leads to very high iteration numbers for an increasing number of subdomains.*



(a) *Heterogeneities across the interface*

(b) *Heterogeneities along the interface*

(c) *Combined heterogeneities*

Figure 6.5: *Material patterns for the heterogeneity analysis. The substructuring has been done such, that the setup at the top shows heterogenities only across the interface and the middle one only along the interface. The checkerboard pattern at the bottom shows both, heterogeneities across and along the interface. The beams were clamped on the left side and subjected to a uniform downward force on the right.*

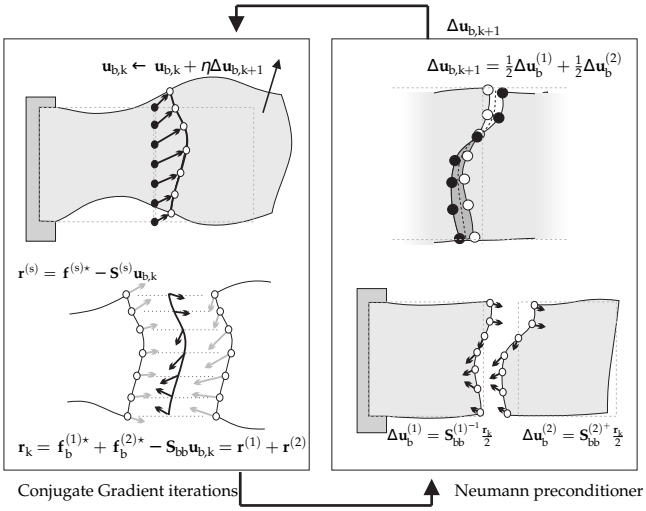


Figure 6.6: *Sketch of the Problem related to heterogenities across the interface. The figure was taken from [17]. Through the simple sum of contributions in the preconditioner, all substructures contribute equally to the search direction. The Neumann step thus equally displaces both interface sides. If, however, one side of the interface shows a much higher stiffness than the other, the natural solution would actually be such, that the soft material shows a much higher displacement than the stiff one. If this discrepancy is not dealt with, convergence rates can be severely damaged. One possible solution would be K-scaling, as described in [18]. Also, thanks to its independent substructure-based search directions, FETI-S is not that susceptible to this problem.*

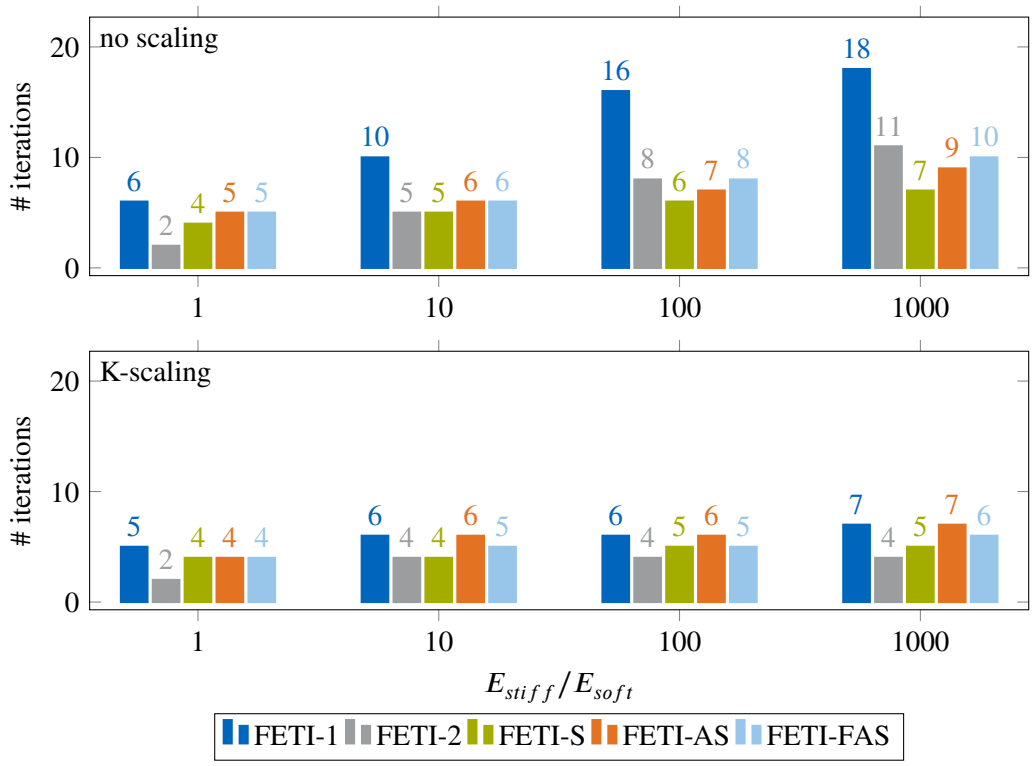


Figure 6.7: *Results for the convergence analysis of the FETI algorithms for materials across the interface. The left figure shows the results without scaling, the left figure shows the results with scaling. The graphs do show two things. First of all, the multi preconditioned schemes seem to be far less effected by this type of heterogeneity, that the singe preconditioned ones. Secondly, the scaling is very effective in reducing the iteration numbers for FETI-1. It also seems that K-scaling is mandatory for the FETI-2(Geneo) approach to work properly here. Since the numerical effort is low, it is therefore always recommended to use a k-scaling when high heterogeneities are present.*

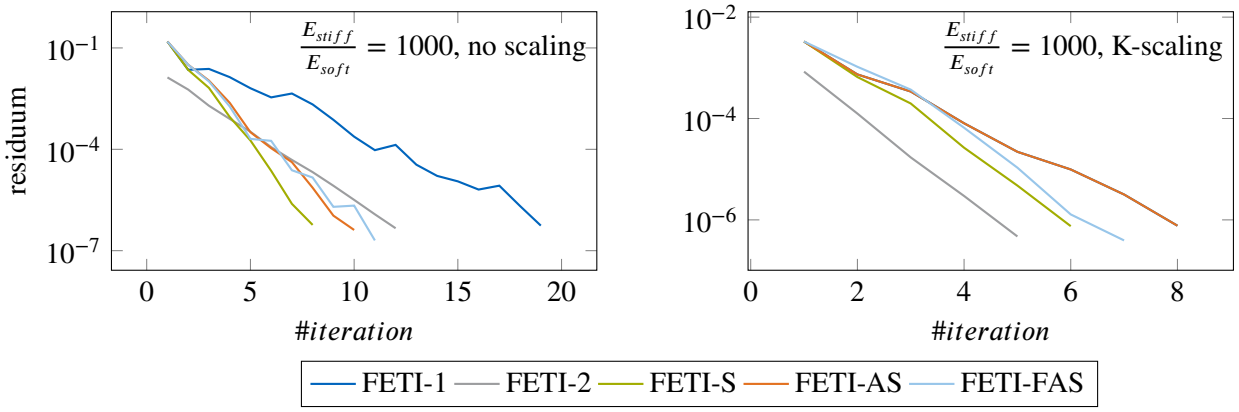


Figure 6.8: *Residuum development for the convergence analysis of the FETI algorithms regarding heterogeneities across the interface, as described in the top of Figure 6.5. Obviously K-scaling drastically improves performance here. The right figure shows that a PCG method(FETI-2) is capable of showing the same convergence rates as an MPCG one(FETI-S) provided that K-scaling is used. In that case, the reduced number of iterations of the FETI-S solver can be attributed to the reduced problem size solemnly. Finally, FETI-S seems to not have benefited much from the Scaling approach. This seems reasonable since substructures interfaces are optimized independently of one another anyway*

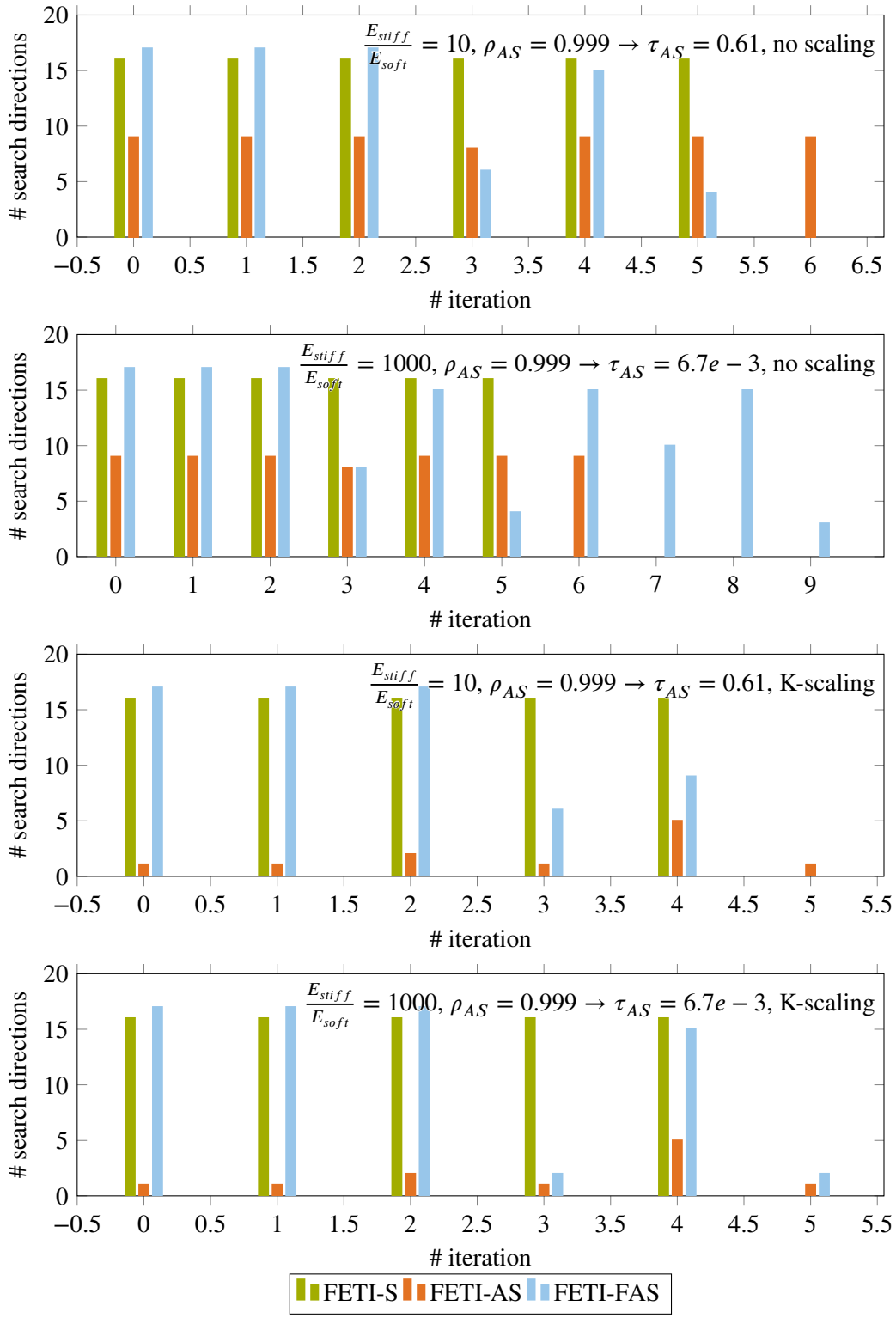


Figure 6.9: *Results for the convergence analysis of the FETI algorithms for materials across the interface, as depicted at the top of Figure 6.5. This figure show the number of search directions used by the different adaptive schemes. Overall, FETI-AS shows superior results here. What is more, K-scaling seems to completely eliminate the problems of heterogeneities across the interface, since in that case, no difference can be observed between a stiffness ratio of 10 and a ratio of 1000.*

