Hybrid 3D simulation methods for the damage analysis of multiphase composites

(Hybride 3D Simulationsmethoden zur Abbildung der Schädigungsvorgänge in Mehrphasen-Verbundwerkstoffen)

DISSERTATION

zur Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
an der Fakultät Bauingenieurwesen
der Bauhaus-Universität Weimar

vorgelegt von
Dipl.-Ing. Kai Schrader
geboren am 17.09.1977 in Schmalkalden
Weimar, Dezember 2012

Mentor:
Univ.-Prof. Dr.-Ing. habil. Carsten Könke

Gutachter:
Prof. Dr. Ir. Jan G. Rots, Delft University of Technology
Univ.-Prof. Dr. rer. nat. habil. Klaus Gürlebeck,
Bauhaus-Universität Weimar

Tag der Disputation: 21. Juni 2013

Abstract

Modern digital material approaches for the visualization and simulation of heterogeneous materials allow to investigate the behavior of complex multiphase materials with their physical nonlinear material response at various scales. However, these computational techniques require extensive hardware resources with respect to computing power and main memory to solve numerically large-scale discretized models in 3D. Due to a very high number of degrees of freedom, which may rapidly be increased to the two-digit million range, the limited hardware ressources are to be utilized in a most efficient way to enable an execution of the numerical algorithms in minimal computation time. Hence, in the field of computational mechanics, various methods and algorithms can lead to an optimized runtime behavior of nonlinear simulation models, where several approaches are proposed and investigated in this thesis.

Today, the numerical simulation of damage effects in heterogeneous materials is performed by the adaption of multiscale methods. A consistent modeling in the three-dimensional space with an appropriate discretization resolution on each scale (based on a hierarchical or concurrent multiscale model), however, still contains computational challenges in respect to the convergence behavior, the scale transition or the solver performance of the weak coupled problems. The computational efficiency and the distribution among available hardware resources (often based on a parallel hardware architecture) can significantly be improved. In the past years, high-performance computing (HPC) and graphics processing unit (GPU) based computation techniques were established for the investigation of scientific objectives. Their application results in the modification of existing and the development of new computational methods for the numerical implementation, which enables to take advantage of massively clustered computer hardware resources. In the field of numerical simulation in material science, e.g. within the investigation of damage effects in multiphase composites, the suitability of such models is often restricted by the number of degrees of freedom (d.o.f.s) in the three-dimensional spatial discretization. This proves to be difficult for the type of implementation method used for the nonlinear simulation procedure and, simultaneously has a great influence on memory demand and computational time.

In this thesis, a hybrid discretization technique has been developed for the three-dimensional discretization of a three-phase material, which is respecting the numerical efficiency of

nonlinear (damage) simulations of these materials. The increase of the computational efficiency is enabled by the improved scalability of the numerical algorithms. Consequently, substructuring methods for partitioning the hybrid mesh were implemented, tested and adapted to the HPC computing framework using several hundred CPU (central processing units) nodes for building the finite element assembly. A memory-efficient iterative and parallelized equation solver combined with a special preconditioning technique for solving the underlying equation system was modified and adapted to enable combined CPU and GPU based computations.

Hence, it is recommended by the author to apply the substructuring method for hybrid meshes, which respects different material phases and their mechanical behavior and which enables to split the structure in elastic and inelastic parts. However, the consideration of the nonlinear material behavior, specified for the corresponding phase, is limited to the inelastic domains only, and by that causes a decreased computing time for the nonlinear procedure. Due to the high numerical effort for such simulations, an alternative approach for the nonlinear finite element analysis, based on the sequential linear analysis, was implemented in respect to scalable HPC. The incremental-iterative procedure in finite element analysis (FEA) during the nonlinear step was then replaced by a sequence of linear FE analysis when damage in critical regions occured, known in literature as saw-tooth approach. As a result, qualitative (smeared) crack initiation in 3D multiphase specimens has efficiently been simulated.

Kurzfassung

Moderne digitale Ansätze zur Visualisierung und Simulation heterogener Materialien ermöglichen eine detaillierte Analyse der Strukturantworten auf unterschiedlichen räumlichen Skalen. Zur rechnerinternen Abbildung von nichtlinearen Schädigungsvorgängen im dreidimensionalem Raum werden sehr hohe Computer-Hardwareressourcen (bezogen auf die Rechenleistung und verfügbaren Hauptspeicher) für akzeptable Rechenzeiten benötigt, da bei dieser Art der Untersuchung die Anzahl der Freiheitsgrade in den zugrundeliegenden Finite-Elemente-Modellen rasant ansteigt. Die Implementierung der hierzu benötigten Algorithmen muss daher so effizient wie möglich vorgenommen und an die verfügbare Rechnerinfrastruktur angepasst sein. Die numerische Umsetzung von diskreten Riss- oder kontinuumsmechanischen Schädigungsmodellen in hierarchischen oder konkurrenten Mehrskalensimulationen für mehrphasige Materialien mit entsprechender Auflösung birgt weiteres Optimierungspotenzial, insbesondere im Hinblick auf eine konsistente Modellierung in 3D. Der detaillierten und damit aufwendigen geometrischen Modellierung der Teilchenverbunde und deren Diskretisierung sowie der durch die Diskretisierung adaptierten nichtlinearen Konstitutivbeziehungen zur Abbildung von Schädigungsphänomenen steht die begrenzte Rechenleistung verfügbarer Workstation-Computer gegenüber. Zudem kompensiert eine weitreichende Modellidealisierung bzw. Abstraktion den Rechenaufwand auf Kosten der Ergebnisqualität des komplexen Strukturantwortverhaltens. Diese Arbeit verfolgt den Ansatz, die für die Initiierung und Ausbreitung von Schädigungsvorgängen massgebende materielle Phase in heterogenen Mesoskalen-Modellen (zum Beispiel eine volumenbezogenen Interfacezone in klassischen Matrix-Einschluss-Verbunden) hochaufgelöst zu diskretisieren und simultan die zugrundeliegenden Algorithmen hochskalierbar zu parallelisieren, um damit ein effizientes (wiederholtes) Lösen der verteilten globalen Gleichungssysteme im Rahmen von Schädigungssimulationen zu gewährleisten.
Zusätzlich besteht die Möglichkeit, durch eine hybride Diskretisierung elastische und inelastische Materialbereiche initial festzulegen, um Schädigungsvorgänge ausschliesslich in den inelastischen Bereichen zuzulassen, wodurch sich die Modelldimension (z.B. in Bezug auf die zu speichernden Geschichtsvariablen) signifikant reduzieren lässt. Das Schädigungsverhalten in den inelastischen Bereichen wird dabei vereinfachend als isotrop beschrieben und partiell regularisiert und fungiert damit als stabilisierend für die Konvergenzfindung. Weiterhin können Methoden zur sequentiellen linearen Analyse mit

Steifigkeitsdegradation numerisch sehr flexibel adaptiert und parallelisiert werden.
Die stetige Zunahme und Verfügbarkeit von massiv-parallelen Computer-Hardwareressourcen in Form von Hochleistungsclustern und -rechenzentren bietet zudem die Möglichkeit die entsprechenden Algorithmen bezüglich ihrer Skalierbarkeit an diese Systeme anzupassen. Unter Nutzung von mehreren hundert simultan arbeitenden many-core Recheneinheiten können dann numerische Simulationen unter Berücksichtigung von Schädigungseffekten in dreidimensional aufgelösten, mit mehreren Millionen Freiheitsgraden beschriebenen Modellen, durchgeführt werden. Weitere Effizienzsteigerungen können zudem durch den Einsatz von hybriden Architekturen mit mehreren CPU-GPU Knoten (unter zusätzlicher Berücksichtigung von graphischen Recheneinheiten) realisiert werden.
Im Rahmen dieser Arbeit ist die numerische Effizienz der implementierten Algorithmen durch Cluster-basiertes High-Performance Computing des Höchstleistungsrechenzentrums Stuttgart (HLRS) evaluiert worden. Dabei wurde unter anderem der hybride NEC Nehalem Cluster sowie der seit Anfang des Jahres 2012 zur Verfügung stehende Cray XE6 Cluster mit bis zu 512 CPU-Einheiten zur Berechnung herangezogen. Die hier vorgelegte Arbeit stellt dabei einen entscheidenden Schritt für zukünftige Mehrskalensimulationen von schädigungsinduzierten heterogenen Werkstoffen unter Nutzung von High-Performance Computerarchitekturen dar.

Nomenclature

Abbreviations

3D	three-dimensional
ccNUMA	cache-coherent non-unified memory access
CG	conjugate gradients
coo	coordinate storage format
CPU	central processing unit
csr	compressed sparse row as well as compressed row storage (crs)
CUDA	compute unified device architecture
CZM	cohesive zone model
d.o.f.s	degrees of freedom
DD	domain decomposition
DMP	distributed memory processing
DN	Dirichlet-Neumann
FCM	fictitious crack model
FE	finite element
FEA	finite element analysis
FEM	finite element method
FETI-DP	finite element tearing and interconnecting method - dual primal
GPGPU	general-purpose graphics processing unit
GPU	graphics processing unit
HG	hourglass stabilization
HLRS	high-performance computing center Stuttgart
HPC	high-performance computing
ITZ	interfacial transition zone
LD	load-displacement curve
LEFM	linear-elastic fracture mechanics
LU	cholesky factorization
mJPCG	modified Jacobi-point based preconditioned conjugate gradients
MPI	message-passing interface
ndcsr	nodal compressed sparse row
NN	Neumann-Neumann
NR	Newton-Raphson
NUMA	non-unified memory access

openMP	open specifications for multi-processing
PCG	preconditioned conjugate gradients
PDE	partial differential equation
PPCG	parallelized preconditioned conjugate gradients
SI	international system of units
SLA	sequential linear analysis
SMP	shared memory processing
SP	saddle-point
SPE	saddle-point equation
STS	saw-tooth softening
VITZ	volumetric interfacial transition zone
XFEM	extended finite element method

General notations

$\delta(\cdot)$	variation of $\cdot$
$\|\cdot\|$	euclidian vector norm of $\cdot$, also defined as $\|\cdot\|_2$
$\|\cdot\|$	absolute value of a scalar expression $\cdot$
$\mathcal{O}(\cdot)$	mathematical term depending on $\cdot$
$\det(\cdot)$	determinant of $\cdot$
$\mathrm{e}^{(\cdot)}$	exponential function of $\cdot$
$\mathrm{grad}(\cdot)$	gradient of $\cdot$
$\mathrm{lin}(\cdot)$	linearization of $\cdot$
$\ln(\cdot)$	natural logarithm of $\cdot$
$(\cdot)^T$	transpose of $\cdot$
$(\cdot)^{-1}$	inverse of $\cdot$
$\otimes$	tensor product
ν	Poisson's ratio
E	Young's modulus
f_t	tensile strength
G_f	fracture energy
g_f	specific fracture energy
$\boldsymbol{K}_{bb}$	assembled submatrix respecting the boundary nodal d.o.f.s
$\boldsymbol{K}_{bi}$	assembled submatrix respecting the coupled terms $_{bi}$
$\boldsymbol{K}_{ii}$	assembled submatrix respecting the interior nodal d.o.f.s
$\boldsymbol{S}$	Schur complement operator
$\boldsymbol{B}$	element-wise strain displacement matrix
$\boldsymbol{D}_e, \boldsymbol{D}_k$	differential operators
$\boldsymbol{f}^{ext}$	external load vector
$\boldsymbol{f}^{gl}$	global load vector

$\boldsymbol{f}^{int}$	internal load vector
$\boldsymbol{K}^{gl}$	global stiffness matrix
$\boldsymbol{K}_e$	local element stiffness matrix
$\boldsymbol{N}$	element-wise matrix of shape functions
$\widetilde{\boldsymbol{S}}$	global assembled Schur complement operator
$\boldsymbol{u}^{gl}$	vector of global nodal degrees of freedom
$\mathbb{E}$	fourth order material tensor
$\boldsymbol{\sigma}$	stress tensor
$\boldsymbol{\varepsilon}$	strain tensor
$\boldsymbol{\varepsilon}^{el}$	elastic strain tensor
$\boldsymbol{\varepsilon}^{inel}$	inelastic strain tensor
$\boldsymbol{\varepsilon}^{tot}$	total strain tensor
$\boldsymbol{\tau}$	Kirchhoff stress tensor
$\boldsymbol{C}$	Cauchy-Green tensor
$\boldsymbol{E}$	Green-Lagrange strain tensor
$\boldsymbol{F}$	tensorial deformation gradient
$\boldsymbol{I}$	identity matrix
$\boldsymbol{S}$	material stress tensor
$\boldsymbol{T}$	Cauchy stress tensor
$\boldsymbol{T}_{P_1}$	first Kirchhoff stress tensor
$\boldsymbol{T}_{P_2}$	second Kirchhoff stress tensor
σ_Y	yield stress

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The life-time assessment of engineering structures relies on sophisticated material models, integrating all different aspects of damage initiation and deterioration over the expected life-time of a structure. Therefore, the current material models in engineering applications are integrating modern approaches from material science via multiscale methods. Especially for heterogeneous materials, these multiscale approaches allow a detailed insight into the material physics on appropriate scales [Raabe 2004]. In (material) engineering science, the investigation of 3D material behavior, such as the damage initiation and propagation at different scales, is based on complex simulation models, which may require extensive resources, if they capture the heterogeneous nature and also include the specific material behavior of each material phase. By this, they are able to represent the damage behavior of the quasi-brittle material (such as concrete or mortar mixtures as complex matrix-inclusion based composites) up to the macroscale, where the damage is induced by accumulated effects of micro-cracking. A detailed illustration of these phenomena at microscale, observed in ultra high-performance concrete [Möser 2006] and in lime mortar [ETH 2004], is given in figure 1.1. As an additional benefit of multiscale modeling, the parameter identification for separate phases is easier to investigate than identifying the constitutive parameters for heterogeneous material mixtures. The necessary up-scaling of the results at micro- and mesoscale to macroscale can either be done by analytical or numerical homogenization techniques or by applying concurrent multiscale methods. In general, a major drawback of using multiscale methods is the tremendous increase in degrees of freedom (d.o.f.s.) of the resulting equation systems when studying models at micro- or mesoscale. In damage simulations, the incremental-iterative approach requires the repeated solution of the linearized equation system, resulting in an even more crucial computing time consumption.

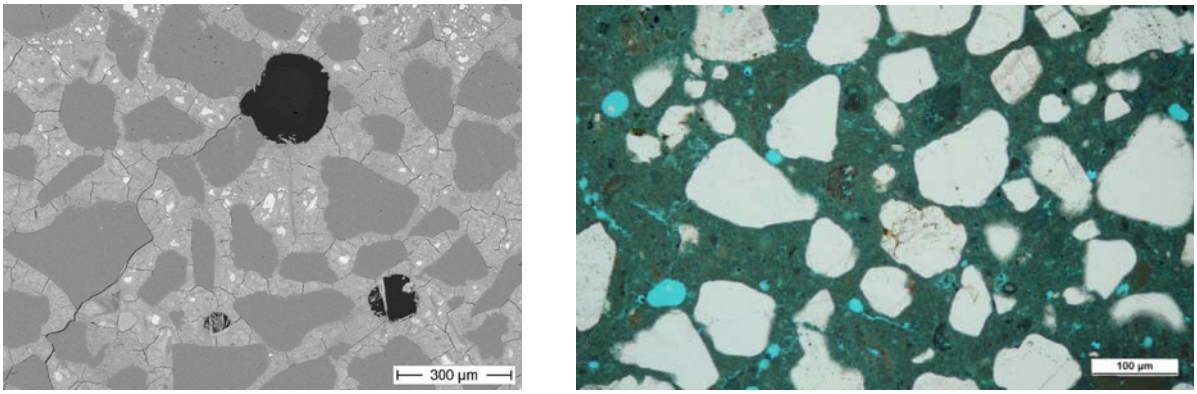


Figure 1.1: Visualization of concrete (left, [Möser 2006]) and lime mortar (right, [ETH Zürich 2004]) at the microscale, using the scanning electron microscope (SEM).

The 'close to reality description' of the material behavior of multiphase composites at sub-macroscale (under specific boundary and loading conditions) is associated with the characterization of very complex physical processes, not only in the state of damage and collapse, but also in the initial configuration. Additionally, numerical models in 2D do not cover important out-of-plane effects in these materials at meso- or microscale, so that the final simulation model should be built in 3D, if at all numerically possible. There, different material phases are connected over a small interfacial transition zone (ITZ), which is dedicated to be the weakest link and therefore, the beginning of the initiation of (micro-) cracks and their accumulation and propagation to and through other phases, which often results in macroscale failure in materials and consequently, in the entire engineering structure. To capture the behavior of such critical zones in a more realistic way, it is necessary to obtain a deep insight into the geometrical and mechanical characteristics and to transfer this information in corresponding numerical models. At a first step meso- or microscale models are considered with a high resolution of such zones and, hence, the phenomenological description of physical processes in 3D should be limited to the representation of mechanical degradation effects. The evolution of the initiation and propagation of damage effects, which has to be described, is based on the fundamentals of continuum mechanics and continuum damage mechanics. The type of material law considered for the phenomenological model of the ITZ is essential for a representation of the close to reality damage effects in multiphase materials. Here, the phenomenological model is often based on observed physical characteristics quantified in corresponding material properties and the resulting mathematical description of the physical (mechanical) problem results in partial differential equations (PDEs). However, in most cases, the analytical solution of this type of equations is not feasible. Today, in technical engineering modern approaches for the numerical computation of such differential equation systems are available. The finite element method (FEM, [Bathe 1995]) as a standard discretization technique can be applied to solve such system of PDEs numerically, similar to the

analysis of mechanical problems at the macroscale.

The (nonlinear) finite element method is a robust and approved technique, which can be used for the computational representation and discretization of material failure in multiphase composites. This numerical approximation technique is the result of the weak variational form of equilibrium described by the underlying Navier partial differential equation. It is respecting the kinematic relation, the material law as well as the equilibrium of the internal and external forces. The large-scale simulation of heterogeneous material specimens strongly depends on the type of the applied computational method (in respect to the phenomenological description) and on its efficiency and performance in order to obtain qualitative and quantitative results, being close to realistic observations in acceptable computing time. In this thesis, combined (hybrid) approaches for the suitable discretization, the efficient solution and the sophisticated representation of damage effects in multiphase materials are addressed. The main content is the investigation of a general hybrid discretization technique for three-phase materials and a distributed solution strategy adapted for the final system of equations. This hybrid discretization considers a volumetric ITZ in high-resolution around the inclusions, which are embedded in a regular grid. Due to the high numerical effort for the memory demand and the computing time, which is mainly caused by the high number of unknowns of the resulting system of equations in 3D, a scalable computing framework for the application of the developed algorithms on high-performance computers is proposed to be analysed. The initial separation of sub-domains governed by their fundamentally different material behaviors leads to the separation and decomposition of elastic and inelastic subproblems. This enables a drastic reduction in global degrees of freedom (d.o.f.s) and, consequently, results in a significant decrease of the solution time. Furthermore, the author will use an initial and updated decomposed damage zone during the nonlinear (damage) simulation. This strategy should reduce the high memory demand required for storing all necessary history data, which can be extensive for large-scale FE models in 3D. The first objective of this thesis is the

- development of algorithms for the coupling of adaptive (irregular polyhedron) and grid-based (regular Cartesian hexahedron) meshes and applying a discretization techniques for the 3D hybrid meshing of multiphase materials in 3D, including the aligned meshed transition regions describing the volumetric ITZ

The most time-consuming task of the computation procedure is the numerical solution of the underlying equation system, which also depends on the level of detail used for the discretized mechanical task. Therefore, it is necessary to develop adequate numerical methods for an efficient computation of the linearized problem and simultaneously, to use the available hardware resources in the most efficient way. In recent years, the distributed computing based on the message-passing interface standard (MPI) has been

proven valuable, enabling the distributed computation of linear equation systems, utilizing as many computational nodes available in a high-performance computing framework [Cehavir et al. 2010]. Furthermore, during the past years, hybrid CPU-GPU architectures were developed using multiple graphic processing units (GPU), which advance within a high-scalable implementation of basic linear algebra subroutines compared to 'CPU only' computations. By this, the parallel code execution using different hardware architectures (based on CPU and GPU) may simultaneously be utilized. Consequently, a memory-advantageous iterative MPI-solver strategy based on the conjugate gradient method (CG) may be chosen and accelerated by an efficient preconditioning technique. The execution of the sparse-matrix vector product, which occurs during the preconditioned conjugate gradient method (PCG), is the most time-consuming task (per iteration step), and has to be evaluated in respect to different matrix storage formats and hardware architectures. In general, the parallelization techniques are based on standard overlapping or non-overlapping domain decomposition methods for FE problems. They can be improved by considering an elastic-inelastic domain split and thereby, enabling the decomposition of a reduced nonlinear domain. Hence, the implementation concept should respect different parallel hardware architectures as well as its application on high-performance computers, also being an other important aim of this thesis. The developed algorithms should then be evaluated on the hybrid CPU-GPU NEC Nehalem cluster and on the new petaflop system CRAY XE6 (available since the beginning of 2012) at the high-performance computing center Stuttgart (HLRS).

The utilization of memory-efficient iterative solver techniques combined with domain decomposition (DD) methods can lead to a significantly improved computational performance. Due to the fact that the solution of the linearized global equation system is the most time-consuming task in linear as well as nonlinear simulations (with the repeated solution of the linearized step), it is important to select an optimal combination of the solver, the preconditioning and the parallelization techniques as well as considering the architectural features of the hardware applied. In large-scale finite element analysis, with many million degrees of freedom, direct solver techniques based on the Gaussian elimination or Cholesky (LU) factorization may no longer be suitable due to their memory-extensive application. However, direct parallel solvers [Amestoy et al. 2007] have the disadvantage of a high-memory demand induced by the direct factorization of the assembled matrices for each sub-domain. Therefore, the classical use of the Schur complement method (where the Schur complement matrix is explicitly extracted for each sub-domain) requires extensive computing time and memory demand. Due to the dense matrix structure of the Schur complement system it is difficult to store, to factorize and to solve the assembled Schur complement system directly. This issue has an increasing influence when the number of degrees of freedom is increasing. Improvements, such as out-of-core strategies for the memory-extensive factorization task, may simultaneously increase the computational

solver time due to the slower access to the workspace medium. Modern FETI (Finite element tearing and interconnecting) and FETI-DP (dual-primal) methods however, can decrease the memory demand. On one hand, the solver stability mainly depends on the separation of the boundaries discretized in primal and dual nodes, which are difficult to arrange for irregular aligned or hybrid FE meshes by embedding fine aligned meshed domains in a regular grid. On the other hand, for iterative solution methods, the used preconditioning technique can be critical either in regards to computing time or to extensive memory demand (depending on the type of discretized problem). Due to the high number of unknowns in the resulting equation system, an efficient parallelization technique should be implemented to simultaneously store and solve the distributed equation system. Therefore, the second aim of this work is

- the development and parallelization of algorithms for the partitioned solution strategies based on the hybrid discretization technique and their resulting decomposed systems of equations for the efficient 3D simulation of the delamination and degradation behavior between different phases during the linear and nonlinear FE analysis

In this thesis, firstly, a parallelized version of the preconditioned conjugate gradient method (PPCG) based on domain decomposition is used without explicitly building the Schur complement system. By this, the numerically expensive usage of the classical Schur complements is avoided. The iterative computation of the resulting equation system for the nonlinear problem is also executed by using the preconditioned conjugate gradient method (CG, [Kelley 1995]). Furthermore, the influence of different preconditioning techniques [Basermann et al. 1997] for the CG computation is investigated and the parallelization of the preconditioned conjugate gradient method, which is based on the synchronous computation of the assembled substructures, is addressed. Secondly, the preconditioning technique is restricted to a scaled main-diagonal precondition strategy with a special scaling parameter taking the upper or lower bound of the spectral radius of the assembled sub-matrices into account. This reduces the computing time of building the preconditioning matrix as well as memory demand and also the time needed to execute the matrix-vector product involved by the preconditioning matrix. Moreover, for the computation of the sparse matrix-vector products for each sub-domain several different matrix storage formats are taken into account. Due to the implementation of a nodal storage scheme of the distributed FE data, a nodal compressed row storage is used to improve the performance (compared to standard coordinate storage (coo) or compressed row storage (csr or crs)). Furthermore, the solver concept is adapted to multiprocessor systems as well as to hybrid CPU-GPU clusters, such as the NEC Nehalem cluster with access up to 32 GPU nodes, namely the Nvidia Tesla technology of 2010. There, the sparse matrix-vector operations (spmv) is distributed and outsourced to several graphics processing units (GPU) based on the Nvidia Tesla architecture. This enables the high-scalable spmv execution in a hybrid CPU-GPU approach. As a result of the combined

spmv execution on CPU and GPU hardware devices [Papadrakakis et al. 2011], a hybrid computation model with further improvements in regards to acceptable memory demand and computing time is implemented. The programming framework used for the GPUs is based on CUDA, which was introduced in 2007.
Finally, in this work the parallelized HPC computing framework is applied for hybrid meshed and partitioned multiphase specimens, especially for the numerical evaluation of the qualitative initiation of damage effects in multiphase materials. Other approaches than the conventional way of modeling material nonlinearities [Wriggers 2008], like the consideration of material discontinuities by using the extended finite element method (XFEM) or the classical material point based nonlinear simulation techniques such as the Newton-Raphson (NR), modified NR or arc-length methods often lead to numerical instabilities and convergence problems during the establishment of the nonlinear post-peak paths. This is more crucial in the three-dimensional case of modeling material failures, especially with heterogeneous characteristics or multi-physical properties. Due to this, the saw-tooth softening approach introduced by [Rots et al. 2006] provides an alternative approach to model material nonlinearities in a sequential way due to the repeated computation of the stepwise modified linearized problem. Hence, one important feature of this advance is the ensured convergence of the underlying procedures. By this, the realistic representation of softening branches for quasi-brittle materials can be reproduced. This is even the case for three-dimensional models and for critical branching points of the post-peak slopes where numerical problems due to the inexact convergence behavior of classical iteration schemes (NR or modified NR, arc-length) could occur. In this work, a scalable version of the sequential linear analysis (SLA) will be implemented applied for the hybrid-meshed heterogeneous specimens for the representation of induced damage effects in a nonlinear simulation model. By this, the developed solver strategy will be also adapted for hybrid high-performance computing frameworks taking different parallel hardware architectures into account.

1.2 Intention of this work

To summarize, the overall intention of this work is

- to develop a hybrid discretization technique for multiphase composites,
- to adapt a high-scalable solver strategy for the hybrid discretization technique and
- to simulate the nonlinear material behavior induced by damage effects in multiphase composites applied for a volumetric interfacial transition zone

Consequently, some important criteria are to be respected in this work for the three-dimensional modeling and simulation of multiphase materials such as

- 3D modeling is necessary for the investigation of out-of-plane effects

- Increased computational efficiency is required for the numerical methods in 3D
- Concentration on the qualitative description of the damage behavior in 3D
- Highly-scalable computations based on a parallelized implementation as well as the suitability of the programming framework for HPC is proposed.

The remainder of the thesis is structured as follows: Chapter 2 will give an overview of state-of-the-art simulation models respecting small interface regions in multiphase materials. Chapter 3 will describe theoretical notations in respect to continuum mechanics and to the finite element discretization technique. In chapter 4, the numerical discretization of multiphase specimens at mesoscale as well as the numerical decomposition technique suitable for the proposed scalable simulation approach is presented. In chapter 5, linearized solution techniques based on non-overlapping domain decomposition methods will be reviewed. While in chapter 6, the hybrid computing model for a high-performance computing framework will be described also including the evaluation of numerical examples for the linear-elastic finite element analysis. Chapter 7 will then state the extension of the adapted distributed and linearized solution technique, considering a nonlinear material model to allow damage simulations of hybrid 3D multiphase specimens. Chapter 8 will include the summary of the investigations made in this thesis and finally, chapter 9 will give concluding remarks and an outlook for potential future research activities.

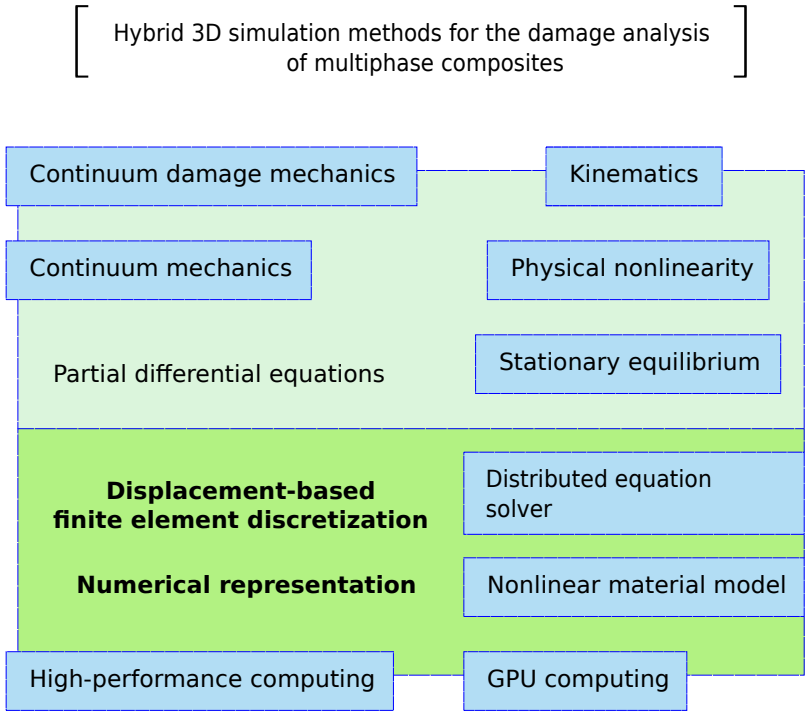


Figure 1.2: Topical overview involved with creating a scalable nonlinear FE simulation model in 3D.

Chapter 2

State-of-the-art simulation models for small interface regions in multiphase materials

This chapter gives an overview of and an introduction to the theory of material modeling of failure in heterogeneous and quasi-brittle materials. The aim is, to adapt these phenomenological models to small interface regions between distinct phases in multiphase materials. Starting with the introduction to continuum mechanics, the author follows concepts of damage modeling for material continuum and discontinuum, being essential for the numerical representation of failure and fracture of heterogeneous materials. Concrete, as a very complex heterogeneous and quasi-brittle material, continues to be the subject of research in material science as well as in the community of computational mechanics, which in the past decades led to many major developments in the field of continuum and discontinuum modeling of damage in multiphase composites. Such approaches are, among others, proposed in [Rots 1988], [Mazars et al. 1989], [Schlangen et al. 1992], [Červenka 1994], [Ozbolt et al. 1996] and [Feenstra et al. 1996]. Furthermore, in the field of modeling failure and fracture one differs between smeared and discrete damage models in respect to the investigation of the initiation and propagation of cracks, since here the irreversible effects are addressed in combination with aspects of different regularization techniques to enable the representative numerical application of such models and also their validation with experimental data.

2.1 Continuum models for heterogeneous materials

Quasi-brittle materials, such as different mortar or concrete mixtures, rough ceramics, multiphase composites etc., often show a very complex geometrical structure, which can be classified and disclosed in several material phases and investigated at various spatial sub-macro scales. The complexity of the geometrical description as well as the partial me-

chanical characteristics of these materials lead to the conclusion that these characteristics need to be taken into account for a phenomenological representation of the constitutive behavior. To analyse this, the continuum modeling approach at numerical level can be used, which is the so-called smeared (macroscale) approach for specimens or components of real-life structures and materials. The smeared modeling of material nonlinearity at macro-level was state-of-the-art for a long time.

During the past decade, phenomenological relations were developed to describe the material behavior of heterogeneous materials at macroscale in the field of continuum mechanics and material science. Moreover, this approach was extended for the mesoscale, the microscale as well as even for the nanoscale, capturing the natural heterogeneous characteristic at these scales in numerical models. By this sophisticated approach, several distinct material phases as well as their coupling are described according to their phenomenological characteristics observed from reality. It particularly considers specific material properties, where actual approaches also tend to take quantified parameters at the corresponding spatial scale into account. The connection to the next close scale is mainly driven by the so-called scale transition techniques, also known as homogenization or scale-bridging techniques, leading to hierarchical (separated) or concurrent (simultaneous) multiscale models. Due to the high numerical effort which is necessary to realize such simulation models, the spatial extension and order of dimension, the resolution or the number of phases and also the complexity of the material laws applied are often limited, particularly for three-dimensional models. This is even more crucial, if the proposed material behavior is induced by irreversible damage effects, often the case in modeling failure and fracture. Even the way to the collaps of a structure, like e.g. caused by tensile softening, is generally a highly nonlinear process.

A meso- or microscale related continuum, e.g. for concrete, may be analysed as a composite of different (geometrical) complex inclusions embedded in a (more or less) homogeneous mortar matrix with a small interface region connecting the inclusions with the matrix material: the so-called interfacial transition zone (ITZ). The ITZ is geometrically and chemically complex, and finally, from a mechanical point of view, a sensitive link of bonding. This phase is dedicated to the beginning of (diffuse) micro-cracking and their cumulation results in a softening response behavior as well as in the generation of macrocracks. An important research aim was and still is to describe the damage effects in an appropriate way to enable the resulting response behavior of the complete composite to be optimized to enable a transfer from the fine to the coarse scale. Beside the geometrical and physical information of the individual phases, damage based information can be illustrated in a qualitative way, like e.g. by the representation of initialization or propagation of existing crack patterns. Figure 2.1 and 2.2 illustrate the realization of some computational continuum models with heterogeneous characteristics as well as their damage behavior caused by the crack propagation induced by applying tensile loading conditions.

This led to the development of various techniques in the field of the continuum damage mechanics, yielding to the continuum and discontinuum damage models. Some of which are described in the following sections.

2.2 Material damage modeling

Damage occurs by the initiation and propagation of accumulated micro-cracks. In real-life this can lead to material failure, fracture and fatigue over time and finally, to a higher sensitivity to such phenomena since this increases the probability of an abrupt structural collapse. The micro-cracking 'localizes' in zones of high stress concentration and their evolution to one large macroscopic crack in the fracture process zone or in a localized crack band. Continued increasing or cyclic loading conditions yield to exceeded inelastic strains or discrete crack openings causing degradation effects and release debonding forces within increased dissipated energy which finally results in a stress-free crack at the macroscale. The modeling of such material failures in solid mechanics is classified by two major techniques depending on the type of the kinematic description: a) the discrete damage modeling, with the explicit representation of strong or weak discontinuities (e.g. considering a jump in the displacement or strain field) and b) the continuous representation of the localized strains.
The first type of discontinuous damage models is ideally compromised by the well-known fictitious crack model (FCM, [Hillerborg et al. 1976]), which was introduced for linear-elastic fracture mechanics (LFEM). The development of discrete crack modeling techniques including several modifications was the result. The consideration of zero-thickness interface elements at the cracking surfaces within a stress-free tipping point of the macro-crack (also realized in 3D) is recommended in [Carol et al. 1997b]. Further developments yield to discrete cohesive zone models (CCM, see section 2.3.1).

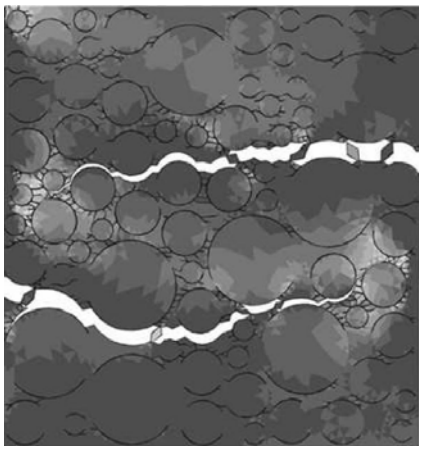


Figure 2.1: 2D representation of a multiphase continuum model and resulting crack patterns [Wang et al. 1999].

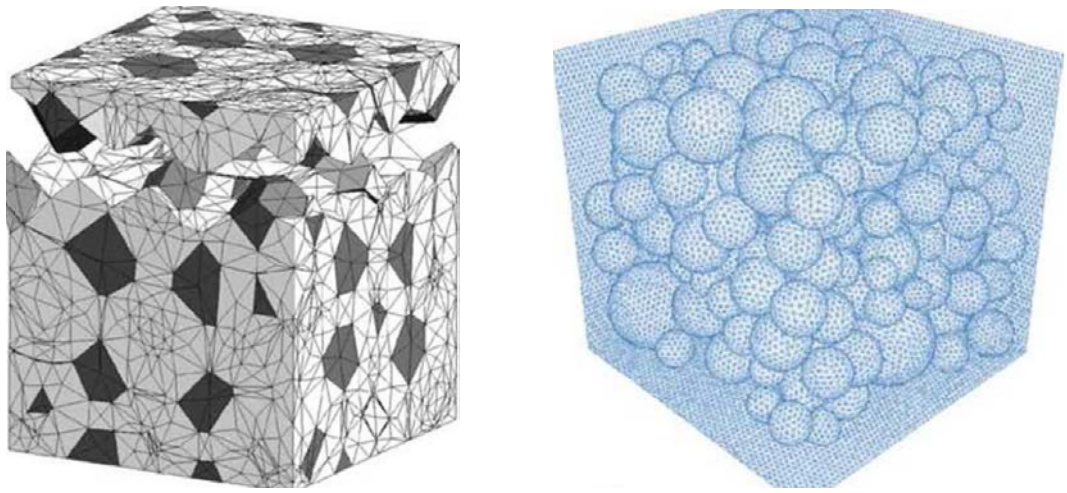


Figure 2.2: 3D representation of multiphase continuum models and resulting crack pattern (left) [Caballero et al. 2006, Wriggers et al. 2006].

The second type of continuum damage mechanics is based on smeared continuous crack approaches as stated in [Ozbolt et al. 1996] and [Oliver 1996]. Here, the crack initiation is caused by a specific criterion evaluating the critical stress state (e.g. initiated by the maximum principle stress in case of tensile-softening in LFEM) depending on the actual stress state at material point level of the material continuum, which can be seen as the spatial initial point of cracking. The propagation is evaluated corresponding to the adapted approaches from the classical theory of inelasticity and plasticity. There, the inelastic state of the strain field can represent the (smeared) crack expansion, which then leads to the development of the strain-softening models [Simo et al. 1994]. A simple technique to observe this phenomena is to use a (local or non-local) isotropic damage law. Independently from the orientation of the crack, the resulting effects of degradation is assumed to be equal in all directions. Furthermore, the exceeded dissipation of energy during the fracture process yields to a degradation of the material stiffness. There are issues with these smeared models which are induced by mesh size dependencies and by localization phenomenon as well as the underestimation of the dissipated energy. Therefore, the demand to discover regularization techniques for mesh-size objectivity (see section 2.4.1) was the result. A further critical point is the *path following* which is applied to represent the post-peak softening slope with multiple, and thus, ambiguous branching points, often leading to convergence problems. This is particularly a challenge in three-dimensional damage modeling. A common point in all developments is their demand for regularization strategies yielding to non-local formulations [Pijaudier-Cabot et al. 1987], [Jirásek 1998]. Further extensions combine the elasto-plastic and the (non-local) damage formulation for the initiation and propagation of smeared cracks adapted for the simulation of material nonlinearities in mesoscale models.
Microplane models [Carol et al. 1997a] and rotating crack models [Jirásek et al. 1998] delivered further innovations for the discontinuous modeling of damage and fracture. A

further innovative concept, as proposed in this work, is the sequentially linear analysis which considers saw-tooth diagrams at material point level as distinguished in [Rots 2001], [Rots et al. 2004] and [Rots et al. 2006].

2.3 Discrete models

2.3.1 Cohesive zone model (CZM)

In linear-elastic fracture mechanics (LEFM), where an abrupt material failure at material point level occurs, the fictitious crack model of [Hillerborg et al. 1976] was introduced for the application to quasi-brittle materials, such as concrete. The mechanical approach however, is based on the assumption that propagating micro-cracks, which are accumulated during the fracture process, are smeared as one macroscopic or fictitious crack only within the corresponding fracture process zone (fig. 2.3). This results in a macroscopic stress-free crack tip as well as in a fracture zone close to the crack, still enabling the stress to transfer between the opposite fracture surfaces of the crack. Some approaches only consider normal stresses in relation to the normal direction of the crack opening. The typical cohesive zone model, as e.g. described in [Galvez et al. 2002] shows an extended approach taking the tangential stress components into account. Furthermore, Carol et al. use a non-associative plasticity formulation for the evaluation of the crack initiation and propagation combined with a coupled normal/shear interface model suitable for discrete crack analysis [Carol et al. 1997b]. This kind of mixed formulation also considers Mode-I (for uniaxial tension) and Mode-IIa (for compression with shear) fracture energy dissipation and enables the presentation of tangential debonding and damage effects. Volume expansion such as the dilatancy effect (e.g. which is observed for concrete) and material interlocking processes can also be handled by this method. Hence, these models can explicitly represent cohesive cracking among the element boundaries considering zero-thickness interface elements [Carol et al. 2002]. However, here the numerical effort increases specifically due to the indispensable necessity of adaptive remeshing. A threedimensional implementation with a modified (hyperbolic) yield condition (as the corresponding cracking criteria and representative numerical tests in 2D) can be found in [Schrader 2005] and, moreover, with modifications considering degradation effects during cyclic loading in [Most et al. 2006]. The softening behavior described in such formulations is specified by using linear and bilinear functions [Hillerborg et al. 1976] as well as exponential functions [Petersson 1981]. Further remarks on softening functions (especially for the concrete material) can be found in [Rots 1992] and [Červenka 1994]. The numerical disadvantages of such discrete modeling concepts are induced by the non-symmetric matrix structure of the consistent tangential material modulus, which demands a suitable solution technique for the global system of equations. Moreover, during the chosen *path following algorithm*, the stress state at each integration point and at each load step is evaluated, often realized

by implicit return mapping techniques [Simo et al. 1994], which are more compromising in comparison to explicit methods. Newer approaches tend to an improved version of an implicit closed point projection as mentioned in [Prokop 2008].
To summarize, CZM is applicable for a three-dimensional discretization, but still is critical at numerical level.

2.3.2 Material discontinuum models

To avoid mesh dependencies and the necessity of remeshing in crack growth simulation models, displacement based discontinuity formulations were developed. The extended finite element method (XFEM, [Sukumar et al. 2000]), for example, is an improved technique, which respects discontinuities inside of the finite element patches extending the partition of unity concept [Melenk et al. 1996], [Sukumar et al. 2004]. Within the XFEM, mesh independent material interfaces and discontinuities may be represented without the explicit discretization along the element boundaries. Here, the extention to the standard displacement based FE evolution results in the enrichment of the original shape functions by special functional terms of the heaviside or levelset function, existing in several modifications [Sukumar et al. 2001]. These mathematical operators are used for the approximation of the discontinuous displacement field, which represents the crack trajectory inside of the finite element. Thereby, this technique enables the modeling of a propagating crack through an initial and constant finite element mesh by additional introduced degrees of freedom along the crack path up to the element edge. Further developments are taking the cohesive characteristic of the crack interfaces into account [Moës et al. 2002], avoiding stress singularities near the tipping point of the crack. This approach also avoids the explicit representation of cracks by interface elements [Carol et al. 2002] as well as the remeshing, which in general, is a numerical expensive task [Belytschko et al. 1999].

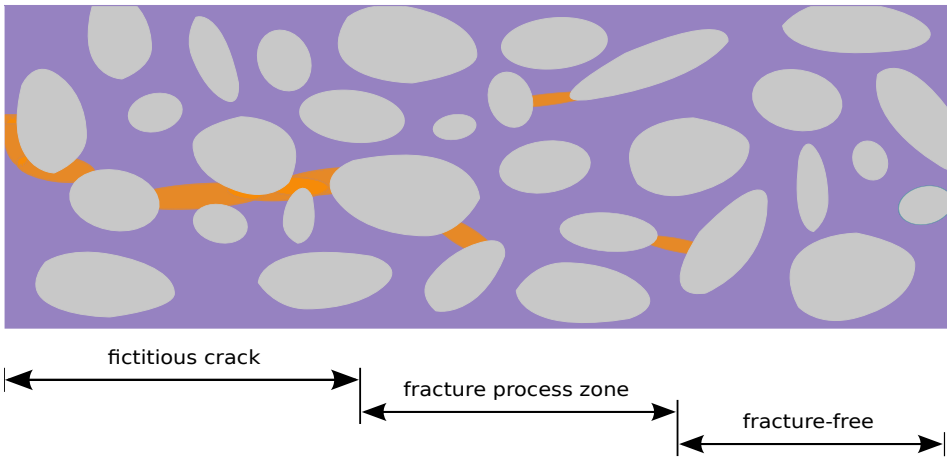


Figure 2.3: Fracture process in quasi-brittle materials (e.g. concrete) utilizing the FCM model of Hillerborg.

However, applying the XFEM technique to compute complex problems in 3D including material failure still comprises unsolved issues and numerically yields to an ill-conditioned system of equations.

2.4 Continuum softening models

2.4.1 Strain-softening driven energy dissipation

Continuum damage models represent opening cracks as smeared displacement jumps, which finally result in inelastic strains. In such a smeared approach of continuum softening models, the strain-softening occurs by the establishment of a softening band, which is characterized by localized strains in the fracture process zone and can be classified as a separated zone of material softening. The softening band, which is then surrounded by intact material, typically has a thickness corresponding to the size of the finite element mesh density as a constraint and may generally localize in one layer of finite elements [Jirásek et al. 2003]. This introduces an objective mesh bias, since the mesh density is usually not consistent. Motivated by the consideration of the thickness as a fixed design parameter, several approaches recommend that the value of such a crack band width may correspond to the smallest element size used in the analysis and also depends on the orientation of the crack band. However, the orientation of the softening band is generally unknown and may cause problems in respect to the objectivity of the resolved softening band and also to refined meshes, as the resulting strain field tends to a solution with strong discontinuities. Moreover, the material heterogeneity, the criterion of crack initiation, and the type of crack trajectory led to several damage models [Jirásek 1998], such as

(i) Isotropic damage models

(ii) Anisotropic damage models

(iii) Rotating crack models

Isotropic damage models are very sensitive to the mesh density and tend to localize along the mesh lines. In comparison to the rotating crack model the mesh-size objectivity has been improved, but stress locking influences the strain localization and, by this, the establishment of a potential softening band occurs, resulting in unconverged solutions. The transition of such models to scalar damaging, however, may overcome the stress locking phenomena. Finally, continuum softening models respecting an anisotropic damage law can represent the trajectory of the localized strain band more realistically. The evaluation of the crack initiation is based on the maximum principle stress or the Rankine-like crack criterion, also used in the field of the fracture mechanics. Since the softening behavior is strongly indicated by the dissipated energy during the degradation process and simultaneously depends on the thickness of the softening band, the adjustment of the post-peak softening modulus according to the element size is recommended. This leads

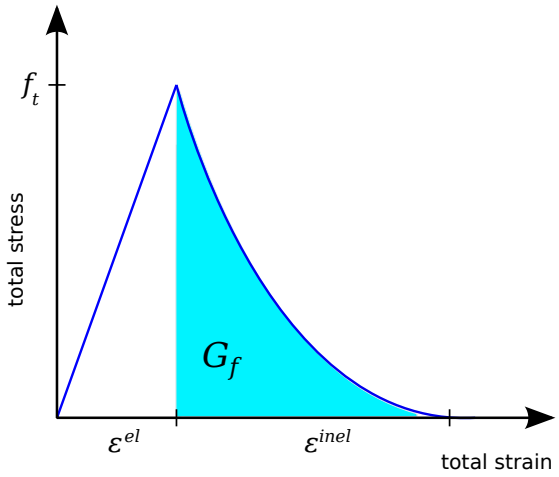


Figure 2.4: Exponential tensile-softening curve and the evolution of fracture energy G_f in the post-peak region with f_t as the tensile strength, ε^{el} as the elastic and the ε^{inel} as inelastic strain component.

to several regularization techniques to obtain mesh-size objectivity for arbitrary meshes. Some regularization techniques considering the fracture-energy approach are proposed in this work, and therefore, will be reviewed in detail in the following section.

2.4.2 Fracture energy based regularization

The fracture-energy regularization is motivated by ensuring the correct energy dissipation in the localized softening band, which is corresponding to the area under the post-peak stress-strain curve. In case of material tensile softening, the proposed dissipation per unit area may be determined by considering the Mode-I fracture energy as a known material property as well as the crack band width resulting from the spatial discretization. Here, the achieved dissipated energy arises from the (internal) incremental deformation work (induced by the post-peak (inelastic) stress and strain states) and is cumulated during the structural fracture process, being equal to the fracture energy - an important material property in simulation models for fracture. Further developments yield to partial or full regularized continuum softening models, where strong displacement-based discontinuities (cracks) are smeared over the thickness of the softening band and may be replaced by corresponding inelastic strains with an adjusted energy dissipation. Other approaches result in non-local continuum damage models, also with higher order gradients of the internal variables [Bažant et al. 2002]. In [Rots et al. 2006] three techniques for a strain-based, a stress-based and a combined strain-stress based regularization approach are suggested for the saw-tooth model with tensile-softening.
In the next chapter (followed this overview) notations of the continuum mechanics and the finite element based discretization technique will be presented.

Chapter 3

Notations for the finite element based discretization of multiphase materials

The mechanical behavior of a material continuum with bulk material properties can be described by fundamental formulations of the continuum mechanics, whereby a comprehensive overview is given in [Jog 2007]. The three main differential equations, which are induced in the Navier differential equation system, are described by the kinematic equation (to obtain the deformation gradient), the constitutive relation (to derive internal forces from the deformation gradient) and the equilibrium equation (coupling internal and external forces). In the following, a comprehensive notation for the evolution of the theory of linear elasticity and nonlinear inelasticity is given. Later in this chapter, the finite element method will be introduced as numerical approximation technique for the solution of the Navier differential equation system followed by efficient element formulations adapted for the discretization of multiphase specimens.

3.1 Continuum mechanics

3.1.1 Kinematics

The motion of a single material point of a continuum [Wriggers 2008], which is changing the position from the reference to the momentary position (see fig. 3.1) can be described by the equation

$$\boldsymbol{x} = \phi(\boldsymbol{X}; t) = \boldsymbol{x}(\boldsymbol{X}) \tag{3.1}$$

with the momentary position $\boldsymbol{x}$ being described by a function of the reference coordinate position $\boldsymbol{X}$ at a discrete time t. The corresponding material deformation gradient $\boldsymbol{F}$ is defined by

$$\boldsymbol{F} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}} = \boldsymbol{x} \cdot \nabla \tag{3.2}$$

This causes the transformation of a material line element $\partial \boldsymbol{X}$ at the reference position to a material line element $\partial \boldsymbol{x}$ at the momentary position by

$$\partial \boldsymbol{x} = \boldsymbol{F} \partial \boldsymbol{X} \tag{3.3}$$

in which $\boldsymbol{F}$, generally is an unsymmetric tensor. Moreover, the material penetrability is avoided by the introduction of the condition

$$\mathrm{J} = \det(\boldsymbol{F}) > 0 \tag{3.4}$$

Here, the determinant J of the deformation gradient $\boldsymbol{F}$ describes the volume ratio between reference and momentary configuration of a differential volume element deformed by $\boldsymbol{F}$, such as

$$\mathrm{J} = \frac{\partial \mathrm{V_R}}{\partial \mathrm{V_M}} \tag{3.5}$$

with $\mathrm{V_R}$ as the volume of the reference configuration and $\mathrm{V_M}$ as the volume of the momentary configuration. Additionally, $\boldsymbol{F}$ enables the derivation of different deformation measurements. The right Cauchy-Green deformation tensor yields to

$$\boldsymbol{C} = \boldsymbol{F}^{\mathrm{T}} \boldsymbol{F} \tag{3.6}$$

as well as the Green-Lagrange strain tensor

$$\boldsymbol{E} = \frac{1}{2}(\boldsymbol{C} - \boldsymbol{I}) \tag{3.7}$$

The left Cauchy-Green deformation tensor (Finger tensor) is defined by

$$\boldsymbol{b} = \boldsymbol{F} \boldsymbol{F}^{\mathrm{T}} \tag{3.8}$$

where $\boldsymbol{b}$, the Hencky strain tensor, can be expressed by

$$\boldsymbol{\varepsilon} = \frac{1}{2} \ln(\boldsymbol{b}) \tag{3.9}$$

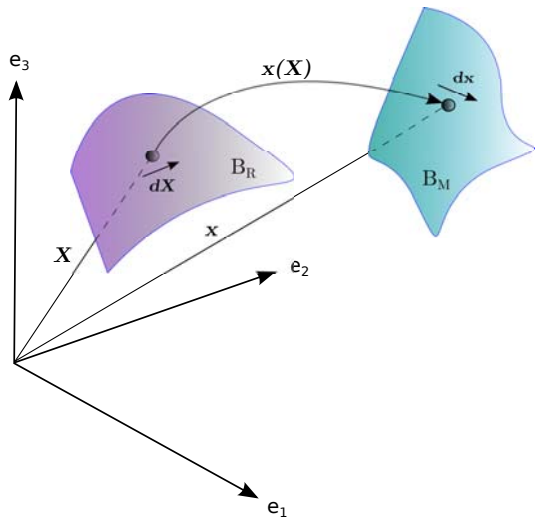


Figure 3.1: Illustration of the motion at material point level from the reference to the momentary position.

