The Deflated Preconditioned Conjugate Gradient Method Applied to Composite Materials

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. dr. ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
woensdag 1 februari 2012 om 15:00 uur

door

Tom Bernard JÖNSTHÖVEL
wiskundig ingenieur

geboren te Dordrecht.

Dit proefschrift is goedgekeurd door de promotor:
Prof. dr. ir. C. Vuik.

Copromotor: Dr. A. Scarpas
Copromotor: Dr. ir. M.B. van Gijzen

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof. dr. ir. C. Vuik,	Technische Universiteit Delft, promotor
Dr. A. Scarpas,	Technische Universiteit Delft, copromotor
Dr. ir. M. B. van Gijzen,	Technische Universiteit Delft, copromotor
Prof. dr. R. H. Bisseling,	Universiteit Utrecht, the Netherlands
Prof. dr. M. Papadrakakis,	National Technical University of Athens, Greece
Prof. dr. ir. C. W. Oosterlee,	Technische Universiteit Delft
Prof. dr. T. A. Laursen	Khalifa University, Abu Dhabi
Prof. dr. ir. A. W. Heemink	Technische Universiteit Delft, reservelid

The work described in this dissertation was carried out at the Department of Structural Mechanics, the Faculty of Civil Engineering, and, at the Department of Applied Mathematical Analysis, the Faculty of Information Technology and Systems, Delft University of Technology, the Netherlands.

The Deflated Preconditioned Conjugate Gradient Method
Applied to Composite Materials.
Dissertation at Delft University of Technology
Copyright © 2012 by T.B. Jönsthövel

ISBN 978-94-6191-180-3

– All of old. Nothing else ever. Ever tried. Ever failed. No matter. Try again. Fail again. Fail better.

from Worstward Ho by Samuel Beckett

– This 'shuddering before the beautiful', this incredible fact that a discovery motivated by a search after the beautiful in mathematics should find its exact replica in Nature, persuades me to say that beauty is that to which the human mind responds at its deepest and most profound.

from Truth and beauty by S. Chandrasekhar

– High-quality, useful energy, is localized energy. Low quality, wasted energy, is chaotically diffuse energy. Things can get done when energy is localized; but energy loses its potency to motivate change when it has become dispersed.

from Creation revisited by Peter Atkins

I WOULD LIKE TO THANK

Pieter van Rijn Jos Jönsthövel Alian Hardenberg Bettie Jönsthövel-Geers Yora Rienstra Michiel van Werven Wouter van Berkel Anne van Rijn Evelien de Weger Leonie Meine Jansen Tom Schneider Leon Schmitz Louise Korthals Sophie Plimpton Steven Ford Brown Boubadioup (Schelto Jochem Jasper Jeroen Tommy Jan-Joost) Scott MacLachlan Martin van Gijzen Kees Vuik Tom Scarpas Cor Kasbergen Xueyan Liu Mirrella Villani Alexander Schmets Tim Mulders Iris Jönsthövel Marianne Geers Hans Geers Nils Jönsthövel Suzanne Bruins Femke Bruins Cas Jönsthövel Stan Klerks Manolis Papadrakakis Frank Custers Markus Oeser Niki Kringos Maarten Sitvast Flip & Marianne Bruins Tijn Brands Wilmar van de Zande Fleur Schermer Mert Erdem Tod Laursen Rob Bisseling Kees Oosterlee Paul Childs Yok Tang Arnold Heemink

Acknowledgements

It has been a remarkable four years. I remember walking into Kees Vuik's office after an unsuccessful year as an IT consultant and financial expert. With hindsight I realize returning to TU Delft was less about the failure in two professions that made me feel so welcome here than the particular time in my life and realization there was still much more to learn as an academic In short, during work on my Master's Degree I glimpsed the world of numerical analysis, but I could hardly see what was behind the horizon. Apart from the theoretical results on deflation theory, I can conclude that the years of my PhD studies have been the most (trans)formative years of my life and have truly taken my level of professional skills and expertise to another level. Moreover, I have in those years literally travelled the globe and connected to people from inside and outside academics, of which some have become good friends. I cannot imagine feeling more complete and being utterly satisfied with my life if it weren't for those challenges, chances and opportunities that this project has given me.

My PhD research was sponsored and initiated by Tom Scarpas, the head of the Group of Infrastructural Materials at the Department of Civil Engineering at TU Delft. I am deeply indebted to him for letting me have the freedom to work on this project under excellent working conditions, with solid financial as well as infrastructural support, and with wonderful colleagues. Next, I would like to thank Cor Kasbergen for his never ending support, his patience and kindness. He has shown me the way in the ancient worlds of FORTRAN and non-object oriented programming. I can't imagine finishing this project in the same period of time without his help. I would also like to thank Xueyan Liu for his never ending stream of questions and for keeping me in touch with my recently acquired Asian side. My sincerest respect goes to Frank Custers. He has done a terrific job of setting up the computing cluster with minimal resources and limited time. He has been a valuable sparring partner, and I enjoyed our discussions on how to improve the performance and capabilities of the cluster.

The other two persons who played a crucial part in this research are of course my supervisors Kees Vuik and Martin van Gijzen. I enjoyed our collaboration and I am sure I will often think about those meetings to discuss the exciting and better than expected numerical results. You have been great tutors and mentors to me and I am very grateful to both of you for showing me the way to academic maturity and giving me confidence to become an independent researcher with a truly open mind. I hope we can continue to work together on exciting future research projects that may lead to yet more high-level academic publications. As mentioned before, I have been given the opportunity to travel around the globe to talk about deflation at seminars and conferences, and interact with fellow researchers for longer periods of time during my extended stays in Boston, USA and Sydney, Australia. I would like to thank Scott MacLachlan for a month of truly inspiring research at Tufts University, USA. He has also helped me reach a higher level of thinking and showed me alternate career paths that could lead to a continuation of my academic career.

In Boston I was hosted by Steven Ford Brown, an inspiring writer, music critic, and above all translator of poetry from nearly forgotten foreign poets. He has been great company and I truly enjoyed our discussions on (American) politics, music and life. I would also like to thank him for his textual comments and suggestions.

In Sydney I was welcomed by Markus Oeser who offered me a place to work at the University of New South Wales during my extended stay-over on my trip to China. I would like to thank him for his kind help and for giving me an insider's view to the peculiarities of the Australian way of life.

In addition to my academic career I have pursued the dream of a career in music. In the last ten years I have developed a successful career as a pianist in a band with a record deal, as a composer of music soundtracks for film, as well as an accompanying pianist in a theatre tour. Moreover, I have had the chance to collaborate with dozens of professional musicians who have ultimately not only raised my level of playing, but my level of thinking about music as well. It is no secret that mathematics and music are closely related and I consider them both as art. Although my thoughts on this subject haven't matured enough to be entrusted to paper, I have only managed in recent years to connect those two sides of my personality and let them be mutually influential. Proper research is not possible without giving your thoughts the chance of settling down, and I can't imagine any better way of nurturing my ideas about new research than when working on music. I would therefore like to thank my creative friends for their, probably, unintentional support to this research. Especially, Pieter, Anne, Mert and, Louise.

Without the support of my dearest friends, at this time of writing, I would be lying under a Parisian bridge, drinking bad wine and singing (out-of-tune) silly songs. Wilmar and Tijn. You guys have both been there when I needed you the most, offering me a place to sleep, listening to my badly told stories and anecdotes as well as enduring all my 'exciting' but hopeless ideas. You know I love you guys.

Although inevitably our lives took a different direction, I don't believe I would be the person I am today, both academically and personally, without sharing with her that special period of my life. Thank you, dear Suzanne.

Inspired by a book of Alain de Botton, not long ago I decided I would enjoy life better by being alone. And although Alain also warned me something untoward might happen, and although it may or may not be a coincidence that her name is an anagram of Alain, this concept of being alone suffered a severe blow when I met her. De Buurvrouw. Because of her, life has taken another unexpected turn, sweet Alian.

I would like to dedicate my academic achievements to my wonderful parents, Jos and Bettie, who I am indebted to for their unceasing support, encouragement and love. I also want to thank my lovely sister Iris. For the joy you bring in life, your enthusiasm, creativity and never-ending stream of inspiring ideas. And at last, my older brother Nils. Although our characters could not be more different, I love you nonetheless, and most of all your humor and contagious habit of putting things in perspective.

Finally, I would like to thank all the members of the doctoral examination committee for their time and effort and I would like to thank all of those that have been of invaluable support to my successful completion of this research.

Tom Jönsthövel, Amsterdam, February 2012.

Contents

Acknowledgements v

1 Introduction 1
- 1.1 Background composite materials . . . 1
- 1.2 Shifting length scales: from meso to micro level . . . 1
- 1.3 Simulation tools for pavement engineering . . . 2
- 1.4 Solving the stiffness matrix: numerical solution methods . . . 4
- 1.5 Scope of the thesis . . . 5
- 1.6 Outline of the thesis . . . 6

2 Structural mechanics 9
- 2.1 Continuum model . . . 9
  - 2.1.1 Strain . . . 10
  - 2.1.2 Stress . . . 11
- 2.2 Equilibrium equation . . . 12
- 2.3 Balancing forces . . . 14
- 2.4 Material response . . . 16
  - 2.4.1 Elasticity . . . 16
  - 2.4.2 Plasticity . . . 16
  - 2.4.3 Viscosity . . . 18
- 2.5 Implementation of material response . . . 19
  - 2.5.1 Dissipation of energy . . . 19
  - 2.5.2 Multiplicative decomposition . . . 19
  - 2.5.3 Generalized model local dissipation . . . 21
  - 2.5.4 Hyperelastic response . . . 22
  - 2.5.5 Plastic response . . . 22
  - 2.5.6 Viscoelastic response . . . 25

3 Discretization virtual work equation 29
- 3.1 Discretization of the linearized virtual work equation . . . 29
  - 3.1.1 Finite-Element method . . . 30

Element type and shape functions 30
Gauss points and numerical integration 31
3.1.2 Stiffness matrix 32
Static mechanics 32
Dynamic mechanics 34
3.2 Stiffness matrix for composite materials 36
3.2.1 General properties of stiffness matrix 36
3.2.2 Discontinuities entries stiffness matrix 36
3.2.3 Non-linear material properties 37
3.3 Concluding remarks 38

4 Solving the linear system: overview of solution methods 39
4.1 Factorization methods 39
4.1.1 LU factorization 40
Cholesky factorization 41
Software implementation: MUMPS 41
4.1.2 Incomplete factorization 42
ILU(0) decomposition 42
Software implementation: ILUPACK 42
4.2 Multigrid 43
4.2.1 Basic multigrid 44
4.2.2 Multigrid Components 45
Coarse grid specification 45
Smoother 46
Restriction operator 46
Prolongation operator 46
4.2.3 Algebraic multigrid 47
Smoothed aggregation 48
4.2.4 Software implementation: ML (Trilinos) 48
4.3 Krylov subspace methods 51
4.3.1 Conjugate Gradient method 51
4.3.2 Preconditioning 53
4.3.3 Preconditioned Conjugate Gradient method 53
Diagonal scaling 54
ILU decomposition 55
SA-AMG 55
4.3.4 Software implementation: AztecOO (Trilinos) 56
4.4 Concluding remarks 56

5 **Deflation theory** **57**
- 5.1 Composite materials: Preconditioned CG 57
  - 5.1.1 Motivating numerical experiments 58
    - Convergence results 59
- 5.2 Introduction to deflation 63
  - 5.2.1 Deflated Preconditioned CG 63
- 5.3 Rigid body modes deflation 65
  - 5.3.1 Motivation: rigid bodies in composite materials 65
  - 5.3.2 Construction of deflation vectors 66
  - 5.3.3 Computing rigid body modes of a finite element 68
  - 5.3.4 Additional work DPCG compared to PCG 70
  - 5.3.5 Illustrative example: artificial representation of composite material 71
- 5.4 Recursive deflation 75
  - 5.4.1 Motivation: condition numbers of Deflated matrices 77
  - 5.4.2 Recursive Deflation strategy 78
    - Illustrative example: 1D Poisson equation 81
  - 5.4.3 Deflation vectors in the neighborhood of a jump 82
    - Illustrative example: 1D Poisson equation (**continued**) 83

6 **Parallel implementation deflation** **87**
- 6.1 Parallel computing 88
  - 6.1.1 Domain decomposition 88
    - Subdomain mapping operators 89
    - Local and global stiffness matrix 90
    - Local and global vectors 91
    - Parallel matrix-vector product 91
    - Parallel inner product 92
- 6.2 Parallel Deflated Preconditioned Conjugate Gradient method 92
  - 6.2.1 Building-blocks of parallel DPCG 92
    - Parallel coloring algorithm: construction and identification of rigid bodies distributed over subdomains 92
    - Computing rigid body modes in parallel 96
    - Efficient computation of P in parallel 97
- 6.3 Subdomain deflation 98
- 6.4 Parallel implementation of other solvers 101
  - 6.4.1 Parallel PCG method 101
  - 6.4.2 Parallel SA-AMG 101

7 Numerical examples 103
7.1 Description of cases . . . 103
7.2 Description of hardware and software . . . 107
7.3 Rigid body mode deflation . . . 108
7.4 Subdomain deflation . . . 112
7.5 Performance DPCG and comparison to other state-of-the-art methods 117
  7.5.1 CAPA-3D: DPCG and MUMPS . . . 117
  7.5.2 Trilinos: DPCG and ML (SA-AMG) . . . 119
7.6 Concluding remarks . . . 126

8 Application to advanced material models 129
8.1 Generalized elasto-visco-plastic material model . . . 129
8.2 Numerical experiments . . . 131
8.3 Concluding remarks . . . 134

9 Summary and Conclusions 135

10 Future research 143
10.1 On the improvement of deflation vectors . . . 143
  10.1.1 Assessment of quality deflation vectors . . . 143
  10.1.2 Polynomial updates of deflation vectors . . . 143
10.2 Computing and implementation . . . 145
  10.2.1 Recycling of deflation vectors . . . 145
  10.2.2 GPU computing . . . 146
  10.2.3 Nonsymmetric matrices . . . 146
  10.2.4 Reducing the number of deflation vectors . . . 147
  10.2.5 Efficient computation with deflation vectors . . . 147

A Notation 149

Bibliography 158

List of publications 159

Curriculum Vitae 161

List of Figures

1.1 Tomography scan of slice of asphaltic material. 3

2.1 Mapping of relative position vector from the reference to the current configuration 10
2.2 Traction **t** acting on an infinitesimal element da 12
2.3 Simplified representation of elastic material. 17
2.4 Example of relation between strain and stress. 18
2.5 Example of plastic yield surface. 18
2.6 Simplified example of 1D viscous laminar flow. 19
2.7 Schematic representation of multiplicative decomposition. 20

3.1 Finite element mesh applied to computer model of asphalt column. . . 29
3.2 Tetrahedral element with local coordinate system (ξ, η, ζ) 30

5.1 Schematic 2D representation of 3D test cases (II) and (III), figure (c) and (d) contain one and four aggregates respectively. 59
5.2 Deterioration of rates of convergence of PCG for increasing number of aggregates and stiffness. -.- homogeneous material, – E ratio $\mathcal{O}(10^3)$, – E ratio $\mathcal{O}(10^5)$. 60
5.3 50 smallest eigenvalues of $M^{-1}K$, $M = \text{diag}(K)$. $\times$: 1 aggregate, $\diamond$: 4 aggregates, $\circ$: 8 aggregates, $*$: homogeneous material 61
5.4 Principle of kernel deflation 67
5.5 (a) spherical coordinates, (b) rotation around origin of tetrahedral element in x, y-plane 69
5.6 Nonzero pattern of matrix $K_Z = KZ$. 70
5.7 Convergence of PCG and kernel DPCG for 1 and 4 aggregates respectively. -.- PCG (homogeneous material), – PCG ($E \approx \mathcal{O}(10^3)$), – kernel DPCG ($E \approx \mathcal{O}(10^3)$). 73
5.8 Convergence of PCG and kernel DPCG for 1, 4 and 8 aggregates respectively. -.- PCG (homogeneous material), – PCG ($E \approx \mathcal{O}(10^3)$), – PCG ($E \approx \mathcal{O}(10^5)$),– DPCG ($E \approx \mathcal{O}(10^3)$), – DPCG ($E \approx \mathcal{O}(10^5)$). 74

5.9 spectrum of $M^{-1}K$ where $[c_1, c_2, c_3] = [1, 10^4, 10^8]$. 82
5.10 sparsity pattern C_0, C_1 and C_2. Nonzero elements represented by symbols; corresponding to deflated material, interface elements and remaining elements pictured by bold crosses, circles and non bold crosses respectively. 83
5.11 spectrum of $M^{-1}C_i$ ($\star$ correct, $+$ wrong choice deflation vectors) compared to spectrum of $M^{-1}K$ $(+)$ 84
5.12 Convergence of DPCG and PCG where $[c_1, c_2, c_3] = [1, 10^4, 10^8]$ and DPCG$^+$, DPCG$^-$ represent correct and wrong choice of deflation vectors respectively. 85

6.1 Schematic representation of domain decomposition of composite material consisting of five bodies 89
6.2 non-zero pattern for local stiffness matrix, K_1, and, global stiffness matrix, K. 90
6.3 Assembled global vector, $\mathbf{u} \in \mathbb{R}^n$. 91
6.4 Rigid bodies divided over subdomains. 94
6.5 Deflation matrix Z, divided over subdomains. 96
6.6 Sparsified global stiffness matrix, $\hat{K}$; no connectivity between nodes in subdomains of domain decomposition except for boundary nodes. . 98
6.7 Non-zero pattern subdomain deflation vectors, $Z_{\mathrm{sd}} \in \mathbb{R}^{n\times(4\times6)}$, for domain decomposition with 4 domains. 99
6.8 Overlap between subdomains, and the corresponding entries in truncated matrix, $\hat{K}$. 100

7.1 the artificial case: FE mesh and schematic representation of cylinder containing three aggregates represented by spheres. 104
7.2 the asphalt core case: FE mesh representing core of asphaltic material containing aggregates (yellow), bitumen (red) and air voids (blue). . . 105
7.3 the cube of asphalt case: mesh representing cube of asphaltic material containing aggregates (light green), bitumen (dark green) and air voids (blue). 106
7.4 the artificial case: Convergence of PCG and DPCG (bold line) for cylinder containing three aggregates 109
7.5 the asphalt core case: deflation strategy, identify sets of elements corresponding to material: (a) aggregates, (b) bitumen and (c) air voids.110
7.6 the asphalt core case: Convergence of PCG and DPCG for a real slice of asphaltic material 111
7.7 the asphalt core case: L_2 norms of the residuals for DPCG without 7.7(a), and, with 7.7(b) subdomain deflation for 2, 8, 16, and, 32 subdomains. 113

7.8 the cube of asphalt case: L_2 norms of the residuals for DPCG without 7.8(a), and, with 7.8(b) subdomain deflation for 8, 16, 32, and, 64 subdomains. 114
7.9 the cube of asphalt case: number of nonzeros in matrix, Z, and the relative increase of nonzeros in matrix, KZ, using subdomain deflation for 8, 16, 32, and, 64 subdomains. 116
7.10 the cube of asphalt case: Cumulative timing all stages of DPCG (IC preconditioner) for 8, 16, 32 and 64 subdomains. 118
7.11 the asphalt core case: Cumulative timing all stages of PCG, DPCG for 4, 16, and, 64 subdomains. 121
7.12 the asphalt core case: L_2 norms of the residuals 122
7.13 the asphalt core case: Ritz values derived from (D)PCG 122
7.14 the cube of asphalt case: Cumulative timing all stages of PCG, DPCG for 4, 8, and, 64 subdomains. 124
7.15 the cube of asphalt case: L_2 norms of the residuals 125
7.16 the cube of asphalt case: Ritz values derived from (D)PCG 125

8.1 Loading function creep test, 150 load steps. 131
8.2 Total strain in y-direction of elasto-visco-plastic element. 132
8.3 Plastic strain in y-direction of elasto-visco-plastic element. 132
8.4 Number of DPCG iterations per internal Newton-Raphson iteration. . 133
8.5 Boxplot of the number of DPCG iterations for all load steps (equals 2540 Newton-Rahpson iterations). 133

10.1 Two bodies, Ω_a, and, Ω_b, in finite element mesh. 146

List of Tables

4.1 Default parameters for SA-AMG 50

5.1 2304 elements, the extreme eigenvalues and condition number of preconditioned stiffness matrices. $\mathcal{O}\left(10^n\right)$ represents the jump in E modulus of aggregates and bitumen. 62

6.1 Local number of connected rigid body modes in neighboring domains 95
6.2 Mapping temporary global numbering rigid body modes to local numbering. 95
6.3 Temporary set of rigid bodies and definite global numbering. 95

7.1 Young's moduli for different materials 103
7.2 Young's modulus for different materials 108
7.3 the artificial case: wall clock time(s) PCG and DPCG 109
7.4 the asphalt core case: wall clock time(s) PCG and DPCG 111
7.5 the cube of asphalt case: wall clock time(s) DPCG for subdomain deflation 115
7.6 the cube of asphalt case: wall clock time(s) DPCG and MUMPS for 8, 16, 32, and, 64 subdomains 118
7.7 the asphalt core case: Wall clock time and number of iterations of (D)PCG 120
7.8 the cube of asphalt case: Cumulative timing all stages of PCG, DPCG for 4, 8, and, 64 subdomains. 123
7.9 the asphalt core and the cube of asphalt case: wall-clock times (s) using SuperLU, distributed memory version 2.5 126

8.1 Parameters of elasto-plastic and visco-elastic components of elasto-visco-plastic material model. 130

A.1 Tensor and index notation 150
A.2 Basic tensor computations. 150

List of Algorithms

1 Balancing forces: solving the non-linear virtual work equation 16
2 Balancing forces: solving the discretized non-linear virtual work equation 35
3 Gaussian elimination algorithm 41
4 ILU(0) algorithm 43
5 Conjugate gradient algorithm 52
6 Preconditioned conjugate gradient algorithm 54
7 Deflated preconditioned CG solving $K\mathbf{u} = \mathbf{f}$ 64
8 Identification of rigid bodies in FE mesh 93

1

Introduction

1.1 Background composite materials

In many real-life applications, large scale engineered structures make use of composite materials. These are materials which consist of a mixture of components that form one integrated body. Composite materials have their own specific material properties which (may) differ from the material properties of each individual component.

The current trends towards a more sustainable use of resources at the one hand, and ever increasing demands on the durability and safety of engineered structures on the other hand, require a new, multidisciplinary approach towards the engineering practice. By enabling the visualization of the internal distributions of stresses and strains in a body of material, we have a valuable tool in understanding the mechanisms and processes leading to material deterioration. In addition, this approach enables the quantification of the interaction between the material and the geometric characteristics of the body.

1.2 Shifting length scales: from meso to micro level

Until recently, because of the extremely long execution time, memory and storage space demands, the majority of computer simulations of composite materials such as asphalt concrete, rocks and cement concrete were performed by means of homogenization techniques. New, advanced characterization techniques elucidate the fundamental physico-chemical interactions and processes that govern the macroscopic behavior of composite materials. The macroscopic response of structures to external influences such as forces, moisture or temperature gradients, should be understood and prone to design from the more fundamental material levels upward. Hence, component interaction is the most critical factor in determining the overall mechanical response of the composite material and, that, by being able to control and specify the characteristics of the interaction, the material designer can not only optimize the mechanical performance but, also, tailor the short and long term response to address specific environmental and/or loading demands.

In the framework of this thesis, we focus on the composite material asphalt concrete in the context of pavement engineering. Asphalt concrete consists of a mixture of bitumen, aggregates and air voids, which have different materials properties. We use material models which combine the elastic, plastic, and, viscous material properties to simulate the mechanical response and component interaction. In pavement engineering, these three material properties determine the volumetric shape of the material, which, in general, accounts for the condition and usability of the pavement.

Naturally, the difference between the stiffness, which is a function of the elastic, plastic, and, viscous material properties, of the air voids, bitumen, and, the aggregates is significantly large, especially at higher temperatures. Moreover, plastic and viscous behavior is not likely to be observed for stone aggregates under room temperatures. There are many benefits when modeling asphalt concrete as a composite since then aging effects, cracking and deterioration of the material due to moisture penetration can be carefully modeled thus providing the desired insight as to why material properties degrade or improve when the individual components of the composite change. This leads to an integrated material design; a synergy between laboratory tests and computer simulations, which reduces the need for extensive and expensive testing of the material. Different models and parameters for elasticity, plasticity, and viscosity can be applied to the different materials of the mixture. Changing the amount of aggregates in the mix does not influence the elastic, plastic and viscous material parameters of the bitumen.

Even though numerical simulation of asphalt concrete at micro scale has strong physical advantages, there are also trade offs to be considered. First of all, the construction of the computational mesh is much more involved compared to modeling homogeneous materials. After we make tomography scans of samples of materials fine meshes have to be constructed, representing the geometry, the location and the mechanical characteristics of the individual components. Typically this can be achieved by means of Computed Tomography (CT) X-ray scans. Figure 1.1 shows a typical CT scan of a slice of a cylinder of asphalt concrete. The aggregates, the bitumen and the air voids are clearly visible. Many successive CT slices are necessary for construction of high quality three-dimensional surface renderings by means of specialized software tools like Simpleware ScanFE [77]. Then, additional software like CUBIT [73] is necessary for the generation of three-dimensional material-per-element meshes.

1.3 Simulation tools for pavement engineering

We simulate the mechanical response of composite materials by means of the mathematical framework developed by Malvern, Desai [61, 26], which is extended and improved by Holzapfel, Lurie, Scarpas, and, Schellekens [58, 42, 74, 75].

In this thesis we use the large deformation, continuum formulation proposed in [74]. The mathematical framework has been implemented into the computational platform

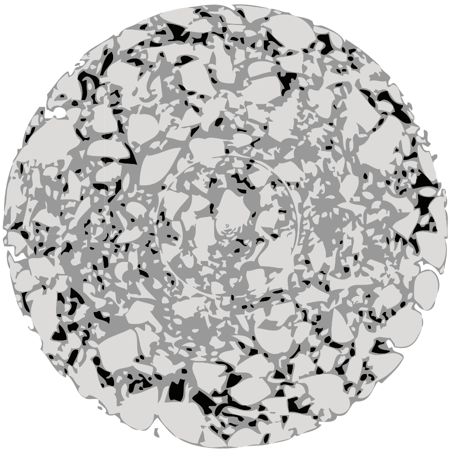


Figure 1.1: Tomography scan of slice of asphaltic material.

CAPA-3D, by the group of Mechanics of Infrastructural materials of the Faculty of Civil Engineering and Geosciences of TU Delft. The software is designed for static or dynamic analysis of very large scale three-dimensional pavement and soil engineering models. It consists of a sophisticated user interface, a powerful band-optimizing mesh generator, high quality user-controlled graphical output, several material and element types, and a variety of specialized algorithms for the more efficient analysis of pavement constructions. Among others, these include a moving load simulation algorithm and a contact algorithm.

The underlying mathematical framework involves (highly) non-linear constitutive (material) models which are coupled to the mechanical response by the laws of energy conservation, and, dissipation. This formulation leads to three-dimensional coupled, partial differential equations (PDEs) which are linearized, and, subsequently, discretized by the Finite Element (FE) method. The linearization leads to the stiffness matrix, which represents the internal stress of the materials. As the CT scans give rise to high-quality, very fine meshes, the stiffness matrix involves millions of degrees of freedom. In general the stiffness matrix is symmetric positive definite (SPD), hence is non-singular and, thus, the resulting system of linear equations has an unique solution. Moreover, the stiffness matrix is sparse.

The efficient solution of linear systems with the stiffness matrix is the key to realistic simulations and leads to the main question of this thesis.

How to built an efficient solver for the linear systems resulting from the simulation of the mechanical response of composite materials?

1.4 Solving the stiffness matrix: numerical solution methods

In the framework of this thesis, the FE computations in CAPA-3D involve the simulation of an *inhomogenous* material, where the difference in properties of materials leads to large differences in the entries of the resulting stiffness matrix. Modern iterative and direct solvers are capable of handling these large systems of equations which was unthinkable ten years ago. Still, even with the continuously increasing power of CPUs and the introduction of multiple computing cores on one CPU, the demand for efficient, parallel computing algorithms is higher. Well proven techniques like the Krylov subspace methods, multigrid and direct solution methods are mashed up in all forms of hybrid (iterative) solvers. Due to the availability of open source software, linear solvers have become available for every engineer with a modern desktop computer. Unfortunately, as mixing different medicine to cure an unknown disease is not likely to work, randomly mashing up numerical methods is also no guarantee for an efficient, fast and moreover robust solver. The particularities of each numerical method and their interaction when combined is often not well understood or acknowledged. In depth knowledge of the underlying physical phenomena, the discretization and meshing techniques are key to a successful solution algorithm. To return to our 'medicine' metaphor, a good recipe can only be written if it is understood what, and especially, how to cure.

For the PDEs with heterogeneous coefficients considered in this thesis, there are several state of the art (black box) solvers available. Direct solution methods, the FETI method, and algebraic multigrid methods (AMG) are among the most popular solvers and preconditioners.

For small to medium scale problems parallel direct solvers such as MUMPS, PARDISO, or SuperLU [56, 68, 76, 12, 25, 7] are good choices with respect to cost and efficiency. However, the performance of parallel direct solvers degrades when solving linear systems corresponding to three-dimensional meshes. The bandwidth of the stiffness matrix, hardware limitations, delays in communication due to overhead and latency and the arithmetic complexity (recursion) induce a boundary on the scalability of parallel direct solvers when applied to three-dimensional problems.

Several high quality, well parallelisable public domain direct solvers exist. The FETI and AMG methods are also robust but are often much less expensive than direct solution methods and have been discussed in [35, 85, 54, 53, 37, 85]. One AMG adaptation, smoothed aggregation (SA-AMG), has been demonstrated to be a successful parallel preconditioner to iterative methods, for a number of structural mechanics applications [3, 4, 17]. The two most relevant studies of SA-AMG to the simulations considered in this thesis are those of [4, 8], both of which focus on micro-FE modeling of bone deformation, based on micro-CT scans of human bones.

When considering the class of iterative solution methods, the Conjugate Gradient (CG) method [41] is a natural choice as the stiffness matrix is SPD. The CG method is composed of only one matrix-vector multiplication and two inner-products per iteration. In exact arithmetic, the CG method constructs the exact solution within n steps where the stiffness matrix has dimension $n \times n$. Although in theory CG always converges, in practice the amount of iterations of CG is determined to a large extend by the condition number of the stiffness matrix [34]. Linear systems with large jumps in coefficients, alike those resulting from the composite materials considered in this thesis, have a large condition number, hence, slow convergence of CG [83].

Preconditioning is the standard technique for improving the convergence of CG. Common choices of preconditioners are diagonal scaling of the stiffness matrix and incomplete Cholesky factorization without fill in, i.e. IC(0) [62]. However, treating the linear system with a traditional preconditioning technique is not sufficient for our type of application. We will show that there is a direct correlation between the rigid body modes of the components in the composite materials and the condition number of the corresponding stiffness matrix. By removing the rigid body modes of these components from the stiffness matrix we improve the condition number and hence the convergence of CG. The deflation based preconditioners have successfully been applied within the field of computational fluid dynamics, with excellent results on problems with discontinuous jumps in coefficients [81, 32, 80]. We will extend the technique of subdomain deflation, introduced in [66], towards rigid body modes deflation or the mathematically equivalent kernel deflation to remove the effect of the rigid body modes from the linear system. In this thesis we present a new deflation strategy of using rigid body modes based on the underlying geometry and the physical properties of the problem. Moreover, we note that as far as we know this is the first successful application of deflation based preconditioning applied to *coupled* systems of PDEs.

1.5 Scope of the thesis

In this thesis we focus on an efficient iterative solution method for solving large sparse linear systems resulting from the simulation of the mechanical response of composite materials in the context of pavement engineering. In contrast to direct solvers, iterative solvers have more favorable properties for solving linear systems for three-dimensional problems. Iterative solvers are fast, they do not require vast amounts of memory and are highly parallelizable without losing their scalability. We construct a parallelizable iterative method, based on the deflation technique, which is more resource efficient and faster compared to available parallel direct methods. The method considered in this thesis is the DPCG method introduced in [66] and extended towards rigid body modes deflation.

We will benchmark the performance of DPCG for different preconditioners and compare the performance of the method to PCG as well as a direct solver and the state-of-the-art SA-AMG. Moveover, we will compare the performance of SA-AMG using default parameters as a preconditioner for both PCG and DPCG with that of SA-AMG using an optimal choice of parameters as a preconditioner to PCG.

All methods are implemented within a parallel environment using Trilinos [40] and CAPA-3D [19]. We will provide an overview of the DPCG method, and discuss the parallel implementation of the DPCG method into an existing FE software package. Finally, we present numerical experiments on FE meshes from real-life cores of asphalt concrete as case studies for this comparison.

1.6 Outline of the thesis

The structure of this thesis is as follows.

Chapter 2: Structural Mechanics. This chapter gives an introduction to the mathematical framework for the simulation of the mechanical response of composite materials. The framework is built on large deformation of the material. We introduce the driving quantities for deformation and stress, respectively the deformation gradient tensor and First- and Second-Piola Kirchoff stress tensors. We provide definitions for the elastic, plastic, and, viscous material properties as well as the corresponding material models in the framework of large deformation. We introduce the partial differential equations captured by the virtual work equation and we solve these non-linear equations with the Newton-Raphson method. In the last part of this chapter we discuss the algorithm for returning to force equilibrium.

Chapter 3: Discretization virtual work equation. We discretize the linearized virtual work equation, which is introduced in Chapter 2, by means of the Finite-Element (FE) method. This leads to the discretized linearized virtual work equation. We give the algorithm for the solution of the discretized linearized virtual work equation based on the Newton-Raphson method and non-linear virtual work equation of Chapter 2. We introduce the stiffness matrix, which is defined as the Jacobian of the Newton-Raphson method. In the last part of this chapter we give an overview of the properties of the stiffness matrix in the context of composite materials.

Chapter 4: Solving the linear system: overview of solution methods. The efficient solution of the linear systems with the stiffness matrix is key to the simulation of the mechanical response of composite materials. In this chapter we provide an overview of the state-of-the-art of linear solution methods for symmetric positive definite matrices. We consider factorization methods, multigrid methods, and, (preconditioned) iterative solution methods. We motivate our method of choice, the Preconditioned Conjugate Gradient (PCG) method.

Chapter 5: Deflation theory. In this chapter we discuss the performance and the limitations of the PCG method applied to composite materials. We illustrate and explain these limitations by introduction of (small) artificial three-dimensional cases that involve the simulation of asphaltic concrete, which consists of relatively stiff aggregates embedded in a matrix of soft bitumen, resulting in significant differences in the stiffness between the bitumen and aggregate elements especially at higher temperatures. We introduce the deflation operator and describe how to construct the deflation based preconditioner to improve the performance of PCG by using the rigid body modes of the components of the composite materials involved, which leads to the Deflated Preconditioned Conjugate Gradient (DPCG) method. We show theoretically and experimentally convergence rates, independent of the number of aggregates and the differences in stiffness coefficients. In the last part of this chapter we discuss the recursive deflation operator which is key to the construction of the optimal deflation strategy for composite materials.

Chapter 6: Parallel implementation deflation. In this thesis only parallel solution methods are taken into account. The DPCG method that we introduced in Chapter 5 is given as a serial algorithm. In this chapter we discuss the parallel implementation of the deflation operator for parallel algorithms based on domain decomposition. We propose a solver combining rigid body modes deflation, deflation based on the subdomains of the domain decomposition, and, local preconditioners that have limited global error reduction capabilities. We also provide an overview on the parallel implementation of PCG and SA-AMG.

Chapter 7: Numerical examples. In this chapter we consider three numerical examples for comparing the performance and robustness of the linear solvers introduced in Chapter 4 and 5. The experiments considered are one, small, artificial test case, and two real-life engineering cases. We compare the DPCG method, with various preconditioners, to the SA-AMG method, and, the PCG method. We show that the DPCG method is robust, has limited set-up time, is easy to implement, and, has excellent parallel scalability properties.

Chapter 8: Civil Engineering Applications. In this chapter we show that plastic and viscous effects, which are key to many simulations in civil engineering, have no direct influence on the performance of the DPCG method We consider the artificial test case introduced in Chapter 7.

Chapter 9: Summary and conclusions. In this chapter we present the main conclusions and give a summary of the most important research results.

Chapter 10: Future research. We present ideas for future research. We provide a short overview on how we might improve existing deflation vectors. We apply this idea to subdomain deflation and give some results for a real-life test case. Furthermore, we include some recommendations on how the rapidly evolving field of GPU computing can be used to speed up the DPCG method in a parallel computing environment.

This thesis is based on the proceeding papers [87, 44], and, the journal papers [45, 48, 46, 47].

2

Structural mechanics

In the first part of this chapter we introduce the fundamentals of structural mechanics as described in [74, 29, 13, 23]. The fundamentals of structural mechanics form a mathematical framework to model the relation between the force exerted on a body of material, the resulting change in internal force, and, the volumetric change of the body. This framework is used for the simulations of composite materials.

We describe the relation between stress and strain by means of the deformation gradient, the balance of forces, the virtual work equation and we provide a brief overview on the three material properties that constitute the absorption and dissipation of energy: elasticity, plasticity and viscosity.

In the second part of this chapter we describe the implementation of material response and the relation to the virtual work equation. We introduce the Clausius-Planck law for the dissipation of energy and we define the multiplicative decomposition of the deformation gradient to combine the three material properties given in the first part of this chapter.

2.1 Continuum model

In Figure 2.1 a body V in the reference configuration (time $t = t_0$) is subjected to an external force, and deforms into the body v in the current configuration ($t = t_1$). We define position vector X as the position of point P in body V, and position vector x as the position of corresponding point p in body v. Both position vectors are defined in coinciding Cartesian base systems, $\{E_i : i = 1, 2, 3\}$, and $\{e_i : i = 1, 2, 3\}$, in the reference and current configuration respectively. The relative position vector $d\mathsf{X}$ of two material points P and Q in the reference configuration relates to the relative position vector $d\mathsf{x}$ of corresponding material points p and q in the current configuration as,

$$d\mathsf{x} = \mathsf{F} d\mathsf{X} \tag{2.1}$$

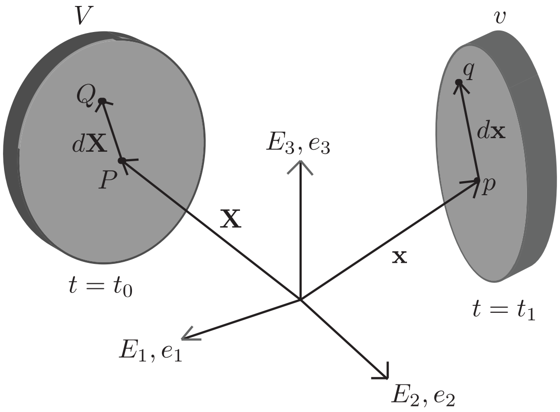


Figure 2.1: Mapping of relative position vector from the reference to the current configuration

in which F is the deformation gradient tensor and is defined as,

$$\begin{aligned} \mathsf{F} &= \frac{\partial \mathsf{x}}{\partial \mathsf{X}} \\ &= \begin{bmatrix} \frac{\partial x_1}{\partial X_1} & \frac{\partial x_1}{\partial X_2} & \frac{\partial x_1}{\partial X_3} \\ \frac{\partial x_2}{\partial X_1} & \frac{\partial x_2}{\partial X_2} & \frac{\partial x_2}{\partial X_3} \\ \frac{\partial x_3}{\partial X_1} & \frac{\partial x_3}{\partial X_2} & \frac{\partial x_3}{\partial X_3} \end{bmatrix} \end{aligned} \tag{2.2}$$

The difference between position vectors x and X is given by displacement vector $\mathbf{u}$,

$$\mathsf{x} = \mathsf{X} + \mathbf{u} \tag{2.3}$$

$$d\mathsf{x} = d\mathsf{X} + d\mathbf{u} \tag{2.4}$$

2.1.1 Strain

We write the deformation gradient tensor as

$$\mathsf{F} = \mathbf{I} + \frac{\partial \mathbf{u}}{\partial \mathsf{X},} \tag{2.5}$$

or in index notation as,

$$F_{ij} = \delta_{ij} + \frac{\partial u_i}{\partial X_j}. \tag{2.6}$$

We introduce the right Cauchy-Green deformation tensor,

$$\begin{aligned} \mathbf{C} &= \mathbf{F}^T\mathbf{F} \qquad (2.7) \\ &= \left(\mathbf{I} + \frac{\partial \mathbf{u}}{\partial \mathbf{X}}\right)^T \left(\mathbf{I} + \frac{\partial \mathbf{u}}{\partial \mathbf{X}}\right) \\ &= \mathbf{I} + \left(\frac{\partial \mathbf{u}}{\partial \mathbf{X}}\right)^T \left(\frac{\partial \mathbf{u}}{\partial \mathbf{X}}\right) + \left(\frac{\partial \mathbf{u}}{\partial \mathbf{X}}\right)^T + \left(\frac{\partial \mathbf{u}}{\partial \mathbf{X}}\right). \end{aligned}$$

In addition to the right Cauchy-Green deformation tensor, we introduce the Lagrangian-Green strain tensor $\mathbf{E}$,

$$\mathbf{E} = \frac{1}{2}\left(\mathbf{C} - \mathbf{I}\right), \qquad (2.8)$$

or in index notation as,

$$E_{ij} = \frac{1}{2}\left(F_{ki}F_{kj} - \delta_{ij}\right), \;\; i, j \in \{1, 2, 3\}. \qquad (2.9)$$

The main diagonals of the 2nd order tensor $\mathbf{E}$ in terms of the displacements are given by,

$$E_{ii} = \frac{\partial u_i}{\partial X_i} + \frac{1}{2}\left[\left(\frac{\partial u_1}{\partial X_1}\right)^2 + \left(\frac{\partial u_2}{\partial X_2}\right)^2 + \left(\frac{\partial u_3}{\partial X_3}\right)^2\right] \qquad (2.10)$$

and the off diagonals, by,

$$E_{ij} = \frac{1}{2}\left[\frac{\partial u_i}{\partial X_j} + \frac{\partial u_j}{\partial X_i} + \frac{\partial u_1}{\partial X_i}\frac{\partial u_1}{\partial X_j} + \frac{\partial u_2}{\partial X_i}\frac{\partial u_2}{\partial X_j} + \frac{\partial u_3}{\partial X_i}\frac{\partial u_3}{\partial X_j}\right], \;\; i \neq j. \qquad (2.11)$$

We note, that in this thesis, we only consider large deformations as these are common practice in continuum mechanics [22]. We refer to [20] for the small strain formulation of such systems.

2.1.2 Stress

Forces applied to a surface area of a body are expressed as pressure and have the derived quantity of Pascal $\left([Pa] = \frac{[N]}{[m]^2}\right)$. The forces per unit area are called stress.

Figure 2.2 illustrates the normal vector $\mathbf{n}$ and traction force $\mathbf{t}$ acting on an infinitesimal element da. We express traction force $\mathbf{t}$ as,

$$\mathbf{t} = \lim_{da \to 0} \frac{d\mathbf{q}}{da}, \qquad (2.12)$$

where $d\mathbf{q}$ is an infinitesimal force. We introduce the Cauchy stress tensor σ in the current configuration,

$$\mathbf{t} = \sigma \cdot \mathbf{n}. \qquad (2.13)$$

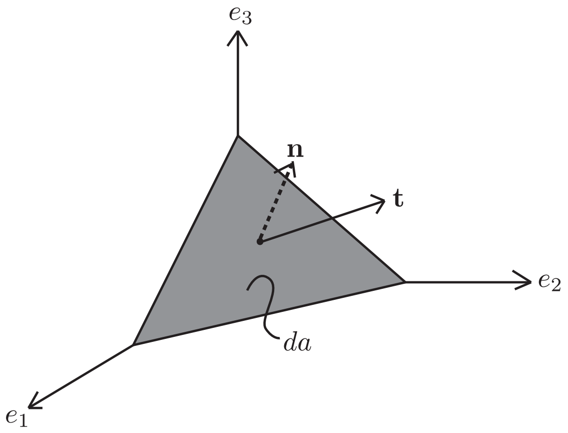


Figure 2.2: Traction **t** acting on an infinitesimal element da

The Cauchy stress is defined in the current, yet unknown configuration, hence, we introduce the First Piola-Kirchhoff stress tensor P in the reference configuration. The First Piola-Kirchhoff stress tensor results from a pullback operation on the Cauchy stress tensor from the current to the reference configuration taking into account the surface area change,

$$\mathsf{T} = \mathsf{P} \cdot \mathsf{N} \tag{2.14}$$

where T is the traction vector with respect to the reference configuration, and N the normal to the corresponding infinitesimal element dA in the reference configuration. The First Piola-Kirchhoff stress tensor P relates to the Cauchy stress tensor σ as,

$$\mathsf{P} = J \cdot \sigma \cdot \mathsf{F}^{-T} \tag{2.15}$$

where $J = \det(\mathsf{F})$ is the Jacobian used for the volumetric transformation between the current and the reference configuration. Because the stress tensor P leads to asymmetric systems, we introduce the symmetric Second Piola-Kirchhoff stress tensor S

$$\mathsf{S} = J \cdot \mathsf{F}^{-1} \cdot \sigma \cdot \mathsf{F}^{-T} \tag{2.16}$$

We note that the Piola-Kirchhoff stress tensor has no direct physical interpretation but is necessary to compute the Cauchy stress. The First and Second Piola-Kirchhoff stress tensors relate as $\mathsf{P} = \mathsf{FS}$.

2.2 Equilibrium equation

At a fixed moment in time all forces exerted on a body and internal forces within the body must be in balance. The balance of forces is defined as,

$$\int_v \mathbf{div}(\sigma) + \mathbf{f} - \rho \mathbf{g} dv = 0, \tag{2.17}$$

where σ represents the distributed loads acting on the body, $\mathbf{f}$ are body forces which can be considered as source terms, and $\rho\mathbf{g}$ is the gravitational force with mass density ρ.

When the system is unbalanced, the cumulative external and internal forces are unequal, this difference is expressed by the residual equation,

$$\mathbf{r} = \mathbf{div}(\sigma) + \mathbf{f} - \rho\mathbf{g}. \tag{2.18}$$

To obtain balance of forces we minimize the residual equation. This problem is hard to solve, hence, we introduce the weak formulation of the residual equation, by multiplying the residual force with a virtual velocity $\delta\mathbf{v}$, yielding the virtual work per unit volume per unit time,

$$\delta W = \int_v \mathbf{r} \cdot \delta\mathbf{v} dv. \tag{2.19}$$

We substitute Equation 2.18 into Equation 2.19, apply Green's Theorem [50], and, obtain the virtual work in the current configuration,

$$\delta W(\mathbf{x}) = -\int_v \sigma : \delta\mathbf{d} dv + \int_v \mathbf{f} \cdot \delta\mathbf{v} dv + \int_a \mathbf{t} \cdot \delta\mathbf{v} da - \int_v \rho\mathbf{g} \cdot \delta\mathbf{v} dv, \tag{2.20}$$

where, $\delta\mathbf{d}$ is the virtual rate of the deformation tensor $\mathbf{d}$ in the current configuration. By substitution of Equation 2.15 and 2.16 into Equation 2.20 we obtain the virtual work in the reference configuration,

$$\delta W(\mathbf{X}) = -\int_V \mathbf{P} : \delta\dot{\mathbf{F}} dV + \int_V \mathbf{f}_0 \cdot \delta\mathbf{v} dV + \int_A \mathbf{t}_0 \cdot \delta\mathbf{v} dA - \int_V \rho_0\mathbf{g} \cdot \delta\mathbf{v} dV. \tag{2.21}$$

The virtual work is the sum of the virtual work of the internal and external forces,

$$\begin{aligned} \delta W(\mathbf{X}) &= \delta W_{int}(\mathbf{X}) - \delta W_{ext}(\mathbf{X}) && (2.22)\\ \delta W_{int}(\mathbf{X}) &= \int_V \mathbf{P} : \delta\dot{\mathbf{F}} dV && (2.23)\\ \delta W_{ext}(\mathbf{X}) &= \int_V \mathbf{f}_0 \cdot \delta\mathbf{v} dV + \int_A \mathbf{t}_0 \cdot \delta\mathbf{v} dA - \int_V \rho_0\mathbf{g} \cdot \delta\mathbf{v} dV && (2.24) \end{aligned}$$

where $\delta\dot{\mathbf{F}}$ is the virtual deformation rate. Because the external forces do not vary in time (or load step) we rewrite the external work as,

$$\delta W_{ext}(\mathbf{X}) = \delta\mathbf{v} \cdot \mathbf{f}_{ext}, \tag{2.25}$$

where,

$$\mathbf{f}_{ext} = \int_V \mathbf{f}_0 dV + \int_A \mathbf{t}_0 dA - \int_V \rho_0\mathbf{g} dV, \tag{2.26}$$

and, at equilibrium,

$$\delta W(\mathbf{X}) = \delta W_{int}(\mathbf{X}) - \delta W_{ext}(\mathbf{X}) = 0. \tag{2.27}$$

2.3 Balancing forces

Due to the non-linear material response and the geometry of the material the virtual work equilibrium equation, given by Equation 2.27, is non-linear. We use the modified Newton-Raphson solution method [43] to solve this non-linear problem.