6.3.2 Heterogeneities along the interface

Heterogeneities along the interface are somewhat more difficult. The ill-conditioning can not be coped with by simple scaling methods. The material distribution used for this analysis is provided in Figure 6.5. Iteration numbers are provided in Figure 6.10, residua development in Figure 6.11 and a visualization of the number of used search directions in Figure 6.12.

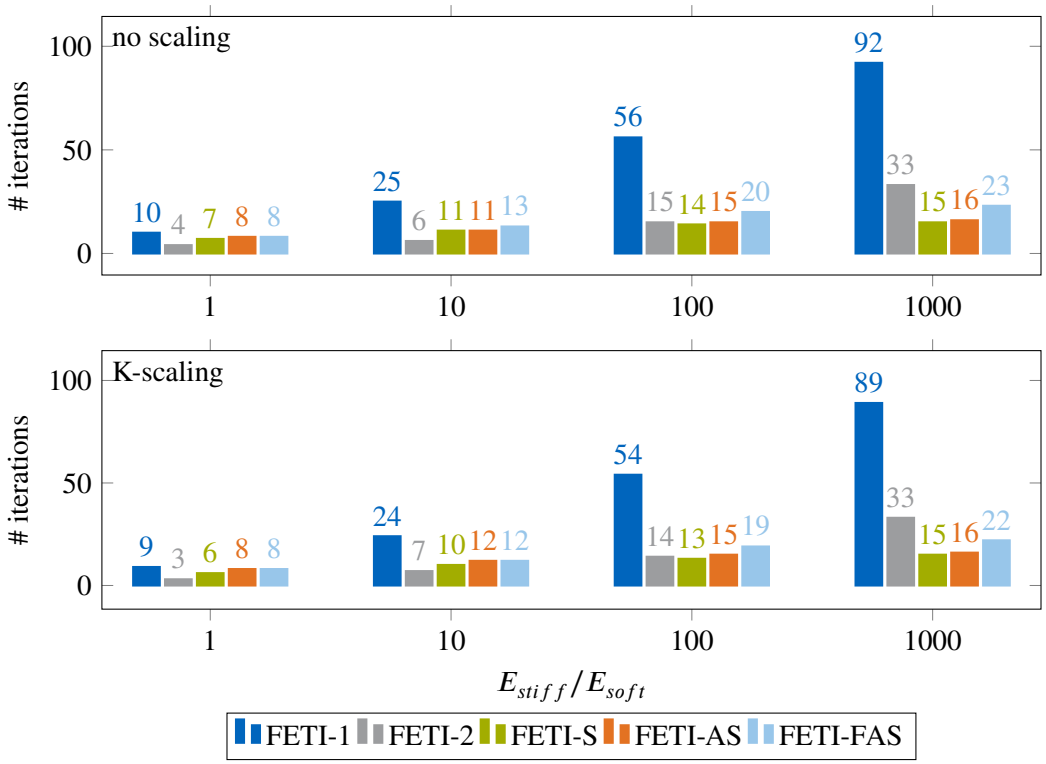


Figure 6.10: *Results for the convergence analysis of the FETI algorithms for materials along the interface. Again, FETI-2 shows severe problems. As expected, scaling does not improve the iteration counts significantly, since it is not designed to cope with this kind of heterogeneity. The multi preconditioned algorithms seem to be able to cope with the problem quite well. In contrast to the heterogeneities across the interface from Figure 6.7, Geneo is capable of effectively detecting the bad modes and iteration counts can therefore be kept low successfully with FETI-2.*

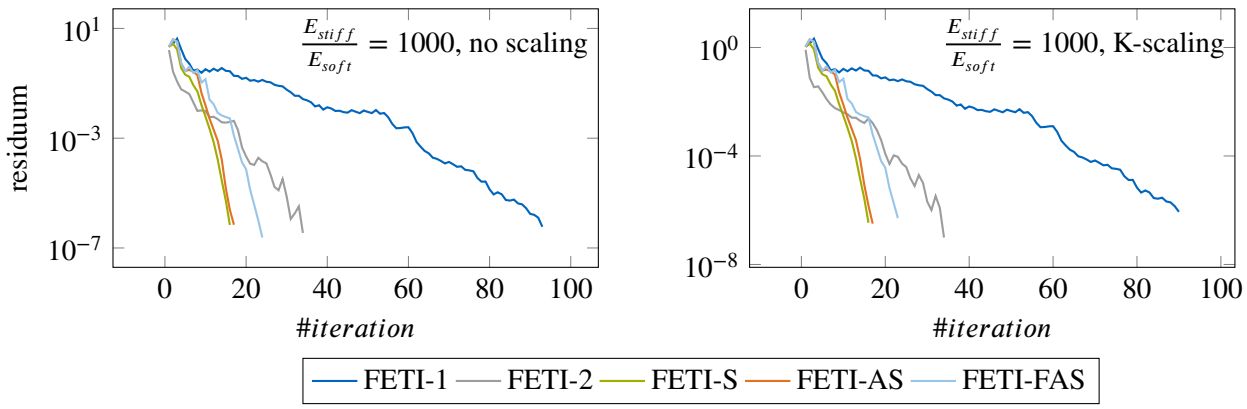


Figure 6.11: *Residuum development for the convergence analysis of the FETI algorithms regarding heterogeneities along the interface, as described at the middle of Figure 6.5. Obviously, K-scaling does not effect the convergence properties at all. What is more, MPCG methods show significantly better results than the PCG ones. In particular, Geneo performed much worse than for the case of heterogeneities across the interface. It remains to be analysed whether a larger coarse space would improve its properties, but, as explained earlier, increasing the coarse problem has significant drawbacks.*

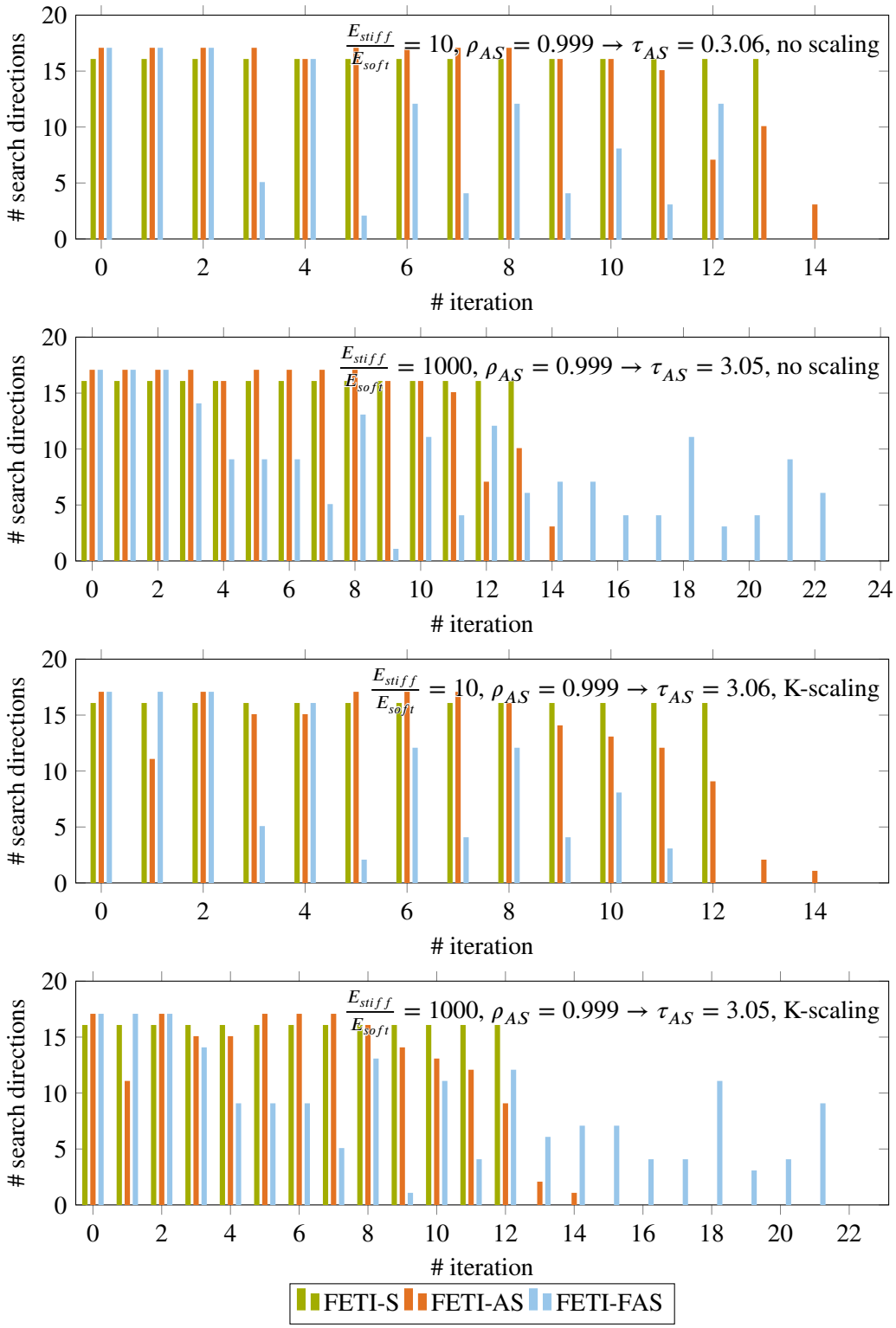


Figure 6.12: *Results for the convergence analysis of the FETI algorithms for materials along the interface, as described at the middle of Figure 6.5. This figures visualizes the number of search directions used by the multi preconditioned schemes during the iterations. At least for the contraction factor of 0.999, the overall improvement is not satisfactory. FETI-AS, however still gives quite nice results with the standard parameters as defined in Section 5.4.*

6.3.3 Combined heterogeneities

In real engineering problems, especially when automatic graph partitioners like Metis are used, one will most likely have to deal with a combined form of heterogeneity. Therefore a setup that contains both shall now be considered. The material pattern is depicted in the bottom of Figure 6.5. Iteration numbers are provided in Figure 6.13, numbers of used search directions in Figure 6.15 and the residuum development in Figure 6.14.