3.1.2 Stress tensors

The Cauchy stress tensor $\boldsymbol{T}$ induces an infinitesimal surface force related to the deformed momentary configuration of a surface element. Based on $\boldsymbol{T}$, several stress measurements can be defined. Therewith, the Kirchhoff stress tensor results in

$$\boldsymbol{\tau} = \mathrm{J}\boldsymbol{T} \tag{3.10}$$

The material stress tensor is defined by

$$\boldsymbol{S} = \boldsymbol{F}^{-1}\boldsymbol{T}\boldsymbol{F}^{-\mathrm{T}} \tag{3.11}$$

and consequently

$$\boldsymbol{T} = \boldsymbol{F}\boldsymbol{S}\boldsymbol{F}^{\mathrm{T}} \tag{3.12}$$

The first Piola-Kirchhoff stress tensor yields to

$$\boldsymbol{T}_{\mathrm{P}_1} = \mathrm{J}\boldsymbol{T}\boldsymbol{F}^{\mathrm{T}} \tag{3.13}$$

and the symmetric second Piola-Kirchhoff stress tensor is given by

$$\boldsymbol{T}_{\mathrm{P}_2} = \boldsymbol{F}^{-1}\boldsymbol{T}_{\mathrm{P}_1} = \mathrm{J}\boldsymbol{F}^{-1}\boldsymbol{T}\boldsymbol{F}^{\mathrm{T}} \tag{3.14}$$

Finally, the Mandel stress tensor is

$$\boldsymbol{M} = \boldsymbol{S}\boldsymbol{C} = \boldsymbol{F}^{-1}\boldsymbol{T}\boldsymbol{F} \tag{3.15}$$

Strain and stress tensors can be used to formulate constitutive relations for the linear elasticity and, furthermore, for their extension to the material inelasticity.

3.1.3 Constitutive relations of linear elasticity

The constitutive relation between the Green-Lagrange strain tensor $\boldsymbol{E}$ and the second Piola-Kirchhoff stress tensor is defined by

$$\boldsymbol{T}_{\mathrm{P}_2} = \mathbb{C}\boldsymbol{E} \tag{3.16}$$

which is also known as Hook's law for linear elastic materials with $\mathbb{C}$ as the fourth order elasticity tensor. The linearization of the Green-Lagrange strain tensor results in the linearized strain tensor $\boldsymbol{\epsilon}$

$$\boldsymbol{\epsilon} = \mathrm{lin}\boldsymbol{E} \tag{3.17}$$

The linearized stress tensor $\boldsymbol{\sigma}$ results in

$$\boldsymbol{\sigma} = \mathbb{E}\boldsymbol{\epsilon} \tag{3.18}$$

with $\mathbb{E}$ as the elastic material matrix. Respecting the stress and strain components, eq. (3.17) and eq. (3.18), respectively, Hook's law can be written as

$$\begin{pmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \tau_{xy} \\ \tau_{xz} \\ \tau_{yz} \end{pmatrix} = \mathbb{E} \begin{pmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{zz} \\ \gamma_{xy} \\ \gamma_{xz} \\ \gamma_{yz} \end{pmatrix} \tag{3.19}$$

In case of isotropic material behavior, the material elasticity matrix considering the material properties ν (Poisson's ratio) and E (Young's modulus) can be stated as

$$\mathbb{E} = \frac{\mathrm{E}}{(1+\nu)(1-2\nu)} \begin{pmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{pmatrix} \tag{3.20}$$

The kinematic and the constitutive relation are completed by the differential equilibrium formulation connecting internal and external physical components in respect to the first law of thermodynamics (where consistent conversion of physical energy is assumed). The energy law of mechanics provides variational principles leading to the weak form of the final Navier differential equation. The solution of such PDEs can be done approximately by a numerical discretization technique: The finite element method (FEM) which will be used in this work and by this, is being introduced later in this chapter.

3.1.4 Equilibrium equation and Navier differential equation

The internal forces are represented by the linearized stress tensor $\boldsymbol{\sigma}$ and the external forces $\boldsymbol{p}$ (e.g. body loads) are connected by a differential operator. The notation results in

$$-\boldsymbol{p} = \boldsymbol{\sigma} \cdot \nabla = \boldsymbol{D}_e \cdot \boldsymbol{\sigma} \tag{3.21}$$

With the constitutive relation of eq. (3.18), the introduction of differential operator $\boldsymbol{D}_k$ and the external displacements $\boldsymbol{u}$ (describing the change from the reference $\boldsymbol{X}$ to the momentary position $\boldsymbol{x}$) eq. (3.17) changes to

$$\boldsymbol{\epsilon} = \boldsymbol{D}_k \cdot \boldsymbol{u} \tag{3.22}$$

Finally, the fundamental Navier differential equation considering eq. (3.18) and (3.22) is given with

$$-\boldsymbol{p} = \boldsymbol{D}_e \, \mathbb{E} \, \boldsymbol{\epsilon} = \boldsymbol{D}_e \, \mathbb{E} \, \boldsymbol{D}_k \, \boldsymbol{u} \tag{3.23}$$

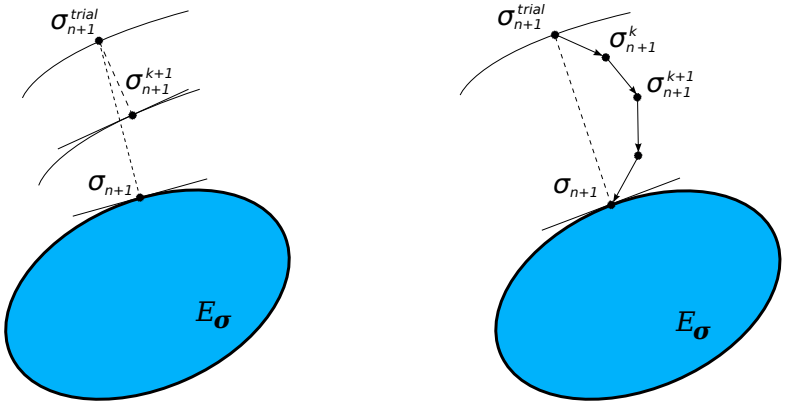


Figure 3.2: Illustration of the closest point projection as implicit return mapping procedure evaluating plastic strains: Iterative backtracking scheme of the elastic trial stress state $\boldsymbol{\sigma}^{trial}$, at last hitting the flow condition $f(\boldsymbol{\sigma}_{n+1}) = 0$ (left) and the improved closest point projection.

In the following, some notations are described in respect to material inelasticity, which are then closing this section.

3.1.5 Extension to material inelasticity

Following the classical plasticity formulation [Simo et al. 1998], the linearized strain tensor is separated in an elastic and an inelastic part with

$$\boldsymbol{\epsilon} = \boldsymbol{\epsilon}^{\mathrm{tot}} = \boldsymbol{\epsilon}^{\mathrm{el}} + \boldsymbol{\epsilon}^{\mathrm{pl}} \tag{3.24}$$

The linear-elastic constitutive relation regarding Hook's law of eq. (3.18) is modified to

$$\boldsymbol{\sigma} = \mathbb{E}(\boldsymbol{\epsilon}^{\mathrm{tot}} - \boldsymbol{\epsilon}^{\mathrm{pl}}) \tag{3.25}$$

The evaluation of the stress state (elastic or inelastic) is performed by using a defined yield condition $f(\boldsymbol{\sigma})$. This depends on the actual stress state $\boldsymbol{\sigma}$ and the material specific yield stress σ_Y as a scalar value. In general, f (without the consideration of hardening effects) can be expressed as

$$f(\boldsymbol{\sigma}) = |\boldsymbol{\sigma}| - \sigma_Y = \begin{cases} < 0 & \text{elastic} \\ = 0 & \text{inelastic} \end{cases} \tag{3.26}$$

Since the evaluation of $f(\boldsymbol{\sigma})$ for multiaxial stress states (resulting from inelastic material behavior) does not exactly result in zero, mapping techniques such as return mapping methods were developed for the computation of the inelastic stress and strain components. One technique is the 'closest point projection' algorithm where the back-mapping starts from an initial elastic trial stress state $\boldsymbol{\sigma}^{trial}$ and goes on in direction of the surface of the elastic stress domain $E_{\boldsymbol{\sigma}}$ (fig. 3.2). This approach is performed at material point level, e.g. at each integration point of a finite element. However, this is numerically expensive

(with an increasing order of finite elements) and more importantly, depending on the complexity of the underlying constitutive law, for which the convergence behavior is not always guaranteed. Additionally, the closed form of these rate-independent (plasticity) formulations require a relation, which quantifies the evolution of the inelastic strains beside the flow condition and the definition of a plastic potential: the so-called plastic flow rule. The change of the material tensor during the inelastic state is characterized by the elasto-plastic tangent modulus. Further evolution equations are pointed out in [Simo et al. 1998] with some algorithmic experience in respect to implicit return mapping techniques by the author shown in [Schrader 2005].
For ductile materials the von-Mises yield condition [Mises 1913] considers the equivalent von-Mises stress in respect to the second invariant of the stress tensor and, by that, evaluates the inelastic stress state as follows

$$\sigma_V = \sqrt{3 \cdot I_2} \tag{3.27}$$

Hence, considering the second invariant of the stress tensor $\boldsymbol{\sigma}$ (with s_{ij} as components of deviatoric part $\boldsymbol{\sigma}^{\mathrm{dev}}$) which is given with

$$I_2 = \frac{1}{2} s_{ij} s_{ij} = {\tau_{xy}}^2 + \tau_{yz}^2 + {\tau_{zx}}^2 - (\sigma_{xx}\sigma_{yy} + \sigma_{yy}\sigma_{zz} + \sigma_{zz}\sigma_{xx}) \tag{3.28}$$

the von-Mises yield condition (assuming isotropic material) results in

$$f(\boldsymbol{\sigma}) = |\boldsymbol{\sigma}| - \sigma_Y = \sqrt{3 \cdot I_2} - \sigma_Y \tag{3.29}$$

in which σ_Y describes the initial state of the plastic flow, e.g considering the tensile strength as a material property used for uniaxial tensile loading conditions.
An alternative flow hypothesis according to the Tresca flow condition can be evaluated

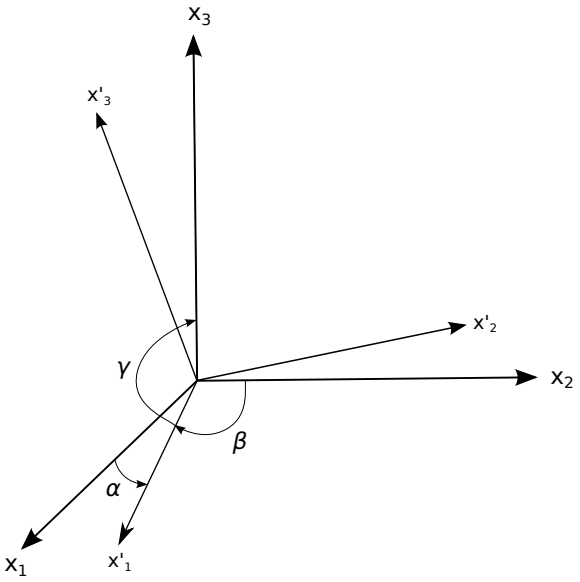


Figure 3.3: Transformation of the original Cartesian to the principle coordinate system.

by

$$f(\boldsymbol{\sigma}) = (\sigma_{max} - \sigma_{min}) - \sigma_Y \tag{3.30}$$

with σ_{max} and σ_{min} indicating the maximum and minimum principle stress defined in the principle coordinate system (fig. 3.3). The corresponding (principle) stress tensor $\hat{\boldsymbol{\sigma}}$ with zero shear components in diagonal form results in

$$\hat{\boldsymbol{\sigma}} = \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{pmatrix} \tag{3.31}$$

with

$$\sigma_{min} \cup \sigma_{max} \in \{\sigma_1; \sigma_2; \sigma_3\} \tag{3.32}$$

The transformation of the stress tensor into the principle stress components considering the transformation matrix $\boldsymbol{T}$ results in

$$\hat{\boldsymbol{\sigma}} = \boldsymbol{T}\boldsymbol{\sigma}\boldsymbol{T}^T \tag{3.33}$$

with

$$\boldsymbol{T} = \begin{pmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{pmatrix} \tag{3.34}$$

Since $\boldsymbol{T}$ is orthogonal, it yields to

$$\boldsymbol{T}^T = \boldsymbol{T}^{-1} \tag{3.35}$$

According to figure 3.3, the cosine of the angles α, β and γ determines the entries of matrix $\boldsymbol{T}$ with

$$t_{11} = cos\alpha \tag{3.36}$$

for x_1' and x_1 and

$$t_{12} = cos\beta \tag{3.37}$$

for x_1' and x_2 and

$$t_{13} = cos\gamma \tag{3.38}$$

for x_1' and x_3, where the angles result from the rotated x_1' axis in relation to the original axis x_1, x_2 and x_3, respectively. To obtain the principle stress state, the solution of the following eigenvalue problem

$$|\sigma_{ij} - \lambda\delta_{ij}| = \begin{vmatrix} \sigma_{11} - \lambda & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} - \lambda & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} - \lambda \end{vmatrix} = 0 \tag{3.39}$$

delivers three eigenvalues λ_i for the cubic equation resulting from the computation of the determinant

$$\det|\sigma_{ij} - \lambda\delta_{ij}| = -\lambda^3 + I_1\lambda^2 - I_2\lambda + I_3 = 0 \tag{3.40}$$

with

$$\sigma_1 = \max(\lambda_1, \lambda_2, \lambda_3) \tag{3.41}$$

and

$$\sigma_3 = \min(\lambda_1, \lambda_2, \lambda_3) \tag{3.42}$$

Considering σ_1 and σ_3 as well as the first invariant of the stress tensor, the second component of the principle stress tensor σ_2 results in

$$\sigma_2 = I_1 - \sigma_1 - \sigma_3 \tag{3.43}$$

with the first invariant

$$I_1 = \sigma_{11} + \sigma_{22} + \sigma_{33} \tag{3.44}$$

3.1.6 The Rankine criterion

Considering the classical plasticity formulation, the yield condition gives the criterion of a limit state of a bounded elastic domain. This limit state can be seen as the initiation of plastic flow in ductile materials for which the von-Mises yield condition is widely used. Nevertheless, there are several criterions, which are evaluating the stress state of the material softening, especially in quasi-brittle materials. For this reason, the Rankine criterion [Carol et al. 2001], which is widely accepted in the field of fracture mechanics, is introduced. The elastic peak following the Rankine criterion can be defined for three cases considering the principle stresses σ_1, σ_2 and σ_3 as well as the tensile strength f_t according to the following equation with

$$f_R(\boldsymbol{\sigma}) = \begin{cases} \sigma_1 - f_t & \sigma_i < 0 \quad i = 2,3 \\ \sqrt{\sigma_1^2 + \sigma_2^2} - f_t & \sigma_i < 0 \quad i = 3 \\ \sqrt{\sigma_1^2 + \sigma_2^2 + \sigma_3^2} - f_t & \sigma_i \geq 0 \quad i = 1,..,3 \end{cases} \tag{3.45}$$

The evaluation of such criterions demands for return mapping techniques for the backtracking of a trial elastic stress state to the initial inelastic stress state of plastic flow or also to the state of smeared crack initiation in respect to physically nonlinear (damage) modeling approaches. Alternatively, the sequential linear analysis is based on the sawtooth softening model [Rots et al. 2006], which is reviewed in chapter 7 and which is adapted in this work for hybrid meshed 3D multiphase specimens. It offers a sophisticated alternative in modeling nonlinear material behavior. Since mapping techniques are

no longer needed, the evaluation of eq. (3.26) is imminent. In this work, the framework of a sequential linear analysis for the stiffness degradation of multiphase composites will be using the Rankine criterion at material point level to detect tipping points for the initiation of material softening mechanism. The notation for the mathematical constructs of continuum mechanics is complemented by some fundamental formulations of the finite element method (FEM) as numerical solution technique for the partial differential equation systems. An introduction is given in the next section.

3.2 Finite element method

The finite element method [Bathe 1995] describes an approximative solution technique for partial differential equations resulting from the discretization of different physical problems. The main approach is to separate the (material) domain of interest G in n finite elements G_e as subparts of G (see fig. 3.4)

$$G = \bigcup_{e=1}^{n} G_e \tag{3.46}$$

Starting with a 3D elasticity problem $\mathcal{V}$ which is bound by domain Ω, this problem can be described by the equilibrium equation

$$\sigma_{ij,j} + b_i = 0 \tag{3.47}$$

with the constitutive relation

$$\sigma_{ij} = C_{ijkl}\varepsilon_{kl} \tag{3.48}$$

and the kinematic equation of

$$\varepsilon_{kl} = \frac{1}{2}(U_{k,l} + U_{l,k}) \equiv \nabla_s(U)_{kl} \tag{3.49}$$

The boundary conditions are subjected to

$$\Omega = \Omega_t + \Omega_u, \qquad \Omega_t \cap \Omega_u = \emptyset \tag{3.50}$$

and

$$\sigma_{ij} n_j = t_i \quad \text{on} \quad \Omega_t \quad u_i = \hat{U}_i \quad \text{on} \quad \Omega_u \tag{3.51}$$

Here, σ_{ij} are the components of the stress tensor, b_i the components of the body force and n_i the unit outward normal. U_i are the displacement field, $\hat{U}_i$ are the displacement

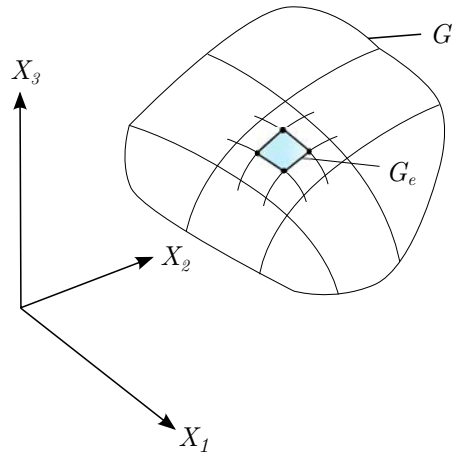


Figure 3.4: Illustration of the material domain G discretized by finite element patches G_e.

based boundary conditions and C_{ijkl} is the material tensor. The variational weak form of eq. (3.47) with zero body force is given as

$$\int_{\mathcal{V}} \delta \nabla_s(U)_{ij} C_{ijkl} \nabla_s(U)_{kl} d\mathcal{V} - \int_{\Omega} \delta U_i t_i d\Omega = 0 \tag{3.52}$$

With the introduction of the strain-displacement operator B_{ij} and u_i as the component of displacements eq. (3.49) changes to

$$\varepsilon_i = B_{ij} u_j \tag{3.53}$$

Furthermore, with the operator N_{ij} (including the shape functions) and u_i as the component of the displacement, the interpolation field results in the relation

$$U_i = N_{ij} u_j \tag{3.54}$$

Respecting eq. (3.53) and eq. (3.54) the weak form of eq. (3.52) changes to

$$\int_{\mathcal{V}} \delta (B_{ij} u_j)^T C_{ijkl} B_{kl} u_l d\mathcal{V} - \int_{\Omega} \delta (N_{ij} u_j)^T t_i d\Omega = 0 \tag{3.55}$$

Reformulation and separation of virtual displacements yields to

$$\delta u_j^T \left(\int_{\mathcal{V}} B_{ij}^T C_{ijkl} B_{kl} d\mathcal{V} u_l - \int_{\Omega} N_{ij}^T t_j d\Omega \right) = 0 \tag{3.56}$$

which is in equilibrium for any virtual displacements. Therewith and with the extraction of the components of the element stiffness

$$K_{ij} = \int_{\mathcal{V}} B_{ij}^T C_{ijkl} B_{kl} d\mathcal{V} \tag{3.57}$$

and the components of the force vector

$$f_i = \int_{\Omega} N_{ij}^T t_j d\Omega \tag{3.58}$$

the final element-based relationship between stiffness and corresponding external forces can be rewritten to

$$K_{ij} u_j = f_i \tag{3.59}$$

Considering the assembly operator $\boldsymbol{R}$ for the element stiffness matrix $\boldsymbol{K}_e^k$ such as

$$\boldsymbol{K}^{gl} = \sum_{k=1}^{n} \boldsymbol{R}^T \boldsymbol{K}_e^k \boldsymbol{R} \tag{3.60}$$

and for the element forces

$$\boldsymbol{f}^{gl} = \sum_{k=1}^{n} \boldsymbol{R} \boldsymbol{f}^k \tag{3.61}$$

the global system of equations in matrix-vector notation is given as

$$\boldsymbol{K}^{gl}\boldsymbol{u}^{gl} = \boldsymbol{f}^{gl} \tag{3.62}$$

The approach in this thesis separates the material region in elastic (e.g. matrix material) and inelastic (e.g. inclusion material and volumetric ITZ) parts to solve efficiently the resulting partial differential equation systems using the FEM. Introducing the elastic-inelastic (element-based) domain split, which is proposed in this work, eq. (3.62) changes to

$$\left(\boldsymbol{K}^{gl}_{(el)} + \boldsymbol{K}^{gl}_{(inel)}\right)\boldsymbol{u}^{gl} = \boldsymbol{f}^{gl}$$

Finally, the element stiffness matrix corresponding to the elastic or inelastic domain is obtained by

$$\boldsymbol{K}_{(el)} = \int_{\mathcal{V}_1} \boldsymbol{B}^T \boldsymbol{C}_{(el)} \boldsymbol{B} d\mathcal{V}_1 \tag{3.63}$$

and

$$\boldsymbol{K}_{(inel)} = \int_{\mathcal{V}_2} \boldsymbol{B}^T \boldsymbol{C}_{(inel)} \boldsymbol{B} d\mathcal{V}_2 \tag{3.64}$$

with $\boldsymbol{C}_{(el)}$ as the elastic and $\boldsymbol{C}_{(inel)}$ as the inelastic material tensor equal to $\mathbb{E}$ as described in section 3.1.3.

3.3 Conclusion

Notations of the continuum mechanics and of the finite element method provide theoretical instruments, which can be applied to the mathematical formulation of the physical problem, especially the material inelasticity resulting from the material heterogenity of a multiphase specimen. The numerical approximation of the partial differential equations resulting from the physical description is based on the standard finite element discretization yielding to the numerical representation of simulation models. Their extension considering material nonlinear effects should enable the simulation of damage-induced multiphase composites. Due to this, the following sections include

- the evaluation of efficient finite element formulations (section 3.4),
- the discretization of the multiphase geometry (section 4.1) as well as
- the partitioning of regular and hybrid finite element meshes (section 4.3).

3.4 Efficient element formulations

3.4.1 Finite element integrands in 3D

Respecting the shape functions N_i of the used element type, the interpolation scheme for the coordinates and for the displacement field of one element in respect to the isoparametric concept in 3D is given as

$$\boldsymbol{X}_e = \sum_{i=1}^{n} N_i(\xi, \eta, \zeta)\boldsymbol{X}_i \tag{3.65}$$

with n as the number of integration points and

$$\boldsymbol{u}_e = \sum_{i=1}^{n} N_i(\xi, \eta, \zeta)\boldsymbol{u}_i \tag{3.66}$$

The gradients of the displacement field are given with

$$\text{grad}\ \boldsymbol{u}_e = \sum_{i=1}^{n} \boldsymbol{u}_i \otimes \nabla_{\boldsymbol{x}} N_i \tag{3.67}$$

being equivalent to the notation of the derivatives with

$$\frac{\partial \boldsymbol{u}_e}{\partial \boldsymbol{X}} = \sum_{i=1}^{n} \frac{\partial N_i(\xi, \eta, \zeta)}{\partial \boldsymbol{X}} \boldsymbol{u}_i \tag{3.68}$$

used to obtain the strains or variations of the strains. The computation of the derivates requires the transformation of the derivatives (using the local coordinates ξ, η and ζ) into a global space of coordinates where the displacement field is defined. The derivatives of the displacement field respecting the local to global transfer of coordinates results in

$$\frac{\partial \boldsymbol{u}_e}{\partial \boldsymbol{X}} = \frac{\partial \boldsymbol{u}_e}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{X}} = \left(\sum_{i=1}^{n} \frac{\partial N_i(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} \boldsymbol{u}_i \right) \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{X}} \quad \text{with} \quad \boldsymbol{\xi} = \{\xi, \eta, \zeta\} \tag{3.69}$$

Considering the interpolation function of eq. (3.65), the derivatives of $\boldsymbol{\xi}$ can be determined by

$$\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{X}} = \left(\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{\xi}} \right)^{-1} = \left(\sum_{i=1}^{n} \frac{\partial N_i(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} \boldsymbol{X}_i \right)^{-1} = \boldsymbol{J}_e(\boldsymbol{\xi})^{-1} \tag{3.70}$$

This is the so-called Jacobi transformation where the Jacobi matrix $\boldsymbol{J}_e$ can efficiently be calculated using the chain rule

$$\boldsymbol{J}_e = \sum_{i=1}^{n} \boldsymbol{X}_i \otimes \nabla_{\boldsymbol{\xi}} N_i = \begin{pmatrix} X_{1,\xi} & X_{1,\eta} & X_{1,\zeta} \\ X_{2,\xi} & X_{2,\eta} & X_{2,\zeta} \\ X_{3,\xi} & X_{3,\eta} & X_{3,\zeta} \end{pmatrix} \tag{3.71}$$

with the computation of the components $X_{n,m}$

$$X_{m,k} = \sum_{i=1}^{n} N_{i,k} X_{mi} \tag{3.72}$$

The integration of the shape functions and their derivatives via the element domain Ω_e to the local space as well as the local to global transformation of coordinates for one element is represented as

$$\begin{aligned} \int_{\Omega_e} g(\boldsymbol{X}) d\Omega_e &= \int_{\Omega_e} g(\boldsymbol{\xi}) \mid \boldsymbol{J}_e \mid d\Omega_e \\ &= \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} g(\xi, \eta, \zeta) \mid \boldsymbol{J}_e \mid d\xi d\eta d\zeta \end{aligned} \tag{3.73}$$

where the numerical approximation of this integral using n integration points and the corresponding weights w_i of the Gaussian quadrature has the following notation

$$\begin{aligned} &\int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} g(\xi, \eta, \zeta) \mid \boldsymbol{J}_e \mid d\xi d\eta d\zeta \equiv \\ &\qquad = \sum_{i=1}^{n} g(\xi_i, \eta_i, \zeta_i) \mid \boldsymbol{J}_e(\xi_i, \eta_i, \zeta_i) \mid w_i \end{aligned} \tag{3.74}$$

Finally, the general shape functions of some 3D finite elements with m nodes according to the isoparametric concept are given as follows

- for the trilinear ($m = 8$) and quadratic ($m = 20$, $m = 27$) hexahedral element

$$N_i(\xi, \eta, \zeta) = \frac{1}{2}(1 + \xi_i \xi)\frac{1}{2}(1 + \eta_i \eta)\frac{1}{2}(1 + \zeta_i \zeta) \quad \text{with} \quad i = 1, ..., m \tag{3.75}$$

- for the linear 4-noded tetrahedral element

$$N_1 = 1 - \xi - \eta - \zeta \tag{3.76}$$

$$N_2 = 1 - \xi - \eta - \zeta \tag{3.77}$$

$$N_3 = 1 - \xi - \eta - \zeta \tag{3.78}$$

$$N_4 = 1 - \xi - \eta - \zeta \tag{3.79}$$

- as well as for the quadratic 10-noded tetrahedral element

$$N_1 = \lambda(2\lambda - 1) \quad N_2 = \xi(2\xi - 1) \tag{3.80}$$

$$N_3 = \eta(2\eta - 1), \quad N_4 = \zeta(2\zeta - 1) \tag{3.81}$$

$$N_5 = 4\xi\lambda, \quad N_6 = 4\xi\eta \tag{3.82}$$

$$N_7 = 4\eta\lambda, \quad N_8 = 4\zeta\lambda \tag{3.83}$$

$$N_9 = 4\xi\zeta, \quad N_{10} = 4\eta\zeta \tag{3.84}$$

with $\lambda = 1 - \xi - \eta - \zeta$.

- for the linear 5-noded pyramid solid element

$$N_i(\xi,\eta,\zeta) = \frac{1}{2}(1+\xi_i\xi)\frac{1}{2}(1+\eta_i\eta)\frac{1}{2}(1+\zeta_i\zeta) \quad \text{with} \;\; i = 1,...,4 \tag{3.85}$$

$$N_5 = \frac{1}{2}(1+\zeta) \tag{3.86}$$

These basic shape functions were then used for the implementation of the standard (full) finite element integration in 3D. Regarding the computational efficiency, the integration concept has been compared to the reduced integration and to the voxel integration technique, which was also applied to the grid-based part of hybrid meshes used in this work.

3.4.2 Reduced integration for 3D finite elements

The most efficient method for the numerical integration of finite elements of 3D elasticity problems is a reduced integration strategy, comparable to [Wriggers 2009], where the necessary number of Gauss integration points is decreased and consequently, the numerical effort and memory demand required. The simplest way to reduce the computing time is to use a one-point integration rule for the element midpoint. By this, the volume integration defined by function g with the natural coordinates ξ, η and ζ leads to

$$\boldsymbol{K}_e^{1\times1} = \boldsymbol{K}_e^{1\times1}\,|_{\xi=\eta=\zeta=0} = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} g(\xi,\eta,\zeta) \mid \boldsymbol{J}_e \mid d\xi d\eta d\zeta \tag{3.87}$$

In general, one distinguishes between two types of reduced integration

- the stabilization-free reduced integration or
- the reduced integration with stabilization

Reviewing the linear tetrahedron and the linear pyramid solid the reduced integration technique will not lead to a rank-insufficient stiffness matrix, if only one integration point is used. Therefore, it is not necessary to use a stabilization technique for the element stiffess matrix compensating the rank decrease. For a linear hexahedral element with eight regular integration points the one-point Gaussian quadrature approximation yields to

$$\begin{aligned} \boldsymbol{K}_e^{1\times1} &\equiv \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} g(\xi_i,\eta_j,\zeta_k) \mid \boldsymbol{J}_{ijk} \mid w_i w_j w_k = \\ &= 8g(\xi_1,\eta_1,\zeta_1) \mid \boldsymbol{J}_{111} \mid \end{aligned} \tag{3.88}$$

element type	interpolation type	point	natural coordinates	weight
		i	ξ,η_i,ζ_i	w_i
tetrahedron solid	linear	1	$\{0.25; 0.25; 0.25\}$	0.166667
hexaeder solid	linear	1	$\{0; 0; 0\}$	2.0
pyramid solid	linear	1	$\{0; 0; -0.5\}$	4.740740

Table 3.1: One-point integration rules for linear tetrahedron, linear hexahedron and linear pyramid solid elements.

with the characteristics as follows

- $n = 1$ as the number of Gaussian points
- $\xi_1 = \eta_1 = \zeta_1 = 0$ as the natural coordinates for the element mid point
- $w_i = w_j = w_k = 2$ as the weighting factors
- $\mid \boldsymbol{J}_{111} \mid$ as the Jacobian determinant

according to table 3.1. Here, the increase in computational efficiency will be offset, since the underintegrated elements alone are no longer stable due to the resulting rank decrease of the element matrix. Hence, stabilization techniques for underintegrated elements were developed. In the following, the hourglass stabilization for a reduced integrated linear hexahedral element will be reviewed.

3.4.3 Hourglass stabilization technique

The basic concept of the stabilization of finite elements results in the eigenvalue analysis applied to a single finite element. Zero eigenvalues are related to rigid body modes, which do not influence the element stiffness. Additional zero eigenvalues are occuring from the so-called hourglass modes which, however, contribute to the element stiffness and hence, artificial stiffness for the underintegrated elements needs to be added for an application of such elements. The final element stiffness matrix $\overline{\boldsymbol{K}}_e$ with stabilization is composed by [Belytschko 1984] with

$$\boldsymbol{K}_e \equiv \overline{\boldsymbol{K}}_e = \boldsymbol{K}_e^{1\times 1} + \boldsymbol{K}_e^{stab} \tag{3.89}$$

The stabilized stiffness matrix for hexahedral elements considering the stabilization vectors $\boldsymbol{\gamma}$ results in

$$\boldsymbol{K}_e^{stab} = \sum_{i=1}^{12} \alpha_i \boldsymbol{\gamma}_i \boldsymbol{\gamma}_i^T \tag{3.90}$$

To construct these stabilization vectors, four hourglass base vectors are introduced

$$\boldsymbol{h}_1 = \{\ 1, -1,\ \ 1, -1,\ \ 1, -1,\ \ 1, -1\ \}^T \tag{3.91}$$

$$\boldsymbol{h}_2 = \{\ 1,\ \ 1, -1, -1, -1, -1,\ \ 1,\ \ 1\ \}^T \tag{3.92}$$

$$\boldsymbol{h}_3 = \{\ 1, -1, -1,\ \ 1, -1,\ \ 1,\ \ 1, -1\ \}^T \tag{3.93}$$

$$\boldsymbol{h}_4 = \{\ 1, -1,\ \ 1, -1, -1,\ \ 1, -1,\ \ 1\ \}^T \tag{3.94}$$

Moreover, the geometry-dependent hourglass shape vectors with their values $\overline{\gamma}_{ij}$ [Flanagan et al. 1981] and [Belytschko 1984] avoiding the hourglass modes are given with

$$\overline{\gamma}_{ij} = h_{ij} - N_{j,k} \sum_{n=1}^{8} x_k^n h_{in} \tag{3.95}$$

where $N_{j,k}$ denotes the derivatives of the shape functions and x_k^n the nodal coordinates. The resulting 12 stabilization vectors can be written as

$$\boldsymbol{\gamma}_i = \{\overline{\gamma}_{k1}, 0, 0, \overline{\gamma}_{k2}, 0, 0, ..., \overline{\gamma}_{k8}, 0, 0\}^T \tag{3.96}$$

with $i = k$ and $k = 1, .., 4$,

$$\boldsymbol{\gamma}_i = \{0, \overline{\gamma}_{k1}, 0, 0, \overline{\gamma}_{k2}, 0, ..., 0, \overline{\gamma}_{k8}, 0\}^T \tag{3.97}$$

with $i = 5, .., 8$ and $k = 1, , , 4$,

$$\boldsymbol{\gamma}_i = \{0, 0, \overline{\gamma}_{k1}, 0, 0, \overline{\gamma}_{k2}, ..., 0, 0, \overline{\gamma}_{k8}\}^T \tag{3.98}$$

with $i = 9, .., 12$ and $k = 1, , , 4$.
According to [Wriggers 2009], the stabilization parameters α_i have less influence on the application in respect to standard 3D engineering problems in solid mechanics and can be selected with high flexibility. In this approach, eq. (3.90) following [Belytschko 1984] is modified using β as a scaled parameter, which is derived from the relation between the maximum eigenvalue $\lambda_{max}^{1\times1}$ of the reduced integrated stiffness matrix and the maximum eigenvalue λ_{max}^{stab} of the stabilized part of the stiffness matrix (resulting from the spectral decomposition). The resulting eigenvalue problems which then need to be solved are

$$(\boldsymbol{K}_e^{1\times1} - \lambda_i^{1\times1}\boldsymbol{I})\boldsymbol{v}_e = \boldsymbol{0} \quad \text{with} \quad \boldsymbol{K}_e^{1\times1} \in R^{n\times n} \quad \text{and} \quad i = 1, ..., n \tag{3.99}$$

for the part of the stiffness matrix respecting the reduced integration and

$$(\boldsymbol{K}_e^{stab} - \lambda_i^{stab}\boldsymbol{I})\boldsymbol{v}_e = \boldsymbol{0} \quad \text{with} \quad \boldsymbol{K}_e^{stab} \in R^{n\times n} \quad \text{and} \quad i = 1, ..., n \tag{3.100}$$

for the stabilizing term. Assuming the stabilization parameters α_i

$$\alpha_i = \beta = \text{const.} \tag{3.101}$$

as well as the modification of eq. (3.90)

$$\boldsymbol{K}_e^{stab} = \sum_{i=1}^{12} \alpha_i \boldsymbol{\gamma}_i \boldsymbol{\gamma}_i^T = \beta \sum_{i=1}^{12} \boldsymbol{\gamma}_i \boldsymbol{\gamma}_i^T \tag{3.102}$$

this leads to the α-free and also β-free stabilized stiffness part $\overline{\boldsymbol{K}}_e^{stab}$

$$\overline{\boldsymbol{K}}_e^{stab} = \sum_{i=1}^{12} \boldsymbol{\gamma}_i \boldsymbol{\gamma}_i^T \tag{3.103}$$

The scaling parameter β results in

$$\beta = \frac{\lambda_{max}^{1\times1}}{\lambda_{max}^{stab}} \tag{3.104}$$

By this, as well as by considering eq. (3.103), the computation of the reduced integrated and stabilized element stiffness matrix according to eq. (3.89) changes to

$$\overline{\boldsymbol{K}}_e = \boldsymbol{K}_e^{1\times1} + \beta\overline{\boldsymbol{K}}_e^{stab} \tag{3.105}$$

The kinematic relation is modified by

$$\boldsymbol{\varepsilon} = \boldsymbol{B}\boldsymbol{u} = \begin{pmatrix} \varepsilon^0 \\ \widetilde{\boldsymbol{\varepsilon}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{B}^{1\times1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B}^{stab} \end{pmatrix} \boldsymbol{u} \tag{3.106}$$

with operator $\boldsymbol{B}^{1\times1}$ resulting from the reduced integration and operator $\boldsymbol{B}^{stab}$ taking the stabilized part into account, and is defined as follows when considering eq. (3.96) up to eq. (3.98)

$$\boldsymbol{B}^{stab} = \begin{pmatrix} \boldsymbol{\gamma}_u^T & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{\gamma}_v^T & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{\gamma}_w^T \end{pmatrix} \tag{3.107}$$

with

$$\boldsymbol{\gamma}_u^T = \sum_{i=1}^{4} \boldsymbol{\gamma}_i^T, \quad \boldsymbol{\gamma}_v^T = \sum_{i=5}^{9} \boldsymbol{\gamma}_i^T, \quad \boldsymbol{\gamma}_w^T = \sum_{i=9}^{12} \boldsymbol{\gamma}_i^T \tag{3.108}$$

Due to the modified material operator $\boldsymbol{C}$, the conjugate stresses $\widetilde{\boldsymbol{\sigma}}$ and extended strains $\widetilde{\boldsymbol{\varepsilon}}$ are introduced by the following constitutive relation

$$\boldsymbol{\sigma} = \boldsymbol{C}\boldsymbol{\varepsilon} = \begin{pmatrix} \boldsymbol{\sigma}^0 \\ \widetilde{\boldsymbol{\sigma}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{C}_e & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}^{stab} \end{pmatrix} \begin{pmatrix} \varepsilon^0 \\ \widetilde{\boldsymbol{\varepsilon}} \end{pmatrix} \tag{3.109}$$

with the fictitious material operator $\boldsymbol{C}^{stab}$ for the stabilized part constructed by the scaling parameter β

$$\boldsymbol{C}^{stab} = \begin{pmatrix} \beta & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \beta \end{pmatrix} \tag{3.110}$$

For the computation of the corresponding eigenvalue problems of eq. (3.99) and eq. (3.100), the von-Mises-Wielandt algorithm [Golub et al. 2000] was used.

3.4.4 Voxel-based integration technique and global matrix assembly

If the finite element mesh is based on a regular grid (which is the case in voxel discretizations (fig. 3.5, left)) respecting a different element length per Cartesian direction (but with constant element lengths over all elements), the resulting element stiffness matrix (assuming material homogeneity) is identical and has to be calculated only once. This is the result of an equivalent Jacobi transformation and provides a constant Jacobi determinant for all (voxel) elements. Furthermore, the nodal incidence of each element or

1.	INIT: $\boldsymbol{K} \in R_3^{nxn}$; $\boldsymbol{x}_0 \notin \boldsymbol{0}$
2.	$\boldsymbol{y}_i = \boldsymbol{K}\boldsymbol{x}_{i-1}$
3.	$\omega_i = \boldsymbol{y}_i^T \cdot \boldsymbol{x}_{i-1}$
4.	$\boldsymbol{x}_i = \frac{\boldsymbol{y}_i}{\|\|\boldsymbol{y}_i\|\|}$
5.	IF $\omega_i - \omega_{i-1} < TOL$ BREAK; ELSE GOTO 2.
6.	SET: $\lambda_{\max} = \omega_i$

Table 3.2: Von-Mises-Wielandt algorithm: Determination of the maximum eigenvalue of matrix $\boldsymbol{K}$.

its corresponding voxel combined with the initial element stiffness matrix is taken into account for an efficient (and also parallel) assembly of the global matrix. For isotropic material and assuming a constant Poisson's ratio eq. (3.63) can then be rewritten as

$$\boldsymbol{K} = \int_{\mathcal{V}} \boldsymbol{B}^T \boldsymbol{C} \boldsymbol{B} d\mathcal{V} = E \int_{\mathcal{V}} \boldsymbol{B}^T \boldsymbol{C}_0 \boldsymbol{B} d\mathcal{V} = E \cdot \boldsymbol{K}_0 \tag{3.111}$$

which leads to a constant initial stiffness $\boldsymbol{K}_0$ and thereby, still enables the variation of the material property E per element. For the assembly of the global stiffness $\boldsymbol{K}^{gl}$ of such voxel discretizations (taking the initial stiffness $\boldsymbol{K}_0$ into account), the following definition describes all stiffness parts as nodal blocks of $\boldsymbol{K}$

$$\frac{1}{E} K_{ij} = \bigcup_{m=1}^{N_{\mathcal{A}}} \mathcal{A}_{ij}^{(m)} = \text{const.} \tag{3.112}$$

In this formula, all components $\mathcal{A}_{ij}^{(m)}$ of one nodal block m are associated with the entries of a n by n nodal block matrix. Here, n is the number of existing degrees of freedom per FE node (e.g. 3 by 3 for three translational d.o.f.s per node) and $N_{\mathcal{A}}$ denotes the number of nodal blocks depending on the order of the finite element used for the initial stiffness

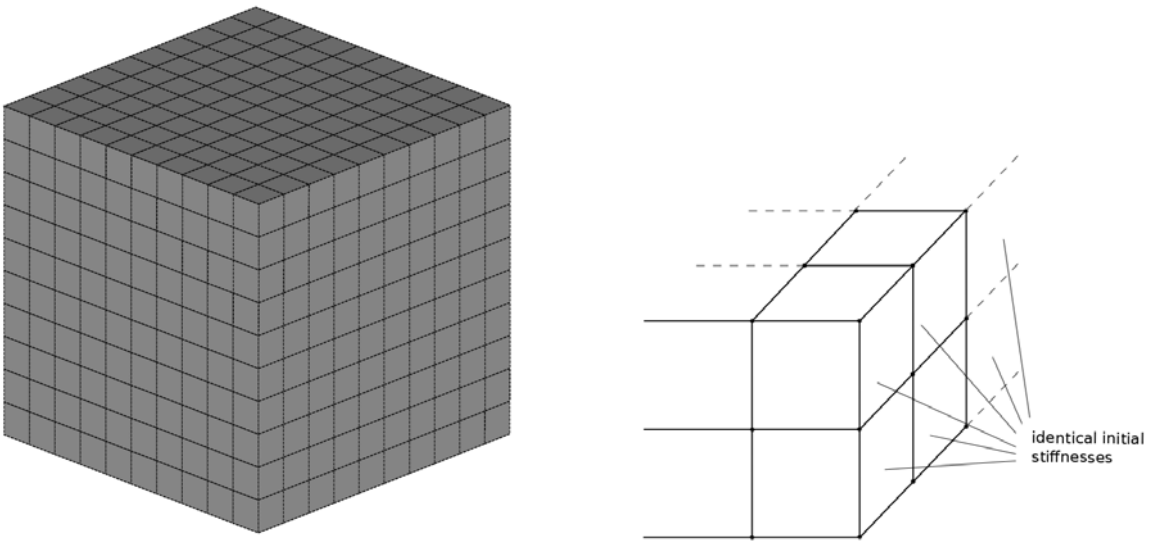


Figure 3.5: Regular voxel discretization (left) with equivalent element stiffnesses.

computation. The nodal FE block m results from node k and node l which are included in the node set $\mathcal{N}_e$ defining the actual finite element with

$$\{k \cup l\} \subset \mathcal{N}_e \tag{3.113}$$

and

$$\mathcal{N}_e = \{N_1, ..., N_N\} \tag{3.114}$$

Therefore, the global assembly based on all nodal blocks of the initial stiffness matrix considering a variation of Young's modulus E of element n can be formulated as

$$K_{ij}^{gl} = \bigcup_{n=1}^{N_e} \bigcup_{m=1}^{N_\mathcal{A}} E^{(n)} \mathcal{A}_{ij}^{(m)} \tag{3.115}$$

Considering the cholesky decomposition scheme, the global stiffness can be expressed as

$$\boldsymbol{K}^{gl} = \boldsymbol{\mathcal{L}}^T \boldsymbol{\mathcal{D}} \boldsymbol{\mathcal{L}} \tag{3.116}$$

However, the storage of the assembled stiffness only considers the upper or the lower triangle matrix of nodal blocks with non-zero entries. The (off-diagonal) nodal block-based storage of the upper or lower tringle matrix $\mathcal{L}$ results in

$$\mathcal{L}_{ij} = \bigcup_{n=1}^{N_e} \bigcup_{m=1}^{N_\mathcal{L}} E^{(n)} \mathcal{A}_{ij}^{(m)} \qquad \text{if } k \neq l \tag{3.117}$$

The block-based storage of the diagonal matrix $\mathcal{D}$ yields to

$$\mathcal{D}_{ij} = \bigcup_{n=1}^{N_e} \bigcup_{m=1}^{N_\mathcal{D}} E^{(n)} \mathcal{A}_{ij}^{(m)} \qquad \text{if } k = l \tag{3.118}$$

The number of nodes N of the finite element (assuming a fixed element type for the voxel discretization) determines the number of off-diagonal nodal blocks N_L with

$$N_L = \frac{1}{2} \cdot (N^2 - N) \tag{3.119}$$

and the number of diagonal nodal blocks N_D with

$$N_D = N \tag{3.120}$$

The storage scheme, denoted as nodal compressed sparse row storage (*ndcsr*), will be applied for the distributed computation of the global equation system, particularly suitable for repeated matrix-vector operations, where more algorithmic details are given in chapter 5.

3.4.5 Notes on nodally integrated finite elements and smoothed finite element methods (S-FEM, FS-FEM, ES-FEM)

A new direction of improving the element formulation in respect to the classical element integration concept, like e.g. the Gaussian quadrature, is based on a concept of nodally integrated finite elements applicable for 2D and also 3D finite element problems [Tanaka et al. 2006], [Liu et al. 2009]. The element integration is based on the nodes defining the element edges, the element faces as well as the corresponding volume of the finite element. The nodal integration directly considers the nodal coordinates as limit values for the integration over the defined edges and faces and, additionally, a special smoothing function which fulfills the partition of unity requirement, is introduced. The FS-FEM technique [Nguyen-Thoi et al. 2009] improves this approach by a strain smoothing technique applied for each coincident face of two different 3D finite elements defining a smoothed volume, which is considered for the nodal integration. The computational effort for 3D finite element meshes using S-FEM concepts is numerically more expensive than the conventional quadratures based on integration points. This results from an increased bandwidth of the global system of equations as stated also in [Nguyen-Thoi et al. 2009], where the FS-FEM technique was applied to 3D visco-elastoplastic mechanical problems. Even though an increased accuracy was achieved in respect to the FE error estimation, but due to the lack of performance increase in 3D the implementation of such integration techniques was not considered in this work.

3.4.6 Concluding remarks

With the basic constructs of continuum mechanics, a FE based discretization technique considering three phases of a multiphase specimen may result in a hybrid mesh. There, each distinct phase is described by a specific material behavior, which is enabled by the elastic-inlastic domain split of the whole specimen. The hybrid usage of grid-based and aligned meshing techniques in 3D demands to efficient element integration techniques such as the reduced integration and a grid-based voxel element integration, which were reviewed in this chapter. Additionally, a scalable nodal block based storage of the finite element data may enable a fast simultaneous element integration and the assembly of the global matrix with improved speed-up and scalability by its parallel execution. For the nonlinear simulation model this is a fast technique for the nodal block based re-integration of the finite elements, which are involved with a change in the material tensor (e.g. if nonlinear material effects are considered) and with that, a fast modification of the distributed coefficient matrices is being proposed. This is also an important preliminary step for the adaption of a scalable solver strategy, being suitable for its application on high-performance computers. Due to this, numerical experiences regarding a hybrid meshing technique and also the partitioning approach for the application of domain decomposition methods will be presented in the following chapter.

Chapter 4

Numerical discretization of multiphase materials

In this work, the main focus is on the discretization of 3D multiphase materials based on a volumetric interfacial transition zone (VITZ) in high resolution, coupling different material phases, e.g. the matrix material and the inclusion material. For the VITZ, an aligned meshing procedure is prefered, which enables the necessary discretization among the material boundaries of inclusions and the smooth transition as well as the conform connection to a regular grid. The main research interest is to get an inside view of the material behavior of the VITZ in 3D, especially the successive process of degradation in the case of initiated damage in multiphase composites.

4.1 Discretization: Multiphase geometry and meshing in 3D

4.1.1 Introduction

An initial underlying grid, which is proposed for obtaining the hybrid mesh, is describing the matrix material zone and is meshed as coarse as possible. By this, an imminent reduction of the number of nodal d.o.f.s may be realized (compared to the standard meshing techniques applied to the whole specimen resulting in one irregular mesh only). Additionally, a linear-elastic constitutive law is applied to the matrix zone, also decreasing the number of history data to be stored. Consequently, the reduced numerical effort with respect to the evaluation of strain and stress states (in conjuction with return mapping techniques during a nonlinear simulation framework) is decreased. This results in an overall higher computational efficiency, which constitutes another aim of this work. Furthermore, the technique of an initial elastic-inelastic domain split (fig. 4.1 (1)), divides the structural specimen in zones of grid-based matrix material (with linear-elastic structural behavior) and irregular meshed bounding boxes including inclusions and its surrounding interfacial transistion zone, the VITZ. This zone may be considered as the weakest coupling link for the bonding between the matrix and the inclusion material,

where the initiation and accumulation of micro-cracks mostly starts and thereby, the material behavior of the whole specimen is significantly influenced and appropriated up to the point of the abrupt structural failure. Due to this, the material behavior of the VITZ may be assumed to be physically nonlinear. This approach is a preliminary step to consider computer-tomographic (ct) based voxel models, see figure 4.1 (2), where regions of main interest may be replaced by a large-scale aligned mesh with a high resolution in a similar way. Additionally, in such models, the random based heterogeneity of the material properties of the matrix material may also be considered as illustrated in figure 4.1 (3, middle). The combination of these techniques then results in hybrid meshed and partitioned multiphase specimens ([Mandel 1994], see fig. 4.1, bottom).