First we linearize Equation 2.27. We introduce the derivative of an arbitrary function g in the direction of a vector $\Delta\mathbf{u}$,

$$D_{\Delta\mathbf{u}}\left[g\left(\mathsf{x}\right)\right] = \lim_{\varepsilon\to 0}\left(\frac{\partial g\left(\mathsf{x}+\varepsilon\Delta\mathbf{u}\right)}{\partial\varepsilon}\right). \tag{2.28}$$

The directional derivative $D_{\Delta\mathbf{u}}$ satisfies the following rules of differentiation, where A, B and X are arbitrary tensors,

$$\begin{aligned} D_{\Delta\mathbf{u}}\left[\mathsf{A}:\mathsf{B}\right] &= D_{\Delta\mathbf{u}}\left[\mathsf{A}\right]:\mathsf{B}+\mathsf{A}:D_{\Delta\mathbf{u}}\left[\mathsf{B}\right] && (2.29)\\ D_{\Delta\mathbf{u}}\left[\mathsf{A}\right] &= \frac{\partial\mathsf{A}}{\partial\mathsf{X}}:D_{\Delta\mathbf{u}}\left[\mathsf{X}\right]. && (2.30)\end{aligned}$$

Assume $\delta W\left(\mathsf{X}_0\right) > 0$ where X_0 is the spatial vector and assume that with one step in the direction of $\Delta\mathbf{u}$ the equilibrium is reached, $\delta W\left(\mathsf{X}_0+\Delta\mathbf{u}\right) = 0$. Hence, we linearize the virtual work equation around X_0 in the direction of $\Delta\mathbf{u}$,

$$\delta W \tilde{=} \delta W\left(\mathsf{X}_0\right) + D_{\Delta\mathbf{u}}\left[\delta W\left(\mathsf{X}_0\right)\right]. \tag{2.31}$$

The linearized virtual work equation equals to zero at equilibrium,

$$\delta W\left(\mathsf{X}_0\right) + D_{\Delta\mathbf{u}}\left[\delta W\left(\mathsf{X}_0\right)\right] = 0. \tag{2.32}$$

Write Equation 2.32 as,

$$\delta W_{int}\left(\mathsf{X}_0\right) - \delta W_{ext}\left(\mathsf{X}_0\right) + D_{\Delta\mathbf{u}}\left[\delta W_{int}\left(\mathsf{X}_0\right)\right] - D_{\Delta\mathbf{u}}\left[\delta W_{ext}\left(\mathsf{X}_0\right)\right] = 0, \tag{2.33}$$

where, δW_{int}, δW_{ext} as defined in 2.23 and

$$\begin{aligned} D_{\Delta\mathbf{u}}\left[\delta W_{int}\left(\mathsf{X}_0\right)\right] &= \int_V D_{\Delta\mathbf{u}}\left[\mathsf{P}\right]:\delta\dot{\mathsf{F}}\,dV + \int_V \mathsf{P}:D_{\Delta\mathbf{u}}\left[\delta\dot{\mathsf{F}}\right]dV && (2.34)\\ D_{\Delta\mathbf{u}}\left[\delta W_{ext}\left(\mathsf{X}_0\right)\right] &= 0. \end{aligned}$$

We expand the directional derivative of the internal work. The directional derivative of the First Piola-Kirchoff stress tensor,

$$D_{\Delta\mathbf{u}}\left[\mathsf{P}\right] = \frac{\partial\mathsf{P}}{\partial\mathsf{F}}:D_{\Delta\mathbf{u}}\left[\mathsf{F}\right], \tag{2.35}$$

where,

$$D_{\Delta \mathbf{u}}\left[\mathbf{F}\right] = D_{\Delta \mathbf{u}}\left[\frac{\partial \mathbf{x}}{\partial \mathbf{X}}\right] = \lim_{\varepsilon \to 0} \frac{\partial}{\partial \mathbf{X}}\left(\frac{\partial \mathbf{x} + \varepsilon \Delta \mathbf{u}}{\partial \varepsilon}\right) = \frac{\partial \Delta \mathbf{u}}{\partial \mathbf{X}} = \nabla_0 \Delta \mathbf{u}. \tag{2.36}$$

It can be shown that, [74],

$$\frac{\partial \mathbf{P}}{\partial \mathbf{F}} = \mathbf{I} \otimes \mathbf{S} + \mathbf{F} \cdot \mathbb{C} \cdot \mathbf{F}^{T}. \tag{2.37}$$

where, $\mathbb{C}$, represents the fourth order elasticity tensor and is defined as, $\mathbb{C} = \frac{\partial \mathbf{S}}{\partial \mathbf{E}}$. The virtual deformation rate $\delta \dot{\mathbf{F}}$ is defined as,

$$\delta \dot{\mathbf{F}} = D_{\delta \mathbf{v}}\left[\frac{\partial \mathbf{v}}{\partial \mathbf{X}}\right] = \lim_{\varepsilon \to 0} \frac{\partial}{\partial \mathbf{X}}\left(\frac{\partial \mathbf{v} + \varepsilon \delta \mathbf{v}}{\partial \varepsilon}\right) = \frac{\partial \delta \mathbf{v}}{\partial \mathbf{X}} = \nabla_0 \delta \mathbf{v}. \tag{2.38}$$

The directional derivative of the virtual deformation rate $\delta \dot{\mathbf{F}}$ is zero by definition,

$$D_{\Delta \mathbf{u}}\left[\delta \dot{\mathbf{F}}\right] = D_{\Delta \mathbf{u}}\left[\frac{\partial \delta \mathbf{v}}{\partial \mathbf{X}}\right] = 0. \tag{2.39}$$

We substitute the expressions for $D_{\Delta \mathbf{u}}\left[\mathbf{P}\right]$, $\delta \dot{\mathbf{F}}$ and $D_{\Delta \mathbf{u}}\left[\delta \dot{\mathbf{F}}\right]$ into Equation 2.34, which yields,

$$\begin{aligned} D_{\Delta \mathbf{u}}\left[\delta W_{int}\left(\mathbf{X}_0\right)\right] &= \int_V \left(\mathbf{I} \otimes \mathbf{S} + \mathbf{F} \cdot \mathbb{C} \cdot \mathbf{F}^{T}\right) : \nabla_0 \Delta \mathbf{u} : \nabla_0 \delta \mathbf{v} dV + 0 \\ &= \int_V \left(\left(\nabla_0 \Delta \mathbf{u} : \left(\mathbf{I} \otimes \mathbf{S}\right)\right) + \left(\nabla_0 \Delta \mathbf{u} : \mathbf{F} \cdot \mathbb{C} \cdot \mathbf{F}^{T}\right)\right) : \nabla_0 \delta \mathbf{v} dV \\ &= \int_V \left(\nabla_0 \Delta \mathbf{u} \cdot \mathbf{S}\right) : \nabla_0 \delta \mathbf{v} dV + \\ & \quad \int_V \left(\nabla_0 \Delta \mathbf{u} : \mathbf{F} \cdot \mathbb{C} \cdot \mathbf{F}^{T}\right) : \nabla_0 \delta \mathbf{v} dV. \end{aligned} \tag{2.40}$$

After substitution of Equation 2.40, 2.23, and, 2.24 into Equation 2.32 we obtain the linearized virtual work equation at equilibrium,

$$\begin{aligned} \int_V \left(\nabla_0 \Delta \mathbf{u} \cdot \mathbf{S}\right) : \nabla_0 \delta \mathbf{v} dV &+ \int_V \left(\nabla_0 \Delta \mathbf{u} : \mathbf{F} \cdot \mathbb{C} \cdot \mathbf{F}^{T}\right) : \nabla_0 \delta \mathbf{v} dV \\ &= \delta \mathbf{v} \cdot \mathbf{f}_{ext} - \int_V \mathbf{P} : \nabla_0 \delta \mathbf{v} dV. \end{aligned} \tag{2.41}$$

The Newton-Raphson scheme for solving the non-linear virtual work equation is given by Algorithm 1.

Algorithm 1 Balancing forces: solving the non-linear virtual work equation

```
for $t = t_0 ... t_{\text{end}}$ do
    Compute increment of external load
    for $i = 0$ until convergence do
        Determine, $D_{\Delta \mathbf{u}}\left[\delta W_{int}\left(\mathsf{X}_i\right)\right]$, given reference configuration $\mathsf{X}_i$
        if $i = 0$ then
            $\delta W_{int}\left(\mathsf{X}_0\right) = 0$,
        end if
        Solve $D_{\Delta \mathbf{u}}\left[\delta W_{int}\left(\mathsf{X}_i\right)\right] = \delta W_{ext}\left(\mathsf{X}_i\right) - \delta W_{int}\left(\mathsf{X}_i\right)$ obtaining $\mathbf{u}$
        Update displacements, yielding the current configuration, $\mathbf{x}_i = \mathsf{X}_i + \mathbf{u}$, compute the deformation gradient $\mathsf{F} = \frac{d\mathbf{x}_i}{d\mathsf{X}_i}$
        Compute internal force, $\delta W_{int}\left(\mathbf{x}_i\right)$
        Test for convergence, $\delta W_{ext}\left(\mathbf{x}_i\right) - \delta W_{int}\left(\mathbf{x}_i\right) < \varepsilon$
    end for
end for
```

2.4 Material response

In previous section we derived the (linearized) virtual work equation. In this section we give a brief overview of the three material properties, elasticity, plasticity and viscosity. In most real-life applications, the bodies will consist of different types of material and a combination of the three properties, elasto-visco-plasticity, is used.

2.4.1 Elasticity

In Figure 2.3, a one dimensional spring is attached to two moveable boundaries with, at time $t = t_0$, positions x_0, and x_1. When an external pressure σ is applied to the boundary at position x_1 the spring stretches to a new boundary with position x_2 at $t = t_1$. The difference between positions x_2 and x_1 is the strain ε of the spring. We observe elastic behavior when the relation between the external pressure σ and the strain σ is a linear function in time. This linear behavior must be valid for both the loading and unloading phase, the material must return to its original state. In the case of Figure 2.3, the boundaries of the spring must return to positions x_0 and x_1 respectively after unloading.

2.4.2 Plasticity

Again, we consider the spring of Figure 2.3, but we assume that the material has plastic properties. When the applied force is not too large the spring will regain its original shape after unloading. But after a certain threshold the applied force becomes too large and the spring will yield. During unloading the spring has been deformed permanently and we observe a change in volume. The law of conservation of mass implies that the density of the body must have changed. This effect is illustrated by Figure 2.4. We have a linear (elastic) relation between the strain ε and stress σ when the stresses in the material are small. When the elastic limit or yield point has been reached, plasticity is observed. The strain-stress relation is no longer linear and when overstretched, the material will break (break point). Plasticity has two phases, the hardening and softening phase. The hardening phase is spanning the range from the yield stress to the ultimate response. The softening phase represents the range from the ultimate response to the break point.

The domain of admissible stresses that determine the plastic behavior of a material are predefined by the plastic response surface, illustrated in Figure 2.5. The plastic response surface is a function of stress and irrecoverable strain. This means that the surface will grow/shrink in time/iterations. For uniaxial compression tests (uniform loading of the body of material along one axis) the stress path is depicted by the dotted line in Figure 2.5. The variable I_1 is the first invariant and represents summation of the normal stress components, σ_{xx},σ_{yy} and σ_{zz}. The variable $\sqrt{J_2}$ is the second invariant and represents the relation between the deviatoric stress components, S_{xy}, S_{xz} and S_{yz} and are defined as $S_{ij} = \sigma_{ij} - \frac{1}{3}\text{tr}(\sigma)\delta_{ij}$, where $i, j \in \{x, y, z\}$. From Figure 2.5 it is apparent that for an uniaxial compression test the ratio between a stress and strain increment is constant. In this example the ratio is equivalent to an angle of 60°.

The total stress will show elastic behavior when it is still in the domain of admissible stresses. This phase corresponds to the elastic response curve of Figure 2.4. However, when the stress exceeds the elastic limit the material will start to build up plasticity. This is the yield point. The material is in the hardening phase until the point of ultimate plastic response has been reached. From this point the material will no longer harden but it will soften. The physical interpretation is that the material starts to show micro cracks in its internal structure.

Both phases are characterized by the hardening and softening parameters, that are unique to each material. In preempt to Section 2.5, we note that the stress-strain relation illustrated in Figure 2.4 comes from a return mapping procedure that utilizes the plastic response surface. In the return mapping procedure we assume elastic behavior of the material and we back-calculate the corresponding plastic behavior.

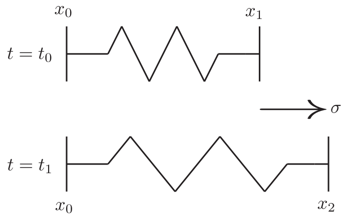


Figure 2.3: Simplified representation of elastic material.

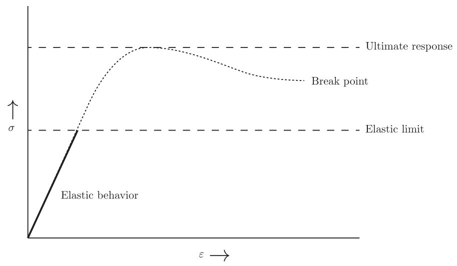


Figure 2.4: Example of relation between strain and stress.

2.4.3 Viscosity

Viscosity is defined as the internal friction of a fluid. In Figure 2.6 we illustrate the flow of a viscous fluid between two plates. We apply force F at the right boundary, hence the upper plate moves with constant velocity v and we fix the position of the lower plate. Due to the viscous material property, at the upper and lower boundaries, the fluid will have the same velocity as the corresponding boundary surfaces. Hence, the fluid is moving with velocity v near the upper plate and is stationary near the lower plate. For a fixed period of time, the volume of fluid in the area $abcd$ will deform into area $abc'd'$. The fluid is in a state of continuously increasing shear strain, defined as the ratio of the displacement dd' to the length of the flow l. We define A as the surface area between the two plates, the ratio $\frac{F}{A}$ is the shear stress exerted on the fluid.

We define the viscosity η of a fluid as the ratio of the shear stress to the change of shear strain,

$$\eta = \frac{F/A}{\Delta dd'/l} = \frac{F/A}{v/l}. \tag{2.42}$$

The viscosity of a material strongly depends on the temperature. For example, at higher temperatures, the upper layer of asphaltic materials absorbs sunlight and the internal heat of the material increases. Hence, the asphalt becomes less viscous and more fluid like. The same effect is observed when heavy forces are applied to the material. Due to pressure the material will become less viscous and will soften.

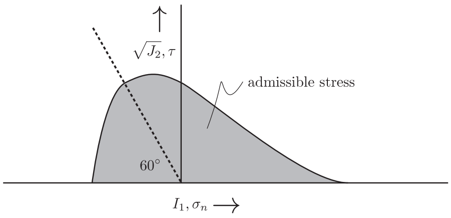

$\uparrow$ $\sqrt{J_2}, \tau$
admissible stress
60°
$I_1, \sigma_n \longrightarrow$

Figure 2.5: Example of plastic yield surface.

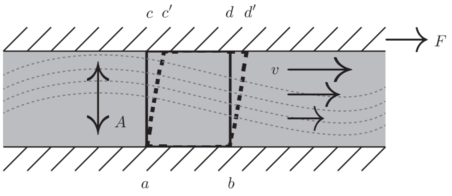


Figure 2.6: Simplified example of 1D viscous laminar flow.

2.5 Implementation of material response

In this section we introduce a mathematical framework to describe the elasto-visco-plasticity material response. We introduce the Clausius-Planck inequality, the law of dissipation of energy within materials, and subsequently we derive the constitutive relations for the material models.

2.5.1 Dissipation of energy

The energy-dissipation equation captures the response of material to externally applied forces. True dissipation of energy is only valid for inelastic systems. Because of this inelastic behavior (plasticity and viscosity), energy (heat) is dissipated over the system when the material responds to the applied forces. In other words, when forces are being applied to the system, mechanical processes within the material are initiated. For elastic materials these processes are reversible. The stress is only a function of the deformation (and temperature) and the system will return to its original state during unloading. However, for plastic and viscous materials the stress becomes a function of deformation and variables associated with the memory properties of the material. From a certain point in time (yielding point), with endured loading, the mechanical processes are irreversible. For instance, when plasticity applies, the system will experience permanent deformation.

The loss of energy is defined by the Clausius-Planck inequality,

$$\mathcal{D} = \mathsf{P} : \dot{\mathsf{F}} - \dot{\Psi} \geq 0, \tag{2.43}$$

where $\mathsf{P} : \dot{\mathsf{F}}$ represents the work per unit volume per unit time and Ψ is known as the Helmholtz free-energy function or, when solely a function of the deformation gradient F, the strain energy function. The Helmholtz free energy function is a potential, i.e. (virtual) work per unit volume. At any point in the system and at all times the internal dissipation $\mathcal{D}$ should be non-negative.

2.5.2 Multiplicative decomposition

We extend our current framework of the sole deformation gradient to combine the three material properties. We want to measure and compute the effects of elasticity, plas-

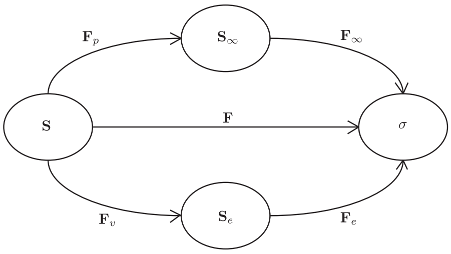


Figure 2.7: Schematic representation of multiplicative decomposition.

ticity and viscosity separately. Hence, we introduce the multiplicative decomposition of the deformation gradient.

In Figure 2.7 we illustrate the decomposition of the deformation gradient of a material in which the elastoplastic and viscoelastic components act in parallel, where,

$$\mathsf{F} = \mathsf{F}_\infty \cdot \mathsf{F}_p, \tag{2.44}$$

$$\mathsf{F} = \mathsf{F}_e \cdot \mathsf{F}_v. \tag{2.45}$$

in which, F_∞, is the elastic component of the deformation gradient of the elastoplastic element, F_p, is the plastic component of the deformation gradient of the elastoplastic element, F_e, is the elastic component of the deformation gradient of the viscoelastic element, and, F_v, is the viscous component of the deformation gradient of the viscoelastic element. We define the elastic and plastic right Cauchy-Green strain tensor of the elasto-plasitc component as,

$$\mathsf{C}_\infty = \mathsf{F}_\infty^T \cdot \mathsf{F}_\infty, \tag{2.46}$$

$$\mathsf{C}_p = \mathsf{F}_p^T \cdot \mathsf{F}_p, \tag{2.47}$$

and the elastic and viscous right Cauchy-Green strain tensor of the visco-elastic component as,

$$\mathsf{C}_e = \mathsf{F}_e^T \cdot \mathsf{F}_e, \tag{2.48}$$

$$\mathsf{C}_v = \mathsf{F}_v^T \cdot \mathsf{F}_v. \tag{2.49}$$

Therefore we have,

$$\begin{aligned} \mathsf{C} &= \mathsf{F}^T \cdot \mathsf{F} && (2.50) \\ &= \mathsf{F}_p^T \cdot \mathsf{C}_\infty \cdot \mathsf{F}_p && (2.51) \\ &= \mathsf{F}_v^T \cdot \mathsf{C}_e \cdot \mathsf{F}_v. && (2.52) \end{aligned}$$

The relation between the Cauchy stress, σ, and the Second Piola-Kirchhoff stress tensor, S, for plasticity is given by,

$$J^{-1} \cdot \mathsf{F}_\infty \cdot \mathsf{S}_\infty \cdot \mathsf{F}_\infty^T = \sigma = J^{-1} \cdot \mathsf{F} \cdot \mathsf{S} \cdot \mathsf{F}^T, \tag{2.53}$$

and the Second Piola-Kirchhoff stress tensor, $\mathbf{S}$, for viscosity is given by,

$$J^{-1} \cdot \mathbf{F}_e \cdot \mathbf{S}_e \cdot \mathbf{F}_e^T = \sigma = J^{-1} \cdot \mathbf{F} \cdot \mathbf{S} \cdot \mathbf{F}^T, \tag{2.54}$$

hence, it is sufficient to compute the values of $\mathbf{S}_\infty$ and $\mathbf{F}_\infty$ to compute the value of $\mathbf{S}$.

2.5.3 Generalized model local dissipation

The Helmholtz free energy function for a elasto-visco-plastic material model can be expressed as,

$$\Psi = \Psi_v \left(\mathbf{C}_e\right) + \Psi_p \left(\mathbf{C}_\infty, \xi_p\right). \tag{2.55}$$

Here ξ_p is a measure of the plastic deformation. The Clausius-Planck inequality of Equation 2.43 leads to,

$$\mathbf{S} : \frac{1}{2}\dot{\mathbf{C}} - \left[\frac{\partial \Psi_p}{\partial \mathbf{C}_\infty} : \dot{\mathbf{C}}_\infty + \frac{\partial \Psi_p}{\partial \xi_p} \cdot \dot{\xi}_p\right] - \left[\frac{\partial \Psi_v}{\partial \mathbf{C}_e} : \dot{\mathbf{C}}_e\right] \geq 0 \tag{2.56}$$

We reformulate Equation 2.56 as,

$$\left[\mathbf{S} - 2\mathbf{F}_v^{-1} \cdot \frac{\partial \Psi_v}{\partial \mathbf{C}_e} \cdot \mathbf{F}_v^{-T} - 2\mathbf{F}_p^{-1} \cdot \frac{\partial \Psi_p}{\partial \mathbf{C}_\infty} \cdot \mathbf{F}_p^{-T}\right] : \frac{1}{2}\dot{\mathbf{C}} \tag{2.57}$$

$$+ \quad \left[2\mathbf{F}_\infty \cdot \frac{\partial \Psi_p}{\partial \mathbf{C}_\infty} \cdot \mathbf{F}_\infty^T \cdot \mathbf{F}_\infty^{-T} : \mathbf{F}_\infty \cdot l_p - \frac{\partial \Psi}{\partial \xi_p} \cdot \dot{\xi}_p\right] \tag{2.58}$$

$$+ \quad \left[2\mathbf{F}_e \cdot \frac{\partial \Psi_v}{\partial \mathbf{C}_e} \cdot \mathbf{F}_e^T \cdot \mathbf{F}_e^{-T} : \mathbf{F}_e \cdot l_v\right] \geq 0. \tag{2.59}$$

By standard arguments the stress tensor, $\mathbf{S}$, can be additively decomposed into a viscoelastic, $\mathbf{S}_v$, and a plastic component, $\mathbf{S}_p$,

$$\mathbf{S} = 2\mathbf{F}_v^{-1} \cdot \frac{\partial \Psi_v}{\partial \mathbf{C}_e} \cdot \mathbf{F}_v^{-T} + 2\mathbf{F}_p^{-1} \cdot \frac{\partial \Psi_p}{\partial \mathbf{C}_\infty} \cdot \mathbf{F}_p^{-T} \tag{2.60}$$

$$= \mathbf{S}_v + \mathbf{S}_p. \tag{2.61}$$

And, hence, we obtain the following constitutive relations for plastic response,

$$\left[2\mathbf{F}_\infty \cdot \frac{\partial \Psi_p}{\partial \mathbf{C}_\infty} \cdot \mathbf{F}_\infty^T \cdot \mathbf{F}_\infty^{-T} : \mathbf{F}_\infty \cdot l_p - \frac{\partial \Psi}{\partial \xi_p} \cdot \dot{\xi}_p\right] \geq 0, \tag{2.62}$$

and, viscous response,

$$\left[2\mathbf{F}_e \cdot \frac{\partial \Psi_v}{\partial \mathbf{C}_e} \cdot \mathbf{F}_e^T \cdot \mathbf{F}_e^{-T} : \mathbf{F}_e l_v\right] \geq 0. \tag{2.63}$$

2.5.4 Hyperelastic response

In this thesis, we describe hyperelasticity with the Neo-Hookean constitutive model,

$$\mathsf{S} = \mu\mathsf{I} - \mu\det(\mathsf{F}^{\mathrm{T}}\mathsf{F})^{-\alpha}(\mathsf{F}^{\mathrm{T}}\mathsf{F})^{-1}, \tag{2.64}$$

and

$$\mathbb{C} = \frac{2\mu\alpha}{\det(\mathsf{F}^{\mathrm{T}}\mathsf{F})^{\alpha}}(\mathsf{F}^{\mathrm{T}}\mathsf{F})^{-1} \otimes (\mathsf{F}^{\mathrm{T}}\mathsf{F})^{-1} - \frac{2\mu}{\det(\mathsf{F}^{\mathrm{T}}\mathsf{F})^{\alpha}}\frac{\partial(\mathsf{F}^{\mathrm{T}}\mathsf{F})^{-1}}{\partial(\mathsf{F}^{\mathrm{T}}\mathsf{F})}, \tag{2.65}$$

where $\alpha = \frac{\nu}{1-2\nu}$ and ν, μ are the Poisson ratio and the Lamé material constant, respectively. We refer to [74] for an extensive overview of hyperelastic constitutive models.

2.5.5 Plastic response

We use the algorithm for computing the plastic response as given in [55]. In the intermediate configuration, for the elastoplastic component of the model, the Helmholtz free energy is given by,

$$\Psi = \Psi\left(\mathsf{C}_\infty, \xi\right), \tag{2.66}$$

and,

$$\dot{\Psi} = \frac{\partial\Psi}{\partial\mathsf{C}_\infty} : \dot{\mathsf{C}}_\infty + \frac{\partial\Psi}{\partial\xi} : \dot{\xi}. \tag{2.67}$$

Rewrite Equation 2.67 with respect to the deformation gradient as,

$$\dot{\Psi} = 2\mathsf{F}_\infty\cdot\frac{\partial\Psi}{\partial\mathsf{C}_\infty}\cdot\mathsf{F}_p^{-T} : \dot{\mathsf{F}} - 2\mathsf{C}_\infty\cdot\frac{\partial\Psi}{\partial\mathsf{C}_\infty}\cdot\mathsf{F}_p^{-T} : \dot{\mathsf{F}}_p - \frac{\partial\Psi}{\partial\xi} : \dot{\xi}. \tag{2.68}$$

Hence, the Clausius-Planck local dissipation inequality reads

$$\begin{aligned} \mathcal{D} &= \mathsf{P} : \dot{\mathsf{F}} - \dot{\Psi} && (2.69)\\ &= \left[\mathsf{P} - 2\mathsf{F}_\infty\cdot\frac{\partial\Psi}{\partial\mathsf{C}_\infty}\cdot\mathsf{F}_p^{-T}\right] : \dot{\mathsf{F}} + 2\mathsf{C}_\infty\cdot\frac{\partial\Psi}{\partial\mathsf{C}_\infty} : l_p + q\dot{\xi} \geq 0, && (2.70) \end{aligned}$$

from which we obtain by standard argumentation the First Piola-Kirchhoff stress tensor, which is defined as,

$$\mathsf{P} = 2\mathsf{F}_\infty\cdot\frac{\partial\Psi}{\partial\mathsf{C}_\infty}\cdot\mathsf{F}_p^{-T}, \tag{2.71}$$

and the dissipation inequality,

$$\mathbf{\Sigma} : l_p + q\dot{\xi} \geq 0, \tag{2.72}$$

where, $\mathbf{\Sigma} = \mathsf{C}_\infty\mathsf{S}_\infty$, is the Mandel stress and, $\mathsf{S}_\infty = 2\frac{\partial\Psi}{\partial\mathsf{C}_\infty}$, the Second Piola-Kirchhoff stress tensor defined in the intermediate configuration.

On the basis of the inequality of Equation 2.72 we define the following constrained minimization problem,

$$\begin{aligned} \min \quad & - \left(\mathbf{\Sigma} : l_p + q\dot{\xi} \right), & (2.73) \\ s.t. \quad & f\left(\mathbf{\Sigma}, q\right). & (2.74) \end{aligned}$$

This minimization problem is equivalent to the following set of plastic evolution equations,

$$\begin{aligned} l_p &= \dot{\mathsf{F}}_p \cdot \mathsf{F}_p^{-1} = \lambda \mathsf{N}, & (2.75) \\ \dot{\xi} &= \lambda \frac{\partial f}{\partial q}, & (2.76) \\ \lambda &\geq 0 \ ; \ f\left(\mathbf{\Sigma}, q\right) \leq 0 \ ; \ \lambda f\left(\mathbf{\Sigma}, q\right) = 0, & (2.77) \end{aligned}$$

where, λ, is the plastic consistency parameter, $\mathsf{N} = \frac{\partial f}{\partial \mathbf{\Sigma}}$, and, $f\left(\mathbf{\Sigma}, q\right)$, is the plastic response surface. The flow rule given by Equation 2.75 can be written as,

$$\frac{\partial \mathsf{F}_p}{\partial t} = \lambda \mathsf{N} \cdot \mathsf{F}_p. \tag{2.78}$$

We compute the elastic deformation gradient to obtain the Second Piola-Kirchhoff stress in the reference configuration,

$$\mathsf{F}_\infty^{t+\Delta t} = \mathsf{F}^{t+\Delta t} \cdot \left(\mathsf{F}_p^{t+\Delta t}\right)^{-1}. \tag{2.79}$$

We assume no plastic deformation takes place during the time interval, $[t, t + \Delta t]$,

$$\begin{aligned} \mathsf{F}_p^{t+\Delta t} &= \mathsf{F}_p^t, & (2.80) \\ \xi^{t+\Delta t} &= \xi^t. & (2.81) \end{aligned}$$

We introduce an approximation of the elastic deformation gradient F_∞,

$$\tilde{\mathsf{F}}_\infty^{t+\Delta t} = \mathsf{F}^{t+\Delta t} \cdot \left(\mathsf{F}_p^t\right)^{-1}. \tag{2.82}$$

We solve the evolution laws of Equation 2.78 for the time interval $[t, t + \Delta t]$ analytically,

$$\mathsf{F}_p^{t+\Delta t} = \left[e^{\Delta \lambda \mathsf{N}}\right]^{t+\Delta t} \mathsf{F}_p^t, \tag{2.83}$$

hence,

$$\begin{aligned} \mathsf{F}_\infty^{t+\Delta t} &= \mathsf{F}^{t+\Delta t} \cdot \left(\mathsf{F}_p^t\right)^{-1} \left[e^{-\Delta \lambda \mathsf{N}}\right]^{t+\Delta t} & (2.84) \\ &= \tilde{\mathsf{F}}_\infty^{t+\Delta t} \left[e^{-\Delta \lambda \mathsf{N}}\right]^{t+\Delta t}. & (2.85) \end{aligned}$$

The exponential can be approximated by a first order Taylor expansion,

$$e^{-\Delta\lambda \mathsf{N}} = \mathsf{I} - \Delta\lambda \mathsf{N} + \frac{(\Delta\lambda)^2}{2!}\mathsf{N}^2 + \ldots. \tag{2.86}$$

Elaborate Equation 2.84 with the use of Expression 2.86 and ignoring the second order term further to,

$$\mathsf{F}_{\infty}^{t+\Delta t} = \tilde{\mathsf{F}}_{\infty}^{t+\Delta t} - \Delta\lambda \mathsf{W}^{t+\Delta t} \tag{2.87}$$

where,

$$\mathsf{W}^{t+\Delta t} = \tilde{\mathsf{F}}_{\infty}^{t+\Delta t} \cdot \mathsf{N}^{t+\Delta t} \tag{2.88}$$

It is apparent that Equation 2.87 constitutes an elastic predictor – plastic corrector solution for the deformation tensor.

We derive the following hardening rule by means of Equation 2.75 and a backward Euler time integration scheme,

$$\xi^{t+\Delta t} = \xi^{t} + \left[\Delta\lambda \left(\frac{\partial f}{\partial q}\right)\right]^{t+\Delta t}. \tag{2.89}$$

A system of residual equations can be formulated using Equation 2.87 and 2.89,

$$R = \begin{pmatrix} R_{\mathsf{F}_{\infty}} \\ R_f \end{pmatrix} = \begin{pmatrix} \mathsf{F}_{\infty}^{t+\Delta t} - \tilde{\mathsf{F}}_{\infty}^{t+\Delta t} + \Delta\lambda \mathsf{W}^{t+\Delta t} \\ \left[f\left(\boldsymbol{\Sigma}, q\right)\right]^{t+\Delta t} \end{pmatrix}. \tag{2.90}$$

Note that the residual R_f is equal to the plastic response surface $\left[f\left(\boldsymbol{\Sigma}, q\right)\right]^{t+\Delta t}$. This corresponds to the objective of reducing the (trial) elastic stress state to the plastic response surface as described in Section 2.4.2. Hence, the plastic response on time $t + \Delta t$ is desired to be zero. We use the Newton-Raphson procedure [30] to solve the preceding residual equations,

$$\begin{pmatrix} \mathsf{F}_{\infty}^{t+\Delta t} \\ [\Delta\lambda]^{t+\Delta t} \end{pmatrix}_{r+1} = \begin{pmatrix} \mathsf{F}_{\infty}^{t+\Delta t} \\ [\Delta\lambda]^{t+\Delta t} \end{pmatrix}_{r} - \left(\left([J]^{t+\Delta t}\right)^{-1}\right)_{r} \begin{pmatrix} [R_{\mathsf{F}_{\infty}}]^{t+\Delta t} \\ [R_f]^{t+\Delta t} \end{pmatrix}_{r} \tag{2.91}$$

where,

$$\left(\left([J]^{t+\Delta t}\right)^{-1}\right)_{r} = \begin{pmatrix} \frac{\partial(R_{\mathsf{F}_{\infty}})_{11}}{\partial(\mathsf{F}_{\infty})_{11}} & \frac{\partial(R_{\mathsf{F}_{\infty}})_{11}}{\partial(\mathsf{F}_{\infty})_{12}} & \cdots & \frac{\partial(R_{\mathsf{F}_{\infty}})_{11}}{\partial(\mathsf{F}_{\infty})_{33}} & \frac{\partial(R_{\mathsf{F}_{\infty}})_{11}}{\partial(\Delta\lambda)} \\ \frac{\partial(R_{\mathsf{F}_{\infty}})_{12}}{\partial(\mathsf{F}_{\infty})_{11}} & \frac{\partial(R_{\mathsf{F}_{\infty}})_{12}}{\partial(\mathsf{F}_{\infty})_{12}} & \cdots & \frac{\partial(R_{\mathsf{F}_{\infty}})_{12}}{\partial(\mathsf{F}_{\infty})_{33}} & \frac{\partial(R_{\mathsf{F}_{\infty}})_{12}}{\partial(\Delta\lambda)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial(R_{\mathsf{F}_{\infty}})_{33}}{\partial(\mathsf{F}_{\infty})_{11}} & \frac{\partial(R_{\mathsf{F}_{\infty}})_{33}}{\partial(\mathsf{F}_{\infty})_{12}} & \cdots & \frac{\partial(R_{\mathsf{F}_{\infty}})_{33}}{\partial(\mathsf{F}_{\infty})_{33}} & \frac{\partial(R_{\mathsf{F}_{\infty}})_{33}}{\partial(\Delta\lambda)} \\ \frac{\partial(R_{\mathsf{f}})}{\partial(\mathsf{F}_{\infty})_{11}} & \frac{\partial(R_{\mathsf{f}})}{\partial(\mathsf{F}_{\infty})_{12}} & \cdots & \frac{\partial(R_{\mathsf{f}})}{\partial(\mathsf{F}_{\infty})_{13}} & \frac{\partial(R_{\mathsf{f}})}{\partial(\Delta\lambda)} \end{pmatrix}. \tag{2.92}$$

2.5.6 Viscoelastic response

Similar to the elastoplastic component, we derive the expressions for the viscoelastic component in the intermediate configurations. In the intermediate configuration, for the viscoelastic component of the model, the Helmholtz free energy function can be set up as

$$\Psi = \Psi\left(\mathbf{C}_e\right). \tag{2.93}$$

Since the Helmholtz free energy function of the viscoelastic component only depends on the elastic strain tensor, it can also be referred to as a Strain Energy function. Its time derivative can therefore be found as

$$\dot{\Psi} = \frac{\partial \Psi}{\partial \mathbf{C}_e} : \dot{\mathbf{C}}_e. \tag{2.94}$$

By means of expression 2.48, Equation 2.94 can be further elaborated as,

$$\dot{\Psi} = 2\mathbf{F}_e \cdot \frac{\partial \Psi}{\partial \mathbf{C}_e} \cdot \mathbf{F}_v^{-T} : \dot{\mathbf{F}} - 2\mathbf{C}_e \cdot \frac{\partial \Psi}{\partial \mathbf{C}_e} \cdot \mathbf{F}_v^{-T} : \dot{\mathbf{F}}_v. \tag{2.95}$$

Hence the Clausius-Planck local dissipation inequality reads,

$$\begin{aligned} \mathcal{D} &= \mathbf{P} : \dot{\mathbf{F}} - \dot{\Psi} && (2.96) \\ &= \left[\mathbf{P} - 2\mathbf{F}_e \cdot \frac{\partial \Psi}{\partial \mathbf{C}_e} \cdot \mathbf{F}_v^{-T}\right] : \dot{\mathbf{F}} + 2\mathbf{C}_e \cdot \frac{\partial \Psi}{\partial \mathbf{C}_e} : l_v \geq 0, && (2.97) \end{aligned}$$

with, $l_v = \dot{\mathbf{F}}_v \mathbf{F}_v^{-1}$. From which by standard argumentation the first Piola-Kirchhoff stress tensor is obtained as,

$$\mathbf{P} = 2\mathbf{F}_e \cdot \frac{\partial \Psi}{\partial \mathbf{C}_e} \cdot \mathbf{F}_v^{-T}, \tag{2.98}$$

and, the dissipation inequality,

$$\mathbf{\Sigma} : l_v \geq 0, \tag{2.99}$$

where, $\mathbf{\Sigma} = \mathbf{C}_e \mathbf{S}_e$, is the Mandel stress and, $\mathbf{S}_e = 2\frac{\partial \Psi}{\partial \mathbf{C}_e}$, is the second Piola-Kirchhoff stress tensor defined in the intermediate configuration. The following evolution law can be found

$$l_v = \mathbf{C}_v^{-1} : \mathbf{\Sigma}, \tag{2.100}$$

with,

$$\mathbf{C}_v^{-1} = \frac{1}{2\eta_D}\left(\mathbf{I} - \frac{1}{3} I \otimes I\right) + \frac{1}{9\eta_V} I \otimes I, \tag{2.101}$$

while η_D and η_V are the deviatoric and volumetric viscosities which may be deformation dependent,

$$\begin{aligned} \eta_D &= \eta_D\left(\mathbf{\Sigma}\right) > 0, && (2.102) \\ \eta_V &= \eta_V\left(\mathbf{\Sigma}\right) > 0. && (2.103) \end{aligned}$$

Therefore,

$$l_v = \dot{\mathbf{F}}_v \mathbf{F}_v^{-1} = \mathbf{C}_v^{-1} : \boldsymbol{\Sigma}, \tag{2.104}$$

which can be written as,

$$\frac{\partial \mathbf{F}_v}{\partial t} = \left(\mathbf{C}_v^{-1} : \boldsymbol{\Sigma}\right) \cdot \mathbf{F}_v. \tag{2.105}$$

In Section 2.5.2 it was indicated that to obtain the second Piola-Kirchhoff stress in the reference configuration we need to compute the elastic deformation gradient,

$$\mathbf{F}_e^{t+\Delta t} = \mathbf{F}^{t+\Delta t} \cdot \left(\mathbf{F}_v^{t+\Delta t}\right)^{-1}. \tag{2.106}$$

If we assume no further viscous deformation takes place during the time interval $[t, t+\Delta t]$ then,

$$\mathbf{F}_v^{t+\Delta t} = \mathbf{F}_v^{t}. \tag{2.107}$$

We introduce an approximation for the elastic deformation gradient $\mathbf{F}_e$,

$$\tilde{\mathbf{F}}_e^{t+\Delta t} = \mathbf{F}^{t+\Delta t} \cdot \left(\mathbf{F}_v^{t}\right)^{-1}. \tag{2.108}$$

We solve the evolution laws of Equation 2.105 for the time interval $[t, t+\Delta t]$ analytically,

$$\mathbf{F}_v^{t+\Delta t} = \left[e^{\Delta \mathbf{C}_v^{-1} : \boldsymbol{\Sigma}}\right]^{t+\Delta t} \mathbf{F}_v^{t+\Delta t}, \tag{2.109}$$

where, $\Delta \mathbf{C}_v^{-1} = \mathbf{C}_v^{-1} \Delta t$. Hence,

$$\begin{aligned} \mathbf{F}_e^{t+\Delta t} &= \mathbf{F}^{t+\Delta t} \cdot \left(\mathbf{F}_v^{t+\Delta t}\right)^{-1} \left[e^{-\Delta \mathbf{C}_v^{-1} : \boldsymbol{\Sigma}}\right]^{t+\Delta t} && (2.110) \\ &= \tilde{\mathbf{F}}_e^{t+\Delta t} \left[e^{-\Delta \mathbf{C}_v^{-1} : \boldsymbol{\Sigma}}\right]^{t+\Delta t}. && (2.111) \end{aligned}$$

We approximate the exponential with a Taylor expansion,

$$e^{-\Delta \mathbf{C}_v^{-1} : \boldsymbol{\Sigma}} \approx \mathbf{I} - \Delta \mathbf{C}_v^{-1} : \boldsymbol{\Sigma} + \frac{\left(\Delta \mathbf{C}_v^{-1} : \boldsymbol{\Sigma}\right)^2}{2!}. \tag{2.112}$$

We expand Equation 2.110 with the approximation of the exponential in Equation 2.112, and, ignoring the second order term. We obtain,

$$\mathbf{F}_e^{t+\Delta t} = \tilde{\mathbf{F}}_e^{t+\Delta t} - \Delta \mathbf{C}_v^{-1} : \mathbf{W}^{t+\Delta t}, \tag{2.113}$$

where,

$$\mathbf{W}^{t+\Delta t} = \tilde{\mathbf{F}}_e^{t+\Delta t} \boldsymbol{\Sigma}^{t+\Delta t}. \tag{2.114}$$

A system of residual equations can be formulated using Equation 2.113,

$$R = R_{\mathbf{F}_e} = \mathbf{F}_e^{t+\Delta t} - \tilde{\mathbf{F}}_e^{t+\Delta t} + \Delta \mathbf{C}_v^{-1} : \mathbf{W}^{t+\Delta t}. \tag{2.115}$$

We use the Newton-Raphson procedure to solve the preceding residual equations,

$$\left(\mathsf{F}_e^{t+\Delta t}\right)_{r+1} = \left(\mathsf{F}_e^{t+\Delta t}\right)_r - \left(\left([J]^{t+\Delta t}\right)^{-1}\right)_r \left([R_{\mathsf{F}_e}]^{t+\Delta t}\right)_r , \tag{2.116}$$

where,

$$\left(\left([J]^{t+\Delta t}\right)^{-1}\right)_r = \begin{pmatrix} \frac{\partial (R_{\mathsf{F}_e})_{11}}{\partial (\mathsf{F}_e)_{11}} & \frac{\partial (R_{\mathsf{F}_e})_{11}}{\partial (\mathsf{F}_e)_{12}} & \cdots & \frac{\partial (R_{\mathsf{F}_e})_{11}}{\partial (\mathsf{F}_e)_{33}} \\ \frac{\partial (R_{\mathsf{F}_e})_{12}}{\partial (\mathsf{F}_e)_{11}} & \frac{\partial (R_{\mathsf{F}_e})_{12}}{\partial (\mathsf{F}_e)_{12}} & \cdots & \frac{\partial (R_{\mathsf{F}_e})_{12}}{\partial (\mathsf{F}_e)_{33}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial (R_{\mathsf{F}_e})_{33}}{\partial (\mathsf{F}_e)_{11}} & \frac{\partial (R_{\mathsf{F}_e})_{33}}{\partial (\mathsf{F}_e)_{12}} & \cdots & \frac{\partial (R_{\mathsf{F}_e})_{33}}{\partial (\mathsf{F}_e)_{33}} \end{pmatrix} . \tag{2.117}$$

3

Discretization virtual work equation

In the first part of this chapter we give a brief overview of the finite-element (FE) method, we derive the discretized linearized virtual work equation and introduce the stiffness matrix. In the second part of this chapter we identify the issues related to solving the discretized virtual work equation, that are key to the main research question of this thesis: how to solve the stiffness matrix resulting from structural mechanics efficiently.

3.1 Discretization of the linearized virtual work equation

In order to solve the linearized virtual work equations of Section 2.3, we discretize in space and time. In static mechanics time is non existent, the external load steps are increased by discrete quantities. In dynamic mechanics we have time as an additional degree of freedom as the velocity as well as the acceleration of the system are taken into account. We introduce the discretization for static and dynamic mechanics separately. We use finite-elements (FE) [11, 90] to discretize the space. In FE, the choice of elements and shape functions affects the stability and accuracy of the numerical solution. In dynamic mechanics, time will be discretized by the Newmark integration scheme [78].

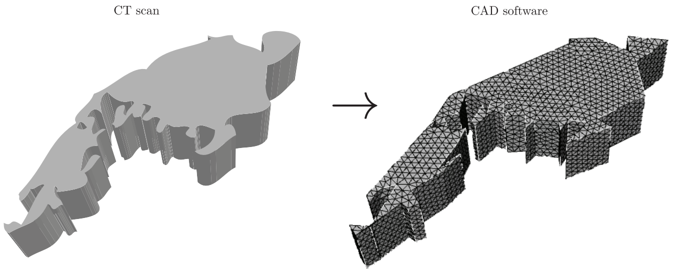


Figure 3.1: Finite element mesh applied to computer model of asphalt column.

3.1.1 Finite-Element method

In this section we provide a brief overview of the FE method. Figure 3.1 shows a body of stone, derived from the CT scan shown in Figure 1.1, and, the resulting mesh that we obtained from the CUBIT mesh generator. The mesh in Figure 3.1 consists of four noded, tetrahedral elements. In general, meshing of composite materials requires a large number of elements, for two reasons. First, composite materials consist of many (small) bodies that have non-regular shapes. These shape require (locally) a large number of elements to obtain a high quality (non-negative element Jacobians, small element-to-element ratio), well defined mesh. Second, in preempt to the remainder of this section, for our application, we use low-accuracy elements. These elements are cheap for numerical integration (evaluation of the internal forces), and yield less degrees of freedom per element, at a cost of having much finer meshes compared to higher-accuracy elements when we want to sustain a satisfactory (global) accuracy of the solution.