6.4 Partitioning

For a domain decomposition method like FETI, the question of how to form the substructures comes naturally. Partitioning is typically done by automatic graph partitioning algorithms like Metis or Chaco. For some specific cases, however, it can be advantageous to choose partitions manually. Particularly, for the examples of a composite material, one can choose partitions such that the problems of heterogenities along the interface, as described in Section 6.3 are avoided.
Nonetheless, even automatic graph partitioners offer some parameters to choose from. This section is therefore denoted to an analysis of different partitioning approaches.
Figure 6.16 shows the four different partitioning schemes considered. The plate is clamped on the left side and subjected to a uniform surface load in y-direction on the right face.
The partitions have been chosen such, that the interface problem has roughly the same size on all four setups. We can note right at this point, that stripe-like partitioning like in the bottom two subfigures, is far less efficient when it comes to keeping the number of interface unknowns low. For an equally sized interface problem, the number of substructure is more than three times less than for the regular partitioning in the top-left figure, a potential weak-point when it comes to parallizability.
However, as [7] writes, stripe-like partitioning may offer the advantage of a reduced interconnectivity in the substructure stiffness matrices, solving the local problems may becomes faster thus.
The results of this analysis are presented in Figure 6.17.

6.5 Incompressibility

Another well known problem for convergence is material incompressibility. The problem has been thoroughly analysed in [26]. In a linear elasticity, plane strain formulation, one can write

$$\boldsymbol{\sigma} = \boldsymbol{\mathcal{D}}\boldsymbol{\epsilon} = 2\mu\boldsymbol{\epsilon} + \lambda(tr(\epsilon))\mathbf{1} = \frac{E}{1+\nu}\boldsymbol{\epsilon} + \frac{\nu E}{(1+\nu)(1-2\nu)}tr(\boldsymbol{\epsilon})\mathbf{1} \tag{6.4}$$

Obviously, this term becomes problematic if $\nu \to 0.5$. In fact, the condition number of the stiffness matrix increases both with ν and with the model size. The problem has been analysed for the introduced FETI type of solvers. The setup is described in Figure 6.20. No material heterogenities were used here, in order to focus on incompressibility exclusively.

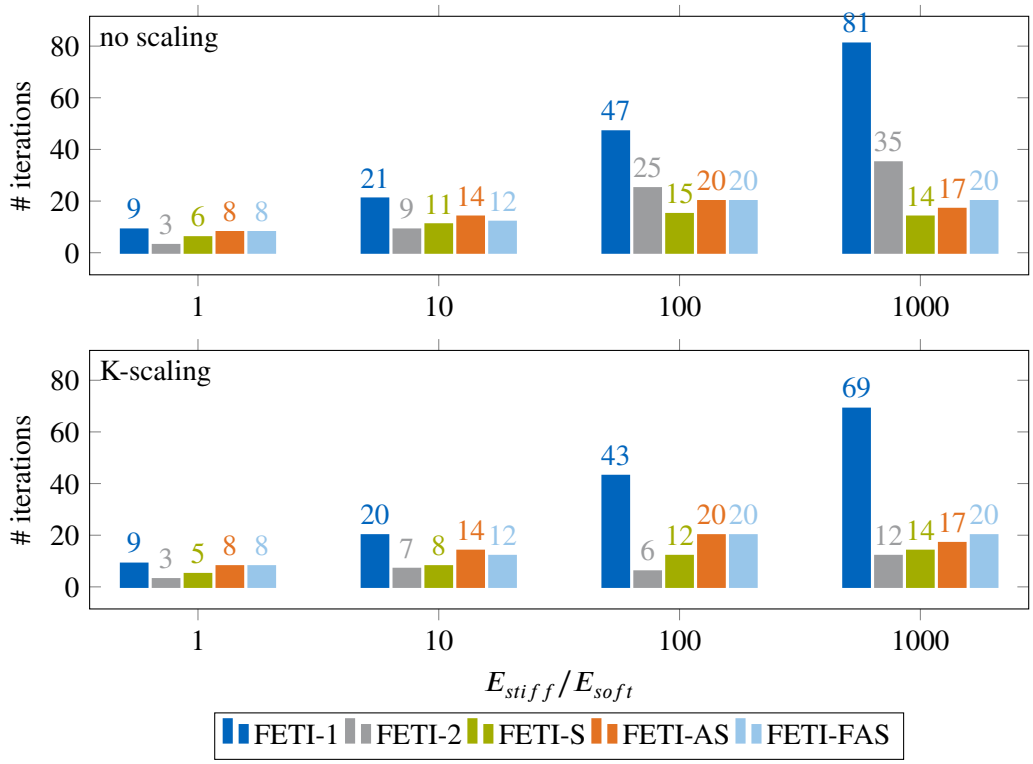


Figure 6.13: *Results for the convergence analysis of the FETI algorithms for materials across and along the interface. As expected, FETI-1 performs very poorly. K-scaling does not really improved the iteration numbers, except for FETI-2, where we have already seen in Figure 6.12 that it is not able to build an appropriate coarse grid for the case of heterogenities. As could be expected from Figures 6.7 and 6.10, the multi-preconditioned algorithms show superior properties.*

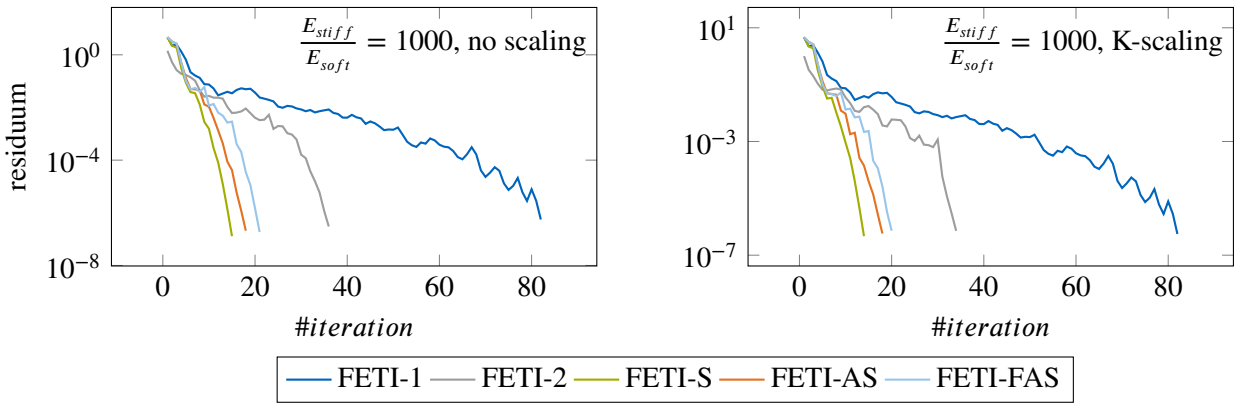


Figure 6.14: *Residuum development for the convergence analysis of the FETI algorithms regarding combined heterogeneities, as depicted at the bottom of Figure 6.5. This figure shown the development of the residual over the iterations for al solvers. Obviously, K-scaling is not capable of significantly reducing the the total number of iterations, except for FETI-2(Geneo), where we have already observed that it seems to be necessary for the construction of an effective coarse space.*