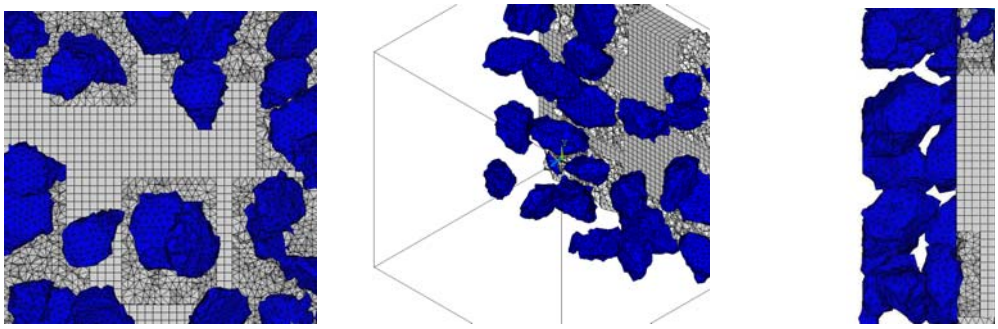
4.1.2 Inclusion-matrix geometry model

Establishing a numerical simulation model for heterogeneous materials on the mesoscale requires two basic steps: firstly, the description of the heterogeneous geometry and secondly, its effective discretization. In a first approach, the chosen geometry modeling is based on the Delaunay tesselation [Caballero et al. 2006], appropriating the final shape of the inclusion. The Delaunay tessilation has the main advantage of capturing the heterogeneous nature of the shape of composites more precisely, as this is not the case for ellipsoidal or spherical modeling approaches of such entities [Häfner et al. 2003], [Wriggers et al. 2006]. The resulting heterogeneous geometries consist of three distinct phases at mesoscale (fig. 4.2):

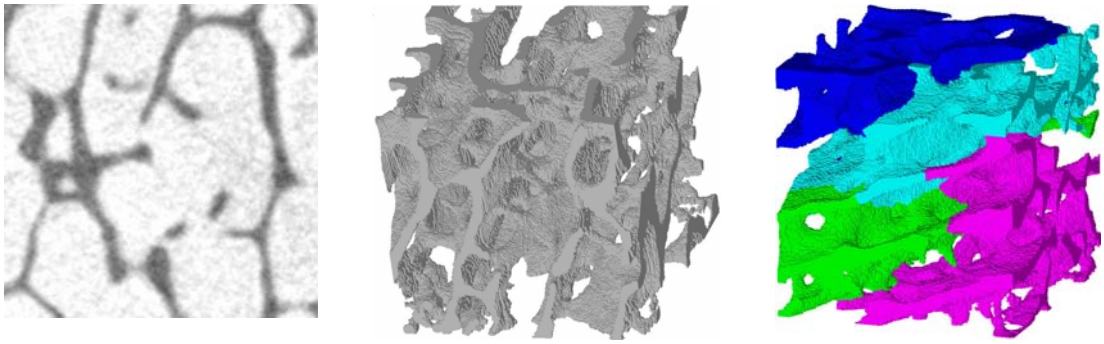
- the inclusions of any irregular shape (zone 1)
- which are embedded in homogeneous matrix material (zone 2)
- a volumetric interfacial transition zone (VITZ) between inclusion and matrix material (zone 3)

Two different strategies allow the generation of these types of heterogeneous models: Either by direct transferring the 3D image data ([Garboczi 2002], [Shan et al. 2004]), namely the voxel models, on the appropriate scales [Kim et al. 2003] or the models may artificially be generated based on the statistical information of their geometry [Wang et al. 1999]. The proposed modeling technique of this work follows the second approach. Here, a special algorithm randomly selects different area-non-coadjacent or cone-point-adjacent tetrahedrons starting with the coarsest triangulation, respecting specific parameters: These can be the number of selections, the factor of density package as the volume ratio or the kind of selection of coadjacent connected tetrahedrons to vary the quantity and arrangements of the final inclusions (see fig. 4.3, fig. 4.4 and fig. 4.6). The degenerated geometry in shapes of a single inclusion (fig. 4.5, 4.7) is then obtained by cutting through the selected tetrahedron with a fixed distance from each cone point. To improve the performance of nonlinear simulations in 3D, one needs to differenciate between the elastic, (e.g. the region

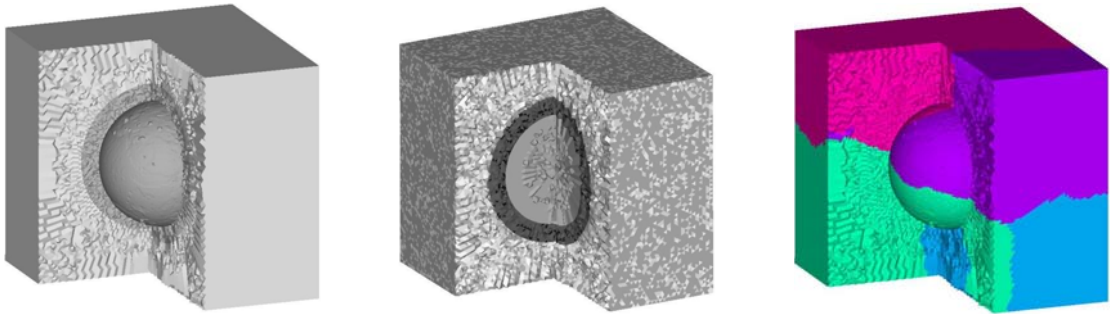
Elastic-inelastic domain split (1):



Voxel based discretization and partitioning (2):



Heterogeneous materials and random distribution of material properties (3):



Combination (1)-(3):

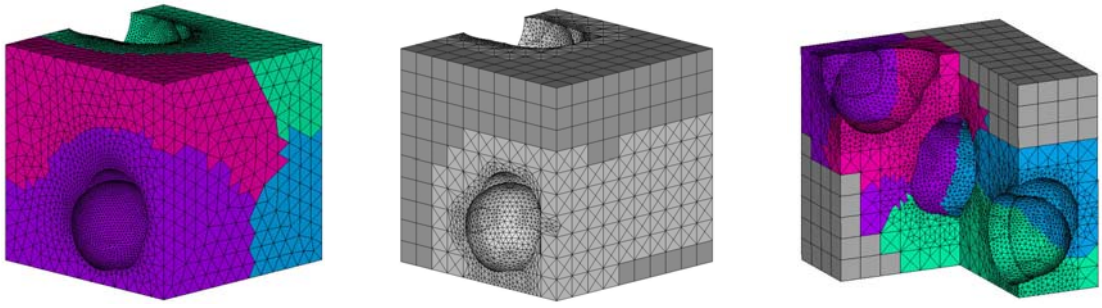


Figure 4.1: Combining different discretization and decomposition techniques.

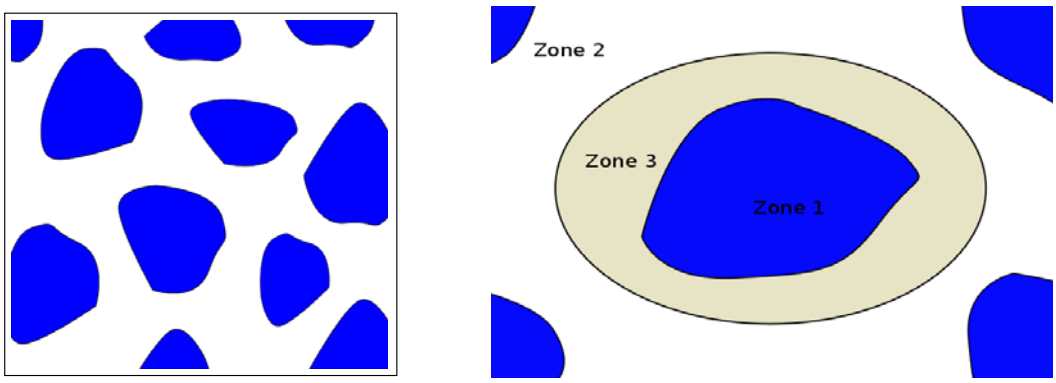


Figure 4.2: Initial inclusion-matrix geometry (left) and heterogeneous modeling with three distinct phases: inclusion (zone 1), matrix material (zone 2) and volumetric interfacial transition zone (zone 3).

of homogeneous matrix material) and the inelastic regions (e.g. the aggregates or inclusions and the volumetric ITZ around them, where damage effects are mainly initiated). Each inclusion has its own bounding box, which represents a separated material phase, if the volume of the corresponding inclusion is excluded: a volumetric interfacial transition zone. The reduction of the orthogonal grid by such bounding boxes describes the volume of the homogeneous matrix respecting the linear-elastic material behavior. Finally, this approach leads to a geometrically three-phase as well as heterogeneous modeling of multiphase materials with different shape and sizes of the inclusions (fig. 4.8), [Schrader et al. 2011]. In these examples, the volume ratio of (non-smeared and coarse mesh based) inclusions totals to approximately 10%.

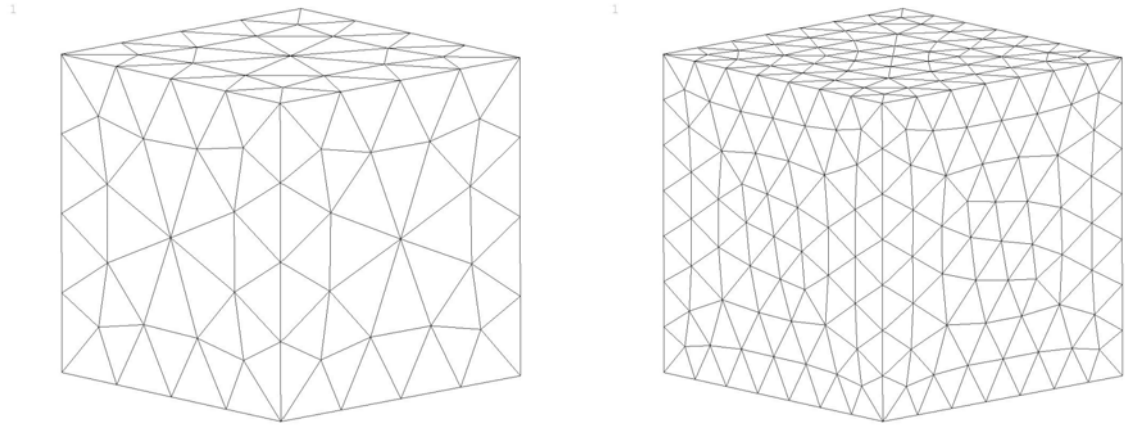


Figure 4.3: Coarse and fine Delaunay triangulation to obtain different aggregate shapes and sizes.

4.1.3 Hybrid 3D meshing techniques

Here, firstly, the boundary between the inclusions und their bounding boxes and secondly, the bounding boxes and the inclusions themself are converted to a Delaunay mesh. After,

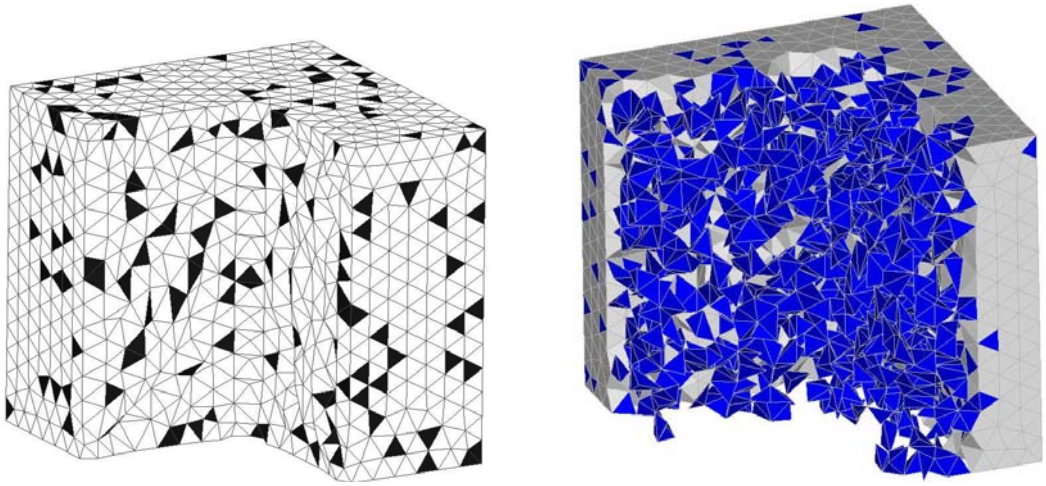


Figure 4.4: Random based selection of 13% of total tetrahedrons.

Figure 4.5: Geometrical degenerated tetrahedrons.

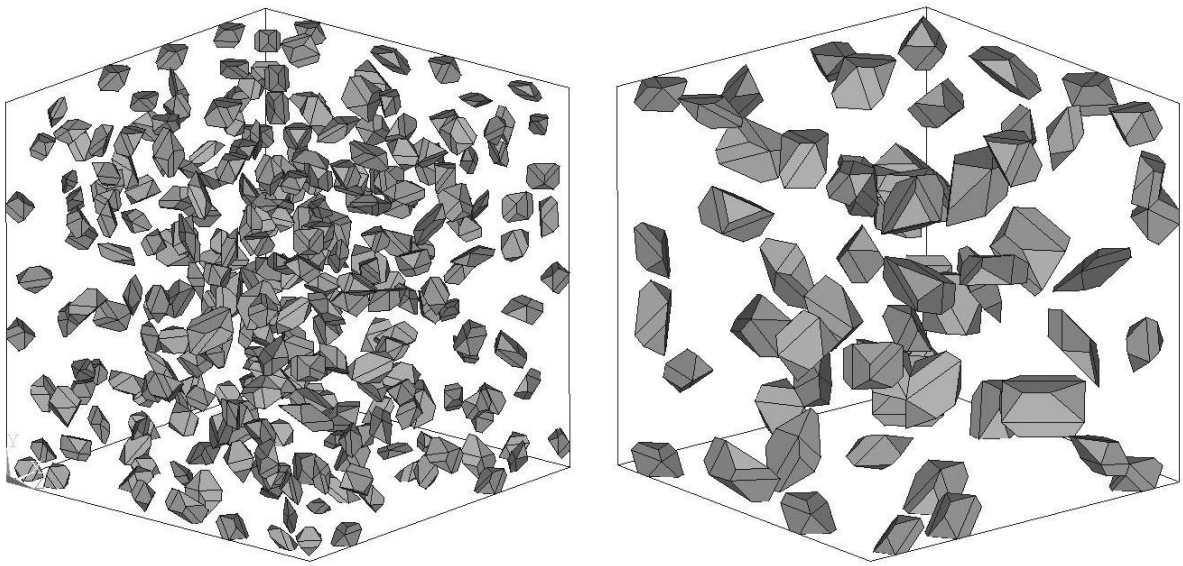


Figure 4.6: Geometrical models of embedded inclusions made of degenerated tetrahedrons of two different triangulations.

an element shifting enables the modification of the number of elements belonging either to the inclusions or to the volumetric ITZ. Consequently, this yields to an improvement of the irregularity of the meshed shapes and results in an increase of the volume ratio.

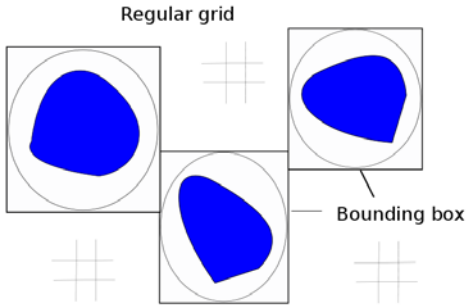


Figure 4.7: Embedding of bounding boxes (including the aligned mesh) in the regular matrix grid.

The remaining orthogonal grid then represents the linear-elastic matrix material, which is converted to a regular mesh based on hexahedral elements with linear shape functions (as illustrated in fig. 4.8). The elastic region is to be meshed as coarse and regular as possible to reduce the number of d.o.f.s. Additionally, the stiffness of this domain is assumed to be constant during the simulation time in respect to linear-elasticity, where return mapping techniques are not required, and hence, the simulation time is reduced as well as it is not necessary to update the material tensor and the stiffness matrix of the corresponding elements of this domain. Consequently, a structured grid discretization is applied in the matrix region and the bounding boxes in inclusion regions with its interfacial transition zone are created. The material region within the boxes is then discretized by an aligned mesh and the total volume of the cube is discretized as an orthogonal grid (fig. 4.8, right).

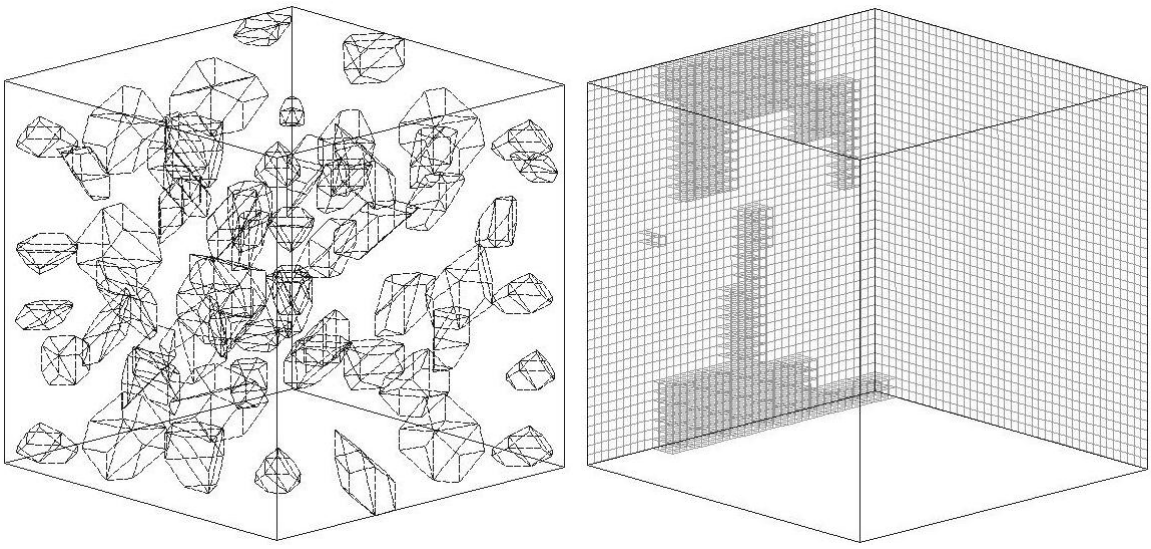


Figure 4.8: Geometrical degenerated tetrahedrons as initial inclusions obtained from the Delaunay triangulation (left) and the underlying orthogonal grid with the detection of inclusion-free volume (right).

4.2 FE discretization with initial elastic-inelastic domain split

The aligned meshing technique for the VITZ ensures that there are no material discontinuities within the elements. Since the cracking phenomena in quasi-brittle materials, specially in high-performance concrete (UPC), occurs not only in areas around, but also within the major aggregates, the material behavior may be described as nonlinear. Therefore, it is necessary to separate the discretization of the aggregates and their adjazent zones, acting as a kind of volume interfacial transition zone with different material properties and specific material behavior based on a fine tetrahedron mesh. The advantage is the material decomposition of the total structure in separate phases such as the aggregates and the volumetric interfaces (inelastic) as well as the matrix material (linear-elastic), resulting in the proposed elastic-inelastic domain split. The condensation of d.o.f.s related to the interior matrix, denoted as ^{el}, is performed by using the *Schur complement* method, since here the connecting boundary between matrix and volumetric ITZ is assumed to be known. Based on the linear-elastic behavior of the matrix or the mortar phase, the entries of the Schur complement operator are constant during the total nonlinear computation procedure. The reduced global problem (iteratively be solved in respect to the d.o.f.s of the interior inelastic problem only) is denoted as ^{inel} and the d.o.f.s of the connecting boundaries are denoted as $_b$. The global reordered FE problem, considering eq. (3.62) of chapter 3 according to interior $_i$ and boundary $_b$ d.o.f.s yields to

$$\boldsymbol{f} = \boldsymbol{K}\boldsymbol{u} = \left(\boldsymbol{K}^{(el)} + \boldsymbol{K}^{(inel)}\right)\boldsymbol{u} \tag{4.1}$$

which in detail leads to

$$\boldsymbol{f} = \left(\begin{pmatrix} \boldsymbol{K}_{ii} & \boldsymbol{K}_{bi}^T & \boldsymbol{0} \\ \boldsymbol{K}_{bi} & \boldsymbol{K}_{bb} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}^{(el)} + \begin{pmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{K}_{bb} & \boldsymbol{K}_{bi} \\ \boldsymbol{0} & \boldsymbol{K}_{bi}^T & \boldsymbol{K}_{ii} \end{pmatrix}^{(inel)} \right) \boldsymbol{u} \tag{4.2}$$

and

$$\boldsymbol{u} = \begin{pmatrix} \boldsymbol{u}_i^{(el)} \\ \boldsymbol{u}_b \\ \boldsymbol{u}_i^{(inel)} \end{pmatrix} \quad \text{and} \quad \boldsymbol{f} = \begin{pmatrix} \boldsymbol{f}_i^{(el)} \\ \boldsymbol{f}_b \\ \boldsymbol{f}_i^{(inel)} \end{pmatrix} \tag{4.3}$$

with the assembled stiffness matrix $\boldsymbol{K}$, the nodal force vector $\boldsymbol{f}$ and the vector $\boldsymbol{u}$ for the unknown nodal d.o.f.s and the corresponding block matrices $\boldsymbol{K}_{nm}$. Respecting the phase separation during the FE discretization, with ^{v} indicating the volumetric ITZ fraction, ^{a} the aggregates and ^{i} the interfacial boundary surface between volumetric ITZ and the aggregates, the nonlinear assembled matrix $\boldsymbol{K}^{(inel)}$ may be written as

$$\boldsymbol{K}^{(inel)} = \begin{pmatrix} \boldsymbol{K}_{bb} & \boldsymbol{K}_{bi} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{K}_{bi}^T & \boldsymbol{K}_{ii}^v & \boldsymbol{0} & \boldsymbol{K}_{ij}^v \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{K}_{ii}^a & \boldsymbol{K}_{ij}^a \\ \boldsymbol{0} & \boldsymbol{K}_{ij}^{v\,T} & \boldsymbol{K}_{ij}^{a\,T} & \boldsymbol{K}_{jj}^i \end{pmatrix}^{(inel)} \tag{4.4}$$

Consequently, if discrete (zero-thickness) interface elements between domains (denoted by ^{v} and ^{a}) are inserted, the block $\boldsymbol{K}^i_{jj}$ corresponds to an assembled matrix resulting from all interface elements. For the (static) condensation of the elastic d.o.f.s, the first row of eq. (4.2) is used to isolate $\boldsymbol{u}_i^{(el)}$

$$\boldsymbol{u}_i^{(el)} = \boldsymbol{K}_{ii}^{(el)^{-1}} \left(\boldsymbol{f}^{(el)} - \boldsymbol{K}_{bi}^{(el)^T} \boldsymbol{u}_b \right) \tag{4.5}$$

Respecting the above, and also the second row of eq. (4.2), the explicit expression of the condensed operator $\boldsymbol{\Xi}^{(el)}$ of the elastic domain is given as

$$\boldsymbol{\Xi}^{(el)} = \boldsymbol{K}_{bb}^{(el)} - \boldsymbol{K}_{bi}^{(el)} \boldsymbol{K}_{ii}^{(el)^{-1}} \boldsymbol{K}_{bi}^{(el)^T} \tag{4.6}$$

and the condensed nodal force vector of the elastic domain results in

$$\boldsymbol{f}_b^{(el)} = -\boldsymbol{K}_{bi}^{(el)} \boldsymbol{K}_{ii}^{(el)^{-1}} \boldsymbol{f}_i^{(el)} \tag{4.7}$$

Thus, the inelastic problem of condensed d.o.f.s in the elastic domain is obtained by

$$\begin{pmatrix} \boldsymbol{\Xi}^{(el)} + \boldsymbol{K}_{bb}^{(inel)} & \boldsymbol{K}_{bi}^{(inel)} \\ \boldsymbol{K}_{bi}^{(inel)^T} & \boldsymbol{K}_{ii}^{(inel)} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_b \\ \boldsymbol{u}_i^{(inel)} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_b \\ \boldsymbol{0} \end{pmatrix}^{(el)} + \begin{pmatrix} \boldsymbol{f}_b \\ \boldsymbol{f}_i \end{pmatrix}^{(inel)} \tag{4.8}$$

The idea of the initial elastic-inelastic domain split causes the question of the hybrid decomposition or substructuring technique enabling a scalable computation of heterogeneous specimens. This and also the generation of an inital partitioned damage zone are described in the next section.

4.3 Graph based FE mesh partitioning

4.3.1 FE mesh partitioning for static load balancing

For an ideal load-balanced performance of numerical parallel computation using the FE method, there exist two opportunities: Either the partitioning of the finite element mesh (non-overlapping or overlapping)

- with a fixed number of equal subdomains, in regards to the number of FE nodes,
- the domain-wise numerical integration of finite elements,
- the domain-wise assembly of the global coefficient matrix and
- the application of substructuring or domain decomposition methods for the numerical solution of the global equation system

or the direct partitioning of the global coefficient matrix by

- the application of parallel solver techniques (with implicit substructuring or domain decomposition methods) for the numerical computation of the distributed global system of equations.

In this thesis, the first approach has been chosen using the initially decomposed finite element mesh (which may be regular or irregular, aligned or grid-based) with the opportunity to consider different types of finite elements in 2D or 3D. For the mesh partitioning, the sequential open-source library METIS [Karypis et al. 1998] considering one element type and an improved version [Karypis et al. 2011] for mixed finite element meshes involving different types of finite elements has been applied. A parallel MPI-based partitioning library (PARMETIS) is also available [Schloegel et al. 2002]. This leads to ideal load-balanced partitionings especially for the aligned meshed compounds. The detection and storage of the coupling nodes are independently performed (without the partitioning library) and therefore, the communication overhead between coupled subdomains is limited to the number of the coupling nodes connecting the boundaries of the adjacent domains. This information is then stored in a matrix $\boldsymbol{M}_c$

$$\boldsymbol{M}_c = \begin{pmatrix} m_{2,1} & & & \\ m_{3,1} & m_{3,2} & & \\ \dots & \dots & & \\ m_{n,1} & m_{n,2} & \dots & m_{n,n-1} \end{pmatrix} \tag{4.9}$$

with the conditions for m_{ij}

$$m_{ij} = \begin{cases} 1 & \Gamma_i \cap \Gamma_j \neq 0 \\ 0 & \text{else} \end{cases} \tag{4.10}$$

where entry m_{ij} is set to 1, if the domain i is connected to domain j, otherwise it is set to 0. Matrix entries m_{ii} on the main diagonal are not considered.
Basically, METIS includes algorithms, which enable the conversion of a finite element mesh into sparse graphs. The corresponding graph of a finite mesh is then to be used to compute nodal or dual partitionings as illustrated in figure 4.10 for an academic three-phase specimen based on a tetrahedral mesh. The partitioning is computed in three tasks (fig. 4.9): the coarsening of the graph, the computation of an initial partitioning of the coarse graph and the un-coarsening phase for a successively multilevel refinement of the partitioning applied for the larger graphs.

4.3.2 FE mesh partitioning for dynamic load balancing

The dynamic load-balancing is necessary, if a change in the partial differential equations occurs and a change in the finite element mesh is consequently imminent. In the field of continuum mechanics, the simulation of physical or material nonlinearities in heterogeneous materials considering damage effects often leads to a localization of the damage zone where the numerical effort for the subdomains involved increases disproportionately. Due to this, the dynamic repartitioning is then to be used during the nonlinear simulation (e.g. as an adaptive partitioning strategy). In this work, a simultaneous repartitioning of

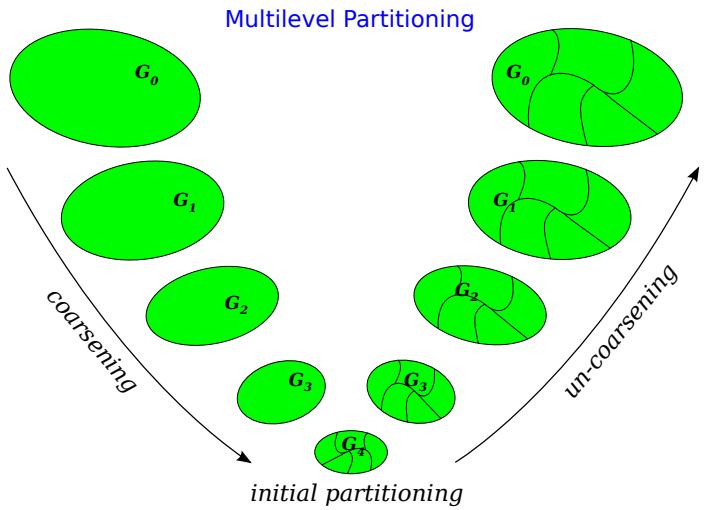


Figure 4.9: Load-balanced mesh partitioning based on sparse graphs: Left: Coarsening the initial sparse graph G_0 (representing the finite element mesh) to obtain an initial coarse partitioning of the smaller graph G_4. After, incremental un-coarsening of G_4 and its corresponding partitioning to generate the final partitioning of the original graph G_0.

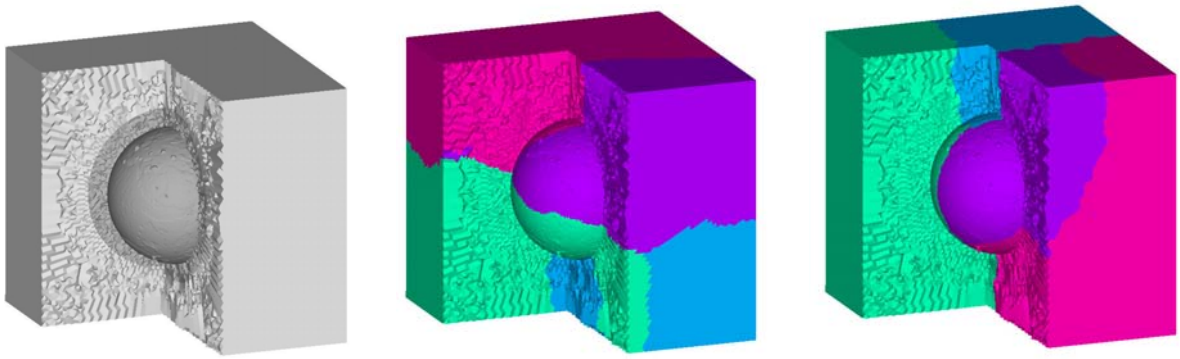


Figure 4.10: Left: Initial three-phase matrix-inclusion system applying the mesh partitioning with nodal (middle) and dual mesh partitioning (right) results of the library METIS (four subdomains).

an approximated damage zone, which has initially been described as inelastic, is applied. Further details to this approach are given in the following section.

4.3.3 Partitioning of hybrid meshes respecting a load-balanced damage zone

Before the mesh partitioning is performed, a special elastic-inelastic domain split is applied to improve the performance of the computational model illustrated in figure 4.11. In this approach, proposing a nonlinear simulation model, material nonlinear effects are considered for the aligned mesh of the volume interfacial transition zone as well as for the embedded inclusions. Consequently, a nonlocal response regarding the damage effects is induced by a smaller domain and therewith, the overall numerical effort for the nonlinear computational model is decreased. Due to the irregularity of such potential damage zones, the aligned mesh including the VITZ in high resolution has to be (load-balanced)

partitioned, if the scalability of such a simulation model is also proposed. Thereby, stress and strain state evaluation can simultaneously be executed and moreover, the change of the mesh characteristics in this region (according to the induced damage effects, the resulting crack band formation and/or failure of elements) can be analysed dynamically during the simulation runtime. Thus, the repartitioning of such irregular inelastic domains may be computed independently of the initial partitioning of the total mesh also yielding to nodally equal-sized subdomains, suitable for the load-balanced substructuring of heterogeneous and hybrid meshed specimens. The partitioning algorithms for mixed meshes (fig. 4.13) was adapted from the latest stable METIS release [Karypis 2011], Version 5.0.2. Furthermore, the elastic-inelastic domain split of a structure or specimen, as illustrated in figures 4.11, 4.12 and 4.13, yields to a regular meshed domain with linear-elastic material behavior. This elastic domain, which is meshed as coarse and regular as possible, can be adapted to an efficient voxel integration technique. Consequently, the element stiffness matrix is to be stored only once and does not have to be updated for the assembled stiffness matrix, which is usually necessary. Additionally, this reduces the numerical effort as well as the simulation time required.

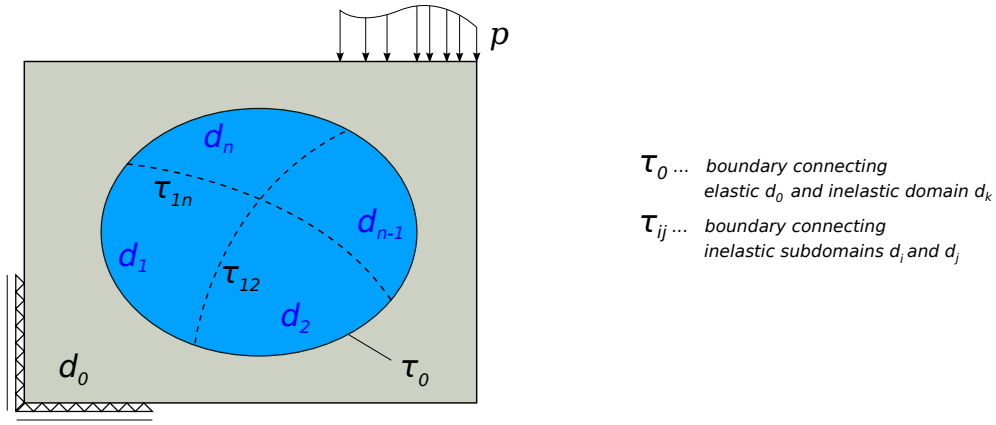


Figure 4.11: Consideration of the heterogeneity during domain decomposition: One elastic domain d_0 including the inelastic region partitioned in four domains d_1 till d_4.

4.3.4 Parameter evaluation and performance of mesh partitioning algorithms

METIS offers the possibility to activate several combinations of parameters (tab. 4.1) which quantitatively and qualitatively lead to different mesh partitionings. The selected parameters of the following table have the greatest influence on the computational time needed to generate the substructuring. The parameter set for the evaluation of the METIS software package (as illustrated in table 4.2 (for a homogeneous mesh) and table 4.3 (for a hybrid mesh)) shows the influence on the computational time necessary for generating the partitioning with the lowest and equal distributed number of coupling nodes. In both cases, the default parameter set (tab. 4.2, third row) delivers the best performance simultaneously with the lowest number of finite element nodes of a connecting boundary

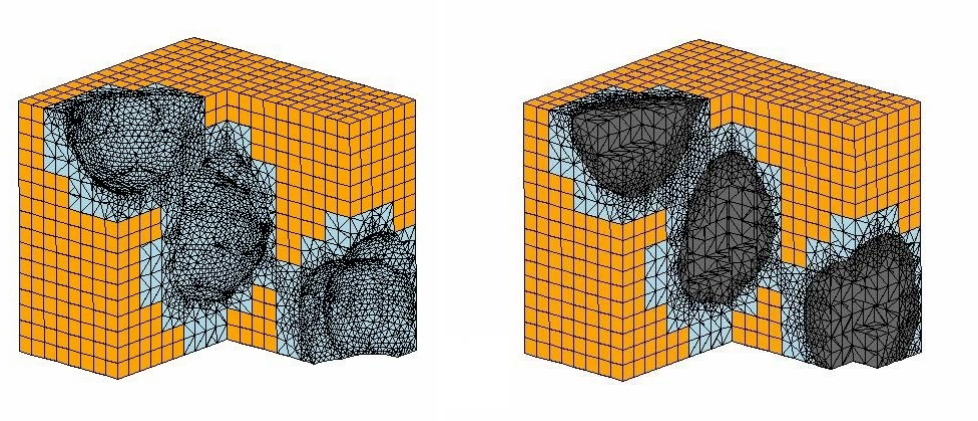


Figure 4.12: Consideration of the heterogeneity in domain decomposition: Regular grid-based elastic domain and aligned mesh of the inelastic region. Left: Aligned mesh of VITZ and irregular mesh of inclusions. Right: aligned mesh transition into inclusions.

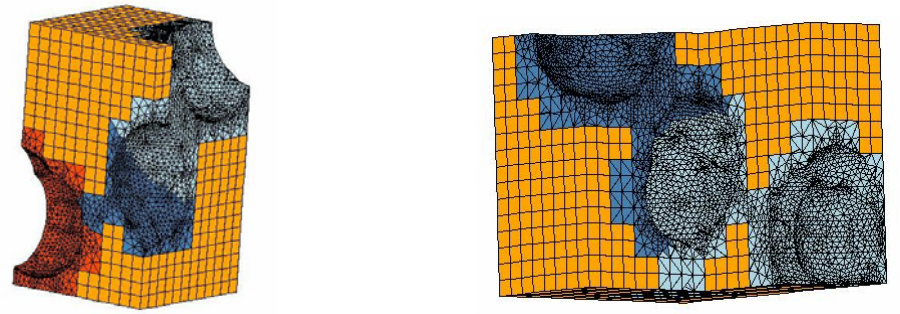


Figure 4.13: Consideration of the heterogeneity in domain decomposition: Nodal mesh partitioning applied for the aligned mesh as a decomposed volumetric interfacial zone.

option	**task**
METIS_OPTION_GTYPE	specifies the graph type (nodal or dual)
METIS_OPTION_PTYPE	specifies the partitioning type
METIS_OPTION_OBJTYPE	specifies edge cut or communication minimization
METIS_OPTION_CTYPE	specifies the coarsening strategy
METIS_OPTION_IPTYPE	determines the algorithm used during initial partitioning

Table 4.1: Parameter options for the mesh partitioning algorithms of the open-source library METIS.

between the different subdomains. Regarding the METIS documentation the parameters *ncut*, *nseps* and *niter* were used as default values. The parameter $gtype = dual$ is the default setting and delivers the best quantitative values for homogeneous and hybrid FE meshes in respect to the number of FE nodes per domain and the number of coupling nodes as well as the reduced load imbalance. For further information, the METIS manual (version 5.0.2) is recommended [Karypis 2011].

ptype		**objtype**		**ctype**		**iptype**		**speed-up**
rb	kway	cut	vol	rm	shem	grow	random	
x		x		x		x		0.69
x		x		x			x	0.67
x		x			x	x		*0.64*
x		x			x		x	-
	x	x		x		x		0.93
	x		x	x		x		0.73
	x	x			x	x		0.96
	x		x		x	x		0.79

Table 4.2: Parameter and performance evaluation: Speed-ups using different option sets of METIS for the partitioning of homogeneous meshes.

ptype		**objtype**		**ctype**		**iptype**		**speed-up**
rb	kway	cut	vol	rm	shem	grow	random	
x				x		x		0.69
x				x			x	0.72
x					x	x		0.69
x					x		x	*0.65*
	x	x		x				1.01
	x	x			x			1.00
	x		x	x				0.93
	x		x		x			0.97

Table 4.3: Parameter and performance evaluation: Speed-ups using different option sets of METIS for the partitioning of hybrid meshes.

In chapter 5, some substructuring and domain decomposition techniques are reviewed presenting the fundamental formulations for the numerical parallel computation of various decomposed problems discretized for the time-independent solution.

Chapter 5

Linearized (time-independent) solution methods based on domain decomposition

The presented numerical discretization of multiphase material using the finite element method requires efficient and scalable solution techniques for the numerical computation of the resulting linearized equation systems. In this chapter, iterative solver techniques, typical substructuring and domain decomposition methods are reviewed and several numerical experiments regarding the Dirichlet-Neumann substructuring, the Schur complement method and also the finite element tearing and interconnecting dual-primal method (FETI-DP) are evaluated. Furthermore, the preconditioned approach for the conjugate gradient method is described in repect to distributed computing for the application in high-performance computing frameworks. Verified numerical results are then presented in chapter 6.

5.1 Numerical computation of linear systems of equations

Following the standard displacement based finite element method (according to section 3.2) a linear system of equations is given with

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{f} \tag{5.1}$$

where $\boldsymbol{K} \in R^{n \times n}$ is the global stiffness matrix, $\boldsymbol{u} \in R^n$ the global displacement vector and $\boldsymbol{f} \in R^n$ the global load vector. If $\boldsymbol{K}$ is a regular, symmetric and positive definite matrix the above equation can be solved numerically using direct or iterative solver techniques which are available in sequential and parallel algorithms. The parallelization technique of such systems results in the decomposition of the underlying finite element discretization by applying substructuring methods for which an overview is given in [Mandel 1994], [Farhat et al. 1994] and [Toselli et al. 2005].

Before classical approaches for (non-overlapping) domain decomposition (DD) are presented, the next section gives an overview on iterative solution techniques.

5.2 Special iterative solution methods

In general, the discretized problem (using FEM, extended FEM, or other numerical approximation techniques) leads to a linear (or nonlinear) system of equations, where specific characteristics of such systems (matrix bandwidth, ill-conditioned, unsymmetric or decomposed problems, etc.) determine the a-priori type of a solution technique. There are two standard approaches, namely

- direct and
- iterative

solver techniques. Due to the extensive memory and computing time consumption of direct solution techniques (e.g. [PARDISO 2007]) within increasing dimension of the discretized problem in respect to degrees of freedom (as e.g. for the Gaussian elimination or for the Cholesky factorization), the suitability of the type of solver is limited in regards to large-scale linear equation systems and to domain decomposition, although the factorized matrix has explicitly to be stored. Due to this limitation, an iterative solution technique of the *Krylov subspace* type is applied in this work. Several of these iterative solution strategies are available:

- Preconditioned Conjugate Gradient method (PCG)
- Bi-Conjugate Gradients stabilized (BiCGstab)
- Generalized Minimal Residual Method (GMRES)
- Conjugate Gradient Squared (CGS)

PCG [Basermann et al. 1997] is a preconditioned variant of the conjugate gradient method. Here, memory demand and computing time are depending on the type of preconditioning matrix (e.g. the preconditioner). Since there is no ideal preconditioning technique available, a compromise strategy for building the preconditioning matrix has to be found, combining reasonable solver speed-ups with acceptable memory demand and also considering the special characteristics of the final system of equations. Further developments such as the BiCG method or the CGS method enable to handle an unsymmetric linear system of equations, where modified versions of GMRES may be used to solve ill-conditioned linear equation systems.

5.3 Domain decomposition methods

Domain decomposition (DD), first described in [Schwarz 1890], is a numerical method for solving a boundary value problem by decomposing the origin domain in several subdomains. There are two types of DD: the overlapping and the non-overlapping domain decomposition. In the following sections, the non-overlapping DD method will be the basic instrument for the developed solution strategy. Original non-overlapping domain decomposition methods [Farhat et al. 1994] are generally divided into two groups: the primal or the dual methods. The primal method is based on the *Schur complement* method. The dual method is described by the so-called FETI (Finite Element Tearing and Interconnecting) method, which originally derived from the Dirichlet-Neumann and Neumann-Neumann substructuring techniques.

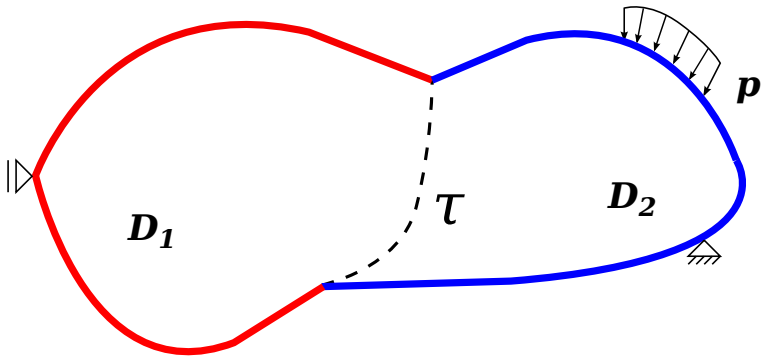


Figure 5.1: Substructuring based on displacement based finite element method and non-overlapping domain decomposition.

All these techniques have the following in common: The DD notation of the assembled submatrices (considering interior and boundary related matrix components according to fig. 5.1) is given with

$$\boldsymbol{K}^{(j)} = \begin{pmatrix} \boldsymbol{K}_{ii}^{(j)} & \boldsymbol{K}_{bi}^{(j)^T} \\ \\ \boldsymbol{K}_{bi}^{(j)} & \boldsymbol{K}_{bb}^{(j)} \end{pmatrix} \qquad \text{with} \quad j = 1, 2 \tag{5.2}$$

splitting the stiffness parts in interior $_i$ and boundary $_b$ related d.o.f.s. By this, in the following sections the iterative computation of the global vector of unknowns will be described and an introduction to the Schur complement method will be given next.

5.3.1 (Direct) Schur complement method

The Schur complement operator [Schwarz et al. 2004] is basically a static condensation of the interior nodal d.o.f.s. to the boundary related d.o.f.s if a domain is decomposed in a fixed number of subdomains each with a corresponding Schur complement matrix.

The assembly of such Schur complements considering all subdomains results in a reduced, reordered and dense Schur complement system of equations. By this, eq. (5.1) can be divided in the following components, representing two different adjacent subdomains, which are denoted by $^{(1)}$ and by $^{(2)}$ and connected by a common boundary with assembled stiffness parts (denoted by $\widetilde{\ }$)

$$\begin{pmatrix} \boldsymbol{K}_{ii}^{(1)} & \boldsymbol{0} & \boldsymbol{K}_{ib}^{(1)} \\ \boldsymbol{0} & \boldsymbol{K}_{ii}^{(2)} & \boldsymbol{K}_{ib}^{(2)} \\ \boldsymbol{K}_{ib}^{(1)^T} & \boldsymbol{K}_{ib}^{(2)^T} & \widetilde{\boldsymbol{K}}_{bb} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_i^{(1)} \\ \boldsymbol{u}_i^{(2)} \\ \boldsymbol{u}_b \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_i^{(1)} \\ \boldsymbol{f}_i^{(2)} \\ \widetilde{\boldsymbol{f}}_b \end{pmatrix} \tag{5.3}$$

After the decomposition, each domain has interior $_i$ and coupling or boundary $_b$ nodes, which are connecting the adjacent domains with the corresponding nodal d.o.f.s, which are included in the vectors $\boldsymbol{u}_i$ and $\boldsymbol{u}_b$. The notation for the system of equations for one domain j (no assembly regarding the boundary of adjacent subdomains) results in

$$\begin{pmatrix} \boldsymbol{K}_{ii} & \boldsymbol{K}_{ib} \\ \boldsymbol{K}_{ib}^T & \boldsymbol{K}_{bb} \end{pmatrix}^{(j)} \begin{pmatrix} \boldsymbol{u}_i \\ \boldsymbol{u}_b \end{pmatrix}^{(j)} = \begin{pmatrix} \boldsymbol{f}_i \\ \boldsymbol{f}_b \end{pmatrix}^{(j)} \tag{5.4}$$

Considering the second row of eq. (5.4) and the matrix $\boldsymbol{K}_{ii}$ being regular, the vector $\boldsymbol{u}_i$ of the first row can be expressed by

$$\boldsymbol{u}_i^{(j)} = \boldsymbol{K}_{ii}^{-1(j)} (\boldsymbol{f}_i^{(j)} - \boldsymbol{K}_{ib}^{(j)} \boldsymbol{u}_b) \tag{5.5}$$

The substitution of eq. (5.4) into eq. (5.5) yields to

$$\left(\boldsymbol{K}_{bb} - \boldsymbol{K}_{ib}^T \boldsymbol{K}_{ii}^{-1} \boldsymbol{K}_{ib} \right)^{(j)} \boldsymbol{u}_b = \left(\boldsymbol{f}_b - \boldsymbol{K}_{ib}^T \boldsymbol{K}_{ii}^{-1} \boldsymbol{f}_i \right)^{(j)} \tag{5.6}$$

The so-called Schur complement $\boldsymbol{S}$ of the matrix $\boldsymbol{K}_{ii}$, which is condensed to the boundary d.o.f.s. indexed by $_b$ then results in

$$\boldsymbol{S}^{(j)} = \left(\boldsymbol{K}_{bb} - \boldsymbol{K}_{ib}^T \boldsymbol{K}_{ii}^{-1} \boldsymbol{K}_{ib} \right)^{(j)} \tag{5.7}$$

Hence, the decomposition in n subdomains results in the global Schur complement system

$$\begin{aligned} \left(\widetilde{\boldsymbol{K}}_{bb} - \sum_{j=1}^{n} \boldsymbol{K}_{ib}^{(j)^T} \boldsymbol{K}_{ii}^{(j)^{-1}} \boldsymbol{K}_{ib}^{(j)} \right) \boldsymbol{u}_b &= \widetilde{\boldsymbol{f}}_b - \sum_{j=1}^{n} \boldsymbol{K}_{ib}^{(j)^T} \boldsymbol{K}_{ii}^{(j)^{-1}} \boldsymbol{f}_i^{(j)} \\ &= \boldsymbol{f}_{b,mod} \end{aligned} \tag{5.8}$$

with the Schur complement matrix

$$\widetilde{\boldsymbol{S}} = \widetilde{\boldsymbol{K}}_{bb} - \sum_{j=1}^{n} \boldsymbol{K}_{ib}^{(j)^T} \boldsymbol{K}_{ii}^{(j)^{-1}} \boldsymbol{K}_{ib}^{(j)} \tag{5.9}$$

and the assembled matrix with the connecting boundary related d.o.f.s

$$\widetilde{\boldsymbol{K}}_{bb} = \sum_{j=1}^{n} \boldsymbol{K}_{bb}^{(j)} \tag{5.10}$$

and also the connecting boundary related vector of nodal forces as

$$\widetilde{\boldsymbol{f}}_b = \sum_{j=1}^{n} \boldsymbol{f}_b^{(j)} \tag{5.11}$$

the final Schur complement system results in

$$\widetilde{\boldsymbol{S}}\boldsymbol{u}_b = \boldsymbol{f}_{b,mod} \tag{5.12}$$

Since the global Schur matrix $\widetilde{\boldsymbol{S}}$ is a dense matrix with non-zero entries for nearly all coefficients, it is numerically expensive to build, to store and to factorize it with direct solvers. Therefore, it is recommended to solve the global decomposed system of equations iteratively without the necessity to compute such Schur complements $\boldsymbol{S}^{(j)}$ explicitly. Generally, the iterative procedure of solving the decomposed global system of equations is based on the repeated computation of the matrix-vector product of eq. (5.4) and on the assembly of all nodal load vectors $\boldsymbol{f}_b^{(j)}$ per subdomain j (during each iteration step of the PCG method). This yields to the global assembled nodal load vector $\widetilde{\boldsymbol{f}}_b$ in respect to all boundary degrees of freedom

$$\widetilde{\boldsymbol{f}}_b = \sum_{j=1}^{n} \boldsymbol{f}_b^{(j)} = \sum_{j=1}^{n} \boldsymbol{K}_{ib}^{T(j)} \boldsymbol{u}_i^{(j)} + \widetilde{\boldsymbol{K}}_{bb}\boldsymbol{u}_b \tag{5.13}$$

The Schur complement system, explicitly extracted or not, requires the partial factorization of $\boldsymbol{K}_{ii}^{(j)}$ with direct solvers, which has then explicitly to be stored, which is a numerical memory-expensive task for a large-scale linear system of equations. Consequently, the iterative computation seems more memory-efficient and, therefore, the D-N and N-N substructuring techniques are reviewed as basic relaxation schemes for the solution of the boundary-related d.o.f.s.

5.3.2 Dirichlet-Neumann and Neumann-Neumann method

Non-overlapping iterative domain decomposition is based on the Dirichlet-Neumann (D-N) and the Neumann-Neumann (N-N) method [Klawonn et al. 2001]. These iterative techniques differ from the direct Schur complement method where the Schur matrix is explicitly extracted, following the previous section. Therefore, the Schur complement boundary problem is iteratively computed based on the originally assembled submatrices considering a relaxation scheme for the connecting d.o.f.s of the different subdomains. For the first type (D-N), where two subdomains are taken into account, the Dirichlet problem in Ω_1 can be expressed by

$$\boldsymbol{K}_{ii}^{\Omega_1} \cdot \boldsymbol{u}_i^{\Omega_1} = \boldsymbol{f}_{i,ext}^{\Omega_1} - \boldsymbol{K}_{ib}^{\Omega_1} \cdot \boldsymbol{u}_b \tag{5.14}$$

The coupling forces in Ω_1 can be computed with

$$\boldsymbol{f}_b = \boldsymbol{K}_{bi}^{\Omega_1} \cdot \boldsymbol{u}_i^{\Omega_1} + \boldsymbol{K}_{bb}^{\Omega_1} \cdot \boldsymbol{u}_b \tag{5.15}$$

After, the Neumann problem in Ω_2 may be solved as

$$\begin{pmatrix} \boldsymbol{K}_{bb}^{\Omega_2} & \boldsymbol{K}_{bi}^{\Omega_2} \\ \\ \boldsymbol{K}_{ib}^{\Omega_2} & \boldsymbol{K}_{ii}^{\Omega_2} \end{pmatrix} \begin{pmatrix} \hat{\boldsymbol{u}}_b \\ \\ \boldsymbol{u}_i^{\Omega_2} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_{b,ext} - \boldsymbol{f}_b \\ \\ \boldsymbol{f}_{i,ext}^{\Omega_2} \end{pmatrix} \tag{5.16}$$

The relaxation in respect to the interface related boundary d.o.f.s (considering the relaxation parameter ω_i) for the next iteration step results in

$$\boldsymbol{u}_b^{(i+1)} = \omega_i \cdot \boldsymbol{u}_b^{(i)} + (1 - \omega_i) \cdot \hat{\boldsymbol{u}}_b^{(i+1)} \tag{5.17}$$

Again, the interior d.o.f.s of the subdomain are denoted by index $_i$, and the boundary related d.o.f.s are denoted by index $_b$, respectively. The external forces (subject to boundary related d.o.f.s) are described by $\boldsymbol{f}_{b,ext}$. The algorithm for the solution of the Dirichlet-Neumann problem has been summarized in table 5.1.

1.	$\boldsymbol{K}_{ii}^{\Omega_1} \cdot \boldsymbol{u}_i^{\Omega_1} = \boldsymbol{f}_{i,ext}^{\Omega_1} - \boldsymbol{K}_{ib}^{\Omega_1} \cdot \boldsymbol{u}_b$	Dirichlet problem in Ω_1
2.	$\boldsymbol{f}_b = \boldsymbol{K}_{bi}^{\Omega_1} \cdot \boldsymbol{u}_i^{\Omega_1} + \boldsymbol{K}_{bb}^{\Omega_1} \cdot \boldsymbol{u}_b$	Coupling forces in Ω_1
3.	$\begin{pmatrix} \boldsymbol{K}_{bb}^{\Omega_2} & \boldsymbol{K}_{bi}^{\Omega_2} \\ \\ \boldsymbol{K}_{ib}^{\Omega_2} & \boldsymbol{K}_{ii}^{\Omega_2} \end{pmatrix} \begin{pmatrix} \hat{\boldsymbol{u}}_b \\ \\ \boldsymbol{u}_i^{\Omega_2} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_{b,ext} - \boldsymbol{f}_b \\ \\ \boldsymbol{f}_{i,ext}^{\Omega_2} \end{pmatrix}$	Solve Neumann problem in Ω_2
4.	$\boldsymbol{u}_b^{(i+1)} = \omega_i \cdot \boldsymbol{u}_b^{(i)} + (1 - \omega_i) \cdot \hat{\boldsymbol{u}}_b^{(i+1)}$	Relaxation
5.	If $\quad ||\boldsymbol{\epsilon}_i|| < TOL$	Break
6.	Else	Goto 1.