Element type and shape functions

The composite materials in this thesis are meshed with unstructured grids and tetrahedral elements. We consider a tetrahedral element with local coordinate system (ξ, η, ζ), illustrated in Figure 3.2, where the relation between the local and the natural coordinate system is represented by the Jacobian J,

$$\begin{pmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \\ \frac{\partial}{\partial \zeta} \end{pmatrix} = J \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix} = \begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{pmatrix} \begin{pmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{pmatrix}. \tag{3.1}$$

We use linear shape functions to interpolate between the element nodes. The linear shape function N_i is defined as,

$$\mathsf{N}_i(\mathbf{x}_\xi) = c_{1,i}\xi + c_{2,i}\eta + c_{3,i}\zeta + c_{4,i} \tag{3.2}$$

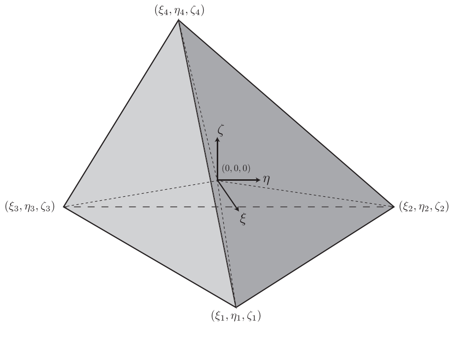

(ξ_4, η_4, ζ_4)
ζ
$(0,0,0)$
η
(ξ_3, η_3, ζ_3)
(ξ_2, η_2, ζ_2)
ξ
(ξ_1, η_1, ζ_1)

Figure 3.2: Tetrahedral element with local coordinate system (ξ, η, ζ)

where, $i = 1, .., 4$, and local position vector, $\mathbf{x}_\xi = (\xi, \eta, \zeta)^{\mathrm{T}}$. By definition, for all $i, j = 1, .., 4$, the shape function, N_i, must satisfy,

$$\mathsf{N}_i(\mathbf{x}_\xi) = \delta_{ij}(\mathbf{x}_\xi) = \left\{ \begin{array}{l} 1, \text{ for } \mathbf{x}_\xi = (\xi_i, \eta_i, \zeta_i)^{\mathrm{T}} \\ 0, \text{ for } \mathbf{x}_\xi = (\xi_j, \eta_j, \zeta_j)^{\mathrm{T}}, i \neq j \end{array} \right. . \tag{3.3}$$

We determine the coefficients of the shape functions by solving, for each $i = 1, .., 4$, the 4×4 system,

$$\begin{bmatrix} \xi_1 & \eta_1 & \zeta_1 & 1 \\ \xi_2 & \eta_2 & \zeta_2 & 1 \\ \xi_3 & \eta_3 & \zeta_3 & 1 \\ \xi_4 & \eta_4 & \zeta_4 & 1 \end{bmatrix} \begin{bmatrix} c_{1,i} \\ c_{2,i} \\ c_{3,i} \\ c_{4,i} \end{bmatrix} = \mathbf{e}_i. \tag{3.4}$$

For given local coordinate $\mathbf{x}_\xi$ the shape functions must satisfy,

$$\sum_{i=1}^{4} N_i \left(\mathbf{x}_\xi\right) = 1. \tag{3.5}$$

We define the interpolated displacement field $\bar{\mathbf{u}}$ at position $\mathbf{x}_\xi$ as,

$$\bar{\mathbf{u}}(\mathbf{x}_\xi) = \sum_{i=1}^{4} \mathsf{N}_i \left(\mathbf{x}_\xi\right) \cdot \mathbf{u}_i, \tag{3.6}$$

where $\mathbf{u}_i$ are the known displacements at nodes $i = 1, .., 4$.

Gauss points and numerical integration

Consider a continuous function f defined on domain, Ω. The integral, I_f, of f over Ω is defined as,

$$I_f = \int_\Omega f d\Omega. \tag{3.7}$$

The domain Ω is meshed into n tetrahedral elements, Ω_i, therefore $\Omega = \bigcup_{i=1}^{n} \Omega_i$, and,

$$I_f = \int_{\Omega_1} f d\Omega_1 + \cdots + \int_{\Omega_n} f d\Omega_n = \sum_{i=1}^{n} \int_{\Omega_i} f d\Omega_i. \tag{3.8}$$

We evaluate the integrals of the functions defined on the elements by numerical integration. We have,

$$\begin{array}{rcl} I_{f_i} & = & \int_{\Omega_i} f d\Omega_i \\ & = & \int_{\Omega_i} f dx dy dz \\ & = & \int_{\Omega_i} f \left|J\right| d\xi d\eta d\zeta \end{array} \tag{3.9}$$

The integral defined on the (ξ, η, ζ) space of Equation 3.9 can be approximated by,

$$\begin{aligned} \int_0^1 \int_0^{1-\xi} \int_0^{1-\xi-\eta} f\,|J|\,d\xi d\eta d\zeta &= f\left(\xi_a, \eta_a, \zeta_a\right)|J| \int_{\Omega_i} d\xi d\eta d\zeta \\ &\tilde{=} f\left(\xi_a, \eta_a, \zeta_a\right)|J|\,\tfrac{1}{6} \\ &= \tilde{I}_{f_i} \end{aligned} \tag{3.10}$$

where, (ξ_a, η_a, ζ_a), is the Gauss point of element Ω_i. Optimal positions for Gauss points have been calculated for many types of elements, such that error, $\left|I_{f_i} - \tilde{I}_{f_i}\right|$, is minimized.

In general, the addition of more Gauss points to an element will increase the accuracy of the numerical evaluation of integrals defined on that element, but increases the computational costs. For linear elements, considered in this thesis, one Gauss point gives the exact approximation to the numerical integrals of the interpolated functions built on linear shape functions.

3.1.2 Stiffness matrix

Static mechanics

In Section 2.3 we have derived the linearized virtual work equation at the equilibrium. We apply the finite elements formulation of Section 3.1.1 to the linearized virtual work equation.

For each element we have N shape functions, N_k, as described in Section 3.1.1. We derive the following quantities by using an isoparametric formulation,

$$\nabla_0 \Delta \mathbf{u} = \sum_{k=1}^{N} \Delta \mathbf{u}_k \otimes \nabla_0 N_k = \sum_{k=1}^{N} \Delta \mathbf{u}_k \otimes \frac{\partial N_k}{\partial \mathbf{X}}, \tag{3.11}$$

$$\nabla_0 \delta \mathbf{v} = \sum_{k=1}^{N} \delta \mathbf{v}_k \otimes \nabla_0 N_k = \sum_{k=1}^{N} \delta \mathbf{v}_k \otimes \frac{\partial N_k}{\partial \mathbf{X}}. \tag{3.12}$$

in which $\Delta\mathbf{u}$ and $\delta\mathbf{v}$ are the displacement increment and virtual velocity respectively. We substitute the former expression into Equation 2.41 and expand all integrals.

The virtual internal work.

$$\begin{aligned} \delta W_{int} &= \int_V \mathbf{P} : \nabla_0 \delta \mathbf{v}\, dV \\ &= \int_V \mathbf{P} : \left(\sum_{k=1}^{N} \delta \mathbf{v}_k \otimes \frac{\partial N_k}{\partial \mathbf{X}}\right) dV \\ &= \sum_{k=1}^{N} \delta \mathbf{v}_k \int_V \mathbf{P} : \left(\mathbf{I} \otimes \frac{\partial N_k}{\partial \mathbf{X}}\right) dV \\ &= \sum_{k=1}^{N} \delta \mathbf{v}_k \int_V \mathbf{P} : \mathbf{B}_0\, dV. \end{aligned} \tag{3.13}$$

The derivative in the direction of the incremental displacement of the internal work,

$$\begin{aligned}
D_{\Delta\mathbf{u}}\delta W_{int} &= \int_V (\nabla_0\Delta\mathbf{u}\cdot\mathbf{S}):\nabla_0\delta\mathbf{v}dV + \int_V \left(\nabla_0\Delta\mathbf{u}:\mathbf{F}\cdot\mathbb{C}\cdot\mathbf{F}^T\right):\nabla_0\delta\mathbf{v}dV \\
&= \int_V (\nabla_0\Delta\mathbf{u}\cdot\mathbf{S}):\nabla_0\delta\mathbf{v}dV + \int_V \left(\mathbb{C}:\mathbf{F}^T\cdot\nabla_0\Delta\mathbf{u}\right):\mathbf{F}^T\cdot\nabla_0\delta\mathbf{v}dV \\
&= \int_V \left(\left(\sum_{l=1}^{N}\Delta\mathbf{u}_l\otimes\frac{\partial N_l}{\partial\mathbf{X}}\right)\mathbf{S}\right):\left(\sum_{k=1}^{N}\delta\mathbf{v}_k\otimes\frac{\partial N_k}{\partial\mathbf{X}}\right)dV \\
&+ \int_V \mathbf{F}^T\left(\sum_{l=1}^{N}\Delta\mathbf{u}_l\otimes\frac{\partial N_l}{\partial\mathbf{X}}\right):\mathbb{C}:\mathbf{F}^T\left(\sum_{k=1}^{N}\delta\mathbf{v}_k\otimes\frac{\partial N_k}{\partial\mathbf{X}}\right)dV. \qquad (3.14)
\end{aligned}$$

Because we have,

$$\left(\Delta\mathbf{u}_l\otimes\frac{\partial N_l}{\partial\mathbf{X}}\right)\cdot\mathbf{S}:\left(\delta\mathbf{v}_k\otimes\frac{\partial N_k}{\partial\mathbf{X}}\right)=\Delta\mathbf{u}_l\cdot\frac{\partial N_l}{\partial\mathbf{X}}\cdot\mathbf{S}\cdot\frac{\partial N_k}{\partial\mathbf{X}}\cdot\mathbf{I}\cdot\delta\mathbf{v}_k, \qquad (3.15)$$

and,

$$\mathbf{F}^T\cdot\left(\Delta\mathbf{u}_l\otimes\frac{\partial N_l}{\partial\mathbf{X}}\right)=\Delta\mathbf{u}_l\cdot\left(\mathbf{F}\otimes\frac{\partial N_l}{\partial\mathbf{X}}\right), \qquad (3.16)$$

we rewrite Equation 3.14 as,

$$\begin{aligned}
D_{\Delta\mathbf{u}}\delta W_{int} &= \sum_{l=1}^{N}\sum_{k=1}^{N}\delta\mathbf{v}_k\left[\int_V\frac{\partial N_l}{\partial\mathbf{X}}\cdot\mathbf{S}\cdot\frac{\partial N_k}{\partial\mathbf{X}}\cdot\mathbf{I}dV\right. \\
&+ \left.\int_V\left(\mathbf{F}\otimes\frac{\partial N_l}{\partial\mathbf{X}}\right):\mathbb{C}\ :\left(\mathbf{F}\otimes\frac{\partial N_k}{\partial\mathbf{X}}\right)dV\right]\Delta\mathbf{u}_l \\
&= \sum_{l=1}^{N}\sum_{k=1}^{N}\delta\mathbf{v}_k\left[\int_V\nabla_0 N_l\cdot\mathbf{S}\cdot\nabla_0 N_{\mathbf{k}}\cdot\mathbf{I}dV\right. \\
&+ \left.\int_V(\mathbf{B}_L)_l:\mathbb{C}\ :(\mathbf{B}_L)_k\,dV\right]\Delta\mathbf{u}_l,
\end{aligned} \qquad (3.17)$$

where,

$$\nabla_0 N_l = \frac{\partial N_l}{\partial\mathbf{X}} \qquad (3.18)$$

$$(\mathbf{B}_L)_l = \left(\mathbf{F}\otimes\frac{\partial N_l}{\partial\mathbf{X}}\right). \qquad (3.19)$$

At last, the discretized form of the external virtual work,

$$\delta W_{ext}=\sum_{k=1}^{N}\delta\mathbf{v}_k\cdot(\mathbf{f}_{ext})_k. \qquad (3.20)$$

This leads to the discretized linearized virtual work equation at equilibrium,

$$\begin{aligned}\sum_{l=1}^{N}\sum_{k=1}^{N}\delta\mathbf{v}_k\left[\int_V \nabla_0 N_l\cdot\mathbf{S}\cdot\nabla_0 N_\mathbf{k}\cdot\mathbf{I}dV \quad + \quad \int_V (\mathbf{B}_L)_l : \mathbb{C}\ : (\mathbf{B}_L)_k\, dV\right]\Delta\mathbf{u}_l & \\ = \sum_{k=1}^{N}\delta\mathbf{v}_k\left(\mathbf{f}_{ext}\right)_k - \sum_{k=1}^{N}\delta\mathbf{v}_k\int_V \mathbf{P} : \mathbf{B}_0 dV. &\end{aligned} \tag{3.21}$$

By definition Equation 3.21 must hold for all $\delta\mathbf{v}_k$, hence we eliminate $\delta\mathbf{v}_k$ from the equation. We write the discrete equilibrium equation in short notation as,

$$K\Delta\mathbf{u} = \Delta\mathbf{f}_{ext} - \Delta\mathbf{f}_{int}, \tag{3.22}$$

where,

$$K = K^\sigma + K^c, \tag{3.23}$$

is defined as the stiffness matrix. The matrices, K^σ, K^c, correspond with the geometrical stress components and the constitutive components respectively.

Analoguous to Algorithm 1 we introduce Algorithm 2 which represents the evaluation of the discretized non-linear virtual work equation by means of the Newton-Raphson method[1].

Dynamic mechanics

We obtain dynamic mechanics when we extend the formulation of the discretized linearized virtual work equation to the time domain. We take into account not only the displacement field but also velocity and acceleration of the body. Consider the linearized virtual work equation extended to the time domain,

$$K\mathbf{u} + C\mathbf{v} + M\mathbf{a} = \mathbf{f}_{ext}, \tag{3.24}$$

where, C, is the damping matrix, M, the mass matrix, $\mathbf{v}$, the velocity vector, and, $\mathbf{a}$, the acceleration vector. We note that, $\dot{\mathbf{u}} = \mathbf{v}$, and, $\ddot{\mathbf{u}} = \mathbf{a}$. We solve the second order differential equation of Equation 3.24 with the Newmark integration scheme [64]. We write the acceleration and velocity as function of displacement and solve for the unknown displacement field, $\mathbf{u}$, hence,

$$\begin{aligned}\mathbf{u}^{t+\Delta t} &= \mathbf{u}^t + \Delta t\mathbf{v}^t + (\Delta t)^2\left[\left(\tfrac{1}{2}-\beta\right)\mathbf{a}^t + \beta\mathbf{a}^{t+\Delta t}\right],\\ \mathbf{v}^{t+\Delta t} &= \mathbf{v}^t + \Delta t\left[(1-\gamma)\,\mathbf{a}^t + \gamma\mathbf{a}^{t+\Delta t}\right].\end{aligned} \tag{3.25}$$

---

[1] Algorithm 2 is the key algorithm of the FE software package CAPA-3D [19] that implements the mathematical framework of Chapters 3 and 4. All numerical results in this thesis have been produced explicitly- or implicitly with CAPA-3D. In the scope of this thesis we have parallelized CAPA-3D by means of domain decomposition.

Algorithm 2 Balancing forces: solving the discretized non-linear virtual work equation

```
for $t = 0 \ldots t_{\text{end}}$ do
    Compute external load $\mathbf{f}^t_{ext}$
    for $i = 0$ until convergence do
        Assemble stiffness matrix $K^{t,i}$
        if $i = 0$ then
            $\mathbf{f}^0_{int} = 0$ and $\Delta\mathbf{f}^0 = \mathbf{f}^0_{ext} - \mathbf{f}^0_{int}$
        end if
        Solve system $K^{t,i}\Delta\mathbf{u}^i = \Delta\mathbf{f}^i$
        Update displacements, $\mathbf{u}^{i+1} = \mathbf{u}^i + \Delta\mathbf{u}^i$
        Compute internal force, $\mathbf{f}^{i+1}_{int}$ and $\Delta\mathbf{f}^{i+1} = \mathbf{f}^{i+1}_{ext} - \mathbf{f}^{i+1}_{int}$
        Test for convergence, $\frac{\Delta\mathbf{f}^{i+1}}{\Delta\mathbf{f}^0} < \varepsilon$
    end for
end for
```

The predictor of time step, t,

$$\begin{aligned} \bar{\mathbf{u}}^{t+\Delta t} &= \mathbf{u}^t + \Delta t\mathbf{v}^t + (\Delta t)^2\left(\tfrac{1}{2} - \beta\right)\mathbf{a}^t, \\ \bar{\mathbf{v}}^{t+\Delta t} &= \mathbf{v}^t + \Delta t\left(1-\gamma\right)\mathbf{a}^t. \end{aligned} \tag{3.26}$$

By substitution of Equation 3.26 into Equation 3.25 we obtain,

$$\begin{aligned} \mathbf{u}^{t+\Delta t} &= \bar{\mathbf{u}}^{t+\Delta t} + (\Delta t)^2\beta\mathbf{a}^{t+\Delta t}, \\ \mathbf{v}^{t+\Delta t} &= \bar{\mathbf{v}}^{t+\Delta t} + \Delta t\gamma\mathbf{a}^{t+\Delta t}. \end{aligned} \tag{3.27}$$

Analoguous to the velocity field, an expression for the acceleration is obtained from Equation 3.27,

$$\mathbf{a}^{t+\Delta t} = \frac{1}{(\Delta t)^2\beta}\left(\mathbf{u}^{t+\Delta t} - \bar{\mathbf{u}}^{t+\Delta t}\right). \tag{3.28}$$

We embed the Newmark integration scheme into the Newton-Raphson iteration scheme of Algorithm 2 to obtain a force equilibrium for every timestep, Δt. Define the displacements, velocity and acceleration for time, $t+\Delta t$, at the $(k+1)^{\text{st}}$ Newton-Raphson iteration as,

$$\begin{aligned} \mathbf{u}^{t+\Delta t}_{k+1} &= \mathbf{u}^{t+\Delta t}_k + \Delta\mathbf{u}^{t+\Delta t}_k, \\ \mathbf{v}^{t+\Delta t}_{k+1} &= \bar{\mathbf{v}}^{t+\Delta t} + \tfrac{\gamma}{(\Delta t)\beta}\left(\mathbf{u}^{t+\Delta t}_{k+1} - \bar{\mathbf{u}}^{t+\Delta t}\right), \\ \mathbf{a}^{t+\Delta t}_{k+1} &= \tfrac{1}{(\Delta t)^2\beta}\left(\mathbf{u}^{t+\Delta t}_{k+1} - \bar{\mathbf{u}}^{t+\Delta t}\right), \end{aligned} \tag{3.29}$$

where, $\mathbf{u}^{t+\Delta t}_0 = \bar{\mathbf{u}}^{t+\Delta t}$. Substitution of Equation 3.29 into Equation 3.24 yields,

$$K^*\Delta\mathbf{u}^{t+\Delta t}_{k+1} = \tilde{\mathbf{f}}^{t+\Delta t}_{ext,k+1} - \mathbf{f}^{t+\Delta t}_{int,k+1}, \tag{3.30}$$

where,

$$\begin{aligned}
&K^* = K + \tfrac{\gamma}{\Delta t\beta}C + \tfrac{1}{(\Delta t)^2\beta}M,\\
&\tilde{\mathbf{f}}^{t+\Delta t}_{ext,k+1} = \mathbf{f}^{t+\Delta t}_{ext} - C\mathbf{v}^{t+\Delta t}_{k} - M\mathbf{a}^{t+\Delta t}_{k},\\
&\mathbf{f}^{t+\Delta t}_{int,k+1} = K\mathbf{u}^{t+\Delta t}_{k}.
\end{aligned}$$

3.2 Stiffness matrix for composite materials

In this section we discuss the properties of the stiffness matrix in the context of composite materials.

3.2.1 General properties of stiffness matrix

In previous chapters, we have seen that due to the shift in length scales as well as the required accuracy in the simulations of asphaltic (composite) materials and with the choice of elements, shape functions, and number of Gauss points, we need to solve the non-linear virtual work equation on fine meshes that contain a large number of elements. By definition, the stiffness matrix of Equation 3.23 is an assembly of element matrices, hence, for these fine meshes, the stiffness matrix has large dimension and is sparse, and ill-conditioned. We assume that an arbitrary domain is meshed by m tetrahedral, linear, elements, yielding element stiffness matrices, $K_e \in \mathbb{R}^{12\times 12}$. We introduce the element operator, $N_e \in \mathbb{R}^{12\times n}$, that maps a global vector to an element vector, $\mathbf{u}_e = N_e\mathbf{u}$. We assemble stiffness matrix, $K \in \mathbb{R}^{n\times n}$,

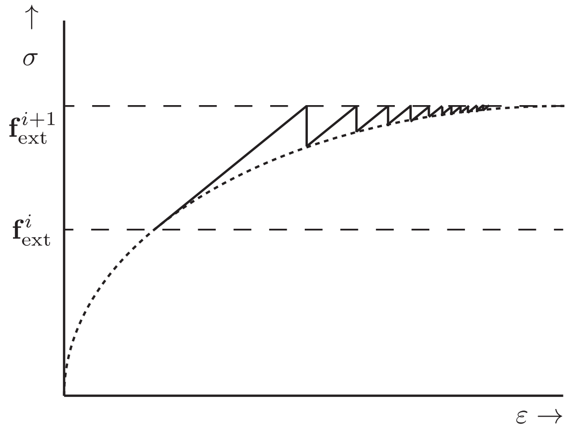
$$K = \sum_{e}^{m} N_e^T K_e N_e. \tag{3.31}$$

The stiffness matrix is, in fact, the Jacobian of the Newton-Raphson method, represented by, K, in Algorithm 2. The stiffness matrix is symmetric and positive definite for elastic, constrained systems; hence, $\forall \mathbf{u} \neq 0 : \mathbf{u}^T K\mathbf{u} > 0$ and all eigenvalues of, K, are real and positive. Furthermore, $\frac{1}{2}\mathbf{u}^T K\mathbf{u}$, is the strain energy stored within the system for displacement vector, $\mathbf{u}$, [11]. Energy is defined as a non-negative entity; hence, the strain energy must be non-negative also.

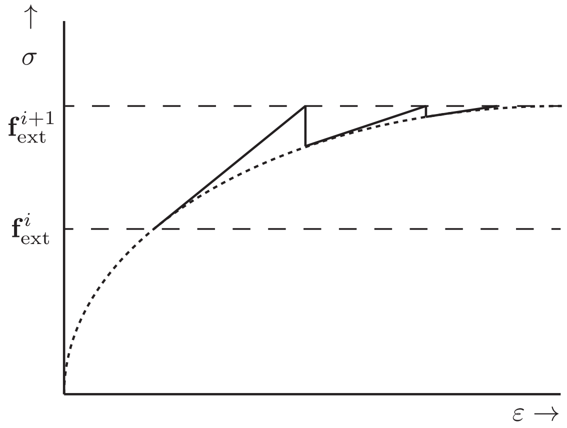
We evaluate the internal forces by numerical integration, which is relatively inexpensive because we have only one Gauss-point per element. We argue that evaluation of the Jacobian, thus, solving the linear system that corresponds to the stiffness matrix, is the most time-consuming step of the Newton-Raphson method.

3.2.2 Discontinuities entries stiffness matrix

We have seen that the geometrical properties of the elements, the shape functions and the constitutive equations determine the entries of the stiffness matrix. Composite



(a) Convergence of Newton-Raphson method using initial stiffness Jacobian



(b) Convergence of Newton-Raphson method using true stiffness Jacobian

materials yield meshes where neighboring elements may have different material properties, hence constitutive models. We have seen in Section 2.4 that the stiffness of a material is determined by its Young's modulus, Lamé constants and Poisson ratio. The matrix entries may vary in orders of magnitude if the different material models have equivalent differences in stiffness. We will see in Chapter 4 that these high jumps between the entries of the stiffness matrix result in very ill-conditioned systems and require tailor made solution methods to efficiently solve the linear system.

3.2.3 Non-linear material properties

To reduce the computation time of Algorithm 2, the entries of the stiffness matrix are fixed during a prescribed number of iterations or load steps. We refer to this principle as 'initial stiffness'. In preempt to Chapter 4, direct solution methods store the factorization of the stiffness matrix and thus, benefit from this approach. However, for (highly) non-linear material properties such as hyper-elasticity, plasticity and viscosity we observe (large) variations in the true entries of the stiffness matrix over load increments or time for static and dynamic mechanics respectively. The poor approximation of the Jacobian of the linearized virtual work equation for such highly non-linear materials may result in a large number of Newton-Raphson iterations. This effect is illustrated in Figure 3.3(a) and 3.3(b) for an arbitrary energy curve. The true stiffness matrix yields an accurate Jacobian and, hence, quadratic convergence of Newton-Raphson. The initial stiffness matrix yields a poor approximation of the Jacobian and the convergence of Newton-Raphson is slow.

In this thesis we consider large strain deformations. For certain stress levels we may exceed the elastic limit of the elasto-visco-plastic material models, and observe hardening and softening due to the visco, and plastic components. These effects yield significant contributions to the entries of the stiffness matrix, and hence, changes the properties of the matrix. The approach of initial stiffness would lead to very slow convergence or, in some cases, stagnation of the Newton-Rapshon method. In this thesis we reassemble the stiffness matrix after every load increment or time step.

3.3 Concluding remarks

In Chapter 2, and 3, we introduced a mathematical framework for the simulation of composite materials. We derived the non-linear virtual work equation, describing the force balance within in a body of material. We solve the virtual work equation with the Newton-Raphson method, and obtained the linearized virtual work equation. We discretized the linearized virtual work equation with the FE method, and obtained the discretized linearized virtual work equation. We introduced the stiffness matrix, which is the Jacobian of the Newton-Raphson method and the driving force of the algorithm. Due to the large differences in material stiffness, we have large jumps in the entries of the stiffness matrix. The stiffness matrices considered in this thesis are large, sparse, and ill-conditioned. In the application considered in this thesis, we have highly non-linear material behavior, hence, for a stable Newton-Rapshon method, we reassemble the stiffness matrix after every load or time step.

4

Solving the linear system: overview of solution methods

In Chapter 2 and 3, we have introduced a mathematical framework of the basics of structural mechanics, we provided the finite element discretization of the linearized virtual work equation and discussed the difficulties related to solving the stiffness matrix. In this chapter we give an overview of established numerical methods for solving large, sparse, symmetric positive definite matrices. We describe the applicability in this context as well as the pros and cons of each numerical solution method. The overview is based on the works of [82, 72, 86].

In general, we solve a linear system of equations,

$$K\mathbf{u} = \mathbf{f}, \tag{4.1}$$

where K is a large, sparse, symmetric positive definite matrix of dimension $n \times n$ and $\mathbf{u}$, $\mathbf{f}$ represent vectors of dimension n. We distinguish between three classes of numerical solution methods. The factorization methods, the multigrid methods, and, the Krylov subspace methods. The factorization methods solve Equation 4.1 by means of matrix factorization and obtain the solution $\mathbf{u}$ by forward elimination and back substitution. The multigrid methods solve Equation 4.1 by using a hierarchy of meshes, solving the linear system on the coarsest grid and smoothing of the global error. The Krylov subspace methods are essentially search algorithms that, given a minimization constraint, construct a solution space, for which the solution converges within a predefined tolerated error margin. Naturally, all three classes have limitations in terms of stability, computer resources and numerical accuracy, hence it is common practice that methods are combined to form one integrated solver.

In Section (4.1) we discuss the factorization methods, in Section 4.2 the multigrid methods, and in Section (4.3) the Krylov subspace methods. At the end of every section we give a brief introduction to the available software implementations of each method.

4.1 Factorization methods

The factorization methods compute the inverse of matrix K by means of matrix factorization. In this section we provide the most basic form, LU factorization or Gaussian elimination, in which we decompose the matrix as $K = LU$, where L is a lower triangular and U an upper triangular matrix. The standard Gaussian elimination algorithm has been improved and stabilized by the addition of pivoting, reordering, and many other numerical add-ons. We refer to [24] for an extensive overview of these methods. In general, the performance of factorization methods depends on the bandwidth, dimension and condition of the matrix.

For any given symmetric positive definite matrix there exists a stable factorization that can be computed without pivoting. However, the stiffness matrix in our application is large, sparse and has a large bandwidth due to the underlying (unstructured) meshes and the coupled three dimensional partial differential equations, inducing high connectivity between nodes and thus degrees of freedom. The fill-in is large and thus memory demands are high. For this reason, factorization methods are often not regarded as the method of choice for solving these large linear systems. However, an important advantage of factorization methods is their robustness: they can, to a large extent, be used as black box solver for solving a wide range of problems. Several high quality, well parallelisable public domain direct solvers exist [56, 68, 76, 12], in this thesis we consider MUMPS [**?**].

In preempt to Section 4.2 and 4.3, we use factorization methods to accelerate Krylov subspace methods and to solve the linear systems resulting from multigrid methods. In Chapter 7 we compare the performance of factorization methods in various numerical examples in combination with multigrid and Krylov subspace methods.

4.1.1 LU factorization

Given a non-singular matrix K, there exist Gauss transformations $M_1 ... M_{k-1}$, which are lower triangular matrices, such that the matrix U given by,

$$M_{n-1} M_{n-2} \cdots M_2 M_1 K = U \tag{4.2}$$

is upper triangular [86]. The inverse of $M_{n-1} ... M_1$ is given by,

$$L = (M_1 ... M_{n-1})^{-1} \tag{4.3}$$

which implies that $K = LU$. The matrix L is lower triangular and $\text{diag}(L) = I$. Once the LU decomposition is obtained Equation 4.1 is easily solved. First, we solve $L\mathbf{v} = \mathbf{f}$ and then the upper triangular system $U\mathbf{u} = \mathbf{v}$.

We obtain factors L and U by Gaussian elimination, given by Algorithm 3. We note that due to round off errors and machine precision Gaussian elimination may give arbitrary poor results, even for well conditioned systems. With partial or complete pivoting or by applying iterative improvements Algorithm 3 can be improved significantly. Implementations of the LU decomposition, such as provided in the LAPACK software library, make use of these enhanced algorithms.

Algorithm 3 Gaussian elimination algorithm

```
Compute $K = LU$ with $K \in \mathbb{R}^{n \times n}$
for $k = 1, ..., n-1$ do
  if $K_{kk} = 0$ then
    quit
  else
    for $i = k+1, ..., n$ do
      $\eta = \frac{K_{ik}}{K_{kk}}$
      $K_{ik} = \eta$
      for $j = k+1, ..., n$ do
        $K_{ij} = K_{ij} - \eta K_{kj}$
      end for
    end for
  end if
end for
```

Cholesky factorization

For symmetric positive definite matrices there exists a unique lower triangular $R \in \mathbb{R}^{n \times n}$ with positive diagonal entries such that $K = RR^T$. This is the Cholesky factorization. Because of the symmetry of K, we only store the upper part. In general the computation of the Cholesky factorization takes halve the amount of work and memory compared to the Gaussian elimination.

Software implementation: MUMPS

MUMPS is public domain software and developed during the Esprit IV European project PARASOL (1996-1999) by CERFACS, ENSEEIHT-IRIT and RAL [7]. The MUMPS package computes an LU decomposition of a given matrix. The MUMPS software is parallel, based on MPI, and available as both shared and distributed memory implementation.

MUMPS computes the LU decomposition using a multifrontal approach [6]. In the multi-frontal approach, all elimination operations take place with dense submatrices, the frontal matrices. The overall factorization of the sparse matrix using a multifrontal scheme can be described by an assembly tree, where each node corresponds to the computation of a Schur complement, and each edge represents the transfer of the contribution block from the child node to the parent node (or father) in the tree. This parent node assembles (or sums) the contribution blocks from all its child nodes with entries from the original matrix.

The algorithms use a dynamic distributed task scheduling technique to accommodate numerical pivoting and to allow the migration of computational tasks to lightly loaded processors. Large computational tasks are divided into subtasks to enhance parallelism. Asynchronous communication is used throughout the solution process to efficiently overlap communication with computation. MUMPS can also determine the rank and a null-space basis for rank-deficient matrices, and can return a Schur complement matrix. It contains classical pre- and postprocessing facilities; for example, matrix scaling, iterative refinement, and error analysis.

We have implemented MUMPS in CAPA-3D by using the distributed coordinate format. The stiffness matrix K, as well as the right hand side $\mathbf{f}$ and solution vector $\mathbf{u}$ are distributed over subdomains. The entries of K are given in triples (i, j, α), which are the local rows, columns and values of the matrix respectively. We refer to Chapter 6 for more details on the parallel implementation of the software that is used in the scope of this research.

4.1.2 Incomplete factorization

LU factorization methods generate a lower triangular matrix L and upper triangular matrix U such that $K = LU$. However, because of the fill in, complete decomposition of an arbitrary large, sparse matrix is often expensive in terms of CPU time and memory size. We introduce a splitting of K,

$$K = LU - R, \tag{4.4}$$

where $R \neq \emptyset$ and LU the incomplete decomposition of K. The incomplete factorization methods are not intended to be stand-alone solution methods, but always combined with other solution methods. We refer to [84] for an extensive overview of the application of ILU methods as accelerator for Krylov subspace methods applied to symmetric positive definite matrices.

The simplest choice of the incomplete decomposition is zero fill-in LU decomposition (ILU(0)).

ILU(0) decomposition

The general idea behind ILU(0) decomposition is to find L, U such that $K_{ij} = (LU)_{ij}$ for $K_{ij} \neq 0$ and $R_{ij} = (LU)_{ij}$ for $K_{ij} = 0$. This means that $R = K - LU$ is zero in the non-zero entries of K. In general there exist many pairs of L and U that satisfy these requirements. One possible ILU(0) decomposition for K is given by Algorithm 4. The number of memory positions which are needed to perform the ILU(0) decomposition, is solely determined by the zero pattern of matrix K.

Software implementation: ILUPACK

ILUPACK [12] is public domain software and was developed by Bollhoefer from TU Braunschweig in collaboration with Saad and Schenk. The software computes the

Algorithm 4 ILU(0) algorithm

```
for $i = 2...n$ do
  for $k = 1...i-1$ and $(i,k) \in NZ(K)$ do
    $K_{ik} = \frac{K_{ik}}{K_{kk}}$
    for $j = k+1...n$ and $(i,j) \in NZ(K)$ do
      $K_{ij} = K_{ij} - K_{ik}K_{kj}$
    end for
  end for
end for
```

incomplete LU (ILU) factorization of a given matrix by means of a recursive multilevel strategy in which for a hierarchy of systems the matrices are reordered, scaled and an inverse-based ILU with diagonal pivoting is applied [12]. The user can affect the amount of fill-in in the factorization by adjusting parameters. The user can provide a threshold for ILU, a bound on the norm of the inverse of the factors, the allowed elbow space for the fill-in in memory, the maximum number of non-zeros per row and a threshold for the approximate Schur complements. There are also various reordering schemes which may improve the performance or stability of the method, depending on the application.

We have implemented ILUPACK in CAPA-3D by using the solver as an accelerator of a Krylov subspace method on each subdomain. We compute the incomplete factorization of the local stiffness matrices where only the diagonal entries of the local stiffness matrices have overlap between neighboring domains. We refer to Section 3.2 where we show that the stiffness matrix is an assembly of symmetric indefinite element matrices with Neumann boundary conditions. By applying boundary conditions we obtain the symmetric positive definite stiffness matrix. In preempt to Chapter 6, the parallel implementation of CAPA-3D may yield local stiffness matrices corresponding to subdomains that are not subjected to the essential boundary conditions and therefore indefinite. The incomplete factorization of these indefinite matrices may not exist or yield very inaccurate local solutions. Hence, by introducing overlap of the diagonal entries we assure that all local stiffness matrices are non-singular and, hence, preserve numerical accuracy.

4.2 Multigrid

Multigrid is a multilevel method in which a hierarchy of grids is constructed on which the linear system is solved recursively. Multigrid methods are highly efficient, the number of arithmetic operations needed to solve the problem is proportional to the number of unknowns in the problem considered. Moreover, multigrid methods are very general, they can be applied with full efficiency to many problems, such as general domains, general boundary conditions, higher dimensions, scalar differential equations, and, systems of differential equations.

Multigrid is the method of choice for solving discretized elliptic equations on equidistant, structured grids. For these type of problems it has optimal performance and grid independent convergence factors [18, **?**, 88]. However, multigrid is suitable for solving general partial differential equations on unstructured grids, but for these type of problems the method has not always optimal performance.

The four key components that determine the performance, stability and robustness of the multigrid method are the *coarse grid specification*, *smoother*, *restriction*, and, *prolongation operator*. The optimal choice of the multigrid components is well known for discretized elliptic partial differential equations on equidistant structured grids, but not for general discretized partial differential equations on unstructured grids.

In this section we introduce the basic principle of multigrid, the four multigrid components and we discuss Smoothed Aggregation Algebraic Multigrid (SA-AMG), which is designed for solving partial differential equations that involve (linear) elasticity. In the last part we consider the ML module of the Trilinos software package, which is a state-of-the-art multigrid solver, as well as the optimal choices for SA-AMG.

4.2.1 Basic multigrid

For a formal description of multigrid we introduce a sequence of coarser and coarser grids, characterized by Ω_k,

$$\Omega_l, \Omega_{l-1}, \ldots, \Omega_0. \tag{4.5}$$

The coarsest grid is Ω_0 whereas the index l corresponds to the finest grid, Ω_l. On grid Ω_k Equation 4.1, is given by,

$$K_k \mathbf{u}_k = \mathbf{f}_k, \tag{4.6}$$

where, K_k, is the corresponding stiffness matrix with dimension, $n_k \times n_k$, for which, $n_k < n$. The displacement vector, $\mathbf{u}_k$, and, force vector, $\mathbf{f}_k$, have dimension n_k. If the solution of Equation 4.6 is approximated by $\mathbf{u}_k^m$, the error $\delta\mathbf{u}_k^m$, and, residual $\mathbf{r}_k^m$ are given by,

$$\begin{aligned} \delta\mathbf{u}_k^m &= \mathbf{u}_k - \mathbf{u}_k^m, & (4.7)\\ \mathbf{r}_k^m &= \mathbf{f}_k - K_k\mathbf{u}_k^m. & (4.8) \end{aligned}$$

We have, $\mathbf{u}_k = \delta\mathbf{u}_k^m + \mathbf{u}_k^m$, hence, we introduce the defect equation,

$$K_k \delta\mathbf{u}_k^m = \mathbf{r}_k^m. \tag{4.9}$$

Multigrid is based on two principles. We consider solving the defect equation by using (basic) iterative methods, as an error averaging process, and, a smooth quantity on an arbitrary grid may be approximated on a coarser grid. This is the principle of smoothing. By solving the defect equation on the coarsest grid with a direct method, we obtain an approximation to the original system, but at a much lower cost. The (smooth) error and residuals are transferred between the sequence of grids by means of the restriction and prolongation operators. We write the four key components of multigrid as (linear) operators,

$$\begin{aligned} K_k &: G(\Omega_k) \rightarrow G(\Omega_k), & (4.10)\\ S_k &: G(\Omega_k) \rightarrow G(\Omega_k), & (4.11)\\ I_k^{k-1} &: G(\Omega_k) \rightarrow G(\Omega_{k-1}), & (4.12)\\ I_{k-1}^{k} &: G(\Omega_{k-1}) \rightarrow G(\Omega_k), & (4.13) \end{aligned}$$

where K_k is the discrete operator that corresponds to Ω_k for $k = l, \ldots, 0$. The operator S_k denotes the iteration operator corresponding to given smoothing methods on Ω_k. The operators I_k^{k-1},I_{k-1}^{k} are the restriction and prolongation operators respectively. One choice for I_k^{k-1} can be the injection operator. For instance, the residual on a fine grid Ω_k is mapped directly to the coarser grid Ω_H. No weighting has been applied. Other operators are based on (full) weighting.

The operator K_k has different eigenmodes which correspond to low and high frequency error components in the solution. The fundamental idea of multigrid is to reduce the high frequency error components by smoothing and the low frequency error components by coarse grid corrections [82]. As said before, the performance, stability and robustness of multigrid depends on the four key components. These components are chosen in such a way that they handle specific eigenmodes of operator K_k. In real-life applications it may be hard to choose the optimal components uniformly for a large class of problems. For the discretized, linearized virtual work equation of Chapter 3, unstructured grids, and, large discontinuities in the matrix entries require special specific, tailored multigrid components. High anisotropy in the discretized differential operator negatively affects the performance of multigrid, however, only isotropic materials are considered in the scope of this research.

4.2.2 Multigrid Components

Coarse grid specification

The coarse grid specification for structured, equidistant grids is straightforward. Assume the finest grid has mesh size, h, then, we could have, $H = 2h$, as a logical choice. However, the problems that we observe within structural mechanics do not have structured grids due to the irregular shapes of the materials. We predetermine a sequence of coarser grids and use a mesh generator to obtain the coarse grids, this allows for an easy choice of the restriction and prolongation operators. These methods may be complicated when the grid is cracking as the original cohesion will disappear. In that case, a new sequence of coarse grids is constructed at every load and time step.

Smoother

For the large systems of equations that we consider in this research, we want the smoother to be cheap in terms of computing memory and floating point operations. Hence, iterative solvers are often the method of choice. Basic iterative methods like Gauss-Seidel and Jacobi often lack efficiency because of the complexity or conditioning of the system. More sophisticated methods, like Krylov subspace methods such as CG, and, GMRES for unsymmetric problems, are a good choice because they are relatively cheap in operation and they can handle specific eigenmodes of the linear operator. Another type of smoothers are polynomial smoothers. The Chebyshev polynomial smoother is well used in parallel multigrid and minimizes over a predefined range of eigenvalues and, hence, complementary to the damping of the low energy modes by the coarse grid corrections [1]. The main disadvantage of the Chebyshev polynomial smoother is that the (extreme) eigenvalues of the linear operator have to be computed in advance. The common approach is to estimate the lower bound of the 'smoothing' spectrum by using the extreme eigenvalue of the linear operator, which is automatically determined for most (algebraic) multigrid implementations, and a scaling parameter. ILU smoothers are not considered, as they lose much of their smoothing property for PDEs defined on three dimensional meshes [82]. Smoothing by collective relaxation is often used for systems of coupled PDEs. All unknowns at each single grid point are relaxed simultaneously.

Restriction operator

For structured grids the restriction operator is often chosen as a weighting process. The distance between grid nodes determines the weight of restriction. The weights at each coarse grid nodes must add up to one. Similar techniques are also applied at an unstructured mesh.

Prolongation operator

A natural prolongation operator is the scaled transpose of the restriction operator. This means that the same weights are applied as with the restriction operator. This operator is identical to constant interpolation in the coordinate direction of the component and bilinear interpolation in the other coordinate directions.

4.2.3 Algebraic multigrid

For PDEs with heterogeneous coefficients that are discretized on unstructured meshes, algebraic multigrid (AMG) approaches [71, 82, 85] offer similar scalability, although at a higher cost per iteration (see, for example, [60] for a comparison of structured and unstructured multigrid approaches). While the fundamental complementarity of the multigrid approach doesn't change within AMG, the way in which the coarse-grid problems are defined does. In geometric multigrid schemes, the coarse-grid operators and intergrid transfer operators (interpolation and restriction) are determined based on explicit knowledge of the grid geometry and discretized PDE. In contrast, interpolation operators for AMG are defined in matrix-dependent ways [5, 71], while the restriction and coarse-grid operators are given by variational conditions (when K is symmetric and positive definite) [65]. Thus, the challenge in achieving efficient multigrid performance is focused on the definition of appropriate matrix-dependent interpolation operators.

In the case of scalar PDEs, there is a wide variety of possible approaches for defining AMG-style interpolation operators [71, 85, 16, 14, 59, 67]. These approaches are largely based on assumptions about the ellipticity of the underlying differential operator and, for scalar PDEs, they typically result in defining interpolation to closely match a given vector (often the constant vector) with the range of interpolation. For systems of PDEs, such as the discretized linearized virtual equation of Chapter 3, more care must be taken, as the ellipticity of the equations of linear elasticity, for example, depends strongly on both the boundary conditions and Lamé coefficients of the system. As a result, there has been much research into the development of efficient AMG approaches for problems in solid mechanics.

For systems of PDEs, there are several possible AMG approaches. Within the setting of classical AMG (often called Ruge-Stüben AMG) [71, 15], these approaches are commonly labeled as the variable-based, point-based (or, equivalently, node-based), and unknown-based approaches [21]. The variable-based approach applies AMG as a black-box, ignoring the structure of the PDE system and, as such, is understood to be effective only for very weakly coupled systems (such as systems with no differential coupling) [21]. The unknown-based approach applies scalar AMG to each component of the system, in a block Jacobi or block Gauss-Seidel manner, and was originally applied to systems of linear elasticity in [70]. Most commonly used is the point-based or node-based approach, where all variables discretized at a common spatial node are treated together in the coarsening and interpolation processes. This approach was first proposed for linear elasticity in [70] and has been extended in [37, 51]. An extension framework to improve AMG for elasticity problems was proposed in [10], which uses a hybrid approach with nodal coarsening, but interpolation based on the unknown-based approach.

The two major disadvantages of AMG are high set up costs, and, generally, non-optimal components. The components are constructed on the basis of compromises between numerical work and overall efficiency. However, in our application of PDEs on unstructured grids, geometric multigrid is too difficult to apply and AMG is the method of choice.

Smoothed aggregation

Despite the recent developments in the use of classical AMG for structural mechanics problems, a much more common algebraic multigrid approach for elasticity problems is in the framework of smoothed aggregation multigrid [85]. In smoothed aggregation, the coarse grid is created by aggregating degrees-of-freedom node wise, through a greedy process that aims to create aggregates of size 3^d for a d-dimensional problem. Such an aggregation is used to define a partition of unity on the grid. A tentative interpolation operator is then defined in groups of 6 columns (for three-dimensional mechanics) by restricting the set of global rigid body modes to each aggregate based on this partition. Smoothed aggregation improves this tentative interpolation operator by applying a single relaxation step (or smoother) to it, leading to an overlapping support of the block-columns of interpolation and giving much more robust performance than unsmoothed (or plain) aggregation. Smoothed aggregation has been demonstrated as a successful parallel accelerator for iterative solution methods for a number of structural mechanics applications [3, 4, 17]. In Section 4.3.2 we will discuss the use of SA-AMG in iterative solution methods.

4.2.4 Software implementation: ML (Trilinos)

The Trilinos Project [40] by SANDIA is an effort to develop algorithms and enabling technologies within an object-oriented software framework for the solution of large-scale, complex multi-physics engineering and scientific problems. Trilinos consists of different modules that can be combined to built an integrated solver. The package is parallel by default.

ML is part of Trilinos and designed to solve large sparse linear systems of equations arising primarily from elliptic PDE discretizations [33]. ML is used to define and build multigrid solvers, and it contains black-box classes to construct highly-scalable SA-AMG solvers.

We have exported the stiffness matrix and right hand side from CAPA-3D to ML/Trilinos.

The optimal parameters for SA-AMG for solving the linearized virtual work equation are derived from [3, 4, 17, 33] and by trial-and-error and are given in Table 4.1. We have one parameter set and two different coarsening schemes. One coarsening scheme is the standard coarsening scheme for SA-AMG as provided by ML. We refer to [33] for more details. Another coarsening strategy, to which we refer as the VBMetis coarsening scheme, makes use of specific information on the coupling of the degrees of freedom. We provide the list of grid nodes, the number of degrees of freedom per grid node as well as the direction of the displacement, either the x-,y- and z-direction. Due to (essential and natural) boundary conditions, the number of degrees of freedom per node may vary. One limitation of this method, due to the parallel implementation of ML/Trilinos, is that all degrees of freedom of one arbitrary node should be assigned to one computing domain.

Table 4.1: Default parameters for SA-AMG

maximum number of coarsening levels	10
multigrid cycle	V-cycle
aggregation type	Uncoupled-MIS. Uses maximal independent set technique to define aggregates. Aggregates can span computing domains.
SA-AMG damping factor	$\frac{4}{3}$
smoother type	Chebychev polynomial. Smoother damps errors between ρ and $\frac{\rho}{\alpha}$, where ρ is spectral radius of $\mathrm{diag}(K)^{-1}K$
smoother damping factor	$\frac{3}{2}$
smoother scheme	pre-, and post-smoothing
Chebychev alpha	30
coarse level solver	direct solver (Cholesky)
maximum number of unknowns on coarsest level	27

4.3 Krylov subspace methods

In this section we introduce the Krylov subspace methods to solve Equation 4.1. Krylov subspace methods are a class of iterative methods that generate a sequence of approximate solutions by solving a minimization problem over a particular subspace. The stiffness matrix, K, in Equation 4.1 is only involved in matrix-vector multiplications and the most popular Krylov subspace algorithms are built on the computation of inner products, addition of vectors, and, scalar-vector and matrix-vector multiplications. Hence, in general Krylov subspace methods require a (relatively) small amount of memory overhead to solve the problem.