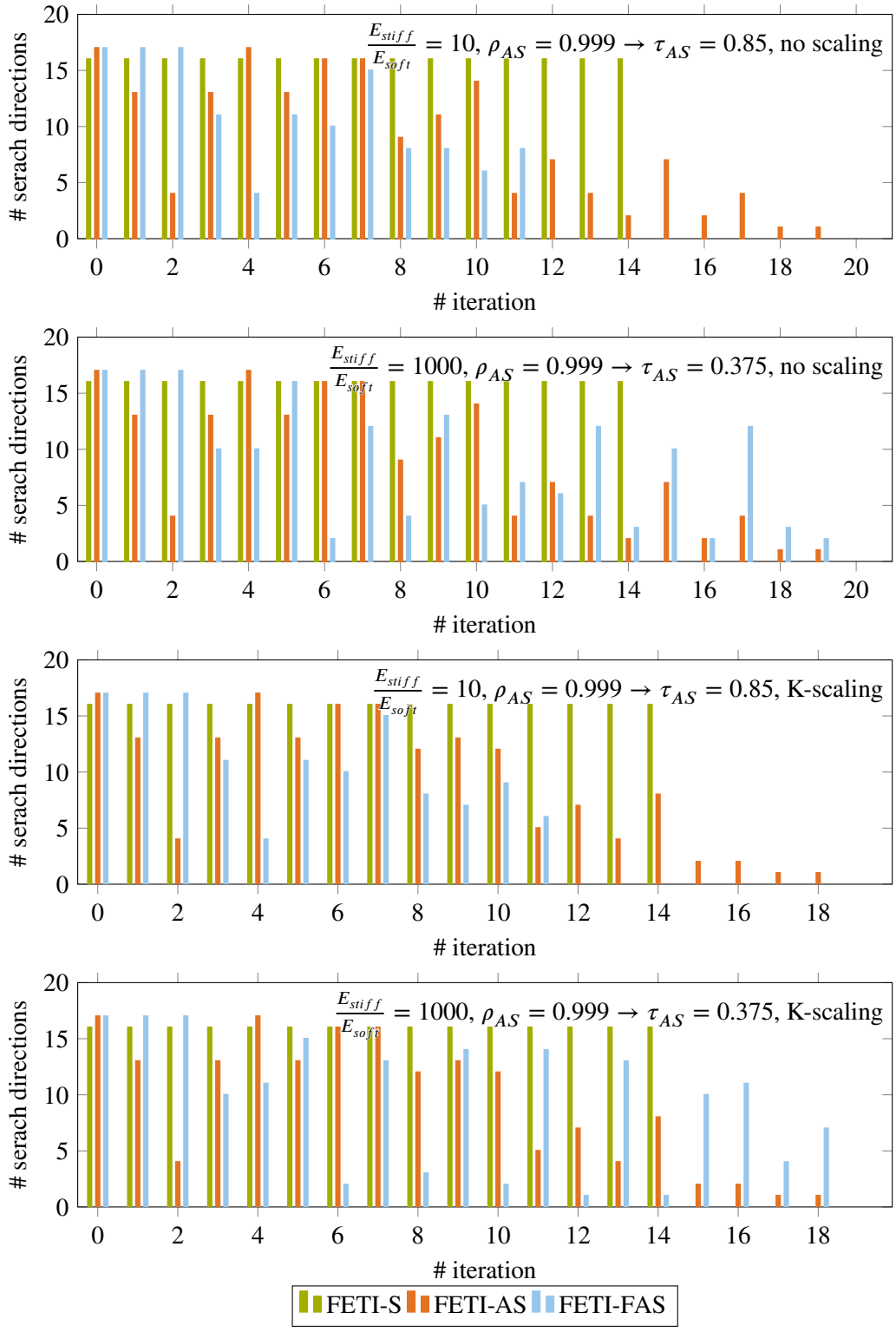
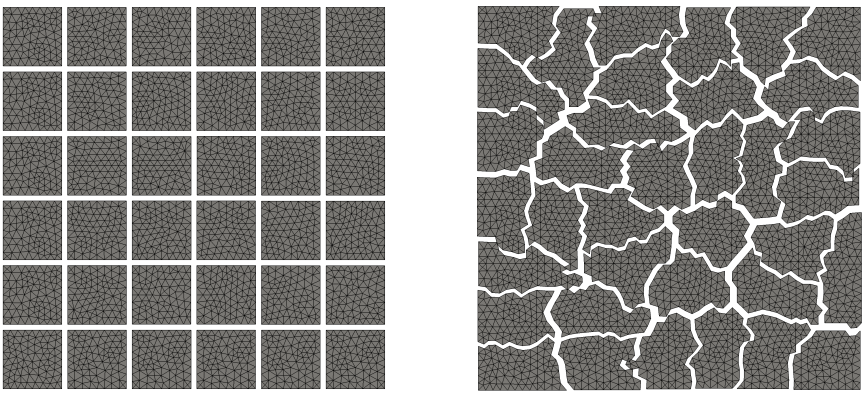
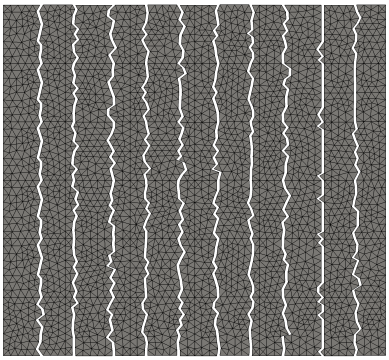


Figure 6.15: *Results for the convergence analysis of the FETI algorithms regarding combined heterogeneities, as depicted at the middle of Figure 6.5. This figure shows the number of search directions used by the multi preconditioned solvers for all iteration steps. Both FETI-AS ans FETI-FAS were capable of significantly reducing the total number of search directions used.*

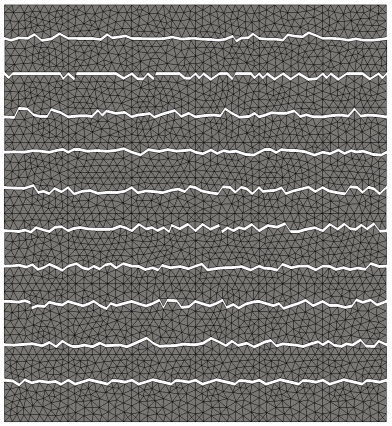


(a) regular partitioning

(b) automatic partitioning



(c) vertical stripes



(d) horizontal stripes

Figure 6.16: *Different partitioning schemes. The top left example was manually partitioned into equally sized substructures keeping the number of interface unknowns low. The top right example was partitioned using chaco in GMSH. Finally, the bottom two examples were created using the Simple Partitioner Plug-in of GMSH. Here, the number of subdomains had to be drastically reduced, in order to get roughly the same size for the interface problem as in the top two examples. No material heterogenities were considered here. The results of this analysis are presented in Figure 6.17*

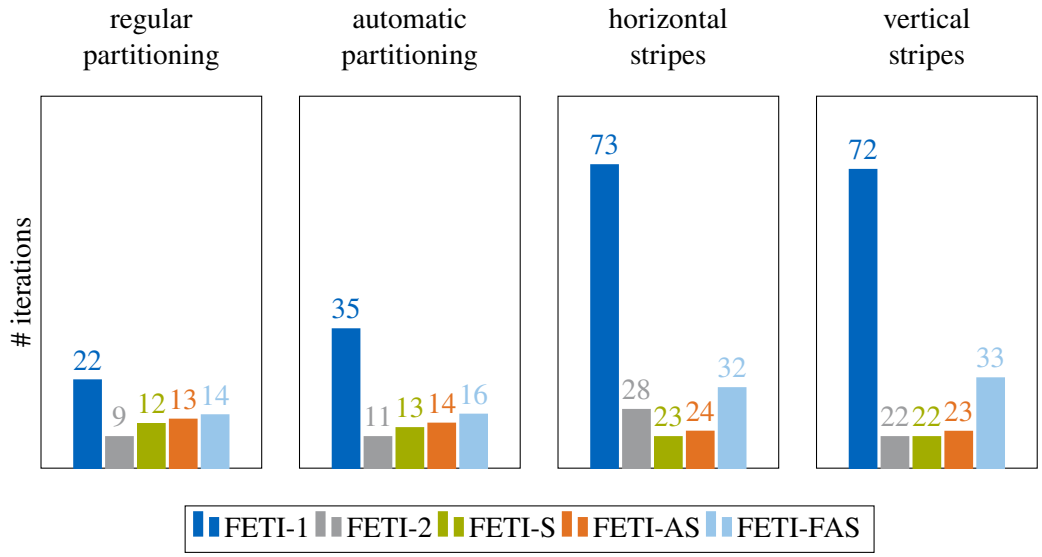


Figure 6.17: *Results for the analysis of different partitioning schemes as described in Figure 6.16. Firstly, we note that no dramatic difference between the regular partitioning and the automatic partitioning can be observed. Secondly, the stripe-like partitions performed far worse than the first two ones. Although less susceptible than the others, even the multi preconditioned schemes showed an increase of close to 100% in that case. Event though, it was not possible to collect actual CPU-times, due to the Matlab implementation, taking into account, that the total number of substructures had to be significantly reduced for the stripe-like setups, it can be safely assumed that the advantage of reduced interconnectivity in the local stiffness matrices for stripe-like partitions does not justify choosing such a partitioning scheme.*
A comparison with regards to the number of search directions between the multi preconditioned algorithms for this example is provided in Figure 6.19.

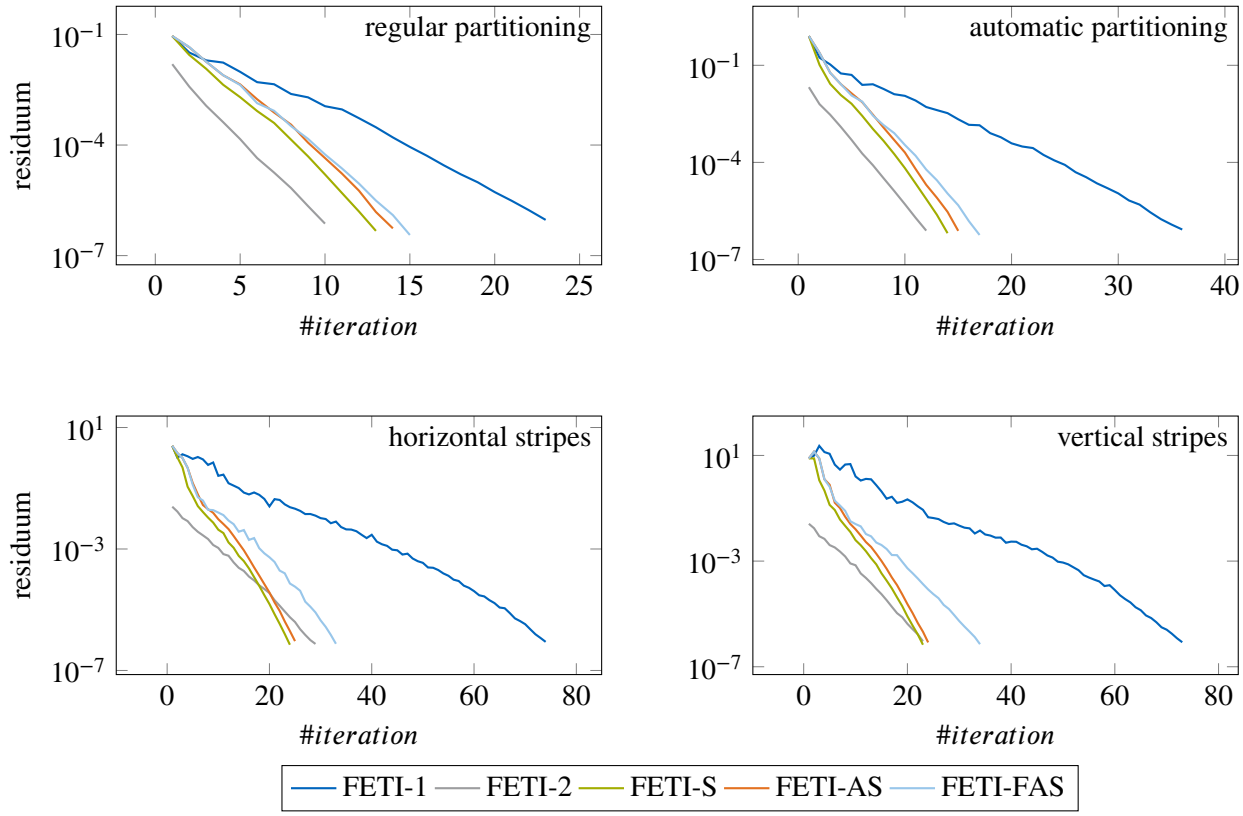


Figure 6.18: *Residual development for the partitioning scheme analysis described in Figure 6.16. One can note that FETI-2 with a Geneo coarse grid shows good convergence rate for the first two partitioning schemes, whereas the FETI-2 convergence rate is significantly lower for stripe-like partitioning schemes. It shall be noted, that in the first to schemes, 6 Geneo modes were used per substructure, whereas the last two ones were calculated with 18 Geneo modes per substructure, so that the overall size of the coarse problem is the same for all four examples.*