Table 5.1: Algorithm for the boundary-related solution applying the Dirichlet-Neumann substructuring technique.

5.3.3 FETI-DP method

Iterative domain decomposition methods for non-overlapped partitionings, such as e.g. the FETI-DP method (Finite Element Tearing and Interconnecting, dual-primal, [Klawonn et al. 2007]), are more efficient than the classical Schur complement method in respect to memory demand. For any dual-primal FETI decomposition the d.o.f.s of the resulting domain boundaries are split in primal and dual variables and are respectively indicated by the indices $_\Pi$ and $_\Delta$. All other domain interior d.o.f.s are marked as interior variables. Summarizing the dual and interior variables (denoted by index $_B$) the unknown nodal vector has three components

- the displacement vector with interior and dual variables $\boldsymbol{u}_B$,
- the displacement vector with primal variables $\widetilde{\boldsymbol{u}}_\Pi$ and
- the vector of Lagrangian multipliers $\boldsymbol{\lambda}$

The fundamental equation of the FETI-DP discretization is given as

$$\begin{pmatrix} \boldsymbol{K}_{BB} & \widetilde{\boldsymbol{K}}_{\Pi B}^{T} & \boldsymbol{B}^{T} \\ \widetilde{\boldsymbol{K}}_{\Pi B} & \widetilde{\boldsymbol{K}}_{\Pi\Pi} & \boldsymbol{0} \\ \boldsymbol{B} & \boldsymbol{0} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_B \\ \widetilde{\boldsymbol{u}}_{\Pi} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_B \\ \widetilde{\boldsymbol{f}}_{\Pi} \\ \boldsymbol{0} \end{pmatrix} \tag{5.18}$$

being rearranged in regard to its variables $\boldsymbol{u}_B$, $\widetilde{\boldsymbol{u}}_{\Pi}$ and $\boldsymbol{\lambda}$. Consequently, the global FETI-DP matrix in eq. (5.18) consists of the following components

- $\boldsymbol{K}_{BB}$ as block-diagonal matrix resulting from global interior and dual d.o.f.s
- $\widetilde{\boldsymbol{K}}_{\Pi\Pi}$ as assembled block-diagonal matrix resulting from primal d.o.f.s and
- $\boldsymbol{B}$ as jump operator connecting dual d.o.f.s of different domains

Finally, the classical FE equation system can be reformulated using the above notation. The local stiffness matrix is given with

$$\boldsymbol{K}^{(i)} = \begin{pmatrix} \boldsymbol{K}_{II}^{(i)} & \boldsymbol{K}_{\Delta I}^{(i)T} & \boldsymbol{K}_{\Pi I}^{(i)T} \\ \boldsymbol{K}_{\Delta I}^{(i)} & \boldsymbol{K}_{\Delta\Delta}^{(i)} & \boldsymbol{K}_{\Pi\Delta}^{(i)T} \\ \boldsymbol{K}_{\Pi I}^{(i)} & \boldsymbol{K}_{\Pi\Delta}^{(i)} & \boldsymbol{K}_{\Pi\Pi}^{(i)} \end{pmatrix} \tag{5.19}$$

and the global vector of unknown variables as well as the global load vector result in

$$\boldsymbol{u}^{(i)} = \begin{pmatrix} \boldsymbol{u}_I^{(i)} \\ \boldsymbol{u}_{\Delta}^{(i)} \\ \boldsymbol{u}_{\Pi}^{(i)} \end{pmatrix}; \qquad \boldsymbol{f}^{(i)} = \begin{pmatrix} \boldsymbol{f}_I^{(i)} \\ \boldsymbol{f}_{\Delta}^{(i)} \\ \boldsymbol{f}_{\Pi}^{(i)} \end{pmatrix} \tag{5.20}$$

with $_I$ as interior, $_\Delta$ as dual and $_\Pi$ as primal indices. The condition

$$\boldsymbol{B}\boldsymbol{u}_B = \boldsymbol{0} \quad \text{with} \quad b_{ij} \in \{0;1\} \tag{5.21}$$

has to be fulfilled in respect to the interior and dual d.o.f.s. The interior and dual variables can be summarized and denoted by index $_B$. Hence, the vectors are

$$\boldsymbol{u}_B^{(i)} = \begin{bmatrix} \boldsymbol{u}_I^{(i)} & \boldsymbol{u}_{\Delta}^{(i)} \end{bmatrix}^T \tag{5.22}$$

and

$$\boldsymbol{f}_B^{(i)} = \begin{bmatrix} \boldsymbol{f}_I^{(i)} & \boldsymbol{f}_{\Delta}^{(i)} \end{bmatrix}^T \tag{5.23}$$

For the stiffness matrices (local, subdomain based and assembled) without primal d.o.f.s follows

$$\boldsymbol{K}_{BB}^{(i)} = \begin{pmatrix} \boldsymbol{K}_{II}^{(i)} & \boldsymbol{K}_{\Delta I}^{(i)T} \\ \\ \boldsymbol{K}_{\Delta I}^{(i)} & \boldsymbol{K}_{\Delta\Delta}^{(i)} \end{pmatrix} \tag{5.24}$$

and the assembly of all local matrices $\boldsymbol{K}_{BB}^{(i)}$ yields to the global block diagonal matrix

$$\boldsymbol{K}_{BB} = \mathrm{diag}_{i=1}^{N}[\boldsymbol{K}_{BB}^{(i)}] \tag{5.25}$$

Moreover, the elimination of the interior, dual and primal d.o.f.s leads to

$$\begin{pmatrix} \boldsymbol{K}_{BB} & \widetilde{\boldsymbol{K}}_{\Pi B}^{T} & \boldsymbol{B}^{T} \\ \\ \boldsymbol{0} & \widetilde{\boldsymbol{S}}_{\Pi\Pi} & -\widetilde{\boldsymbol{S}}_{\alpha\Pi}^{T} \\ \\ \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{F} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_B \\ \\ \widetilde{\boldsymbol{u}}_{\Pi} \\ \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_B \\ \\ \widetilde{\boldsymbol{f}}_{\Pi} - \widetilde{\boldsymbol{K}}_{\Pi B}\boldsymbol{K}_{BB}^{-1}\boldsymbol{f}_B \\ \\ -\boldsymbol{d} \end{pmatrix} \tag{5.26}$$

with

$$\widetilde{\boldsymbol{S}}_{\Pi\Pi} = \widetilde{\boldsymbol{K}}_{\Pi\Pi} - \widetilde{\boldsymbol{K}}_{\Pi B}\boldsymbol{K}_{BB}^{-1}\widetilde{\boldsymbol{K}}_{\Pi B}^{T} \tag{5.27}$$

as Schur complement operator in respect to $\widetilde{\boldsymbol{K}}_{\Pi\Pi}$ and the notation for $\boldsymbol{F}$ and $\boldsymbol{d}$

$$\boldsymbol{F} = \boldsymbol{B}\boldsymbol{K}_{BB}^{-1}\boldsymbol{B}^{T} + \boldsymbol{B}\boldsymbol{K}_{BB}^{-1}\widetilde{\boldsymbol{K}}_{\Pi B}^{T}\widetilde{\boldsymbol{S}}_{\Pi\Pi}^{-1}\widetilde{\boldsymbol{K}}_{\Pi B}\boldsymbol{K}_{BB}^{-1}\boldsymbol{B}^{T} \tag{5.28}$$

$$\boldsymbol{d} = \boldsymbol{B}\boldsymbol{K}_{BB}^{-1}\boldsymbol{f}_B - \boldsymbol{B}\boldsymbol{K}_{BB}^{-1}\widetilde{\boldsymbol{K}}_{\Pi B}^{T}\widetilde{\boldsymbol{S}}_{\Pi\Pi}^{-1}\left(\widetilde{\boldsymbol{f}}_{\Pi} - \widetilde{\boldsymbol{K}}_{\Pi B}\boldsymbol{K}_{BB}^{-1}\boldsymbol{f}_B\right) \tag{5.29}$$

Hence, the final reduced equation system, which reduces eq. (5.18) to the Lagragian multipliers, is given as

$$\boldsymbol{F}\boldsymbol{\lambda} = \boldsymbol{d} \tag{5.30}$$

In most cases $\boldsymbol{F}$ will not explicitly be generated and a (preconditioned) conjugate gradient method has been used for the iterative computation of the particular equation system with a specified preconditioning technique to approximate $\boldsymbol{K}_{II}^{(i)}$. In the following section it is shown how the Saddle-point equation system obtained from the above described FETI-DP discretization will be solved. For this, a conjugate gradient version of an Uzawa [Wang 2009] iteration scheme is modified to solve the resulting FETI-DP Saddle-point equation (SPE).

5.3.4 Modified FETI-DP Saddle-point problem

By eliminating the primal variables the basic formulation for FETI-DP discretizations of the previous section can be reformulated as Saddle-point equation system (SPE). The

basic notation of the FETI-DP Saddle-point system is given by

$$\begin{pmatrix} \boldsymbol{K}_{BB} & \boldsymbol{B}^T \\ \boldsymbol{B} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_B \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_{B,mod} \\ \boldsymbol{0} \end{pmatrix} \tag{5.31}$$

with

$$\boldsymbol{f}_{B,mod} = \boldsymbol{f}_B - \widetilde{\boldsymbol{K}}_{\Pi B}^T \widetilde{\boldsymbol{u}}_\Pi \tag{5.32}$$

resulting from the first row of eq. (5.18). The second row of the basic equation system (5.18) can be written as

$$\widetilde{\boldsymbol{K}}_{\Pi\Pi}\, \widetilde{\boldsymbol{u}}_\Pi = \widetilde{\boldsymbol{f}}_\Pi - \widetilde{\boldsymbol{K}}_{\Pi B}\, \boldsymbol{u}_B \tag{5.33}$$

Hence, the main feature of Saddle-point equation systems enables a comfortable solution approach, which allows to utilize iterative solver techniques for eq. (5.31) without knowing the exact inverse of $\boldsymbol{K}_{BB}$ and without loss of the optimal scaling properties of the FETI-DP algorithm. One additional advantage occurs at the resulting right hand side where all terms according to Langragian multipliers $\boldsymbol{\lambda}$ are zero: It reduces the original problem to block $\boldsymbol{K}_{BB}$ and enables the iterative computation of $\boldsymbol{u}_B$. Using the first row of eq. (5.31) it follows

$$\boldsymbol{K}_{BB}\, \boldsymbol{u}_B = \boldsymbol{f}_B - \widetilde{\boldsymbol{K}}_{\Pi B}^T\, \widetilde{\boldsymbol{u}}_\Pi - \boldsymbol{B}^T \boldsymbol{\lambda} \tag{5.34}$$

Transferring eq. (5.33) into eq. (5.34) results in

$$\boldsymbol{K}_{BB}\, \boldsymbol{u}_B = \boldsymbol{f}_B - \widetilde{\boldsymbol{K}}_{\Pi B}^T \widetilde{\boldsymbol{K}}_{\Pi\Pi}^{-1} \left(\widetilde{\boldsymbol{f}}_\Pi - \widetilde{\boldsymbol{K}}_{\Pi B} \boldsymbol{u}_B \right) - \boldsymbol{B}^T \boldsymbol{\lambda} \tag{5.35}$$

With the definition of $\widetilde{\boldsymbol{f}}_{B,S}$

$$\widetilde{\boldsymbol{f}}_{B,S} = \widetilde{\boldsymbol{K}}_{\Pi B}^T \widetilde{\boldsymbol{K}}_{\Pi\Pi}^{-1} \widetilde{\boldsymbol{f}}_\Pi \tag{5.36}$$

and introduction of the interior-dual Schur complement operator $\widetilde{\boldsymbol{S}}_{BB}$ as

$$\widetilde{\boldsymbol{S}}_{BB} = \boldsymbol{K}_{BB} - \widetilde{\boldsymbol{K}}_{\Pi B}^T \widetilde{\boldsymbol{K}}_{\Pi\Pi}^{-1} \widetilde{\boldsymbol{K}}_{\Pi B} \tag{5.37}$$

The final notation leads to

$$\boldsymbol{K}_{BB}\, \boldsymbol{u}_B = \boldsymbol{f}_B - \widetilde{\boldsymbol{f}}_{B,S} + \left(\boldsymbol{K}_{BB} - \widetilde{\boldsymbol{S}}_{BB} \right) \boldsymbol{u}_B - \boldsymbol{B}^T \boldsymbol{\lambda} \tag{5.38}$$

After, the reduced and inverted FETI-DP SPE is obtained by

$$\boldsymbol{u}_B = \widetilde{\boldsymbol{S}}_{BB}^{-1} \left(\boldsymbol{f}_B - \widetilde{\boldsymbol{f}}_{B,S} - \boldsymbol{B}^T \boldsymbol{\lambda} \right) = \widetilde{\boldsymbol{S}}_{BB}^{-1} \left(\widetilde{\boldsymbol{g}}_{B,S} - \boldsymbol{B}^T \boldsymbol{\lambda} \right) \tag{5.39}$$

The modified FETI-DP SPE follows with

$$\begin{pmatrix} \widetilde{\boldsymbol{S}}_{BB} & \boldsymbol{B}^T \\ \boldsymbol{B} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_B \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \widetilde{\boldsymbol{g}}_{B,S} \\ \boldsymbol{0} \end{pmatrix} \tag{5.40}$$

Taking the condition $\boldsymbol{B}\boldsymbol{u}_B = \boldsymbol{0}$ and replacing $\boldsymbol{u}_B$ with the first row of eq. (5.40) it results in

$$\boldsymbol{B}\widetilde{\boldsymbol{S}}_{BB}^{-1}\boldsymbol{B}^T\boldsymbol{\lambda} = \boldsymbol{B}\widetilde{\boldsymbol{S}}_{BB}^{-1}\widetilde{\boldsymbol{g}}_{B,S} \tag{5.41}$$

with

$$\widetilde{\boldsymbol{Q}}_{BB} = \boldsymbol{B}\widetilde{\boldsymbol{S}}_{BB}^{-1}\boldsymbol{B}^T \tag{5.42}$$

and

$$\widetilde{\boldsymbol{c}}_{B,S} = \boldsymbol{B}\widetilde{\boldsymbol{S}}_{BB}^{-1}\widetilde{\boldsymbol{g}}_{B,S} \tag{5.43}$$

The gradient of the quadratic function

$$F(\boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{\lambda}^T\widetilde{\boldsymbol{Q}}_{BB}\boldsymbol{\lambda} - \widetilde{\boldsymbol{c}}_{B,S}^T\boldsymbol{\lambda} \tag{5.44}$$

results in

$$\begin{aligned} -\nabla F(\boldsymbol{\lambda}^{(k)}) &= \|\widetilde{\boldsymbol{c}}_{B,S} - \widetilde{\boldsymbol{Q}}_{BB}\boldsymbol{\lambda}\| \\ &= \|\boldsymbol{B}\widetilde{\boldsymbol{S}}_{BB}^{-1}\widetilde{\boldsymbol{g}}_{B,S} - \boldsymbol{B}\widetilde{\boldsymbol{S}}_{BB}^{-1}\boldsymbol{B}^T\boldsymbol{\lambda}^{(k)}\| \\ &= \|\boldsymbol{B}\boldsymbol{u}_B^{(k)}\| \\ &= 0 \end{aligned} \tag{5.45}$$

which corresponds to the basic function of the conjugate gradient method. Back-substitution into eq. (5.45) yields to

$$\begin{aligned} -\nabla F(\boldsymbol{\lambda}^{(k)}) =& \|\boldsymbol{B}\left(\boldsymbol{K}_{BB} - \widetilde{\boldsymbol{K}}_{\Pi B}^T\widetilde{\boldsymbol{K}}_{\Pi\Pi}^{-1}\widetilde{\boldsymbol{K}}_{\Pi B}\right)^{-1}\left(\boldsymbol{f}_B - \widetilde{\boldsymbol{K}}_{\Pi B}^T\widetilde{\boldsymbol{K}}_{\Pi\Pi}^{-1}\widetilde{\boldsymbol{f}}_{\Pi}\right) \\ &- \left(\boldsymbol{B}\left(\boldsymbol{K}_{BB} - \widetilde{\boldsymbol{K}}_{\Pi B}^T\widetilde{\boldsymbol{K}}_{\Pi\Pi}^{-1}\widetilde{\boldsymbol{K}}_{\Pi B}\right)^{-1}\boldsymbol{B}^T\right)\boldsymbol{\lambda}^{(k)}\| \\ =& \|\boldsymbol{B}\boldsymbol{u}_B^{(k)}\| \\ =& 0 \end{aligned} \tag{5.46}$$

The convergence is ensured, if

$$\rho(\boldsymbol{B}\widetilde{\boldsymbol{S}}_{BB}^{-1}\boldsymbol{B}^T) < \frac{2}{\alpha} \tag{5.47}$$

is fulfilled - with α as a fixed step size and ρ as the spectral radius. The next section includes several numerical tests, considering different methods to create a suitable preconditioning matrix, either approximating the inverse of the interior-dual Schur complement operator or using it implicitly as preconditioning matrix solving the Schur problem with the conjugate gradient method iteratively and by this, to have the possibility to evaluate the quality of such preconditioners.

5.4 Numerical testing

5.4.1 Direct versus iterative solving

The first numerical test (results shown in table 5.2) is a simple, regular meshed FE block structure, decomposed in two subdomains (D_1 and D_2, marked with different colors, see figure 5.2). There, the total stiffness matrix for the assembled finite element stiffnesses of both subdomains has been computed. The factorization of the total stiffness matrix and the direct factorization as well as the partial factorization (explicitly extracting the Schur complement matrix) for both subdomains were then performed for five different element sizes (with up to 150,000 nodal d.o.f.s.). The computing times (measured in seconds) are listed in table 5.2. This implementation is based on the multifrontal MPI-solver package MUMPS [Amestoy et al. 2007], version 4.7.3. For the numerical evaluation the following has been considered

- standard Linux workstation with 8GB RAM,
- basic linear algebra subroutines (BLAS) and
- two different CPU Architectures: iCore2 Duo (2.4GHz) and AMD Opteron (1.9GHz).

As highlighted in columns 6 and 8, the absolute times for the partial factorization and the computation of the Schur complement matrix are always higher than the direct factorization of the submatrix. If the factorization time of the total stiffness matrix is included the efficiency is decreasing. A further disadvantage occurs since the computing time for the direct factorization is exponentially increasing with each mesh refinement due to the moderate increase of the number of nodal d.o.f.s, a more crucial task in the partial factorization. Based on this strategy, as expected, the distributed computation of large-scale linear equation systems is numerically expensive in respect to the computing time and memory demand.
Hence, a second numerical test was performed for the same example of figure 5.2, taking an iterative substructuring method based on Dirichlet-Neumann (as described in section 4.3.1) into account. The results for the same mesh refinements are listed in table 5.3, comparing computational times for the sequential and parallel computation of the linear

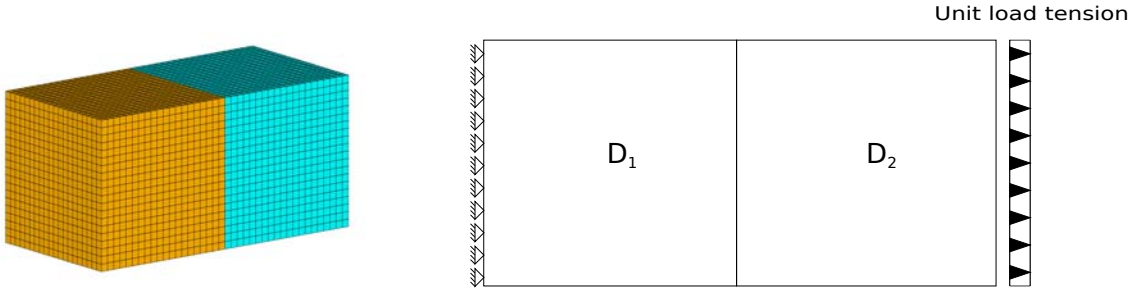


Figure 5.2: Simple block structure decomposed in two domains with a maximum of 150,000 d.o.f.s.

mesh	d.o.f.s total	d.o.f.s Ω_1	d.o.f.s Ω_2	direct total	direct Ω_1	direct Ω_2	Schur Ω_1	Schur Ω_2
1.0	50288	23876	25076	8.41	3.19	3.44	4.49	4.92
1.1	71409	26906	28206	9.15	3.90	4.20	5.65	6.12
1.2	97682	33712	35224	12.07	5.80	5.98	8.20	9.07
1.3	118342	55540	57652	28.17	13.56	14.79	20.01	21.42
1.4	154468	72318	74838	45.17	22.17	22.94	33.34	35.45

Table 5.2: Direct factorization versus explicit Schur complement extraction (as partial factorization) with an increasing mesh refinement.

equation system, considering the direct factorization (using the multifrontal solver) and the iterative Dirichlet-Neumann (D-N) substructuring technique. With an accuracy of 10^{-6}, the iterative D-N method was more efficient for all investigated meshes in the sequential and also in the parallel case. All solver techniques react similarly in respect to the numerical effort, if the mesh size is changed as illustrated in the diagrams of figure 5.3.

mesh	d.o.f.s	direct iCore2/seq.	direct Opt./seq.	direct Opt./np=4	D-N Opt./seq	D-N Opt./np=4
1.0	50288	12.56	12.02	5.27	8.27 (3)	3.46 (2)
1.1	71406	23.08	22.13	9.71	14.96 (4)	5.32 (2)
1.2	97682	38.88	37.07	16.17	24.00 (3)	11.01 (2)
1.3	118342	54.45	51.39	20.54	34.75 (3)	12.90 (3)
1.4	154468	89.48	82.98	31.74	54.79 (3)	20.04 (3)

Table 5.3: Comparison of direct factorization based computation (direct) with iterative Dirichlet-Neumann (D-N) method, sequential times (seq.) for Intel iCore2 and AMD Opteron (Opt./seq.), parallel times for the Opteron architecture using four MPI processes (Opt./np=4); accuracy: 10^{-6}, time in seconds and number of iterations given in brackets.

5.4.2 Modified FETI-DP SPE

Algorithmic implementation

The following sequential algorithms are describing the iterative solution process of the FETI-DP Saddle-point equation. Here, the Uzawa iteration scheme for Saddle-point problems was adapted and modified. The first version, as shown below in table 5.4, is the standard Uzawa algorithm solving the FETI-DP SPE. After obtaining the solution of the dual-interior d.o.f.s and the vector of the Langragian multipliers, an updated step at the end of the iteration then computes the vector of primal nodal unknowns. A second CG-based Uzawa version additionally includes the preconditioned CG method for the iterative computation of the (interior-dual) Schur complement problem. With the introduction of three additional vectors, the conjugate gradient version of the standard

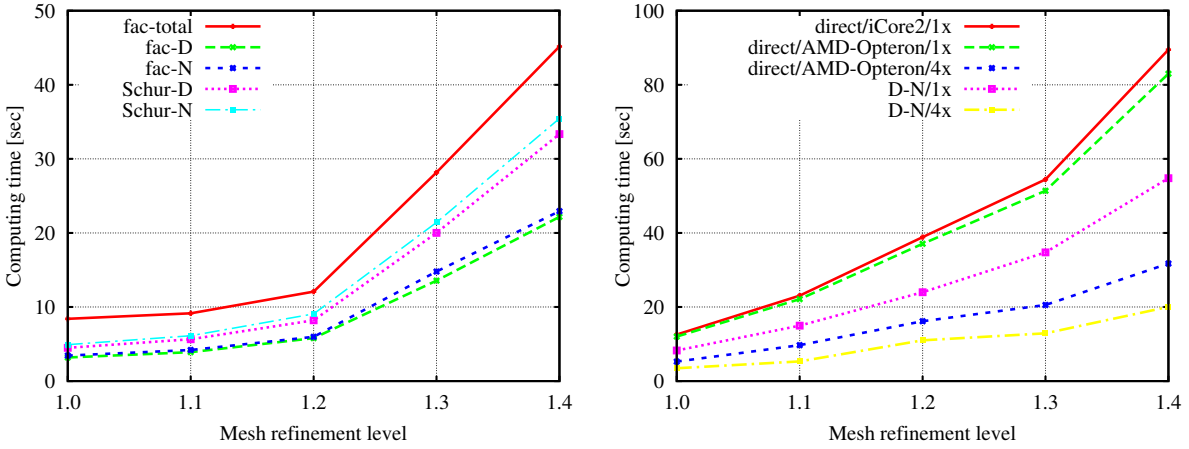


Figure 5.3: Mesh refinement and computing time of partial factorization for a dual domain split (left) and solver times for different meshes according to direct factorization (sequential on Intel iCore2: direct/seq/Core2; sequential on AMD Opteron: direct/seq/Opt; parallel on AMD Opteron: direct/par/Opt) and following the Dirichlet-Neumann iteration (sequential D-N on AMD Opteron: D-N/seq; parallel D-N on AMD Opteron with four MPI processes: D-N/par/np=4).

Uzawa algorithm is obtained. Furthermore, a preconditioned conjugate gradient method is implicitly introduced for iteratively solving the Schur complement problem and, by this, avoids the direct factorization of $\widetilde{\boldsymbol{S}}_{BB}$. The algorithm is shown in table 5.5.

	Require: $\widetilde{\boldsymbol{S}}_{BB}$ is invertible.
1.	Choose $\boldsymbol{\lambda}^{(0)}$ and $\boldsymbol{u}_B^{(1)} = \widetilde{\boldsymbol{S}}_{BB}^{-1}\left(\boldsymbol{f}_B - \widetilde{\boldsymbol{f}}_{B,S} - \boldsymbol{B}^T\boldsymbol{\lambda}^{(0)}\right)$ Factorize $\widetilde{\boldsymbol{K}}_{\Pi\Pi}$ with $\widetilde{\boldsymbol{f}}_{B,S} = \widetilde{\boldsymbol{K}}_{\Pi B}^T\widetilde{\boldsymbol{K}}_{\Pi\Pi}^{-1}\widetilde{\boldsymbol{f}}_{\Pi}$ and $\widetilde{\boldsymbol{S}}_{BB}^{-1} = \left(\boldsymbol{K}_{BB} - \widetilde{\boldsymbol{K}}_{\Pi B}^T\widetilde{\boldsymbol{K}}_{\Pi\Pi}^{-1}\widetilde{\boldsymbol{K}}_{\Pi B}\right)^{-1}$
2.	**for** $k = 1, 2, \ldots$ **do**
3.	$\boldsymbol{q}^{(k)} = -\boldsymbol{B}\boldsymbol{u}_B^{(k)}$
4.	$\boldsymbol{p}^{(k)} = \boldsymbol{B}^T\boldsymbol{q}^{(k)}$
5.	$\boldsymbol{h}^{(k)} = \widetilde{\boldsymbol{S}}_{BB}^{-1}\boldsymbol{p}^{(k)}$
6.	$\alpha^{(k)} = \frac{\boldsymbol{q}^{(k)^T}\boldsymbol{q}^{(k)}}{\boldsymbol{p}^{(k)^T}\boldsymbol{h}^{(k)}}$
7.	$\boldsymbol{\lambda}^{(k)} = \boldsymbol{\lambda}^{(k-1)} - \alpha^{(k)}\boldsymbol{q}^{(k)}$
8.	$\boldsymbol{u}_B^{(k+1)} = \boldsymbol{u}_B^{(k)} + \alpha^{(k)}\boldsymbol{h}^{(k)}$
	end for
9.	Update $\widetilde{\boldsymbol{u}}_{\Pi}^{(k+1)} = \widetilde{\boldsymbol{K}}_{\Pi\Pi}^{-1}\left(\widetilde{\boldsymbol{f}}_{\Pi} - \widetilde{\boldsymbol{K}}_{\Pi B}\boldsymbol{u}_B^{(k+1)}\right)$

Table 5.4: Uzawa algorithm with exact iteration steps for FETI-DP SPE.

Numerical convergence tests

To illustrate the (FETI-DP) discretization, an example in 2D with two elements (each for one domain) is shown in figure 5.4, where the primal d.o.f.s are marked with a black dot at

	Require: $\widetilde{\boldsymbol{S}}_{BB}$ is invertable.
1.	Choose $\boldsymbol{\lambda}^{(0)}$ and $\boldsymbol{u}_B^{(1)} = \widetilde{\boldsymbol{S}}_{BB}^{-1}\left(\boldsymbol{f}_B - \widetilde{\boldsymbol{f}}_{B,S} - \boldsymbol{B}^T\boldsymbol{\lambda}^{(0)}\right)$
	Factorize $\widetilde{\boldsymbol{K}}_{\Pi\Pi}$ with $\widetilde{\boldsymbol{f}}_{B,S} = \widetilde{\boldsymbol{K}}_{\Pi B}^T\widetilde{\boldsymbol{K}}_{\Pi\Pi}^{-1}\widetilde{\boldsymbol{f}}_{\Pi}$
	and $\widetilde{\boldsymbol{S}}_{BB} = \left(\boldsymbol{K}_{BB} - \widetilde{\boldsymbol{K}}_{\Pi B}^T\widetilde{\boldsymbol{K}}_{\Pi\Pi}^{-1}\widetilde{\boldsymbol{K}}_{\Pi B}\right)$ and build preconditioner $\boldsymbol{M}$
2.	Set $\boldsymbol{d}^{(1)} = -\boldsymbol{q}^{(1)} = \boldsymbol{B}\boldsymbol{u}_B^{(1)}$
3.	**for** $k = 1, 2, ...$**do**
4.	$\boldsymbol{p}^{(k)} = \boldsymbol{B}^T\boldsymbol{d}^{(k)}$
5.	$\boldsymbol{h}^{(k)} = \widetilde{\boldsymbol{S}}_{BB}^{-1}\boldsymbol{p}^{(k)}$ as
5.1	Choose $\boldsymbol{h}_0^{(k)}$ and $\widetilde{\boldsymbol{r}}_0 = \boldsymbol{p}^{(k)} - \widetilde{\boldsymbol{S}}_{BB}\boldsymbol{h}_0^{(k)}$
5.2	$\widetilde{\boldsymbol{h}}_0 = \boldsymbol{M}\widetilde{\boldsymbol{r}}_0$
5.3	$\widetilde{\boldsymbol{d}}_0 = \widetilde{\boldsymbol{h}}_0$
5.4	**for** $j = 1, 2, ...$ **do**
5.5	$\widetilde{\alpha}_j = \frac{\widetilde{\boldsymbol{r}}_j^T\widetilde{\boldsymbol{h}}_j}{\widetilde{\boldsymbol{d}}_j^T\widetilde{\boldsymbol{S}}_{BB}\widetilde{\boldsymbol{d}}_j}$
5.6	$\boldsymbol{h}_{j+1}^{(k)} = \boldsymbol{h}_j^{(k)} + \widetilde{\alpha}_j\widetilde{\boldsymbol{d}}_j$
5.7	$\widetilde{\boldsymbol{r}}_{j+1} = \widetilde{\boldsymbol{r}}_j - \widetilde{\alpha}_j\widetilde{\boldsymbol{S}}_{BB}\widetilde{\boldsymbol{d}}_j$
5.8	$\widetilde{\boldsymbol{h}}_{j+1} = \boldsymbol{M}\widetilde{\boldsymbol{r}}_{j+1}$
5.9	$\widetilde{\beta}_j = \frac{\widetilde{\boldsymbol{r}}_{j+1}^T\widetilde{\boldsymbol{h}}_{j+1}}{\widetilde{\boldsymbol{r}}_j^T\widetilde{\boldsymbol{h}}_j}$
5.10	$\widetilde{\boldsymbol{d}}_{j+1} = \widetilde{\boldsymbol{h}}_{j+1} + \beta_j\widetilde{\boldsymbol{d}}_j$
5.11	**end for** (implicit PCG)
6.	$\alpha^{(k)} = \frac{\boldsymbol{q}^{(k)^T}\boldsymbol{q}^{(k)}}{\boldsymbol{p}^{(k)^T}\boldsymbol{h}^{(k)}}$
7.	$\boldsymbol{\lambda}^{(k)} = \boldsymbol{\lambda}^{(k-1)} + \alpha^{(k)}\boldsymbol{d}^{(k)}$
8.	$\boldsymbol{u}_B^{(k+1)} = \boldsymbol{u}_B^{(k)} - \alpha^{(k)}\boldsymbol{h}^{(k)}$
9.	$\boldsymbol{q}^{(k+1)} = -\boldsymbol{B}\boldsymbol{u}_B^{(k+1)}$
10.	$\beta^{(k)} = \frac{\boldsymbol{q}^{(k+1)^T}\boldsymbol{q}^{(k+1)}}{\boldsymbol{q}^{(k)^T}\boldsymbol{q}^{(k)}}$
11.	$\boldsymbol{d}^{(k+1)} = -\boldsymbol{q}^{(k+1)} + \beta^{(k)}\boldsymbol{d}^{(k)}$
12.	**end for** (Uzawa-CG)
13.	Update $\widetilde{\boldsymbol{u}}_{\Pi}^{(k+1)} = \widetilde{\boldsymbol{K}}_{\Pi\Pi}^{-1}\left(\widetilde{\boldsymbol{f}}_{\Pi} - \widetilde{\boldsymbol{K}}_{\Pi B}\boldsymbol{u}_B^{(k+1)}\right)$

Table 5.5: CG version of Uzawa algorithm for FETI-DP SPE.

the corresponding node. The duplicated dual d.o.f.s related to the node are marked with a small rectangle. The numerical test is based on an example discretized by six elements with two elements per domain and is partitioned in a way that all domains are sequentially coupled and each boundary is connecting two domains only. Again, the primal nodes are denoted with black dots and the dual nodes with small rectangles, respectively. The partitioning is illustrated in figure 5.5 (upper left) the FETI-DP discretization (bottom left) and the duplication of the dual nodes are shown (bottom, right). The computation considers linear-elastic material behavior as well as a plane stress state. In this example,

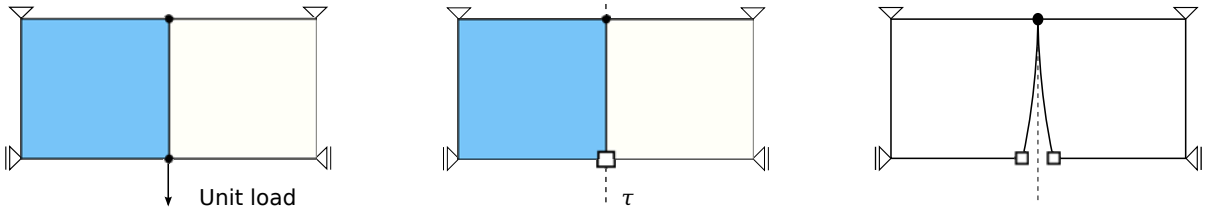


Figure 5.4: Example of one element per domain (left), the resulting FETI-DP discretization (middle) and the duplication of the dual node (right).

the algorithms of the tables 5.4 and 5.5 were used for the computation of the unknowns of vector $\boldsymbol{u}_B$ and $\boldsymbol{\lambda}$ of the FETI-DP Saddle-point problem and compared to the D-N relaxation procedure.
The numerical results can be seen in table 5.6 below, stating their necessary iteration steps ($TOL \leq 10^{-6}$) and their accuracy. The results show the improved convergence behavior when using a modified CG version of the Uzawa algorithm instead of the standard Uzawa iteration procedure for the computation of the FETI-DP Saddle-point equation system. For large discretized problems the Uzawa-CG with an implicit preconditioned conjugate gradient method (iPCG) iteratively solving the Schur complement problem will lead to an additional decrease of time. The motivation of the last numerical experiment was to get an inside of the convergence behavior, if an explicit preconditioning matrix $\boldsymbol{M}^{-1}$ for the

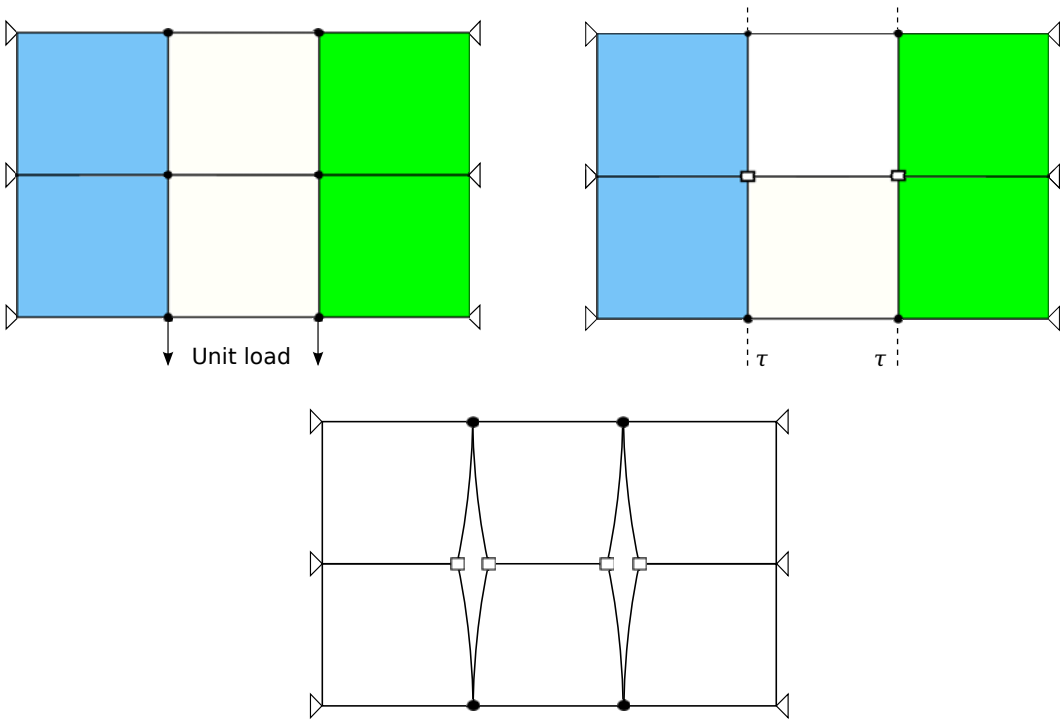


Figure 5.5: Example of FETI-DP discretization for three domains with two elements per domain: Partitioning in three domains (upper left); FETI-DP discretization (upper right); duplication of the dual nodes (bottom).

preconditioning of $\widetilde{\boldsymbol{S}}_{BB}$ is applied. The following preconditioning matrices were adapted for $\boldsymbol{M}^{-1}$ according to table 5.7 (and results are shown in table 5.8) stating the number of iterations, the obtained accuracy and the relative solution error.

5.4.3 Concluding remarks

The numerical tests described in the previous sections have proven, that the FETI-DP method is suitable, if orthogonal regular domain boundaries exist. This differs for the discretization of multiphase materials, if an aligned meshing strategy for the boundaries of the material phases is used and a mixed mesh is proposed. Thereby, the lack of a setup for defining the primal nodes (the so-called fixing nodes) leads to an unstable iterative computation of the FETI-DP equation system, in most cases without numerical convergence. Nevertheless, if such a fixing node strategy exists for large-scale material-heterogeneous specimens, the FETI-DP problem can be modified to a Saddle-point problem. It is shown that the restated notation can numerically be solved by modifying the Uzawa algorithm in respect to different approximation techniques for a suitable preconditioning matrix. Therefore, in this work, the memory saving FETI-DP method was replaced by the iterative Schur complement method and has been combined with the preconditioned conjugate gradient method. The occuring higher memory demand is compensated by avoiding the explicit Schur complement extraction and by using a memory-efficient preconditioning technique, which will be proposed in the following sections. Consequently, the implemented iterative solver is based on a parallelized version of the preconditioned conjugate gradient method (PPCG, [Basermann et al. 1997]) using the message-passing interface standard 2.0 (MPI, [Gropp et al. 1999]). This is described in the next section. Results for the efficiency of this approach are given in chapter 6.

5.5 (Hybrid) parallelized preconditioned conjugate gradients

The implementation of the MPI-parallelized sparse iterative solver is based on the iterative conjugate gradient method (CG, [Kelley 1995]) and the preconditioned CG version

algorithm	**number** iterations	**accuracy**	**residual norm** $\|\|\widetilde{\boldsymbol{r}}_{k+1}\|\|_{F,\text{iPCG}}$	**note**
D-N	>10	1.0e-06	–	-
Uzawa	9	3.4e-07	–	-
Uzawa-CG	4	exact	–	-
Uzawa-CG iPCG (1)	4	1.0e-06	1.6e-04	with implicit PCG
Uzawa-CG iPCG (2)	4	4.6e-10	2.5e-07	with implicit PCG
Uzawa-CG iPCG (3)	4	exact	6.4e-19	with implicit PCG

Table 5.6: Iteration steps and accuracy for a varity of different iteration techniques.

$\text{diag}\left[\boldsymbol{K}_{BB}\right]^{-1}$	the inverted main diagonal of $\boldsymbol{K}_{BB}$
$\text{diag}\left[\widetilde{\boldsymbol{S}}_{BB}\right]^{-1}$	the inverted main diagonal of $\widetilde{\boldsymbol{S}}_{BB}$
$\text{diag}_{j=1}^{N}\left[\widetilde{\boldsymbol{S}}_{BB}^{(j)}\right]^{-1}$	the inverted blockdiagonals of $\widetilde{\boldsymbol{S}}_{BB}^{(j)}$
$\widetilde{\boldsymbol{S}}_{BB}^{-1}$	the explicit inversion of $\widetilde{\boldsymbol{S}}_{BB}$

Table 5.7: Overview of different preconditioning matrices.

preconditioner $\boldsymbol{M}^{-1}$	**iterations**	**accuracy**	**relative error norm** $\lim \frac{\|\boldsymbol{E}-\boldsymbol{M}^{-1}\widetilde{\boldsymbol{S}}_{BB}\|_F}{\|\widetilde{\boldsymbol{S}}_{BB}\|_F}$	**algorithm**
$\text{diag}\left[\boldsymbol{K}_{BB}\right]^{-1}$	3	exact	0.038	Uzawa-CG
$\text{diag}\left[\widetilde{\boldsymbol{S}}_{BB}\right]^{-1}$	3	exact	0.022	Uzawa-CG
$\text{diag}_{j=1}^{N}\left[\widetilde{\boldsymbol{S}}_{BB}^{(j)}\right]^{-1}$	4	exact	0.013	Uzawa-CG
$\widetilde{\boldsymbol{S}}_{BB}^{-1}$	4	exact	0	Uzawa-CG

Table 5.8: Relative error norm for different approximated preconditioning matrices replacing the interior-dual Schur complement operator.

(PCG, [Loghin et al. 2003]). Steps 2 to 14 of table 5.9 are to be computed per iteration step and per domain during the parallelized procedure with nd as the number of domains obtained from the METIS partitioning. The global assembly of scalar values ($\widetilde{a}_2$, $\widetilde{b}_1$) as well as values of the vectors ($\widetilde{\boldsymbol{u}}_b, \widetilde{\boldsymbol{d}}_b$) are then computed and transferred by the *MPI_Allreduce* operation. A hybrid approach is to outsource the algorithmic parts of the PPCG to other hardware architectures, e.g. to graphics processing units. Such a hybrid implementation concept was realized in chapter 6, with corresponding results shown in section 6.5. There, the computation of the matrix-vector products (second row of table 5.9 considering the domain stiffness matrix $\boldsymbol{K}^{(j)}$) is simultaneously executed by several graphics processing units, based on the Nvidia Tesla architecture within a high-performance computing framework. By applying this approach, the domain stiffness matrix $\boldsymbol{K}^{(j)}$ is built and assembled from all elements of the corresponding subdomain of the decomposed finite element mesh (considering the partitioning techniques as described in section 4.3) and is stored at the corresponding graphics processing unit. A suitable and scalable preconditioning technique, which is adaptable for each subdomain of the decomposed structural problem is necessary for the acceleration of the distributed PCG iteration. The following sections give more details on the modified preconditioning techniques used in this work.

1.	INIT: $\boldsymbol{r} = \boldsymbol{f}$; $\widetilde{a}_1 =< \boldsymbol{r}; \boldsymbol{z} >$; $\boldsymbol{d} = \boldsymbol{z} = \widetilde{M^{-1}}\boldsymbol{r}$
2.	$\begin{pmatrix} \boldsymbol{K}_{ii} & \boldsymbol{K}_{ib} \\ \boldsymbol{K}_{ib}^T & \boldsymbol{K}_{bb} \end{pmatrix}^{(j)} \begin{pmatrix} \boldsymbol{d}_i \\ \boldsymbol{d}_b \end{pmatrix}^{(j)} = \begin{pmatrix} \hat{\boldsymbol{d}}_i \\ \hat{\boldsymbol{d}}_b \end{pmatrix}^{(j)}$
3.	$a_2^{(j)} =< \boldsymbol{d}^{(j)} ; \hat{\boldsymbol{d}}^{(j)} >$
4.	$\widetilde{a}_2 = \sum_{j=1}^{nd} a_2^{(j)}$; $\widetilde{\boldsymbol{d}}_b = \sum_{j=1}^{nd} \hat{\boldsymbol{d}}_b{}^{(j)}$; $\hat{\boldsymbol{d}}_b^{(j)} = \widetilde{\boldsymbol{d}}_b$
5.	$\alpha = \widetilde{a}_1 \widetilde{a}_2^{-1}$
6.	$\boldsymbol{z}^{(j)} = \boldsymbol{r}^{(j)} - \alpha \hat{\boldsymbol{d}}^{(j)}$; $\boldsymbol{r}^{(j)} = \boldsymbol{z}^{(j)}$; $\boldsymbol{z}^{(j)} = {\boldsymbol{M}^{(j)}}^{-1} \boldsymbol{r}^{(j)}$
7.	$\boldsymbol{h}^{(j)} = \boldsymbol{h}^{(j)} + \alpha \boldsymbol{d}^{(j)}$
8.	$b_1^{(j)} =< \boldsymbol{r}^{(j)} ; \boldsymbol{z}^{(j)} >$
9.	$\widetilde{b}_1 = \sum_{j=1}^{nd} b_1^{(j)}$
10.	$\beta = \widetilde{b}_1 \widetilde{a}_1^{-1}$; $\widetilde{a}_1 = \widetilde{b}_1$
11.	IF $\sqrt{\widetilde{b}_1} < TOL$ GOTO 14.
12.	ELSE $\boldsymbol{d}^{(j)} = \boldsymbol{z}^{(j)} + \beta \boldsymbol{d}^{(j)}$; $\hat{\boldsymbol{d}}^{(j)} = 0$
13.	$b_1^{(j)} = a_2^{(j)} = \widetilde{b}_1 = \widetilde{a}_2 = 0$ GOTO 2.
14.	SET $\widetilde{\boldsymbol{u}}_b = \sum_{j=1}^{nd} \boldsymbol{h}_b^{(j)}$; $\boldsymbol{u}_b^{(j)} = \widetilde{\boldsymbol{u}}_b$; $\boldsymbol{u}_i^{(j)} = \boldsymbol{h}_i^{(j)}$

Table 5.9: Algorithm for the parallelized preconditioned conjugate gradient method (PPCG).

5.6 Preconditioning techniques for conjugate gradients

5.6.1 In general

The preconditioning technique used is crucial for the memory demand and computing time required. The construction of the preconditioning matrix can be extensive in respect to main memory and determines the computing time of the CG iteration procedure itself. In this work, firstly, a parallelized version of the preconditioned conjugate gradient method (PPCG) was implemented based on domain decomposition without explicitly building the Schur complement system. Secondly, the preconditioning technique was restricted to a scaled main diagonal precondition technique with a special scaling parameter based on the upper or lower bound of the spectral radius of the assembled submatrices. This has the advantage that it reduces the time for building the preconditioning matrix as well as the memory demand and also the time for executing the matrix-vector product involving the preconditioning matrix. Finally, the sparse matrix-vector product performed for each subdomain considers different matrix storage formats. Due to the nodal storage scheme of the distributed FE data, a nodal compressed row storage can be used to improve the performance compared to the standard coordinate storage (*coo*) or the compressed row storage (*csr*).

5.6.2 Regular eigenvalue problems

In general, the FE discretization in quasi-static or time-independent discretizations in solid mechanics yields to a regular, symmetric and positive definite coefficient matrix $\boldsymbol{K}$ and by this, to a regular and solvable system of equations if the vector of nodal unknowns $\boldsymbol{u}$ as well as the vector of nodal forces $\boldsymbol{f}$ are taken into account. If $\boldsymbol{K}$ is a real matrix and the vector $\boldsymbol{f}$ is equal to zero such

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{0} \tag{5.48}$$

a linear homogeneous problem occurs. The assumption of

$$\det(\boldsymbol{K}) \neq 0 \tag{5.49}$$

and the extension of eq. (5.48) such as the matrix entries of $\boldsymbol{K}$ depend on the term λ yields to

$$\boldsymbol{K}(\lambda) \cdot \boldsymbol{u} = \boldsymbol{0} \tag{5.50}$$

with

$$\boldsymbol{K}(\lambda) = [k_{ij}(\lambda)] \tag{5.51}$$

The solution of this equation leads to a trival ($\boldsymbol{u} = \boldsymbol{0}$) or a non-trival solution of $\boldsymbol{u}$, whereby the non-trival solution occurs, if one value λ_i yields to the singularity of $\boldsymbol{K}$ with

$$\boldsymbol{u} \in R^n \quad \text{and} \quad \lambda_i = 1, 2, ..., n \tag{5.52}$$

so that

$$\det[\boldsymbol{K}(\lambda)] = 0 \tag{5.53}$$

The solution for the values of λ are the eigenvalues λ_i with corresponding eigenvectors $\boldsymbol{u}_i$. In general, the determination of all eigenvalues results in the solution of the eigenvalue problem of eq. (5.53), leading to a crucial computation of many determinants. The regular eigenvalue problem is defined as

$$\boldsymbol{K}(\lambda) = \boldsymbol{K}_0 - \lambda \cdot \boldsymbol{I} \tag{5.54}$$

with $\boldsymbol{I}$ being the identity matrix. Considering eq. (5.54) and eq. (5.50), the regular eigenvalue problem results in

$$(\boldsymbol{K}_0 - \lambda \cdot \boldsymbol{I}) \cdot \boldsymbol{u} = \boldsymbol{0} \tag{5.55}$$

or respectively

$$\boldsymbol{K}_0\boldsymbol{u} = \lambda\boldsymbol{u} \tag{5.56}$$

The assumption

$$\boldsymbol{K}_0 = \boldsymbol{K} \tag{5.57}$$

then leads to the solution of

$$\det(\boldsymbol{K} - \lambda \cdot \boldsymbol{I}_n) = 0 \equiv p_n(\lambda) \tag{5.58}$$

as the n-fold characteristic polynomial of the grade n with

$$p_n(\lambda) = (-\lambda)^n + \sum_{i=1}^{n} a_i \cdot (-\lambda)^{n-i} \tag{5.59}$$

and a_i as the reel coefficients of p_n.
The minimum and maximum eigenvalue are in (close) relation with the condition number of the coefficient matrix $\boldsymbol{K}$. In this work, the idea is to determine a global approximated (minimum and/or maximum) eigenvalue by minimum and/or maximum eigenvalues for each corresponding subdomain matrix of the distributed system of equations or the distributed eigenvalue problem, respectively. These global (minimum and/or maximum) eigenvales can be used for approximating the condition number of the matrix $\boldsymbol{K}$ and can then be considered for a numerical scaling strategy building the distributed preconditioning matrix and accelerating the distributed solver based on the preconditioned conjugate gradient method. For this reason, preconditioning techniques involving the numerical solution of corresponding eigenvalue problems will be presented in the following sections.

5.6.3 Jacobi-point preconditioning

The most memory-efficient preconditioning for a linear equation system results in a preconditioning matrix $\boldsymbol{M}^{-1}$, which approximates the inverse of matrix $\boldsymbol{K}$ such that

$$\boldsymbol{M}^{-1}\boldsymbol{K}\boldsymbol{u} = \boldsymbol{M}^{-1}\boldsymbol{f} \tag{5.60}$$

results from the consideration of the main diagonal of matrix $\boldsymbol{K}$ with

$$\boldsymbol{D} = \mathrm{diag}[\boldsymbol{K}] \tag{5.61}$$

Therefore, the preconditioning matrix $\boldsymbol{M}^{-1}$, which is the so-called Jacobi-point preconditioner, as

$$\boldsymbol{M}^{-1} = \omega \boldsymbol{D}^{-1} \tag{5.62}$$

is taking a scalar value ω into account. Eq. (5.62) is addressed in the following section, where the development of a technique for the determination of the scaling parameter ω and the adaption for a distributed preconditioning technique are the focus. Hence, a distributed and scalable computation based on preconditioned conjugate gradients (according to eq. (5.60)) with a maximized efficiency in respect to memory and time demand, is aimed.

5.6.4 Approximation of the condition number and preconditioning

The convergence behavior of such CG algorithms depends on the condition number of the global stiffness matrix $\boldsymbol{K}$, as well as on the preconditioning matrix, which in best case approximates the inverse of $\boldsymbol{K}$. Here, the relation between the condition number and a suitable preconditioning technique is described in regards to computational efficiency of such iterative solvers.