Definition 4.3.1. *We define the Krylov subspace $\mathcal{K}_m$, based on stiffness matrix, $K \in \mathbb{R}^{n\times n}$, and vector, $\mathbf{v} \in \mathbb{R}^n$, as,*

$$\mathcal{K}^m\left(K;\mathbf{v}\right) = span\left\{\mathbf{v}, K\mathbf{v}, K^2\mathbf{v}, ..., K^{m-1}\mathbf{v}\right\}, \tag{4.14}$$

where,

$$\forall \mathbf{x} \in \mathcal{K}^m\left(K;\mathbf{v}\right) : \mathbf{x} = \lambda_0\mathbf{v} + \lambda_1 K\mathbf{v} + ... + \lambda_{m-1}K^{m-1}\mathbf{v}, \tag{4.15}$$

with, $\lambda_k \in \mathbb{R}$.

4.3.1 Conjugate Gradient method

In Section 3.2 of Chapter 3 we have seen that the stiffness matrix, K, in Equation 4.1, is symmetric positive definite for all applications of composite materials that are considered in this research, hence the Conjugate Gradient (CG) is the Krylov method of choice [41]. The full CG method is presented by Algorithm 5. As we have discussed in Chapter 1, the Krylov methods are an important component in the development of integrated (hybrid) linear solvers, as these methods have the capability to 'filter' out unfavorable eigenmodes of the system. We will introduce this coupling between the CG method and other linear solvers in Section 4.3.2. In this section we will briefly discuss the philosophy of the CG method.

The main idea behind the CG method is best explained by considering the solution of Equation 4.1 as the unique minimizer of function $\phi(\mathbf{u})$, which is defined by,

$$\phi(\mathbf{u}) = \frac{1}{2}\mathbf{u}^{\mathrm{T}}K\mathbf{u} - \mathbf{u}^{\mathrm{T}}\mathbf{f}, \tag{4.16}$$

where, $\nabla\phi(\mathbf{u}) = \mathbf{f} - K\mathbf{u}$, and, K, $\mathbf{u}$, and, $\mathbf{f}$, are defined as in Equation 4.1. Under these assumptions, an approximate minimizer of $\phi(\mathbf{u})$ can be regarded as an approximate solution of Equation 4.1. The CG method constructs the minimizer of, $\phi(\mathbf{u})$, in Equation 4.16, by using the method of steepest descent along a set of search directions, $\mathcal{P}_k = [\mathbf{p}_1, ..., \mathbf{p}_k]$, that are linear independent, K-conjugate,

Algorithm 5 Conjugate gradient algorithm

```
Start with $\mathbf{r}_0 = \mathbf{f} - K\mathbf{u}_0$, $\mathbf{p}_0 = \mathbf{r}_0$
$\mathbf{r}_k \neq 0$
for $k = 0, 1, \dots$ do
    $\alpha_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T K \mathbf{p}_k}$
    $\mathbf{u}_{k+1} = \mathbf{u}_k + \alpha_k \mathbf{p}_k$
    $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k K \mathbf{p}_k$
    $\beta_k = \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}$
    $\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$
end for
```

$\forall \mathbf{p}_i, \mathbf{p}_j \in \mathcal{P}_k,\ i \neq j,\ \mathbf{p}_i K \mathbf{p}_j = 0$, and, $\mathbf{p}_k^{\mathrm{T}} \mathbf{r}_{k-1} \neq 0$. An important property of the CG method is that, provided that the search directions are independent, and $\mathbf{u}_k$ solves Equation 4.1, the method convergences in at most n steps, as, $\mathcal{R}(\mathcal{P}_n) = \mathbb{R}^n$.

We introduce the energy norm,

$$||\mathbf{u}||_K = \left(\mathbf{u}^T K \mathbf{u}\right)^{\frac{1}{2}}. \tag{4.17}$$

and define the error, δu_k, in the k-th approximation of the solution of Equation 4.1 as,

$$\delta \mathbf{u}_k = \mathbf{u} - \mathbf{u}_k, \tag{4.18}$$

where, u_k, is the approximation to the exact solution, u. We show that minimizing the energy norm of the error, $\delta \mathbf{u}_k$, over the Krylov subspace, $\mathcal{K}^{k-1}(K; \mathbf{r}_0)$, is equivalent to the minimization of Equation 4.16,

$$\min_{\delta \mathbf{u}_k \in \mathcal{K}^{k-1}(K;\mathbf{r}_0)} \delta \mathbf{u}_k^{\mathrm{T}} K \delta \mathbf{u}_k = \min_{\mathbf{u}_k \in \mathcal{K}^{k-1}(K;\mathbf{r}_0)} \mathbf{u}_k^{\mathrm{T}} K \mathbf{u}_k - 2\mathbf{u}_k^{\mathrm{T}} \mathbf{f} + \mathbf{u}^{\mathrm{T}} \mathbf{f}. \tag{4.19}$$

We note that minimizing the error in the energy norm is, in fact, minimizing the strain energy over the Krylov subspace $\mathcal{K}^{k-1}(K; r_0)$ for linear elasticity. This implies that, for a given distributed static load, we construct a displacement vector that has an optimal distribution of the internal force over the material.

Theorem 10.2.6 in [34] provides a bound on the error of the approximations computed by the CG method. Denote the ith eigenvalue of K in nondecreasing order by $\lambda_i(K)$ or, simply, λ_i. After k iterations of the CG method, the error is bounded by

$$\left\|\delta \mathbf{u}_k\right\|_K \leq 2 \left\|\delta \mathbf{u}_0\right\|_K \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k \tag{4.20}$$

where $\kappa = \kappa(K) = \dfrac{\lambda_n}{\lambda_1}$ is the spectral condition number of K. While this bound is not always sharp, the error reduction capability of the CG method is generally limited when the condition number is large. The condition number of K typically increases when the number of elements increases or when the relative stiffnesses of the materials change. For plastic and viscous behavior, this can result in a series of increasing numbers of iterations as the stiffness changes with every load or time step. Here we focus on a single load and time step, although this is an important question for future research as plasticity and viscosity are key to realistic simulations.

The convergence of CG is not only affected by the condition number but also by the number and distribution of very small eigenvalues, which has been shown in[83]. The eigenvectors corresponding to the smallest eigenvalues do have a significant contribution to the global solution but may need a significant number of iterations to convergence locally. Hence, very small eigenvalues can increase the number of iterations. In preempt to Chapter 5, we will see clusters of very small eigenvalues who are isolated from the other part of the spectrum of the stiffness matrix. These clusters of eigenvalues will have a strong negative effect on the performance of the CG method for the PDEs considered in this research.

4.3.2 Preconditioning

As the performance of the CG method is highly dependent on the extreme eigenvalues of the linear operator, we precondition the linear system to obtain more favorable extreme eigenvalues [72].

The preconditioned stiffness matrix reads,

$$M^{-1}K\mathbf{u} = M^{-1}\mathbf{f}, \tag{4.21}$$

where, M, is the left preconditioner and assumed to be symmetric positive definite, hence, non-singular. The bound on the number of CG iterations, given by Equation (4.20), is also valid when CG is applied to the preconditioned matrix. We know that the CG method converges very fast in the energy norm if, $\kappa(M^{-1}K) \approx 1$. The preconditioned system is evaluated in every iteration of CG, hence we require that the preconditioning matrix must be cheap to construct and inexpensive to apply. In general, preconditioner, M, is an approximation of the linear operator, and ideally such that the eigenvalues of the preconditioned system cluster around 1.

4.3.3 Preconditioned Conjugate Gradient method

The principle of improving the performance of the CG method by preconditioning of the original linear system has been introduced by Reid and Axelson [69, 9]. The preconditioner for the CG method must meet a number of requirements. The preconditioner, M, must be a symmetric positive definite matrix, as the CG method is designed specificially for symmetric systems and perform best for positive definite matrices. Also, we must find a decent approximation of the stiffness matrix K and it should be easy

Algorithm 6 Preconditioned conjugate gradient algorithm

```
Start with $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, $\mathbf{z}_0 = M^{-1}\mathbf{r}_0$ and $\mathbf{p}_0 = \mathbf{z}_0$
$\mathbf{r}_k \neq 0$
for $k = 0, 1, \dots$ do
    $\alpha_k = \frac{\mathbf{r}_k^T \mathbf{z}_k}{\mathbf{p}_k^T A \mathbf{p}_k}$
    $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
    $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A \mathbf{p}_k$
    $\mathbf{z}_{k+1} = M^{-1}\mathbf{r}_{k+1}$
    $\beta_k = \frac{\mathbf{r}_{k+1}^T \mathbf{z}_{k+1}}{\mathbf{r}_k^T \mathbf{z}_k}$
    $\mathbf{p}_{k+1} = \mathbf{z}_{k+1} + \beta_k \mathbf{p}_k$
end for
```

to solve $\mathbf{y}$ from $M\mathbf{y} = \mathbf{s}$ as it will be solved every iteration step. In order to preserve symmetry we introduce the M-inner product. When we replace the Euclidean inner product with the M-inner product and rewrite Algorithm 5, we obtain the Preconditioned Conjugate Gradient (PCG) method given by Algorithm 6. Note that just one extra line is added to the original algorithm, hence the implementation of the PCG method is similar to the original CG method.

It is clear that, since the preconditioner is an 'approximation' to the linear operator, the choice of the preconditioner strongly depends on the properties of the linear operator. Hence, for an arbitrary stiffness matrix that results from the discretization of PDEs, an 'optimal' preconditioner exists. We define optimal as the trade-off between the computational expense of the preconditioner and its ability to map the preconditioned system to a more favorable spectrum. In recent years, there have been many publications on suitable preconditioners for elliptic PDEs and linear elasticity [35, 85, 3, 4, 17, 4, 8].

In this research we compare four preconditioners which are selected on basis of different favorable properties. In Chapter 5 and 7, we compare the results of PCG with different preconditioners applied to composite materials.

Diagonal scaling

The most obvious choice for a preconditioner based on an approximation of K is $M = D$ where D contains the main diagonal elements of K. The diagonal scaling preconditioner scales the main diagonal elements of K to 1. The main advantages of the diagonal scaling preconditioner are the minimal computing and storage costs. For an $n \times n$ matrix the storage costs are n real numbers and the computational costs are n flops. Within the field of high performance computing, the diagonal scaling preconditioner is often used when solving extremely large problems on large clusters of computers (1000+ computing cores). The algorithm is embarrassingly parallel as no communication is required to solve with the preconditioner. Although diagonal scaling has limited effect on the performance of PCG for most ill-conditioned systems, it is an excellent method for benchmarking other preconditioners.

ILU decomposition

In Section 4.1.2 we discussed the properties of the ILU factorization methods. The use of ILU decomposition as a preconditioner for CG has been first proposed in [62]. The performance of the ILU preconditioner is mainly determined by the allowed amount of fill-in of the bandwidth of the stiffness matrix, which is controlled by the drop tolerance. We have derived the optimal drop tolerance by trail-and-error. In preempt to Chapter 7, we will see that the ILU decomposition preconditioner for the stiffness matrix in this research yields large memory costs and is computationally intenstive. We also discussed the difficulties of using the ILU decomposition preconditioner in a parallel computing environment. The stiffness matrices associated with the parallel domains may be (near) singular, yielding inaccurate results and, hence, possible divergence of PCG. This potential pitfall is solved by synchronization of the main diagonals of all local stiffness matrices, assuring non-singularity. We elaborate on this in Chapter 6.

Remark. The diagonal scaling preconditioner is in fact an ILU factorization method, where only the main diagonal elements are taken into account. For very large systems, the stiffness matrix is usually divided over many parallel computing domains. The ILU factorization method has less ability of global conditioning of the stiffness matrix, because the local stiffness matrix connectivity decreases. The diagonal scaling preconditioner and the ILU factorization method tend to have equal performance for these parallelized systems.

SA-AMG

In recent years there have been many publications on using SA-AMG as preconditioner for linear elasticity. The two most relevant studies of SA-AMG to the simulations considered here are those of [4, 8], both of which focus on micro-FE modeling of bone deformation, based on micro-CT scans of human bones. The SA-AMG method has 'optimal' multigrid efficiency for these type of problems. The application of the SA-AMG as preconditioner is straightfoward. The PCG method is built on recursion and does not store the Krylov subspace, hence, the preconditioner, M, cannot be flexible. We fix the number of SA-AMG cycles, as well as the number of (pre- and post-) smoothing steps and coarse grid corrections. As discussed in Section 4.2.3, we have two variants of SA-AMG. The first variant uses the default aggregation scheme, but has limited set-up costs. The second variant determines the aggregation scheme based on the coupling of the PDEs, but has significant set up costs. The latter has optimal multigrid efficiency for linear elasticity, but the performance for non-linear, large deformation, elasto-visco-plastic PDEs has to be determined.

4.3.4 Software implementation: AztecOO (Trilinos)

AztecOO is a package within Trilinos, which was introduced in Section 4.2.4. The Trilinos software is built on the Epetra framework, in which parallel objects such as matrices, vectors, and, graphs, are defined that are thus inherited by all modules. This enables for a fast implementation of (new) parallel algorithms.

AztecOO contains many standard Krylov subspace methods and gives access to many preconditioners using Ifpack (ILU decomposition), ML (Multigrid/AMG), and, AztecOO itself. Moreover, the implementation of new preconditioners using Epetra objects and other modules is fast and straightforward. Users can also override the default convergence tests in Aztec00.

We have exported the stiffness matrix and right hand side from CAPA-3D to Aztec00/Trilinos.

4.4 Concluding remarks

Within the field of engineering, the PCG method is widely used because it is easy to implement, PCG iterations are cheap, and the storage demands are modest and fixed. However, there remain pitfalls in its use in practical simulations. As discussed in this chapter, its performance depends on the conditioning and/or spectrum of the matrix. This can be improved by the use of an appropriate preconditioner, but this adds to the work and storage required by the algorithm. Consequently, the convergence of low-energy modes can be slow and, more importantly, poorly reflected in the residual associated with an approximate solution. The established alternative is the use of direct solution methods or multigrid methods. For our application, the choice of direct or iterative methods should not be made based on the solution of a single linearized system but, rather, based on the full nonlinear, time-dependent equations to be solved. When using simple nonlinear constitutive relations, direct methods may be preferable, if the factorization of one stiffness matrix can be reused many times. If, on the other hand, the stiffness matrix changes significantly with every linearization, the use of PCG may be preferable, particularly if the number of Newton iterations can be controlled. The study performed here focuses on the latter idea, that a single factorization cannot be reused enough to make direct methods competitive. In preempt to Chapter 7, for large sparse systems the costs of factorization are substantially greater than a single iterative solve and, so, the use of iterative solvers can reduce the total required computational time.

5

Deflation theory

In this chapter we discuss the performance of the PCG method applied to composite materials with varying material stiffness. We introduce the deflation operator, and, discuss the use of specific information of the underlying mesh and the material properties to construct a more robust numerical solver: the Deflated Preconditioned Conjugate Gradient method. We will introduce different deflation strategies and we illustrate the efficiency of the methods with (small) artificial test cases. In the last section of this chapter we introduce the optimal deflation strategy for composite materials: recursive rigid body mode deflation.

5.1 Composite materials: Preconditioned CG

In Chapter 3 we argued that the stiffness matrix K is ill conditioned due to the large discontinuities in the entries of the matrix and the shear size of the underlying unstructured meshes. We have seen in Chapters 1 and 3, that asphaltic materials consists of elements that significantly vary in stiffness. Hence, the stiffness matrix K has presumably large differences in the extreme eigenvalues and therefore the condition number of stiffness matrix, K, will be large. Given the PCG iteration bound of Equation 4.20 we expect a large number of iterations for this kind of problems. Moreover, the number of aggregates in a composite material determines the number of (large) discontinuities in the stiffness matrix. Hence, the number of aggregates has a direct correlation with the condition number of the matrix. The increase of the number of aggregates may, therefore, result in an increase in the number of very small eigenvalues and deterioration of the convergence rates of PCG.

In this section we apply the PCG method to various samples of composite materials to determine the relation between the physical structure of the problem (number of 'aggregates'), the values of the material parameters (discontinuities) and the robustness of the PCG method. As we focus on the properties of the spectrum of the preconditioned stiffness matrix and not so much on finding the perfect preconditioner, we choose diagonal scaling as preconditioner.

5.1.1 Motivating numerical experiments

All experiments in this section involve the same domain and an equal number of elements. We only consider relatively small problems in order to be able to analyse the eigenvalues with MATLAB [63]. All matrices and material problems were generated by CAPA-3D [19]. The aim of the experiments is to find out how the convergence rate of PCG depends on the material properties and geometry of the volumes. Hence, there will be an emphasis on the relation between the number of PCG iterations and the material stiffness and number of aggregates. Increasing the number of elements will result in a more ill-conditioned problem and therefore the number of PCG iterations increases.

The experiments will involve four different set-ups.

I. Homogeneous material (bitumen) and no aggregates
II. One aggregate in bitumen layer
III. Four aggregates in bitumen layer
IV. Eight aggregates in bitumen layer

For all experiments non-linear, hyper elastic material behavior is taken into account. The bituminous material is considered as a rubber like material, with low stiffness. Furthermore, we assume that under the same conditions, the aggregates do not deform and float in a sea of bituminous material. We consider the discretized virtual work equation of Chapter 3. We have linear, 4 noded elements with three directions of displacement at every node. At the main diagonal of the stiffness matrix we have the elastic node contributions. The dominating term for the main diagonal is the Youngs modulus (elasticity) over the Poisson ratio (compressibility), $\frac{E}{1-2\nu}$ where $0 < \nu \leq \frac{1}{2}$. We only consider normal compressible materials, $0.2 \leq \nu \leq 0.45$, therefore E is the parameter of interest. The E of bitumen will be kept at a constant value of 200 MPa. The E of the aggregates will vary between $\mathcal{O}(10^5)$ and $\mathcal{O}(10^9)$. The results will be related to the ratio between the E of the bitumen and aggregates respectively.

For simplicity, Figure 5.1 shows a 2D representation of 3D test cases (II) and (III). Domain Ω is divided into Ω_a and Ω_b which represent the aggregate and bitumen subdomains respectively. We have $\Omega = \Omega_a \cup \Omega_b$, where $\Omega_a = \bigcup_i^n \Omega_a^i, n = 1, 4$ and $\Omega_b = \bigcup_i^3 \Omega_b^i$. The aggregates are only placed within domain Ω_b^2 such that the boundary conditions at Γ_3 and Γ_4 are not acting on the aggregate elements directly. The aggregates are considered as rigid bodies. We emphasize that the aggregates are actually groups of elements, which share the same material properties.

As the total number of elements remains the same, the size of the aggregates must decrease when the number of aggregates increases. The boundary conditions are prescribed displacements (Dirichlet b.c.) at boundary Γ_3 and fixed support, i.e. no

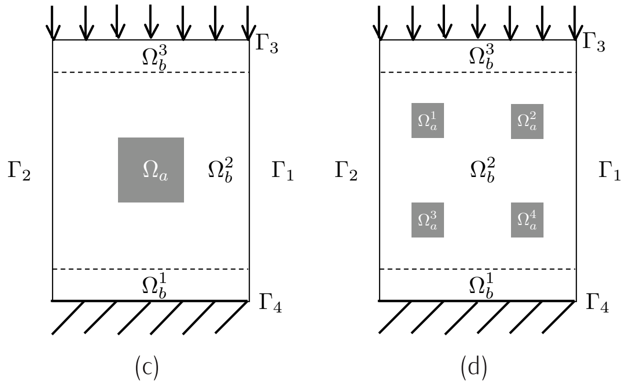


Figure 5.1: Schematic 2D representation of 3D test cases (II) and (III), figure (c) and (d) contain one and four aggregates respectively.

displacements (Dirichlet b.c.), at boundary Γ_4. At boundaries Γ_1, Γ_2 there is unconstrained displacement (homogeneous Neumann b.c.) in every direction.

Convergence results

Figure 5.2 shows the convergence results of PCG with diagonal scaling for all cases (I) to (IV). We note that case (I), the red dotted line, is used as a benchmark for the other cases. We can conclude that there is a direct correlation between the number of iterations, the material stiffness, and the number of aggregates. As the ratio between the elastic moduli of the bitumen and aggregates increases, the extreme eigenvalues shift in opposite directions, the condition number increases and the number of iterations increases. And when the number of aggregates increases the condition number remains unchanged in order of magnitude but the number of iterations still increases. We have seen that preconditioning will reduce the condition number and therefore the number of iterations. But the introduction of more aggregates has clearly an effect on the spectrum of eigenvalues.

Figure 5.3 shows the smallest eigenvalues of $M^{-1}K$ of all four cases. We note that the aggregates are independent subdomains relative to each other. Hence, no aggregate contains nodes from other aggregates. In this way we can consider the aggregates as rigid bodies within a layer of bitumen. It can be shown that all rigid body modes of the aggregates correspond to the smallest eigenvalues of K. In three dimensions we have six rigid body modes, hence we expect 6, 24 and 48 smallest eigenvalues that correspond to the rigid body modes for cases (II), (III) and (IV) respectively. This is precisely what is observed for all cases. Moreover, the increase in very small eigenvalues is clearly visible as there is a jump between the values of the largest eigenvalues corresponding to the rigid body modes and the remaining

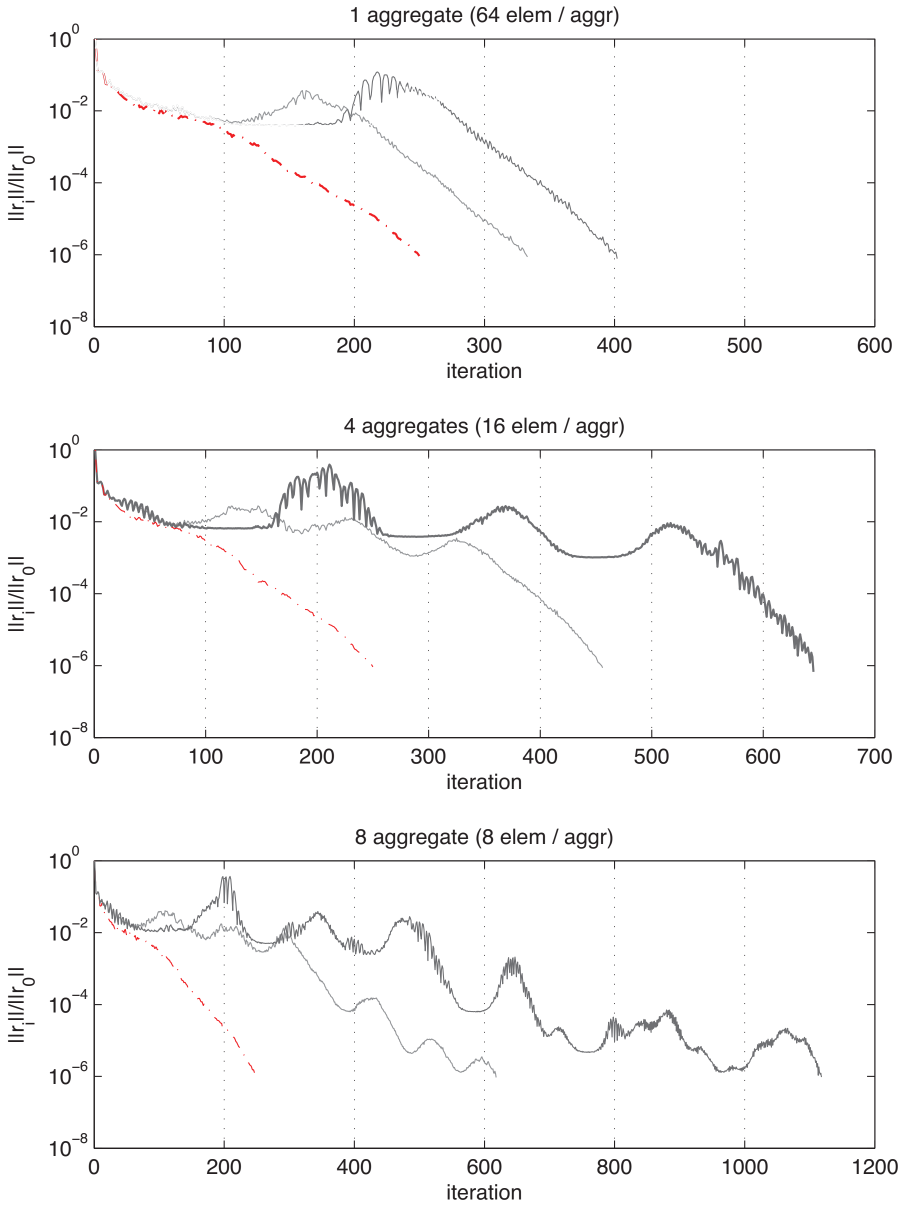


Figure 5.2: Deterioration of rates of convergence of PCG for increasing number of aggregates and stiffness. -.- homogeneous material, – E ratio $\mathcal{O}(10^3)$, – E ratio $\mathcal{O}(10^5)$.

eigenvalues in the spectrum of $M^{-1}K$.

As the condition number of the stiffness matrix is of the same order for a different number of aggregates we do not expect a large increase of iterations for an increasing

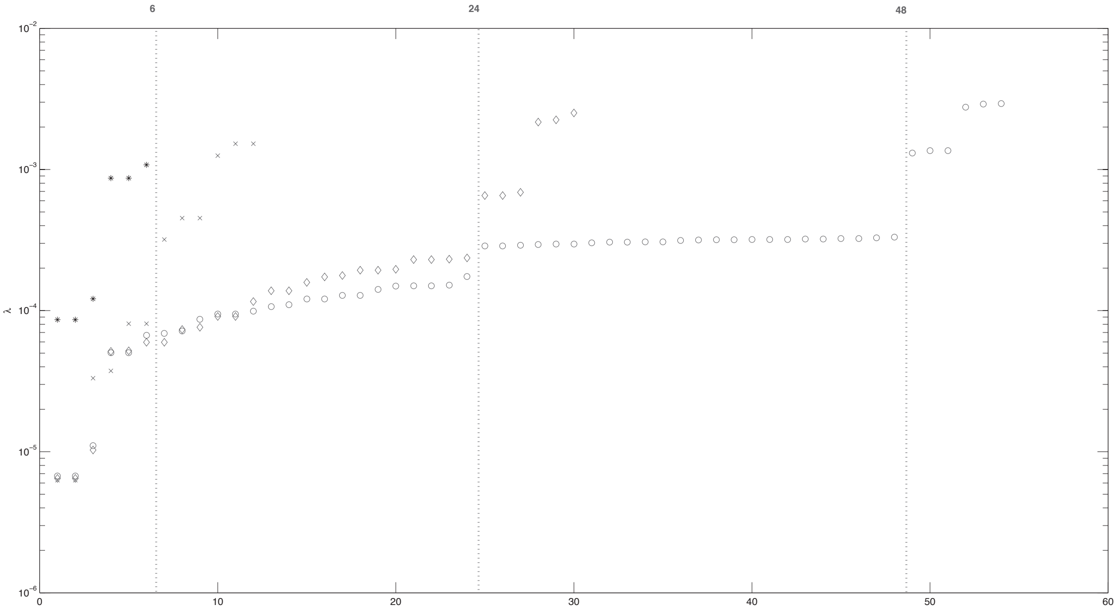


Figure 5.3: 50 smallest eigenvalues of $M^{-1}K$, $M = \text{diag}(K)$. $\times$: 1 aggregate, $\diamond$: 4 aggregates, $\circ$: 8 aggregates, $\ast$: homogeneous material

number of aggregates based on the condition number. However, from Figure 5.3 we expect a slowly converging solution due to the large number of smallest eigenvalues, which is indeed observed in Figure 5.2. The clustering of eigenvalues is clearly visible for cases (III) and (IV), we observe small plateaus which correspond to the slow converging components of the solutions. The number of plateaus increases when the number of aggregates increases due to the number of smallest eigenvalues, i.e., rigid body modes.

The extreme eigenvalues and condition numbers of the (preconditioned) stiffness matrices of test cases (I), (II) and (III) are given in Table 5.1. Two observations stand out when interpreting the results. The smallest eigenvalue of the non-preconditioned problem is almost invariant with respect to increasing E ratio. The largest eigenvalue of the non-preconditioned problem is not invariant with respect to an increasing E ratio. The order of the largest eigenvalue increases proportionally to the increase in E ratio. Obviously, both observations do not hold when a diagonal scaling preconditioner is applied. In contrary, the inverse effect is observed. The smallest eigenvalue becomes even smaller as the E ratio increases and the largest eigenvalue is a constant value. We have seen that an upperbound for convergence of CG is related to the condition number of the stiffness matrix. Hence, we can expect an increasing number of iterations when the E ratio increases.

Case (I)		λ_{min}	λ_{max}	κ
$\mathcal{O}(10^2)$	K	0.0425	$4.15 \cdot 10^3$	$9.76 \cdot 10^4$
	$M^{-1}K$	$8.62 \cdot 10^{-5}$	5.9248	$6.87 \cdot 10^4$
Case (II)				
$\mathcal{O}(10^3)$	K	0.0460	$1.30 \cdot 10^6$	$2.82 \cdot 10^7$
	$M^{-1}K$	$6.29 \cdot 10^{-6}$	5.9248	$9.42 \cdot 10^5$
$\mathcal{O}(10^5)$	K	0.0460	$1.30 \cdot 10^8$	$2.82 \cdot 10^9$
	$M^{-1}K$	$6.64 \cdot 10^{-8}$	5.9248	$8.92 \cdot 10^7$
Case (III)				
$\mathcal{O}(10^3)$	K	0.0450	$1.01 \cdot 10^6$	$2.24 \cdot 10^7$
	$M^{-1}K$	$6.48 \cdot 10^{-6}$	5.9239	$9.15 \cdot 10^5$
$\mathcal{O}(10^5)$	K	0.0450	$1.01 \cdot 10^8$	$2.24 \cdot 10^9$
	$M^{-1}K$	$6.90 \cdot 10^{-8}$	5.9239	$8.59 \cdot 10^7$

Table 5.1: 2304 elements, the extreme eigenvalues and condition number of preconditioned stiffness matrices. $\mathcal{O}(10^n)$ represents the jump in E modulus of aggregates and bitumen.

5.2 Introduction to deflation

We have shown in the previous section that the number of iterations to convergence for PCG is highly dependent on the number of aggregates in a mixture as well as the ratio of the Young's moduli. Increasing the number of aggregates introduces correspondingly more (clustered) small eigenvalues in stiffness matrix, K. The jumps in the Young's moduli are related to the size of the small eigenvalues. We know from [83] that the smallest eigenvalues correspond to the slow converging components of the solution. Thus, we look to design a preconditioner that directly addresses these modes.

Formally to define the deflation preconditioner, we split the solution of Equation (4.1) into two parts [32],

$$\mathbf{u} = \left(I - P^{\mathrm{T}}\right)\mathbf{u} + P^{\mathrm{T}}\mathbf{u}, \tag{5.1}$$

where P is a projection matrix that is defined as,

$$P = I - KZ(Z^{\mathrm{T}}KZ)^{-1}Z^{\mathrm{T}}, \text{ for } Z \in \mathbb{R}^{n\times m}, \tag{5.2}$$

where $\mathcal{R}(Z)$ represents the deflation subspace, i.e., the space to be projected out of the system, and I is the identity matrix of appropriate size. We assume that $m \ll n$ and that Z has rank m. Under these assumptions, $E \equiv Z^{\mathrm{T}}KZ$ is symmetric positive definite and may be easily computed and factored. Hence,

$$\left(I - P^{\mathrm{T}}\right)\mathbf{u} = ZE^{-1}Z^{\mathrm{T}}K\mathbf{u} = ZE^{-1}Z^{\mathrm{T}}\mathbf{f} \tag{5.3}$$

can be computed directly, and the difficult computation is of $P^{\mathrm{T}}u$. Because KP^{T} is symmetric,

$$KP^{\mathrm{T}} = PK, \tag{5.4}$$

and P is a projection, we solve the deflated system,

$$PK\hat{\mathbf{u}} = P\mathbf{f}, \tag{5.5}$$

for $\hat{\mathbf{u}}$ using the PCG method and multiply the result by P^{T} giving $\mathbf{u} = ZE^{-1}Z^{T}\mathbf{f}+P^{T}\hat{\mathbf{u}}$. We note that (5.5) is singular; however, the projected solution $P^{\mathrm{T}}\hat{\mathbf{u}}$, is unique, as it has no components in the null space, $\mathcal{N}(PK) = \mathrm{span}\{Z\}$. Moreover, from [49, 83], the null space of PK never enters the iteration process, and the corresponding zero-eigenvalues do not influence the solution.

5.2.1 Deflated Preconditioned CG

The deflation method was proposed in [66]. A practical variant of the Deflated Preconditioned Conjugate Gradient (DPCG) method [81] is given by Algorithm 7. The DPCG method extends PCG, enhancing stability and robustness when solving for symmetric, and positive definite systems, but requires extra storage for the deflation matrix Z.

Algorithm 7 Deflated preconditioned CG solving $K\mathbf{u} = \mathbf{f}$

```
Select $\mathbf{u}_0$. Compute $\mathbf{r}_0 = (\mathbf{f} - K\mathbf{u}_0)$, set $\hat{\mathbf{r}}_0 = P\mathbf{r}_0$
Solve $M\mathbf{y}_0 = \hat{\mathbf{r}}_0$ and set $\mathbf{p}_0 = \mathbf{y}_0$
for $j = 0, 1, \dots$ until convergence do
    $\hat{\mathbf{w}}_j = PK\mathbf{p}_j$
    $\alpha_j = \frac{(\hat{\mathbf{r}}_j, \mathbf{y}_j)}{(\hat{\mathbf{w}}_j, \mathbf{p}_j)}$
    $\hat{\mathbf{u}}_{j+1} = \hat{\mathbf{u}}_j + \alpha_j \mathbf{p}_j$
    $\hat{\mathbf{r}}_{j+1} = \hat{\mathbf{r}}_j - \alpha_j \hat{\mathbf{w}}_j$
    Solve $M\mathbf{y}_{j+1} = \hat{\mathbf{r}}_{j+1}$
    $\beta_j = \frac{(\hat{\mathbf{r}}_{j+1}, \mathbf{y}_{j+1})}{(\hat{\mathbf{r}}_j, \mathbf{y}_j)}$
    $\mathbf{p}_{j+1} = \mathbf{y}_{j+1} + \beta_j \mathbf{p}_j$
end for
$\mathbf{u} = ZE^{-1}Z^{\mathrm{T}}\mathbf{f} + P^T\hat{\mathbf{u}}_{j+1}$
```

Moreover, $PK\mathbf{u}$ in Algorithm 7 is computed in every iteration. However, the unfavorable eigenvalues due to the discontinuities in the stiffness matrix are treated by the deflation method making these costs worthwhile. For many problems the DPCG method is robust for even highly ill-conditioned problems [79].

To obtain a useful bound for the error of DPCG for positive semi-definite matrices, we define the effective condition number of a semi-definite matrix $D \in \mathbb{R}^{n \times n}$ with rank $n - m$, $m < n$, to be the ratio of the largest and smallest positive eigenvalues; analogous to Equation (4.20),

$$\kappa_{\text{eff}}(D) = \frac{\lambda_n}{\lambda_{m+1}}. \tag{5.6}$$

Theorem 2.2 from [32] implies that a bound on the effective condition number of PK can be obtained.

Theorem 5.2.1. *Let P be defined as in (5.2), and suppose there exists a splitting $K = C + R$, such that C and R are symmetric positive semi-definite with null space of C, $\mathcal{N}(C) = span\{Z\}$. Then for ordered eigenvalues λ_i,*

$$\lambda_i(C) \leq \lambda_i(PK) \leq \lambda_i(C) + \lambda_{max}(PR). \tag{5.7}$$

Moreover, the effective condition number of PK is bounded by,

$$\kappa_{eff}(PK) \leq \frac{\lambda_n(K)}{\lambda_{m+1}(C)}. \tag{5.8}$$

Proof. See [32] (p445). □

While the large jumps in matrix entries due to strongly varying material properties in the FE discretization induce unfavorable eigenvalues (either large or small) in the spectrum of stiffness matrix K, the effective condition number of PK is bounded by the smallest eigenvalue of C and the largest eigenvalue of K. To remove the discontinuities and, thus, eliminate those unfavorable eigenvalues, we decouple the submatrices of stiffness matrix K that correspond to different materials by finding the correct splitting. The eigenvalues of the decoupled submatrices then determine the spectrum of PK. However, due to the large differences in stiffness, the values of the eigenvalues for the different submatrices can still vary over several order of magnitudes. To achieve a scalable solution algorithm, we couple this deflation procedure with another preconditioner to map the spectra of the submatrices onto the same region, around 1. This deflation technique can be used in conjunction with any ordinary preconditioning technique, giving a two-level approach, treating the smallest and largest eigenvalues by deflation and preconditioning, respectively. By choosing a favorable combination of deflation and preconditioning, a better spectrum is obtained, yielding a smaller effective condition number and fewer iterations. For a symmetric preconditioner $M = LL^{\mathrm{T}}$, e.g. diagonal scaling, the result of Theorem 5.2.1 extends to [32, Thm 2.3],

$$\kappa_{\text{eff}}(L^{-1}PKL^{-T}) \leq \frac{\lambda_n(L^{-1}KL^{-T})}{\lambda_{m+1}(L^{-1}CL^{-T})}. \tag{5.9}$$

5.3 Rigid body modes deflation

5.3.1 Motivation: rigid bodies in composite materials

For any finite-element computation, we consider subsets of unconstrained elements as rigid bodies. Their corresponding (sub) stiffness matrices are assemblies of the element stiffness matrices. In the context of asphalt concrete, the aggregates are subsets of elements, with their Young's modulus as a shared property; the bitumen and the air voids are defined similarly.

When a matrix, K_{unc}, represents a rigid body, i.e. an unconstrained mechanical problem (with no essential boundary conditions), the internal energy equals zero for the rigid body displacements, as the system remains undeformed, and the matrix is positive semi-definite, $\forall u : u^{\mathrm{T}} K_{unc} u \geq 0$. More specifically, the number of rigid body modes of any unconstrained volume equals the number of zero eigenvalues of its corresponding stiffness matrix. When an arbitrary matrix, A, has zero eigenvalues, the kernel $\mathcal{N}(A)$ is non-trivial. Moreover, the basis vectors of the kernel of a stiffness matrix represent the principal directions of the rigid body modes. In general, two types of rigid body modes exist: translations and rotations. In three dimensions, this implies six possible rigid body modes and, hence, six kernel vectors can be associated with the rigid body modes. For the linearized virtual work equation of Chapter 3, the physical interpretation of these kernels are the rigid body modes of the (linear) elastic components of the material.

In the Section 5.1.1 we concluded that the number of aggregates times the number of rigid body modes per aggregate (6 in three dimensions) is equal to the number of small eigenvalues of stiffness matrix K. By using deflation, we can 'augment' the Krylov subspace with pre-computed rigid body modes of the aggregates, and thus remove all corresponding small eigenvalues from the system. As a result, the number of iterations of the Deflated Preconditioned Conjugated Gradient method is nearly not affected by jumps in material stiffness or by the number of aggregates. This is a significant improvement over many other preconditioning techniques whose performance degrades even for simpler heterogenous problems [80].

5.3.2 Construction of deflation vectors

To fully comprehend the construction of the deflation vectors by rigid body modes, we use the following experiment. Assume that we have a cube of bitumen containing one aggregate which is shown in Figure 5.4. The subdomains Ω_1 and Ω_2 can be considered as bitumen and aggregates respectively. Clearly, without the constraints of the surrounding bitumen material, the aggregate of Ω_2 will act as a rigid body. With kernel deflation we aim to solve on both subdomains separately. We separate subdomain Ω_1 from subdomain Ω_2 and apply new boundary conditions to the domains. We assume that the aggregates that are not influenced by the boundary conditions of the whole domain act as rigid bodies, therefore we assume homogeneous Neumann boundary conditions on the aggregates. The bitumen will be restricted by the aggregates and we apply homogeneous Dirichlet boundary conditions.

Consider the assembly of the stiffness matrix, K, as given in Equation 3.31, for the domain in Figure 5.4 which is meshed with k elements. Assume that $K_b = \sum_{e\in\Omega_1} N_e^{\mathrm{T}} K_e N_e$, $K_a = \sum_{e\in\Omega_2} N_e^{\mathrm{T}} K_e N_e$ and $K_\Gamma = \sum_{e\in\Omega_1\cap\Omega_2} N_e^{\mathrm{T}} K_e N_e$. We assume $K = C + R$ according to Theorem 2.2 of [32] where $C = K_b + K_a - K_\Gamma$ and $R = K_\Gamma$. Matrix C consists of two independent block matrices which correspond to the bitumen and aggregate domains. Matrix R consists of only those node contributions of elements from Ω_1 that lie on the intersection of domain Ω_1 and Ω_2. We note that by the removal of the bitumen-aggregates boundary nodes from the bitumen subdomain, Dirichlet and Neumann boundary conditions are automatically imposed on the bitumen and aggregate submatrices in matrix C. The matrix C contains one singular submatrix corresponding to the aggregate and one positive definite submatrix corresponding to the bitumen. Moreover, because of the Dirichlet boundary conditions, the bitumen domain is statically determined and numerically well conditioned.

We apply Theorem 2.2 of [32] to $C = K_b + K_a - K_\Gamma$ and $R = K_\Gamma$. We have, $Z = \mathcal{N}(C) = \mathrm{span}\{Z_a\}$ where $Z_a = \{z_a^1, \dots, z_a^6\}$ with z_a^j the j-th base vector of

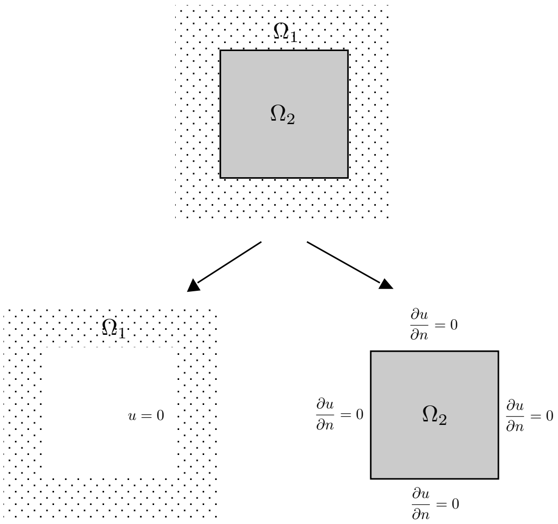

Ω_1
Ω_2
Ω_1
$u = 0$
$\frac{\partial u}{\partial n} = 0$
$\frac{\partial u}{\partial n} = 0$
Ω_2
$\frac{\partial u}{\partial n} = 0$
$\frac{\partial u}{\partial n} = 0$

Figure 5.4: Principle of kernel deflation

the null space of K_b which correspond to all six rigid body modes of the aggregate. We emphasize that by this choice of deflation subspace Z the rigid body modes are eliminated from the iterative solution process and thus, removes the bad influence of the newly acquired Neumann boundary conditions from the aggregate subdomains.

Extension of the previous experiment to an arbitrary number of aggregates, is straightforward. Assume that we only consider problems where there are r independent aggregates and one homogeneous layer of bitumen. Two arbitrary aggregates do not share elements and thus nodes. We apply the splitting $K = C + R$ where $C = K_b + \sum_r K_{a_r} - K_\Gamma$ and $R = K_\Gamma$. Matrix C contains r independent singular submatrices that correspond to the aggregates and one positive definite submatrix that corresponds to the bitumen. The deflation subspace $Z = \mathcal{N}(C) = \bigcup_r \text{span}\{Z_{a_r}\}$ with $Z_{a_r} = \{z^1_{a_r}, ..., z^6_{a_r}\}$. The dimension of Z will be $r \times 6$. With respect to the software implementation, because all aggregates are independent there is no overlap of the non zero elements in the deflation vectors. Hence, we can store Z for any problem size within just six vectors.

We assume that the stiffness of the aggregates does not change during the simulation, i.e. the Newton-Raphson iterative process for the computation of the virtual work. As the stiffness submatrices corresponding to the aggregates do not change, the rigid body modes will not change and the deflation space remains unchanged. Hence, for any simulation where the geometry and stiffness of the aggregates does not change only one evaluation of the rigid body modes is needed.

5.3.3 Computing rigid body modes of a finite element

We know from [11] that the rigid body modes of a finite element are spanned by the kernel base vectors of the corresponding element stiffness matrix. We will show a fast and cheap solution for the computation of the rigid body modes. The same principle can be easily extended to sets of finite-elements of arbitrary shape and order. We note that the rigid body modes are only defined by the geometric properties of the element.

In three dimensions, a finite-element stiffness matrix for solid mechanics has 6 rigid body motions: three translations and three rotations. For simplicity we consider a 4 noded tetrahedral element; however, all derivations can be extended to N noded elements without loss of generality. The coordinate vector of the element is given by,

$$\left\{ \begin{array}{cccccccccccc} x_1 & y_1 & z_1 & x_2 & y_2 & z_2 & x_3 & y_3 & z_3 & x_4 & y_4 & z_4 \end{array} \right\}^{\mathrm{T}}$$

A translation can be considered as a uniform displacement of every node in a given direction. To obtain three orthogonal translations, we choose the x,y and z directions, respectively. The three translation vectors are given by,

$$\begin{array}{c} \left\{ \begin{array}{cccccccccccc} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{array} \right\}^{\mathrm{T}} \\ \left\{ \begin{array}{cccccccccccc} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{array} \right\}^{\mathrm{T}} \\ \left\{ \begin{array}{cccccccccccc} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{array} \right\}^{\mathrm{T}}. \end{array}$$

The rotations can be easily described using the spherical coordinate system,

$$x = r\cos(\theta)\sin(\phi), \quad y = r\sin(\theta)\sin(\phi), \quad z = r\cos(\phi),$$

where

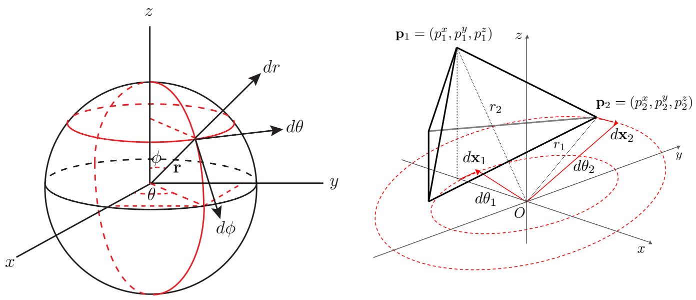
$$r = \sqrt{x^2 + y^2 + z^2}, \quad \theta = \tan^{-1}\left(\frac{y}{x}\right), \quad \phi = \cos^{-1}\left(\frac{x}{r}\right),$$

and θ and ϕ are given as as in Figure 5.5(a).