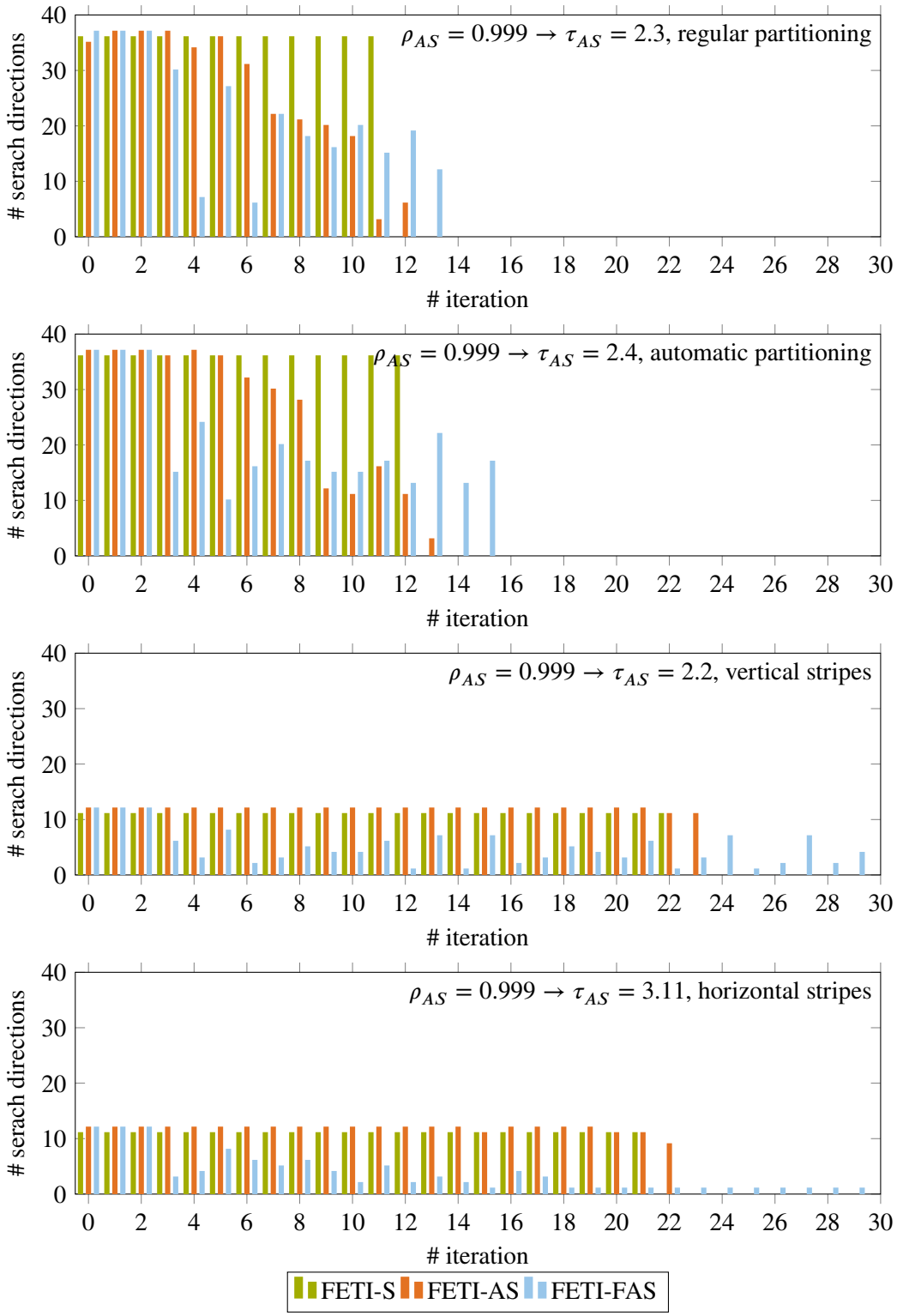


Figure 6.19: *Results for the analysis of different partitioning schemes as described in Figure 6.16. Particularly, this figure shows the number of chosen search directions for FETI-S and FETI-FAS. Although probably not optimal, the FETI-FAS algorithm is quite efficient in reducing the total number of search directions during the iterations. All four examples have been calculated with the default FETI-FAS parameters, as described in Section 5.4.1.*

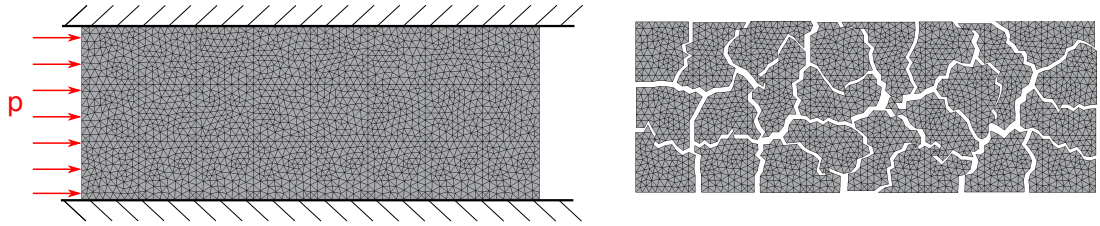


Figure 6.20: *Setup for the incompressibility analysis. The beam is divided into 24 substructures as shown in the right subfigure. Both the bottom and the top surface are clamped. A constant pressure is applied on the left side. The results for different Poission numbers are shown in Figure 6.21*

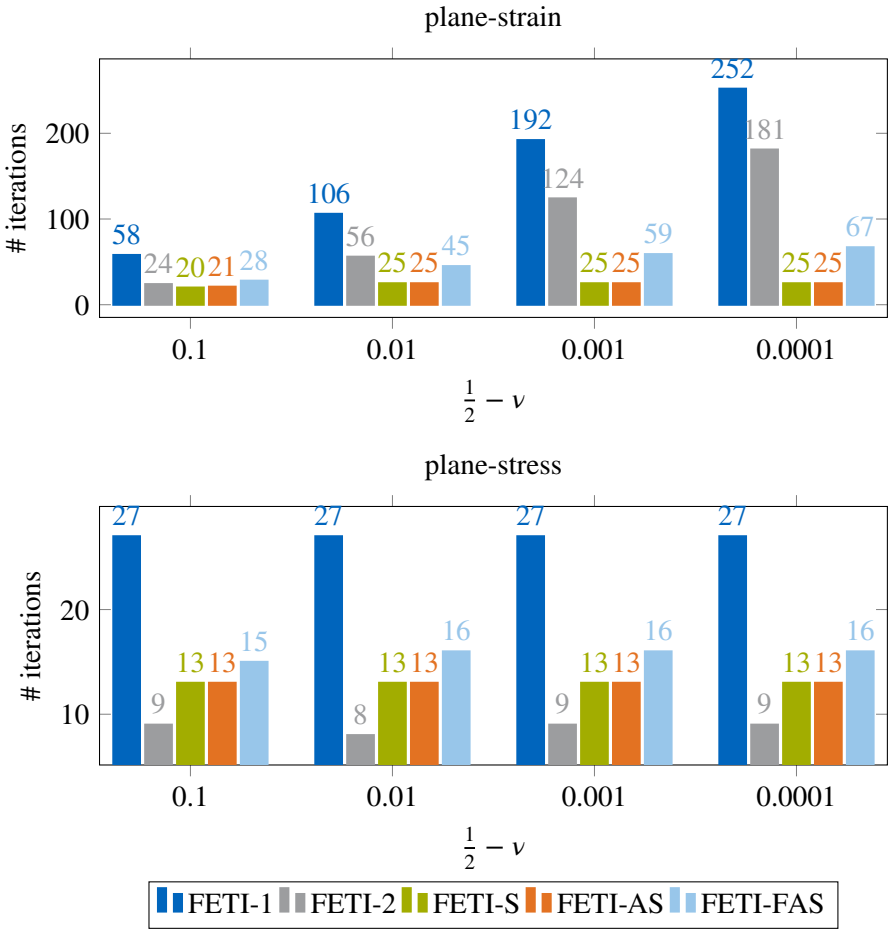


Figure 6.21: *Results for the incompressibility analysis described in Figure 6.21. Since incompressibility is only a problem in plane-strain calculations the same calculations with a plane-stress formulations are provided in the right figure. The figures clearly show, that the standard FETI-1 solver is very vulnerable in nearly incompressible simulations.*