Lemma 1.1.
Let $\boldsymbol{K}_n = [k_{ij}] \in \boldsymbol{R}^{n\times n}$ be a n-by-n matrix, symmetric, positive definite and regular, and $\boldsymbol{I}$ the identity matrix. Then

$$\boldsymbol{K}\boldsymbol{K}^{-1} = \boldsymbol{I}. \tag{5.63}$$

The definition of the condition number considering *lemma 1.1* for the regular matrix $\boldsymbol{K}$ of eq. (5.60) is given by

$$\begin{aligned} \kappa(\boldsymbol{K}, \boldsymbol{K}^{-1}) = \|\boldsymbol{K}\boldsymbol{K}^{-1}\| &= \|\boldsymbol{I}\| \\ &= 1 \end{aligned} \tag{5.64}$$

with $\boldsymbol{I}$ as the identity matrix, $\boldsymbol{K}$ as the global coefficient matrix, $\boldsymbol{K} \in \boldsymbol{R}^{n\times n}$, and $\kappa(\boldsymbol{K})$ as the condition number of matrix $\boldsymbol{K}$. Based on the Jacobi preconditioning, the approximation of $\boldsymbol{K}^{-1}$ or the computation of the preconditioner $\boldsymbol{M}^{-1}$ is defined as

$$\boldsymbol{K}^{-1} \sim \boldsymbol{M}^{-1} = \mathrm{diag}[\boldsymbol{K}]^{-1} = \boldsymbol{D}^{-1} \tag{5.65}$$

Using eq. (5.65), the condition number according to eq. (5.64) will change to

$$\begin{aligned} \kappa(\boldsymbol{K}, \boldsymbol{D}^{-1}) = \|\boldsymbol{K}\boldsymbol{D}^{-1}\| \\ \geq 1 \end{aligned} \tag{5.66}$$

By introducing the scaling parameter ω for the relation between the condition numbers $\kappa(\boldsymbol{K}, \boldsymbol{K}^{-1})$ and $\kappa(\boldsymbol{K}, \boldsymbol{D}^{-1})$, ω results in

$$\omega = \frac{\kappa(\boldsymbol{K}, \boldsymbol{K}^{-1})}{\kappa(\boldsymbol{K}, \boldsymbol{D}^{-1})} \tag{5.67}$$

Lemma 1.2.
Let $\boldsymbol{K}_n = [k_{ij}] \in \boldsymbol{R}^{n\times n}$ be a n-by-n matrix, symmetric, positive definite and regular. Then

$$\boldsymbol{K}\boldsymbol{I} = \boldsymbol{K}. \tag{5.68}$$

With *lemma 1.2* and consideration of eq. (5.64) and eq. (5.66) changes eq. (5.67) to

$$\omega = \frac{\|\boldsymbol{K}\boldsymbol{K}^{-1}\|}{\|\boldsymbol{K}\boldsymbol{D}^{-1}\|} = \frac{\|\boldsymbol{I}\|}{\|\boldsymbol{K}\boldsymbol{D}^{-1}\|} \sim \frac{\|\boldsymbol{D}\|}{\|\boldsymbol{K}\|} \tag{5.69}$$

Lemma 1.3.
Let $\boldsymbol{K}_n = [k_{ij}] \in \boldsymbol{R}^{n\times n}$ be a n-by-n matrix, symmetric, positive definite and regular. Then

$$\omega\|\boldsymbol{K}_n\| = \|\omega\boldsymbol{K}_n\|. \tag{5.70}$$

With *lemma 1.3* eq. (5.69) leads to

$$\|\boldsymbol{K}^{-1}\| \sim \|\omega\boldsymbol{D}^{-1}\| = \|\boldsymbol{M}^{-1}\| \tag{5.71}$$

Therefore, the preconditioning Matrix $\boldsymbol{M}^{-1}$ is equivalent to

$$\boldsymbol{M}^{-1} = \omega\boldsymbol{D}^{-1} \sim \boldsymbol{K}^{-1} \tag{5.72}$$

Due to the presentation of the condition number as the upper bound of the spectral radius expressed by the maximum eigenvalue of a regular, symmetric und positive definite matrix (which is generally fulfilled), the scaling parameter ω can be computed as

$$\omega = \frac{\|\boldsymbol{D}\|}{\|\boldsymbol{K}\|} = \frac{\lambda_{max}(\boldsymbol{D})}{\lambda_{max}(\boldsymbol{K})} \tag{5.73}$$

5.6.5 Approximation of the scaling range

By using the scaled Jacobi preconditioning the induced solution error is limited to an automized approximation of the scaling range of the values for ω. Due to this, a second scaling parameter α_L is introduced which modifies eq. (5.72) to

$$\boldsymbol{M}^{-1} = \alpha_L\omega\boldsymbol{D}^{-1} \tag{5.74}$$

The preconditioned vector of residuals $\boldsymbol{z}$ considering eq. (5.9) results in

$$\begin{aligned}\boldsymbol{z} &= \boldsymbol{M}^{-1}\boldsymbol{r} \\ &= \alpha_L\omega\boldsymbol{D}^{-1}\boldsymbol{r} \\ &= \alpha_L\frac{\lambda_{max}(\boldsymbol{D})}{\lambda_{max}(\boldsymbol{K})}\boldsymbol{D}^{-1}\boldsymbol{r}\end{aligned} \tag{5.75}$$

With

$$b_1 = \boldsymbol{z}^T\boldsymbol{r} \tag{5.76}$$

the square root of the norm of the residuals as the break criteria of the chosen tolerance t for the PPCG interation considering eq. (5.12) leads to

$$\sqrt{b_1} = \sqrt{\left(\alpha_L\omega\boldsymbol{D}^{-1}\boldsymbol{r}\right)^T\boldsymbol{r}} \leq t \tag{5.77}$$

or

$$\frac{t^2}{\left(\alpha_L \omega \boldsymbol{D}^{-1}\boldsymbol{r}\right)^T \boldsymbol{r}} \geq 1 \tag{5.78}$$

with eq. (5.78), the limit value of α_L can be expressed as

$$\alpha_L \geq \frac{t^2}{\left(\omega \boldsymbol{D}^{-1}\boldsymbol{r}\right)^T \boldsymbol{r}} \tag{5.79}$$

By restricting the limit value of α_L and decreasing it by 10^{-1} a minimum of PCG iterations is assured. This will change eq. (5.79) to

$$\alpha_L = \frac{10\, t^2}{\left(\omega \boldsymbol{D}^{-1}\boldsymbol{r}\right)^T \boldsymbol{r}} \tag{5.80}$$

Moreover, the scaling range for the values of α_L can be given as

$$10\, t^2 \leq \alpha_L \left(\omega \boldsymbol{D}^{-1}\boldsymbol{r}\right)^T \boldsymbol{r} \leq \left(\omega \boldsymbol{D}^{-1}\boldsymbol{r}\right)^T \boldsymbol{r} \tag{5.81}$$

which is equivalent to

$$\frac{10\, t^2}{\left(\omega \boldsymbol{D}^{-1}\boldsymbol{r}\right)^T \boldsymbol{r}} \leq \alpha_L \leq 1 \tag{5.82}$$

The induced residual error ϵ will be computed by

$$\epsilon = \frac{\|\boldsymbol{f} - \boldsymbol{r}\|_F}{\|\boldsymbol{f}\|_F} \tag{5.83}$$

with vector $\boldsymbol{f}$ containing the values of the right hand side of the equation system, required to be preconditioned. In the following section the eigenvalue scaling strategy is adapted to a modified and parallelized version of the Jacobi preconditioning technique.

5.6.6 Modified and parallelized Jacobi preconditioning

To obtain a scalable version of this type of preconditioning, the Jacobi-point preconditioning technique and the scaling strategy are being combined and modified. The parallelization based on the domain decomposition method then simultaneously considers the main diagonal of each domain matrix. Moreover, the modified preconditioning technique (to obtain ${\boldsymbol{M}^{(j)}}^{-1}$ per domain) results from the scaled main diagonal or the scaled block diagonal of the global domain matrices $\boldsymbol{K}^{(j)}$ such that

$$\boldsymbol{M}^{(j)} = \begin{pmatrix} \boldsymbol{M}_{ii} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{M}_{bb} \end{pmatrix}^{(j)} = \begin{pmatrix} \operatorname{diag}\left[\boldsymbol{K}_{ii}\right] & \boldsymbol{0} \\ \boldsymbol{0} & \operatorname{diag}\left[\boldsymbol{K}_{bb}\right] \end{pmatrix}^{(j)} \tag{5.84}$$

where the necessary assembly (*MPI_Allreduce*) of the connecting boundaries results in

$$\widetilde{\boldsymbol{M}}_{bb} = \sum_{j=1}^{nd} \boldsymbol{M}_{bb}^{(j)} \tag{5.85}$$

The matrix inversion and modified scaling then yields to

$$\boldsymbol{M}^{(j)^{-1}} = \alpha_L \omega \begin{pmatrix} \boldsymbol{M}_{ii}^{(j)} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{M}_{bb}^{(j)} \end{pmatrix}^{-1} \tag{5.86}$$

$$= \alpha_L \omega \begin{pmatrix} \mathrm{diag}\big[\boldsymbol{K}_{ii}\big]^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \widetilde{\boldsymbol{M}}_{bb}^{-1} \end{pmatrix}^{(j)} \tag{5.87}$$

with the determination of the values for ω and α_L resulting from the findings of the previous section. The resulting eigenvalue problem which is to be solved to obtain ω in regard to eq. (5.73) can be given as

$$(\boldsymbol{K} - \lambda_i \boldsymbol{I})\boldsymbol{v} = \boldsymbol{0} \quad \text{with} \quad \boldsymbol{K} \in \boldsymbol{R}^{n \times n} \quad \text{and} \quad i = 1, ..., n \tag{5.88}$$

Then, all eigenvalues can be computed by

$$\det(\boldsymbol{K} - \lambda_i \boldsymbol{I}) = 0 \quad \text{with} \quad \boldsymbol{K} \in \boldsymbol{R}^{n \times n} \quad \text{and} \quad i = 1, ..., n \tag{5.89}$$

and from the set of solutions for λ the maximum eigenvalue will be chosen. Respectively, for the determination of $\lambda_{max}(\boldsymbol{D})$ eq. (5.89) changes to

$$\det(\boldsymbol{D} - \lambda_i \boldsymbol{I}) = 0 \quad \text{with} \quad \boldsymbol{D} \in \boldsymbol{R}^{n} \quad \text{and} \quad i = 1, ..., n \tag{5.90}$$

with $\boldsymbol{D}$ as the main diagonal of $\boldsymbol{K}$. Considering a non-overlapping domain decomposition method, the eigenvalues problem for each subdomain j results in as follows

$$\det(\boldsymbol{K}^{(j)} - \lambda_i^{(j)} \boldsymbol{I}) = 0 \tag{5.91}$$

and will consequently result in

$$\det(\boldsymbol{D}^{(j)} - \lambda_i^{(j)} \boldsymbol{I}) = 0 \tag{5.92}$$

For the computation of the upper bound of the spectral radius (which is equivalent to the maximum eigenvalue), the von-Mises-Wielandt algorithm as described in chapter 3 (section 3.4.3) is adapted. A decreased accuracy still yields to reasonable eigenvalues in acceptable computational time. Considering the maximum eigenvalue $\lambda_{max}^{(j)}$ per domain j if n domains are taken into account, the average maximum eigenvalue for $\boldsymbol{K}$ is computed as

$$\widetilde{\lambda}_{max}(\boldsymbol{K}) = \frac{1}{n} \sum_{j=1}^{n} \lambda_{max}^{(j)}(\boldsymbol{K}) \tag{5.93}$$

as well as for $\boldsymbol{D}$

$$\widetilde{\lambda}_{max}(\boldsymbol{D}) = \frac{1}{n} \sum_{j=1}^{n} \lambda_{max}^{(j)}(\boldsymbol{D}) \tag{5.94}$$

and, by that, a consistent global scaling parameter ω for each domain is obtained by

$$\omega = \omega^{(j)} = f(\widetilde{\lambda}_{max}(\boldsymbol{K}), \widetilde{\lambda}_{max}(\boldsymbol{D})) = const. \tag{5.95}$$

This kind of determination of the scaling parameter is also shown in the following section, here respecting an alternative preconditioning approach, the so-called Schur preconditioning.

5.6.7 Schur preconditioning

In order to gain an optimal convergence rate for the iterative computation of the final equation system and for a comparison to the Jacobi-point preconditioner, a modified Schur preconditioning technique is applied ([Loghin et al. 2003], [Langer 2008]). The construction of the preconditioning matrix ${\boldsymbol{M}^{(j)}}^{-1}$ derives from the original LU factorization (which is shown in the following notation for ${\boldsymbol{M}^{(j)}}^{-1}$) as the exact inverse of $\boldsymbol{K}^{(j)}$

$${\boldsymbol{M}^{(j)}}^{-1} = {\boldsymbol{K}^{(j)}}^{-1} \tag{5.96}$$

$$= \begin{pmatrix} \boldsymbol{I} & -\boldsymbol{K}_{ii}^{-1}\boldsymbol{K}_{bi}^{T}{\widetilde{\boldsymbol{S}}_{bb}}^{-1} \\ \boldsymbol{0} & {\widetilde{\boldsymbol{S}}_{bb}}^{-1} \end{pmatrix}^{(j)} \begin{pmatrix} \boldsymbol{K}_{ii}^{-1} & \boldsymbol{0} \\ -\boldsymbol{K}_{bi}\boldsymbol{K}_{ii}^{-1} & \boldsymbol{I} \end{pmatrix}^{(j)} \tag{5.97}$$

with ${\widetilde{\boldsymbol{S}}_{bb}}^{-1}$ as the inverted assembled Schur complement of matrix $\boldsymbol{K}$ (according to the nodal d.o.f.s of the connecting boundary) with

$${\widetilde{\boldsymbol{S}}_{bb}}^{-1} = \left(\sum_{j=1}^{n} \boldsymbol{S}_{bb}^{(j)} \right)^{-1} = \left(\sum_{j=1}^{n} \left(\boldsymbol{K}_{bb} - \boldsymbol{K}_{bi}\boldsymbol{K}_{ii}^{-1}\boldsymbol{K}_{bi}^{T} \right)^{(j)} \right)^{-1} \tag{5.98}$$

However, the exact construction of ${\widetilde{\boldsymbol{S}}_{bb}}^{-1}$ and ${\boldsymbol{M}^{(j)}}^{-1}$ per domain is numerically expensive. Moreover, the quality of the preconditioning matrix related to its convergence properties for the iterative CG solver depends on the quality of the approximation of the inverse of $\boldsymbol{K}_{ii}$. Hence, the approximated inverses ${\widetilde{\boldsymbol{M}}_S}^{-1}$ and $\boldsymbol{M}_{K_{ii}}^{-1}$ are introduced, where ${\widetilde{\boldsymbol{M}}_S}^{-1}$ describes the approximation of ${\widetilde{\boldsymbol{S}}_{bb}}^{-1}$ and $\boldsymbol{M}_{K_{ii}}^{-1}$ of $\boldsymbol{K}_{ii}^{-1}$, respectively. Now, the modified notation for ${\boldsymbol{M}^{(j)}}^{-1}$ decomposed in ${\boldsymbol{M}_1^{(j)}}^{-1}$ and ${\boldsymbol{M}_2^{(j)}}^{-1}$ yields to

$${\boldsymbol{K}^{(j)}}^{-1} \cong {\boldsymbol{M}^{(j)}}^{-1} = {\boldsymbol{M}_1^{(j)}}^{-1}{\boldsymbol{M}_2^{(j)}}^{-1} \tag{5.99}$$

using the LU factorization leading to

$${\boldsymbol{M}^{(j)}}^{-1} = \begin{pmatrix} \boldsymbol{I} & -\boldsymbol{M}_{K_{ii}}^{-1}\boldsymbol{K}_{bi}^{T}\widetilde{\boldsymbol{M}}_S^{-1} \\ \boldsymbol{0} & \widetilde{\boldsymbol{M}}_S^{-1} \end{pmatrix}^{(j)} \begin{pmatrix} -\boldsymbol{M}_{K_{ii}}^{-1} & \boldsymbol{0} \\ -\boldsymbol{K}_{bi}\boldsymbol{M}_{K_{ii}}^{-1} & \boldsymbol{I} \end{pmatrix}^{(j)} \tag{5.100}$$

The scaling strategy with parameter ω for such decomposed preconditioning matrices ${\boldsymbol{M}^{(j)}}^{-1}$ (as described in the previous section) consequently results in a modified version of the Schur preconditioning corresponding to the notation presented in the next section 5.6.8.

1.	INIT: $\boldsymbol{z}^{(0)} = \boldsymbol{d}^{(0)}$ and $\hat{\boldsymbol{z}} = 0$; determine $\alpha_L \omega$
2.	$\hat{\boldsymbol{z}}_i = {\boldsymbol{M}_{K_{ii}}^{(j)}}^{-1} \boldsymbol{d}_i$
3.	$\boldsymbol{z}_b^{(1)} = \boldsymbol{z}_b^{(0)} - \boldsymbol{K}_{bi} \hat{\boldsymbol{z}}_i$
4.	$\boldsymbol{z}_b^{(1)} = \alpha_L \omega \widetilde{\boldsymbol{M}}_S^{-1} \boldsymbol{z}_b^{(1)}$
5.	$\boldsymbol{z}_i^{(1)} = \boldsymbol{z}_i^{(0)} - {\boldsymbol{K}_{bi}}^T \boldsymbol{z}_b^{(1)}$
6.	$\boldsymbol{z}_i^{(1)} = \alpha_L \omega \boldsymbol{M}_{K_{ii}}^{-1} \boldsymbol{z}_i^{(1)}$
7.	SET: $\boldsymbol{z} = \begin{pmatrix} \boldsymbol{z}_i^{(1)} \\ \boldsymbol{z}_b^{(1)} \end{pmatrix}$

Table 5.10: Algorithm for the vector-based Schur preconditioning according to eq. (5.104).

5.6.8 Modified Schur preconditioning

In this work, two versions of the modified Schur preconditioning are compared: firstly, using the factorized matrix of $\boldsymbol{K}_{ii}^{(j)}$, secondly the inverted and scaled main diagonal of $\boldsymbol{K}_{ii}^{(j)}$ as ${\boldsymbol{M}_{K_{ii}}^{(j)}}^{-1}$. In both cases the non-scaled inverted main diagonal of ${\widetilde{\boldsymbol{S}}_{bb}}^{(j)}$ as $\widetilde{\boldsymbol{M}}_S^{(j)}$ has been proven to be sufficient in comparison to the performance of the CG method without any preconditioning. There, a main diagonal preconditioning based on $\mathrm{diag}[\boldsymbol{K}^{(j)}]$ was used to avoid an explicit factorization of $\boldsymbol{K}_{ii}$. By this

$$ {\boldsymbol{M}_{K_{ii}}^{(j)}}^{-1} = \left(\mathrm{diag}\left[\boldsymbol{K}_{ii}^{(j)} \right] \right)^{-1} \tag{5.101} $$

and

$$ {\widetilde{\boldsymbol{M}}_S^{(j)}}^{-1} = \left(\mathrm{diag}\left[\widetilde{\boldsymbol{S}}_{bb} \right] \right)^{-1} \tag{5.102} $$

can explicitly be formulated. Considering the scaling parameters this leads to

$$ \begin{aligned} {\boldsymbol{M}^{(j)}}^{-1} = \alpha_L \omega & \begin{pmatrix} \boldsymbol{I} & -\left(\mathrm{diag}\left[\boldsymbol{K}_{ii} \right] \right)^{-1} \boldsymbol{K}_{bi}^T \left(\mathrm{diag}\left[\widetilde{\boldsymbol{S}}_{bb} \right] \right)^{-1} \\ \boldsymbol{0} & \left(\mathrm{diag}\left[\widetilde{\boldsymbol{S}}_{bb} \right] \right)^{-1} \end{pmatrix}^{(j)} \\ & \begin{pmatrix} -\left(\mathrm{diag}\left[\boldsymbol{K}_{ii} \right] \right)^{-1} & \boldsymbol{0} \\ -\boldsymbol{K}_{bi} \left(\mathrm{diag}\left[\boldsymbol{K}_{ii} \right] \right)^{-1} & \boldsymbol{I} \end{pmatrix}^{(j)} \end{aligned} \tag{5.103} $$

Consequently, this results in an efficient vector scaling rather than in a numerically-expensive matrix factorization which would finally lead to a dense matrix-vector operation caused by the preconditioning. For the implementation of the Schur preconditioning technique it is required to reduce the number of numerical operations involving $\boldsymbol{M}_1^{(j)}$ and $\boldsymbol{M}_2^{(j)}$ to a minimum in order to enable an efficient execution of the Schur preconditioning during the iterative CG procedure. Hence, the following algorithm of table 5.10 has been developed for the precondition of vector $\boldsymbol{d}$ to obtain vector $\boldsymbol{z}$. After initialization of $\boldsymbol{z}$, five

computational steps of the algorithm need simultaneously to be performed per subdomain (equal to eq. (5.104)) such as

$$\boldsymbol{z}^{(j)} = \boldsymbol{M}^{(j)^{-1}} \boldsymbol{d}^{(j)} = \boldsymbol{M}_1^{(j)^{-1}} \boldsymbol{M}_2^{(j)^{-1}} \begin{pmatrix} \boldsymbol{d}_i \\ \boldsymbol{d}_b \end{pmatrix}^{(j)} = \begin{pmatrix} \boldsymbol{z}_i \\ \boldsymbol{z}_b \end{pmatrix}^{(j)} \tag{5.104}$$

This and the previous chapter provide the mechanical and mathematical background for a scalable implementation concept with the addressed goal to enable large-scale simulations of the (nonlinear) material behavior of multiphase specimens. With the special focus on the computational efficiency respecting memory and time demand, a high-performance framework was therefore developed and evaluated. The resulting hybrid computing approach combines the scalablility of different hardware architectures. Numerical results for the scalability of the implemented algorithms on two high-performance computers (e.g. both provided by the high-performance computing center Stuttgart - HLRS) are then being presented.

Chapter 6

Hybrid high-performance computing

The following sections give a short overview of and an indroduction to the hardware architectures and parallel programming concepts currently available. In general, there are two different hardware architectures (taking the type of parallel processing into account): namely the shared memory processing (SMP) and also the distributed memory processing (DMP). In the past decades these concepts were technically realized by using central processing units (CPUs). However, with the introduction of the graphics processing unit (GPU) as a computational SMP unit, this hardware architecture has been led to a new era of scientific computing, which is nowadays used in hybrid CPU-GPU high-performance computers. After the technical overview this chapter closes with several benchmarks in respect to linear-elastic FEA of different large-scale discretized problems.

6.1 Parallel hardware architectures

6.1.1 Shared memory processing (SMP)

According to [Robbins et al. 2003] and [Chapman et al. 2007], shared memory computer systems typically use a large block of random access memory which is accessed by several central processing units (CPUs) in a multiple-processor computer system. A shared memory system requires relatively simple programming constructs, since all processors share a single view of data and the communication between the processors can be as fast as their memory accesses to the same location. The challenge of shared memory systems is that many CPUs need fast access to the global memory and therefore also to its cache memory. This has main disadvantages. Firstly, the CPU-to-memory connection causes a bottleneck for the data transfer and secondly, shared memory computers are not scaling well. However, the cache coherence is crucial for the overall performance of these systems. Whenever one cache is updated with information which may be used by other processors, the change in data is to be transmitted to the other processors, otherwise

the different processors will be working with incoherent data. Coherence protocols for cache and memory coherence can provide an extremely high-performance access to information shared between multiple processors, but on the other hand, there is a risk that they become overloaded, which will result in a reduced data transfer. The programming construct using shared memory processing is the *open specifications for multiprocessing* (openMP) interface, which was introduced as an open standard in 1997. Here, SMP hardware architectures (e.g. a CPU-socket with multiple CPU-cores) use the cache-coherent non-uniform memory access (ccNUMA, fig. 6.1) to communicate. Other than the unified memory access systems (UMA), the memory location and latency are important for the performance of the ccNUMA systems. Moreover, NUMA systems are distributed shared

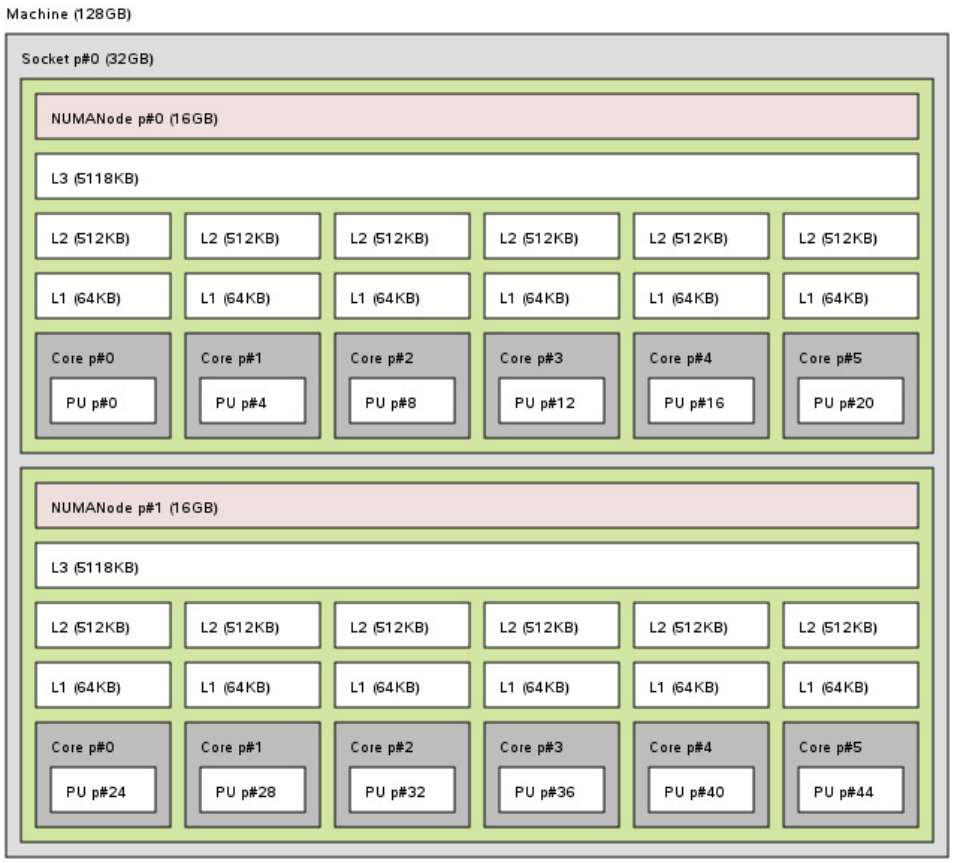


Figure 6.1: Architectural topology of a multicore shared memory CPU system: Detailed view of one socket (of a four socket system) with two ccNuma nodes (16 GB memory for each node) equipped with a 12-core AMD Opteron CPU 6100 series (Magny-Cours).

memory systems, where each NUMA node is connected to its own local memory. Simultaneously, each NUMA node has full access to the non-local memory, but with an increased latency.

6.1.2 Distributed memory processing (DMP)

Distributed memory processing occurs, if several symmetric memory-based NUMA or ccNUMA nodes are connected by network communication links and standard network protocols, such as e.g. the InfiniBand, which is mainly used for the interconnection in high-performance computing. It allows programs to interact with all available processors. The key issue in the programming of distributed memory systems is, how to efficiently distribute the data among the different memories, a task being one of the main responsibility of the programmer. Moreover, the implementation concept often differs depending on the problem which is to be investigated. Hence, a suitable parallel programming environment for DMP systems is to be sought. Due to this, the message-passing interface standard (MPI, [Gropp et al. 1999]) was developed as a standardized and portable message-passing system to concurrently utilze all connected local or non-local ccNuma nodes. This parallelization approach is used and further investigated in this work.

6.1.3 Graphics processing unit (GPU) and general-purpose graphics processing unit (GPGPU)

With the availability of the CUDA programming framework in 2007, basic mathematical operations such as routines of linear algebra can numerically be executed on the GPU. Today, a varity of tools and free software packages are available to enable access to the GPU hardware resources. Currently, hybrid platforms in high-performance supercomputers are used, which provide many GPU boards, as for example are included in rack based systems based on the Tesla or Fermi architecture [NVIDIA 2009e]. The evaluation of the performance of large-scale finite element models using a hybrid CPU-GPU high-performance computing cluster, such as the NEC Nehalem at the high-performance computing center Stuttgart, is one of the objectives of this work. This also creates new opportunities to use general-purpose computations on graphics hardware with multiple graphics processing units (GPGPU, [NVIDIA 2009d], [NVIDIA 2009c], [NVIDIA 2009a]).

6.2 Parallel programming techniques

6.2.1 Open specifications for multi-processing (openMP)

OpenMP [Chapman et al. 2007] is an open standard which implements multithreading for SMP systems, a method of parallelization whereby the master 'thread' (a series of instructions executed consecutively) is managing a specified number of slave threads and divides the tasks among them. The threads then run concurrently within the runtime environment and are allocated to different processors. The section of parallel code is

Figure 6.2: GPU advantage: More GPU transistors as algorithmic logic units (ALU, right) are devoted to data processing rather than data caching and flow control compared to CPU (left).

induced by a preprocessor directive (usually starting with '#pragma omp ...'), causing the threads to be initialized before the section is executed. Each thread has an *id* attached to it (as an integer data type), which starts at the master thread with a value of zero. After the execution of the parallelized code, the threads join back into the master thread and the code runs sequentially to the next parallelized section. This procedure continues until the end of the program. By default, each thread executes the parallelized section of code independently. Work-sharing constructs can be used to divide a task among the threads, enabling each thread to execute its allocated part of the code. Both tasks, parallelism and data parallelism are achieved using openMP in this way. The runtime environment allocates threads to processors depending to the type of usage, the machine load and the scheduling factors of the running hardware system. Moreover, the number of threads is defined by the runtime environment based on the environment variables or, alternatively, directly in the programming code by using openMP functions. These functions are included in the header file *omp.h* which is available in C and C++ language bindings.

6.2.2 Message-passing interface (MPI)

According to [Forum 2008] MPI is a message-passing library interface specification. MPI primarily addresses the message-passing parallel programming model, in which data is moved through cooperative operations on each individual process from the address space of one process to that of another process. Extensions to the classical message-passing model are provided in collective operations, remote-memory access operations, the dynamic process creation and the parallel input and output (I/O). MPI is a specification, not an implementation, however, multiple implementations of MPI are existing. Its specification is suitable for a library interface. Furthermore, MPI is also not a language, and all MPI operations are expressed as functions, subroutines, or methods according to the appropriate language bindings for C, C++, Fortran-77 and Fortran-95, all being part of the MPI standard. The basic goal of the message-passing interface is to develop a widely

used standard for writing message-passing programs. This interface was established as a portable and efficient standard for message-passing. In 1994, the first official MPI version was introduced [Gropp et al. 1994], with updated versions and specifications available today. Furthermore, hybrid programming frameworks enable the combination of openMP and MPI parallelization techniques.

6.2.3 Compute unified device architecture (CUDA)

The compute unified device architecture (CUDA, [NVIDIA 2009f], [NVIDIA 2009g]) is a parallel computer architecture, which was developed by Nvidia. In February 2007, the initial CUDA SDK was published for Microsoft Windows and Linux [NVIDIA 2009b]. CUDA is the computing engine used in Nvidia graphics processing units (GPUs), which is available to software developers in a varity of standard programming languages. Programmers can use *C for CUDA*, which basically is ISO C with specific Nvidia extensions (with several restrictions) compiled through a PathScale Open64 C compiler to code algorithms for the execution on the GPU. CUDA provides a range of computational interfaces also available for Python, Perl, Fortran, Java, Ruby, Lua, MATLAB/Jacket and IDL. CUDA grants the developers access to the virtual instruction set and memory of the parallel computational elements in CUDA GPUs. Using CUDA, the latest Nvidia GPUs become accessible for computation like CPUs. Unlike CPUs however, GPUs have a parallel throughput architecture which executes many concurrent threads slowly, rather than quickly executing a single thread in CPUs. This approach of solving general purpose problems on GPUs is known as GPGPU. In addition to graphics rendering, GPUs are used in game physics calculations in the computer game industry include PhysX. Hence, CUDA is also applied to accelerate non-graphical applications in computational biology, cryptography and various other fields of application. CUDA also works with all Nvidia GPUs from the G8x series onwards, including GeForce, Quadro and the Tesla line. Initial tests in this work have been executed on a GeForce GTX 280 graphic card, followed by the evaluation of the scalability at multiple Tesla units, which was used in a hybrid high-performance computing framework. This is described in section 6.3 and 6.5.1.

6.2.4 Performance characteristics

Sparse storage formats

In CPU as well as in GPU based computations, several matrix storage formats according to table 6.1 can be used to linear algebra operations, such as e.g. the numerical computation of matrix-vector products. The performance of such matrix-vector operations depends on the chosen storage format in conjuction with the hardware architecture (e.g. CPU or GPU). This is crucial for the efficiency of the distributed iterative solution technique based on the parallelized preconditioned conjugate gradients. Conventional matrix

storage formats are the coordinate format (*coo*), which stores the row index, the column index and the value of each matrix entry or the compressed row storage format, which improves *coo* by using index points of the length of each row, storing these row pointers and its column index plus the value of the corresponding matrix entry. In this work, a nodal compressed sparse row storage *ndcsr* format based on FE node incidences was implemented using the advantage of index-free storage of the matrix values. Special storage formats for GPU computing are the ellpack format *ell* and also the hybrid version *hyb*, combining the ellpack with the coordinate storage format. The adequate application of

coo	row and column index, value of matrix value (CPU and GPU)
csr	row pointer, column index, value of the matrix entry (CPU and GPU)
ndcsr	nodal *csr* storage based on nodal FE incidencies (CPU)
ell	ellpack format for graphics processing units (GPU)
hyb	hybrid format as combination of coordinate and ellpack storage formats (GPU)

Table 6.1: Different matrix storage formats for CPU and GPU computing.

such matrix storage schemes depends on the hardware architecture used and on the matrix sparsity, emphased by numerical tests for which results are illustrated later in this chapter.

Memory demand and computing time

The advantage of the *ndcsr* storage format is a decrease of memory demand, as shown in table 6.2, where the matrix entries are stored in double precision (eight bytes for one matrix entry) and the indices as integer data type (four bytes for one index). The scaling of the memory demand is shown in the diagram in figure 6.3 (right) and is compared to the *coo* and *csr* storage scheme. With n as the number of nonzero matrix entries, m as the number of d.o.f.s (assuming three translational d.o.f.s. per finite element node) and $n > m$, the memory efficiency over the standard *coo* format is improving, if the relation n by m is increasing. This is the case for large-scale high-dimensional finite element problems in 3D. Furthermore, the *ndcsr* matrix storage enables a faster execution of the matrix-vector products, being essential for the performance of the iterative solver technique. Numerical results considering a PPCG solver which are comparing the influence of the storage scheme to the solver performance are also shown in the next subsections.

6.2.5 Implementation characteristics

Framework of nodal block allocation

The numerical integration of the element stiffness matrix for all elements of the specific subdomain and the assembly of the global stiffness matrix per subdomain is executed in parallel. For an accelerated computation, the nodal compressed row storage (*ndcsr*) of

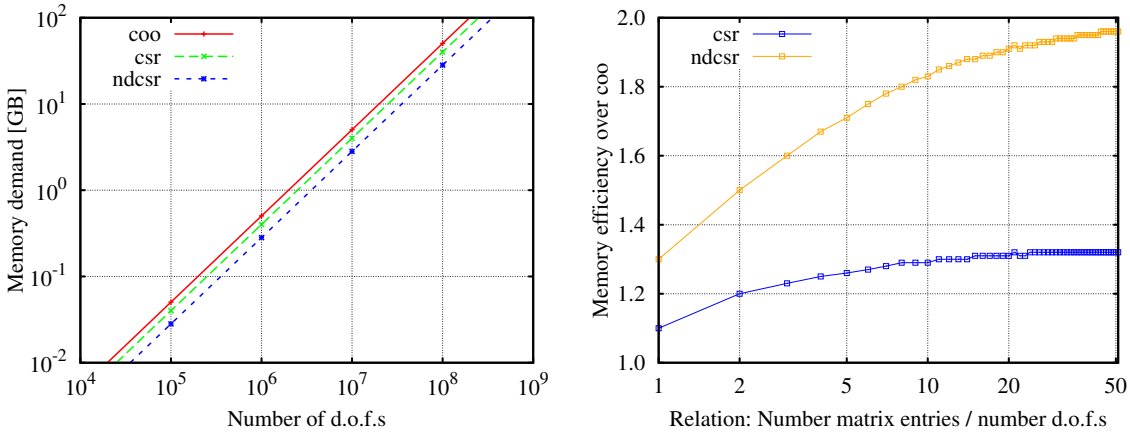


Figure 6.3: Memory demand (in Gigabyte, GB) of a *ndcsr* storage scheme compared to standard *coo* and *csr* storage format depending on the number of global d.o.f.s. (left), Scaling of the memory efficiency of *ndcsr* and *csr* storage schemes compared to the standard *coo* storage format (1.0) depending on the relation of the number of matrix entries and global d.o.f.s.

the finite element data was implemented allowing fast access to specific positions of the coefficient matrices where the indices are determined by the position of the corresponding nodal block instead considering the nodal d.o.f.s. It also enables efficient matrix-vector operations, while comparing it to the standard coordinate storage (*coo*), or alternatively, to the compressed row storage (*csr*). The computational efficiency in regards to the numerical integration, the assembly of the global stiffness matrix and the solution of the resulting equation system are compared and investigated for several FE examples, starting in section 6.4. In this approach, the assembled global stiffness matrix is stored among the finite element nodes in assembled blocks $\widetilde{\boldsymbol{B}}^{(j)}$ per subdomain, with j storing all matrix entries without the necessity to separately store the indices of the matrix entries. Moreover, the incidencial nodes of one finite element are defining one nodal stiffness block (after the numerical integration of that element) in conjunction with the neighboring nodes of the same element. One nodal block $\mathbf{A}_i^{(j)}$ of subdomain j contains all n by n matrix entries of the corresponding nodal block with n as the number of d.o.f.s per node (e.g. $n = 3$ in case of three translational d.o.f.s considered for 3D finite elements).

storage format	**memory** entries	**memory** indices	**memory** total	**efficiency** n:m=3	n:m=5
coo	8n	8n	16n	1.0	1.0
csr	8n	4n+4m	12n+4m	1.2	1.25
ndcsr	8n	8m	8n+8m	1.5	1.67

Table 6.2: Memory demand (in byte) of the global matrix for different matrix storage schemes: n as the number of nonzero matrix entries and m as the number of nodal degrees of freedom (d.o.f.s).

Consequently, the set of assembled nodal blocks $\widetilde{\mathcal{A}}_i^{(j)}$ stored in $\widetilde{\boldsymbol{B}}^{(j)}$ can be expressed as

$$\widetilde{\boldsymbol{B}}^{(j)} = \bigcup_{i=1}^{n^{(j)}} \widetilde{\mathcal{A}}_i^{(j)} = \{1, ..., n^{(j)}\} \tag{6.1}$$

with

$$\widetilde{\mathcal{A}}_i^{(j)} \cap \widetilde{\mathcal{A}}_k^{(j)} = \emptyset, \quad 1 \leq i < k \leq n^{(j)} \tag{6.2}$$

All nodal blocks are stored respecting their neighboring nodes in ascending order. As an advantage, this kind of nodal block assembly can be done independently and simultaneously for all subdomains, which leads to ideal parallel performance scalings, also if the number of subdomains is increasing. Therewith, $\widetilde{\boldsymbol{B}}^{(j)}$ can be split by the type of the global location in the global block matrix, a diagonal or off-diagonal block position

$$\widetilde{\boldsymbol{B}}^{(j)} = \begin{cases} \widetilde{\boldsymbol{B}}_D^{(j)} & \text{nodal blocks of the main diagonal} \\ \widetilde{\boldsymbol{B}}_L^{(j)} & \text{else} \end{cases} \tag{6.3}$$

with the specification of the assembled block diagonals

$$\widetilde{\boldsymbol{B}}_D^{(j)} = \bigcup_{i=1}^{n_D^{(j)}} \widetilde{\mathcal{A}}_{i,D}^{(j)} = \{1, ..., n_D^{(j)}\} \tag{6.4}$$

and for the off-diagonal position, respectively, with

$$\widetilde{\boldsymbol{B}}_L^{(j)} = \bigcup_{i=1}^{n_L^{(j)}} \widetilde{\mathcal{A}}_{i,L}^{(j)} = \{1, ..., n_L^{(j)}\} \tag{6.5}$$

Using this notation, the nodal compressed row storage format, fully scalable based on decomposed finite element meshes, may also be used for the efficient computation of matrix-vector products [Bell et al. 2008a].

Nodal compressed row storage for sparse matrix-vector operations

To improve the performance of the PPCG procedure, the matrix-vector product (step 2 of table 5.9) in conjunction with the assembled submatrix of the corresponding subdomain can efficiently and blockwise be performed [Bulu et al. 2009], using the nodal compressed row storage, which results from the nodal block allocation and the storage scheme of the finite element data [Bell et al. 2008b]. The proposed algorithm of table 6.3 then computes the matrix-vector product while separately considering the entries of the diagonal as well as of the off-diagonal nodal blocks $\widetilde{\boldsymbol{B}}_D^{(k)}$ and $\widetilde{\boldsymbol{B}}_L^{(k)}$, respectively, (with n as the number of nodal d.o.f.s) and replaces the second line of the PPCG algorithm, which is listed in table 5.9. Later in this chapter 6, the time benefits are shown compared to the standard coordinate (*coo*) and the standard compressed row storage (*csr*) schemes for sparse matrices resulting from several benchmark tests.

```
1.    for ii = 0 to n^n − 1 do (Loop over all nodes)
2.        i = ii · n^2
3.        r = ii · n
4.        for j = 0 to n   (Loop over all diagonal block entries)
5.            for k = 0 to n
6.                p_1 = r + j
7.                p_2 = r + k
8.                q = i + n · j + k
9.                d̂_{p_1}^{(k)} = d̂_{p_1}^{(k)} + B_{D,qk}^{(k)} d_{p_2}^{(k)}
10.           end for
11.       end for
12.       for l = 0 to n_{ii}^c − 1
13.           c = n · l
14.           for j = 0 to n (Loop over all off-diagonal block entries)
15.               t = p_m + j
16.               for k = 0 to n
17.                   p = r + j
18.                   q = c + k
19.                   d̂_p^{(k)} = d̂_p^{(k)} + B_{L,tk}^{(k)} d_q^{(k)}
20.                   p = c + j
21.                   q = r + k
22.                   d̂_p^{(k)} = d̂_p^{(k)} + B_{L,tk}^{(k)} d_q^{(k)}
23.               end for
24.           end for
25.       end for
26.   end for
```

Table 6.3: Algorithm for matrix-vector operations considering diagonal and off-diagonal nodal block matrices $\boldsymbol{B}_D^{(k)}$ and $\boldsymbol{B}_L^{(k)}$ per domain k.

6.2.6 Concluding remarks

The computational performance of the different algorithms presented will be evaluated by a benchmark for the sequential comparison between GPU (type of GTX285, GeForce architecture) and CPU (standard linux workstation) computing as shown in section 6.3. Hence, an example for a simplified elastic-inelastic decomposition is considered and the iterative PCG computation is performed, taking different preconditioning techniques into account and especially the modified version of the Schur preconditioning (table 6.5). Section 6.4 will show several scaling tests where homogeneous as well as hybrid meshed and also partitioned specimens were used. The benchmark was executed on a standard four-socket server. Section 6.5 states the results of large-scale computations of micro-structural bone material embedded in a hybrid high-performance computing framework. There, the PPCG method is taking the scaling strategy for building the preconditioning

matrix into account, where the computation of the matrix-vector products is outsourced to several graphics processing units (based on the Nvidia Tesla technology). The hardware used is provided by the NEC Nehalem high-performance computing cluster at the HLRS. A second high-performance computing framework in combination with the CRAY XE6 cluster (in operational mode since the beginning of 2012) allows to increase a large-scale finite element model based on computer-tomographic scans using a casted nickel-alloy specimen up to more than 42 million d.o.f.s (see section 6.6).

6.3 Sequential CPU and GPU computing

6.3.1 Hardware architecture

The chosen CPU system is a four-noded AMD Opteron rack system, where each node contains a six-core Istanbul Opteron with a local memory of 32 GB RAM and 9 MB common L3 cache. The cores have a clock speed of 2800 MHz with 128 kB L1 and 512 kB L2 cache configuration. Additionally, the rack system is connnected to an external graphical subsystem containing two Nvidia Quadro FX 5800 graphic cards with 2 x 4 GB of local memory and a clockspeed of 1300 MHz per computation unit. An illustration of the hardware used is shown in figure 6.4.

Figure 6.4: Nvidia Quadroplex D2 system (left) including two Quadro 5800 FX cards (right).

6.3.2 Example: Elastic-inelastic domain split

The first example evaluated is a simple block structure with a mixed mesh, where a homogeneous and an aligned meshing technique for one of the two domains was used (as shown in figure 6.5). The resulting domain stiffness matrix of the coarse and regular meshed part was condensed to the boundary d.o.f.s using the Schur complement method. Here, the assembled subdomain matrix of the aligned mesh was iteratively solved involving the Schur complement of the first domain as well as several preconditioning techniques (with the corresponding notation as stated in section 5.6.7).

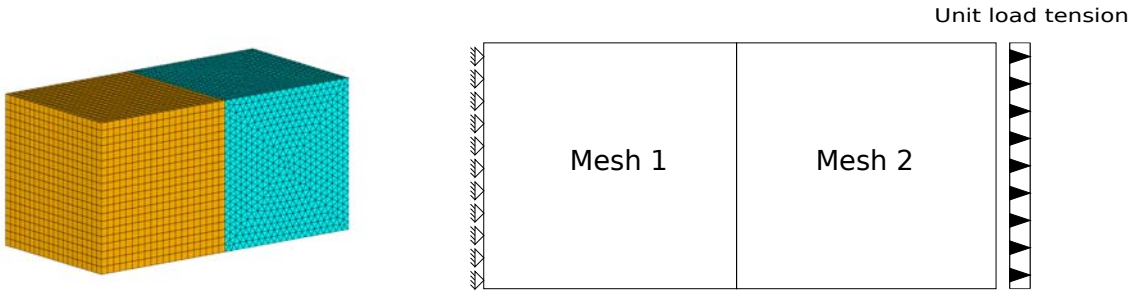


Figure 6.5: Numerical example applying the elastic-inelastic domain split with corresponding boundary and loading conditions.

6.3.3 Benchmark

Table 6.5 shows the results of using one CPU unit and different preconditioning techniques for solving the linear equation system considering an elastic-inelastic decomposition. The results were then compared to the execution time obtained from a Nvidia Quadro Plex 2200 system, as described in the previous section. According to the Quadro Plex 2200, the performance evaluation considers the standard coordinate format for the matrix storage with double precision for the algebraic computations and the texture cache of the GPU unit. The time was measured considering double precision and was moderately higher than the time measured based on 32 bit floating point operations. Additionally, the use of graphical texture cache led to further reductions in execution time and, thereby, compensated some of the computational effort caused by the increase in precision.

domain	**mesh type**	**d.o.f.s**	**sparsity**
elastic	regular	$\sim$ 70000	dense
inelastic	aligned	$\sim$ 90000	sparse

Table 6.4: Size of the two decomposed domains related to the number of degrees of freedom.

precond.	**architecture**	**matrix format**	**iterations**	**time** [sec]	**approximation**
-	1cpu	*coo*	786	15	-
diag	1cpu	*coo*	578	11	$\boldsymbol{D}^{-1}$
scaled diag	1cpu	*coo*	333	7	$\alpha\boldsymbol{D}^{-1}$
Schur	1cpu	*coo*	410	9	$\boldsymbol{M}_{K_{ii}}^{-1}$ & $\boldsymbol{M}_{S}^{-1}$
Schur	1cpu	*coo*	58	1	$\boldsymbol{M}_{S}^{-1}$
scaled diag	1gpu	*coo*	326	2	$\alpha\boldsymbol{D}^{-1}$

Table 6.5: Comparison of the computation times of the CPU and GPU architecture using different preconditioning techniques.

However, the GPU computation time does not include the time for transferring the (input and result) data between the GPU und the CPU, which is a relatively slow process. There-

fore, the implementation of an asynchronous memory copying is necessary, for which an extented implementation based on the CUDA framework was used. Exemplary numerical results obtained from such a hybrid technique are given in section 6.5.4. Since different storage schemes may improve the performance of the GPU (as shown in table 6.6), three different matrix storage formats (applied to sparse and dense matrices) were analysed in relation to the performance of the *coo* format. Additionally, for a performant thread-based parallelization on the graphic processing unit, the GPU-based ellpack format (*ell*) and the hybrid storage format (*hyb*) as combination of the ellpack and the coordinate format were taken into account.

architecture	**matrix sparsity**	**matrix format**	**performance (coo)**
1gpu	sparse	*csr*	-12%
1gpu	sparse	*ell*	+16%
1gpu	sparse	*hyb*	+30%
1gpu	dense	*csr*	+24%
1gpu	dense	*ell*	-180%
1gpu	dense	*hyb*	-160%

Table 6.6: Comparison of the performance results of the GPU architecture using different matrix storage formats in relation to the coordinate format (*coo*).

The next step addresses the development of the MPI implementation, which enables a memory-consistent usage of more than one CPU node. By this, creating a high-performance computing framework with a distributed memory processing or a cache-coherent non-uniform memory architecture (ccNUMA), often realized in systems with multiple CPU sockets, is an objective of this work. The performance evaluation of such frameworks, also considering multiple GPU nodes is realized by using verified numerical results from large-scale linear-elastic finite element analysis (FEA) of several numerical specimens, which are benchmarked in the next sections.

6.4 Distributed computing

6.4.1 Hardware architecture

The applied CPU system to all following examples is a four-socket AMD Opteron rack server of type HP ProLiant DL585 G7 with 128 GB of total memory. Here, each socket contains a 12-core Opteron 6174 processor resulting in eight ccNuma nodes with distributed memory access and with a local size of memory of 16 GB RAM for each socket. Furthermore, each core has a clock speed of 2200 MHz with 128 kB L1 and 512 kB L2 cache configuration. The unidirectional interconnection between the ccNuma nodes is the HyperTransport bus standard 3.0 and the operating system is the openSUSE Linux Enterprise Server (x86_64).

6.4.2 Examples: Homogeneous and hybrid meshed three-phase specimens

For the investigation of various types of FE meshes a scalable finite element kernel was implemented, being suitable for distributed computing and tested for the execution on multi-socket servers (HP ProLiant DL585 G7) as well as on the high-performance computing cluster NEC Nehalem at the HLRS. Different finite element discretization techniques were applied (see fig. 6.6). Moreover, a homogeneous aligned mesh (fig. 6.6, top-left) and a combination of a regular (grid-based) mesh with different element sizes (fig. 6.6, top-right) was generated to finally obtain the hybrid mesh with aligned meshed inclusions embedded in a regular grid (fig. 6.6, bottom left and right).

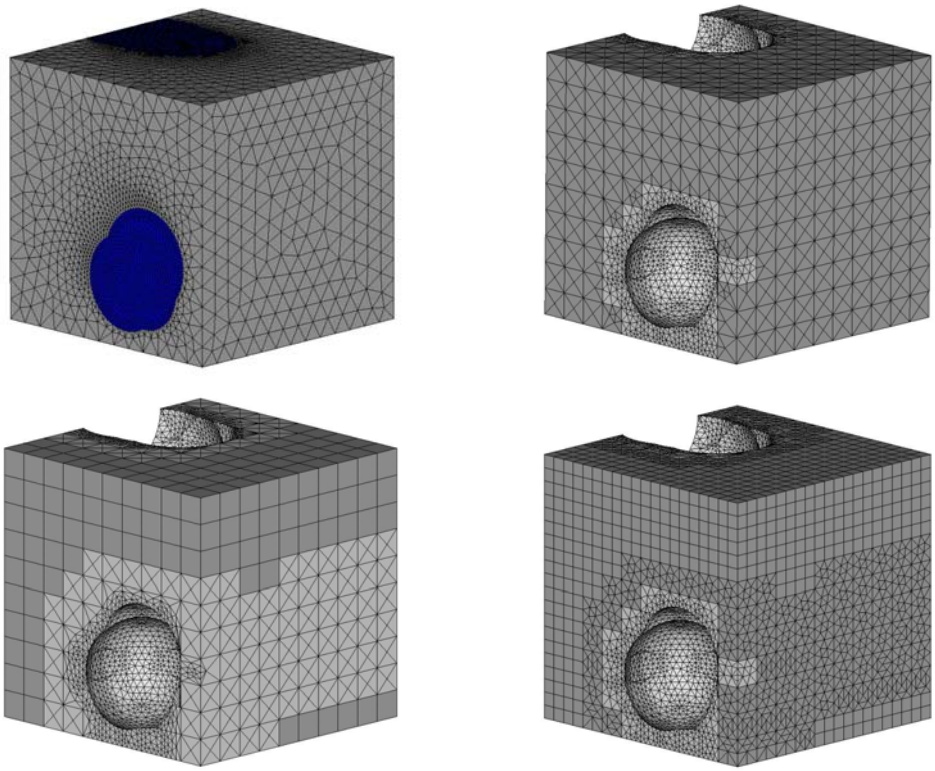


Figure 6.6: Initial heterogeneous specimens and three different meshing techniques (from top-left): Homogeneous aligned mesh, the combined regular and aligned tetrahedral mesh, the hybrid mesh (resulting from the regular grid), the aligned tetrahedral mesh (bottom-left) as well as the refined hybrid mesh.