We derive a rotation $d\theta$ in the x, y-plane, hence $d\phi = 0$ and $dr = 0$. The x-y, x-z and y-z planes contain unique rotations. The corresponding vectors can be found by swapping axis. For an arbitrary point in space which has spherical coordinates (r, θ, ϕ) a change $d\theta$ in the x, y-plane yields a displacement in cartesian coordinates of,

$$dx = -r\sin(\theta)\sin(\phi)d\theta, \quad dy = r\cos(\theta)\sin(\phi)d\theta, \quad dz = 0.$$

Figure 5.5(b) shows the rotation for one element with respect to the origin over angle $d\theta$. By using above expressions, we obtain all three rotation vectors,



(a) (b)

Figure 5.5: (a) spherical coordinates, (b) rotation around origin of tetrahedral element in x, y–plane

- **rotation x–y plane**,

$$\theta_j = \tan^{-1}\left(\frac{y_j}{x_j}\right), \quad \phi_j = \cos^{-1}\left(\frac{z_j}{r_j}\right), \quad \left\{\begin{array}{c} -r_1\sin(\theta_1)\sin(\phi_1) \\ r_1\cos(\theta_1)\sin(\phi_1) \\ 0 \\ -r_2\sin(\theta_2)\sin(\phi_2) \\ r_2\cos(\theta_2)\sin(\phi_2) \\ 0 \\ -r_3\sin(\theta_3)\sin(\phi_3) \\ r_3\cos(\theta_3)\sin(\phi_3) \\ 0 \\ -r_4\sin(\theta_4)\sin(\phi_4) \\ r_4\cos(\theta_4)\sin(\phi_4) \\ 0 \end{array}\right\}$$

- **rotation y–z plane**,

$$\theta_j = \tan^{-1}\left(\frac{z_j}{x_j}\right), \quad \phi_j = \cos^{-1}\left(\frac{y_j}{r_j}\right), \quad \left\{\begin{array}{c} -r_1\sin(\theta_1)\sin(\phi_1) \\ 0 \\ r_1\cos(\theta_1)\sin(\phi_1) \\ -r_2\sin(\theta_2)\sin(\phi_2) \\ 0 \\ r_2\cos(\theta_2)\sin(\phi_2) \\ -r_3\sin(\theta_3)\sin(\phi_3) \\ 0 \\ r_3\cos(\theta_3)\sin(\phi_3) \\ -r_4\sin(\theta_4)\sin(\phi_4) \\ 0 \\ r_4\cos(\theta_4)\sin(\phi_4) \end{array}\right\}$$

- **rotation x–z plane**,

$$\theta_j = \tan^{-1}\left(\frac{z_j}{y_j}\right), \quad \phi_j = \cos^{-1}\left(\frac{x_j}{r_j}\right), \quad \left\{\begin{array}{c} 0 \\ r_1\cos(\theta_1)\sin(\phi_1) \\ -r_1\sin(\theta_1)\sin(\phi_1) \\ 0 \\ r_2\cos(\theta_2)\sin(\phi_2) \\ -r_2\sin(\theta_2)\sin(\phi_2) \\ 0 \\ r_3\cos(\theta_3)\sin(\phi_3) \\ -r_3\sin(\theta_3)\sin(\phi_3) \\ 0 \\ r_4\cos(\theta_4)\sin(\phi_4) \\ -r_4\sin(\theta_4)\sin(\phi_4) \end{array}\right\}$$

We thus compute the null space of each element matrix. Sets of elements make up the bodies of materials, as a collection of elements share a certain property and are

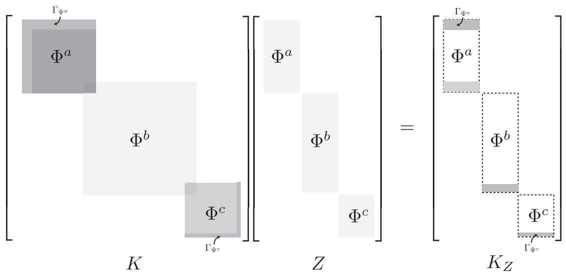


Figure 5.6: Nonzero pattern of matrix $K_Z = KZ$.

neighbors. The rigid body modes of a collection of elements is equal to the assembly of the rigid body modes of the individual elements, taking into account the multiplicity of those degrees of freedom that lie in multiple neighboring elements. In the case of asphaltic materials, we choose the element stiffness as the property for discrimination between elements. We can think of stones, bitumen and air voids. We note that we compute the rigid body modes of each independent body of material. Hence, two bodies of the same material imply 12 deflation vectors. This has a physical meaning also, two bodies will rotate and translate at the same time independently.

5.3.4 Additional work DPCG compared to PCG

The projector, P, is never computed explicitly. We compute the sparse matrix, $K_Z = KZ$, as well as the inverse of the small coarse matrix, $E = Z^{\mathrm{T}} K_Z$, beforehand. Assume the (full rank) deflation space has dimension, $d \ll n$, where, $K \in \mathbb{R}^{n \times n}$, implying, $Z \in \mathbb{R}^{n \times d}$, $E \in \mathbb{R}^{d \times d}$ and $K_Z \in \mathbb{R}^{n \times d}$. Evaluation of $\mathbf{w} = P\mathbf{v}$ is equal to $\mathbf{w} = \mathbf{v} - K_Z E^{-1} Z^{\mathrm{T}} \mathbf{v}$. Stiffness matrix K, deflation vectors Z and matrix K_Z are sparse. We compare the cost of one matrix vector product of the stiffness matrix K and the deflation matrix P by comparing the number of flops. Assume that the average number of nonzeros for each row of K, Z^{T} and K_Z is α, β and γ respectively. The total number of flops, thus work, $W_{K\mathbf{u}}$, of one matrix vector multiplication with stiffness matrix, K, is,

$$W_{K\mathbf{u}} = 2\alpha n. \tag{5.10}$$

The (cumulative) number of flops, thus work $W_{P\mathbf{u}}$, of one matrix vector multiplication with projector, P, is,

$$\begin{aligned} W_{P\mathbf{u}} &= W_{Z^{\mathrm{T}}\mathbf{v}} + W_{E^{-1}\hat{\mathbf{v}}} + W_{K_Z\tilde{\mathbf{v}}} + W_{\mathbf{v}-\bar{\mathbf{v}}} \\ &= (2\beta d) + (2d^2) + (2\gamma n) + n. \end{aligned} \tag{5.11}$$

We derive estimates for γ, and β for non overlapping deflation vectors to compare the matrix-vector multiplications for matrices, K, and, P as defined above. We assume that, in general, we have, $k = \frac{d}{6}$ rigid bodies. We illustrate the matrix-matrix multiplication of, $K_Z = KZ$, with Figure 5.6. In this specific example, the stiffness matrix, K, consists of three submatrices that correspond to rigid bodies, Φ^a, Φ^b, and, Φ^c, and boundary conditions, Γ_{Φ^a}, and, Γ_{Φ^b}. The rigid body modes, and thus, the deflation vectors, correspond to the kernels of these submatrices, hence, we only have nonzero entries in K_Z on the boundaries of the rigid bodies and due to the boundary conditions of the PDEs, therefore, $\gamma \approx 4$. We assume that on average, each rigid body owns $\frac{n}{k}$ degrees of freedom, with an average ratio of $\psi \in (0, 1)$ degrees of freedom on the boundary of the rigid body, hence, the total amount of work of multiplying K_Z, with a dense vector of dimension d is, $8\psi n$. The average number of nonzeros in the rows of Z^{T} is bounded by the average amount of nonzeros in the rigid body modes, divided by the number of rigid body modes, which is six in the three dimensional case considered here. Therefore, $\beta = 4 \times \frac{n}{k} \times \frac{1}{6} = \frac{4n}{d}$. We substitute the values of γ, and, β in Equation 5.11, and replace $W_{K_Z\tilde{\mathbf{v}}} = 2\gamma n$ with $8\psi n$, yielding,

$$W_{P\mathbf{u}} = (9 + 8\psi)n + 2d^2. \tag{5.12}$$

We equate Equation 5.10 to Equation 5.12, and solve for the number of deflation vectors, d, provided that the connectivity, $\alpha \approx \mathcal{O}(10^2)$. We obtain that, $W_{P\mathbf{u}} < W_{K\mathbf{u}}$, if,

$$d < 10^2\sqrt{n}. \tag{5.13}$$

Although the bound on the number of deflation vectors given in Equation 5.13 seems high, we emphasize that the deflation vectors considered in this section are very sparse. We conclude that for, $d \ll n$, non-overlapping deflation vectors, the amount of work for one matrix-vector multiplication with projector, P, is at most, as expensive as one matrix-vector multiplication with stiffness matrix, K.

5.3.5 Illustrative example: artificial representation of composite material

We revisit the examples of Section 5.1.1, given in Figure 5.1. All numerical experiments use the DPCG method based on the rigid body modes of the aggregates. The results are benchmarked against the homogeneous material case (I) in which there are no aggregates. The benchmark is CG with diagonal scaling and represented by the red dotted lines.

Figure 5.7 shows the performance of PCG and DPCG applied to case (II) and (III).

The results of Figure 5.7 show what we expected from the deflation. The rigid body modes are no longer active in the iterative process and only the deformation of the bitumen has to be computed. Hence, the performance of DPCG is identical compared to the homogeneous benchmark case. Moreover, in Figure 5.7 the plots for case (II) and (III) show the same performance but for a different amount of aggregates. Therefore, adding more aggregates to the domain has no influence on the performance of CG. The same behavior can be observed in Figure 5.8 which shows the performance of PCG and DPCG applied to case (II), (III) and (IV). We note that the results of Figure 5.7 are also included in Figure 5.8.

We have seen that the effective condition number of the deflated matrix PK depends on the smallest positive eigenvalue of C which is identical to the smallest eigenvalue of the bitumen submatrix. Hence, the number of iterations of CG is only bounded by the material properties of the bitumen and not by the stiffness of the aggregates.

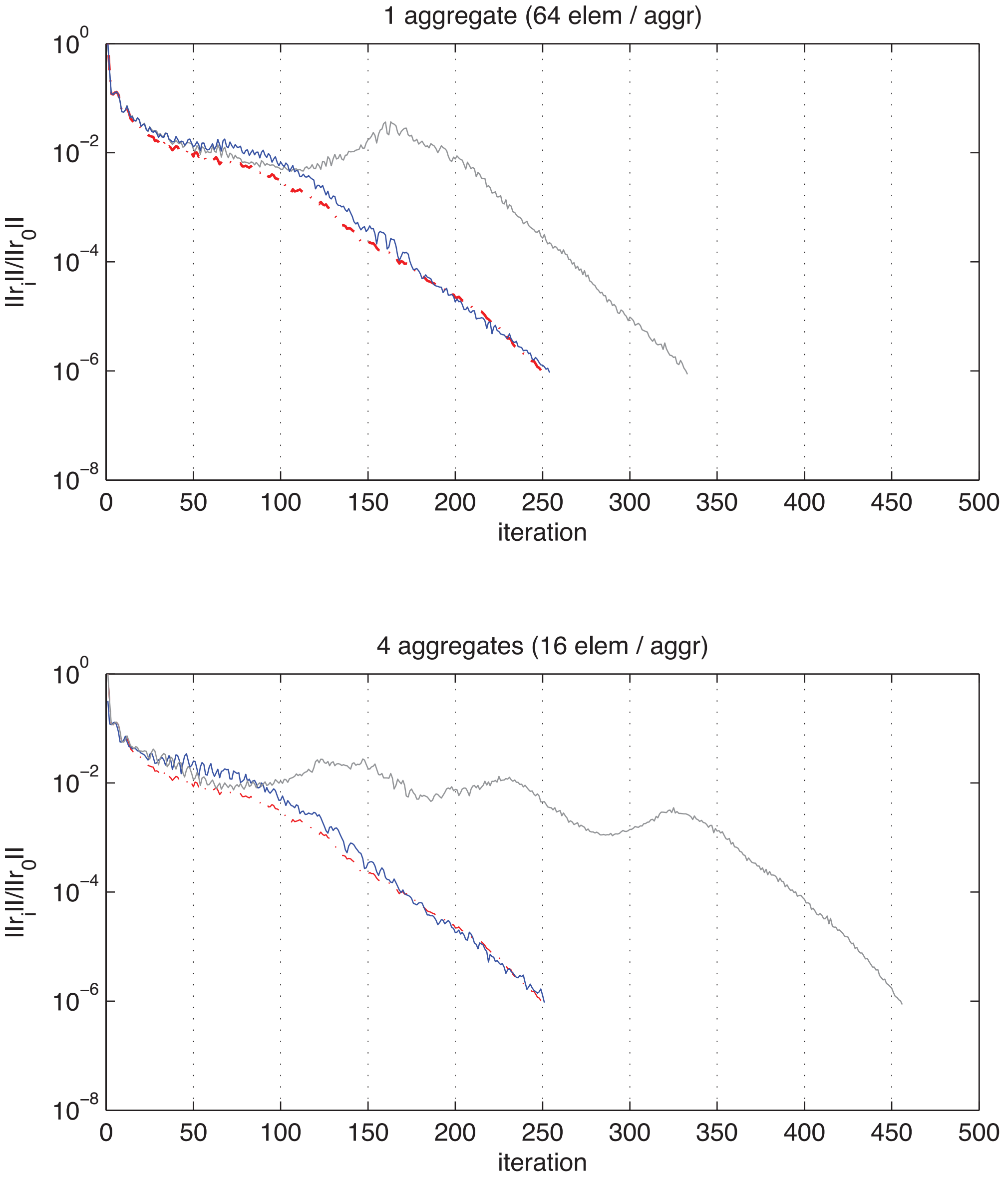


Figure 5.7: Convergence of PCG and kernel DPCG for 1 and 4 aggregates respectively. -.- PCG (homogeneous material), – PCG ($E \approx \mathcal{O}(10^3)$), – kernel DPCG ($E \approx \mathcal{O}(10^3)$).

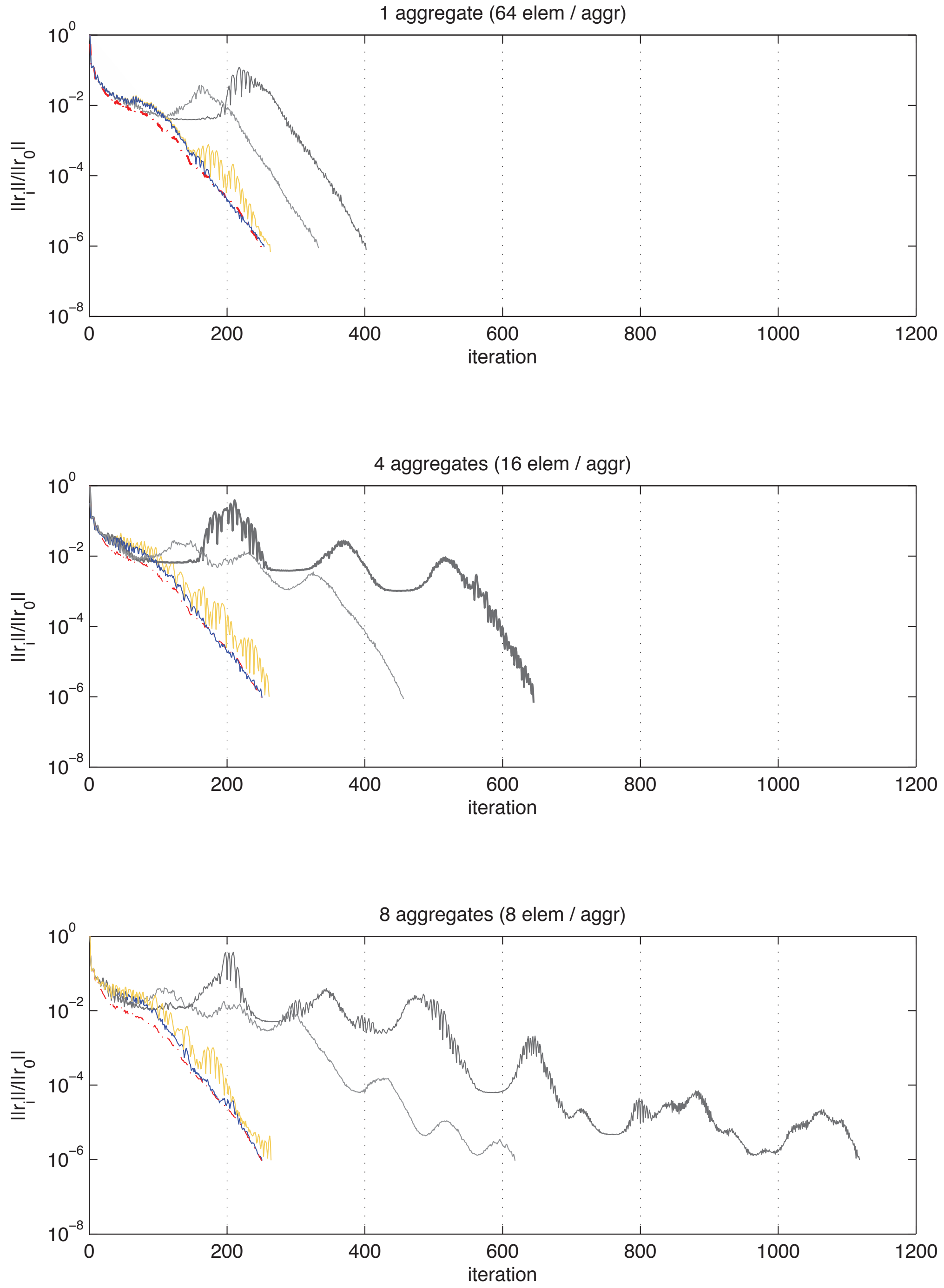


Figure 5.8: Convergence of PCG and kernel DPCG for 1, 4 and 8 aggregates respectively. -.- PCG (homogeneous material), – PCG ($E \approx \mathcal{O}(10^3)$), – PCG ($E \approx \mathcal{O}(10^5)$),– DPCG ($E \approx \mathcal{O}(10^3)$), – DPCG ($E \approx \mathcal{O}(10^5)$).

5.4 Recursive deflation

The deflation of rigid body modes considered in the first part of this chapter works only for composite materials that consist of two materials. In this section we introduce an extension to the rigid body mode deflation for composite materials that consist of an arbitrary number of materials. Because this extension on the deflation operator is rather complex, we illustrate the theory in this section with a one dimensional Poisson problem.

The definition of P given by (5.2) does not provide insight in the effect of individual deflation vectors on the spectrum of PK. The next theorem defines a recursive deflation operator which can be used for more extensive eigenvalue analysis of PK. Moreover, it will justify our choice of deflation vectors on which we elaborate later.

Definition 5.4.1. *$P^{(k)} = I - KZ_k(Z_k^{\mathrm{T}}KZ_k)^{-1}Z_k^{\mathrm{T}}$ with $Z_k = [\tilde{Z}_1, \tilde{Z}_2, ..., \tilde{Z}_k]$, where $\tilde{Z}_j \in \mathbb{R}^{n \times l_j}$ and has rank l_j.*

Theorem 5.4.1. *Let $P^{(k)}$ and Z_k as in Definition 5.4.1, then $P^{(k)}K = P_kP_{k-1}\cdots P_1K$ where $P_{i+1} = I - \tilde{K}_i\tilde{Z}_{i+1}(\tilde{Z}_{i+1}^{\mathrm{T}}\tilde{K}_i\tilde{Z}_{i+1})^{-1}\tilde{Z}_{i+1}^{\mathrm{T}}$, $\tilde{K}_i = P_i\tilde{K}_{i-1}$, $\tilde{K}_1 = P_1K$, $\tilde{K}_0 = K$, $\tilde{Z}_i^{\mathrm{T}}\tilde{K}_{i-1}\tilde{Z}_i^{\mathrm{T}}$ and $Z_i^{\mathrm{T}}KZ_i$ are non-singular because Z_i are of full rank and K is a symmetric positive definite matrix.*

Proof. by induction,

i. show $P_1K = P^{(1)}K$ where $Z_1 = \tilde{Z}_1 \in \mathbb{R}^{n \times l_1}$,

ii. assume $P_{i-1}\tilde{K}_{i-2} = \tilde{K}_{i-1} = P^{(i-1)}K$ where $Z_{i-1} = [\tilde{Z}_{i-1}, \tilde{Z}_{i-2}, \cdots, \tilde{Z}_1]$, show that $P_i\tilde{K}_{i-1} = P^{(i)}K$ where $Z_i = [\tilde{Z}_i, Z_{i-1}$, $Z_{i-1} \in \mathbb{R}^{n \times l(i-1)}$, $\tilde{Z}_i \in \mathbb{R}^{n \times l_i}$ and $l = \sum_{r=i}^{i} l_i$.

For the start of the induction we have to prove [i.]. The induction hypothesis is given by [ii.]. We first show that $P_1K = P^{(1)}K$,

$$\begin{aligned} P_1K &= K - K\tilde{Z}_1(\tilde{Z}_1^{\mathrm{T}}K\tilde{Z}_1)^{-1}\tilde{Z}_1^{\mathrm{T}}K \\ &= K - KZ_1(Z_1^{\mathrm{T}}KZ_1^{\mathrm{T}})^{-1}Z_1^{\mathrm{T}}K \\ &= P^{(1)}K. \end{aligned}$$

which implies that (i.) is proved. For (ii.) we assume $P_{i-1}\tilde{K}_{i-2} = P^{(i-1)}K$, and prove that this implies $P_i\tilde{K}_{i-1} = P^{(i)}K$,

$$\begin{aligned} P^{(i)}K &= K - KZ_i(Z_i^{\mathrm{T}}KZ_i)^{-1}Z_i^{\mathrm{T}}K \\ &= K - \begin{bmatrix} KZ_{i-1} & K\tilde{Z}_i \end{bmatrix} \left(\begin{bmatrix} Z_{i-1}^{\mathrm{T}} \\ \tilde{Z}_i^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} KZ_{i-1} & K\tilde{Z}_i \end{bmatrix} \right)^{-1} \begin{bmatrix} Z_{i-1}^{\mathrm{T}}K \\ \tilde{Z}_i^{\mathrm{T}}K \end{bmatrix} \\ &= K - \begin{bmatrix} KZ_{i-1} & K\tilde{Z}_i \end{bmatrix} E^{-1} \begin{bmatrix} Z_{i-1}^{\mathrm{T}}K \\ \tilde{Z}_i^{\mathrm{T}}K \end{bmatrix} \qquad (5.14) \end{aligned}$$

where,

$$E = \begin{bmatrix} Z_{i-1}^{\mathrm{T}}KZ_{i-1} & Z_{i-1}^{\mathrm{T}}K\tilde{Z}_i \\ \tilde{Z}_i^{\mathrm{T}}KZ_{i-1} & \tilde{Z}_i^{\mathrm{T}}K\tilde{Z}_i \end{bmatrix}$$

The matrix $E = \begin{bmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{bmatrix}$ is a symmetric 2x2 block matrix. Its inverse is defined as follows [57],

$$E^{-1} = \begin{bmatrix} E_{11}^{-1} + E_{11}^{-1}E_{12}(E_{22} - E_{21}E_{11}^{-1}E_{21}E_{11}^{-1}) & -E_{11}^{-1}E_{12}(E_{22} - E_{21}E_{11}^{-1}E_{12})^{-1} \\ -(E_{22} - E_{21}E_{11}^{-1}E_{12})^{-1}E_{12}E_{11}^{-1} & (E_{22} - E_{21}E_{11}^{-1}E_{12}^{-1})^{-1} \end{bmatrix}$$

with,

$$\Psi = \tilde{Z}_i^{\mathrm{T}}K\tilde{Z}_i - \tilde{Z}_i^{\mathrm{T}}KZ_{i-1}\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1}Z_{i-1}^{\mathrm{T}}K\tilde{Z}_i$$

it follows that

$$\begin{aligned} (E^{-1})_{11} &= \left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1} + \left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1}Z_{i-1}^{\mathrm{T}}K\tilde{Z}_i\Psi^{-1}\tilde{Z}_i^{\mathrm{T}}KZ_{i-1}\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1} \\ (E^{-1})_{12} &= -\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1}Z_{i-1}^{\mathrm{T}}K\tilde{Z}_i\Psi^{-1} \\ (E^{-1})_{21} &= -\Psi^{-1}\tilde{Z}_i^{\mathrm{T}}KZ_{i-1}\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1} \\ (E^{-1})_{22} &= \Psi^{-1} \end{aligned}$$

substitute this into (5.14) leads to,

$$\begin{aligned} P^{(i)}K &= K - \begin{bmatrix} KZ_{i-1} & K\tilde{Z}_i \end{bmatrix} \begin{bmatrix} E_{11}^{-1}Z_{i-1}^{\mathrm{T}}K + E_{12}^{-1}\tilde{Z}_i^{\mathrm{T}}K \\ E_{21}^{-1}Z_{i-1}^{\mathrm{T}}K + E_{22}^{-1}\tilde{Z}_i^{\mathrm{T}}K \end{bmatrix} \\ &= K - \left[KZ_{i-1}E_{11}^{-1}Z_{i-1}^{\mathrm{T}}K + KZ_{i-1}E_{12}^{-1}\tilde{Z}_i^{\mathrm{T}}K + K\tilde{Z}_iE_{21}^{-1}Z_{i-1}^{\mathrm{T}}K + K\tilde{Z}_iE_{22}^{-1}\tilde{Z}_i^{\mathrm{T}}K\right] \\ &= K - KZ_{i-1}\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1}Z_{i-1}^{\mathrm{T}}K \qquad (5.15) \\ &\quad -KZ_{i-1}\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1}Z_{i-1}^{\mathrm{T}}K\tilde{Z}_i\Psi^{-1}\tilde{Z}_i^{\mathrm{T}}KZ_{i-1}\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1}Z_{i-1}^{\mathrm{T}}K \\ &\quad +KZ_{i-1}\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1}Z_{i-1}^{\mathrm{T}}K\tilde{Z}_i\Psi^{-1}\tilde{Z}_i^{\mathrm{T}}K \\ &\quad +K\tilde{Z}_i\Psi^{-1}\tilde{Z}_i^{\mathrm{T}}KZ_{i-1}\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1}Z_{i-1}^{\mathrm{T}}K \\ &\quad -K\tilde{Z}_i\Psi^{-1}\tilde{Z}_i^{\mathrm{T}}K \end{aligned}$$

In order to show $P_i\tilde{K}_{i-1} = P^{(i)}K$ we now elaborate $P_i\tilde{K}_{i-1}$,

$$
\begin{align*}
P_i\tilde{K}_{i-1} &= \tilde{K}_{i-1} - \tilde{K}_{i-1}\tilde{Z}_i\left(\tilde{Z}_i^{\mathrm{T}}\tilde{K}_{i-1}\tilde{Z}_i\right)^{-1}\tilde{Z}_i^{\mathrm{T}}\tilde{K}_{i-1} \\
&= P^{(i-1)}K - P^{(i-1)}K\tilde{Z}_i\left(\tilde{Z}_i^{\mathrm{T}}P^{(i-1)}K\tilde{Z}_i\right)^{-1}\tilde{Z}_i^{\mathrm{T}}P^{(i-1)}K \\
&= K - KZ_{i-1}\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1}Z_{i-1}^{\mathrm{T}}K \\
&\quad - \left(K - KZ_{i-1}\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1}Z_{i-1}^{\mathrm{T}}K\right)\tilde{Z}_i \cdot \\
&\quad \left(\tilde{Z}_i^{\mathrm{T}}\left(K - KZ_{i-1}\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1}Z_{i-1}^{\mathrm{T}}K\right)\tilde{Z}_i\right)^{-1} \cdot \\
&\quad \tilde{Z}_i^{\mathrm{T}}\left(K - KZ_{i-1}\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1}Z_{i-1}^{\mathrm{T}}K\right) \\
&= K - KZ_{i-1}\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1}Z_{i-1}^{\mathrm{T}}K \\
&\quad - \left(K\tilde{Z}_i - KZ_{i-1}\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1}Z_{i-1}^{\mathrm{T}}K\tilde{Z}_i\right)\cdot\Psi^{-1}\cdot \\
&\quad \left(\tilde{Z}_i^{\mathrm{T}}K - \tilde{Z}_i^{\mathrm{T}}KZ_{i-1}\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1}Z_{i-1}^{\mathrm{T}}K\right) \\
&= K - KZ_{i-1}\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1}Z_{i-1}^{\mathrm{T}}K \tag{5.16} \\
&\quad -KZ_{i-1}\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1}Z_{i-1}^{\mathrm{T}}K\tilde{Z}_i\Psi^{-1}\tilde{Z}_i^{\mathrm{T}}KZ_{i-1}\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1}Z_{i-1}^{\mathrm{T}}K \\
&\quad +KZ_{i-1}\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1}Z_{i-1}^{\mathrm{T}}K\tilde{Z}_i\Psi^{-1}\tilde{Z}_i^{\mathrm{T}}K \\
&\quad +K\tilde{Z}_i\Psi^{-1}\tilde{Z}_i^{\mathrm{T}}KZ_{i-1}\left(Z_{i-1}^{\mathrm{T}}KZ_{i-1}\right)^{-1}Z_{i-1}^{\mathrm{T}}K \\
&\quad -K\tilde{Z}_i\Psi^{-1}\tilde{Z}_i^{\mathrm{T}}K
\end{align*}
$$

note that (5.15) and (5.16) are identical, so $P_i\tilde{K}_{i-1} = P^{(i)}K$.

□

Theorem 5.4.1 provides us with a theoretical framework in which we construct the deflation vectors. We will see that by subsequently adding rigid body modes of particular sets of elements to the deflation space the number of small eigenvalues of the deflated system is smaller compared to the non-deflated system.

5.4.1 Motivation: condition numbers of Deflated matrices

Let us denote the ith eigenvalue of K in nondecreasing order by $\lambda_i(K)$ or simply by λ_i. Theorem 10.2.6 in [34] provides a bound on the error of CG. After k iterations of the CG method, the error is bounded by,

$$
\left\|u - u_k\right\|_K \leq 2\left\|u - u_0\right\|_K\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k
$$

where $\kappa = \kappa(K) = \frac{\lambda_n}{\lambda_1}$ is the spectral condition number of K, and the K-norm of u is given by $||u||_K = \sqrt{u^{\mathrm{T}} K u}$.

To obtain a useful bound for the error of CG for positive semi-definite matrices we define the effective condition number of a semi-definite matrix $D \in \mathbb{R}^{n \times n}$ with corank m to be the ratio of the largest and smallest positive eigenvalue,

$$\kappa_{\text{eff}}(D) = \frac{\lambda_n}{\lambda_{m+1}}. \tag{5.17}$$

We extend Theorem 5.2.1 as Theorem 5.4.2 which implies that a bound on the condition number of $P^{(k)}K$ can be obtained.

Theorem 5.4.2. *Let $P^{(k)}$ as defined in Definition 5.4.1 and suppose there exists a splitting $K = C + R$ such that C and R are symmetric positive semi-definite with $\mathcal{N}(C) = span\{Z_k\}$ the null space of C. Then for ordered eigenvalues λ_i,*

$$\lambda_i(C) \leq \lambda_i(P^{(k)}K) \leq \lambda_i(C) + \lambda_{max}(P^{(k)}R). \tag{5.18}$$

Moverover, the effective condition number of $P^{(k)}K$ is bounded by,

$$\kappa_{eff}(P^{(k)}K) \leq \frac{\lambda_n(K)}{\lambda_{m+1}(C)}. \tag{5.19}$$

Proof. See [32] (p445). □

The effective condition number of $P^{(k)}K$ is bounded by the smallest eigenvalue of C and the largest eigenvalue of K. For a symmetric preconditioner $M = LL^{\mathrm{T}}$, e.g. diagonal scaling, we extend the result of Theorem 5.4.2 to

$$\kappa_{\text{eff}}(L^{-1}P^{(k)}KL^{-T}) \leq \frac{\lambda_n(L^{-1}KL^{-T})}{\lambda_{m+1}(L^{-1}CL^{-T})}. \tag{5.20}$$

5.4.2 Recursive Deflation strategy

In this section we introduce a strategy to construct the deflation space Z_j for $P^{(j)}K$ of Definition 5.4.1 to obtain decoupled problems using Theorems 5.4.2 and 5.4.1. Our starting point is by observing that null spaces of sets of elements are represented by the rigid body modes of those sets of elements. By choosing sets of elements we define C and the nullspace of C is our deflation space, which is by definition spanned by the rigid body modes.

We have an arbitrary FE mesh Ω consisting of elements e_i, $i = 1, ..., n$ and m materials, sorted by decreasing stiffness. We will elaborate on the importance of the ordering by material stiffness in Section 5.4.3. Material j of the FE mesh can have multiple bodies j_k which is the collection of connected elements that share the same material property. We note that each body of material induces a jump in the entries of the stiffness matrix of which the size depends on the differences in stiffness of the corresponding materials. Hence it is important to distinguish all bodies of all materials as we want to decouple those regions in the stiffness matrix. The set of elements that makes up a body l of the material j is defined as Ω_j^l, where $\Omega = \bigcup_{j=1}^{m}\{\bigcup_{l=1}^{j_k}\Omega_j^l\}$. Let $\mathcal{I} = \{i : e_i \subset \Omega\}$ be defined as the index set of Ω. The index set of Ω_j^l is $\mathcal{I}_j^l = \left\{i : e_i \subset \Omega_j^l\right\}$. We also define index set $\mathcal{I}_j^{l,\Gamma} = \left\{i : \left(e_i \subset \Omega\backslash\Omega_j^l\right) \wedge \left(e_i \cap e_k \neq \emptyset,\ \forall e_k \subset \Omega_j^l\right)\right\}$, which contains all indices of the elements of $\Omega\backslash\Omega_j^l$ that are connected to (the boundary elements of) Ω_j^l.

Start with material $j = 1$ and body $l = 1$, which corresponds to sub-mesh Ω_1^1. This yields the first splitting:

$$
\begin{aligned}
\tilde{K}_0 &= A = C_0 + R_0 \\
C_0 &= \sum_{i\in\mathcal{I}_1^1} N_{e_i}^{\mathrm{T}} K_{e_i} N_{e_i} + \sum_{i\in\mathcal{I}\backslash\left\{\mathcal{I}_1^{1,\Gamma}\cup\mathcal{I}_1^1\right\}} N_{e_i}^{\mathrm{T}} K_{e_i} N_{e_i} \\
R_0 &= \sum_{i\in\mathcal{I}_1^{1,\Gamma}} N_{e_i}^{\mathrm{T}} K_{e_i} N_{e_i}
\end{aligned}
$$

The matrix C_0 consists of the assembly of all finite elements that belong to body $l = 1$ of material $j = 1$. Matrix K_{e_i} is the element stiffness matrix of element e_i with corresponding connectivity matrix N_{e_i}.The matrix R_0 consists of the assembly of all finite elements that share nodes with the elements on the boundary of body $l = 1$ of material $j = 1$ but that are not contained within sub-mesh Ω_1^1. The first splitting yields, $\mathcal{N}(C_0) = \tilde{Z}_1$ and $P_1 = I - \tilde{A}_0\tilde{Z}_1(\tilde{Z}_1^{\mathrm{T}}\tilde{A}_0\tilde{Z}_1)^{-1}\tilde{Z}_1^{\mathrm{T}}$. By this splitting we have decoupled the first body of material 1 from all other materials. The rigid body modes of all elements corresponding to the first body of material 1 are contained in $\mathcal{N}(C_0)$. We construct $\tilde{A}_1 = P_1A = P_1(C_0 + R_0) = P_1C_0 + P_1R_0 = C_0 + \tilde{R}_0$, where $P_1C_0 = C_0$ follows by definition of P_1. Continuing with the second body of material 1 and repeating the previous decoupling step gives

$$
\begin{aligned}
\tilde{K}_1 &= P_1A = C_0 + \tilde{R}_0 = C_1 + R_1 + \tilde{R}_0 \\
C_1 &= \sum_{i\in\mathcal{I}_1^1} N_{e_i}^{\mathrm{T}} K_{e_i} N_{e_i} + \sum_{i\in\mathcal{I}_1^2\backslash\mathcal{I}_1^{1,\Gamma}} N_{e_i}^{\mathrm{T}} K_{e_i} N_{e_i} + \sum_{i\in\mathcal{I}\backslash\bigcup_{l=1}^{2}\left\{\mathcal{I}_1^{l,\Gamma}\cup\mathcal{I}_1^l\right\}} N_{e_i}^{\mathrm{T}} K_{e_i} N_{e_i} \\
R_1 &= \sum_{i\in\mathcal{I}_1^{2,\Gamma}\backslash\mathcal{I}_1^{1,\Gamma}} N_{e_i}^{\mathrm{T}} K_{e_i} N_{e_i}
\end{aligned}
$$

Hence, $\mathcal{N}(C_1) = \tilde{Z}_2$ and $P_2 = I - \tilde{A}_1\tilde{Z}_2(\tilde{Z}_2^{\mathrm{T}}\tilde{A}_1\tilde{Z}_2)^{-1}\tilde{Z}_2^{\mathrm{T}}$. Continue for all bodies and materials. At splitting $m = \sum_{j=1}^{n-1} j_k + l$ for material n and body l,

$$\begin{aligned}
\tilde{K}_{m-1} &= C_{m-2} + \tilde{R}_{m-2} = C_{m-1} + R_{m-1} + \tilde{R}_{m-2} \\
C_{m-1} &= \sum_{q=1}^{n-1} \left[\sum_{r=1}^{q_k} \left[\sum_{i \in \mathcal{P}} N_{e_i}^{\mathrm{T}} K_{e_i} N_{e_i} \right] \right] \\
&+ \sum_{r=1}^{l} \left[\sum_{i \in \mathcal{C}} N_{e_i}^{\mathrm{T}} K_{e_i} N_{e_i} \right] \\
&+ \sum_{i \in \mathcal{U}} N_{e_i}^{\mathrm{T}} K_{e_i} N_{e_i} \\
R_{m-1} &= \sum_{i \in \mathcal{B}} N_{e_i}^{\mathrm{T}} K_{e_i} N_{e_i}
\end{aligned}$$

where,

$$\begin{aligned}
\mathcal{P} &= \mathcal{I}_q^r \backslash \left\{ \bigcup_{j=1}^{q-1} \bigcup_{s=1}^{j_k} \mathcal{I}_j^{s,\Gamma} \right\} \\
\mathcal{C} &= \mathcal{I}_n^r \backslash \left\{ \bigcup_{j=1}^{n-1} \bigcup_{s=1}^{j_k} \mathcal{I}_j^{s,\Gamma} \right\} \cup \left\{ \bigcup_{s=1}^{l-1} \mathcal{I}_n^{s,\Gamma} \right\} \\
\mathcal{U} &= \mathcal{I} \backslash \left\{ \bigcup_{q=1}^{n-1} \left\{ \bigcup_{r=1}^{q_k} \mathcal{I}_q^{r,\Gamma} \cup \mathcal{I}_q^r \right\} \right\} \cup \left\{ \bigcup_{r=1}^{l} \mathcal{I}_n^{r,\Gamma} \cup \mathcal{I}_n^r \right\} \\
\mathcal{B} &= \mathcal{I}_n^{l,\Gamma} \backslash \left\{ \bigcup_{j=1}^{n-1} \bigcup_{s=1}^{j_k} \mathcal{I}_j^{s,\Gamma} \right\} \cup \left\{ \bigcup_{s=1}^{l-1} \mathcal{I}_n^{s,\Gamma} \right\}
\end{aligned}$$

Hence, $\mathcal{N}(C_{m-1}) = \tilde{Z}_m$ and $P_{m-1} = I - \tilde{K}_{m-1}\tilde{Z}_m(\tilde{Z}_m^{\mathrm{T}}\tilde{K}_{m-1}\tilde{Z}_m)^{-1}\tilde{Z}_m^{\mathrm{T}} = P$ with $P = I - AZ(Z^{\mathrm{T}}AZ)^{-1}Z^{\mathrm{T}}$ and $\mathrm{span}\{Z\} = \bigcup_{j=1}^{m} \mathrm{span}\{\tilde{Z}_j\}$. The above expression for $\tilde{K}_{m-1}$ is rather complex. We have divided the index sets needed for assembly of C_{m-1} and R_{m-1} into 4 sub-sets, $\mathcal{P}$, $\mathcal{C}$, $\mathcal{U}$ and $\mathcal{B}$. The set $\mathcal{P}$ contains all element indices that belong to body r of material q except for all elements that are included in boundary element sets of previously assembled materials and bodies. The set $\mathcal{C}$ contains all the element indices that belong to body r of current material n except for all elements that are included in boundary element sets of previously assembled materials and bodies, and the $l-1$ assembled bodies of the current material. The set $\mathcal{U}$ contains all the element indices that belong to materials and bodies that have not been assembled yet. The set $\mathcal{B}$ contains all element indices that belong to elements that lie against the boundary of body l of current material n but without all elements that are contained within boundary sets of previously assembled bodies and materials.

Illustrative example: 1D Poisson equation

In this section we look into the effects of individual deflation vectors on the spectrum of PK. As it is easier to understand and analyze the recursive deflation strategy for a small, one dimensional test case, we consider a one dimensional Poisson equation with discontinuous coefficients. In Chapter 7 we discuss the *performance* of DPCG and the recursive deflation strategy on real-life engineering applications.

The considered 1D Poisson equation reads,

$$-\frac{d}{dx}\left(c(x)\frac{du(x)}{dx}\right) = f(x), \quad x \in [0,l] \tag{5.21}$$

$$u(0) = 0, \quad \frac{du}{dx}(l) = 0 \tag{5.22}$$

where $c(x)$ is a given piecewise constant function, $u(x)$ the unknown displacement field and $f(x)$ the given source term.

We discretize Equation (5.21) with the finite element method using linear first-order shape functions and equally spaced elements of size h. It is well known that in this particular case the finite element stencil for the 1D Poisson equation reads,

$$\begin{bmatrix} c(x_i) & -c(x_{i+1}) \\ -c(x_i) & c(x_{i+1}) \end{bmatrix} \tag{5.23}$$

Introduce a FE mesh for the line $[0,l]$ including 3 domains $\Omega_1 = \{x_1,..,x_4\}$, $\Omega_2 = \{x_5,..,x_8\}$ and $\Omega_3 = \{x_9,..,x_{13}\}$.

For sake of simplicity we will write $c_i = c(x_i)$ where $i = 1,...,13$, $x_1 = h$ and $x_{13} = l$. Furthermore because c_i is constant on each material domain we will use $c_i = c_1$, $c_i = c_2$ and $c_i = c_3$ on Ω_1, Ω_2 and Ω_3 respectively.

After discretization we obtain,

$$K\mathbf{u} = hf(\mathbf{x}) \tag{5.24}$$

where,

$$K = \frac{1}{h}\begin{bmatrix}
2c_1 & -c_1 & & & & & & & & \\
-c_1 & \ddots & \ddots & & & & & \emptyset & & \\
& \ddots & 2c_1 & -c_1 & & & & & & \\
& & -c_1 & c_1+c_2 & -c_2 & & & & & \\
& & & -c_2 & \ddots & \ddots & & & & \\
& & & & \ddots & 2c_2 & -c_2 & & & \\
& & & & & -c_2 & c_2+c_3 & -c_3 & & \\
& & & & & & -c_3 & \ddots & \ddots & \\
& \emptyset & & & & & & \ddots & 2c_3 & -c_3 \\
& & & & & & & & -c_3 & c_3
\end{bmatrix}$$

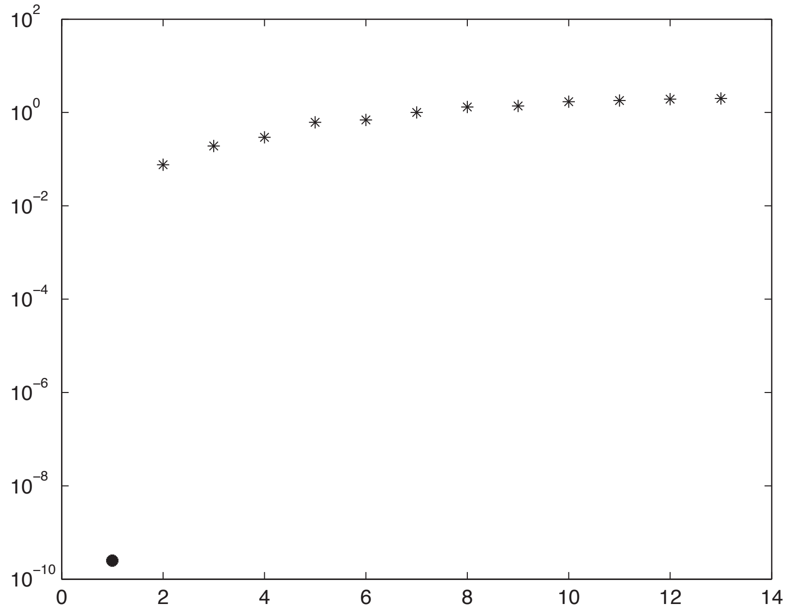


Figure 5.9: spectrum of $M^{-1}K$ where $[c_1, c_2, c_3] = [1, 10^4, 10^8]$.

and $\mathbf{u} = [u_1, u_2, ..., u_{13}]^{\mathrm{T}}$, $\mathbf{x} = [x_1, x_2, ..., x_{13}]^{\mathrm{T}}$. The stiffness matrix K of Equation (5.24) is preconditioned by $M \simeq K$. In this example we take $M = \mathrm{diag}(K)$.

The spectrum of $M^{-1}K$ is given by Figure 5.9. We first note that we only have one very small eigenvalue due to the Neumann boundary conditions. Clearly the smallest eigenvalue, which is of $\mathcal{O}\left(10^{-10}\right)$ and induced by the (only real) rigid body contained in the mesh, is much smaller compared to the other eigenvalues. Moreover, it affects the condition number of $M^{-1}K$. Now we apply the deflation strategy by finding a correct splitting of K. We sort the materials in decreasing order of diffusion. Figure 5.10 shows the sparsity pattern of the three splitting matrices C_0, C_1 and C_2. In matrix C_0 the assembly of the elements belonging to stiffest material 3, is represented by the bold crosses. The interface between weaker material 2 and 3, which goes to R_0 , is represented by the circles and all other elements are represented by the non bold crosses. The second splitting is the decoupling of material 2 from the system, again those elements are represented by the bold crosses. The interface between material 2 and 3 has been removed already, the interface between material 2 and 1 goes to R_1. The remaining splitting is the decoupling of material 1 from the boundary conditions which go to R_2.

5.4.3 Deflation vectors in the neighborhood of a jump

If some elements of a less stiff material are assigned to the element set of a stiffer material, the material stiffness matrices are not decoupled. We illustrate this with a

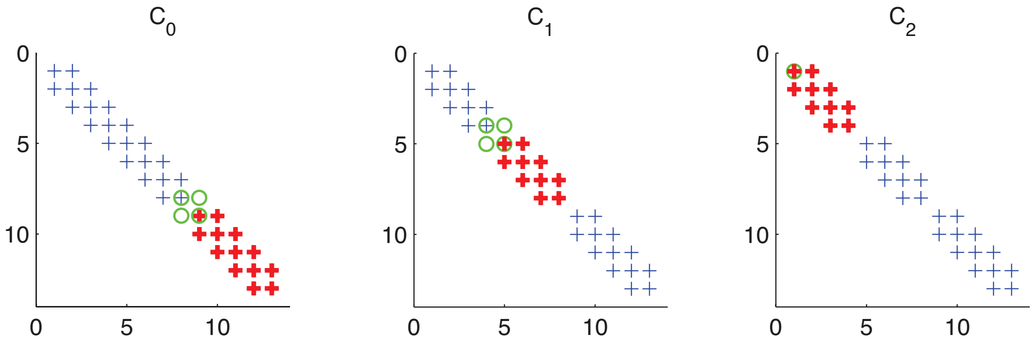


Figure 5.10: sparsity pattern C_0, C_1 and C_2. Nonzero elements represented by symbols; corresponding to deflated material, interface elements and remaining elements pictured by bold crosses, circles and non bold crosses respectively.

simple example. When a node belongs to two elements and two different materials and is assigned to the wrong (least stiff) element with respect to the splitting of K, then by applying the preconditioner the coupling between the stiffness matrices remains. For instance, the 1D Poisson problem and preconditioning based on diagonal scaling, the entry on the main diagonal is $c_1 + c_2$, with $c_1 \ll c_2$. Clearly, when decoupled correctly, we have in splitting of K only c_2 on the main diagonal of C, hence $M^{-1}C$ gives $\frac{c_2}{c_1+c_2} \approx 1$. With a wrong choice of deflation vectors, we have c_1 on the main diagonal of C, hence $M^{-1}C$ gives $\frac{c_1}{c_1+c_2} \approx \frac{1}{c_2} \ll 1$. However all other terms on the diagonal of $M^{-1}C$ will be approximately 1, introducing small eigenvalues for this material and unfavorable local spectrum of eigenvalues of $M^{-1}C$.