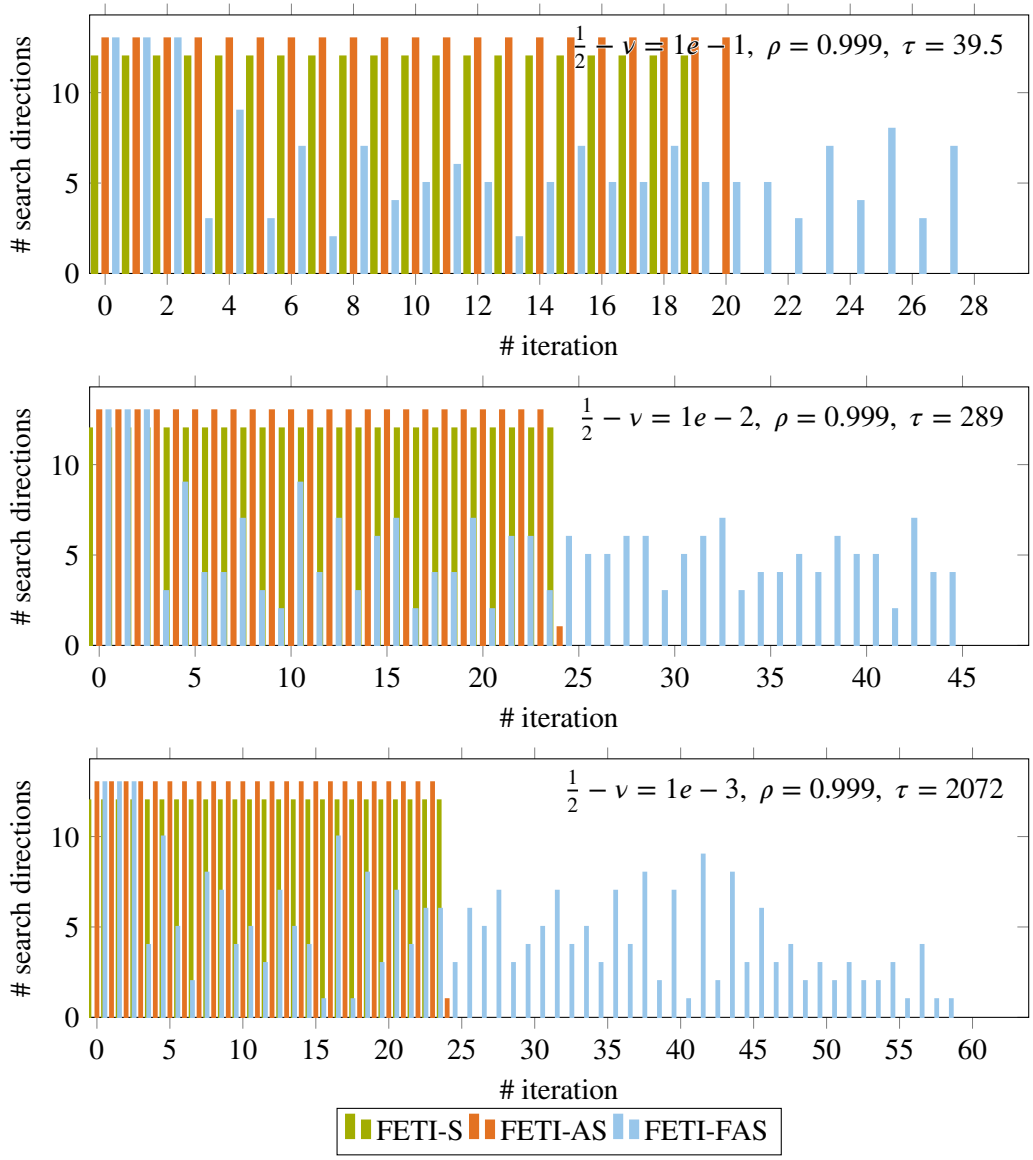


Figure 6.22: *Results for the incompressibility analysis described in Figure 6.21. This figure shows the number of search directions used by the multi preconditioned schemes during the iterations. FETI-AS is not able to reduce search at all with the standard parameters here. FETI-AS does, but it tends to overshoot when approaching $\nu \to 0.5$. Overall, incompressibility seem to be a problem, where the adaptive schemes show no advantage compared to FETI-S*

6.6 Inclusion

The previous sections have indicated that parameter jumps along or across the interface pose a serious problem for DD-type of solvers.
One can however also create homogeneous interface examples, that still lead to highly increased iteration numbers. From an engineering point of view, this is due to substructures that show fundamentally different stiffness properties when considered separately than when considered in composite with the other substructures. The following considerations are inspired by [9]. The setup is briefly described in Figure 6.23 and the numerical results are presented in Figure 6.24.
The number of search directions used my the multi preconditioned solver is depicted in Figure 6.26. Moreover, an visualization of the development of the residuum is provided in Figure 6.25.

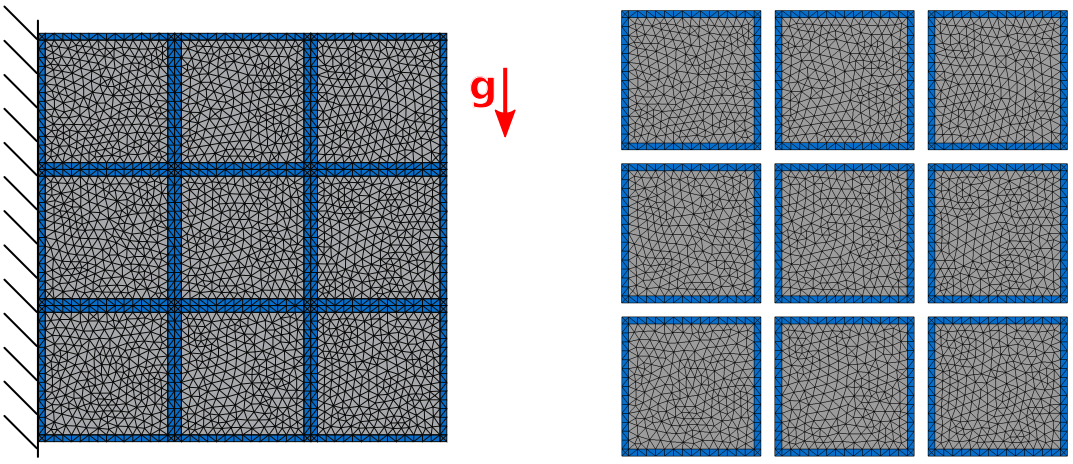


Figure 6.23: *Setup for the inclusion analysis. The left figure shows the material distribution. The right figure depicts the partitioning into substructures. As shown, every substructure consists of a lattice around the substructure border and another material within (inclusion). An setup like this as coefficient jumps neighter along nor across the interface. Nevertheless FETI-solver are challenged by it. Figure 6.24 shows the iteration counts for the different FETI-algorithms*

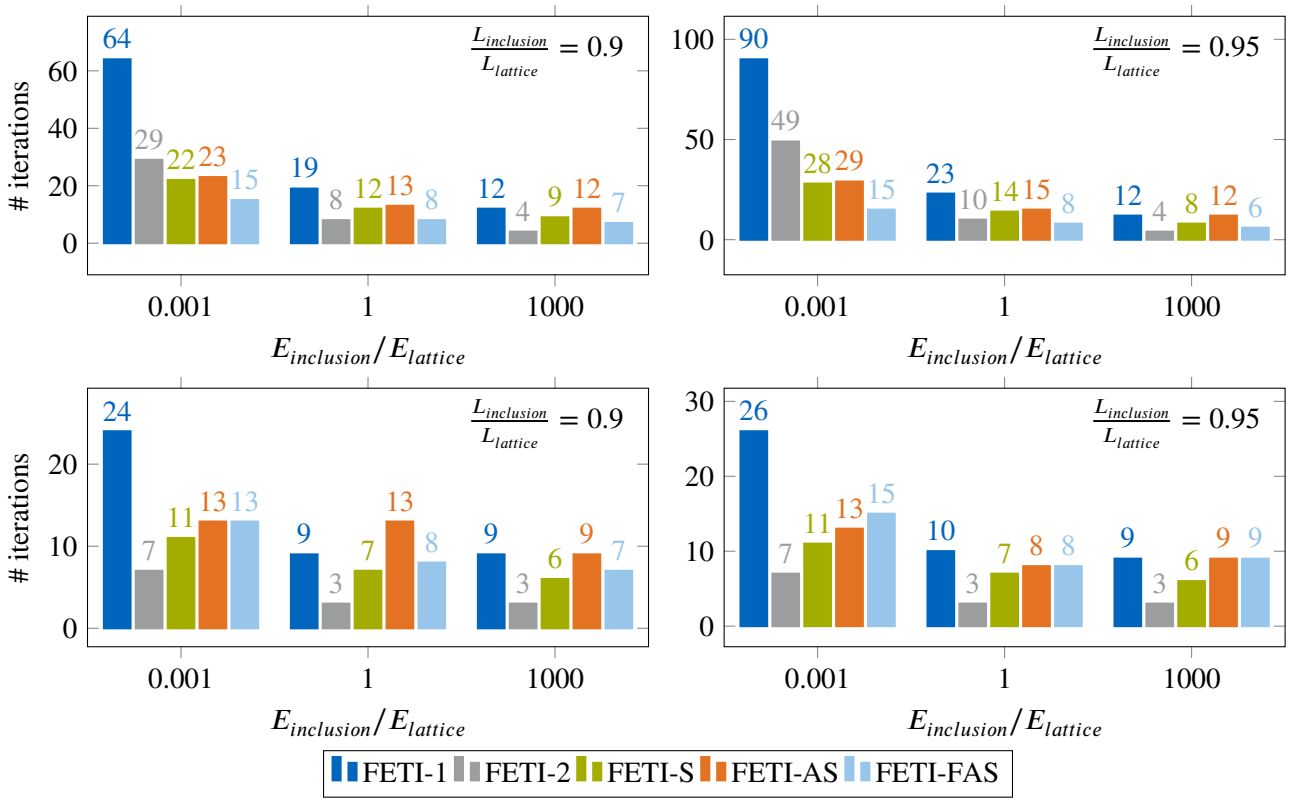


Figure 6.24: *Results for the inclusion problems as described in Figure 6.23. The top two figures show the calculations with the default lumped preconditioner, the bottom ones with a full Dirichlet preconditioner(see Section 3.7. The left figures were calculated with inclusion to substructure ration of* $\frac{L_{inclusion}}{L_{substructure}} = 0.9$*, the right figures with* $\frac{L_{inclusion}}{L_{substructure}} = 0.95$*. The graphs show that FETI-S and FETI-2(Geneo) show less sever increases in the iteration numbers than FETI-1, when the stiffness ratio* $\frac{E_{inclusion}}{E_{lattice}}$ *is decreased. As expected, the problematic case is the one with soft inclusions inside a stiff lattice. In that case, the solver drastically underestimates the deformation. The overestimation of deformation for the contrary case is not critical. What is more, one can also notice that FETI-2 is obviously not able to correctly detect the bad modes, when a lumped preconditioner is used. This is intuitive when looking at the very definition of the two preconditioner types in Equations (3.45)(3.47)(3.52). Since the lumped preconditioner neglects the Inner stiffness components* $\mathbf{K}_{ii}$*, no information abut the inner stiffness change enters the Geneo algorithms.*