From the performance point of view the hybrid meshing technique can significantly reduce the computation time caused by the numerical integration of the finite elements, since here a one-time grid-based element integration is sufficient to generate the initial element stiffness matrix as stated in section 3.4.4. This is applicable to all elements of the region within the grid, if mechanical isotropy (with a constant Poisson's ratio) is assumed. This material region or distinct phase is denoted as a material elastic domain based on one element type and similar material properties. After applying the aligned as well as the hybrid meshing technique, the partitioning of the aligned mesh in statically load-balanced

domains was performed by using METIS (as illustrated in figure 6.9). In comparison to a purely tetrahedral mesh, a nodal partitioning for the resulting irregular mesh was applied, as shown in figure 6.8. For all decomposed meshes, the numerical integration was performed as well as the assembly of the decomposed global coefficient matrices and finally, the distributed system was solved by an iterative MPI-based preconditioned conjugate gradient method with the boundary and loading conditions as given in fig. 6.7. Table 6.7 shows the number of tetrahedrons, the number of hexahedrons and the total number of elements at the three different discretization levels: *msh1* (homogeneous), *msh2* and *msh3* (heterogeneous). Table 6.8 shows the number of off-diagonal nodal blocks, the time for creating the nodal compressed row storage of the finite element data and the necessary memory demand for storing the global coefficient matrix when using the nodal block allocation technique. Moreover, the distributed numerical integration and global matrices assembly were executed measuring the respective elapsed computing times required for these tasks. Table 6.9 states the material properties used for the generation of the homogeneous and heterogeneous test specimens. It is to be noted that the loading case is limited to uniaxial tension induced by an unit load of $1N/mm^2$ applied as surface traction of the top.

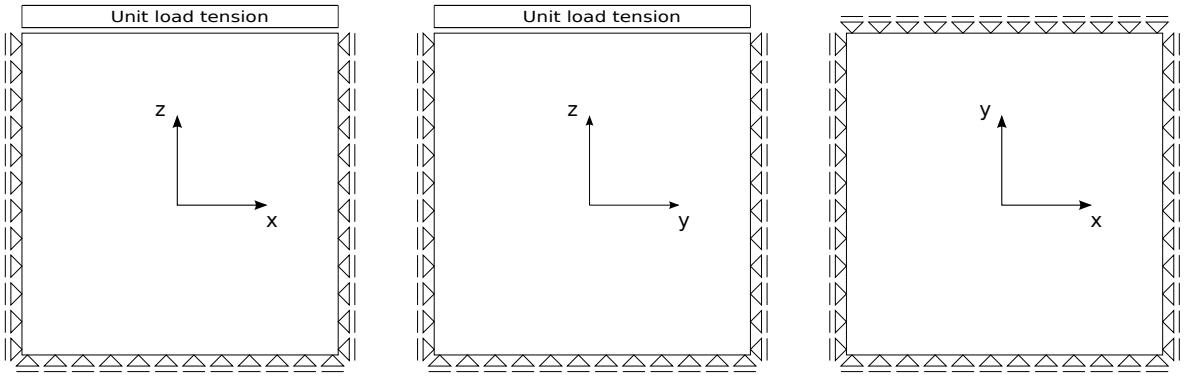


Figure 6.7: Initial boundary and loading conditions applied to all specimens which are investigated.

mesh	tetrahedrons	hexahedrons	total	nodes	d.o.f.s
msh1	2,807,107	-	2,807,107	473,716	1,421,148
msh2	5,126,122	247,000	5,373,122	1,146,016	3,438,048
msh3	38,664,482	247,000	38,911,482	6,904,392	20,713,176

Table 6.7: Size of problems according to homogeneous (*msh1*) and mixed meshes (*msh2* and *msh3*).

6.4.3 Benchmark: Homogeneous mesh

The following numerical results were obtained for the decomposed homogeneous tetrahedral mesh (*msh1*) and are illustrated in figure 6.8. Here, the MPI-parallelized computation of the global stiffness matrix and the parallel solution of the linear equation system with

	mesh 1	**mesh 2**	**mesh 3**
d.o.f.s	1,421,148	3,438,048	20,713,176
nodal ij blocks	3,297,880	9,359,748	49,067,679
sequential time [sec] for			
nodal block allocation	3.0	5.6	39.7
allocated memory [GB]	0.242	0.679	3.599

Table 6.8: Nodal block allocation (sequential): Number of off-diagonal blocks, time for nodal block allocation and allocated memory for the *ndcsr* storage of the final coefficient matrix.

mesh type	**phase**	**property**	**symbol**	**value**	**unit**
homogeneous	-	Young's modulus	E	30000	$[N/mm^2]$
		Poisson's ratio	ν	0.18	[-]
heterogeneous	matrix	Young's modulus	E_{mat}	31000	$[N/mm^2]$
		Poisson's ratio	ν_{mat}	0.18	[-]
	inclusion	Young's modulus	E_{inc}	62500	$[N/mm^2]$
		Poisson's ratio	ν_{inc}	0.18	[-]
	interface	Young's modulus	E_{int}	25000	$[N/mm^2]$
		Poisson's ratio	ν_{int}	0.18	[-]

Table 6.9: Material properties for the homogeneous and heterogeneous mesh applied for the benchmark.

over 1.4 million d.o.f.s were distributed to up to 12 MPI processes. The resulting computing times with respect to the numerical integration and matrix assembly as well as the (ideal) corresponding speed-ups are shown in table 6.10. Furthermore, table 6.10 contains the computing times for the parallel iterative equation solver based on the preconditioned conjugate gradient method, which respects a (residual) error tolerance smaller than 10^{-6}. It is a main disadvantage, if the number of the MPI processes is higher than the number of available ccNuma nodes, since in this case the performance abruptly decreases due to the multiple memory access between different MPI processes (np=12) to the same ccNuma node, mainly being caused by the common L3 cache of each socket.

np	**assembly** ndcrs (sec)	**assembly** speed-up	**solver** ndcrs (sec)	**solver** speed-up	**ansys pcg** csr (sec)	**ansys pcg** speed-up
1	37.3	1.0	77.1	1.0	94.0	1.0
2	18.3	2.0	42.8	1.8	58.8	1.6
4	9.1	4.0	23.1	3.3	33.6	2.8
6	6.0	6.0	16.1	4.8	25.4	3.7
8	4.5	8.0	13.1	5.9	21.9	4.3
12	3.2	11.7	16.0	4.8	-	-

Table 6.10: Absolute time in seconds and the corresponding speed-ups for assembling the finite element data and for solving the global system of equations with the PPCG solver (*msh1*).

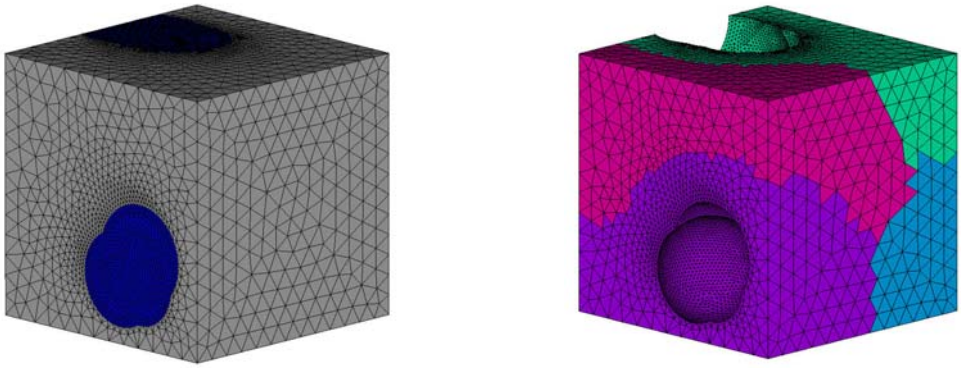


Figure 6.8: Nodal partitioning of an aligned tetrahedral mesh in four nodally equal-sized domains.

np	**format** coo [sec]	**format** csr [sec]	**format** ndcsr [sec]	**ansys** csr [sec]	**speed-up** ndcsr over ansys
1	142.1	143.0	142.6	282.0	2.0
2	74.4	73.9	72.3	149.5	3.9
4	40.2	39.9	39.1	-	6.9
6	30.1	31.0	30.6	-	9.2
8	25.5	25.5	24.2	-	11.7

Table 6.11: Absolute time in seconds for assembling the finite element data and building the global coefficient matrices (*msh2*, 3.4 million d.o.f.s).

6.4.4 Benchmark: Hybrid mesh

For the mixed mesh test the remeshing of the geometry compounds was performed. Here, material homogeneous regions around the cells with embedded inclusions (as a regular grid) were used and an aligned mesh for the remaining volume was considered, resulting in a hybrid mesh (fig. 6.6, bottom). The aligned mesh in this example is partitioned in four nodally equal-sized subdomains, as shown in figure 6.9. Again, the results include the computational times for the numerical integration, the global assembly and the parallelized computation with the PCG method. As previously stated, various storage techniques were investigated and the results based on the *ndcsr* storage were analysed in comparison to the computational time occurring from standard coordinate format (*coo*) and standard compressed sparse row (*csr*, equal to compressed row storage (*crs*)). Table 6.11 states the absolute times for the stiffness assembly including the numerical integration with comparable speed-ups. In table 6.12, the computational times for solving the global equation system with the parallelized PCG method is shown with the corresponding speed-ups. To produce these results the HP ProLiant hardware platform has been used.

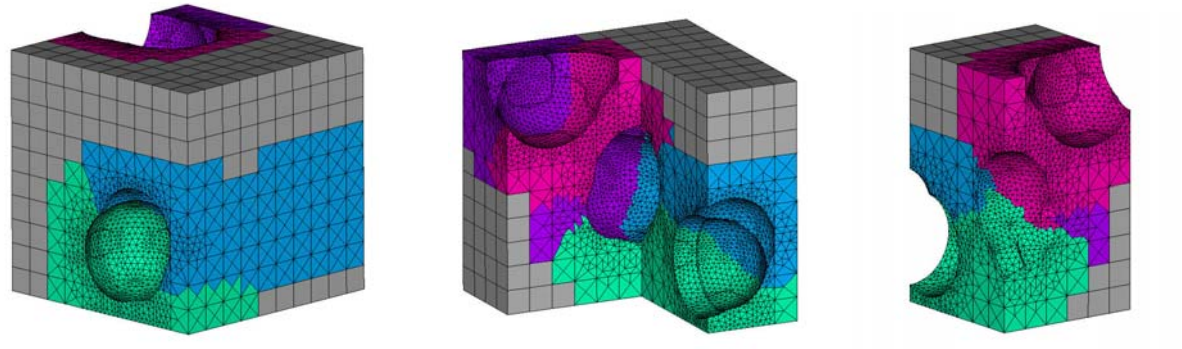


Figure 6.9: Nodal partitioning of a hybrid mesh in four equal-sized domains: Considering the irregular tetrahedral mesh (embedded in a coarse grid) for the load-balanced partitioning (*msh2*): total view (top-left) and two clippings to the inside (top-right, bottom).

np	**format** coo [sec]	**format** csr [sec]	**format** ndcsr [sec]	**ansys** csr [sec]	**speed-up** ndcsr over ansys
1	196.3	164.1	135.5	248.4	1.8
2	112.0	91.2	75.3	193.0	3.3
4	57.4	50.4	40.8	-	6.1
6	38.4	35.5	29.8	-	8.3
8	31.2	28.9	24.2	-	10.3

Table 6.12: Absolute times for solving the global equation system with the parallelized preconditioned conjugate gradient method (*msh2*, 3.4 million d.o.f.s).

6.4.5 Benchmark: Scaling range of the modified Jacobi preconditioning

The distributed solver performance can also be increased by setting an optimized scaling parameter for the preconditioning procedure. This can separately and simultaneously be determined for each subdomain in respect to eq. (5.95) of section 5.9 and an average of a globally suitable value was chosen. This results in an improved preconditioning scaling although the error in the global solution is increasing (in an acceptable manner). The scaling parameters differ when the condition number of the assembled submatrices is changed, mainly being induced by mesh dependencies such as the use of different element types and are influenced by the applied material properties. Nevertheless, the scaling strategy is robust for homogeneous or hybrid meshes. Especially for hybrid meshes (e.g. the discritization of the multiphase material), the influence of different material properties, specified for each phase, as well as the mixture of different element types is relatively low and does not lead to a critical convergence behavior of the proposed solver technique. Table 6.13 shows the convergence behavior and the induced resulting error in the global solution. The global scaling parameters are approximated values (manually as well as computationally), which are obtained from the eigenvalue strategy. A quantitative illustration of the relation between the solution error and the resulting solver speed-up

is given in figure 6.10, where speed-ups are obtained in the 2.0 - 3.0 range accordingly with a corresponding error in the 0.03 - 0.06 range. As stated in table 6.13, the scaling strategy for the preconditioning results in a solver speed-up of 2.22 with an acceptable error of 0.06, respectively. This numerical example corresponds to the example from the previous section, which had been partitioned in eight subdomains and similar boundary and loading conditions according to the previous example were also applied.
It is to be mentioned, that the robustness of this type of scaling strategy was investigated for the uniaxial loading scenario as well as for the case of dead load in the threedimensional space, by means, that the induced solution error in relation to the resulting solver speed-up may differ for biaxial or multiaxial load cases. If the worst case arises, this technique is not such a suitable scaling method for the preconditioning as it is for uniaxial loading conditions and thereby, further investigations as well as numerical modifications may be necessary.

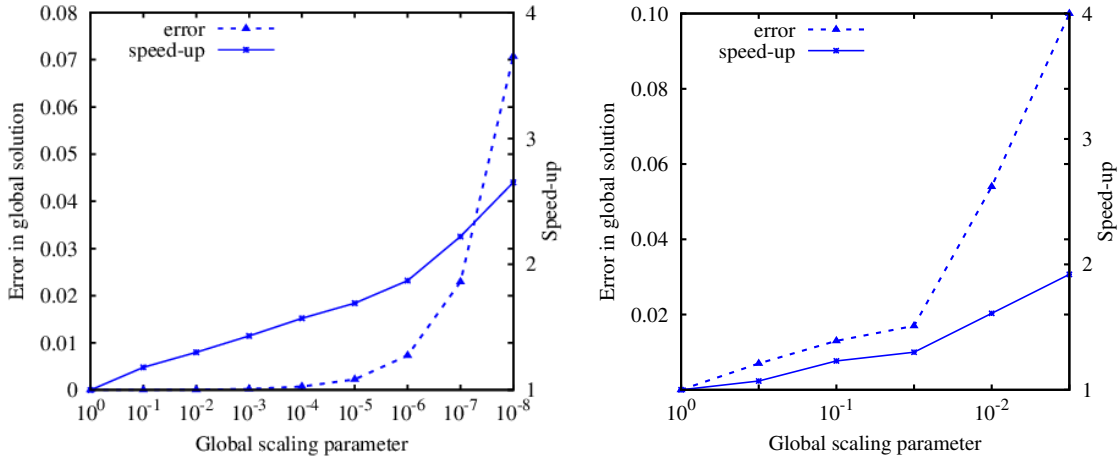


Figure 6.10: Scaling parameters, solution errors and speed-ups for the manually-scaled Jacobi preconditioning considering different load cases: Surface tensile traction (left) and dead (body) load.

scaling parameter $\alpha_L\omega$	**solution error** ϵ_F	**iterations**	**speed-up**	**note**
1.00	0.0	485	1.00	no preconditioning
1.00e-07	0.022923	218	1.87	manually
5.00e-08	0.032267	206	1.87	manually
5.00e-09	0.101021	175	2.65	manually
4.74e-08	0.060387	187	2.22	eigenvalue-based

Table 6.13: Scaling parameter, solution error, number of iterations and speed-up in respect to manually-scaled Jacobi preconditioning and considering a eigenvalue-based computation of $\alpha_L\omega$.

6.4.6 Final remarks

After applying the first numerical tests evaluating the distributed computing model for different finite element based specimens, a framework for high-performance computing was implemented including the full adaption of the MPI-based programming model. Furthermore, a hybrid parallelization technique was implemented respecting the different hardware architectures and also the extension to a hybrid computing model was addressed to enable the use of combined CPU-GPU based architectures. Here, the CUDA programming environment was used, which is accessible for graphics processing units such as the Nvidia Tesla architecture.
The following two sections include the performance measurements and the scaling analysis at two different high-performance clusters. Therefore, a classical finite element analysis in the threedimensional space with respect to linear elasticity was applied to large-scale FE models, built from computer-tomographic scans. The evaluation reviews the computing time for the following numerical tasks

(i) numerical integration and global matrix assembly
(ii) solving the resulting equation system with the PPCG method
(iii) executing the sparse matrix-vector product during the PPCG iterations
(iv) executing of the non-matrix-vector products during the PPCG iterations
(v) communication overhead caused by MPI during the PPCG iterations

The dependencies or the concurrency between the different (MPI-based) processes are given in table 6.14.

task	**type of concurrency**
(i)	simultaneously, independent
(ii	simultaneously, partial dependent
(iii)	simultaneously, fully independent
(iv)	simultaneously, fully independent
(v)	simultaneously, fully dependent

Table 6.14: Type of concurrency of the numerical tasks.

The NEC Nehalem cluster at the high-performance computing center Stuttgart (HLRS) with its Intel Xeon architecture was used, since it provides the architectural features necessary for this type of hybrid computing. Additionally, the relatively new Cray XE6 cluster (AMD Interlagos CPU architecture) was used. The technical details of the equipment and the numerical tests for both systems are given in the following sections.

6.5 HPC framework 1: NEC Nehalem cluster

6.5.1 Hybrid (CPU-GPU) NEC Nehalem cluster at HLRS

The NEC Nehalem cluster at the high-performance computing center Stuttgart (fig. 6.11) consists of several frontend nodes for interactive access as well as several computing nodes for the execution of parallelized numerical applications. Before the utilization of this cluster in 2010, it was placed at 77th position of the TOP 500 list in June 2009 with a peak performance of 62 TFlops. The operating system is based on Scientific Linux SL (release 5.3). For the execution of the binary code the *qsub* batch system is used, enabling several options of the parallel computation to be chosen individually. As a special characteristic, the HPC system is equipped with 32 GPU nodes based on the Nvidia Tesla architecture (fig. 6.11, right) for the extended acceleration of purely CPU-based computations. The

Figure 6.11: Cabinets of the NEC Nehalem cluster at the High-Performance Computing Center Stuttgart (left) equipped with 32 of Tesla GPU S1070 1U rack system (right).

main hardware features of the NEC Nehalem cluster [HLRS 2011] are given in table 6.15.

Peak Performance	62 TFlops
Number of Nodes	700 Dual Sockel Quad Core
Processor	Intel Xeon (X5560) Nehalem, 2.8 GHz, 8MB Cache
Memory/node	12 GB / 24 GB / 48 GB / 144 GB
Disk	80 TB shared scratch (lustre)
Interconnection	Infiniband, GigE
Accelerators	32 nodes Nvidia Tesla S1070 GPGPU

Table 6.15: Technical description of the hardware features of NEC Nehalem cluster at HLRS Stuttgart.

6.5.2 Benchmark: 3D poriferous bone specimen

Another numerical analysis of the solver scaling was investigated at the NEC Nehalem cluster, respecting a large-scale 3D microstructural bone specimen ([Perilli et al. 2003], fig. 6.12). This FE model is characterized by a voxel discretization resulting in several million degrees of freedom. Due to the highly poriferous material, this numerical example leads to an inappropriate matrix structure of the global stiffness matrix (in respect to matrix bandwidth and condition number) compared to the previous examples.

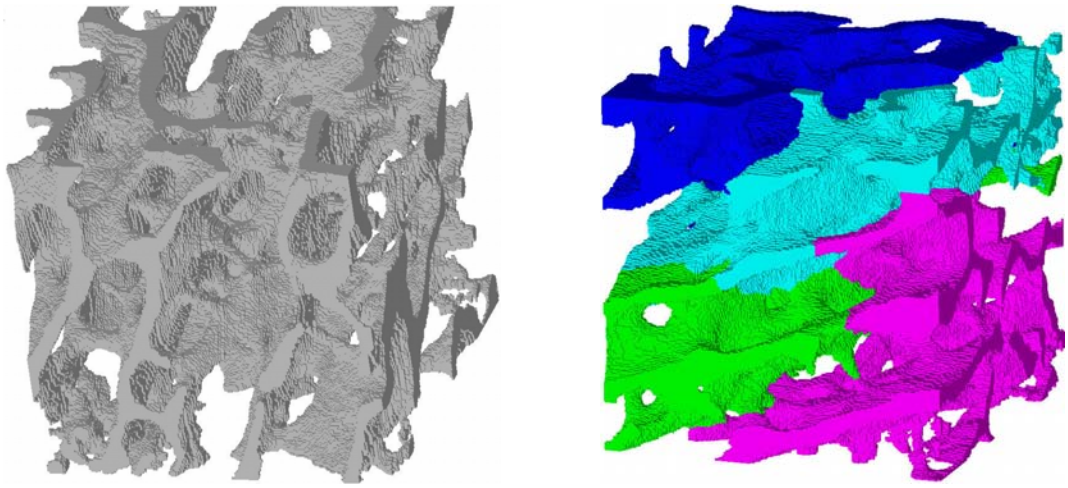


Figure 6.12: 3D poriferous bone specimen as FE structure based on voxel data (perspective view) and the load-balanced decomposed FE structure in four nodally equal-sized domains (8.9 million d.o.f.s).

elements	**number of nodes**	**d.o.f.s**	**nodal FE blocks**	**memory** [GB]
2,434,993	2,987,933	8,963,799	36,824,835	2.403

Table 6.16: Dimensions of the FE problem: Number of elements, number of nodes, number of d.o.f.s, number of nodal FE blocks and memory demand for matrix storage in gigabyte.

In table 6.16, the dimension of the problem such as the number of elements, the number of nodes, the number of resulting d.o.f.s and the number of nodal FE blocks as well as the memory demand are given.

domain j	**coupled nodes**	**number of nodes**	**nodal FE blocks**	**memory** [GB]	**load balance**
1	5,108	488,783	5,418,438	0.547	1.002
2	5,896	487,927	5,548,558	0.546	1.000
3	4,399	489,546	5,545,580	0.554	1.003
4	3,198	488,472	5,570,790	0.550	1.001

Table 6.17: Dimension of the decomposed FE problem: Number of coupled nodes, number of nodes, number of off-diagonal nodal FE blocks and memory demand for the matrix storage in gigabyte and the relative load imbalance.

Here, the voxel model was converted to a regular grid consisting of hexahedral elements with linear shape functions. Furthermore, the decomposition of the FE (voxel) model was done by applying a nodal partitioning of the hexahedral mesh by using METIS. In figure 6.12 (right) the mesh partitioning in four nodally equal-sized partitioned subdomains is illustrated. Moreover, table 6.17 gives quantitative results of the mesh partitioning, including the number of nodes, of coupled nodes and of nodal FE blocks and also the memory demand as well as the load balance factor for each of the four subdomains. For the numerical integration of the finite elements respecting their element stiffness matrix,

different numerical integration techniques were applied yielding to different computational efficiencies. Due to this, the tests were performed and then compared regarding their (elapsed) computational time and their resulting speed-ups considering one and four MPI processes (np), respectively. The influence on the computational performance is stated in table 6.18, where the time (in seconds) for the numerical integration of the finite elements and the time for the sequential assembly of the global stiffness matrix (np=1) or the simultaneous assembly of the global stiffness submatrices (np=4) is compared. The integration techniques involved are eight or six Gauss points, one Gauss point (reduced integration with hourglass stabilization, HG), and the voxel integration technique. Thereby, it is assumed that all voxels correspond to the same resulting element stiffness matrix and thereby, the numerical integration needs to be executed only once.

integration type	**Gauss points**	**time / np=1** [sec]	**time / np=4** [sec]	**speed-up** [-]	**speed-up** over full integr.
full	$ip = 8$	236	61	3.86	3.86
special	$ip = 6$	202	52	3.88	4.54
reduced	$ip = 1$ (+ HG)	157	41	3.82	5.76
voxel	$ip = 8$	78	20	3.90	11.8

Table 6.18: Integration time in seconds with eight and six Gauss points, using one Gauss point for reduced integration with hourglass stabilization (HG) and the voxel integration technique for one and four MPI processes and the resulting speed-ups (including the time for the global matrix assembly).

The linear-elastic FEA considers the boundary and loading conditions of the previous benchmarks corresponding to fig. 6.7. The measurements which were done during the numerical tests include the times for

- building the element stiffness matrices and the assembly of the global stiffness (domain) matrices
- building the preconditioning matrix including the eigenvalue-scaling strategy
- the distributed solving of the global equation system with the PPCG method
- scaling up to 64 MPI processes (equivalent to the number of subdomains).

with results being analysed in detail considering a) multiple CPU nodes and b) multiple CPU-GPU nodes in the following two subsections, also published in [Schrader et al. 2013b].

6.5.3 Benchmark: Multiple CPU nodes

This benchmark was used to evaluate the poriferous bone specimen taking multiple CPU nodes into account. Here, each MPI process uses one cache coherent NUMA node (ccNUMA), with two ccNUMA nodes per CPU socket. Consequently, the dual socket quad-core system as one CPU node provides four ccNUMA nodes.

In figure 6.13, the scaling of the computational time which is needed for the assembly of the global matrix is illustrated. This task can be performed independently for each subdomain, when the load-balanced partitioning is ensured. In all cases the measured time includes the time for the (full) numerical integration of the finite elements as well as the time to build the global matrix (sequential case) or the global submatrices (parallel case). The scaling is performed with 1 up to 128 MPI processes and in table 6.19 and 6.20 the absolute values of the computing time as well as the resulting speed-ups are given for the three matrix storage formats (*coo*, *csr*, *ndcsr*).
Figure 6.14 states the total computing required time in relation to the number of MPI processes (equal number of subdomains or ccNUMA nodes) used, being necessary for solving the global equation system with the parallelized PCG method. The solver time is analysed in respect to the cumulated time which is required for the matrix-vector operations, for the non-matrix-vector operations as well as the elapsed time caused by the MPI communication overhead cumulated during all PPCG iterations. Furthermore, in figure 6.15 the scaling of the computational time considering the computation of the sparse matrix-vector products during the PPCG iterations is illustrated, which is performed simultaneously as an independent task and does thereby not have a negative influence on the performance of the solver. Figure 6.16 illustrates the scaling of the iterative solver, if the time for the computation of the sparse matrix-vector product in each iteration is excluded. Here, the scaling considering the *ndcsr* matrix storage format is equivalent to the scaling of the computing time of the matrix-vector product with equal computational speed-ups (without any loss of performance). Finally, in figure 6.17, the computing time of the MPI communication between the different MPI processes is shown, mainly induced by the $MPI\_Allreduce$ operation during the PPCG iteration. By increasing the number of subdomains the scaling behavior of the PPCG solver (with respect to fig. 6.14) is mostly influenced by a moderately increase in the overall computing time caused by the MPI communication.

matrix	**np=1**	**np=2**	**np=4**	**np=8**	**np=16**	**np=32**	**np=64**	**np=128**
coo	422.4	239.6	107.8	60.0	31.1	15.8	7.2	-
csr	485.0	242.5	121.2	60.5	27.6	14.3	7.2	-
ndcsr	416.4	236.0	106.3	55.9	27.2	14.2	6.9	3.7

Table 6.19: NEC Nehalem cluster (CPU): Quantitative values of total computational time for the parallel assembly of the global stiffness matrices (including the numerical integration).

6.5.4 Benchmark: Hybrid multiple CPU-GPU nodes

This benchmark considers the implementation of a hybrid parallelization technique for the PPCG method combining the CPU and GPU and by this, being suitable for up to 16 Tesla GPU nodes at the NEC Nehalem cluster. Here, each MPI process has access

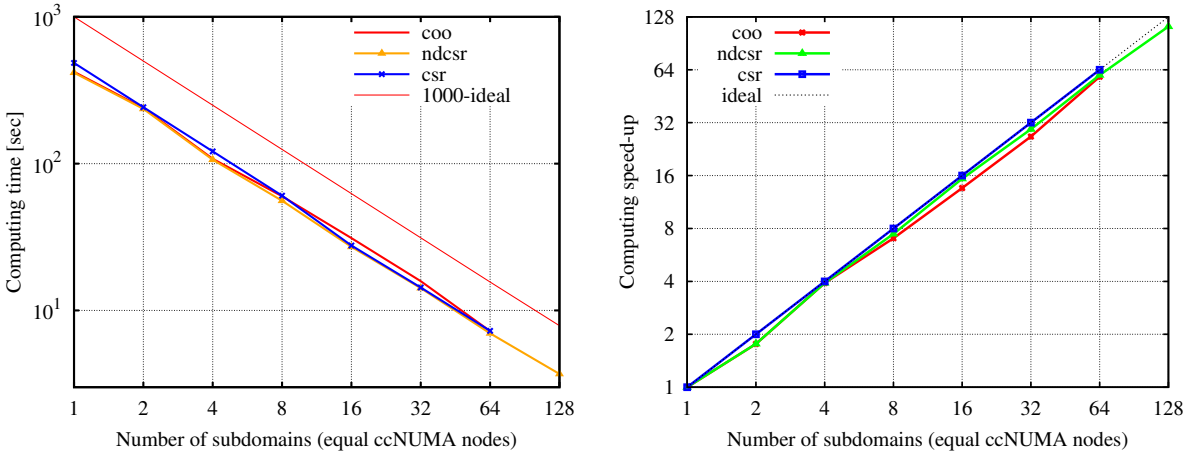


Figure 6.13: NEC Nehalem cluster (CPU): Total computational time for the parallel assembly of global stiffness matrices (including the numerical integration) with increasing number of subdomains (8.9 million d.o.f.s) and the resulting speed-ups (right).

matrix	np=1	np=2	np=4	np=8	np=16	np=32	np=64	np=128
coo	1.0	1.76	3.92	7.04	13.58	26.72	58.64	-
csr	1.0	2.00	4.00	8.02	17.57	33.91	67.36	-
ndcsr	1.0	1.76	3.92	7.45	15.31	29.32	60.35	112.54

Table 6.20: NEC Nehalem cluster (CPU): Resulting speed-ups for the parallel assembly of global stiffness matrices (including numerical integration, 8.9 million d.o.f.s).

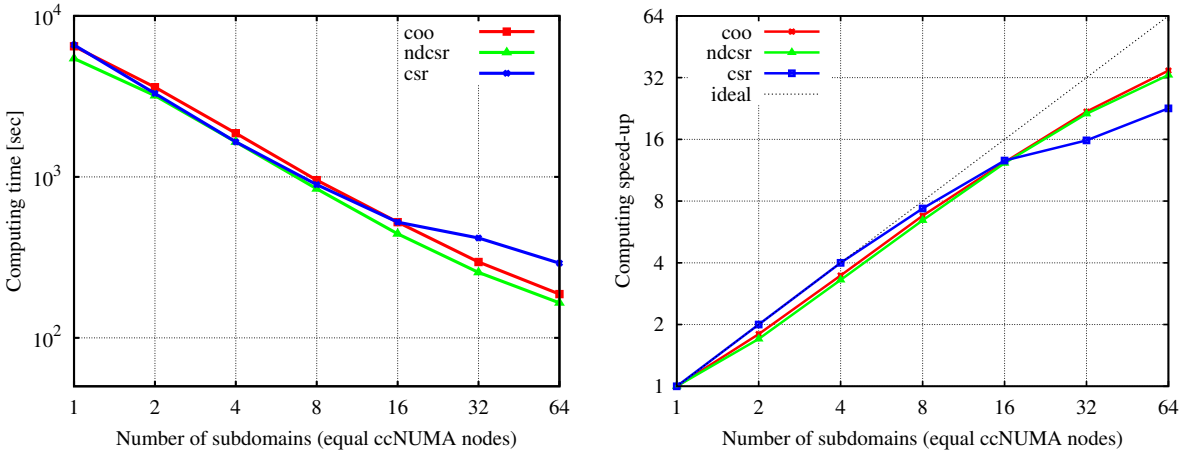


Figure 6.14: NEC Nehalem cluster (CPU): Total computational time for the parallelized preconditioned conjugate gradient method (8.9 million d.o.f.s) and the resulting speed-ups (right).

to one GPU-based subsystem, namely the Nvidia Tesla S1070 GPU. The numerical task to improve the performance results in the outsource of the computation of the sparse matrix-vector product to the GPU, consuming almost 90 percent of the time required for one CPU-based PPCG iteration. Generally, the allocation and execution of the sparse matrix-vector products of the PPCG solver is performed by one Tesla unit. Therefore, the

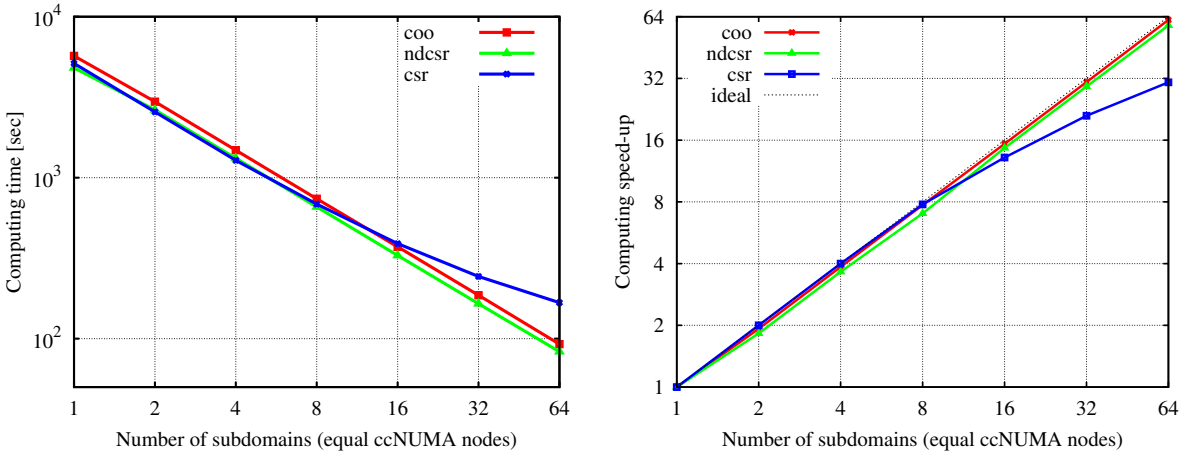


Figure 6.15: NEC Nehalem cluster (CPU): Accumulated time for sparse matrix-vector operations of the PPCG method (8.9 million d.o.f.s) and the resulting speed-ups (right).

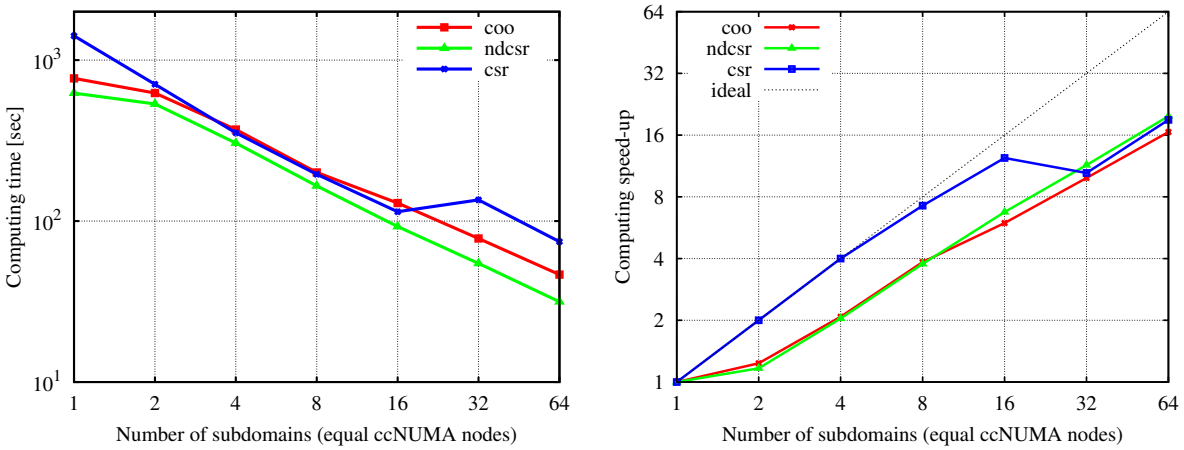


Figure 6.16: NEC Nehalem cluster (CPU): Scaling of accumulated computational times for non-matrix-vector operations of the PPCG method (8.9 million d.o.f.s) and the resulting speed-ups (right).

global submatrices are initially allocated by the GPU considering two vectors: one for the input and one for the result data. For the matrix storage of the assembled and distributed stiffness matrix per subdomain or MPI process, the *coo* storage format is applied on the corresponding GPU unit. The subdomain matrix is then distributed along the maximum number of available GPU threads per GPU, which enables a simultaneous sparse matrix-vector multiplication per matrix block and GPU thread. The results from the hybrid model are then compared to the computational times for the *spmv* operation on the CPU nodes, which is illustrated in figure 6.18, where the different storage formats investigated for CPU-based computation are taken into account. In figure 6.19, an improved data transfer of the nodal result vectors between CPU and GPU was realized and was compared to two other data transfer techniques from and to the host memory. Here, the computational time of the hybrid model includes the time for the memory transfer from

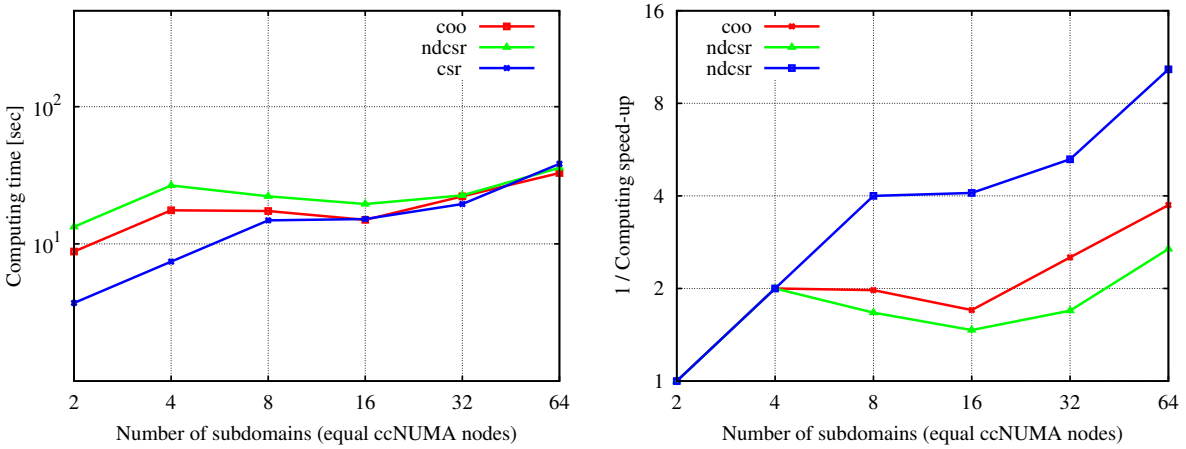


Figure 6.17: NEC Nehalem cluster (CPU): Scaling of accumulated computational times for MPI based communication of the PPCG method (8.9 million d.o.f.s) and the resulting speed-ups (right).

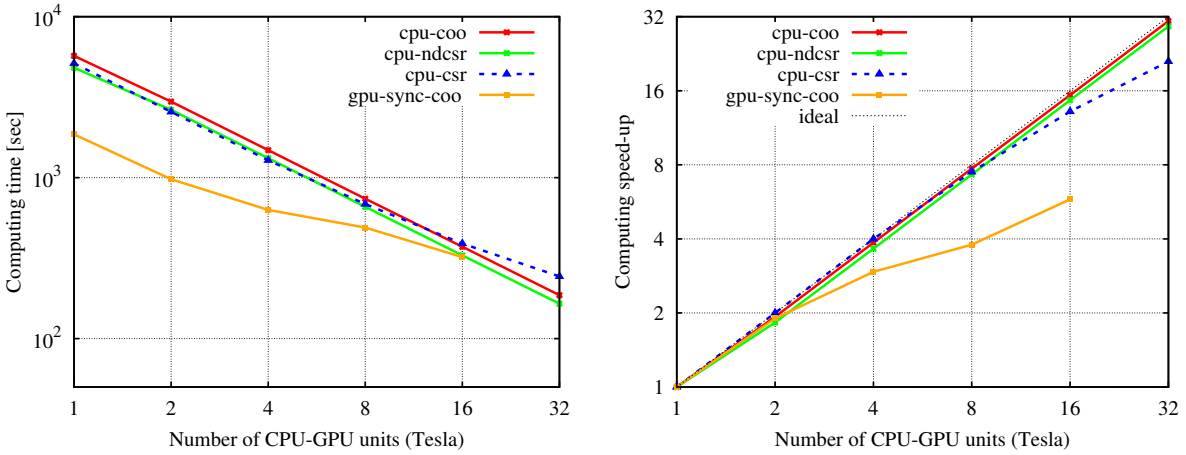


Figure 6.18: NEC Nehalem cluster (CPU-GPU): Accumulated computational times for sparse matrix-vector operations of the PPCG method using the CPU-only and the hybrid CPU-GPU cluster, respectively (8.9 million d.o.f.s) and the resulting speed-ups (right).

the CPU host to the GPU device and vice versa, before and after each *spmv* computation per subdomain. Moreover, the time for synchronous (hybrid-sync), asynchronous (hybrid-async) and mapped memory (hybrid-mapped) data transfer techniques are compared to the computing time required for the CPU-only *spmv* computation using the *coo* matrix storage format (fig. 6.19). The hybrid implementation was realized with the CUDA environment, release 3.1., and combined with the C/MPI programming framework. Results of the linear-elastic FE analysis considering the scalable PCG solver are illustrated (cut view): the initial FE model and the deformation state (fig. 6.20), the displacement state (fig. 6.21) and the strain and stress states for the uniaxial tension loading case (fig. 6.22), which were validated with different commercial and non-commercial FE code ([ANSYS 2012], [Bucher et al. 2007]).

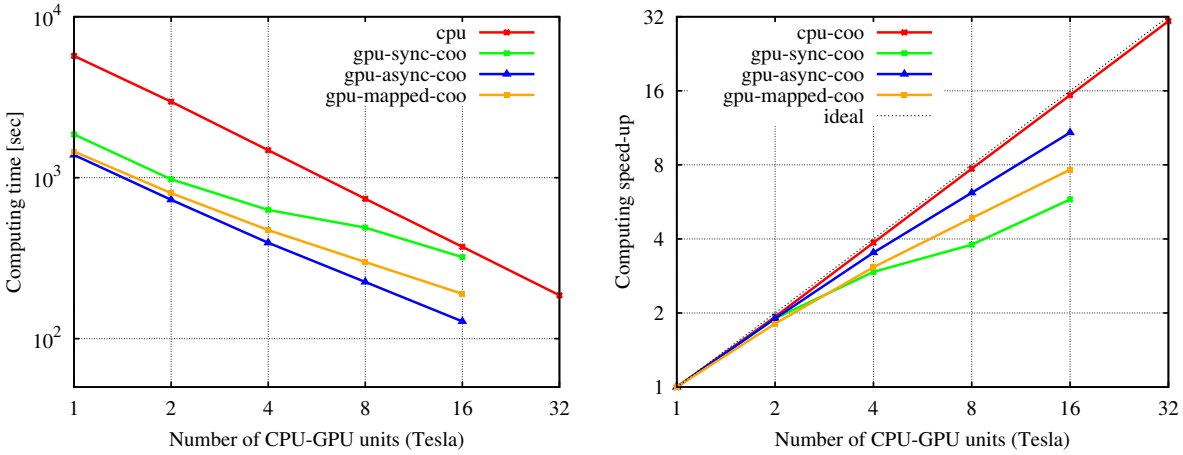


Figure 6.19: NEC Nehalem cluster (CPU-GPU): Accumulated computational times for sparse matrix-vector operations of the PPCG method using the CPU-only and hybrid CPU-GPU cluster with synchronous (hybrid-sync), asynchronous (hybrid-async) and mapped memory (hybrid-mapped) CPU-GPU data transfer for the *coo* matrix storage format and the resulting speed-ups (right).

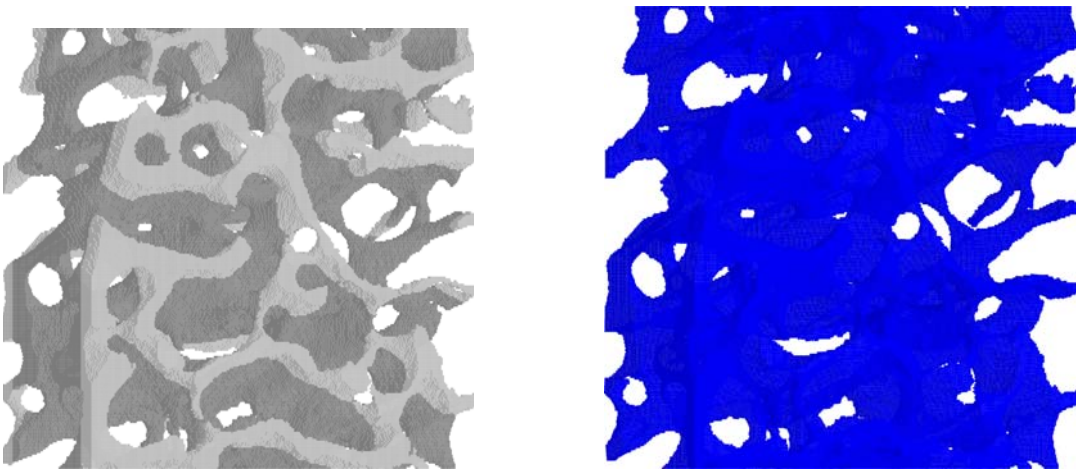


Figure 6.20: NEC Nehalem cluster: Initial FE model und deformation state (right), cut view (8.9 million d.o.f.s).

6.6 HPC framework 2: CRAY XE6 cluster

6.6.1 CRAY XE6 cluster at HLRS

The second high-performance computing framework was tested at the Cray XE6 cluster 'Hermit' (fig. 6.23) at the HLRS, which is in production mode since the beginning of the year 2012. With a computational power of over 3,552 compute nodes (where each XE6 node consists of one AMD Opteron 6276 Interlagos), it is possible to scale parallelized applications up to several ten thousand cores with a total peak performance of nearly one petaflop. Here, the AMD Interlagos processor is a 32-core x86-64 architecture, a composition of a number of 'bulldozer' core modules. It was introduced in 2011 and enables the Cray XE6 cluster to be the fastest system in Europe (at the beginning of the

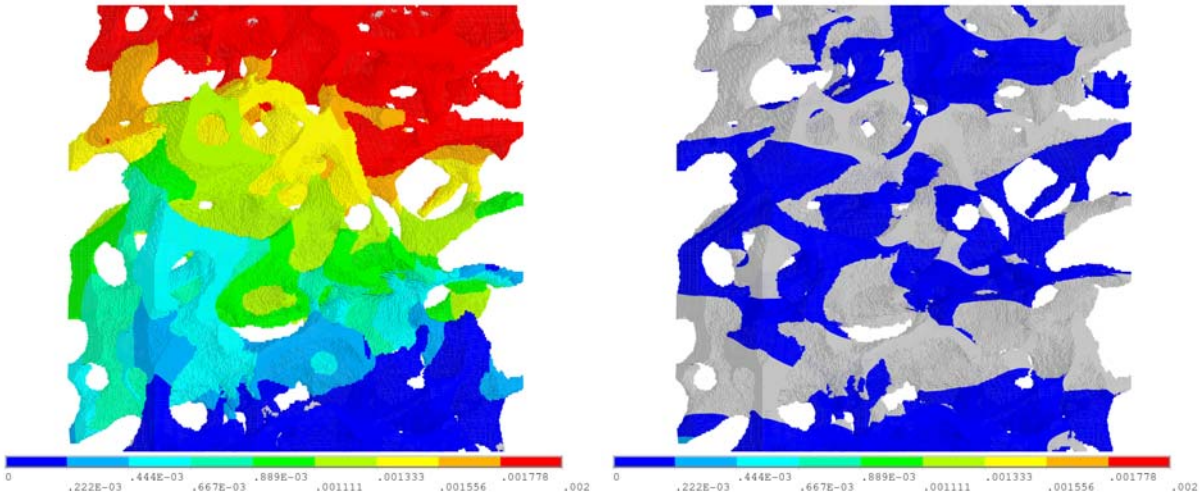


Figure 6.21: NEC Nehalem cluster: Vertical and horizontal (right) displacement state in uniaxial tension case, cut view (8.9 million d.o.f.s).

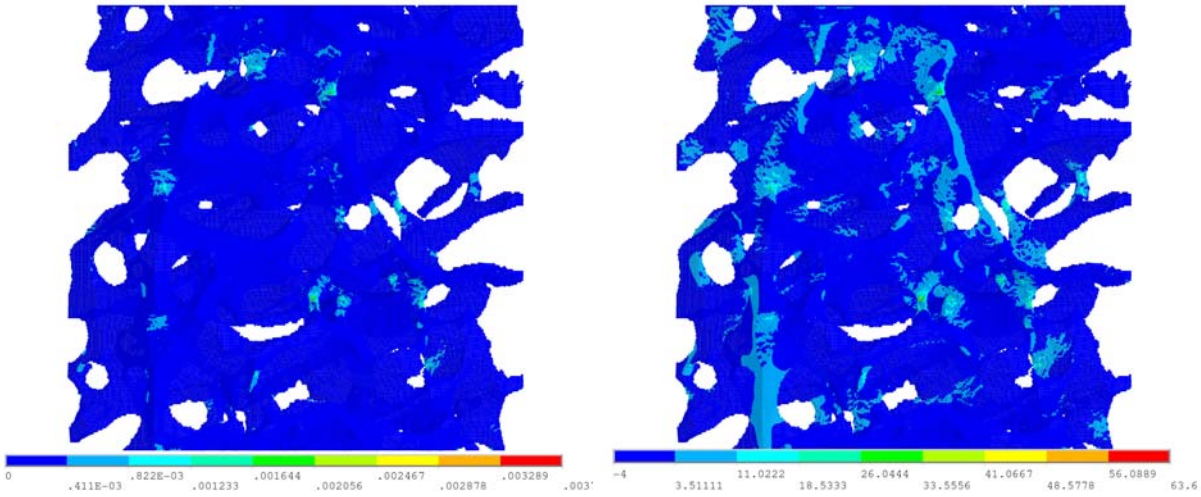


Figure 6.22: NEC Nehalem cluster: First principle strain und stress state (right) in uniaxial tension case, cut view (8.9 million d.o.f.s).

year 2012). The main hardware features of the cluster [HLRS 2012], are listed in table

Figure 6.23: 38 cabinets with 96 compute dual socket nodes of the Cray XE6 cluster at HLRS.

6.21 below.

Peak performance	around 1.045 PFlops
Number of nodes	3552 compute nodes
Number of cores	113,664
Processor node	Dual socket AMD Interlagos, 2.3 GHz, 16-core each
Memory/node	32 GB / 64 GB
Disk capacity	2.7 PB (lustre)
Interconnection	Cray Infiniband
Accelerators	Special pre/post-processing nodes

Table 6.21: Technical description of the Cray XE6 cluster 'Hermit' at HLRS.

6.6.2 HPC Cray XE6 cluster batch system

The job launch of a batch application can be done by using the *qsub* job submission command. Moreover, to run the batch application, the ALPS (application level placement scheduler) uses *aprun* in conjuction with some specified arguments listed in one line of the specified batch file. In combination with the *Torque* header and the *aprun* command the characteristics of the setted job environment can be defined by *Torque* options within the batch file. In table 6.22 some of the *Torque* options are listed.

Torque options	**description**
-l mppwidth=<PE-value>	the number of MPI processes or processing elements (PE)
-l mppdepth=<OMP-value>	the number of (OpenMP-) threads per PE
-l mppnppn=<PN-value>	the number of PEs per node
-l mem=<memory-value>	as the number of memory per node (32GB or 64GB)
-l walltime=<hh:mm:ss>	as the maximum elapsed time of the job

Table 6.22: *Torque* header options used in the batch file to launch and run batch jobs at the CRAY XE6 cluster.