Illustrative example: 1D Poisson equation (continued)

We illustrate the effect of incorrect decoupling by analyzing the spectrum of the splitting matrices for the 1D Poisson equation. Figure 5.11 shows the spectrum of $M^{-1}C_i$ for the correct (star) and wrong (bold cross) choice of deflation vectors compared to the spectrum of $M^{-1}K$. After applying three deflation operations, we observe from the spectrum of $M^{-1}C_2$ that the smallest eigenvalue of the wrong choice of deflation vectors is much smaller than the smallest eigenvalue for the correct choice of deflation vectors, which coincides with the smallest eigenvalue value in the spectrum of $M^{-1}K$. Moreover, we can see from the spectrum of C_0 and C_1 that the wrong choice is clearly been made with respect to coupling of material 3 and material 2. The effective condition number of the wrong choice of deflation vectors will affect the performance of DPCG. Figure 5.12 shows the convergence of the error of DPCG and PCG for correct(+) and wrong(-) choice of deflation vectors. The performance of DPCG($^{-}$) is

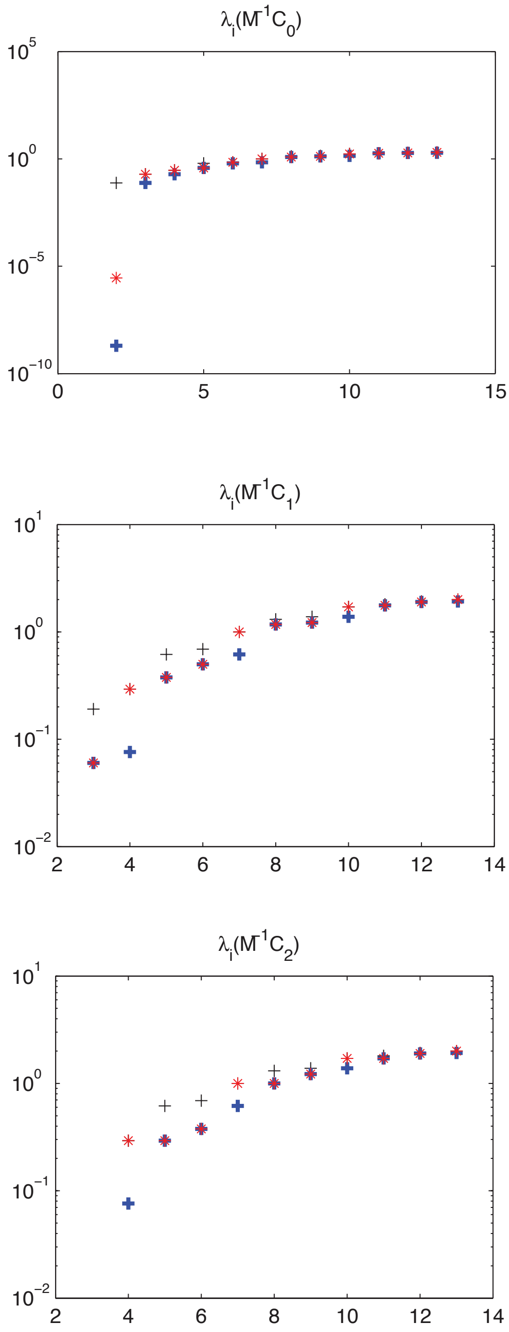


Figure 5.11: spectrum of $M^{-1}C_i$ ($\star$ correct, + wrong choice deflation vectors) compared to spectrum of $M^{-1}K$ (+)

worse than DPCG($^+$), as predicted by the eigenvalues in Figure 5.11.

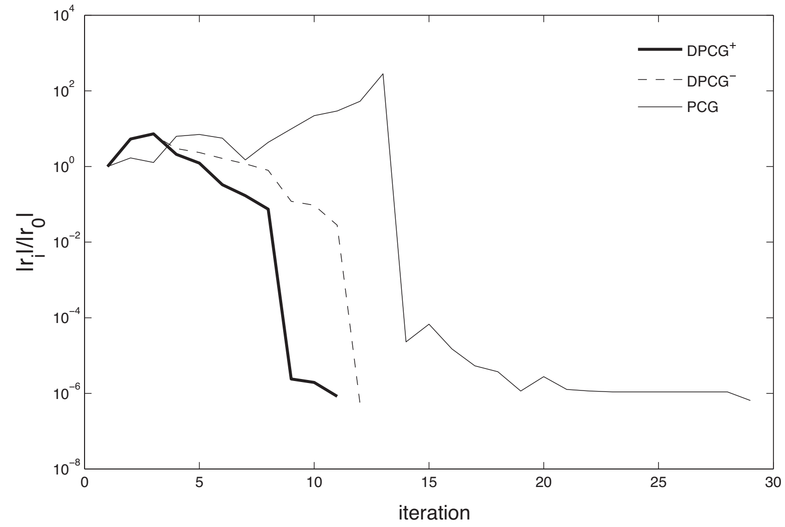


Figure 5.12: Convergence of DPCG and PCG where $[c_1, c_2, c_3] = [1, 10^4, 10^8]$ and DPCG$^+$, DPCG$^-$ represent correct and wrong choice of deflation vectors respectively.

6

Parallel implementation deflation

We have seen in Chapter 3, 4, and 5, that for the simulation of *inhomogenous* (composite) materials, the differences in properties of materials lead to large differences in the entries of the resulting stiffness matrices. We have shown that these jumps in coefficients slow down the convergence of the PCG method for a standard choice of preconditioners. By decoupling regions with homogeneous material properties, and, thus, deflating the unfavorable eigenvalues from the spectrum of the stiffness matrix, we obtained the more robust DPCG method. The deflation vectors are constructed from the rigid body modes of sets of elements that form homogeneous bodies of material. This deflation strategy is an extension of the technique of subdomain deflation, introduced in [66].

The applications involving composite materials demand for massive, unstructured meshes, yielding systems with millions of degrees of freedom. Therefore, the stiffness matrices in the simulations considered in this thesis are large and sparse. Moreover, realistic simulations of the long-term effects of (continuous) loading of structures require many Newton-Raphson iterations involving incremental loads and the evaluation of the Jacobian, thus, stiffness matrix. These implicit constraints on the computational resources as well as on the time needed for computation are met by using parallel computations.

In this chapter we discuss the parallelization of the solution algorithm in the framework of an existing software package. We will discuss the principle of domain decomposition and define parallel matrix-vector products as well as parallel inner products. We introduce an algorithm for the parallel construction of the deflation vectors, and we show how to parallelize the deflation operator and, thus, the DPCG method, using parallel linear operators.

We give a short overview on the use of subdomain deflation (as a complementary operation to rigid body modes deflation) to increase the efficiency of preconditioners that have limited global error reduction capabilities. In the chapter on future and recommended research, we further elaborate on this idea within the framework of enhanced deflations vectors.

In the last part of this chapter, we give a brief introduction to the parallel implementation of the PCG method and Smoothed Aggregation Algebraic Multigrid

(SA-AMG).

6.1 Parallel computing

In this section, we discuss the parallel implementation of the solution algorithms. In the framework of this thesis, we consider a domain, which is the collection of bodies that form a composite material. We mesh the domain, yielding a set of elements, grid nodes, and corresponding degrees of freedom.

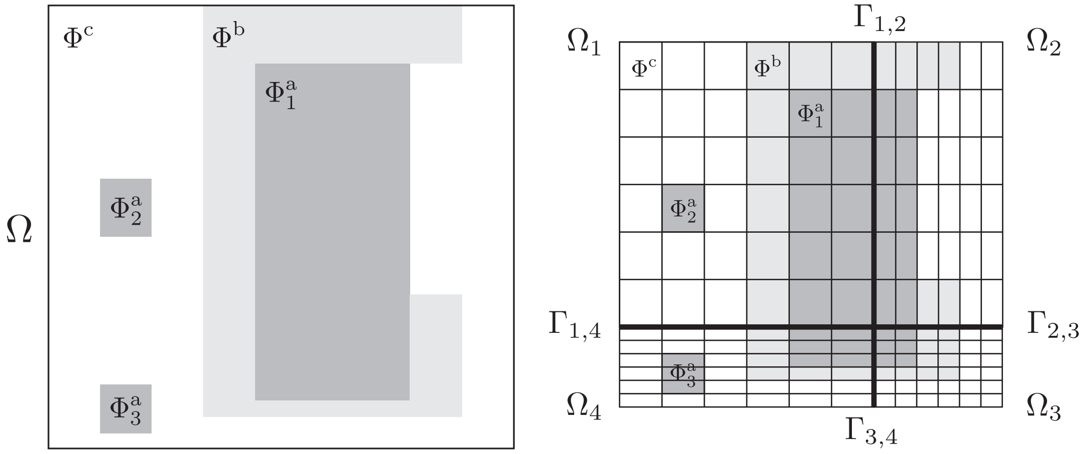
We introduce a (computing) machine that has a fixed number of processors that can work on one process at a time. The aim of parallel computing is to identify sets of tasks that can be done concurrently and to distribute those tasks across processes in such a way that the total computation time to finish all tasks, compared to the sequential evaluation, scales with the number of processes.

The maximum number of tasks that can be executed in parallel is known as the maximum degree of concurrency [36]. This degree of concurrency is determined by the decomposition of the data on which the tasks are performed. In general, a good parallel implementation of an algorithm will maximize the use of concurrency by mapping data-independent tasks onto different processes; hence, this will minimize the total completion time by ensuring processes are available to execute tasks as soon as tasks become executable.

The decomposition of the data is determined by the application. In the framework of this thesis, our data is the set of connected elements that define the mesh. We use graph partitioning to cut the mesh into smaller sets of connected meshes. We identity tasks as the operations on the matrices and vectors that correspond to those smaller meshes. The smaller meshes determine subdomains, which together form the original domain, yielding the domain decomposition.

6.1.1 Domain decomposition

We use parallelism based on domain decomposition as found in [36, 27]. We consider a global domain, Ω, which consists of E elements. We use the multilevel K-way graph partitioning algorithm in ParMETIS [52] to divide the elements over D non-overlapping sets of elements, yielding the domain decomposition, $\Omega = \bigcup_{d=1}^{D} \Omega_d$. Each subdomain holds E_d elements; hence, $E = \sum_{d=1}^{D} E_d$. Elements can share nodes and the associated degrees of freedom that lie in multiple subdomains, but no element is contained in more than one subdomain. Element-wise tasks can be evaluated independently for each subdomain, but the values of any quantity at shared nodes must be communicated for each subdomain after finishing the task. This yields communication between the boundaries of the subdomains. We emphasize that this domain decomposition is very natural when using finite elements, as many important tasks, such as



(a) (b)

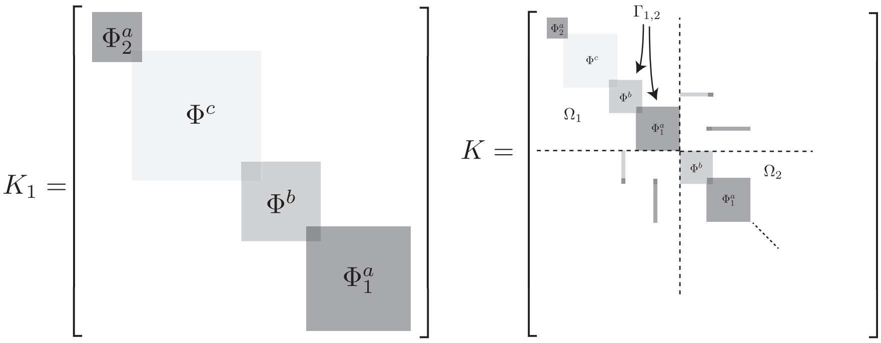
Figure 6.1: Schematic representation of domain decomposition of composite material consisting of five bodies

the evaluation of the internal forces, computation of matrix-vector products and inner products, only need communication between the subdomains after the task is finished.

We illustrate the domain decomposition with an example given by Figure 6.1(a), 6.1(b). Given in Figure 6.1(a) is domain Ω, containing three materials, a, b, and, c, divided over five bodies, Φ_1^a, Φ_2^a, Φ_3^a, Φ^b, and, Φ^c. In Figure 6.1(b), we mesh the domain, Ω, apply the graph partitioning and obtain subdomains Ω_i, where $i = 1, .., 4$. The boundaries between the subdomains, i, j, are given by $\Gamma_{i,j}$.

Subdomain mapping operators

We define two operators for mapping vectors and matrices on subdomains onto the domain and scaling of vectors for shared nodes in multiple subdomains. The mapping operator, M_d, is essentially identical to the finite-element connectivity matrix, N_e, for assembling stiffness matrices, K_e, into $K \in \mathbb{R}^{n \times n}$. Each subdomains holds n_d degrees of freedom. The operator, M_d, has dimension $n^d \times n$, and consists of one and zero entries. We can map vector $\mathbf{u}_d$ from subdomain Ω_d onto domain Ω by $\mathbf{u} = M_d^{\mathrm{T}} \mathbf{u}_d$. The averaging operator, W_d, is diagonal and has dimension, $n_d \times n_d$. It contains ones on the main diagonal when the corresponding degree of freedom lies only in subdomain, Ω_d. When multiple subdomains share a degree of freedom the diagonal entry of W_d is 1 over the number of subdomains that share the degree of freedom.



(a) (b)

Figure 6.2: non-zero pattern for local stiffness matrix, K_1, and, global stiffness matrix, K.

Local and global stiffness matrix

We define the global stiffness matrix as the assembly of all the local stiffness matrices defined on the subdomains. We compute the global stiffness matrix, K, as follows,

$$K = \sum_{d}^{e} M_d^{\mathrm{T}} K_d M_d, \tag{6.1}$$

where, $K_d \in \mathbb{R}^{n_d \times n_d}$, is the local stiffness matrix corresponding to subdomain, Ω_d.

We illustrate the assembly of the global stiffness matrix with the example of Figure 6.1(a) in Figure 6.2(a) and 6.2(b). The assembled local stiffness matrix corresponding to subdomain Ω_1 is given by Figure 6.2(a). The colored squares represent the (potential) nonzero entries in the local stiffness matrix. All bodies of subdomain Ω_1 are represented in the local stiffness matrix. The colored squares show overlap which corresponds to the shared degrees of freedom of all the nodes that lie on the interface between bodies. The ordering of the degrees of freedom determines the mapping of the local stiffness matrix to the global stiffness matrix. In the example of Figure 6.1(b), the degrees of freedom contained in subdomain Ω_1 correspond to the first block in the global stiffness matrix. We assign the degrees of freedom of subdomain Ω_2 to the second block, taking into account the connectivity between the degrees of freedom on the subdomain boundaries and interior points. In the global stiffness matrix, degrees of freedom are uniquely defined, hence, the first block contains the node contributions of the degrees of freedom on the boundaries, $\Gamma_{1,2}$, $\Gamma_{1,3}$, etc.

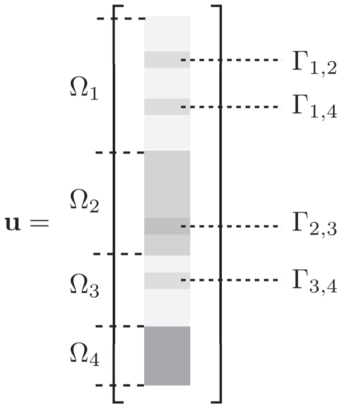


Figure 6.3: Assembled global vector, $\mathbf{u} \in \mathbb{R}^n$.

Local and global vectors

We define global vectors as the assembly of all the local vectors defined on the subdomains. We compute the global vector, $\mathbf{u}$, as follows,

$$\mathbf{u} = \sum_{d}^{e} M_d^{\mathrm{T}} \mathbf{u}_d, \tag{6.2}$$

where, $\mathbf{u}_d \in \mathbb{R}^{n_d}$.

We illustrate the assembly of the global vector, $\mathbf{u}$, with the example of Figure 6.1(a) and 6.3. The colored regions correspond to the entries of the subdomains. The ordering of the degrees of freedom determine the mapping of the local vector to the global vector. In this example, the degrees of freedom of subdomain, Ω_1, correspond to the first block in the global vector. This block also contains the node contributions of the degrees of freedom that lie on the interface between subdomains, Ω_1, Ω_2, and Ω_3.

Remark. Although we define global matrix, K, and global vector, $\mathbf{u}$, these quantities are never explicitly formed in the parallel implementation of the domain decomposition. We store all matrices and vectors locally and by means of a subdomain mapping for each degree of freedom we communicate and process the node contributions of the degrees of freedom that lie on the interfaces between subdomains.

Parallel matrix-vector product

We define the global matrix-vector product as $K\mathbf{u} = \mathbf{v}$, where $K \in \mathbb{R}^{n \times n}$ and $\mathbf{u} \in \mathbb{R}^n$. The parallel matrix-vector product is evaluated by computing $\{K_1\mathbf{u}_1, ..., K_D\mathbf{u}_D\}$, and, combining, $\{\mathbf{v}_1, ..., \mathbf{v}_D\}$, where, K_d and $\mathbf{u}_d$ have dimension, $n^d \times n^d$ and $n^d \times 1$ in subdomain Ω_d respectively. We have $\mathbf{v} = \sum_{d=1}^{D} M_d^{\mathrm{T}} \mathbf{v}_d$. We emphasize that in this formulation the entries of the shared degrees of freedom in the vectors, $\mathbf{u}_d$, should be identical for each domain it is defined on.

Parallel inner product

We define the global inner product as $\lambda = \mathbf{u}^{\mathrm{T}} \mathbf{u}$. The parallel dot product is computed as $\lambda_d = \mathbf{u}_d^{\mathrm{T}} W_d \mathbf{u}_d$, where W_d and u_d have dimension $n^d \times n^d$ and $n^d \times 1$, in subdomain Ω_d respectively. We have, $\lambda = \sum_{d=1}^{D} \lambda_d$. We emphasize that in this formulation the entries of the shared degrees of freedom in the vectors, $\mathbf{u}_d$, should be identical for each domain it is defined on.

6.2 Parallel Deflated Preconditioned Conjugate Gradient method

The parallelization of the deflation operator in the DPCG method given by Algorithm 7 involves two steps. First the construction of the deflation matrix, Z, on each subdomain and, second, the evaluation of $PK\mathbf{p}_j$ for each iteration of DPCG.

6.2.1 Building-blocks of parallel DPCG

We construct the rigid body modes in two steps. After we identify the sets of elements, i.e. rigid bodies, that share the same material properties, we compute the rigid body modes of those sets by using the formulation of rigid body modes for FE elements given in Chapter 5, Section 7.6.

Parallel coloring algorithm: construction and identification of rigid bodies distributed over subdomains

The identification of the rigid bodies is given by Algorithm 8. In three dimensions, each rigid bodies has six rigid body modes and, hence, adds six deflation vectors to the deflation space.

Algorithm 8 Identification of rigid bodies in FE mesh

Target: Given d materials, identify d_j rigid bodies for material j.
Given FE mesh with elements $\Lambda = \{\Lambda_1, ..., \Lambda_E\}$ and nodes $\Sigma = \{\sigma_1, ..., \sigma_N\}$,
Define element i as $\Lambda_i = \{\Sigma_{\Lambda_i}, \chi_{\Lambda_i}, \theta_{\Lambda_i}, \gamma_{\Lambda_i}\}$,
with

- nodeset, $\Sigma_{\Lambda_i} = \bigcup_{j \in \mathcal{I}^{\Sigma}_{\Lambda_i}} \sigma_j$ where, $\mathcal{I}^{\Sigma}_{\Lambda_i}$ contains indices of nodes of element Λ_i,
- neighboring elements, $\chi_{\Lambda_i} = \bigcup_{j \in \mathcal{I}^{\Lambda}_{\Lambda_i}} \Lambda_j$ where, $\mathcal{I}^{\Lambda}_{\Lambda_i}$ contains indices of neighboring elements of element Λ_i,
- material type, θ_{Λ_i},
- rigid body, γ_{Λ_i}

```
for $j = 1, ..., d$ do
   set $d_j = 0$
   for $i = 1, ..., E$ do
      select element, $\Lambda_i$, with $\chi_{\Lambda_i}$, $\theta_{\Lambda_i}$, and, $\gamma_{\Lambda_i}$.
      assign_element_to_body($\Lambda_i$) {
      if $\theta_{\Lambda_i} = j$ then
         if $\gamma_{\Lambda_i} = 0$ then
            $d_j = d_j + 1$, set, $\gamma_{\Lambda_i} = d_j$.
         end if
         for all $\Lambda_k \in \chi_{\Lambda_i}$ do
            select element, $\Lambda_k$, with, $\chi_{\Lambda_k}$, $\theta_{\Lambda_k}$, and, $\gamma_{\Lambda_k}$.
            if $\theta_{\Lambda_k} = j$ then
               if $\gamma_{\Lambda_k} = 0$ then
                  set $\gamma_{\Lambda_k} = \gamma_{\Lambda_i}$,
                  put element, $\Lambda_k$, on the heap of element, $\Lambda_i$.
               end if
            end if
         end for
      end if
      }
      for all $\Lambda_k$ on heap of $\Lambda_i$ do
         assign_element_to_body($\Lambda_k$)
      end for
   end for
end for
```

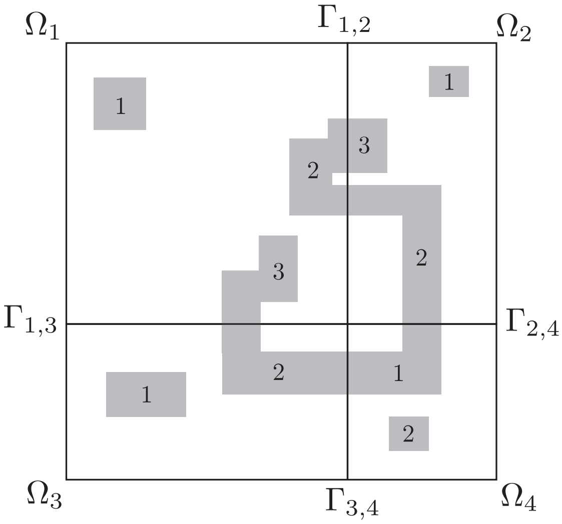


Figure 6.4: Rigid bodies divided over subdomains.

The construction of the rigid bodies by Algorithm 8 is a serial algorithm. The recursive call to function assign_element_to_body does not allow for easy parallelization. We explain the concept of the parallel search algorithm by the example given in Figure 6.4. The concept is based on a global reduction of connected rigid bodies. We send the constructed sets of local rigid bodies to a master process, after which we apply serial Algorithm 8 to these connected sets again, yielding the global numbering of rigid bodies. In Figure 6.4 we have four subdomains which all contain two materials, and multiple rigid bodies. We assume that we have identified the rigid bodies for one particular material in each domain with Algorithm 8. Clearly, the rigid bodies 2 and 3 of subdomain Ω_1, the rigid bodies 2 and 3 of subdomain Ω_2, the rigid body 1 of subdomain Ω_4, and the rigid body 2 of subdomain Ω_3 are all connected and form one rigid body. We introduce a global numbering of rigid bodies that 'unifies' the connected rigid bodies of the neighboring subdomains yielding one rigid body, and, hence, five global rigid bodies remain. The global numbering is constructed as follows,

1. For all elements on the boundaries between the subdomains, communicate the local number of the rigid body the element is contained in to the neighboring elements of the neighboring domains.

2. Send the list of local rigid bodies and domain connectivity to a master process. In the example of Figure 6.4, the list is given by Table 6.1.

3. Give rigid bodies temporary global number on master process. In the example of Figure 6.4, we have 10 rigid bodies. We number sequentially, starting with domain Ω_1; hence, we obtain the connectivity as given in Table 6.2.

Ω_1	1	2	3	4
1	x			
2	x	2, 3		
3	x		2	

Ω_2	1	2	3	4
1		x		
2	2	x		1
3	2	x		

Ω_3	1	2	3	4
1			x	
2	3		x	1

Ω_4	1	2	3	4
1		2	2	x
2				x

Table 6.1: Local number of connected rigid body modes in neighboring domains

4. Construct set of elements $\Lambda = \{\Lambda_1, ..., \Lambda_l\}$, where global rigid body i is defined as $\Lambda_i = \{\emptyset, \chi_{\Lambda_i}, \theta_{\Lambda_i}, \gamma_{\Lambda_i}\}$, with χ_{Λ_i} the connected global rigid bodies, θ_{Λ_i} the material type, and γ_{Λ_i} the definite global rigid body numbers. In this example $l = 10$.
5. Apply Algorithm 8 to set $\Lambda = \{\Lambda_1, ..., \Lambda_l\}$. The sets of elements and the definite global numbering is given by Table 6.3. We can clearly see from the example of Figure 6.4 that we should obtain 5 global rigid bodies which are $\Lambda_1, \{\Lambda_2, \Lambda_3, \Lambda_5, \Lambda_6, \Lambda_8, \Lambda_9\}, \Lambda_4, \Lambda_7$ and Λ_{10}.
6. Master process broadcasts all global rigid body numbers to subdomains.

global	1	2	3	4	5	6	7	8	9	10
(Ω_i, local)	(1,1)	(1,2)	(1,3)	(2,1)	(2,2)	(2,3)	(3,1)	(3,2)	(4,1)	(4,2)

Table 6.2: Mapping temporary global numbering rigid body modes to local numbering.

$\Lambda_1 = \{\emptyset, \emptyset, 1, \mathit{1}\}$
$\Lambda_2 = \{\emptyset, \{\Lambda_5, \Lambda_6\}, 1, \mathit{2}\}$
$\Lambda_3 = \{\emptyset, \{\Lambda_8\}, 1, \mathit{2}\}$
$\Lambda_4 = \{\emptyset, \emptyset, 1, \mathit{3}\}$
$\Lambda_5 = \{\emptyset, \{\Lambda_2, \Lambda_9\}, 1, \mathit{2}\}$
$\Lambda_6 = \{\emptyset, \{\Lambda_2\}, 1, \mathit{2}\}$
$\Lambda_7 = \{\emptyset, \emptyset, 1, \mathit{4}\}$
$\Lambda_8 = \{\emptyset, \{\Lambda_3, \Lambda_9\}, 1, \mathit{2}\}$
$\Lambda_9 = \{\emptyset, \{\Lambda_5, \Lambda_8\}, 1, \mathit{2}\}$
$\Lambda_{10} = \{\emptyset, \emptyset, 1, \mathit{5}\}$

Table 6.3: Temporary set of rigid bodies and definite global numbering.

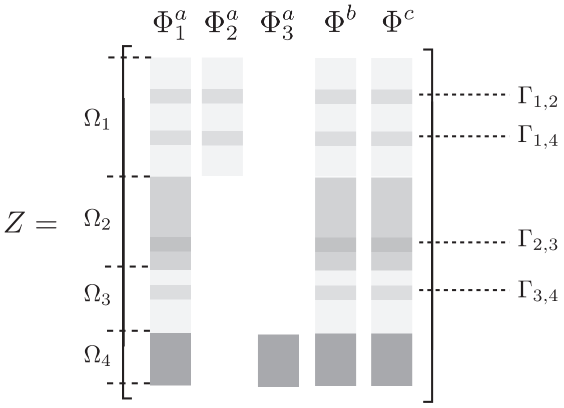


Figure 6.5: Deflation matrix Z, divided over subdomains.

Computing rigid body modes in parallel

After we identified the global rigid bodies, i.e. sets of elements, by global reduction and recursively applying Algorithm 8, we compute the rigid body modes by means of the formulation of rigid body modes for FE elements given in Section 5.3.3 of Chapter 5.

We explain the storage and construction of the deflation vectors by the example of Figure 6.1(a). We apply Algorithm 8 to mesh and subdomains of Figure 6.1(a) and, thus, identify five rigid bodies, Φ_1^a, Φ_2^a, Φ_3^a, sΦ^b, and, Φ^c, corresponding to three different materials, a, b, and c. The distribution of the global degrees of freedom over the subdomains is illustrated by vector, $\mathbf{u}$, in Figure 6.3. The entries of deflation matrix, Z, containing the $6 \times 5 = 30$ deflation vectors, have the same distribution.

In Figure 6.5, we illustrate the distribution and zero pattern of the deflation matrix, Z, for the example of Figure 6.1(a). We observe that the rigid body modes of rigid body, Φ_2^a, are defined on subdomains Ω_2, Ω_3, and Ω_4, but have only zero entries, because this rigid body is not contained in either one of these subdomains. We emphasize that this approach is different from the distribution of the entries of the global stiffness matrix, K. We define the column and row index space of a matrix as the space from which we select the indices that correspond to the nonzero entries of that matrix. The column index space, as well as the row index space of matrix K, have as many entries as degrees of freedoms in the matrix and are distributed over all subdomains corresponding to the distribution of a global vector. However, the column index space of the deflation matrix, Z, does not coincide with the distribution of the degrees of freedom, but with the distribution of the rigid bodies, and would, therefore, not be distributed over the subdomains. Hence, to simplify the parallel administration, we keep the total number of deflation vectors on all subdomains, and we consider deflation matrix Z, as a collection of global vectors, and not a sparse global matrix. This approach leads to no additional memory requirements and only a negligible amount of additional parallel communication. Only the rigid bodies that are contained in a subdomain induce nonzero entries in the corresponding deflation vectors, and because of the sparse data structure, only the row and column indices and the values of *nonzero* entries are stored. Moreover, as only nonzero entries are communicated to other domains, additional communication is needed for shared degrees of freedom on the subdomain boundaries and for parallel matrix and vector operations.

Efficient computation of P in parallel

In the previous section we discussed the mapping of the column index and row index space of the deflation matrix, Z, onto the subdomains. The computation of rigid body modes only requires the element matrices and is based on the node coordinates; hence, no parallel communication is needed for the assembly of the distributed deflation matrix. We store the nonzero elements of the deflation matrix, Z, in a sparse data structure, yielding a small memory overhead and efficient computation.

We consider the evaluation of $PK\mathbf{p}$,

$$PK\mathbf{p} = K\mathbf{p} - KZE^{-1}Z^{\mathrm{T}}K\mathbf{p},$$

where $K \in \mathbb{R}^{n\times n}$, $Z \in \mathbb{R}^{n\times k}$. Here, $K\mathbf{p} = \mathbf{y}$, is computed as usual, while $K_Z = \tilde{Z} \in \mathbb{R}^{n\times k}$ and $E^{-1} = (Z^{\mathrm{T}}K_Z)^{-1} \in \mathbb{R}^{k\times k}$ are computed only once, before entering the Krylov process (iteration loop). Hence, for each iteration of DPCG, we have three extra operations compared to PCG,

$$\begin{aligned} Z^{\mathrm{T}}\mathbf{y} &= \tilde{\mathbf{y}}, \quad \tilde{\mathbf{y}} \in \mathbb{R}^{k}, \\ E^{-1}\tilde{\mathbf{y}} &= \hat{\mathbf{y}}, \quad \hat{\mathbf{y}} \in \mathbb{R}^{k}, \\ \tilde{Z}\hat{\mathbf{y}} &= \bar{\mathbf{y}}, \quad \bar{\mathbf{y}} \in \mathbb{R}^{n}. \end{aligned}$$

Communication between subdomains is needed for the computation of K_Z, E, and Z^{T}. The computation of K_Z is, in fact, k matrix-vector products of $K\mathbf{z}_i$, where vector $\mathbf{z}_i \in \mathbb{R}^n$ is the ith column of deflation matrix Z. In each iteration, computing $Z^{\mathrm{T}}y$ involves one parallel communication at the cost of k parallel inner products. The coarse matrix, E, is determined at the cost of $k \times k$ parallel inner products. The decomposition of coarse matrix, E, is computed sequentially on each CPU by a direct method such as PARDISO [76] and, thus, we solve $E^{-1}\tilde{\mathbf{y}}$ in each Krylov iteration by back substitution.

Remark. The number of Newton-Raphson iterations for highly non-linear material models may be large if the Jacobian does not 'follow' the deformation of the body, i.e. if the stiffness matrix, which is the tangent of the non-linear virtual work equation, is not adapted in every iteration. In the FE software CAPA-3D, the stiffness matrix is

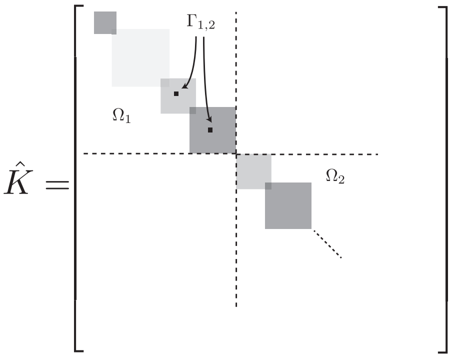


Figure 6.6: Sparsified global stiffness matrix, $\hat{K}$; no connectivity between nodes in subdomains of domain decomposition except for boundary nodes.

recomputed at every load or time step, for static and dynamic mechanics, respectively. Hence, as the stiffness matrix is kept constant, the deflation matrix Z, matrix K_Z, the coarse matrix, E, as well as the decomposition of E, are determined only once every load or time step.

6.3 Subdomain deflation

We consider the subdomains of the domain decomposition method in Section 6.1.1 as connected sets of elements, i.e. rigid bodies. These rigid bodies have no physical meaning as they consist of bodies with varying material properties. We may compute the rigid body modes of these subdomains, and use them as deflation vectors, but no direct relation between the eigenvalues in the spectrum of the stiffness matrix, K, and these deflation vectors exists.

However, some parallel preconditioners have limited global error reduction properties; we refer to these parallel preconditioners as local preconditioners. In this thesis, we use the ILU preconditioner as a local preconditioner. For this preconditioner, all off-diagonal entries in the stiffness matrix corresponding to the connectivity between subdomains are disregarded. This effect is best illustrated by the assembly of the global stiffness matrix, K, in Figure 6.2(b). When we disregard the connectivity between nodes of elements that lie on the boundaries of the subdomains, we get the splitting

$$K = \hat{K} + \Gamma, \tag{6.3}$$

where the structure of sparsified matrix, $\hat{K}$, is given in Figure 6.6. We assemble the local stiffness matrices in matrix $\hat{K}$, but remove the contributions from nodes *of boundary elements* and add those to the 'boundary' matrix, Γ, except for the contributions to the main diagonal of $\hat{K}$ from the nodes *on the subdomain boundaries*, which are illustrated by the black squares in Figure 6.6.

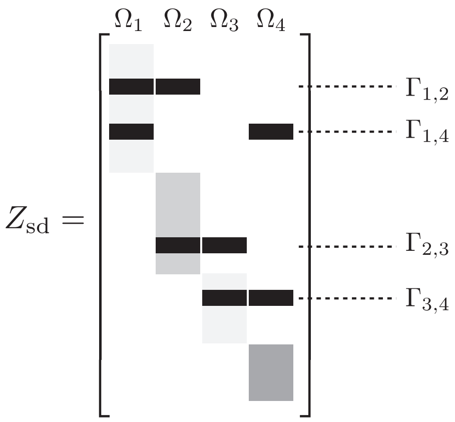


Figure 6.7: Non-zero pattern subdomain deflation vectors, $Z_{\mathrm{sd}} \in \mathbb{R}^{n\times(4\times6)}$, for domain decomposition with 4 domains.

In Section 5.3 of Chapter 5, we show how to decouple regions in the stiffness matrix by means of deflation. We precondition those regions with a local preconditioner, such as the ILU preconditioner considered in this section. In the context of domain decomposition and parallel computing, we choose the subdomains as the decoupled regions and obtain an embarrassingly parallel global preconditioner based on local preconditioners.

The decoupling of regions in the stiffness matrix is determined by the choice of the deflation vectors. We introduced the decoupling of regions corresponding to rigid bodies based on the splitting of the stiffness matrix in Theorem 5.2.1. We extend this decoupling to the deflation of subdomains. We apply Theorem 5.2.1 to the splitting of Equation 6.3. The matrix, C, corresponds to matrix $\hat{K}$ of Equation 6.3. Clearly, matrix $\hat{K}$ is semi-positive definite, as is the interface matrix, Γ. The matrix $\hat{K}$ has D independent regions, with $D \times 6$ null space vectors, thus, deflation vectors. Those subdomain deflation vectors are defined by deflation matrix, $Z_{\mathrm{sd}} \in \mathbb{R}^{n\times(D\times6)}$. For the example of Figure 6.6, the non-zero pattern of matrix, Z_{sd}, is given by Figure 6.7. The subdomain deflation vectors have non-zero entries only on the domains that 'own' the corresponding degrees of freedom. We take into account the overlap between the domains. The corresponding entries for the overlap between subdomains for the example of Figure 6.6 is illustrated by the black rectangles in Figure 6.7. We assume we have two subdomains Ω_1 and Ω_2, that share two degrees of freedom on the boundaries of these domains α and β. The local stiffness matrices contain the contributions of those degrees of freedom, hence, we obtain contributions of both subdomains on the main diagonal of $\hat{K}$. We compute the null space vectors by considering the subdomains as sets of elements and using the algorithm for computing rigid body modes given in Section 5.3.3 of Chapter 5. However, to obtain the correct null space vectors, we scale the rigid body modes of the subdomains for the overlapping

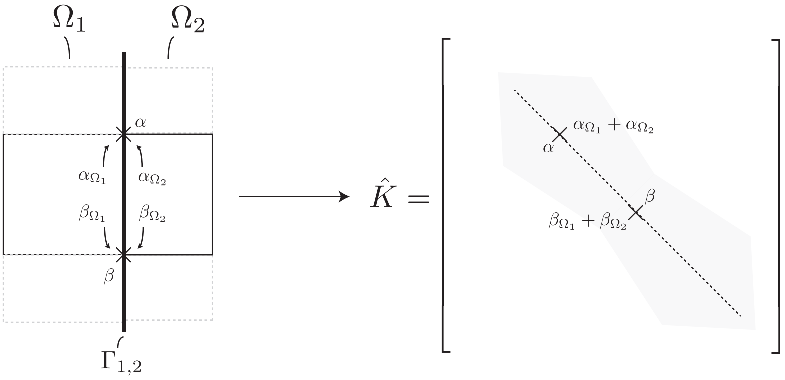


Figure 6.8: Overlap between subdomains, and the corresponding entries in truncated matrix, $\hat{K}$.

entries on the main diagonal of matrix, $\hat{K}$, with the local contributions divided by the sum of the contributions of all subdomains. For the two degrees of freedom in Figure 6.8, we scale by

$$\frac{\alpha_{\Omega_1}}{\alpha_{\Omega_1} + \alpha_{\Omega_2}}, \frac{\alpha_{\Omega_2}}{\alpha_{\Omega_1} + \alpha_{\Omega_2}}, \tag{6.4}$$

and

$$\frac{\beta_{\Omega_1}}{\beta_{\Omega_1} + \beta_{\Omega_2}}, \frac{\beta_{\Omega_2}}{\beta_{\Omega_1} + \beta_{\Omega_2}}, \tag{6.5}$$

for degrees of freedom, α, and, β, respectively.

We combine the deflation of the real rigid bodies and the subdomains with the recursive deflation operator defined by Definition 5.4.1. We assume we have a composite material with γ real rigid bodies and we apply a domain decomposition, yielding D domains. We define the deflation vectors corresponding to the real rigid bodies as $Z_{\text{ag}} \in \mathbb{R}^{n\times(\gamma\times 6)}$, and the deflation vectors corresponding to the subdomains as above $Z_{\text{sd}} \in \mathbb{R}^{n\times(D\times 6)}$. We define the deflation operator for the real rigid bodies as,

$$P_{\text{ag}} = I - KZ_{\text{ag}}(Z_{\text{ag}}^{\mathrm{T}}KZ_{\text{ag}})^{-1}Z_{\text{ag}}^{\mathrm{T}}, \tag{6.6}$$

and the the deflation operator for the subdomains as,

$$P_{\text{sd}} = I - KZ_{\text{sd}}(Z_{\text{sd}}^{\mathrm{T}}KZ_{\text{sd}})^{-1}Z_{\text{sd}}^{\mathrm{T}}. \tag{6.7}$$

We combine the deflation operators, and, obtain,

$$PK = P_{\text{ag}}P_{\text{sd}}K = K - KZ(Z^{\mathrm{T}}KZ)^{-1}Z^{\mathrm{T}}K, \tag{6.8}$$

where, $Z = [Z_{\text{ag}}, Z_{\text{sd}}]$.

Remark. In some deflation strategies, one of the real rigid bodies might consist of the collection of *all* elements in one or more subdomains. In that case, the rigid body modes of that real rigid body are linearly dependent on the null space vectors of those subdomains. Thus, let $\mathbf{z}_i \in \mathbb{R}^{n\times 6}$ correspond to the rigid body modes of rigid body i which overlaps with the rigid bodies of subdomains, then,

$$\exists \lambda = \{\lambda_1, \ldots, \lambda_D\} \;: \mathbf{z}_i = \lambda_1 Z_{\text{sd}}^{\Omega_1} + \ldots + \lambda_D Z_{\text{sd}}^{\Omega_D}, \tag{6.9}$$

where, $Z_{\text{sd}}^{\Omega_i} \in \mathbb{R}^{n\times 6}$, corresponds to the null space vectors of subdomain, Ω_i. In this particular case we omit the rigid body modes of the real rigid body from the deflation space, hence, we obtain,

$$Z = [\mathbf{z}_1, \ldots, \mathbf{z}_{i-1}, \mathbf{z}_{i+1}, \ldots, \mathbf{z}_\gamma] \in \mathbb{R}^{n\times((\gamma-1)\times 6)}. \tag{6.10}$$

6.4 Parallel implementation of other solvers

6.4.1 Parallel PCG method

The PCG algorithm [34] is constructed from basic linear algebraic operations. As described in the previous section, only the matrix-vector operation and inner product require communication. All other linear algebraic operations (e.g. vector scaling and addition) can be done locally; i.e., there is no communication with other subdomains.

The parallelization of the preconditioner is usually much more involved and depends on the type of preconditioner. In this research, we consider diagonal scaling, IC factorization, and SA-AMG. We note that diagonal scaling is, in fact, a matrix-vector operation and hence, can be parallelized with the parallel matrix-vector operation defined in the previous section. In this thesis, we apply the following simple parallelization strategy. We compute the IC factorization only for the local stiffness matrices and ignore all connectivity between the subdomains. With this approach, we might lose any global error reduction properties of the preconditioner, but we gain performance as no communication between the subdomains is needed to compute the decomposition. In Section 4.1.2 we argue that we may lose accuracy if the local stiffness matrices are (nearly) singular; hence, the diagonal entries of the local stiffness matrices are communicated, and, thus, 'synchronized'. We elaborate on the parallel implementation of SA-AMG in the next section.

6.4.2 Parallel SA-AMG

In recent years, two general-purpose parallel algebraic multigrid codes have been developed, alongside a number of other codes aimed at specific applications. One of these codes, BoomerAMG [39] (included in the Hypre package [31]), focuses on classical (Ruge-Stüben) AMG algorithms and their variants, while the other, ML [33] (included in the Trilinos project [40]), focuses on the smoothed aggregation setting. In our experiments below, we make use of ML and Trilinos for the parallel implementation of smoothed aggregation.

There have been a number of studies on the performance of parallel AMG codes for solid mechanics operations. Initial two-dimensional results were reported in [89], based on an AMG treatment that first coarsens appropriately along boundaries shared between processors and then treats the processor interiors. Scalability studies for a simplified AMG approach, based on maximal independent set coarsening of nodes, remeshing the coarse node set, and using geometric grid-transfer operators, for both linear elasticity and nonlinear elastic and plastic solid mechanics are detailed in [2]. This method was compared with smoothed and unsmoothed aggregation in [3], where it was found that the simplified approach was less robust than the aggregation approaches, and that smoothed aggregation was most robust and typically not much more expensive than the other two approaches. A comparison of Ruge-Stüben AMG, smoothed aggregation, and a generalized smoothed aggregation approach (using information from local eigensolves to complement the restricted rigid-body modes) was performed in [17], where smoothed aggregation was shown outperform Ruge-Stüben AMG for most cases. The generalized form offers even greater robustness, but relies on an expensive preprocessing step. One important issue in these studies is the choice of parallel smoother; this was studied in depth in [1], comparing parallel hybrid Gauss-Seidel orderings with polynomial (Chebyshev) smoothers and concluding that polynomial smoothers offer many advantages.

7

Numerical examples

In Chapter 5 and 6, we discussed the theory of rigid body mode and subdomain deflation, and, the parallel implementation. In this chapter we illustrate the performance and robustness of DPCG by numerical experiments.

All the numerical experiments in this chapter are based on three different cases which we describe in Section 7.1. We discuss the hardware and software configurations in Section 7.2.

In Section 7.3 we look into the robustness of DPCG for varying orders of stiffness, in Section 7.4 we show how subdomain deflation can improve the convergence of DPCG, and in Section 7.5 we compare DPCG to the state-of-the-art solution methods that we discussed in Chapter 4.

7.1 Description of cases

All three cases in this chapter concern the analysis of asphaltic materials subjected to an external force. The simulation of the material behavior involves the evaluation of the linearized virtual work equation given in Chapter 3. The asphaltic materials that we consider in this section are composite materials that consist of aggregates, bitumen, and air voids. Although in practice, bitumen is modeled as an elasto-visco-plastic material, in this chapter we only consider the non-linear Neo-Hookean hyperelastic material model. In the next chapter we show that the performance of the DPCG method does not depend on specific material models. The elastic components are the dominating contributions to the stiffness matrix. Large variations due to hardening as well as softening effects in the materials may lead to a significant (permanent) change in the 'stiffness' of the material. Keep in mind that the 'stiffness' of the material constitutes the ordering of the sets of elements that determine the deflation vectors, hence, if the stiffness changes due to these effects, a new ordering may apply and, thus,

aggregate	bitumen	air voids
69000	5000	100

Table 7.1: Young's moduli for different materials

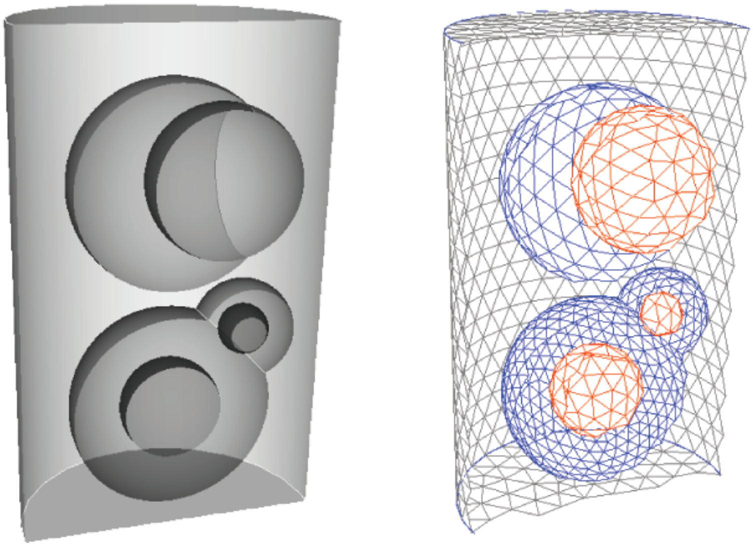
we need to identify the new rigid bodies and hence construct the corresponding set of deflation vectors. However, during an arbitrary iteration of the *modified* Newton-Raphson method, the Jacobian is fixed and hardening as well as softening effects can not influence the Krylov iteration process.

We use the same set of material parameters for the hyperelastic material model in all numerical experiments. The corresponding stiffness coefficients (Young's Moduli) are given in Table 7.1.

the artificial case: small artificial case, cylinder containing aggregates and bitumen

The case given in Figure 7.1 is a cylinder of soft material (air voids) containing three aggregates embedded in a layer of bitumen. The aggregates and surrounding bitumen are represented by spheres. The corresponding mesh holds 23602 elements and yields 12314 degrees of freedom. The number of deflation vectors for the experiments of the artificial case is 36. All vectors are sparse and Z is of full rank.

We study this academic example to illustrate the various effects of the stiffness of the rigid bodies in the composite material on the convergence of PCG without having to take into account any numerical side effects due to badly shaped elements that may come from the conversion of a CT-scan to a finite element mesh.



(a) (b)

Figure 7.1: the artificial case: FE mesh and schematic representation of cylinder containing three aggregates represented by spheres.

the asphalt core & cube of asphalt case: FE mesh from real core of asphalt concrete

The meshes of the asphalt core and the cube of asphalt case, given by Figure 7.2 and 7.3 are derived from real-life samples of asphaltic material obtained by CT scan. Both experiments involve different mesh sizes, yielding approximately 2.3×10^5 and 3×10^6 degrees of freedom, respectively.

The number of deflation vectors for the experiments of the asphalt core case and (iii) are 162 and 342, respectively. All vectors are sparse and Z is of full rank. As described in Section 6.2, the parallel implementation involves the distribution of the degrees of freedom, thus vectors; hence, the rigid body modes of an arbitrary aggregation of materials may be spread over multiple domains. Apart from an increase in iterations due to the preconditioner and deviations in iterations due to the round-off errors induced by the domain decomposition, the parallel implementation of DPCG should yield the same number of iterations as the sequential implementation provided that we have the same deflation space. As a result, the number of iterations of the DPCG method for a given problem is invariant under an increasing number of subdomains, given the same number of deflation vectors.