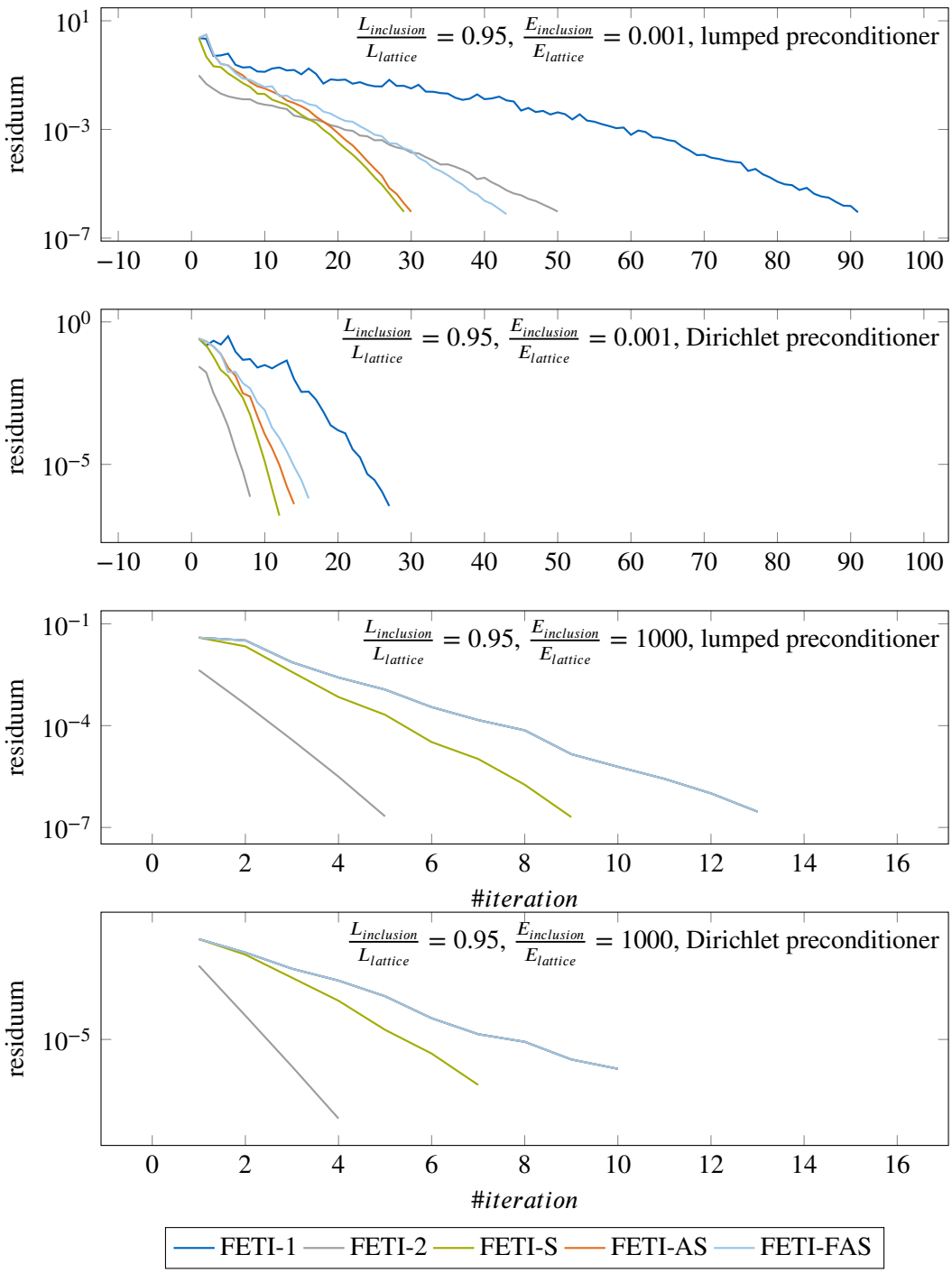


Figure 6.25: *Residuum development for the inclusion analysis with lumped and Dirichlet preconditioners and soft as well as stiff inclusions. The case of hard inclusions obviously does not pose any problems. As mentioned in Figure 6.24 the top two figures again confirm, that Geneo can only play out its advantages in terms of convergence rate, when the Dirichlet preconditioner is used. Otherwise it is not able to detect soft inclusions. However, the Dirichlet preconditioner comes with a significantly increased price, as outlined in Section 3.7.*

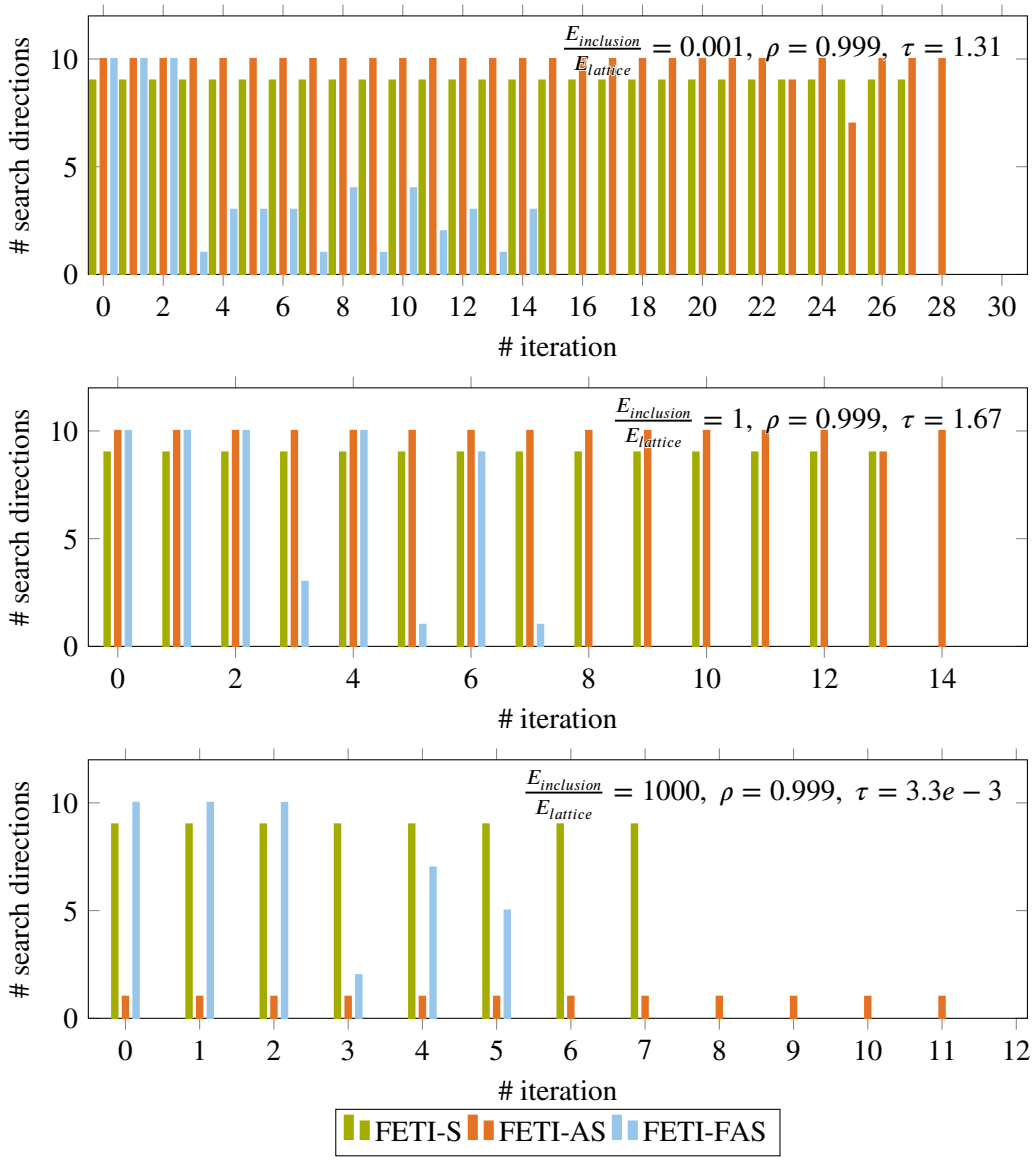


Figure 6.26: *Results for the inclusion problems as described in Figure 6.23 with a lumped preconditioner. This plots compare the number of search directions used during the iterations of the FETI-S and the FETI-FAS algorithm. For the top two figures, the FETI-FAS algorithms shows mediocre results.For the bottom figure, FETI-FAS successfully detects the expandability of almost all search directions ans thereby reduced the computational effort significantly. The application of FETI-FAS did not pose any problems here, since the standard parameters as introduces in Section 6.1 were used.*