6.6.3 Benchmark: 3D large-scale casted nickel-alloy specimen

The second benchmark, performed at the Cray XE6 cluster, evaluates an example which is based on a 3D large-scale voxel discretization of a casted nickel-alloy specimen with visual analytics of geometry characteristics, as given in figure 6.24. Here, the total FE model results in more than 160 million nodal d.o.f.s. Considering the biaxial symmetry of the geometry, the exported FE mesh is equivalent to one quarter of the total specimen (as illustrated in figure 6.24) and was solved by the implemented PPCG solver. The numerical effort needed for the generation of the FE model is shown in table 6.23 and 6.24. Comparing the matrix storage formats *coo* and *ndcsr* in regards to their memory demand (as illustrated in table 6.24), the *ndcsr* storage format decreases the memory demand by

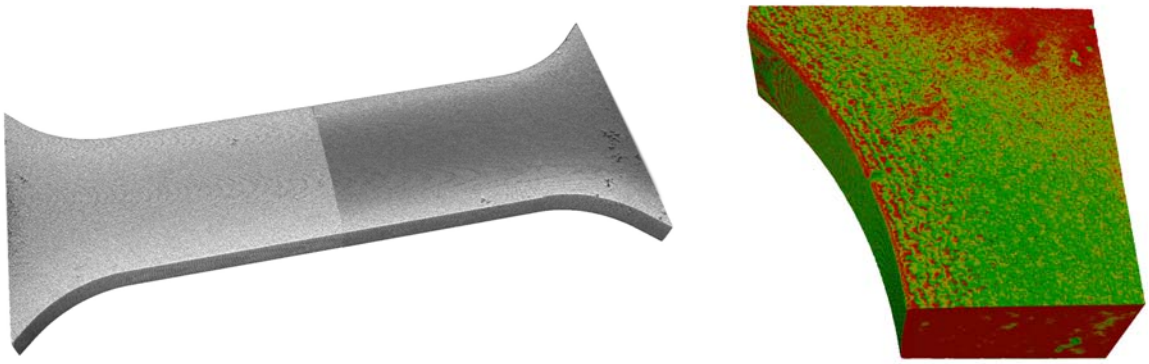


Figure 6.24: Nickel alloy specimen geometry based on computer-tomographic scans: Total view (left), and cut view with visualization of micropores (right).

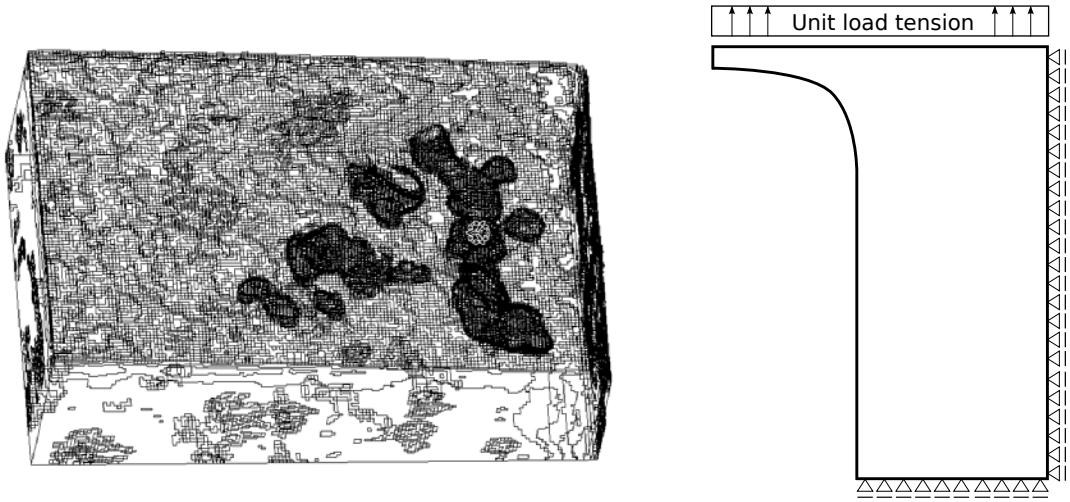


Figure 6.25: Nickel alloy specimen based on computer-tomographic scans: Irregular pores (left) and boundary and loading conditions.

element type	elements total	FE nodes total	d.o.f.s total
linear hexhedron	14,093,177	14,292,274	42,876,822

Table 6.23: Cray XE6 cluster: Element type, the total number of elements and FE nodes as well as the total number of global d.o.f.s.

nearly 18 percent. This is necessary for storing the distributed FE data which are allocated by an equal number of FE blocks storing more than one billion matrix entries. The computational time for the numerical integration and the assembly (measured in the sequential case) is given with around 3,000 seconds for both storage formats. Additionally, in table 6.25, the time needed for the distribution of the data among all MPI processes, the FE nodal block allocation, as well as the preconditioning are determined considering 2 or 256 subdomains (in respect to the number of MPI processes, np). Moreover, the table gives a performance indication regarding the building of the preconditioning matrix for each of the subdomains, including the computations resulting from the eigenvalue scaling strategy. In table 6.26 the computing time for building the preconditioning matrix for up

matrix storage	FE blocks total	block entries total	memory (GB) [GB]	seq. assembly [sec]
coo	184,005,692	1,061,464,238	17.095	3,079
ndcsr	184,005,692	1,061,464,238	12.977	3,001

Table 6.24: Cray XE6 cluster: Quantitative values of the total (three by three) FE blocks, the total number of block entries, the memory demand and assembly time for the sequential case (42.8 million d.o.f.s).

storage	np	block allocation [sec]	distribution [sec]	eigenvalue scaling [sec]	preconditioning [sec]
coo	2	19.7	2.0	21.4	4.9
ndcsr	2	19.6	2.0	17.4	4.7
coo	256	2.8	366.8	23.8	5.7
ndcsr	256	2.9	366.4	23.9	5.7

Table 6.25: Cray XE6 cluster: Computing time (sec) for the nodal block allocation, the distribution of FE data, the modified Jacobi-point preconditioning with the eigenvalue scaling strategy for 2 and 256 subdomains, respectively (42.8 million d.o.f.s).

time of	np=1	np=2	np=4	np=8	np=16	np=32	np=64	np=128	np=256
eigenvalue scaling	19.56	17.35	18.66	19.59	21.13	21.66	22.73	23.38	23.92
precond. matrix	4.70	4.73	4.78	5.40	5.48	5.57	5.77	5.72	5.78
divergency	1.00	0.88	0.95	1.00	1.08	1.11	1.16	1.19	1.22

Table 6.26: Cray XE6 cluster: Quantitative values of total computational time for the decomposed preconditioning matrix and resulting divergency in time with increasing the MPI processes to 256 (42.8 million d.o.f.s).

$\mathbf{np}_i$	$\boldsymbol{\alpha_L\omega}$	divergency $\mathbf{np}_{i-1}$	iterations	speed-up $\mathbf{np}_{i-1}$
1	6.1283e-09	0.000	1425	1.00
2	7.3572e-09	0.200	1434	1.99
4	8.4492e-09	0.148	1446	1.96
8	8.9042e-09	0.054	1447	1.83
16	9.8197e-09	0.103	1451	1.95
32	1.0326e-08	0.052	1454	1.99
64	1.0773e-08	0.043	1456	1.69.
128	1.1126e-08	0.033	1457	1.71
256	1.1461e-08	0.030	1463	1.48

Table 6.27: Number of MPI processes, the scaling parameters for the preconditioning, the number of PPCG iterations and the speed-ups in respect to the modified Jacobi-point preconditioning.

to 256 MPI processes is shown, separated by the numerical operations of the eigenvalue scaling and of the matrix inversion. Due to the allocation of one global vector considering the entries of the global main diagonal only, it is not expected to achieve a time reduction by increasing the number of MPI processes, because of the equal number of numerical op-

erations running. The overall computing time is moderately increasing, which is induced by the MPI communication for data exchange of vector entries at the domain boundaries. Nevertheless, the computing time is comparably low, e.g. in regards to the computing time required for the sequential matrix assembly (see tab. 6.24) or to the time required for solving the global equation system. Therefore, an optimized implementation was not a main task during this work, but the usage of a compressed vector (with only non-zero vector entries) for each subdomain will finally lead to a parallelization of this type of preconditioning technique among all MPI processes. The robustness of the preconditioning technique in uniaxial load cases is shown in table 6.27, where the computed scaling parameter $\alpha_L\omega$ has a maximum divergence of 20 percent with a nearly constant number of iterations and more importantly with an equal error in the global vector of nodal solution. The measured speed-ups of the PPCG solver applied with the modified Jacobi preconditioning are in the 1.83 - 1.99 range, if the number of MPI processes is doubled and is finally addressing up to 64 MPI processes. The speed-up decreases mainly due to the increased MPI communication, which is also illustrated in figure 6.30, and indicates some further optimization of the data exchange regarding the domain boundaries.
The scaling starts with one single node up to 256 CPU nodes at the Cray XE6 cluster, which is building the distributed FE model and is solving it by using the PPCG method with an accuracy as break criterion of 10^{-6}. For the evaluation of the performance, the times for the assembly of the stiffness matrices (including the numerical integration of finite elements) are illustrated in figure 6.26. The times for the PPCG solver (fig. 6.27) were analysed in respect to the cumulated times for the matrix-vector products (fig. 6.28), as well as the times for the non-matrix-vector products (fig. 6.29) and the communication overhead caused by the MPI itself (fig. 6.30). In all diagrams the *ndcsr* and *coo* matrix storage format were compared.

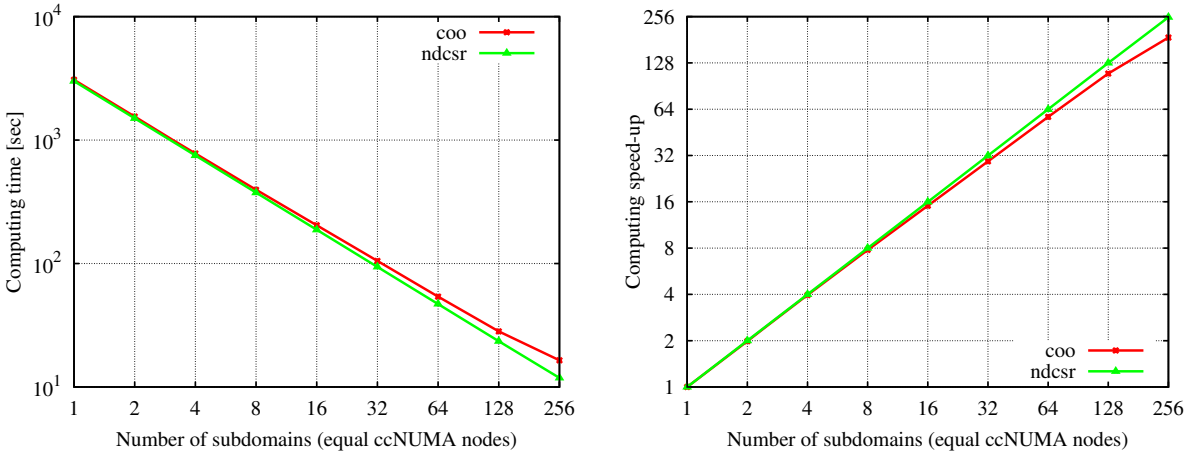


Figure 6.26: Cray XE6 cluster: Total computational time for the parallel assembly of global stiffness matrices (including the numerical integration) with increasing number of subdomains (42.8 million d.o.f.s) and the resulting speed-ups (right).

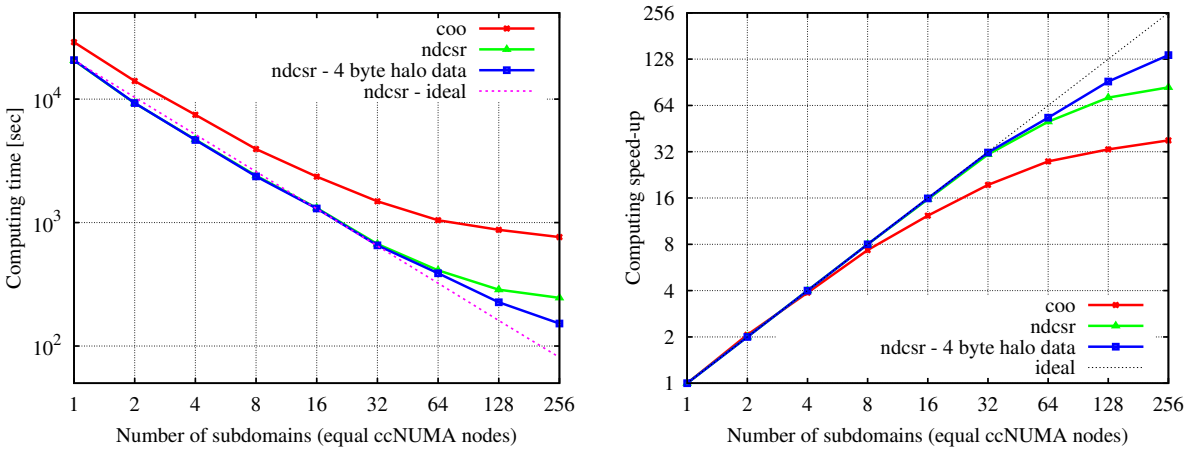


Figure 6.27: Cray XE6 cluster: Total computational time in respect to the parallelized preconditioned conjugate gradient method (42.8 million d.o.f.s) and the resulting speed-ups (right).

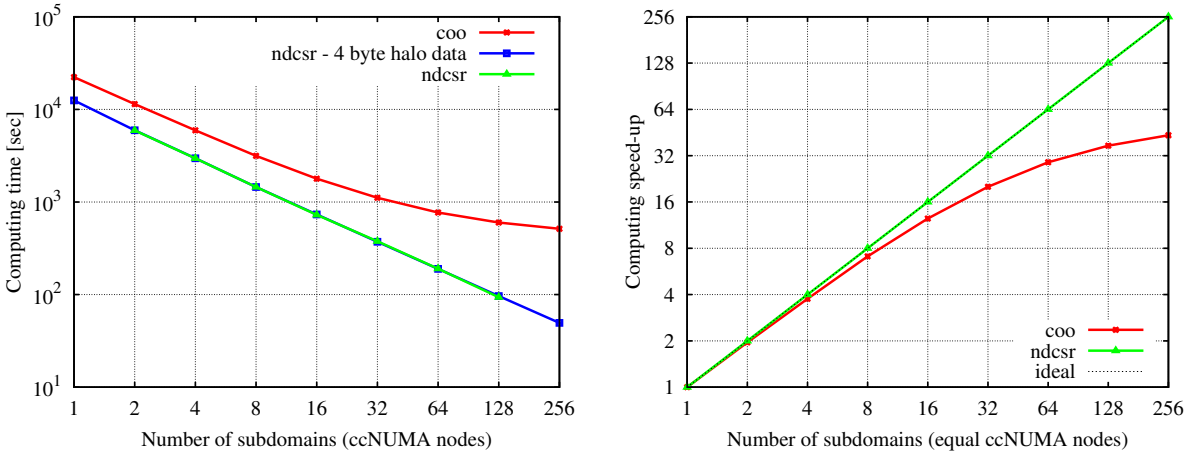


Figure 6.28: Cray XE6 cluster: Accumulated time for sparse matrix-vector operations of the PPCG method (42.8 million d.o.f.s) and the resulting speed-ups (right).

Due to an increased MPI communication between the MPI processes, especially induced by nodal d.o.f.s of the boundaries of connected subdomains, the modified data transfer was implemented by switching the data type of the corresponding *halo* vectors to *single precision*. The modified PPCG solver converged to the exact solution of the original implementation in respect to the memory-efficient *ndcsr* storage scheme. By this approach, the scalability of the PPCG solver can be improved. This is illustrated in figure 6.27, 6.28 as well as in 6.30, and is significantly better if more than 64 subdomains are used. Further improvements can be obtained by the consideration of compressed *halo* vectors. Also, the AMD Interlagos processor of the Cray XE6 cluster contains the new AVX instruction set which can be activated by using the GNU compiler environment [GNU 2011]. Specific details are shown in table 6.28.

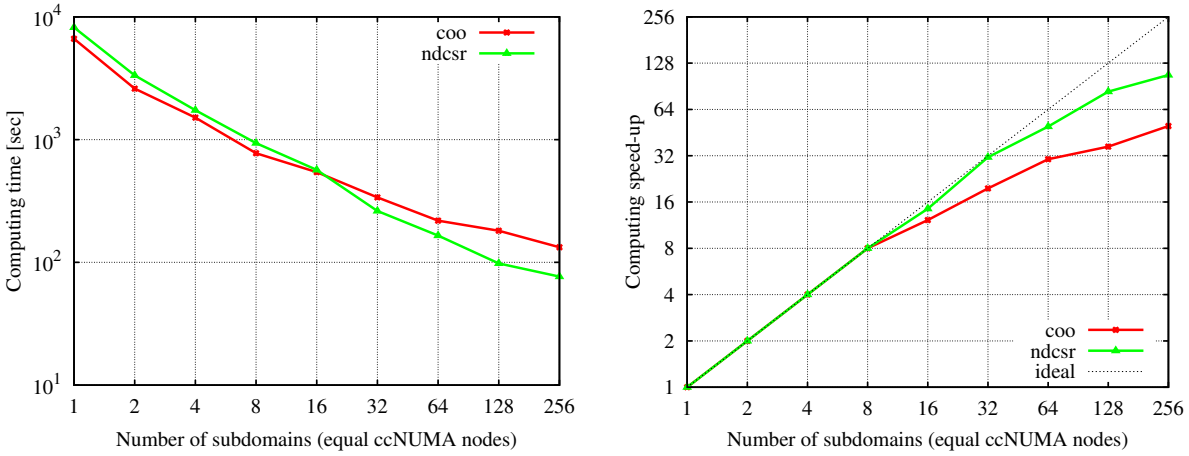


Figure 6.29: Cray XE6 cluster: Scaling of accumulated computational times for non-matrix-vector operations of the PPCG method (42.8 million d.o.f.s) and the resulting speed-ups (right).

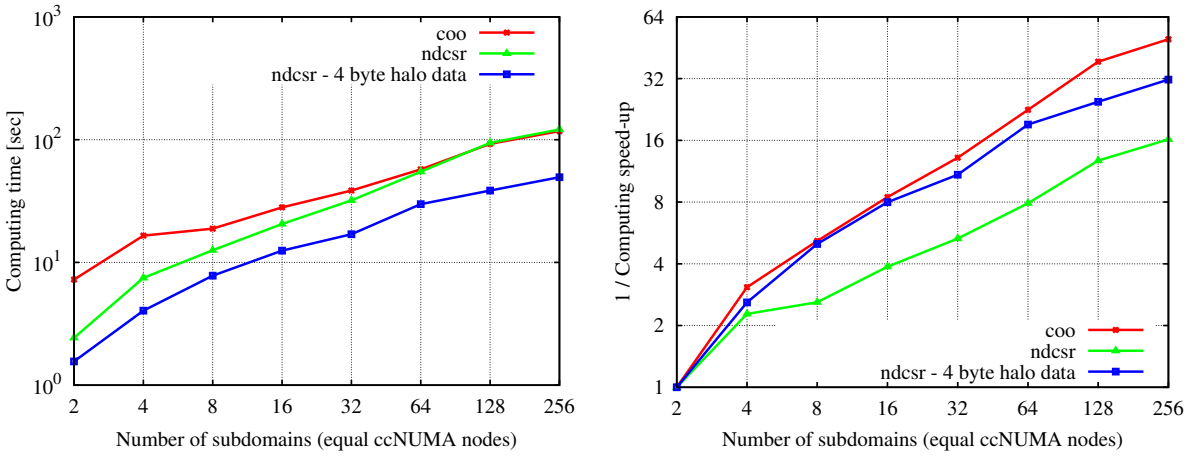


Figure 6.30: Cray XE6 cluster: Scaling of accumulated computational times for MPI based communication of the PPCG method (42.8 million d.o.f.s) and the resulting speed-ups (right).

compiler option	value	description
-march=	bdver1	the bulldozer flag activating the AVX instruction set
-msse4a	-	latest version of the SSE instruction set
-O3	-	code optimization of level 3 (highest)

Table 6.28: GNU compiler options used for the Cray XE6 cluster 'Hermit' at HLRS.

6.7 Concluding remarks

The previous sections of this chapter have given an overview of the scalability and efficiency of FE routines considering the linear-elastic analysis of different large-scale 3D homogeneous and heterogeneous material specimens as well as of artificial composites at different high-performance computers. It was proven, that the storage scheme applied for

the core FE data and for the resulting (sparse) matrices as well as for the chosen preconditioning technique are crucial to reduce the memory demand required. Furthermore, the eigenvalue scaling for the preconditioning, with its notation given in chapter 5, is time-efficient and robust in regards to the solver performance and in respect to the quality of the nodal FE solution. Moreover, the implementation of the parallelized conjugate gradient method, using the modified Jacobi preconditioning with its special eigenvalue scaling (applied for each of the subdomains) and a special nodal compressed row storage of the submatrices significantly improved the solver performance. Additionally, as a final step, the code was adapted for the application on high-performance computers with multiple CPU or multiple CPU-GPU nodes, which improves the performance and scalability even further. In chapter 7 these findings are utilized to improve the performance and to allow a scalability of a nonlinear simulation model, especially adaptable for multiphase composites with a volume-based interfacial transition zone involved with an aligned mesh in high resolution (as described in chapter 4). Here, the special focus is the representation of the softening behavior of such specimens in the critical zone (mainly induced by damage effects), which occurs if the degradation process of the material stiffness initially starts and irreversibly continues under certain loading conditions. This is the reason why the iterative (PPCG) solver technique combined with the elastic-inelastic domain split and the sequential linear analysis (SLA) will be adapted in chapter 7 to enble a scalable incremental-iterative computation of degradation processes in multiphase material.

Chapter 7

Nonlinear material modeling including damage effects

In this final chapter all numerical investigated instruments will be unified in one numerical nonlinear simulation model. Here, the nonlinearity is restricted to consider a nonlinear material behavior only and thereby, a geometric linear structural behavior with small deformations. The proposed nonlinear model is focused on the initiation and propagation of damage effects in multiphase composites, especially an academic example of a hybrid-meshed specimen decribed in the previous chapters. Therefore, a smeared strain-softening damage approach is chosen without the necessity to involve return mapping techniques, which iteratively separate plastic terms from the total terms of tensorial strain and stress components. As a result, a scalable saw-tooth tensile-softening model was developed based on [Rots et al. 2006] and applied for the sensitive volumetric interfacial zone of the decribed three-phase material specimen. The scalable PPCG solver with the modified Jacobi-point preconditioning (chapter 5) was adapted and the high-performance computing framework, as proposed for the large-scale linear-elastic FEA (as described in chapter 6), was taken into account. In the following sections the notation of the nonlinear finite element method is presented followed by the basic description of an isotropic damage law and by regularization techniques to overvome resulting mesh bias occuring in such smeared crack analysis. After, the saw-tooth approach will be presented as well as its modified scalable algorithm. Then follows the numerical verification within 3D notched beam examples. Finally, results of the scalable damage analysis applied to the hybrid-meshed three-phase specimen which considers the elastic-inelastic domain split in combination with efficient element formulations and assembly strategies and also its hybrid partitioning (chapter 3 and 4, respectively) including a initial decomposition and a load-balanced damage zone, are given.

7.1 Nonlinear finite element method

The physical nonlinearity induced by the applied material law, which is considered in the Navier differential equation system, its numerical FE based approximation yields to a change in the stationary global equilibrium equation where the stress-strain relation is no longer to be assumed as linear. This is the case, if a limit load state of a material is reached and the softening process starts to initiate, such as e.g in quasi-brittle materials, where the history of the deformation state has nummerically to be taken into account. Therefore, path following algorithms were developed to solve such types of nonlinear equation systems stepwise, whereby the Newton method (also known as Newton-Raphson method) is a basic numerical technique. Here, it is assumed that the nonlinearity is caused by the nonlinear material law applied for the elements in combination with a constant displacement-strain matrix. This results in an update of the global stiffness matrix in each iterative step due to the corresponding elements which are dedicated to a change of their material tensor based on the (nonlinear) constitutive relation.
Basically, it is assumed that the internal forces (now depending on the deformation state) and external forces are in equilibrium, such as

$$\boldsymbol{f}^{int}(\boldsymbol{u}) = \boldsymbol{f}^{ext} \tag{7.1}$$

with $\boldsymbol{f}^{ext}$ as the vector of external forces as given in eq. (3.58) of chapter 3. The internal forces can be determined by

$$f_i^{int}(\boldsymbol{u}) = \int_\Omega B_{ij}^T \sigma_j(\boldsymbol{\varepsilon}) d\Omega \tag{7.2}$$

Moreover, for a deformation state $\boldsymbol{u}$ with known stresses and also known internal forces $\boldsymbol{f}^{int}(\boldsymbol{u})$ considering its neighboring state $\boldsymbol{u} + \delta\boldsymbol{u}$, the relation between the corresponding internal forces can be described by

$$\boldsymbol{f}^{int}(\boldsymbol{u} + \delta\boldsymbol{u}) = \boldsymbol{f}^{int}(\boldsymbol{u}) + \boldsymbol{G}(\boldsymbol{u})\delta\boldsymbol{u} \tag{7.3}$$

whereby the tangential stiffness matrix $\boldsymbol{G}$ can be expressed as

$$G_{ij} = \frac{\partial f_i}{\partial u_j} \tag{7.4}$$

The load-controlled Newton iteration incrementally applies the final external load $\boldsymbol{f}^{ext}$ in k steps such that firstly, for the kth load increment the current ith increment of the deformation state $\delta\boldsymbol{u}^{k,i}$ is computed by

$$\boldsymbol{G}(\boldsymbol{u}^{k,i-1})\delta\boldsymbol{u}^{k,i} = \boldsymbol{r}^{k,i-1} \qquad\qquad i = 1, 2, ... \tag{7.5}$$

and secondly, the update of the vector of displacements is done by

$$\boldsymbol{u}^{k,i} = \boldsymbol{u}^{k,i-1} + \delta\boldsymbol{u}^{k,i} \tag{7.6}$$

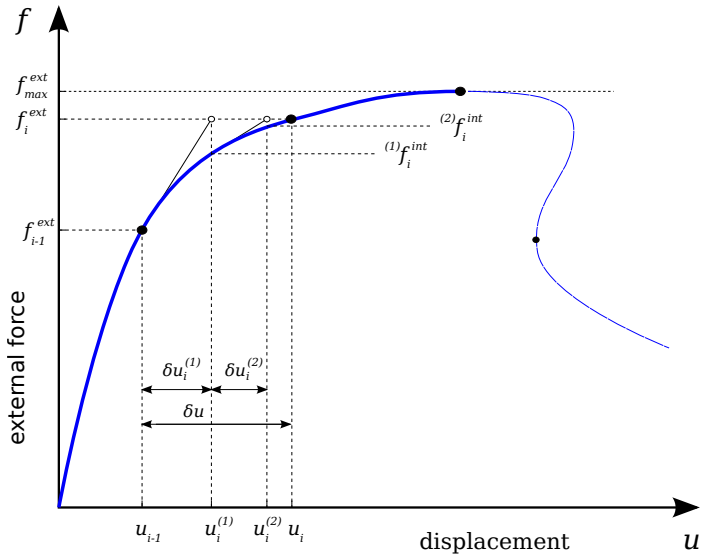


Figure 7.1: Nonlinear load-displacement path followed by the Newton-Raphson algorithm.

Hence, the vector of residuals is defined as

$$\boldsymbol{r}^{k,i-1} = \boldsymbol{f}^{ext,k} - \boldsymbol{f}^{int}(\boldsymbol{u}^{k,i-1}) \tag{7.7}$$

In general, load-controlled Newton schemes are incrementally applicable to reach the maximum structural load (load-carrying capacy). The nonlinear response path can numerically be followed up to its peak, as illustrated in fig. 7.1. Other solution strategies such as the displacement-controlled analysis or the load-displacement-constraint methods overcome this issue, but will not be presented in detail in this work. A further instrument is the applied material model enabling the representation of damage effects. By this, a local isotropic damage formulation will be reviewed in detail in the following section.

7.2 Smeared damage approach: Local isotropic damage model

Assuming mechanical isotropy for the material tensor (constant Poisson's ratio), the simplest notation of an isotropic damage law occurs by

$$\sigma_i = (1-\omega) C_{ij} \varepsilon_j = (1-\omega)\sigma_i^{\text{eff}} \tag{7.8}$$

with $\boldsymbol{\sigma}$ as the nominal stresses and $\boldsymbol{\varepsilon}$ as the vector of total strains. The determination of the scalar damage parameter ω can be done by considering the effective strain as

$$\varepsilon^{\text{eff}} = \frac{1}{E}\lambda_{\max}(\sigma_i^{\text{eff}}) \tag{7.9}$$

whereby the scalar term ε^{eff} is equivalent to the maximum eigenvalue of the effective stress tensor $\boldsymbol{\sigma}^{\text{eff}}$ divided by the Young's modulus, also called the equivalent strain. After, the damage parameter ω (depending on ε^{eff}) results in

$$\omega = \begin{cases} 0 & \varepsilon^{\text{eff}}_{\text{max}} < \varepsilon_0 \\ 1-\beta & \varepsilon^{\text{eff}}_{\text{max}} \geq \varepsilon_0 \end{cases} \tag{7.10}$$

for which β results in

$$\beta = \frac{\varepsilon_0}{\varepsilon_{\max}^{\text{eff}}} e^{-\alpha} \qquad \text{with} \qquad \alpha = \left(\frac{\varepsilon_{\max}^{\text{eff}} - \varepsilon_0}{\varepsilon_u - \varepsilon_0} \right) \tag{7.11}$$

Here, the ultimate strain ε_u is

$$\varepsilon_u = \frac{2 g_f}{f_t} \tag{7.12}$$

considering the tensile strength parameter f_t and the adjusted specific fracture energy g_f. However, the local isotropic strain-softening approach still leads to mesh dependencies mainly influencing the orientation of the localized crack band and moreover, often to an underestimation of the dissipated energy during the fracture process. Therefore, regularization strategies are developed to reduce the mesh bias, where a short explanation of the determination of g_f is given in the following section.

7.3 Effects caused by mesh bias and necessary regularization

In general, continuum damage models can be adapted for the representation of smeared strain-softening bands in quasi-brittle continuums. The softening mechanism is caused by a decreasing stress-strain relation within the post-peak region which area represents the current fracture energy dissipated within the (smeared) crack surfaces. Due to the localization of the damage in one layer of finite elements, the element shape and size and by this, the resulting mesh bias influences the correct orientation of the smeared crack as well as the energy dissipated. By this reason, based on the crack band theory the damage is to assumed as a crack band with a characterisic thickness. This parameter will be taken into account to adjust the fracture energy G_f resulting in the specific fracture energy g_f and consequently, is set to the equivalent element length h of the spatial discretization such that

$$h = \begin{cases} l_e & \text{line elements} \\ A_e^{\frac{1}{2}} & \text{plane elements} \\ V_e^{\frac{1}{3}} & \text{volume elements} \end{cases} \tag{7.13}$$

Then, the adjusted parameter of the fracture energy yields

$$g_f = \frac{G_f}{h} \tag{7.14}$$

In most case it is recommended to set h to the minimal equivalent element length, which is proposed in this thesis, also if irregular meshes are being used. When considering the crack band thickness, it must to be mentioned, that the width of the fracture process zone still depends on the element size but nevertheless, a realistic value of the dissipated energy can be represented. Moreover, several approaches of regularization exist to decrease the effects caused by mesh bias and finally, leading to a mesh independent size of the

fracture process zone by considering a so-called localization limiter. This is the objective of nonlocal damage models for which the author recommends the literature [Jirásek et al. 2003].

7.4 Alternative approach: Saw-tooth softening model

The saw-tooth approach (STS) with tensile-softening [Rots et al. 2006] enables a repeated linear-elastic analysis including an update of the specific material and design or model parameters during the so-called sequential linear analysis (SLA). This results in an inelastic response behavior of the investigated material specimen. The update of the material tensor and the tensile strength depends on the current stress state resulting from the deformation state of the previous load step. This leads to a nonlinear post-peak softening branch without the necessity to apply a (Newton-like) nonlinear solution technique, which is often combined with computationally-expensive return mapping techniques (applied to each integration point) to quantify the history variables of the inelastic state. During the SLA, the STS model specifies the corresponding saw-tooth diagram, which describes the linear or nonlinear stress-strain relation for the softening slope and by this, the overall nonlinear material behavior. In general, the saw-tooth softening diagram basically defines the incremental modification of the material tensor mainly changed by the reduction of the Young's modulus (in isotropic case) and by the tensile strength within a (fixed) number of reduction steps. In [Rots et al. 2004] similar approaches are developed and validated for regular meshed and notched beam examples in 2D.
This chapter includes two major tasks: Firstly, to obtain a robust and stable scalable SLA technique which is based on the STS model within a high-performance computing framework, and secondly, to adapt the saw-tooth model with tensile-softening for large-scale FE models in terms of hybrid discretized multiphase specimens. Additionally, for performance aspects, the characteristics of mesh dependencies and of (partial) regularization techniques are numerically investigated.

7.4.1 Model description: Evolution of the saw-tooth approach with tensile-softening

For uniaxial tensile load cases, the following notations capture the numerical evolution of the stiffness degradation induced by the saw-tooth model with tensile-softening. Here, the stress state in an element is evaluated by a scalar functional term $f(\boldsymbol{\sigma})$, resulting from the invariants or from the combination of the invariants of the stress tensor. Within this condition, for example considering the maximum principle stress (in LEFM) or the Rankine criterion adapted from the discrete or smeared crack analysis, the current stress state nears or exceeds the tensile strength such as

$$0 \leq f(\boldsymbol{J}) - f_t \quad \text{with} \quad J_i \in \{J_1, J_2, J_3\} \tag{7.15}$$

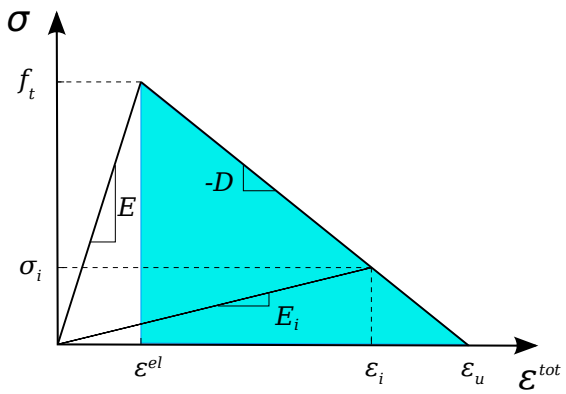


Figure 7.2: Bilinear tensile-softening stress-strain curve and the evolution of D depending on the current Young's modulus E_i.

the Young's modulus of this element is reduced in the following way

$$E_i = \frac{E}{a_i} \quad \text{for} \quad i = 1, 2, ..., N \tag{7.16}$$

with a_i as the fixed reduction factor of the current stage of the tensile stress-strain softening curve. For each current step E_i of the critical element will be modified from the previous SLA step according to

$$E_i = \frac{1}{a} E_{i-1} \tag{7.17}$$

Additionally, the tensile strength for the corresponding critical element is also updated during each global load step. Assuming a bilinear stress-strain curve, as illustrated in figure 7.2, the notation for the update at a critical step i is based on the following criterion considering a linear softening slope: Within the constant scalar tangent modulus D the current tensile strength is described as the distance of the actual (total) strain state to the ultimate strain (representing the smeared crack fully evoluted in the element), which is nearly decreasing to zero, if smeared crack effects are starting to initiate. By this, it follows

$$f_{t,i} = D(\varepsilon_u - \varepsilon_i) \tag{7.18}$$

The current total strain ε_i can then be determined considering the actual Young's modulus E_i with

$$\varepsilon_i = \frac{f_{t,i}}{E_i} \tag{7.19}$$

The substitution to eq. (7.18) yields to

$$f_{t,i} = D\varepsilon_u - D{E_i}^{-1} f_{t,i} \tag{7.20}$$

Then, the isolation of $f_{t,i}$ leads to

$$f_{t,i} = D\varepsilon_u \left(1 + DE_i^{-1}\right)^{-1} \tag{7.21}$$

which finally results in

$$f_{t,i} = E_i \varepsilon_u \big(E_i + D\big)^{-1} D \tag{7.22}$$

Since the softening modulus D remains constant during the saw-tooth procedure with linear tensile-softening, a closed form can be presented by modifying eq. (7.18), if the total inelastic strain (as the difference of the ultimate strain to the maximum elastic strain) is considered

$$f_t = D(\varepsilon_u - \varepsilon_{el}) \tag{7.23}$$

The elastic peak result in

$$f_t = E\varepsilon_{el} \tag{7.24}$$

and modifies eq. (7.23) to

$$D = f_t \big(\varepsilon_u - f_t E^{-1}\big)^{-1} \tag{7.25}$$

Therewith, eq. (7.21) and eq. (7.25) deliver the exact result for the update of $f_{t,i}$ and for the scalar softening modulus D (as recommended in [Rots et al. 2006]). Here, for linear tensile-softening the tensile strength f_t^i is updated with

$$f_t^i = \varepsilon_u E_i \frac{D}{E_i + D} \quad \text{with} \quad D = \frac{f_t}{\varepsilon_u - \frac{f_t}{E}} \tag{7.26}$$

considering the fracture energy G_f of Mode I cracking, the current Young's modulus E_i and the ultimate strain ε_u with

$$\varepsilon_u = \frac{2G_f}{f_t h} \tag{7.27}$$

Moreover, the design parameter h describes the dimension of the element size, which indicates the crack band width. In the threedimensional space, the value of this parameter can be approximated as

$$h = (V_e)^{\frac{1}{3}} \tag{7.28}$$

with V_e as the volume of the smallest element or as the average element volume. For lower order elements the crack band width h may be set corresponding to the avarage element size, which is recommended in [Rots et al. 2004] and which is adapted for mixed 3D FE meshes in this work. The number of saw-teeth N represents the number of updates before a critical element is removed. This parameter is fixed and is also equal for all elements. Finally, the update of the element stiffness matrix considers the modification of the Young's modulus of the material tensor $\mathbb{E}$ for the isotropic case with

$$\mathbb{E} = E_i \mathbb{E}_0 \tag{7.29}$$

and is computed by

$$\boldsymbol{K}_i = \int_{\mathcal{V}} \boldsymbol{B}^T \mathbb{E} \boldsymbol{B} \partial \mathcal{V} = E_i \int_{\mathcal{V}} \boldsymbol{B}^T \mathbb{E}_0 \boldsymbol{B} \partial \mathcal{V} = \frac{1}{a} E_{i-1} \boldsymbol{K}_0 \tag{7.30}$$

since in this case the displacement-strain matrix $\boldsymbol{B}$ is constant. Moreover, the stiffness degradation can be expressed as stiffness decrement $-\Delta \boldsymbol{K}_i$ with

$$-\Delta \boldsymbol{K}_i = \boldsymbol{K}_i - \boldsymbol{K}_{i-1} \tag{7.31}$$

and

$$\begin{aligned} \Delta \boldsymbol{K}_i &= \boldsymbol{K}_{i-1} - \boldsymbol{K}_i \\ &= (E_{i-1} - E_i) \boldsymbol{K}_0 \\ &= \left(1 - \frac{1}{a}\right) E_{i-1} \boldsymbol{K}_0 \end{aligned} \tag{7.32}$$

The introduction of the scalar factor b leads to

$$\Delta \boldsymbol{K}_i = b_i E_{i-1} \boldsymbol{K}_0 \tag{7.33}$$

with b depending on the number of stiffness reductions N until the element is completely removed

$$b_i = \begin{cases} \frac{1}{a} - 1 & 0 \leq i < N - 1 \\ -1 & i = N - 1 \end{cases} \tag{7.34}$$

Furthermore, this requires the update of the assembled nonlinear matrix $\boldsymbol{G}_i$ at step i for the kth damaged element with

$$\boldsymbol{G}_i = \boldsymbol{K}_{i-1}^{gl} + \Delta \boldsymbol{K}_i^{(k)^{gl}} = \sum_{j=1}^{n} \boldsymbol{R}^T \boldsymbol{K}_{i-1}^{(j)} \boldsymbol{R} + \boldsymbol{R}^T \Delta \boldsymbol{K}_i^{(k)} \boldsymbol{R} \tag{7.35}$$

and yields to a nonlinear tensile-softening load-displacement response for the structural post-peak behavior. The remaining stiffness matrix $\boldsymbol{G}$ (which exists due to the accumulated stiffness reductions of m removed elements) results in

$$\boldsymbol{G} = \boldsymbol{K}^{gl} + \sum_{k=1}^{m} \Delta \boldsymbol{K}_i^{(k)^{gl}} \tag{7.36}$$

whereby b will be equal to -1 which finally, after N steps, leads to the complete loss of the stiffness of element k

$$\Delta \boldsymbol{K}_i^{(k)} = -E_{i-1} \boldsymbol{K}_0^{(k)} = -\boldsymbol{K}_{i-1}^{(k)} \tag{7.37}$$

and moreover, at the last step, respectively

$$\boldsymbol{K}_{i-1}^{(k)} + \Delta \boldsymbol{K}_i^{(k)} = \boldsymbol{0} \tag{7.38}$$

7.4.2 Combined strain and strength regularization

Since the described linear STS model depends on the mesh size, [Rots et al. 2006] recommends a mesh regularization strategy. The technique involves a combined update of the ultimate strain ε_u and the tensile strength f_t to achieve a mesh size objectivity and by that, to avoid an underestimation of the dissipated energy. This is valid for non-regularized meshes and is mainly caused by the characteristic of the saw-tooth diagram, where, the cumulated area of each saw-tooth corresponds to the dissipated energy being less than the fracture energy G_f. Moreover, the regularization is based on the fact, that the ultimate strain depends on the crack band width h and always tends to the quantity of the crack opening. Considering the saw-tooth diagram, an increase of the dissipated energy results in an update of the ultimate strain as well as the tensile strength after each SLA step such as

$$f_t^i = k \cdot f_t \quad \text{and} \quad \varepsilon_u^i = k \cdot \varepsilon_u \tag{7.39}$$

in which the scalar value of k can be determined with

$$k = \sqrt{\frac{k_1}{k_2}} \tag{7.40}$$

as the relation of the adjusted fracture energy

$$k_1 = \frac{G_f}{h} \tag{7.41}$$

and the actual energy resulting from the saw-tooth diagram

$$k_2 = \frac{1}{2} \sum_{i=0}^{N-1} \frac{{f_t^i}^2}{E_i} b_i \tag{7.42}$$

with b_i corresponding to eq. (7.34). The following section describes a scalable approach for the saw-tooth based SLA technique with tensile-softening.

7.5 Modified STS model: Scalable SLA with tensile-softening

7.5.1 Elastic-inelastic decomposition

The main effort of a parallelized numerical computation model of a sequential linear analysis results in repeated solving and post-processing procedures, e.g. when extracting and evaluating the global stress and strain tensors to find the most critical element. This can efficiently be executed in parallel for all subdomains. Consequently, the sequential stiffness update of one critical element per iteraton step will have a relatively low impact on the computational time compared to the overall computing time required within one SLA step. The elastic-inelastic domain split allows to reduce the numerical effort by applying the SLA only for the inelastic domain which is considering a load-balanced partitioning.

1.	INIT: $\boldsymbol{f}^{(j)} = \boldsymbol{f}$ and $E_{lim} = 10^{-6}$
2.	Distributed solve $\boldsymbol{K}^{(j)}\boldsymbol{u}^{(j)} = \boldsymbol{f}^{(j)}$
3.	Extract strains and stresses $\boldsymbol{\varepsilon}^{(j)}$ and $\boldsymbol{\sigma}^{(j)}$ per subdomain j
4.	Evaluate $\sigma_{cr}^{(j)} = \max f(\boldsymbol{\sigma})^{(j)}$ for all elements of subdomain j
5.	IF i=N: SET $E_i^{(k)} = E_{lim}$ and $f_{t,i}^{(k)} = 0.01 f_t$ THEN GOTO 10.
6.	ELSE GOTO 7.
7.	Get global critical stress $\widetilde{\sigma}_{cr} = \max \sigma_{cr,k}^{(j)}$ close to $f_{t,i}^{(k)}$ for element k of subdomain j
8.	Reduce $E_i^{(k)} = b_i E_{i-1}^{(k)}$
9.	Update tensile strength $f_{t,i}^{(k)} = \varepsilon_u E_i^{(k)} D(E_i^{(k)} + D)^{-1}$ of critical element k
10.	Compute element stiffness decrement $\Delta\boldsymbol{K}_i^{(k)}$ of critical element k and current SLA step i
11.	Update global stiffness matrix $\boldsymbol{K}^{(j)}$ of subdomain j which includes the critical element k
12.	Update global nodal forces $\widetilde{\boldsymbol{f}} = \frac{f_{t,i}^{(k)}}{\widetilde{\sigma}_{cr}} \boldsymbol{f}^{(j)}$ with $\boldsymbol{f}^{(j)} = \widetilde{\boldsymbol{f}}$
13.	GOTO 2.

Table 7.1: Algorithm of the scalable SLA technique with tensile-softening.

Thereby, one important task is to communicate the *id* of the critical element found at the subdomain n of the decomposed damage zone to the initial subdomain m where the FE data of this element are allocated.

7.5.2 Implementation characteristics

The efficient *ndcsr* storage of the global assembled stiffness matrix, as mentioned in chapter 3, section 3.4.4, only considers nodal blocks with non-zero entries with a continued symmetric character of the global stiffness matrix $\mathcal{G}$ during the update. This is split as follows

$$\mathcal{G}_{ij} = \mathcal{G}_{ij}^{\mathcal{L}} + \mathcal{G}_{ij}^{\mathcal{D}} \tag{7.43}$$

and thereby results in

$$\mathcal{G}_{ij}^{\mathcal{L}} = \bigcup_{n=1}^{N_e} \bigcup_{m=1}^{N_{\mathcal{L}}} E_0^{(n)} \mathcal{L}_{ij}^{(m,n)} - \bigcup_{k=1}^{N_d} \bigcup_{l=1}^{N_{\mathcal{L}}} E^{(k)} \mathcal{L}_{ij}^{(l,d)} \tag{7.44}$$

The block-based reduction of the diagonal part $\mathcal{D}$ is then given as

$$\mathcal{G}_{ij}^{\mathcal{D}} = \bigcup_{n=1}^{N_e} \bigcup_{m=1}^{N_{\mathcal{D}}} E_0^{(n)} \mathcal{D}_{ij}^{(m,n)} - \bigcup_{k=1}^{N_d} \bigcup_{l=1}^{N_{\mathcal{D}}} E^{(k)} \mathcal{D}_{ij}^{(l,d)} \tag{7.45}$$

with E_0 as the initial Young's modulus for each element. As mentioned before, based on the nodal storage scheme (*ndcsr*) storing the FE data, the stiffness reduction can efficiently be performed at the corresponding subdomain, where the critical element k has been identified. Considering eq. (7.33) and eq. (7.34) the *ndcsr* update of the critical

element	d.o.f.s	diag. blocks	entries	off-diag. blocks	entries
		$N_{\mathcal{D}}$	total	$N_{\mathcal{L}}$	total
linear tetrahedron	12	4	24	6	54
linear pyramid solid	15	5	30	10	90
linear hexahedron	24	8	48	28	252
quadratic tetrahedron	30	10	60	45	405
quadratic hexahedron	60	20	120	190	1710

Table 7.2: Nodal diagonal and off-diagonal blocks with corresponding matrix entries for several standard 3D finite elements.

element stiffness at the qth SLA step for the diagonal nodal FE blocks follows with

$$\left[\mathcal{G}_{ij}^{\mathcal{D}}\right]_{(q)}^{(j)} = \left[\bigcup_{n=1}^{N_e^{(j)}} \bigcup_{m=1}^{N_{\mathcal{D}}} E_0^{(n)} \mathcal{D}_{ij}^{(m,n)}\right]_{(q-1)}^{(j)} + b_q E_q^{(k)} \left[\bigcup_{l=1}^{N_{\mathcal{D}}} \mathcal{D}_{ij}^{(l,k)}\right]_{(q)}^{(j)} \tag{7.46}$$

and the off-diagonal nodal FE blocks, respectively, with

$$\left[\mathcal{G}_{ij}^{\mathcal{L}}\right]_{(q)}^{(j)} = \left[\bigcup_{n=1}^{N_e^{(j)}} \bigcup_{m=1}^{N_{\mathcal{L}}} E_0^{(n)} \mathcal{L}_{ij}^{(m,n)}\right]_{(q-1)}^{(j)} + b_q E_q^{(k)} \left[\bigcup_{l=1}^{N_{\mathcal{L}}} \mathcal{L}_{ij}^{(l,k)}\right]_{(q)}^{(j)} \tag{7.47}$$

The number of diagonal and off-diagonal blocks of the critical element (which are modifying the global stiffness matrix of the corresponding subdomain at each SLA step due to the reduced element stiffness matrix) are listed in table 7.2 for several 3D finite elements. In the following two examples are presented adapting the scalable SLA technique for the nonlinear simulation model [Schrader et al. 2013a].

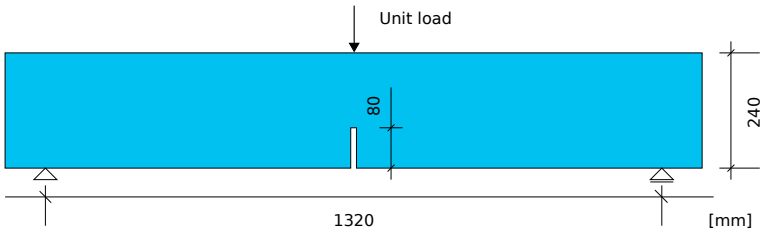
7.6 Numerical example: 3D notched beam

Two threedimensional notched beam examples are taken from the literature: [Hannawald 2010] (with numerical results presented in the following) and [Voormeeren 2011] (considered for further optimizations) were applied to verify the numerical model in respect to the proposed parallelization and solver technique combined with the saw-tooth softening model. In figure 7.3a the dimensions of the 3D notched beam are given considering a thickness t of 160 mm. Table 7.3b contains the used material properties: E as Young's modulus, ν as Poisson's ratio, f_t as the tensile strength and G_f as the Mode I fracture energy. Considering these parameters, the initial SLA values in respect to the saw-tooth tensile-softening diagram are computed for each subdomain before starting the SLA. All relevant values are given in the table of fig. 7.3b.
The hybrid decomposition (as illustrated in fig 7.4) considers the initial load-balanced METIS partitioning for the distributed solver as well as a decomposed damage zone corresponding to the inelastic domain in the region around the notch of the beam considered by the scalable SLA technique and resulting from the elastic-inlastic domain split. The

SLA parameter	description	value	unit
N	number of saw-teeth	20	-
a^{-1}	reduction factor	0.5	-
E	limit Young's modulus	0.03	N/mm^2
D	damage modulus	29.25	N/mm^2
ε_u	ultimate strain	0.06	-
h	crack band width / element size	2.04	mm
g_{f,A_1}	STS diagram fracture energy	0.04	N/mm^2
g_{f,A_2}	specific fracture energy	0.06	N/mm^2
k	regularization term	1.17	-

Table 7.3: Initial saw-tooth parameters for the SLA procedure applied for the 3D notched beam.

remaining region is characterized by linear-elastic material behavior. Fig. 7.5 shows the deformed structure and the distribution of the maximum principle stress concentrated around the notch. As a result of the SLA, the smeared crack, as illustrated in fig. 7.6 considering all degradated elements, propagates along the straight mesh lines of the hexahedral FE mesh. Moreover, fig. 7.7 shows the applied saw-tooth diagram describing the linear softening post-peak slope (left) and the load-displacement curve extracted during the SLA cycles with a corresponding control point in the middle of the beam (right). Some computational aspects in respect to the numerical simulation considering the CRAY XE6 cluster are given in table 7.4.



(a) Dimensions of the 3D notched beam.

property	value	unit
E	26700	N/mm^2
ν	0.18	-
f_t	1.93	N/mm^2
G_f	0.13	N/mm
t	160	mm

(b) Material properties.

Figure 7.3: Dimension and parameter of the 3D notched beam.

Hardware	CRAY XE6 cluster
CPU architecture	multi-core AMD Interlagos
ccNUMA nodes	16
number of d.o.f.s	390,000
average SLA time	1.4 sec
number of SLA cycles	10,000
overall (elapsed) SLA time	4 h 13 min

Table 7.4: Overall computational effort of the scalable SLA technique.

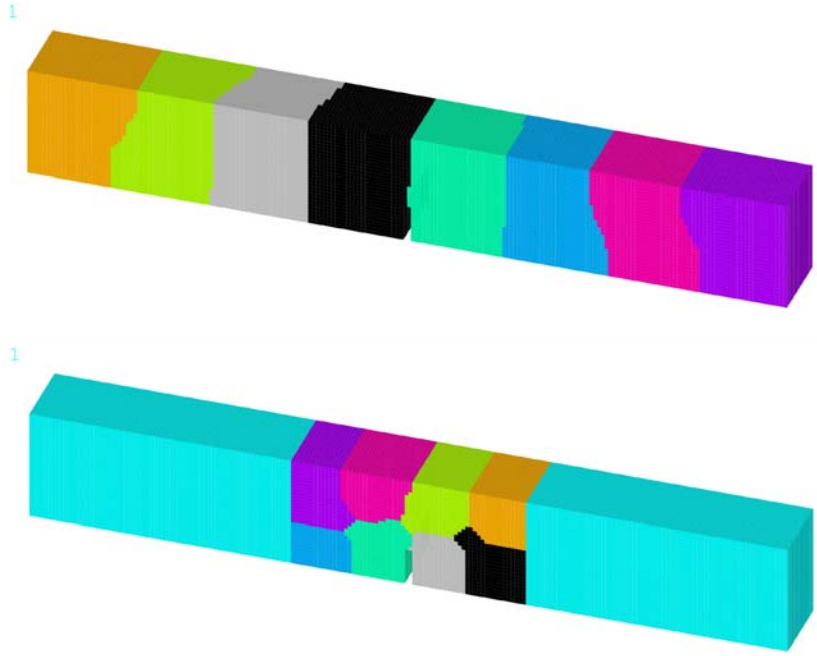


Figure 7.4: Hybrid decomposition for the parallel SLA procedure: Initial decomposition (top) and partitioning of the inelastic domain by using METIS.