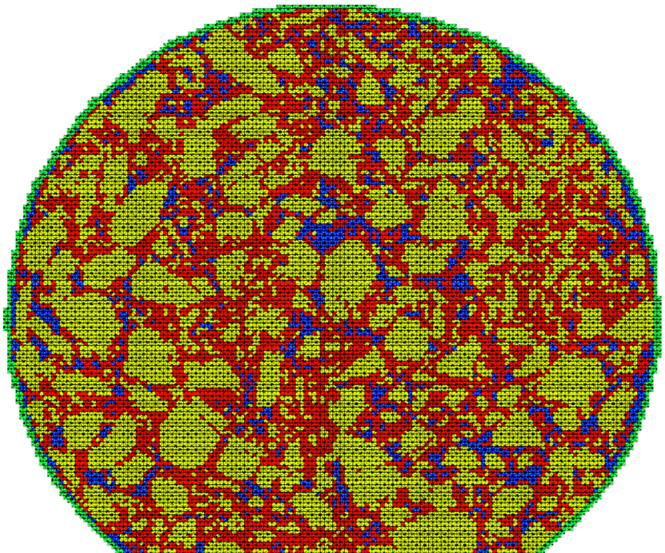


Figure 7.2: the asphalt core case: FE mesh representing core of asphaltic material containing aggregates (yellow), bitumen (red) and air voids (blue).

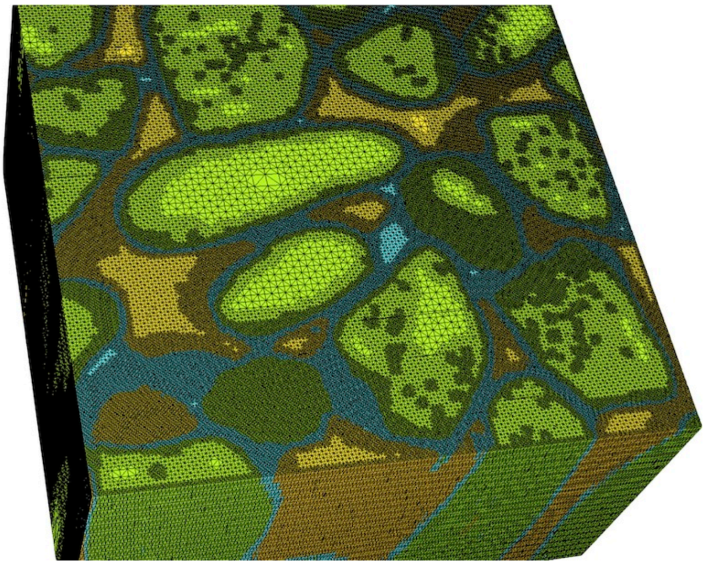


Figure 7.3: the cube of asphalt case: mesh representing cube of asphaltic material containing aggregates (light green), bitumen (dark green) and air voids (blue).

7.2 Description of hardware and software

DELL cluster / CAPA-3D: Dell workstations and CAPA-3D

We have implemented PCG and DPCG into the parallel FE software package CAPA-3D [19], which is programmed in Fortran90 and compiled with the Intel Fortran Compiler. Supporting libraries are the optimized BLAS and LAPACK libraries that are contained in the Intel MKL library. All experiments with CAPA-3D were performed on a cluster of 8 Dell workstations. Each cluster node has 2 Intel Xeon E5450 processors (8 CPUs) running at 3.00GHz, 16GB DDR2 memory, and, are connected by Infiniband. The two Intel Xeon processors have only two channels to access memory, hence, maximum parallelism is gained at 16 concurrent threads running in parallel, divided linearly over the 8 nodes. If the number of concurrent threads exceeds 16, threads that share an Intel Xeon processor compete for direct memory access, thus, slow down other threads and negatively affect the parallel speed-up.

We use this configuration for the domain decomposition by ParMETIS [52], and, generation of the stiffness matrices, force vectors, and deflation vectors by CAPA-3D and qualitative research on the robustness of the DPCG method.

TUFTS cluster / Trilinos: HPC cluster and Trilinos

We have implemented DPCG in the parallel software package Trilinos [40], which is programmed in C^{++} and compiled with the GNU C compiler. All experiments with Trilinos have been performed on the Tufts Linux Research Cluster which is comprised of 103 IBM Linux systems (compute nodes) interconnected via a 10Gig network. Each cluster node has 8 or 12 cores with 2.8Ghz+ Intel Xeon CPUs and 16, 24,32,48 or 96 gigabytes of memory. The 1000+ compute cores have a capacity of about 10 TeraFlops. Each user may access up to a maximum of 256 CPUs.

We use this configuration for quantitative research on the performance of the DPCG method, and compare to the state-of-the-art solvers that are contained in Trilinos, such as SA-AMG and SuperLU.

7.3 Rigid body mode deflation

In Chapter 5 we discussed the theory of rigid body mode deflation. In this section we illustrate the effect of the discontinuities on the convergence of PCG, by comparing results for 4 sets of parameters. The first set (i.) from Table 7.2 contains realistic material parameters. The other sets do not have a direct physical meaning to asphaltic materials, but are used for illustration of the performance of deflation. We have conducted the experiments with two different preconditioners, diagonal scaling and Incomplete Cholesky with a drop tolerance of 10^{-3}. This drop tolerance was determined after performing several tests with ILUPACK [12] and represents a good compromise in terms of memory usage (lower drop tolerance demands more memory) and the speed of the back solve (lower tolerance yields a slower back solve) against the performance of the preconditioner in terms of reduction in number of iterations of DPCG. All experiments were performed with Conguration (i).

the artificial case

We compare DPCG and PCG in combination with diagonal scaling. The case involves a mixture of materials that is subjected to an external force applied to the upper boundary of the volume. Zero-displacement boundary conditions are imposed on the base of the volume, this is, homogenous Dirichlet boundary conditions to all degrees of freedom in the x, z-plane for $y = 0$. We note that the case resembles the uniaxial compression test, which is a standard laboratory test. We observe the convergence behavior of DPCG and PCG for variations in the Young's moduli of the bitumen and aggregates as given in Table 7.2. We compare a standard choice for the values of parameter E [28] with increased stiffness of the aggregates, and decreasing stiffness for the bitumen and air voids.

Figure 7.4 shows the convergence of PCG and DPCG for parameter sets (i.) to (iv.). Clearly the convergence of the solution with PCG is slow and highly oscillating. PCG compared to DPCG is also slower in terms of wall clock time. But due to the small problem size, this is more a qualitative example rather than quantitative. We observe in the plots of sets (i.) to (iii.) that the value of the material stiffness for the aggregates and bitumen does not influence the number of iterations of DPCG.

	aggregate	bitumen	air voids
i.	69000	5000	100
ii.	**690000**	5000	100
iii.	69000	**500**	100
iv.	69000	5000	$\mathbf{10^{-2}}$

Table 7.2: Young's modulus for different materials

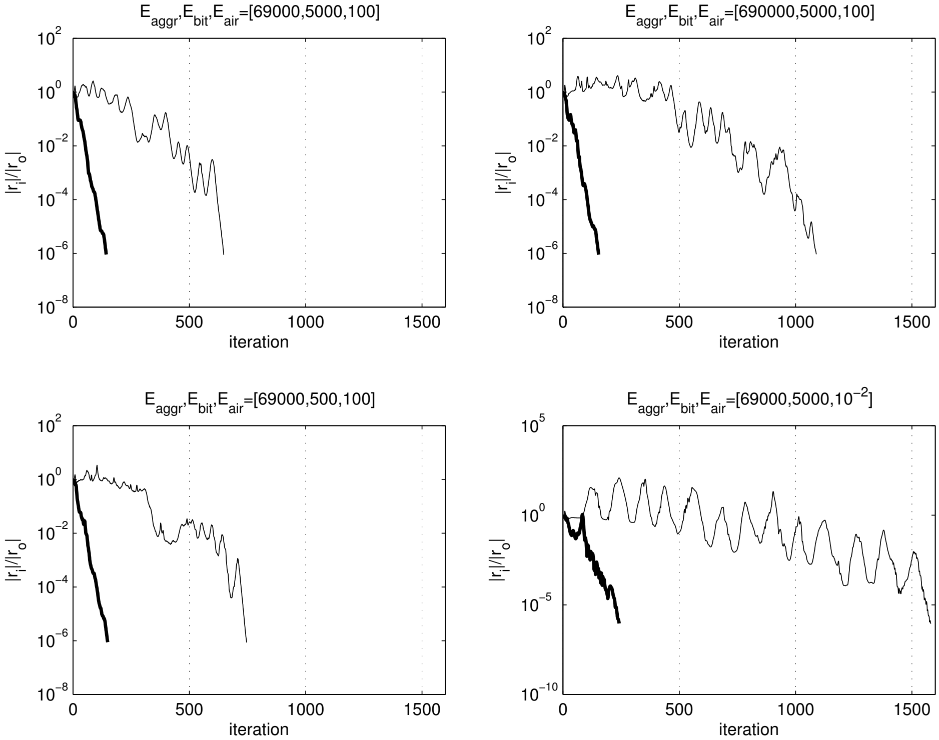


Figure 7.4: the artificial case: Convergence of PCG and DPCG (bold line) for cylinder containing three aggregates

This is what we expected. The stiffness matrices corresponding to the aggregates and bitumen have been decoupled. We observe that the effective condition number is bounded by the smallest eigenvalue of the least stiff material, the air voids. This can also be observed in the plot of set (iv). The number of iterations of DPCG increases from 150 towards 242 for air. As the value of the material stiffness of the air voids

	PCG		DPCG	
	iter	cpu (s)	iter	cpu(s)
i.	648	0.288	143	0.204
ii.	1089	0.477	154	0.175
iii.	746	0.328	149	0.172
iv.	1581	0.677	242	0.276

Table 7.3: the artificial case: wall clock time(s) PCG and DPCG

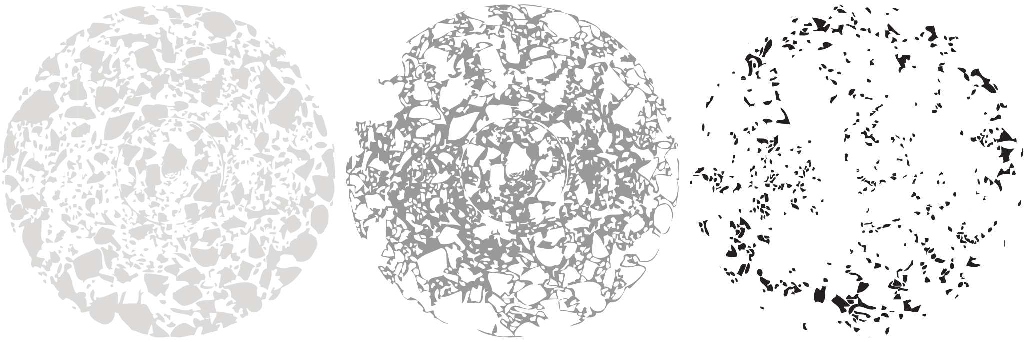

(a) (b) (c)

Figure 7.5: the asphalt core case: deflation strategy, identify sets of elements corresponding to material: (a) aggregates, (b) bitumen and (c) air voids.

changes from 100 to 10^{-2}, the effective condition number increases as well as the number of iterations of both DPCG and PCG. However, this is not surprising as the smallest (non-zero) eigenvalue is determined by the least stiff material, due to the decoupling of the stiffness matrices corresponding to the different materials. When the stiffness decreases, the smallest eigenvalue will become smaller and subsequently the condition number increases. We do not consider this as a shortcoming of the deflation method as it can and must be solved by applying the right preconditioner.

the asphalt core case

We compare DPCG and PCG in combination with incomplete Cholesky with drop tolerance 10^{-3}. The case involves a mixture of materials that is subject to an external force applied to the upper boundary of the volume. Zero-displacement boundary conditions are imposed on three sides of the volume, this is homogenous Dirichlet boundary conditions to all degrees of freedom in the x, z-, x, y- and y, z- planes for $y = 0$, $z = 0$ and $x = 0$ respectively. We observe the convergence behavior of DPCG and PCG for variations in the E modulus of the bitumen and aggregates as given in Table 7.1. We compare a standard choice of parameters [28] with increased stiffness of the aggregates, and decreasing stiffness for the bitumen.

There is a difference between the artificial cylinder and the sample of real asphaltic material. Where it was possible to decouple all materials in case of the cylinder, for an FE mesh obtained from a CT scan this is much more involved. We can see from Figure 7.5 (b) and (c) that there exist many small bodies of material. Each body is represented in the deflation space by six rigid body modes. However, due to overlap, many of these sparse vectors will become zero, implying a singular deflation matrix. Moreover, because of the large number of small bodies and thus deflation vectors, it

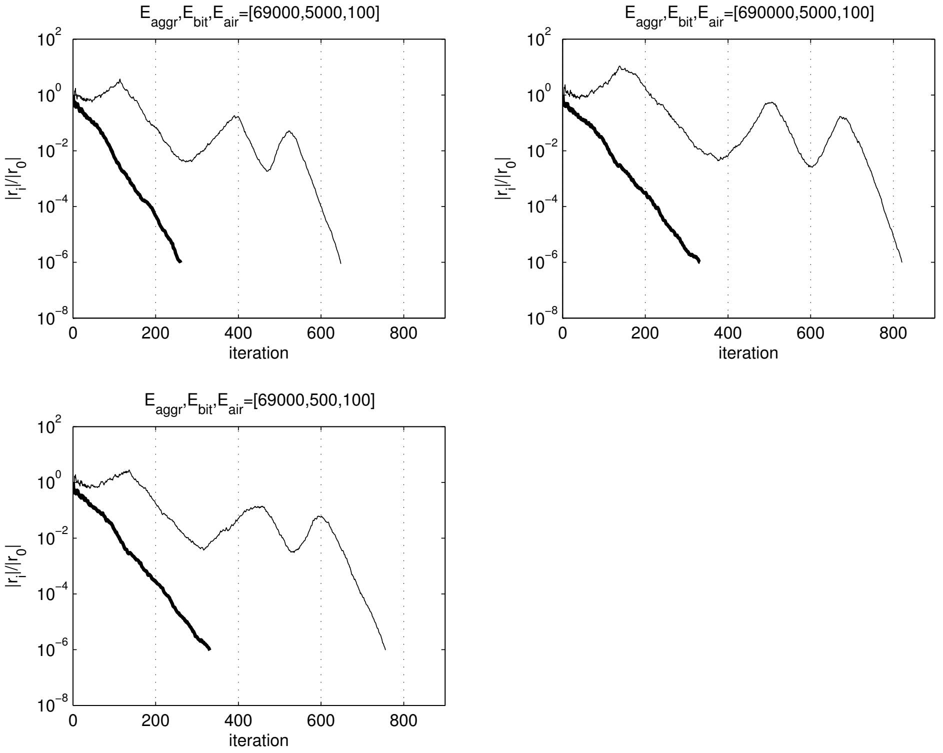


Figure 7.6: the asphalt core case: Convergence of PCG and DPCG for a real slice of asphaltic material

would be more favorable in terms of overhead to collapse these bodies into one entity. Therefore we have used an adapted version of the deflation strategy of Section 5.4.2. By combining sets of elements of different materials, we still have a decoupling when we keep in mind the decreasing order of stiffness for the construction of the splitting of Theorem 5.2.1. We note that we lose some rigid body modes, and hence a worse bound of the condition number for PK but we gain performance because of a large

	PCG		DPCG	
	iter	cpu (s)	iter	cpu(s)
i.	648	13.18	261	7.26
ii.	821	17.48	332	9.31
iii.	756	15.21	331	8.89

Table 7.4: the asphalt core case: wall clock time(s) PCG and DPCG

reduction in deflation vectors and avoid singularity of the deflation matrix. Also we have omitted set (iv.) from this test because the FE software would not run this value of air voids due to collapsing elements (negative Jacobian).

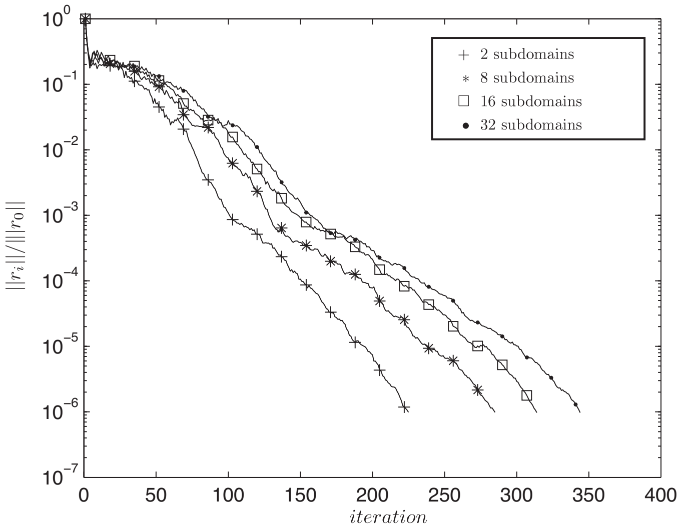
We observe in Figure 7.5 that PCG has a strongly oscillating curve of convergence and DPCG has nearly a straight line. Clearly the unfavorable eigenvalues have been removed by deflation. However, the system is not decoupled completely because the number of iterations is not invariant for different sets of material parameters. But the number of iterations of DPCG is much smaller compared to PCG. The performance of DPCG in terms of wall clock time is also better compared to PCG.

7.4 Subdomain deflation

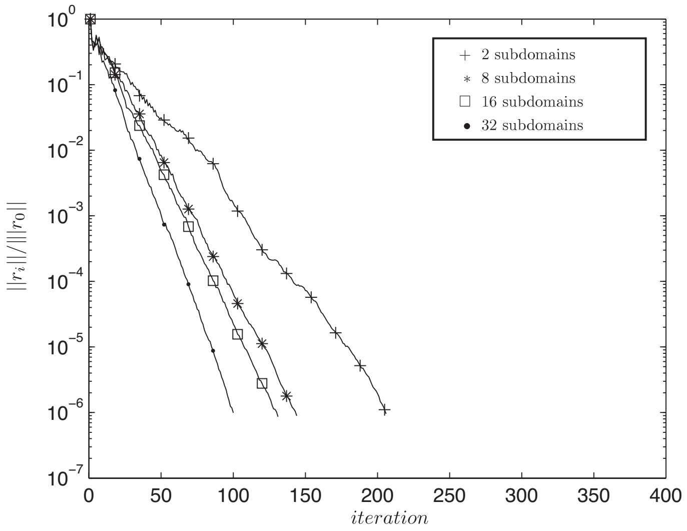
In Section 6.3, we introduced subdomain deflation as an additional tool to the deflation of the rigid body modes of the different bodies in a composite material, to decouple the regions in the global stiffness matrix corresponding to the subdomains and, thus, eliminate the need for a strong (expensive) global preconditioner. We show the potential of subdomain deflation by comparing the convergence behavior of DPCG with and without subdomain deflation for 2, 8, 16, and 32 subdomains. We compare the number iterations for different number of subdomains for the asphalt core case and study the performance of DPCG for the cube of asphalt case. The wall clock time and memory usage of DPCG are only provided for the cube of asphalt case as the variations in these numbers for different subdomains are not significant for the asphalt core case. The DPCG method is preconditioned by Incomplete Cholesky with drop tolerance 10^{-3}. Both cases involve a mixture of materials that is subject to an external force applied to the upper boundary of the volume. Zero-displacement boundary conditions are imposed on three sides of the volume, this is, homogenous Dirichlet boundary conditions to all degrees of freedom in the x, z-, x, y- and y, z- planes for $y = 0$, $z = 0$ and $x = 0$ respectively. All experiments were performed with DELL cluster / CAPA-3D.

the asphalt core case

In Figure 7.7(a), 7.7(b) we present the L_2 norms of the residuals of the DPCG method respectively without, and with subdomain deflation for 2, 8, 16, and, 32 subdomains. Clearly, for an increasing number of subdomains the number of DPCG iterations increases when no subdomain deflation is applied. This is what we expect from Incomplete Cholesky with a fixed drop tolerance. The local stiffness matrices correspond to smaller domains, and hence, the 'local' preconditioner has less global error reduction capabilities. We observe that by decoupling of the regions in the global stiffness matrix corresponding to the subdomains, we can apply these 'local' preconditioners



(a)



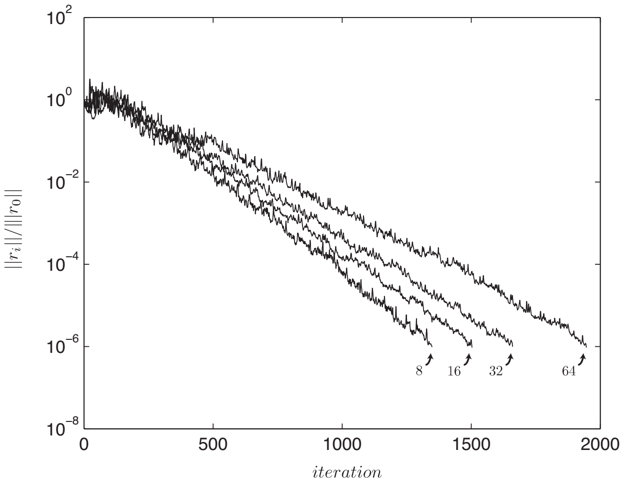
(b)

Figure 7.7: the asphalt core case: L_2 norms of the residuals for DPCG without 7.7(a), and, with 7.7(b) subdomain deflation for 2, 8, 16, and, 32 subdomains.

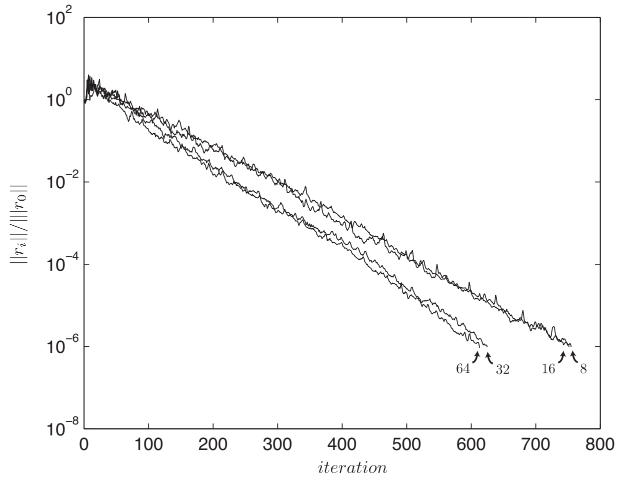
and still preserve the robustness of the DPCG method at the cost of extra (subdomain) deflation vectors.

the cube of asphalt case

In this example we compare the performance of DPCG with, and, without subdomain deflation and analyze the additional work induced by subdomain deflation. In Figure 7.8(a), 7.8(b) we present the L_2 norms of the residuals of the error of the DPCG method without, and, with subdomain deflation respectively for 8, 16, 32, and 64 subdomains. We observe the same convergence behavior as for the previous example, if we do not



(a)



(b)

Figure 7.8: the cube of asphalt case: L_2 norms of the residuals for DPCG without 7.8(a), and, with 7.8(b) subdomain deflation for 8, 16, 32, and, 64 subdomains.

use subdomain deflation the number of DPCG iterations increases for an increasing number of subdomains. On the other hand, the number of DPCG iterations decreases for an increasing number of subdomains if subdomain deflation is applied.

The iteration count and wall clock time (s) for DPCG with, and without subdomain deflation are presented in Table 7.5. We observe that the additional amount of work due to the subdomain deflation vectors does not significantly influence the performance of the DPCG method. Bring in mind that we have true parallelism for DELL cluster / CAPA-3D only up to 16 domains. We observe linear parallel speed-up for DPCG with subdomain deflation going from 8 to 16 domains, while the number of iterations remains constant. Moreover, we gain a factor of 1.5, 1.87, 2.38, and 2.70 in time when using DPCG with subdomain deflation compared to DPCG without subdomain deflation for an increasing number of subdomains. Thus, we conclude that we gain performance of DPCG for an increasing number of subdomains, although the dimension of the deflation space increases due to an increasing number of subdomain deflation vectors.

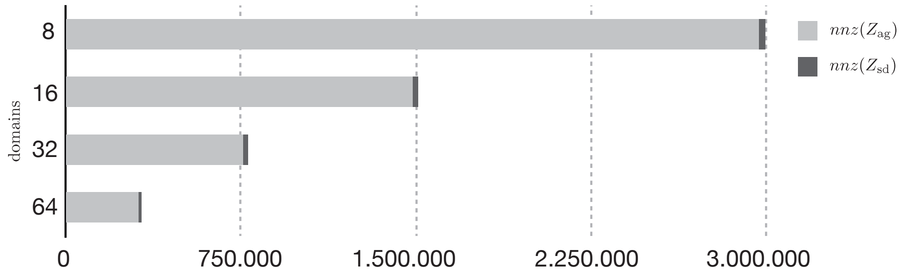
In Figure 7.9(a), 7.9(b) we present the number of nonzeros in the deflation matrix, Z, and the relative increase of nonzeros in matrix, KZ, for an increasing number of subdomains, hence, subdomain deflation vectors.

From Figure 7.9(a) we conclude that the amount of additional memory space for the storage of the subdomain vectors is negligible compared to the storage of the 'original' rigid body mode deflation vectors. However, the additional memory space for the storage of the subdomain deflation vectors will be much larger compared to the storage of the 'original' rigid body mode deflation vectors when the number of domains equals or exceeds the total number of rigid bodies in the composite material, but this does not apply to the experiments considered in this research.

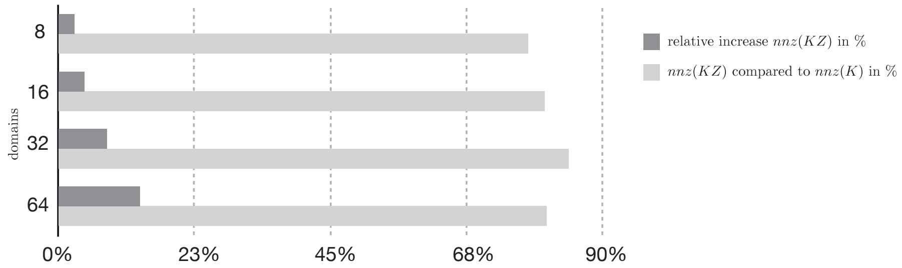
From Figure 7.9(b) we conclude that the number of *additional* nonzeros in matrix, KZ, due to the addition of subdomain deflation vectors, increases almost linearly with the number of subdomains. The additional nonzeros entries in matrix, KZ, naturally lead to a proportionally large number of additional FLOPS for a multiplication with this matrix. However, as the number of parallel processes increases equally, the total

	DPCG	(no subdomain defl)	DPCG	(subdomain defl)
	iter	cpu (s)	iter	cpu(s)
8	1348	781	755	530
16	1503	440	752	235
32	1661	380	625	160
64	1946	421	613	156

Table 7.5: the cube of asphalt case: wall clock time(s) DPCG for subdomain deflation



(a)



(b)

Figure 7.9: the cube of asphalt case: number of nonzeros in matrix, Z, and the relative increase of nonzeros in matrix, KZ, using subdomain deflation for 8, 16, 32, and, 64 subdomains.

FLOP rate should scale linearly with the number of subdomains, and, hence, balance the added amount of work due to the subdomain deflation vectors, but at the expense of zero speed-up. From Table 7.5 we learn that this effect is negligible when the number of subdomains is much smaller than the number of rigid body modes contained in the composite material.

7.5 Performance DPCG and comparison to other state-of-the-art methods

In Chapter 4 we discussed the existing solution methods for solving the linear systems that come from the linearized virtual work equation of Chapter 3. In this section we compare the DPCG method with other state-of-the-art solution methods. We consider direct solution methods and SA-AMG which are implemented in MUMPS, and Trilinos[1], respectively.

Although we do not consider direct solvers as the method of choice for coupled PDEs that are defined on very large three dimensional meshes, we compare the DPCG method implemented in CAPA-3D to the direct factorization method MUMPS for a small number of domains. All experiments with CAPA-3D were performed with DELL cluster / CAPA-3D.

The Trilinos package provides an interface to many state-of-the-art linear solvers. In this research we consider SA-AMG, with different coarsening schemes as solver as well as preconditioner to DPCG. All experiments with Trilinos were performed with TUFTS cluster / Trilinos.

In this section we consider the asphalt core case and (iii). Zero-displacement boundary conditions are imposed on three sides of the volume; that is, homogenous Dirichlet boundary conditions are given for all degrees of freedom in the x, z-, x, y- and y, z- planes for $y = 0$, $z = 0$, and $x = 0$, respectively. In the experiments, we make use of the same set of material parameters given in Table 7.1.

7.5.1 CAPA-3D: DPCG and MUMPS

the cube of asphalt case

We solve the stiffness matrix with MUMPS, and DPCG preconditioned by Incomplete Cholesky with a drop tolerance of 10^{-3} and rigid body modes and subdomain deflation. The resulting number of deflation vectors for D subdomains is, $342 + (D \times 6)$. The wall clock time as well as the number of iterations for both solvers and all domain decompositions are given in Table 7.6. The convergence curve of DPCG for the cube of asphalt case is given in Figure 7.8(b).

We observe in Table 7.6 that MUMPS is outperforming DPCG for all subdomains. Although under optimal parallel conditions, which is one dedicated memory bus for each CPU, the timings are relatively close (roughly 20 % difference in wall clock time, in favor of MUMPS). The analysis of the memory usage of the threads on the cluster

---

[1] Trilinos is implemented with the object-orientated programming language C^{++}, and, thus, has considerably more overhead in terms of memory and CPU compared to libraries compiled with FORTRAN. Therefore we do not compare the results coming from different software packages.

	DPCG		MUMPS
	iter	cpu (s)	cpu (s)
8	755	530	301
16	752	235	198
32	625	160	131
64	613	156	105

Table 7.6: the cube of asphalt case: wall clock time(s) DPCG and MUMPS for 8, 16, 32, and, 64 subdomains

of DELL cluster / CAPA-3D tells that the required amount of memory for MUMPS running with 16 subdomains is 87GB (divided over 16 CPUs) and he required amount of memory for DPCG running with 16 subdomains is 15GB (divided over 16 CPUs). This shows the limitations of factorization methods for large systems of PDEs. Given the same resources, with DPCG we can solve problems that take 6 times the memory of the current test case.

Although MUMPS uses significantly more memory, in Table 7.6 we observe a more favorable speed-up for non-optimal parallel conditions. We conclude that MUMPS is implemented more efficiently by utilizing near-CPU cache memories and special memory queueing strategies [7]. Moreover, with dedicated hardware (vector processors, GPU computing) and cache memory programming, the DPCG algorithm can be further optimized. In Chapter 10 we discuss future research on more efficient implementation of the DPCG algorithm.

In Figure 7.10 we present the cumulative timing of all different stages of the DPCG method for 8, 16, 32, and, 64 subdomains. The set-up time and the run-time of the

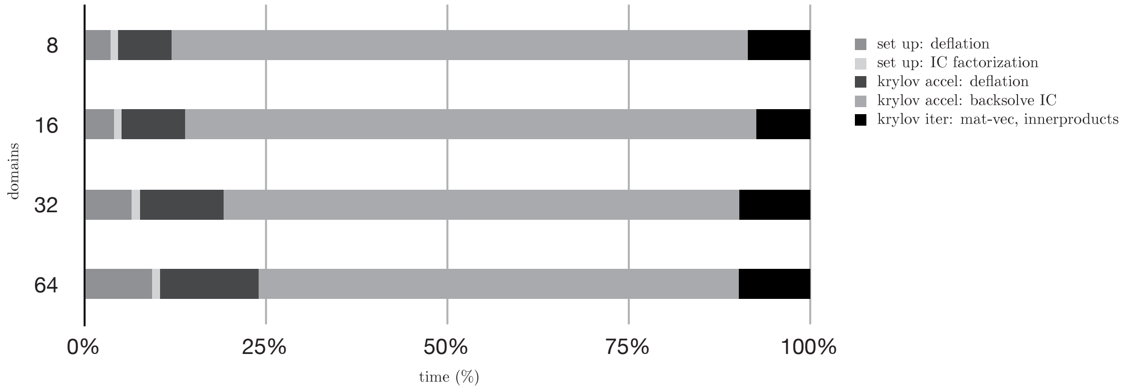


Figure 7.10: the cube of asphalt case: Cumulative timing all stages of DPCG (IC preconditioner) for 8, 16, 32 and 64 subdomains.

deflation operator and the preconditioner are considered, as well as the timing of the 'general' matrix-vector multiplications and inner products. We observe that the relative set-up time of the preconditioner does not change under an increasing number of subdomains, but the set-up time, as well as the run-time, of the deflation operator do. This is expected given the increase in the number of non-zero entries in Z, and, KZ, as presented in Figure 7.9(b). Although we may explain the deterioration of the parallel speed-up of DPCG in Table 7.6 by these results, as argued above, an enhanced implementation of the DPCG algorithm, which would address memory more efficiently is likely to give more favorable results.

7.5.2 Trilinos: DPCG and ML (SA-AMG)

The aim of the experiments in this section is to compare the performance and robustness of our deflation method and (optimized) SA-AMG for mechanical problems with heterogeneous material coefficients. In Section 5.2, we have argued that DPCG is a two-level approach, and that we need a preconditioner to treat both ends of the spectrum of the stiffness matrix. We have seen that SA-AMG is designed to solve homogeneous elastic equations in an optimal way. A natural choice for preconditioning of DPCG would be using a "black box" implementation of SA-AMG for the "decoupled" stiffness matrices. Hence, we compare PCG and DPCG preconditioned by SA-AMG as well as PCG preconditioned by SA-AMG / VBMetis which are included in the ML software. In the case of SA-AMG, we provide the minimal amount of information required and follow the default software options. In the case of SA-AMG / VBMetis, we explicitly provide complete information about the PDE structure and null space. In particular, we provide the degree-of-freedom to node map, including full details about eliminated degrees of freedom due to boundary conditions, and we give directly the rigid body modes of the entire solid body. In both cases, we make use of Chebyshev smoothers. We also include diagonal scaling as preconditioner to have a point of reference from which to compare all methods. The stopping criterion for all computations is $r_i < 10^{-6}$ where r_i is the residual vector at ith iteration.

For a fair comparison of PCG and DPCG, we have implemented both methods using the Trilinos software of TUFTS cluster / Trilinos. Due to the complexity of the meshes and limitations of our meshing software, we only take into consideration the strong scaling effect of the solvers. In the asphalt core case, we compare results for 4, 16 and 64 subdomains, where each subdomain is mapped onto a computing core. In the cube of asphalt case we compare results for 4, 8 and 64 subdomains, because of memory limitations of the SA-AMG / VBMetis solver.

We have included the run time of the decomposition of the stiffness matrix using SuperLU 2.5 (distributed memory) [56], hence, we can compare the performance of a direct solver with the performance of DPCG on the hardware of TUFTS cluster / Trilinos.

The wall clock times as well as the number of iterations of all solvers for all domain decompositions are given in Figure 7.11. Running with 64 subdomains, the DPCG solver with diagonal scaling is clearly the fastest method, requiring 17 seconds and 1663 iterations. The number of iterations for PCG and DPCG for both diagonal scaling and SA-AMG is essentially invariant with the number of subdomains, as expected. The strong scaling is clearly not optimal for this number of unknowns. For instance, the wall clock time for DPCG with diagonal scaling goes from 73 to 37 and down to 17 seconds, reducing by a factor of two when the number of subdomains increases by a factor of four. The costs of the deflation operator for diagonal scaling is almost equal to the costs of the matrix-vector products together with the inner products. The set up time of the deflation operator lies around 20% of the total cost and does not significantly increase when increasing the number of subdomains.

The DPCG method combined with the SA-AMG preconditioner has a good performance in terms of iterations, but shows poor parallel performance. Going from 16 subdomains to 64 subdomains gains very little in terms of speed-up. Also, the SA-AMG preconditioner has three times the set up cost and ten times the operational cost compared to the deflation operator.

The PCG method combined with the SA-AMG / VBMetis preconditioner performs worse than the DPCG method with SA-AMG in terms of iterations as well as wall clock time. Although the ratio between set up time and cost per iteration is almost equal for both methods, the overall cost of the SA-AMG / VBMetis preconditioner is much higher. Again, due to the small problem size, there is no benefit from the parallel implementation.

The L_2 norms of the residuals of Experiment 1 are given in Figure 7.12. We observe that without deflation, i.e. PCG preconditioned by SA-AMG or SA-AMG / VBMetis, not all unfavorable eigenvalues have been removed from the spectrum of $M^{-1}K$. This can also be seen from Figure 7.13, where the 50 smallest Ritz values of $M^{-1}K$ and $M^{-1}PK$ are given. Clearly, the deflated system has no clusters of very small (approximated) eigenvalues whereas the non-deflated system, even though

	4 domains		16 domains		64 domains	
	iter	cpu (s)	iter	cpu (s)	iter	cpu (s)
PCG - diag	4644	109	4642	58	4643	16
DPCG - diag	1665	73	1665	37	1663	17
PCG - SA-AMG	635	150	631	85	637	58
DPCG - SA-AMG	214	66	214	36	217	26
PCG - SA-AMG/VB	298	120	317	51	324	59

Table 7.7: the asphalt core case: Wall clock time and number of iterations of (D)PCG

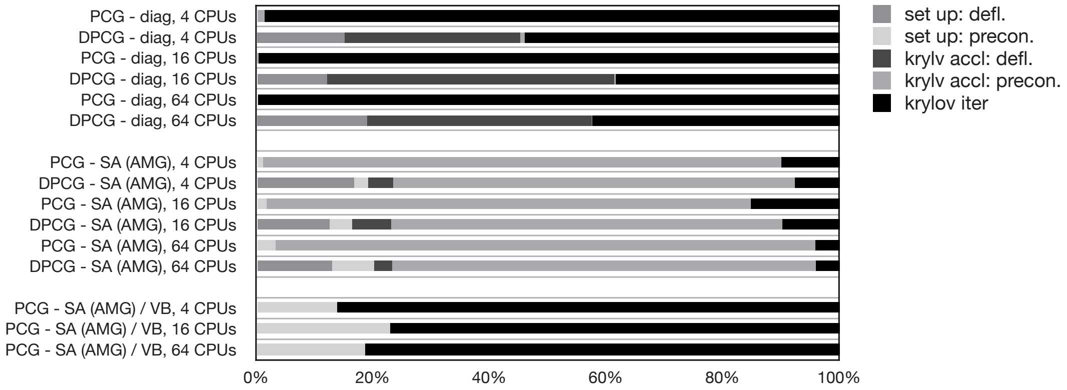


Figure 7.11: the asphalt core case: Cumulative timing all stages of PCG, DPCG for 4, 16, and, 64 subdomains.

preconditioned by SA-AMG / VBMetis, still contains some unfavorable eigenvalues.

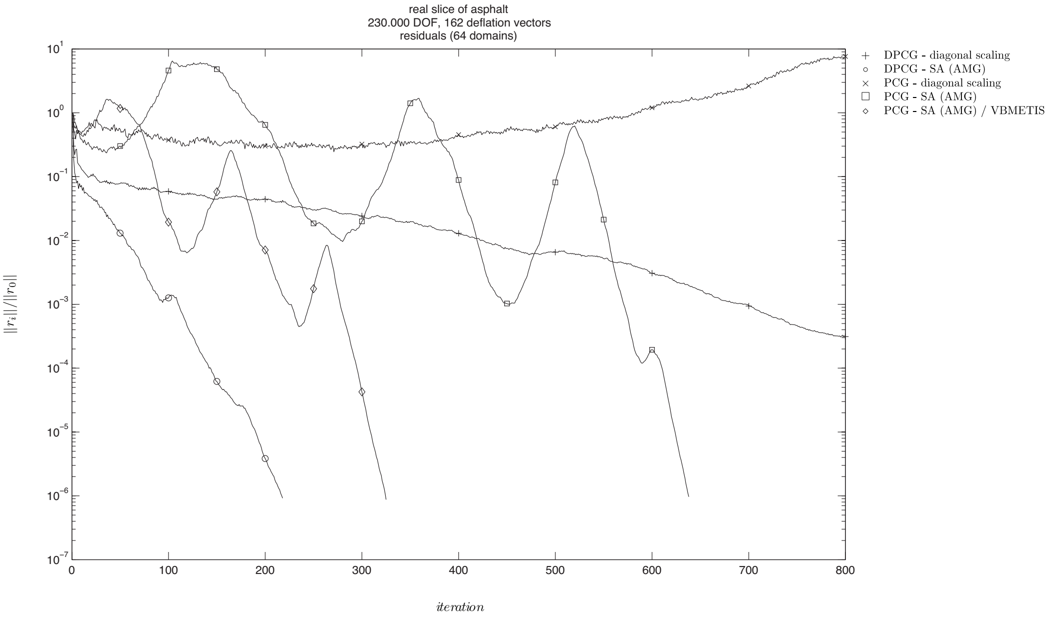


Figure 7.12: the asphalt core case: L_2 norms of the residuals

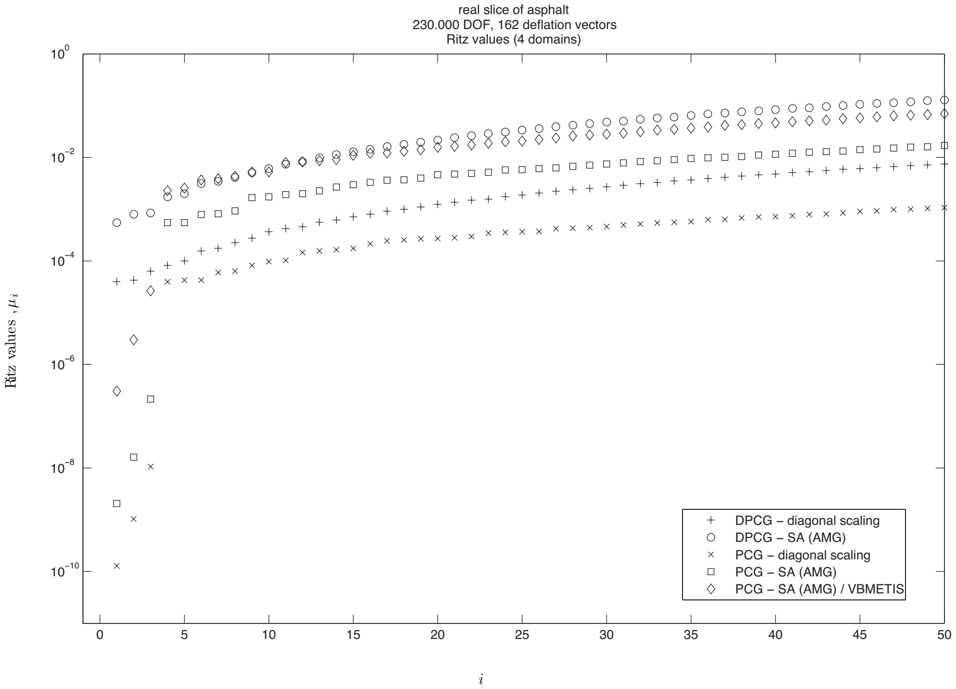


Figure 7.13: the asphalt core case: Ritz values derived from (D)PCG

the cube of asphalt case

The wall clock time as well as the number of iterations of all solvers for all domain decompositions are given in Figure 7.14. Again, we see expected performance from the iteration counts. For PCG, these improve from not converging (within 10000 iterations) with diagonal scaling, to roughly 2000 iterations with SA (AMG), to roughly 380 iterations with SA (AMG) / VBMetis. Here, the added costs of SA(AMG) / VBMetis are very notable, giving an out-of-memory error on 4 CPUs. Also as before, we see the immediate advantage of DPCG, leading to convergence in about 9000 iterations for diagonal scaling, and about 1200 iterations for SA (AMG). In this case, however, the added expense of SA (AMG) / VBMetis pays off, yielding about a 50% speedup over DPCG with diagonal scaling on 64 CPUs (and a greater speedup on 8 CPUs). We note that the SA (AMG) / VBMetis approach is far from a "black-box" preconditioner, and is available only as beta software within the ML package; as such, further improvement may be possible by refining this software and addressing memory issues that were encountered in obtaining these results.

Several interesting observations are possible about the parallel scaling. For the DPCG approaches, the relative cost of the deflation operator increases with the number of subdomains, becoming as expensive the total cost of matrix-vector and inner products; however, the setup time decreases, due to the parallel Cholesky decomposition of E. Excellent strong scaling is observed for DPCG with diagonal scaling, giving speedup of 1.8 and 8.0 for increases in the number of subdomains by factors of 2 and 8, respectively. Slightly less impressive speedup of DPCG preconditioned by SA (AMG) is observed, with factors of 1.87 and 6.54 for increases in the number of subdomains by factors of 2 and 8, respectively. This sub-optimal speed-up is due to the SA (AMG) preconditioner, which involves an extra set-up phase that scales poorly. The speedup of the SA (AMG) / VBMetis approach shows the poorest results, with a factor of only 2.45 when the number of subdomains increases by a factor of 8. This, again, is due to poor scaling of the SA(AMG) / VBMetis preconditioner, although better scaling might be observed with more degrees-of-freedom per subdomain on the 64 CPU scale.

	4 domains		8 domains		64 domains	
	iter	cpu (s)	iter	cpu (s)	iter	cpu (s)
PCG - diag	n.c.	-	n.c.	-	n.c.	-
DPCG - diag	9018	9883	9017	5456	9015	680
PCG - SA-AMG	2018	6687	2016	6906	1942	1123
DPCG - SA-AMG	1210	9450	1206	5043	1199	771
PCG - SA-AMG/VB	o.o.m.	-	376	1118	379	455

Table 7.8: the cube of asphalt case: Cumulative timing all stages of PCG, DPCG for 4, 8, and, 64 subdomains.

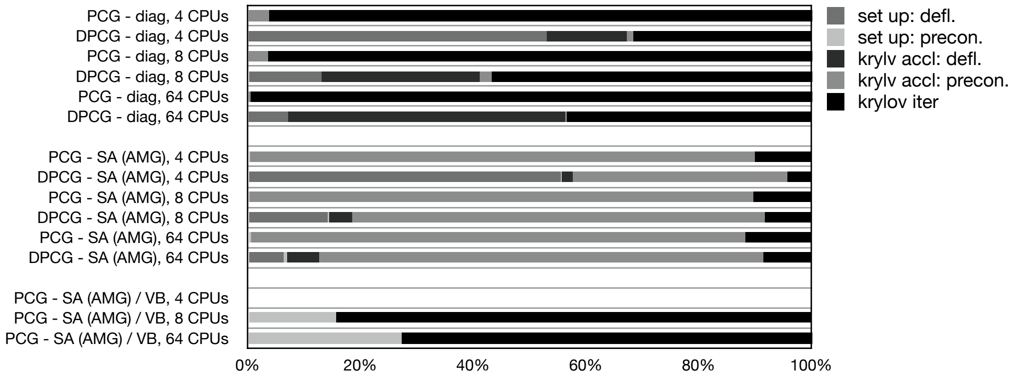


Figure 7.14: the cube of asphalt case: Cumulative timing all stages of PCG, DPCG for 4, 8, and, 64 subdomains.

The L_2 norms of the residuals of Experiment 2 are given in Figure 7.15, and the Ritz values of the preconditioned stiffness matrix derived form the (D)PCG iterations are given in Figure 7.16. We observe that the stiffness matrix preconditioned by SA-AMG / VBMetis yields a more favorable spectrum of eigenvalues compared to preconditioning with the deflation operator combined with SA-AMG. This can also be observed in Figure 7.15, as the residual curve of PCG preconditioned by SA-AMG / VBMetis is steeper than the curve of DPCG preconditioned by SA-AMG, which indicates that the eigenvalues of the spectrum of preconditioned K lie relatively closer to 1. However, compared to PCG preconditioned by diagonal scaling and SA-AMG, all unfavorable eigenvalues have been removed from the spectrum by DPCG and PCG preconditioned by SA-AMG / VBMetis.