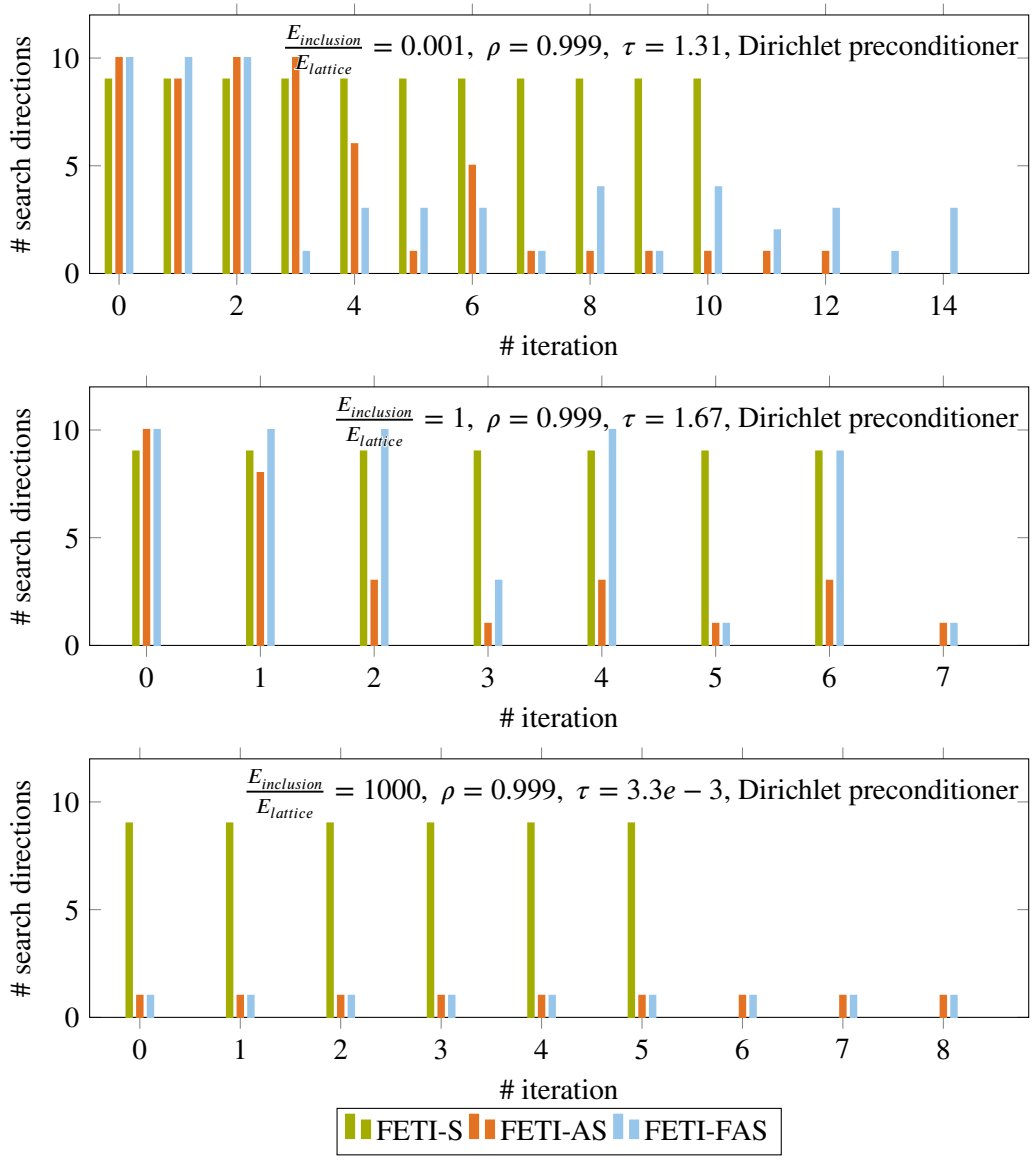


Figure 6.27: *Results for the inclusion problems as described in Figure 6.23 wit a Dirichlet Preconditioner. This plots compare the number of search directions used during the iterations of the FETI-S and the FETI-FAS algorithm. When compared to Figure 6.26, we note that the appropriate choice of the contraction factor τ is obviously, among others, dependent on the preconditioner type used, while FETI-FAS performs reasonably well for both cases.*

Chapter 7

Summary

In conclusion, the FETI-method has been introduced as a prominent domain decomposition method for structural mechanics problems. The underlying motivation for domain decomposition methods has been described in Chapter 1 and a quick comparison to other DD methods has been drawn.
Chapter 3 introduced the standard FETI method(FETI-1), as first described by [7] and derived all relevant equations. It has then been shown, that the FETI-1 method shows some severe deficiencies, especially for the problems mentioned in Chapter 6.
Variations of the FETI-1 method, namely FETI-2 and FETI-S have therefore been described in Chapter 5 and their theoretical potentials have been discussed. Inspired by[22] the concept of an Adaptive Multi Preconditioned Conjugate Gradient algorithm has then been adapted to the FETI-method leading to the FETI-A algorithm. To my knowledge this has not been done before.
Since the FETI-AS algorithm depends on a user specified threshold τ which is delicate to choose, a variation of the algorithm, the Fast Adaptive Simultaneous FETI solver(FETI-FAS) has been proposed in Section 5.4.
All five algorithms have been thoroughly discussed and summarized in Chapter 5.
Chapter 6 has then been dedicated to an intensive analysis and comparison of the five described FETI-algorithms. For that purpose four setups, covering the most challenging situations for DD methods, have been devised and solved with each algorithm. Several conclusions have been drawn.
First of all, the before mentioned inferiority of the FETI-1 method has been confirmed in each setup. Moreover, it has been shown, that FETI-S and FETI-2(Geneo) show comparable results, with slight advantages for the FETI-S(Geneo) approach when it comes to the total iteration number. However, as described in Chapter 5 the Geneo approach involves the solution of eigenvalue problems on the substructure interfaces. The overall conclusion therefore was, that the slightly reduced iteration count of the Geneo approach, does not justify the large computational overhead as compared to the FETI-S algorithm, especially with the efficient implementation as outlined in Section 5.3.
Moreover, the examples generally confirmed that one usually can neglect a large part of the search directions in the FETI-S solver, and still get reasonable iteration numbers. This was the main idea behind the Adaptive Simultaneous FETI solvers. It has also been shown that the choice of an appropriate parameter τ in the FETI-AS algorithm is a delicate issue. While prescribing a desired contraction factor, instead of directly prescribing τ mediated the problem, a general recommendation for τ can still not be drawn.
Our simulations thus suggested that the FETI-FAS solver can be favoured as a more reasonable, easier to choice that performs very well in most cases.

For the purpose of this thesis, an object oriented finite element code (FEMAC) has been written in Matlab[11].

Chapter 8

Outlook

The potentials of the FETI-As approaches have been vividly demonstrated in Chapter 6.
Subsequent research should thus be focused on the development of the Simultaneous FETI methods. When it comes to FETI-AS, a closer look can be paid upon the appropriate choice of τ. It seems reasonable that better rules can be derived.
WE also propose to focus on the herein newly introduced FETI-FAS method. The solver parameters can definitely still be tuned, and an application to dynamic problems remains to be done.
This paper focused on a simple small strain, linear finite element formulation. As a first step, the method should thus be generalized to a non-linear, dynamic formulation.
Since the herein used code was written in Matlab, meaningful CPU-time analyses could not have been carried out and thus remain to be done too.
Special attention should also be given to an profound analyses of the parallel scalability of the method.
Further analyses should be carried out on an efficient low-level implementation like C++ or FORTRAN.

Bibliography

[1] O. Axelsson and I. Kaporin. “Error norm estimation and stopping criteria in preconditioned conjugate gradient iterations”. In: *Numerical Linear Algebra with Applications* 8.4 (2001), pp. 265–286.

[2] R. Bridson and C. Greif. “A Multi-Preconditioned Conjugate Gradient Algorithm 2217”. In: *Computer* (2006), pp. 1–12.

[3] C. Farhat and P. Chen. “A unified framework for accelerating the convergence of iterative substructuring methods with Lagrange multipliers”. In: *International Journal for Numerical Methods in Engineering* 42.January 1997 (1998), pp. 257–288.

[4] C. Farhat and P.-s. Chen. “a Scalable Lagrange Multiplier Based Time-Dependent Problems”. In: *International Journal for Numerical Methods in Engineering* 38.December 1994 (1995), pp. 3831–3853.

[5] C. Farhat, L. Crivelli, and F.-X. Roux. *A transient FETI methodology for large-scale parallel implicit computations in structural mechanics*. 1992.

[6] C. Farhat, A. Macedo, M. Lesoinne, F.-X. Roux, F. Magoulès, and A. D. L. Bourdonnaie. “Two-level domain decomposition methods with Lagrange multipliers for the fast iterative solution of acoustic scattering problems”. In: *Computer Methods in Applied Mechanics and Engineering* 184.2-4 (2000), pp. 213–239.

[7] C. Farhat and F.-X. Roux. “A method of finite element tearing and interconnecting and its parallel solution algorithm”. In: *International Journal for Numerical Methods in Engineering* 32.6 (1991), pp. 1205–1227.

[8] P. Gosselet. “Non-overlapping domain decomposition methods in structural mechanics”. In: *Archives of computational methods in engineering* 13.July 2005 (2006), pp. 515–572. arXiv: 1208.4209.

[9] P. Gosselet. “Simultaneous-FETI and Block-FETI: robust domain decomposition with multiple search directions.” In: (2015), pp. 1–20.

[10] P. Hansbo. “Nitsche’s method for interface problems in computational mechanics”. In: *GAMM-Mitteilungen* 28.2 (2005), pp. 183–206.

[11] Institute of Applied Mechanics - Technical University of Munich. *FEMAC*. Munich, 2016.

[12] S. Kaniel. “Estimates for some computational techniques in linear algebra”. 1966.

[13] J. Mandel and R. Tezaur. “Convergence of a substructuring method with Lagrange multipliers”. In: *Numerische Mathematik* 73.4 (June 1996), pp. 473–487.

[14] G. E. Moore. *Cramming more components onto integrated circuits (Reprinted from Electronics, pg 114-117, April 19, 1965)*. Tech. rep. 1. 1965, pp. 82–85.

[15] J. Nitsche. “Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind”. In: *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 36.1 (1971), pp. 9–15.

[16] D. J. Rixen. “Substructuring and Dual Methods in Structural Analysis”. In: (1997).

[17] D. J. Rixen. “A domain decomposition interface solver with multiple direction of descent for heterogeneous problems”. In: *84th Annual Meeting of the International Association of Applied Mathematics and Mechanics, GAMM* (2013), pp. 18–22.

[18] D. J. Rixen and C. Farhat. “A simple and efficient extension of a class of substructure based preconditioners to heterogenous structural mechanics problems”. In: *International Journal for Numerical Methods in Engineering* 516.April 1998 (1999), pp. 489–516.

[19] D. P. Rodgers. “Improvements in multiprocessor system design”. In: *ACM SIGARCH Computer Architecture News* 13.3 (June 1985), pp. 225–231.

[20] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Jan. 2003.

[21] N. Spillane and D. Rixen. “Automatic spectral coarse spaces for robust finite element tearing and interconnecting and balanced domain decomposition algorithms”. In: *International Journal for Numerical Methods in Engineering* 95.11 (Sept. 2013), pp. 953–990. arXiv: 1010.1724.

[22] N. Spillane. “An Adaptive MultiPreconditioned Conjugate Gradient Algorithm”. In: *SIAM Journal on Scientific Computing* 38.3 (2016), A1896–A1918.

[23] N. Spillane, V. Dolean, P. Hauret, F. Nataf, C. Pechstein, and R. Scheichl. “Abstract robust coarse spaces for systems of PDEs via generalized eigenproblems in the overlaps”. In: *Numerische Mathematik* 126.4 (Apr. 2014), pp. 741–770.

[24] E. Tonti. *The Mathematical Structure of Classical and Relativistic Physics*. 2013.

[25] L. B. da Veiga, L. F. Pavarino, S. Scacchi, O. B. Widlund, and S. Zampini. *Isogeometric BDDC Preconditioners with Deluxe Scaling*. 2014.

[26] B. Vereecke, H. Bavestrello, and D. Dureisseix. “An extension of the FETI domain decomposition method for incompressible and nearly incompressible problems”. In: *Computer Methods in Applied Mechanics and Engineering* 192.31-32 (2003), pp. 3409–3429.

Disclaimer

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

Garching, 19. 09. 2016

(Signature)