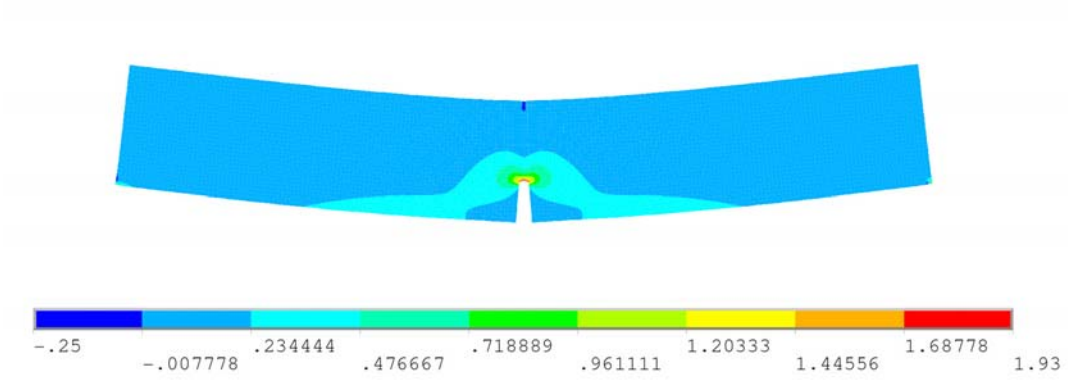


Figure 7.5: Maximum principle stress distribution for the initial load step reaching the limit stress state close to the tensile strength.

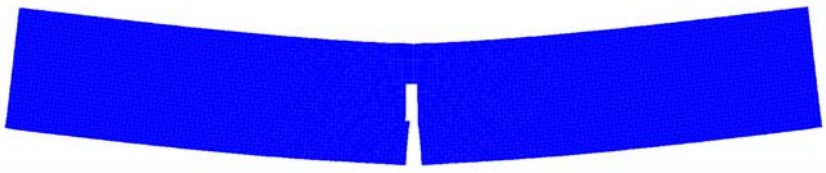


Figure 7.6: Deformation state including the removed elements (as a smeared crack) after 10,000 parallel SLA cycles.

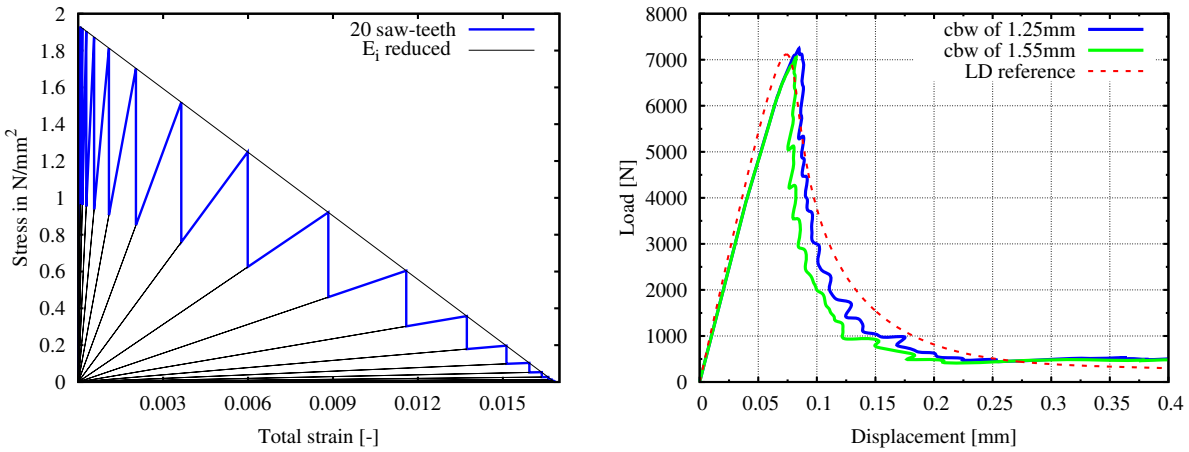


Figure 7.7: Example: Saw-tooth diagram and resulting load-displacement curve with tensile-softening (right) with two different values for the crack band width (cbw).

7.7 Numerical example: Hybrid meshed multiphase specimen

7.7.1 Scalable SLA: Evolution of delamination effects

The application of the hybrid discretization and decomposition technique in combination with the implemented scalable SLA algorithm is done by investigating a three-phase matrix-inclusion model with a very fine mesh for the aligned transition zone (fig. 7.8). Due to using a commercial mesher the mesh is converted by reducing the order of the shape function for chosen elements or by preparing the mesh for the consideration of special integration techniques but at the same time keeping the number of elements and their orientation constant. This enables a further significant reduction in nodal d.o.f.s. The comparison between the commercial software ANSYS and the inhouse library MDiSP is given in table 7.5. The hybrid decomposition evolves, firstly, from the initial load-balanced partitioning of the total hybrid mesh and, secondly, from the initial load-balanced partitioning of the aligned mesh or parts of it including the VITZ separated as a damage zone (fig. 7.9), which has been considered during the scalable SLA. This enables the PPCG technique (based on the initial decomposition) to solve the distributed system of equations in conjuction with the SLA computational procedures. Additionally, this also involves the simultaneous element-based evaluation of the critical stress states for each domain of the second decomposition, a decomposed damage zone (resulting from the elastic-inelastic

mesh	nodes	hexaeder	tetraeder	pyramid solid	d.o.f.s
original	2,064,663	1,936	2,842,487	1,132	6,193,989 (100%)
converted	486,104	1,936	2,842,487	1,132	1,458,312 (23.5%)

Table 7.5: Efficient reduction of the number of nodal d.o.f.s by converting the mesh considering special integration schemes with a constant number of elements and their orientation.

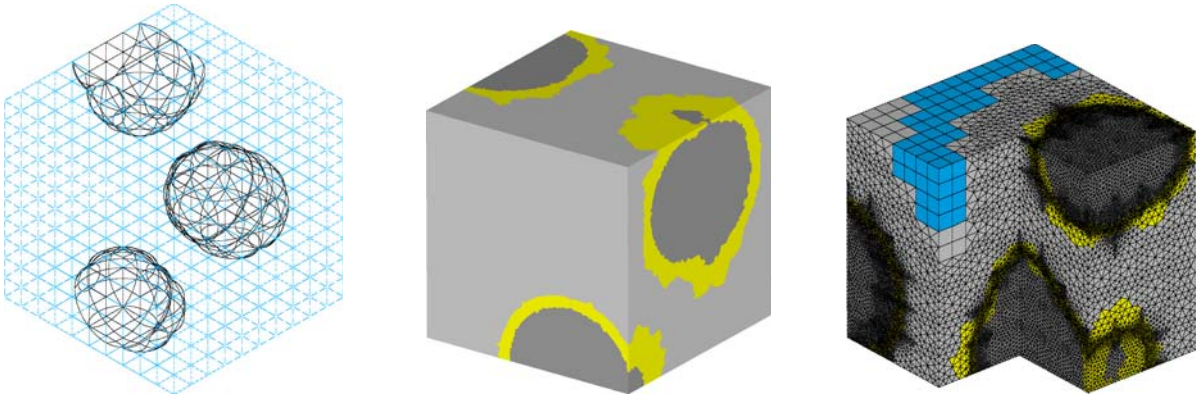


Figure 7.8: Hybrid meshed three-phase specimen: Surfaces of the inclusions and the regular grid (left), the inclusions with the VITZ (colored, middle) and the hybrid mesh with the VITZ (yellow), embedded in the linear-elastic matrix (blue, right).

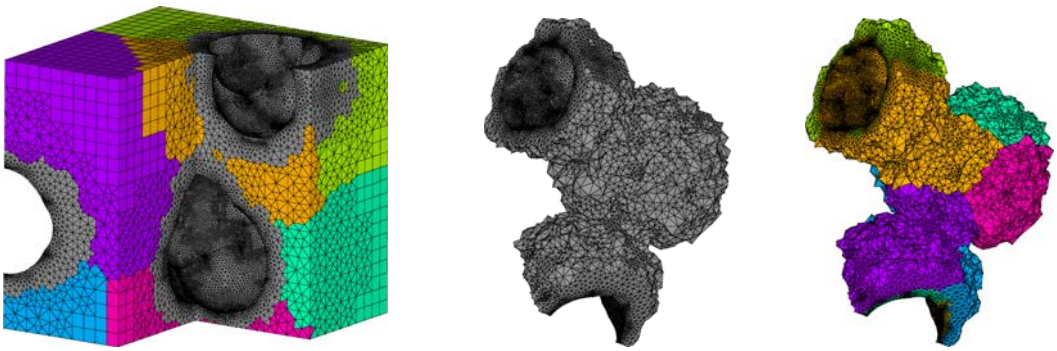


Figure 7.9: Hybrid partitioning: Left: Initial decomposition of the hybrid mesh with the transition zone in high resolution also prepared for the partitioning (gray), middle: Irregular meshing of the transition zone; Right: Load-balanced decomposition of the inelastic VITZ prepared for the nonlinear simulation.

Figure 7.10: Element-based delamination in the region of the VITZ after 1000 (left), 2000 (middle) and 4000 (right) parallel SLA cycles, evaluated by the von-Mises condition as critical criterion.

domain split). This yields to modified material properties of the critical element (Young's modulus as well as tensile strength) and also to its reduction of the element stiffness, which directly influences the entries of the global distributed stiffness matrices.
The decomposed damage zone is dedicated to the scalable computation of the strain and the stress tensors and to the update process of the corresponding current material tensor during the scalable SLA procedures and is based on the saw-tooth approach with tensile-

softening and also uniaxial tensile loading conditions (with a initial surface traction of $1N/mm^2$ on the top). Fig. 7.10 illustrates the application of the suggested method with the scalable SLA solver for a heterogeneous specimen and the resulting hybrid mesh and decomposition. As a result, the accumulated delamination effects arise in the transition zone (using the von-Mises yield condition as critical criterion) which are discovered after 1000, 2000 and 4000 parallel SLA cycles, resulting in an equal number of stiffness reductions distributed via the detected critical finite elements of the decomposed specimen. By this, the successive element failure in the transition zone yields to the delamination between the matrix and the inclusion related material phases.

7.7.2 Scalable SLA: Evolution of smeared crack initiation and propagation

Based on the results of the previous section, the final investigation is related to the practical application of the developed scalable SLA procedures for the smeared crack analysis, here especially interesting in the initiation and also the propagation of cracks in the volumetric interfacial transition zone of the three-phase hybrid meshed specimen. The Rankine cracking criterion for the evaluation of critical elements was adapted and had been combined with a tensile strength based regularization technique, being performed during the update process of the SLA. The regularization parameter represents the relation between the input fracture energy (as material property per unit area) and the fracture energy computed as the area of the defined stress-strain saw-tooth diagram. Furthermore, the preconditioning matrix, which is based on the main-diagonal of the stiffness matrix of the corresponding subdomain, was rebuilt for each SLA cycle where the degradation occurs - assuming an improvement of the overall performance of the nonlinear simulation model. By this approach, the initial scaling values for ω and α_L (using the presented eigenvalue strategy) were used as constant and consequently, had to be computed only once before the SLA was started. The material properties and the proposed parameters for the SLA are given in table 7.6. To validate the qualitative results of a smeared crack analysis by using the scalable SLA with tensile-softening, the linear-elastic FEA in uniaxial tension case was performed with a commercial FE code. Moreover, the hybrid three-phase specimen was tested with and without inclusions and with a small volumetric interfacial zone in high-resolution of the corresponding aligned FE mesh, also applying a decreased Young's modulus. Thereby, the first principle stress evaluation indicated a high stress concentration starting in the VITZ around different inclusions. The objective was then to evaluate the results in respect to the initiation of smeared cracks (as inelastic strains) and to the successive strain-softening concerning the SLA model over hundreds and thousands of SLA cycles with an update of the applied loading conditions. The illustration of the smeared cracks (as cumulated degradated finite elements) is shown in figure 7.12, where the crack was initiated exactly in the region of the maximum stress

SLA parameter	description	value	unit
ν	Poisson's ratio (constant)	0.18	–
E_{VITZ}	Young's modulus VITZ	25000	N/mm^2
E_m	Young's modulus matrix	31000	N/mm^2
E_a	Young's modulus aggregates	62000	N/mm^2
f_t	tensile strength (constant)	3.00	N/mm^2
G_f	fracture energy	0.10	N/mm
N	number of saw-teeth	5	–
a^{-1}	reduction factor	0.5	–
E_{lim}	limit Young's modulus	0.02	N/mm^2
D	damage modulus	21.37	N/mm^2
ε_u	ultimate strain	0.14	–
h	crack band width / element size	0.47	mm
g_{f,A_1}	STS diagram fracture energy	0.15	N/mm^2
g_{f,A_2}	specific fracture energy	0.21	N/mm^1
k	regularization term	1.18	–

Table 7.6: Material properties and initial saw-tooth parameters for the SLA procedure applied for the three-phase hybrid meshed specimen.

concentration. This effect has also been verified by the commercial FE software ANSYS (release 14), given in figure 7.11, where the smeared cracks nearly propagate orthogonal to the uniaxial tension (unit) load. After more than 15,000 SLA cycles, the plane region in the middle of the VITZ was completely filled with critical degradated elements. After 25,000 and more load cycles, a multiple crack expansion occured. A further modification extending the inelastic domain into parts of the grid-based matrix material results in the complete failure of the specimen (shown in figure 7.13). There, the crack path was developing along the straight mesh lines of the regular grid. Moreover, for the extraction of corresponding load-displacement curves four control points were placed at the edge nodes within the half height of the specimen.

The transition of critical elements through the connecting boundary of different subdomains was also possible by the hybrid partitioning approach, where two partitionings are handled at runtime: the initial decomposed mesh and the partitioning of the damage zone (shown in figure 7.14). Additionally, it is recommended by the author to update the preconditioning matrix of the PPCG solver resulting from the modified Jacobi-point preconditioner (considering the eigenvalue scaling strategy), where a performance comparison (considering three different Jacobi-point based preconditioners) is shown in figure 7.15. Due to the change of the condition numbers of the distributed global stiffness matrices induced by the successive degradation of the element stiffnesses during the SLA, the update of the preconditioning matrix of the critical subdomain was necessary. As a result, the memory-efficient main-diagonal update of the preconditioning matrix during each SLA step did not influence the overall SLA performance in respect to the increased computing time of one SLA step. In comparison to the SLA computations where the pre-

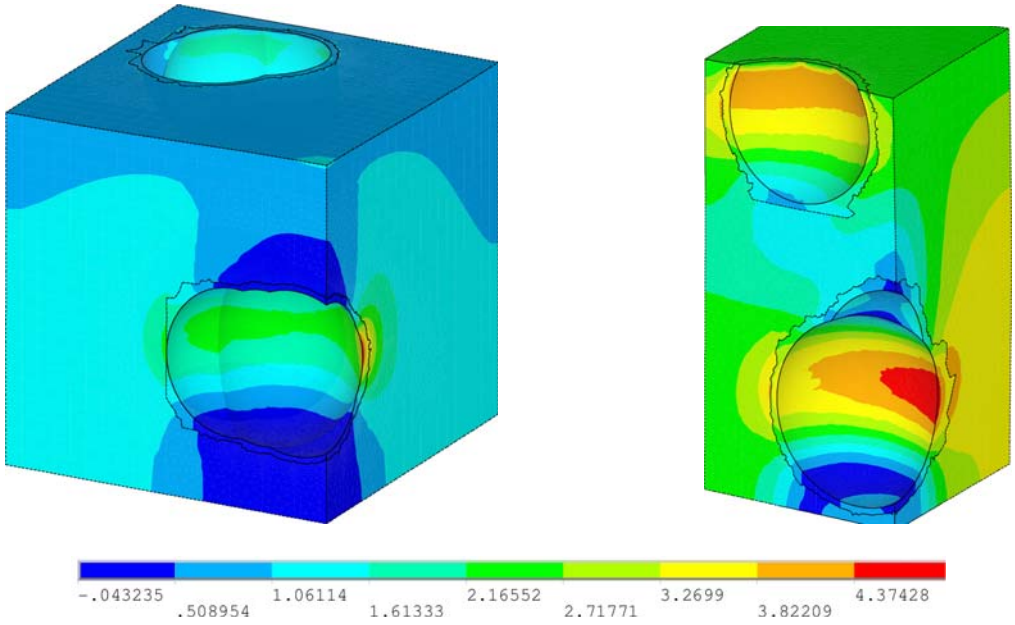


Figure 7.11: Hybrid meshed heterogeneous specimen: Concentration of the first principle stress distribution in respect to linear-elastic FEA, evaluated by commercial FE code (ANSYS).

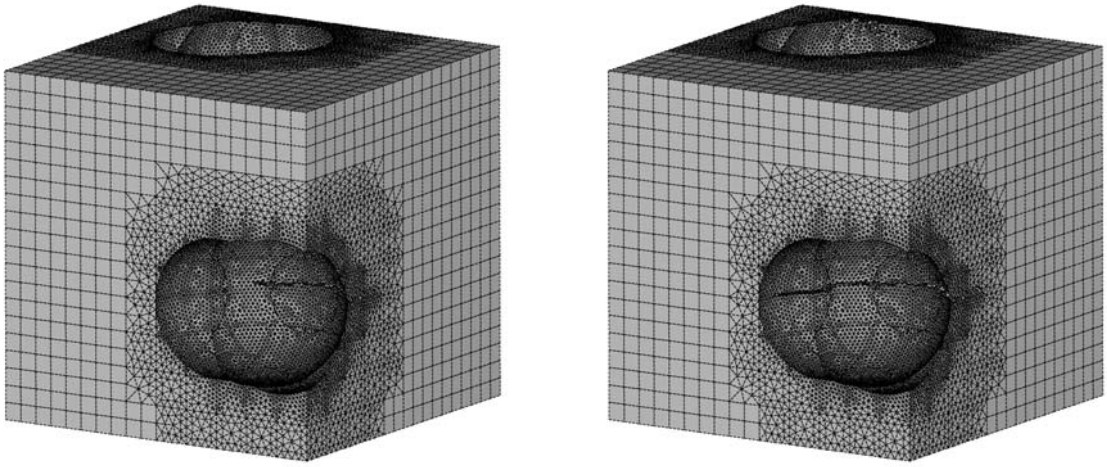


Figure 7.12: Hybrid meshed heterogeneous specimen: (Smeared) crack initiation and propagation after 1000 (left) and 2000 SLA cycles respectively (inclusions excluded).

conditioning submatrix had not been rebuilt (where the degradation occurs), considering the update keeps the number of iterations required by the PPCG solver significantly low. Additionally, an overall (moderate) increase in the number of iterations was observed during the scalable SLA. A summary concerning some data of the computational performance are listed in table 7.7.

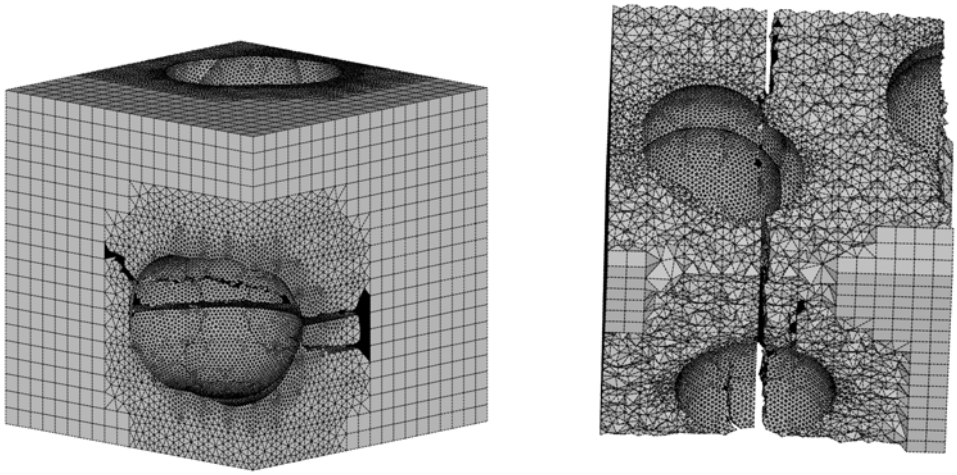


Figure 7.13: Hybrid meshed heterogeneous test specimen: Extended (smeared) crack expansion after 10,000 SLA cycles in the VITZ (left) and fracture of the specimen by propagating crack pattern into the matrix material after more than 50,000 SLA cycles (coarse regular grid, inclusions excluded).

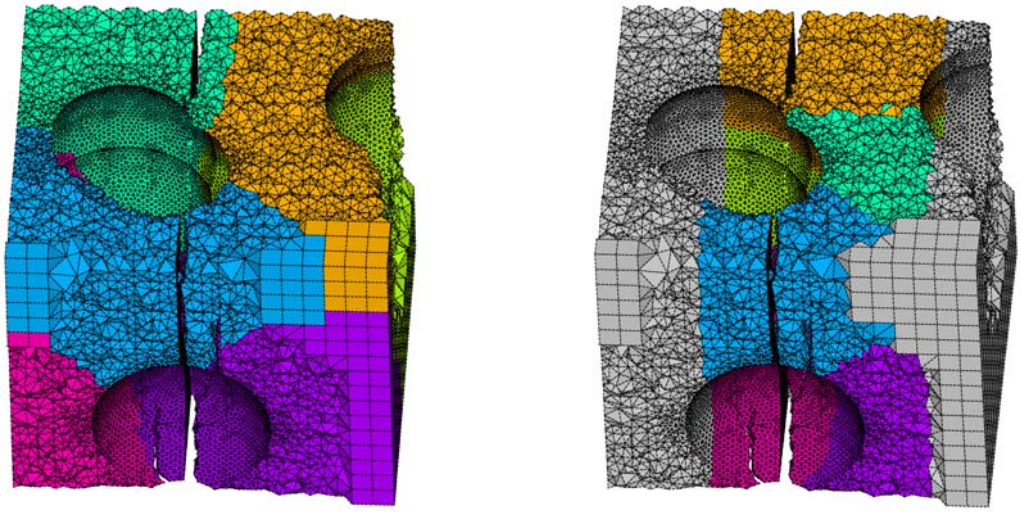


Figure 7.14: Hybrid meshed heterogeneous test specimen: Smeared crack propagation after more than 50,000 SLA cycles through different subdomains of the initial partitioning of the hybrid mesh (left) as well as through the partitioning of the aligned mesh as the adaptive damage zone considered by the scalable SLA procedure (inclusions excluded).

7.8 Concluding remarks

All investigated and developed approaches in this work for the damage analysis of multiphase composites presented have been proven by numerical examples shown in the last section. The hybrid discretization and decomposition technique for a three-phase hybrid meshed specimen was combined with an iterative and scalable solver strategy based on

the preconditioned conjugate gradients. The distributed global system of equations had been efficiently assembled respecting efficient element formulations and an effective nodal storage scheme for the involved finite elements. Furthermore, the performance of the solver had been improved by adapting an eigenvalue strategy for the preconditioning matrix. Additionally, it could be confirmed, that the iterative PPCG solver is suitable for the application of parallelized procedures for the saw-tooth tensile-softening model. This enables to perform the nonlinear response behavior of heterogeneous specimens, especially adapting the scalable SLA for the smeared crack analysis. For such an approach where material nonlinearies with the SLA technique are being simulated, the algorithmic stability (also for the preconditioned solver), the improved parallel performance and the ensurance of numerical convergence are the main advantages of the proposed technique. Nevertheless, for hybrid meshes additional numerical investigations are necessary regarding the softening behavior using regularized strategies based on the dissipated energy, e.g. considering a nonlinear saw-tooth diagram at material point level. This has also to be verified by comparing the results to other established methods of modeling material nonlinearities. Compared to regular meshes of homogeneous structures the evolved stress concentration differs and changes quickly during the SLA softening process when investi-

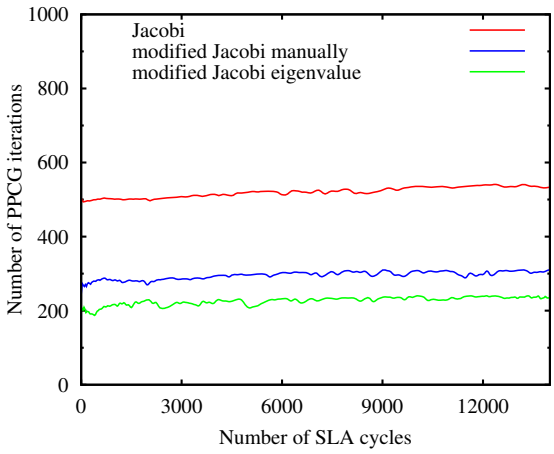


Figure 7.15: Hybrid meshed heterogeneous test specimen and scalable SLA: Evolution of the number of PPCG iterations during the SLA considering different preconditioning techniques.

Hardware	HP ProLiant DL585 G7
CPU architecture	AMD Opteron 6100
ccNUMA nodes	6
d.o.f.s	297,693
average SLA time	2.8 sec
number of SLA cycles	85,000
overall SLA time	2 days 18 h 6 min
memory per ccNUMA nodes	0.253 GB

Table 7.7: Overall computational effort of the scalable SLA technique.

gating three-phase material specimens (under monotonic loading conditions). Especially the damage evolution in such hybrid meshed specimens considers an initial damage zone including the VITZ and is compromising for the threedimensional damage modeling compared to discrete or discontinuous approaches. Moreover, a high-performance computing framework shows the high potential of the combined numerical methods regarding the computational scalability and memory efficiency, which enables a high number of stable load cycles characterized by a significant decrease in the computational (elapsed) time. The decrease in memory also occurs by avoiding a direct factorization of the tangential stiffness matrices for all subdomain, which is necessary, if a direct Schur complement based solver strategy is used. One disadvantage of the iterative solver strategy is the lack of the starting solution in each step (adapted from the previous step), which was difficult to be realized in this work. The mathematical modification of the distributed solver in combination with the used preconditioning technique (respecting the incremental change of the condition number of the corresponding subdomain matrix involved with degradation effects) may solve this issue and therefore, may lead to an improved performance of the scalable SLA technique.

Chapter 8

Summary

A hybrid 3D discretization technique was applied to the artificial geometry of a heterogeneous three-phase material specimen, considering a volumetric interfacial transition zone (VITZ) with high resolution. The inclusions and the surrounding VITZ were embedded in a regular grid, with a strong coupling of the different discretized material regions. An elastic-inelastic domain split was used separating the regular grid with linear-elastic material behavior and the aligned mesh for the inelastic region including the VITZ. This approach was motivated by the development of a scalable nonlinear simulation model based on the finite element method. Therefore, a hybrid domain decomposition technique was developed and adapted for heterogeneous specimens, taking into account an initial partitioning of the complete configuration and a load-balanced partitioning for the inelastic damage zone simultaneously. After, the scalable assembly of the global stiffness matrices of all subdomains, considering numerical techniques for the reduced element integration, was implemented. Furthermore, the voxel integration of the corresponding grid elements was also realized, using a memory-efficient allocation of the distributed finite element data as well as a memory-efficient nodal storage scheme being also suitable for fast matrix-vector operations. Moreover, based on a combination of the techniques for preconditioned conjugate gradients and the iterative Schur complement method, a scalable solution of the partitioned system of equations resulting from the hybrid discretization had been developed. After, the computational efficiency of the solver was improved by applying an eigenvalue scaling strategy for the preconditioning matrices. The scalable iterative solver concept, implemented with the message-passing interface standard (MPI), was then realized for a distributed, consistent load-balanced computing framework. This enabled the hybrid usage of multiple CPU as well as multiple GPU computing nodes. As a result, the scalablility of the implemented algorithms within a high-performance computing framework had been verified by a linear-elastic finite element analysis of large-scale FE problems with several million degrees of freedom on two high-performance clusters, namely the NEC Nehalem with several Tesla-GPU nodes and the CRAY XE6 cluster, at the high-performance computing center Stuttgart. By this, a

nonlinear scalable simulation framework for the damage-based analysis of heterogeneous hybrid-meshed specimens considering the sequential linear analysis (SLA) with a saw-tooth approach for tensile-softening had been developed. Then, the damage initiation in regions of the volumetric interfacial transition zone had been evaluated using a scalable version of the SLA model in conjuction with the modified Jacobi-point based preconditioned CG solver (considering the eigenvalue scaling strategy), improving the scalablility of the nonlinear structural analysis.

A question, which remains unanswered, is the weak coupling between the different discretization techniques in respect to their material description for the scalable application of (concurrent) multiscale models in 3D.

Chapter 9

Conclusions and outlook

In the following concluding remarks (based on the investigations made in this thesis) in respect to ongoing research activities for the mechanical damage analysis of multiphase composites can be summarized as

- The hybrid 3D discretization approach for multiphase material in conjuction with an iterative distributed solver technique is memory-efficient, relatively fast and also robust regarding the convergence behavior, which was realized by a scalable SLA technique to represent material nonlinearities such as damage effects (smeared cracks) which in general is resulting in a strain-softening continuum approach. This differs for Newton-Raphson based computations in conjunction with local or higher order nonlocal damage formulations.
- Due to the regular grid (representing the linear-elastic material region) the definition of the primal nodes, which was failed in this work, in respect to the application of the FETI-DP method may be possible. The resulting distributed system of equations can be efficiently solved by a Saddle-point approximation technique, which was investigated in this work.
- The proposed elastic-inelastic domain split should enable a weak coupling by dual or mixed methods (e.g. the Mortar method) between the different discretizations used for the elastic and for the inelastic region, respectively.
- A scalable HPC framework was realized, and moreover, a high-performance computing center (HLRS), which provide the hardware resources being necessary for large-scale nonlinear FE simulation models in 3D, was used and is recommended by the author for the future work.

The limitation of the hybrid discretization technique applied for multiphase composites to one scale only may be improved by considering a concurrent multiscale approach. Dual methods such as the Mortar method may be used to couple the coarse regular meshed domain (described by a linear-elastic material behavior) and the aligned meshed domain

involved with inelastic material behavior (including the representation of damage effects in that region). The elastic-inelastic domain split presented in this work may be used to model a distributed and concurrent two-scale approach. Therefore, the hybrid discretization approach applied to the material three-phase system may enabling the transport of damage effects to the initially elastic domain. Therefore, a special adaption strategy for the moving boundary combined with the mortar coupling is to be developed to consider continued damage propagation transferred between the different scales. Additionally, methods are also necessary controlling the dynamic load-balancing for the modified subdomains during the simulation runtime.

The hybrid discretization technique enables a distributed computation simultaneously, which reduces the number of degrees of freedom, specially for the 3D modeling of microstructural composites. The regular grid-based mesh or the resulting voxel discretization of the coarse (elastic) problem has the advantage that the FETI-DP method may be applied to such regions, because here it is possible to define the fixing (or primal) nodes at the regular and orthogonal boundary. The resulting distributed FE problem has then to be extended by the FETI-DP discretization of the initially elastic domains and also the Mortar discretization for the scale transition between the elastic and the inelastic subdomains. The consistence of the FETI computational parts (due to adaption for the coarse linear-elastic regions) avoids instabilities of the iterative and distributed solution procedure respecting the global nonlinear problem. Finally, the solution method for the hybrid discretization technique based on the preconditioned conjugate gradient method for a high-performance computing framework is to be extended, involving the mixed Schur-FETI-Mortar coupling of the decomposed problem.

To realize the distributed and concurrent two-scale computation applied to a hybrid CPU-GPU computation technique the hybrid HPC framework developed for the NEC Nehalem and Cray XE6 cluster at HLRS (or at other high-performance computing centers) may be used. The parallelization approach may then be improved by considering a hybrid MPI-openMP framework.

Bibliography

Amestoy P., Duff I., and L'Excellent J.-Y. (2007). Multifrontal massively parallel solver - users guide. *ERCIM News 50*, 1–46.

ANSYS (2012). *Commercial Finite Element Code ANSYS*. ANSYS Inc., Release 14.0.

Basermann A., Reichel B., and Schelthoff C. (1997). Preconditioned cg methods for sparse matrices on massively parallel machines. *Parallel Computing 23*(3).

Bathe K. (1995). *Finite Element Procedures*. Prentice Hall.

Bažant Z. and Jirásek M. (2002). Nonlocal integral formulations for plasticity and damage: Survey of progress. *Journal of Engineering Mechanics, ASCE 128(11)*, 1119–1149.

Bell N. and Garland M. (2008a). Efficient sparse matrix-vector multiplication on cuda - part 1. NVIDIA Technical Report.

Bell N. and Garland M. (2008b). Efficient sparse matrix-vector multiplication on cuda - part 2. NVIDIA Technical Report.

Belytschko T. (1984). Hourglass control in linear and nonlinear problems. *Comp. Meth. in Appl. 43*, 251–276.

Belytschko T. and Black T. (1999). Elastic crack growth in finite elements with minimal remeshing. *Int. J. for Num. Meth. in Eng. 45*(5), 6001–6020.

Bucher C., Schorling Y., Brehm M., and Unger J. (2007). *SLang the Structural Language - Part I User's Manual*. Bauhaus-Universität Weimar. Version 5.1.0.

Bulu A., Fineman J. T., Frigo M., Gilbert J. R., and Leiserson C. E. (2009). Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks. In SPAA 09: Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures New York, USA, ACM.

Caballero A., Lopez C., and Carol I. (2006). 3d meso-structural analysis of concrete specimens under uniaxial tension. *Computer Methods in Applied Mechanics and Engineering 195*(52), 7182–7195.

Carol I. and Bažant Z. (1997a). Damage and plasticity in microplane theory. *International Journal of Solids and Structures 34*, 3807–3835.

Carol I., Brat P., and Lopez M. (1997b). Normal/shear cracking model: Application for discrete crack analysis. *Journal of Engineering Mechanics ASCE. 123*, 765–773.

Carol I., Lopez M., and Roa O. (2002). Discrete crack analysis using zero-thickness interface elements. *Int. J. for Num. Meth. in Eng. 52*(1-2), 193–215.

Carol I., Rizzi E., and Willam K. (2001). On the formulation of anisotropic elastic degradation : 2. generalized pseudo-rankine model for tensile damage. *International Journal of Solids and Structures 38(4)*, 519:546.

Cehavir A., Nukada A., and Matsuoka. A. (2010). High performance conjugate gradient solver on multi-gpu clusters using hypergraph partitionings. *Computer Science - research and Development. 5*, 83–91.

Chapman B., Jost G., and van der Pas R. (2007). *Using OpenMP: Portable Shared Memory Parallel Programming*. ISBN-13:978-0-262-53302-7, Scientific and Engineering Computation.

ETH (2004). Sem for microstructure of lime mortar. *Eidgenössische Technische Hochschule (ETH) Zürich*.

Farhat C. and L. Crivelli F. R. (1994). Extending substructure based iterative solvers to multiple load and repeated analysis. *Computer Methods in Applied Mechanics and Engineering. 117*, 195–209.

Feenstra P. and de Borst R. (1996). A composite plasticity model for concrete. *International Journal of Solids Structures 33*, 707–730.

Flanagan D. and Belytschko T. (1981). An uniform strain hexahedron and quadrilateral with orthogonal hourglass control (in one-point integration of isoparametric finite element analysis). *International Journal for Numerical Method in Engineering 17*, 679–706.

Forum M.-P. I. (2008). *MPI: A Message-Passing Interface Standard, Version 2.1*. High-Performance Computing Center Stuttgart.

Galvez J., Cervenka J., Cendon D., and Saouma V. (2002). A discrete crack approach to normal/shear cracking of concrete. *Cement and Concrete Research 32*.

Garboczi E. J. (2002). Three-dimensional mathematical analysis of particle shape using x-ray tomography and spherical harmonics: Application to aggregates used in concrete. *Cement and Concrete Research 32*, 1621–1638.

GNU (2011). *gcc - gnu compiler collection*. Version 4.6.2 (SUSE Linux), Free software foundation, Inc.(c).

Golub G. H. and van der Vorst H. (2000). Eigenvalue computation in the 20th century. *Journal of Computational and Applied Mathematics 123*.

Gropp W., Lusk E., and Skjellum A. (1994). *Using MPI: portable parallel programming with the message-passing interface.* Cambridge, MA, USA:MIT Press Scientific And

Engineering Computation Series.

Gropp W., Lusk E., and Skjellum A. (1999). *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. MIT Press.

Häfner S., Eckardt S., Luther T., and Könke C. (2003). Mesoscale modeling of concrete: Geometry and numerics. *Computers and Structures 84*, 450–461.

Hannawald J. (2010). Fracture of hollow clay units simulated by sequentially linear analysis. *Modeling of heterogeneous materials (HetMat), Proceedings of the International RILEM Conference on Material Science (MatSci), W. Brameshuber (eds.), RILEM Publications S.A.R.L. II.*

Hillerborg M., Modeer M., and Peterson P. (1976). Analysis of crack formation and crack growth on concrete by means of fracture mechanics and finite elements. *Cement and Concrete Research 6*, 773–782.

HLRS (2011). *Technical description of the NEC Nehalem cluster. https://wickie.hlrs.de/platforms/index.php/NEC_Nehalem_Cluster.*

HLRS (2012). *Technical description of the CRAY XE6 cluster. https://wickie.hlrs.de/platforms/index.php/crayXE6.html.*

Jirásek M. (1998). Nonlocal models for damage and fracture: Comparison of approaches. *International Journal of Solids and Structures 36*, 4133–4145.

Jirásek M. and Patzák B. (2003). Adaptive resolution of localized damge in quasibrittle materials. *Journal of Engineering Mechanics, ASCE 130(6)*, 720–732.

Jirásek M. and Zimmermann T. (1998). Rotating crack model with transition to scalar damage. *Journal of Engineering Mechanics ASCE. 124*, 277–284.

Jog C. S. (2007). *Foundations and Applications of Mechanics: Volume I: Continuum Mechanics* (2nd ed.). CRC Press.

Karypis G. (2011). A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices, version 5.0. User Manual, Department of Computer Science & Engineering, University of Minnesota, Minneapolis.

Karypis G. and Kumar V. (1998). A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. *University of Minnesota, Department of Computer Science / Army HPC Research Center Minneapolis, MN 55455*.

Karypis G. and Kumar V. (2011). A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. *University of Minnesota, Department of Computer Science / Army HPC Research Center Minneapolis, MN 55455*.

Kelley C. T. (1995). Iterative methods for linear and nonlinear equations. *Frontiers in Applied Mathematics, SIAM, Philadelphia* (16).

Kim H. J. and Swan C. C. (2003). Voxel-based meshing and unit-cell analysis of textile composites. *International Journal of Numerical Methods in Engineering 56*, 977–1006.

Klawonn A. and Rheinbach O. (2007). *Robust FETI-DP Methods for heterogeneous three dimensional Elasticity Problems. Comput. Methods Appl. Mech. Engrg. vol 196, pp. 1400-1414.*.

Klawonn A. and Widlund O. B. (2001). Feti and neumann-neumann iterative substructuring methods: Connections and new results. *Comm. Pure Appl. Math. 54*, 5790.

Langer U. (2008). Domain decomposition methods in science and engineneering xvii. *Proceedings of the 17th International Conference on Domain Decomposition Methods at St. Wolfgang, Austria, Springer-Verlag*.

Liu G., Nguyen-Thoi T., and Lam K. (2009). Application of s-fem to the problem of composite materials with initial strain-like terms. *Journal of Sound and Vibration 320(4-5).*

Loghin D. and Wathen A. J. (2003). Schur complement preconditioning for elliptic systems of partial differential equations. *Numerical Linear Algebra with Applications 10*(Issue 5-6), 423–443.

Mandel J. (1994). Balancing domain decomposition. *Communications in Numerical Methods in Engineering 9*, 233–241.

Mazars J. and Pijaudier-Cabot G. (1989). Continuum damage theory - application to concrete. *International Journal for Numerical Methods in Engineering 115*, 345–365.

Melenk J. and Babuška I. (1996). The partition of unity finite element method: Basic theory and applications. *Computer Methods in Applied Mechanics and Engineering 39*, 289–314.

Mises R. (1913). Mechanics of structural solids in elasto-plastic deformation state. *Göttin. Nachr. Math. Phys. 1*, 582–592.

Moës N. and Belytschko T. (2002). Extended finite element method for cohesive crack growth. *Engineering Fracture Mechanics 69*, 812–833.

Möser B. (2006). Rem for the analyis of the microstructure of high-performance concrete. *F.A. Finger-Institut für Baustoffkunde (FIB), Bauhaus-Universität Weimar*.

Most T., Eckardt S., Schrader K., and Deckner T. (2006). An improved cohesive crack model for combined crack opening and sliding under cyclic loading. *In: K. Gürlebeck and C. Könke (eds.), Proc. 17th Int. Conf. on the Applications of Computer Science and Mathematics in Architecture and Civil Engineering (IKM 2006), Weimar, Germany, 12-14 July, 2006.*.

Nguyen-Thoi T., Liu G., Vu-Du H., and Nguyen-Xuan H. (2009). A face-based smoothed finite element method (fs-fem) for visco-elstoplastic analyses of 3d solids using tetrahedral mesh. *Comput. Methods Appl. Mech. Engrg. 198*, 3479–3498.

NVIDIA (2009a). *CUBLAS Library*. Nvidia corporation.

NVIDIA (2009b). *The CUDA Compiler Driver NVCC*. Nvidia corporation.

NVIDIA (2009c). *CUDA-GDB*. Nvidia corporation.

NVIDIA (2009d). *CUFFT Library*. Nvidia corporation.

NVIDIA (2009e). *Next Generation CUDA Compute Architecture: Fermi*. NVIDIA.

NVIDIA (2009f). *NVIDIA CUDA*. Nvidia corporation. Version 2.3.

NVIDIA (2009g). *NVIDIA CUDA C Programming - Best Practices Guide*. Nvidia corporation. CUDA Toolkit 2.3.

Oliver J. (1996). Modelling strong discontinuities in solid mechanics via strain softening constitutive equations. part 1: Fundamentals. part 2: Numerical simulation. *International Journal for Numerical Methods in Engineering 39*, 3575–3624.

Ozbolt J. and Bažant Z. (1996). Numerical smeared fracture analysis. *International Journal for Numerical Methods in Engineering 39*, 635–661.

Papadrakakis M., Stavroulakis G., and Karatarakis A. (2011). A new era in scientific computing: Domain decomposition methods in hybrid cpu-gpu architectures. *Computational Methods Appl. Mech. Eng. 200*, 1490–1508.

PARDISO (2007). *Parallel Sparse Direct Linear Solver - PARDISO*. Computer Science Department, University of Basel, Swiss. Version 3.2.

Perilli E. and Baruffaldi F. (2003). *Proposal for shared collections of X-ray microCT datasets of bone specimens*. Zaragoza, Spain.

Petersson P. (1981). Crack growth and development of fracture process zone in plain concrete and similar materials. *Report TVBM-100, Division of Building Materials, Lund Institute of Technology, Lund, Sweden*.

Pijaudier-Cabot G. and Bažant Z. (1987). Nonlocal damage theory. *Journal of Engineering Mechanics ASCE. 113*, 1512–1533.

Raabe D. (2004). *Continuum scale simulation of engineering materials: fundamentals - microstructures - process applications*. Wiley-VCH.

Robbins K. and Robbins S. (2003). *UNIX systems programming: communication, concurrency, and threads*. (2 ed. ed.). Prentice Hall.

Rots J. (1988). *Computational modeling of concrete fracture*. Ph. D. thesis, Delft University of Technology, The Netherlands.

Rots J. (2001). Sequentially linear continuum model for concrete fracture. *A.A. Balkema Publisher*, 831–839.

Rots J. G., Belletti B., and Invernizzi S. (2006). Event-by-event strategies for modelling concrete structures. *Computational Modelling of Concrete Structures, Proceedings of EURO-C 2006 Meschke, de Borst, Mang and Bicanic (eds), Taylor and Francis Group, London*, 667–678.

Rots J. G. and Invernizzi S. (2004). Regularized sequentially linear saw-tooth softening model. *Journal for Numerical and Analytical Methods in Geomechanics 28*, 821–856.

Rots R. (1992). Softening of concrete loaded in compression. *PhD thesis, TU Eindhoven*.

Schlangen E. and van Mier J. (1992). Experimental and numerical analysis of micromechanisms of fracture of cement-based composites. *Cement & Concrete 14*, 105–118.

Schloegel K., Karypis G., and Kumar. V. (2002). Parallel static and dynamic multiconstraint graph partitioning. *Concurrency and Computation: Practice and Experience 14*, 219–240.

Schrader K. (2005). Algorithmic implementation of an elasto-plastic interface model for discrete crack-face degradation of cohesive cracks. *Diploma thesis, Institute of Structural Mechanics, Bauhaus-Universität Weimar*.

Schrader K. and Könke C. (2011). Hybrid computing models for large-scale heterogeneous 3d microstructures. *International Journal for Multiscale Computational Engineering 9*(4), 365–377.

Schrader K. and Könke C. (2013a). Distributed computing for the nonlinear analysis of multiphase composites. *Advances in Engineering Software 62-63*, 20–32. Special Issue dedicated to Professor Zdenék Bittnar on the occasion of his Seventieth Birthday: Part I.

Schrader K. and Könke C. (2013b). Distributed fe analysis of multiphase composites regarding 3d elasticity problems. In Nagel W. E., Kröner D. H., and Resch M. M. (Eds.), *High Performance Computing in Science and Engineering '12*, pp. 531–545. Springer Berlin Heidelberg.

Schwarz H. (1890). *Gesammelte mathematische Abhandlungen.* Springer Verlag, Berlin, 2. Edition.

Schwarz H. and Koeckler N. (2004). *Numerische Mathematik*. Teubner Verlag Stuttgart.

Shan Z. and Gokhale A. M. (2004). Digital image analysis and microstructure modeling tools for microstructure sensitive design of materials. *International Journal of Plasticity 20*, 1347–1370.

Simo J. and Hughes T. (1998). *Computational Inelasticity*. Springer-Verlag.

Simo J. and Oliver J. (1994). A new approach to the analysis and simulation of strain softening in solids. In *Fracture and Damage in Quasibrittle Structures*.

Sukumar N., Chopp D., Moes N., and Belytschko T. (2001). Modeling holes and inclusions by level sets in the extended finite elment method. *Computation Methods in Applied Mechanics and Enginnering 190*, 6183–6200.

Sukumar N., Huang Z., Prevost J., and Suo Z. (2004). Partition of unity enrichment for bimaterial interface cracks. *International Journal for Numerical Methods in Engineering 59*, 1075–1102.

Sukumar N., Moes N., Moran B., and Belytschko T. (2000). Extended finite element method for three-dimensional crack modeling. *International Journal for Numerical Methods in Engineering 48*, 1549–1570.

Tanaka S., Okada H., Watanabeand Y., and Wakatsuki T. (2006). Application of s-fem to the problem of composite materials with initial strain-like terms. *International Journal of Multiscale Computational Engineering 4(4)*.

Toselli A. and Widlund O. (2005). Domain decomposition methods - algorithms and theory. *Springer series in computational mathematics, Springer-Verlag, Berlin, Germany 34*.

Červenka J. (1994). *Discrete crack modeling in concrete structures*. Ph. D. thesis, University of Colorado, Boulder, Colorado.

Voormeeren L. (2011). Appendix of the master thesis: Sla for notched beams in 2d and 3d. *Delft University of Technology, Faculty of Civil Enginnering, section of Structural Mechanics, Delft, The Netherlands*.

Wang Z. (2009). *Optimization of the parameterized Uzawa Preconditioners for Saddle Point Matrices. Journal of Computational and Applied Mathematics, vol. 226, pp. 136-154, Issue 1. 2009*.

Wang Z. M., Kwan A. K. H., and Chan H. C. (1999). Mesoscopic study of concrete 1: generation of random aggregate structure and finite element mesh. *Computers and Structures 70*(5), 533–544.

Wriggers P. (2008). *Nonlinear finite element methods*. Springer Verlag.

Wriggers P. (2009). *Mixed Finite Element Methods - Theory and Discretization*, Volume 509 of *CISM International Centre for Mechanical Sciences*. Springer Verlag.

Wriggers P. and Moftah S. O. (2006). Mesoscale models for concrete: Homogenisation and damage behaviour. *Finite Elements in Analysis and Design 42*, 623–636.

Appendix A

MDiSP C Library

MDiSP: *Multiple Domain iterative Solving Procedures*

The following description of selected numerical functions in the C programming language (sequantial and MPI-parallelized) are included in the MDiSP C Library created during this work, which were used for all numerical experiments and results, for the benchmarking and for the evaluation of their MPI-scaling within the hybrid high-performance computing framework. The appendix contains:

- I/O data based function calls: Functions for reading the finite element (mesh) data stored in a special (ASCII) format.
- MPI based function calls: MPI-functions involving all started MPI-processes.
- Saw-tooth softening material model based function calls.

A.1 I/O data based function calls

int **FE_READ_EXAMPLE**(	FE_STRUC *myFE, int size, char *dirname, char *example, char *incname, char *mshtype char *hextype char *command char *argv[] char *ndsname char *metis char *metis_v char *output_file)
Objective	Reading the FE mesh and example data for the distributed analysis.
Parameters	*myFE* - structure containing all parameters describing the fe model *dirname* - folder of files *example* - name of the FE example (subfolder) *mshtype* - type of mesh (HEX, TET, HYBRID) *metis* - graph type (DUAL or NODAL) *size* - number of MPI processes

A.2 MPI based function calls

int **GET_BLOCK_MAX_NUM_ij_1**(	FE_STRUC *myFE, int myid, 0)
Objective	First computation and allocation of number global FE blocks for each subdomain among all MPI processes.
Parameters	*myFE* - structure containing all parameters describing the fe model *myid* - id of MPI process *0* - optional

int **GET_BLOCK_MAX_NUM_ij_2**(	FE_STRUC *myFE, int myid, 0)
Objective	Second computation and allocation of number global FE blocks for each subdomain among all MPI processes.
Parameters	*myFE* - structure containing all parameters describing the fe model *myid* - id of MPI process *0* - optional

int **FE_DATA_DISTR_SCALAR**(	FE_STRUC *myFE, MPI_Comm *comm, MPI_Stat stat)
Objective	Distributes finite element data from the MPI host process to all remaining MPI processes.
Parameters	*myFE* - structure containing all parameters describing the fe model *comm* - MPI communicator MPI_COMM_WORLD *stat* - MPI status

int **FE_DATA_DISTR_VECTOR**(	FE_STRUC *myFE, MPI_Comm *comm, MPI_Stat stat)
Objective	Distributes finite element data from the MPI host process to all remaining MPI processes.
Parameters	*myFE* - structure containing all parameters describing the fe model *comm* - MPI communicator MPI_COMM_WORLD *stat* - MPI status

int **BUILD_GLOBAL_MATRIX_MPI**(	FE_STRUC myFE, int myid, int swDOF)
Objective	Simultaneous assembly of global stiffness matrix for each subdomain among all MPI processes.
Parameters	*myFE* - structure containing all parameters describing the fe model *myid* - id of the MPI process *swDOF* - considering nodal d.o.f.s (1 - yes, 0 - no)
int **GET_PRECONDG_SCALING**(	FE_STRUC *myFE, char *mshtype, double ev_tol, int myid, 0)
Objective	Eigenvalue based scaling of the modified Jacobi-Point preconditioning matrix.
Parameters	*myFE* - structure containing all parameters describing the fe model *ev_tol* - accuracy for the eigenvalue iteration *mshtype* - type of mesh (HEX, TET or HYBRID) *myid* - id of the MPI process *0* - optional
int **BUILD_GLOBAL_RHS_MPI**(	FE_STRUC *myFE, int myid, 0)
Objective	Build global right hand side from the applied load (surface tractions or deadload)
Parameters	*myFE* - structure containing all parameters describing the fe model *myid* - id of the MPI process *0* - optional

int **GET_SOLUTION_DISP_MPI**(	FE_STRUC *myFE, int myid, int *out)
Objective	Get displacement solution of the global distributed system of equations.
Parameters	*myFE* - structure containing all parameters describing the fe model *myid* - id of the MPI process *out* - write solution to file (0 - no, 1 - yes)

int **CG_SOLVE_NDCSR_MPI**(	FE_STRUC *myFE, MPI_Comm *MPI_COMM_WORLD, MPI_Stat *stat, int *out, int *alloc)
Objective	Preconditioned MPI-based CG solver using *ndcsr* matrix storage.
Parameters	*myFE* - structure containing all parameters describing the fe model *MPI_COMM_WORLD* - communicator including all MPI processes *stat* - pointer to the MPI status *alloc* - allocate PPCG vectors (0 - no, 1 - yes) *out* - write solution to file (0 - no, 1 - yes)

A.3 Saw-tooth softening material model based function calls

int **PERFORM_SLA_ELEM_MPI**(	FE_STRUC *myFE, int myid, int out, int cycles_out, int sw)
Objective	Scalable SLA procedure based on MPI.
Parameters	*myFE* - structure containing all parameters describing the fe model *myid* - id of MPI process *out* - write solution to file (0 - no, 1 - yes) *cycles_out* - write solution of cycles to file (0 - no, 1 - yes) *sw* - optional

int **GET_SLA_ELEM_STIFFNESS_CRIT**(	FE_STRUC *myFE, int myid, int swDOF, int reduce)
Objective	MPI-based update procedure for the critical element identified from the scalable SLA procedure and update of the global distributed equation system of the subdomain including the critical element.
Parameters	*myFE* - structure containing all parameters describing the fe model *myid* - id of MPI process *reduce* - optional