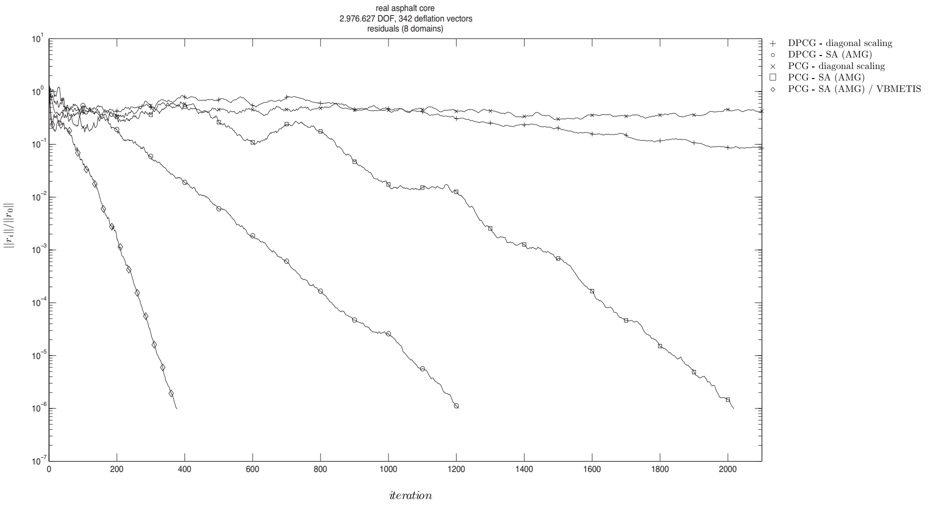


Figure 7.15: the cube of asphalt case: L_2 norms of the residuals

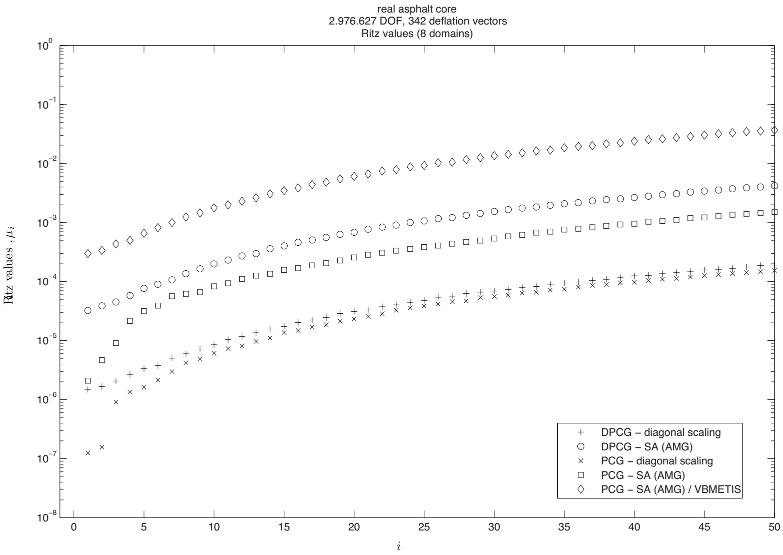


Figure 7.16: the cube of asphalt case: Ritz values derived from (D)PCG

SuperLU

The results for the direct solver for the asphalt core and the cube of asphalt case are given in Table 7.9. Using the parallel direct solver, wall-clock times are uniformly worse than those of DPCG for the smaller test problem; for the larger test problem, memory issues prevented convergence on all but the largest numbers of cores, where the wall-clock time was again substantially worse than all of the iterative approaches considered here.

	4 CPUs	8 CPUs	16 CPUs	64 CPUs
the asphalt core case	317	114	69	56
the cube of asphalt case	*o.o.m*	*o.o.m*	*o.o.m*	3979

Table 7.9: the asphalt core and the cube of asphalt case: wall-clock times (s) using SuperLU, distributed memory version 2.5

7.6 Concluding remarks

In this chapter we show that DPCG is a robust solution method for large jumps in the coefficients of the stiffness matrix. For both the academic test case as well as the real life engineering test cases we have seen that, for the right choice of deflation vectors, the number of iterations of DPCG does not depend on the number and magnitude of the jumps in the entries of the stiffness matrix.

We have seen that in the parallel implementation of (D)PCG the Incomplete Cholesky preconditioner loses the ability of global error reduction due to the loss of connectivity between the locally assembled stiffness matrices. Under an increasing number of subdomains and rigid body mode deflation vectors, the number of iterations decreases significantly. By combining rigid body mode deflation and subdomain deflation, we obtain a robust solver in which the number of iterations decreases with respect to an increasing number of subdomains, at the expense of a larger number of deflation vectors in the deflation space. However, the resulting matrix, KZ_{sd}, has only non-zeros at the interface nodes of the subdomain, hence, the added amount of work is relatively small and outweighs any loss in performance due to the weak local preconditioning.

Although DPCG performs slightly less compared to MUMPS at this stage, timings are relatively close and with further optimization we may increase the performance of DPCG. Moreover, we observed that MUMPS requires significantly more memory compared to (D)PCG and is therefore not a feasible method for very large meshes for the PDEs considered in this research. DPCG performs very well compared to

Smoothed Aggregation AMG (SA-AMG) which is provided with detailed information of the underlying mesh, distribution of degrees of freedom, and, null space of the linear operator. Due to the extensive analysis of the linear operator, the set-up time of SA-AMG is much higher than DPCG.

The performance of DPCG compared to state-of-the-art solution methods is promising. We conclude that DPCG with diagonal scaling as preconditioner is a very competitive method compared to the state-of-the-art methods considered in this chapter. The combination of DPCG with diagonal scaling offers an exceptionally low cost per iteration, giving much better wall-clock performance than PCG, even with the SA-AMG preconditioner. We suspect that DPCG and diagonal scaling combine so well because deflation and scaling are complementary operations, working on the lower and upper part of the spectrum respectively. DPCG with the SA-AMG preconditioner strongly improves the iteration counts over diagonal scaling but, for our experiments, does not improve the wall-clock time to solution. For our larger test problem, the optimized SA-AMG / VBMetis preconditioner does outperform the much simpler DPCG with diagonal scaling approach; however, this approach requires significantly more software development effort and, as such, may not be readily available for all simulation codes. Thus, we demonstrate that the DPCG approach has limited set-up time, is easy to parallelize, memory efficient, scalable, and robust and, as such, is an effective tool in large-scale simulation of the mechanics of composite materials.

8

Application to advanced material models

In Chapter 7, we showed that the DPCG method is a very robust and fast method for a variety of problems involving composite materials and the computational framework that we introduced in Chapter 2. In the numerical experiments of Chapter 7 we considered the non-linear Neo-Hookean hyperelastic material model. In this chapter we discuss the performance of DPCG for a highly non-linear elasto-visco-plastic material model that is used in many civil engineering applications.

The numerical experiments in this chapter are based on the artificial test case described in Section 7.1. In Section 8.1 we discuss the elasto-visco-plastic material model. In Section 8.2 we give the numerical results for the DPCG method applied to the artificial test case.

8.1 Generalized elasto-visco-plastic material model

The case described in Section 7.1 involves asphaltic material that consist of three components, bitumen, aggregates and air voids. The air voids and the aggregates are modeled as hyperelastic materials, the corresponding material parameters are given in Table 8.1, but in many civil engineering applications the bitumen is modeled as an elasto-visco-plastic material. The viscous and plastic effects are related to internal stress. For example, when the sun heats the asphaltic layer on the road, the stiffness of the bitumen changes due to viscous effects. Or, after endured loading we might observe permanent deformation of the asphalt due to plastic effects. The latter phenomenon is clearly visible at traffic lights, where many road slabs have deformed due to the loads induced by heavy traffic reducing their speed.

We consider a generalized elasto-visco-plastic material model, where plasticity is controlled by a Drucker-Prager control surface [23]. This model has two components, an elasto-plastic component and a visco-elastic component. The model is stress additive, the total stress on the model is the sum of the stress on the elasto-plastic and the visco-elastic component. In Chapter 2, we have discussed that the visco-elastic and the elasto-plastic material models have three phases, the linear elastic phase and the non-linear hardening and softening phases. During the linear elastic phase no permanent deformation occurs, the material returns to the original state when unloaded.

Elasto-plastic component	
E_p	5000
ν_p	0.13
c_{γ_0}	2.5
c_{γ_∞}	2.5
ϕ	0.01
δ	0.0
ξ_{lim}	10^4
δ_1	0.0
Visco-elastic component	
E_v	112
ν_v	0.3
η_d	100.0
η_v	10.0

Table 8.1: Parameters of elasto-plastic and visco-elastic components of elasto-visco-plastic material model.

During and after the hardening phase, the material will show permanent deformation and becomes more stiff. When the material reaches the softening phase, it will also show permanent deformation but the stiffness decreases until, eventually, the material cracks.

In Table 8.1 we present the parameters that constitute the material behavior of the bitumen in asphaltic materials. We observe that the Young's modulus of the elasto-plastic component is equal to the Young's modulus of the bitumen modeled as a hyperelastic material in Chapter 7. The c_{γ_0} and c_{γ_∞} are the cohesion parameters of the elasto-plastic component and determine the elastic yield surface, described in Chapter 2. Whenever the applied stress exceeds the elastic yield stress, plasticity is being built up. The remaining internal stress after the return mapping procedure constitutes the amount of permanent deformation and, thus, a change in stiffness. The parameters η_d and η_v are the deviatoric and volumetric viscosity respectively. These are the important parameters of the visco-elastic component.

In the numerical experiments of this chapter we focus on the permanent plastic deformation and the elastic and hardening phase. We determine the effect of these different phases on the performance of the DPCG method. The loading function for the numerical examples in this chapter is presented in Figure 8.1. The function represents a creep test. On the x-axis we have the load steps, on the y-axis we have the normalized applied stress. We apply 150 load steps. In the first load step we apply the full load. We keep constant loading until load step 99. In load step

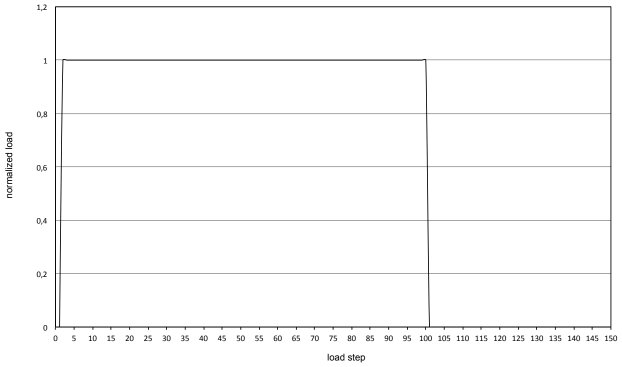


Figure 8.1: Loading function creep test, 150 load steps.

100 until 150 we apply zero load. These load steps are important in this simulation because it allows the material to return to its original configuration; hence, we can determine the permanent deformation due to the viscous and plastic effects.

8.2 Numerical experiments

In this section we consider the performance of DPCG for the two phases of the generalized elasto-visco-plastic material model provided in the previous section. We consider the artificial test case which is described in Section 7.1. Zero-displacement boundary conditions are imposed on three sides of the volume; that is, homogenous Dirichlet boundary conditions are given for all degrees of freedom in the x, z-, x, y- and y, z- planes for $y = 0$, $z = 0$, and $x = 0$, respectively.

Figure 8.2 displays the total strain in the direction of the loading, the y-direction, in an element of bitumen during all 150 load steps. The plastic strain on the element in the y-direction is presented in Figure 8.3. The two phases are clearly visible. The plastic strain becomes non-zero after 36 load steps, plasticity builds up. When zero load is applied the material returns to the original configuration, the total strain decreases but the plastic strain remains constant. Slight curvature in the total strain is visible in the first 99 load steps. During the elastic phase, the first 36 load steps, the total strain shows a linear decrease in time. During the hardening phase, load steps 37 until 99, the change in total strain decreases and we observe (exponential) curvature.

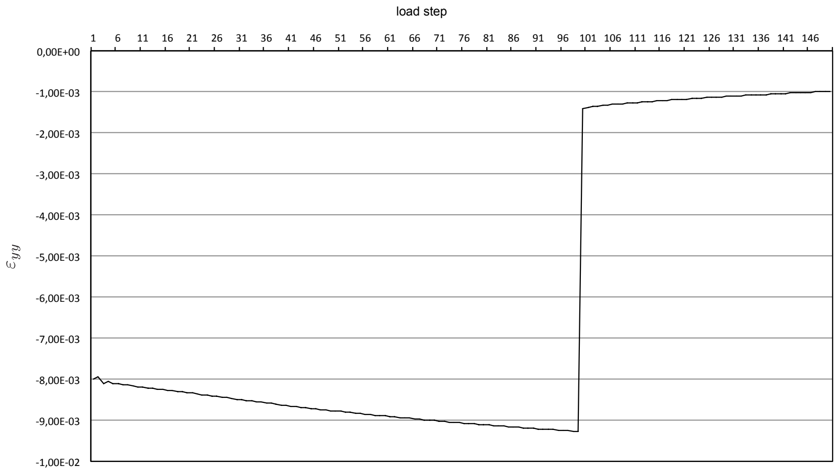


Figure 8.2: Total strain in y-direction of elasto-visco-plastic element.

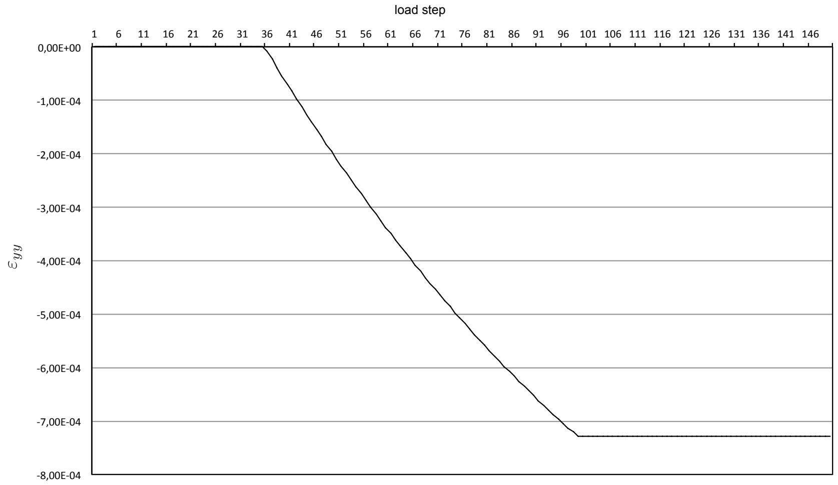


Figure 8.3: Plastic strain in y-direction of elasto-visco-plastic element.

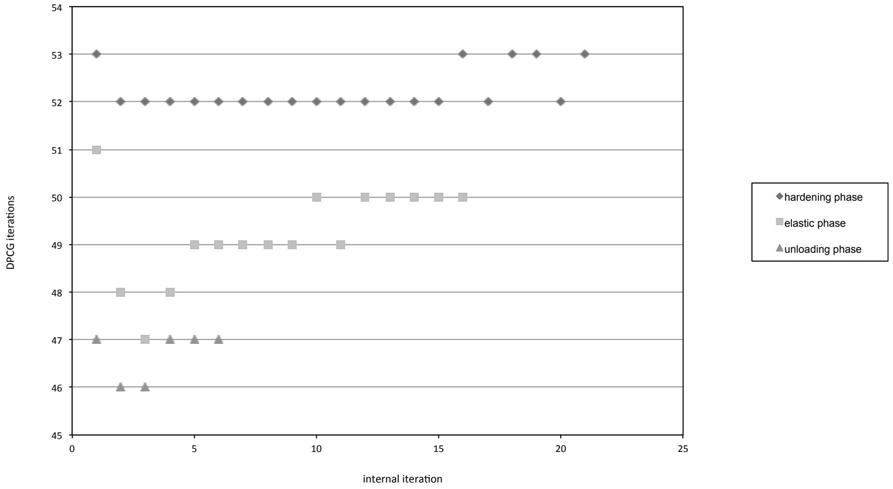


Figure 8.4: Number of DPCG iterations per internal Newton-Raphson iteration.

The number of iterations of the DPCG method for the artificial case are presented in Figure 8.4. We consider the number of DPCG iterations in the elastic phase (load step 1 - 36), the hardening phase (load step 37 - 99) and during the unloading of the material (load step 100 - 150). In Figure 8.4 we present the number of DPCG iterations for load step 2, 38 and 101 for the elastic phase, the hardening phase and the unloading respectively. We observe that the number of DPCG iterations varies slightly, but there is no significant increase between the elastic phase and the hardening phase.

In Figure 8.5 we present a boxplot of the number of DPCG iterations for all load steps. We consider the number of DPCG iterations for each Newton-Raphson iteration in each load step. We have 150 load steps and, on average 16 Newton-

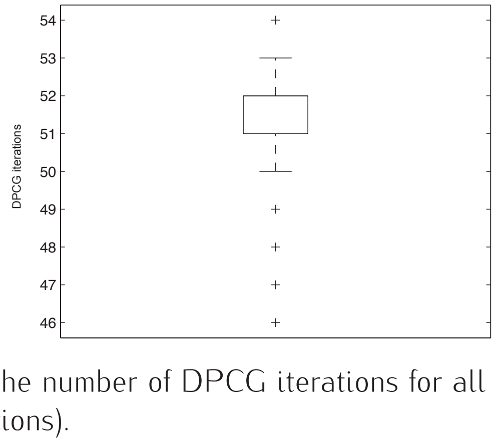


Figure 8.5: Boxplot of the number of DPCG iterations for all load steps (equals 2540 Newton-Rahpson iterations).

Raphson iterations at each load step; hence, we solve 2540 linear systems with the DPCG method. The lower and upper whiskers are defined at 2.7 times the standard deviation, which is at 50 and 53 iterations respectively. Therefore, in this case, 99.3% of all solves with the DPCG method have an iteration count between these boundaries.

8.3 Concluding remarks

In this chapter we show that there is no correlation between the *performance* of the DPCG method and the non-linearity of the material models that constitute the stiffness matrix and the corresponding loading vector. We have discussed the highly non-linear generalized elasto-visco-plastic material model that is well used in many real-life engineering applications. We have seen that during the elastic phase, the hardening phase and the unloading of the material the number of DPCG iterations does not vary significantly. We conclude that the DPCG method is robust for simulations with highly non-linear material models.

9

Summary and Conclusions

We define composite materials as connected bodies of materials. The modeling techniques of composite materials have significantly changed in the previous years, going from homogenization techniques to the microscopic behavior of the connected bodies of materials. In this research we study asphaltic material as an example of a composite material. We use CT-Scans to obtain fine meshes from asphaltic materials in which we distinguish three materials: aggregates, bitumen and air voids.

We introduce a mathematical framework to describe the force balance within the body of materials. The force misbalance leads to the formulation of the non-linear virtual work equation. Within the mathematical framework we consider three non-linear material properties: hyper elasticity, plasticity, and viscosity. These material properties are coupled to the virtual work equation by means of stress-strain relations.

We solve the non-linear work equation with the modified Newton-Raphson method, which yields the linearized virtual work equation. We argue that the most time consuming step within the modified Newton-Raphson method is the evaluation of the Jacobian.

We introduce the Finite-Element (FE) framework and discretize the linearized virtual work equation with linear, tetrahedral elements as the CT-Scans give rise to unstructured grids. The discretized linearized virtual work equation leads to the formulation of the stiffness matrix. We show that the stiffness matrix corresponds to the Jacobian of the modified Newton-Raphson method and we argue that we need an efficient solution method to solve the stiffness matrix, as many Newton-Raphson iterations are needed to obtain a force balance.

For all material models and loading cases considered in this research the stiffness matrix is symmetric positive definite. Moreover, in this research we consider large meshes, hence, the corresponding stiffness matrices are sparse and ill-conditioned.

We provide an overview of the existing solution methods. We argue that the Preconditioned Conjugate Gradient (PCG) method is the method of choice for sparse, symmetric positive definite matrices. Direct solution methods are very robust but very expensive in terms of memory. We discuss the Smoothed Aggregation Algebraic Multigrid method (SA-AMG), which is currently the method of choice for linear elasticity.

The starting numerical experiments involve composite materials consisting of two materials: aggregates and bitumen. These experiments show that the performance of PCG with diagonal scaling as preconditioner is poor. Increasing the stiffness as well as the number of aggregates in the composite materials results in a deterioration of the convergence rates of PCG. The results show plateaus in the convergence behavior which indicate that there are clusters of small eigenvalues related to slow converging components. Analysis of the spectrum of the preconditioned stiffness matrix shows that the smallest eigenvalues correspond to the domains containing aggregates. Moreover, we show that the number of small eigenvalues is equal to the number of rigid body modes of the aggregates.

We introduce the Deflated Preconditioned Conjugate Gradient (DPCG) method to filter out unfavorable eigenvalues from the spectrum of the stiffness matrix. We use the rigid body modes for the disjunct aggregates as the deflation subspace. The DPCG method will prove to be very robust and fast. The removal of rigid body modes results in a mechanical and mathematical well defined problem that only depends on the material properties of the bitumen. The convergence behavior of DPCG is identical to the convergence behavior of the case in which the composite material consists of only bitumen. The addition of more and stiffer aggregates has no effect on the performance of DPCG. Therefore we construct an iterative solver that has aggregate independent convergence behavior.

We introduce the recursive deflation operator and the deflation strategy to construct the most efficient deflation vectors for composite materials that consist of an arbitrary number of materials. We give a mathematical argument for the use of rigid body modes and show the relation between the decoupling of the sub matrices corresponding to individual materials and our deflation strategy. We argue and show that the right choice of deflation vectors leads to the right decoupling of the sub matrices, and thus to 'aggregate' independent convergence behavior of DPCG.

We discuss subdomain deflation in the context of parallel computing and domain decomposition, and argue that this is a useful addition to regular rigid body mode deflation as it improves the effect of 'local' preconditioners, such as Incomplete Cholesky. We also provide a sound mathematical argument on how to combine these two different 'flavors' of deflation in an efficient way.

We show by numerical experiments involving an academic test case as well as a real-life engineering test case that the DPCG method is insensitive to large jumps in the Young's modulus of materials. In the numerical examples considered in this thesis, the amount of work per iteration for the deflation operator of DPCG is comparable to one matrix-vector multiplication. However, this does not imply that DPCG becomes twice as expensive as PCG because the preconditioning step consumes most resources in both time and memory. For most applications, using sparse deflation vectors, DPCG is roughly 30% more expensive in time per iteration compared to PCG.

We show for the real-life engineering test cases that subdomain deflation leads to a very robust parallel DPCG method that addresses the problems with 'local' preconditioning. The number of iterations does not increase for an increasing number of subdomains in both experiments.

We compare three preconditioners with PCG and two with DPCG, using two implementations of smoothed aggregation SA-AMG, and diagonal scaling. For one implementation of SA-AMG, we choose the default parameter set and for the other implementation, we choose an optimal parameter set for the experiments involved. The latter is not used in conjunction with DPCG, due to large set up costs and the cost per iteration. For small and large linear systems, DPCG preconditioned by diagonal scaling proves to be slightly faster than preconditioning with SA-AMG. However, for large linear systems, PCG preconditioned by SA-AMG with an optimal parameter set outperforms DPCG but has a large setup time and requires significantly more software development. The DPCG method is well suited for parallel computing and can be implemented into any existing FE software package by using basic parallel linear algebraic operations.

A priori we would not consider direct solution methods as the methods of choice for the large meshes that come from the CT-scans of real-life samples of asphaltic materials. However, for a fair comparison to the state-of-the-art solution methods we have compare DPCG to MUMPS and SuperLU. Both tests are done on different hardware and software. MUMPS performs slightly better than DPCG, at the expense of higher memory consumption, and DPCG outperforms SuperLU. In the test case that involved the largest mesh, corresponding to a core of asphaltic material, the direct solution methods require roughly six times the amount of memory which is needed by DPCG. We conclude that with further optimization of the implementation of the DPCG algorithm we may increase the performance of DPCG and outperform both MUMPS and SuperLU.

The final conclusion of this thesis is that we have developed an extension to the existing theory on deflation, but aimed at three dimensional systems of coupled PDEs resulting from structural mechanics. We have developed a deflation strategy that removes those eigenvalues in the spectrum of the stiffness matrix that correspond to the rigid body modes of the bodies of material in the composite material. The resulting DPCG method is fast, easy to parallelize, robust, it may be combined with any preconditioner, it works well in simulations that use highly non-linear materials that occur in real-life engineering applications, and is easy to implement into existing (FE) software.

Samenvatting en conclusies

We definiëren composieten als structuren van lichamen van verschillende materialen. Het modelleren van composieten is sterk veranderd in de afgelopen jaren. Van homogeniseren tot het bestuderen van het microscopische gedrag van de lichamen in het composiet. In dit onderzoek bestuderen wij het composiet asfalt. We gebruiken CT-scans om gedetailleerde grids van deze composieten te verkrijgen, waarbij we drie materialen onderscheiden: stenen, bitumen en lucht.

We introduceren een wiskundig raamwerk om de krachten balans in het composiet, tussen en binnen de lichamen, te beschrijven. De krachten balans leidt tot de formulering van de niet-lineaire virtuele werk vergelijking. Binnen dit wiskundig raamwerk beschouwen we drie niet-lineaire materiaal eigenschappen: hyper elasticiteit, plasticiteit en viscositeit. Deze materiaal eigenschappen zijn gekoppeld aan de virtuele werk vergelijking door middel van de spanning-rek relaties.

We verkrijgen de gelineariseerde virtuele werk vergelijking door het oplossen van de niet-lineaire virtuele werk vergelijking met de Newton-Raphson methode. We stellen dat de evaluatie van de Jacobiaan, gemeten in tijd en gevraagde resources, de duurste stap is binnen de Newton-Rapson methode.

Omdat de CT-scans ongestructureerde grids van elementen opleveren, discretiseren wij de gelineariseerde virtuele work vergelijking met gebruik van de Eindige-Elementen methode (FE) en lineaire, tetraëdrische elementen. De assemblage van de stijfheids matrix volgt uit de gediscretiseerde gelineariseerde virtuele werk vergelijking. We laten zien dat de stijfheids matrix overeenkomt met de Jacobiaan van de gemodificeerde Newton-Raphson methode. Tevens stellen we dat we een efficiënte methode nodig hebben om het lineaire systeem behorend bij de stijfheids matrix op te lossen, omdat veel Newton-Raphson iteraties benodigd zijn om een krachten balans te bereiken.

De stijfheids matrix is symmetrisch, positief, definiet voor alle materiaal modellen en aangebrachte belastingen die we beschouwen in dit onderzoek. Tevens, de stijfheids matrices zijn ijl en slecht geconditioneerd omdat we in dit onderzoek zeer grote grids beschouwen.

We geven een overzicht van bestaande oplos methoden. We stellen dat de gepreconditioneerde geconjugeerde gradiënten methode (PCG) de juiste keuze is voor ijle, symmetrische, definiete matrices. Directe oplos methoden zijn erg robuust maar tevens erg kostbaar in het gebruik van geheugen. We bediscussiëren de 'Smoothed-Aggregation' algebraïsch multigrid methode (SA-AMG) die veelvuldig wordt gebruikt voor het oplossen van problemen met lineaire elasticiteit.

De eerste numerieke experimenten betreffen composieten die bestaan uit twee materialen: stenen en bitumen. De PCG methode met diagonaal schaling als preconditionering presteert matig in deze experimenten. Het toenemen van de stijfheid, alsmede het aantal stenen in de composieten resulteert in een verslechtering van de convergentie van PCG. De resultaten laten plateaus in de convergentie zien die erop duiden dat er clusters zijn van kleine eigenwaarden die behoren bij de langzaam convergerende componenten. Een analyse van het spectrum van de gepreconditioneerde stijfheids matrix laat zien dat de kleinste eigenwaarden behoren bij de domeinen die de stenen bevatten. Tevens, we laten ook zien dat het aantal kleine eigenwaarden overeenkomt met de energiewaarden van de rigide lichamen behorend bij de stenen.

We introduceren de gedefleerde geproconditioneerde geconjugeerde gradiënten methode (DPCG) om de ongunstige eigenwaarden uit het spectrum van de stijfheids matrix te verwijderen. We gebruiken de energie waarden van de rigide lichamen voor de disjuncte stenen voor de deflatie ruimte. De DPCG methode zal een robuuste en snelle methode blijken te zijn. Het verwijderen van de energie waarden van de rigide lichamen resulteert in een mechanisch en wiskundig goed gedefinieerd probleem dat qua oplosbaarheid alleen afhangt van de materiaal eigenschappen van het bitumen. De convergentie van DPCG is identiek aan de convergentie van PCG indien het totale volume uit slechts bitumen zou bestaan. De toevoeging van een groter aantal en meer stijve stenen heeft geen effect op de prestatie van DPCG. We kunnen stellen dat DPCG een oplos methode is, waarvan de convergentie niet afhankelijk is van het aantal evenals de eigenschappen van de stenen.

We introduceren de recursieve deflatie operator en een deflatie strategie om de meest efficiënte deflatie vectoren voor composieten die bestaan uit een willekeurig aantal materialen te construeren. We geven een wiskundig argument voor het gebruik van de energie waarden van de rigide lichamen en we laten de relatie zien tussen het ontkoppelen van de sub matrices die behoren bij individuele materialen en onze deflatie strategie. We stellen en laten zien dat de juiste keuze van de deflatie vectoren leidt tot de correcte ontkoppeling van de sub matrices en daardoor tot steen-onafhankelijke convergentie van DPCG.

We bediscussiëren sub domein deflatie in de context van parallel rekenen en domein decompositie en we stellen dat dit een nuttige toevoeging is tot de reguliere rigide lichaam deflatie omdat het de invloed versterkt van 'lokale' preconditioneringen, zoals de Incomplete Cholesky methode. We geven ook een wiskundig argument hoe deze twee deflatie 'smaken' te combineren op een efficiënte manier.

We laten zien door middel van numerieke experimenten, waarin we een academische casus en een praktijk casus beschouwen, dat de DPCG methode ongevoelig is voor grote sprongen in de Young's modulus van materialen. In de numerieke voorbeelden die worden beschouwd in deze dissertatie is de hoeveelheid werk per iteratie voor de deflatie operator in DPCG vergelijkbaar met één matrix-vector vermenigvuldiging. Echter, dit betekent niet dat DPCG twee maal zo duur is als PCG, dit komt doordat de preconditionering vergeleken met deflatie veel meer resources in tijd en geheugen vraagt. Voor de meeste toepassingen waarin we ijle deflatie vectoren gebruiken, is DPCG per iteratie ongeveer 30% duurder in tijd vergeleken met PCG.

We laten zien dat voor problemen uit de praktijk sub domein deflatie tot een heel robuuste parallelle DPCG methode leidt die de problemen adresseert met betrekking tot 'lokale' preconditionering. Het aantal iteraties neemt niet toe voor een toenemend aantal sub domeinen in beide experimenten.

We vergelijken PCG en DPCG met respectievelijk drie en twee preconditioneringen, waarbij we diagonaal schaling en twee implementaties van SA-AMG gebruiken. Voor de eerste implementatie van SA-AMG gebruiken we de standaard parameters en voor de tweede implementatie kiezen we de optimale parameters voor onze experimenten. De laatste wordt niet gebruikt in combinatie met DPCG vanwege de hoge opstart kosten en de kosten per iteratie. Voor kleine en grote lineaire systemen, DPCG gepreconditioneerd met diagonaal schaling is een fractie sneller dan preconditionering met SA-AMG. Echter, voor grote systemen, PCG gepreconditioneerd met SA-AMG en optimale parameters is een stuk sneller dan DPCG maar heeft een grotere opstart tijd en vraagt om significant meer software ontwikkeling. De DPCG methode is geschikt voor parallel rekenen en kan geïmplementeert worden in elk bestaand FE software pakket door gebruik te maken van basale parallelle lineaire algebraïsche operatoren.

A priori beschouwen we directe oplos methoden niet als de juiste methoden voor de grote grids die voortkomen uit de CT-scans van volumes van asfaltische materialen uit de praktijk. Echter, voor een juiste vergelijking met de state-of-the-art oplos methoden vergelijken we DPCG met MUMPS en SuperLU. Beide testen vinden plaats op verschillende hardware en met verschillende software. MUMPS presteert een fractie beter dan DPCG ten koste van een groter geheugen gebruik. DPCG presteert aanzienlijk beter dan SuperLU. In het experiment met het grootste grid, dat behoort bij een cilinder van asfaltisch materiaal, hebben de directe oplos methodes ongeveer zes maal zoveel geheugen nodig in vergelijking met DPCG. We concluderen dat met een verdere optimalisatie van het DPCG algoritme de prestaties van DPCG verbeteren en dat de methode beter zou kunnen presteren dan zowel MUMPS als SuperLU.

De afsluitende conclusie van deze dissertatie is dat we een uitbreiding op de bestaande deflatie theorie hebben ontwikkeld, toegepast op drie dimensionale systemen van gekoppelde partiële differentiaalvergelijkingen die voortkomen uit de mechanica. We hebben een deflatie strategie ontwikkeld die eigenwaarden uit het spectrum van de stijfheids matrix verwijderd die overeenkomen met de energie waarden van de rigide lichamen in de composieten. De resulterende DPCG methode is snel, robuust en kan worden gecombineerd met elke preconditionering. De methode werkt ook goed in de simulaties van materialen uit de praktijk waarin sterk niet-lineaire materiaal eigenschappen een rol spelen en is gemakkelijk te implementeren in bestaande (FE) software.

10

Future research

In this chapter we discuss loose ends and topics that are of interest for future research on deflation. We distinguish between two areas of research. We look into the improvement of existing deflation vectors and we discuss efficient implementation of the deflation operator.

10.1 On the improvement of deflation vectors

In Chapter 5 and 6, we discussed and motivated the principles of rigid body modes deflation as well as subdomain deflation. In this section we bring forward several ideas on how to improve the resulting deflation vectors.

10.1.1 Assessment of quality deflation vectors

To ensure accuracy and stability of DPCG, we need to ensure the linear independence of the deflation vectors, both analytically and numerically. In some numerical experiments, we observed a numerical loss of rank. In these cases, the modified Gram-Schmidt [34] algorithm for reorthogonalization works, but is expensive, and, hence, not useful in real-life engineering applications. In general, we would like an efficient test to indicate problems with rank. For normalized deflation vectors, the condition number of the deflation matrix E would be an useful indicator of deficiency of rank.

Another measure for the quality of the deflation vectors would be the Rayleigh quotient of the stiffness matrix and the normalized deflation vectors. This is an indication on how close the deflation vectors are to the eigenvectors corresponding to the smallest eigenvalues.

We propose the development of a numerical test to determine the quality of deflation vectors. The test has to be cheap in terms of resources and wall-clock time, as deflation vectors may have to be recomputed at every Newton-Raphson iteration.

10.1.2 Polynomial updates of deflation vectors

Let $Z = [z_1, z_2, \ldots, z_k]$ be a given full-rank deflation matrix. We want to ask the following question: *how do we "improve" the collection of vectors given by Z?*

If we start by considering deflated CG (DCG) with no preconditioning, we can get a reasonable answer to this question in the case of eigenvector deflation, by considering replacing deflation, thus, eigenvector z_ℓ for $1 \leq \ell \leq k$ with an eigenvector z_m for $m > k$. Suppose that z_ℓ is an eigenvector associated with an "interior" eigenvalue, while z_m is associated with an "extreme" eigenvalue, then such a switch is expected to improve the deflation performance. This is, of course, difficult to turn into a precise statement without making further assumptions on the distribution of the eigenvalues of K. A similar statement can be formulated for DPCG with preconditioning, based on eigenvectors of the preconditioned system, $M^{-1}K$.

In general, if the columns of Z are a more arbitrary set of deflation vectors than just eigenvectors (of either K or $M^{-1}K$), the question becomes more complicated. We lose the advantage of eigenvector deflation that any collection of k distinct eigenvectors in Z guarantees that deflation matrix Z has full rank. If we "replace" one of the columns, thus, deflation vectors, z_k of Z with an arbitrary vector, $\hat{z}$, we have no guarantee that $\hat{Z} = [z_1, z_2, \ldots, z_{k-1}, \hat{z}]$ remains full rank. Thus, any improvement that we do to the columns of Z, must be done in a way that the resulting matrix $\hat{Z}$, remains full rank. Hence, we cannot arbitrarily improve an individual column of Z, without worrying about the effect that this might have on the rank of the matrix. Instead, we want to find a way to improve the columns of Z without affecting their linear independence. This might be hard to achieve, requiring both assumptions on Z (perhaps beyond just being full rank) and on the process for improving an individual vector. One approach for "improving" the columns of Z is to make "polynomial" updates, of the form,

$$z_\ell \rightarrow (I - \omega_1^{(\ell)} K)(I - \omega_2^{(\ell)} K) \ldots (I - \omega_k^{(\ell)} K) z_\ell, \tag{10.1}$$

$$\text{or } z_\ell \rightarrow (I - \omega_1^{(\ell)} M^{-1}K)(I - \omega_2^{(\ell)} M^{-1}K) \ldots (I - \omega_k^{(\ell)} M^{-1}K) z_\ell, \tag{10.2}$$

for a suitable choice of weights.

Supposing that polynomial updates do, indeed, maintain the full rank nature of deflation matrix Z, the next question is whether or not they actually represent an improvement (in terms of the performance of the DCG/DPCG algorithm or the effective condition number of the deflated system). Here, it seems reasonable to restrict the range of the weights used in the polynomial update so that the columns of Z better represent the eigenvectors associated with the extreme eigenvalues of K or $M^{-1}K$.

In practice, there are several possibilities to improve vectors with a polynomial update. We consider the following options,

- We apply a few steps of the PCG method to the linear system $Kx = 0$ with $x_0 = z_i$, we obtain the minimum energy solution by default.
- The Lanczos method is the method of choice if we look for optimality in the sense of reducing the Rayleigh quotients to its minimum over the polynomial

space (Krylov space). We construct $AQ = QT$, where Q contains the Lanczos vectors based on vector z_i. The eigenvalues of T are the Ritz values of stiffness matrix K. Subsequently, the eigenvectors e from $\tilde{e} = Qe$ correspond to the Ritz vectors of stiffness matrix K.

- Damping of the eigenvalues of $M^{-1}K$ in the interval $[\frac{\lambda_n}{l}, \lambda_n]$ by Chebyshev polynomial updates, where λ_n is the largest eigenvalue of $M^{-1}K$ and l is to be determined. The largest eigenvalue λ_n can be obtained using a small number of DPCG iterations. We improve the deflation vectors by applying the k$^{\text{th}}$-order Chebyshev polynomial $p_k(K) = (1 - \mu_1 M^{-1}K)...(1 - \mu_k M^{-1}K)$ where $\mu_i = \frac{1}{2}(\lambda_n + \frac{\lambda_n}{l}) + \frac{1}{2}(\frac{\lambda_n}{l} - \lambda_n)\cos(\frac{2i-1}{2k}\pi)$.

We recommend further research into polynomial improvement to improve the robustness of DPCG, especially for subdomain deflation.

10.2 Computing and implementation

In this section we discuss future research into the efficient evaluation of the deflation operator. This might be achieved by recycling of deflation vectors and corresponding matrices, as well as taking into account the benefits of cutting edge computing techniques like GPU computing.

10.2.1 Recycling of deflation vectors

We have seen in the numerical examples of Chapter 7 and 8, and argued that there is no effect of highly non-linear materials on the convergence of DPCG. It remains to be seen what is the effect on the performance of DPCG when we reassemble the stiffness matrix at every Newton-Raphson iteration, but, to save time, do not recompute the deflation vectors, and, thus, matrices E, and KZ. As we have seen in Chapter 8, the stiffness of the materials changes due to hardening and softening effects, and thus the deflation vectors may not represent the correct rigid body modes. We propose further research into an efficient "recycling" of deflation vectors. In the case of highly non-linear material we want to recompute the deflation vectors, and, thus KZ, E, and, E^{-1} at every iteration of Newton-Raphson. For less dramatic changes in stiffness it can be interesting to save time, and, reuse the deflation vectors and corresponding matrices of the previous Newton-Raphson iteration.

We note that in this thesis we only considered modified Newton-Raphson. The proposed "recycling" of deflation vectors would have different effects on modified, respectively, standard Newton-Raphson. It remains to be seen what would be the trade-off between less iterations, and thus, run time (standard Newton-Raphson) and less (cumulative) deflation set-up time (modified Newton-Raphson).

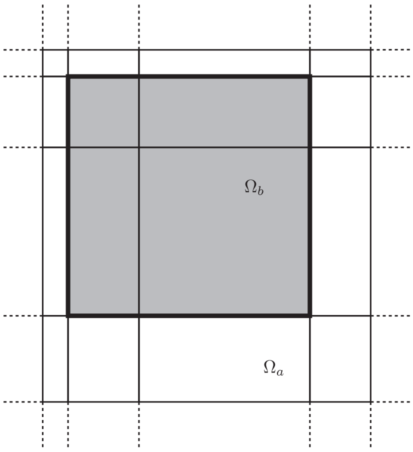


Figure 10.1: Two bodies, Ω_a, and, Ω_b, in finite element mesh.

10.2.2 GPU computing

The field of Graphics Processing Unit (GPU) computing has seen a surge in publications in the past few years. The improvements in the specifications, the explicit parallel architecture, and, low costs make GPUs an attractive alternative to super computers. GPUs are designed to efficiently process basic linear algebraic operations on small (dense) matrices and vectors. The deflation operator, and the parallel implementation, which we discussed in Chapter 5 and 6, seem to be well suited for implementation on the GPU. Moreover, the computation of the 3D rigid body modes would be a natural operation on a GPU. We propose future research into accelerating the evaluation of the deflation operator by the use of GPU computing. We refer to [38] for preliminary results.

10.2.3 Nonsymmetric matrices

Although we did not consider nonsymmetric systems, extensions to the mathematical framework on structural analysis that we introduced in Chapter 2, might lead to such systems. In the field of contact mechanics, imposed artificial "contact" constraints would lead to an nonsymmetric stiffness matrix. In that case, we can not use the theory on rigid body mode deflation of Chapter 5, however, as only small parts of the matrix would be affected by these constraints, we propose future research on deflation and nonsymmetric systems where the symmetric part of the system would be the dominating term. A natural choice would be deflation in combination with GMRES.

10.2.4 Reducing the number of deflation vectors

In Chapter 7 we briefly discussed the loss of rank of matrix Z, due to the collapse of deflation vectors. We illustrate this effect with Figure 10.1. We consider two bodies of different material, Ω_a, and, Ω_b. Bring in mind the deflation strategy of Chapter 5, we disregard all elements on the boundary of the rigid body that share nodes with elements that are owned by a material with higher stiffness. If the material of body Ω_a would have higher stiffness than the material of body Ω_b, we would obtain a deflation vector containing the rigid body modes that coincide with the null space of the stiffness matrix that corresponds to the assembly of all elements in body Ω_a. However, if the body Ω_b has higher stiffness than the material of body Ω_a, and because all elements of body Ω_a are boundary elements, we obtain a deflation vector with only zero entries. Hence, the deflation matrix is not of full rank. As discussed in Chapter 7, we can avoid this collapse of deflation vectors by using overlapping deflation vectors. This overlap leads to a significant reduction of deflation vectors. We propose future research into alternative techniques to use overlapping deflation vectors to reduce the number of deflation vectors without effecting the number of DPCG iterations.

10.2.5 Efficient computation with deflation vectors

In the current implementation of the DPCG algorithm we assemble the deflation vectors and use sparse storage for memory and CPU efficiency. However, depending on the deflation strategy, most deflation vectors are disjoint. With coloring algorithms we might condense (nearly) disjoint deflation vectors for storage and expand the vectors for computations. Moreover, the subdomain deflation vectors can be stored with precisely six (dense) vectors, taking into account the weighting of the boundary nodes when evaluating the deflation operator. We propose future research into the condensing and expansion of deflation vectors for efficient storage and evaluation of the deflation operator.

A

Notation

In the thesis tensors are utilized for the governing equations as it is common practice in structural mechanics. There are many different variations on the original tensor notation. The notation introduced in the book of dr. A. Scarpas [28] will be the notation utilized here. Vectors and matrices can be seen as 1st and 2nd order tensors respectively. Sometimes classic linear algebraic notation will be used, e.g. the transpose of a second order tensor $\left(\mathsf{A}_{ij}\right)^T = \mathsf{A}_{ji}$.

Tensors with index notation.

- 1st order tensor: A_i (vector)
- 2nd order tensor: Aij (matrix)
- 3rd order tensor: A_{ijk}
- 4th order tensor: A_{ijkl}

All tensors without index notation are written bold and with capitals, e.g. fourth order elasticity tensor, $\mathbb{C}$.

Operation	Index notation	Tensor notation
multiply	$A_{ij}B_{ij} = c$	$\mathsf{A} : \mathsf{B} = c$
	$A_{ik}B_{kj} = C_{ij}$	$\mathsf{A} \cdot \mathsf{B} = \mathbb{C}$
	$A_{ij}B_{j} = C_{i}$	$\mathsf{A} \cdot \mathsf{B} = \mathbb{C}$
	$a_i b_i = c$	$\mathbf{a} \cdot \mathbf{b} = c$
	$A_{ij}B_{ik} = C_{jk}$	$\mathsf{A}^T \cdot \mathsf{B} = \mathbb{C}$
	$A_{ij}B_{kl} = C_{ijkl}$	$\mathsf{A} \otimes \mathsf{B} = \mathbb{C}$
derivative	$\frac{\partial x_i}{\partial X_j} = A_{ij}$	$\frac{\partial \mathbf{x}}{\partial \mathsf{X}} = \mathsf{A}$
	$\frac{\partial X_i}{\partial X_j} = \delta_{ij}$	$\frac{\partial \mathsf{X}}{\partial \mathsf{X}} = \mathbf{I}$
	$\frac{\partial X_{ij}}{\partial X_{kl}} = \delta_{ik}\delta_{jl} = C_{ikjl}$	$\mathsf{X} \otimes \frac{\partial}{\partial \mathsf{X}} = \mathbb{C}$
kronicker-delta	$\delta_{kl}\delta_{lj} = \delta_{kj}$	$\mathbf{I} \cdot \mathbf{I} = \mathbf{I}$
	$\delta_{ij}\delta_{kl} = \delta_{ijkl}$	$\mathbf{I} \otimes \mathbf{I}$
	$\delta_{ik}\delta_{jl}$	$\mathbf{I}$

Table A.1: Tensor and index notation

$$\begin{aligned}
\mathsf{A} : \mathsf{B} &= \mathsf{B} : \mathsf{A} \\
\mathsf{A} : \mathsf{B} &= \mathsf{A}^T : \mathsf{B}^T \\
(\mathsf{A} \cdot \mathsf{B})^T &= \mathsf{B}^T \cdot \mathsf{A}^T \\
(\mathsf{A} \cdot \mathsf{B}) : \mathbb{C} &= \left(\mathbb{C}^T \cdot \mathsf{A}\right) : \mathsf{B}^T \\
(\mathsf{A} \cdot \mathsf{B}) : \mathbb{C} &= \mathsf{B} : \left(\mathsf{A}^T \cdot \mathbb{C}\right) \\
\mathbf{a} \otimes \mathbf{b} &= (\mathbf{b} \otimes \mathbf{a})^T \\
\mathbf{a} \otimes \mathbf{b} &= \mathbf{a} \cdot (\mathbf{I} \otimes \mathbf{b}) \\
\mathsf{A} : (\mathbf{b} \otimes \mathbb{C}) &= \mathbf{b} \cdot (\mathsf{A} : (\mathbf{I} \otimes \mathbb{C}))
\end{aligned}$$

Table A.2: Basic tensor computations.

Bibliography

[1] Adams, M., Brezina, M., Hu, J., and Tuminaro, R. Parallel multigrid smoothing: polynomial versus Gauss-Seidel. *J. Comput. Phys. 188* (2003), 593–610.

[2] Adams, M. F. Parallel multigrid solvers for 3d unstructured finite element problems in large deformation elasticity and plasticity. *International Journal for Numerical Methods in Engineering 48*, 8 (2000), 1241–1262.

[3] Adams, M. F. Evaluation of three unstructured multigrid methods on 3D finite element problems in solid mechanics. *International Journal for Numerical Methods in Engineering 55*, 5 (2002), 519–534.

[4] Adams, M. F., Bayraktar, H., Keaveny, T., and Papadopoulos, P. Ultrascalable implicit finite element analyses in solid mechanics with over a half a billion degrees of freedom. In *ACM/IEEE Proceedings of SC2004: High Performance Networking and Computing* (2004).

[5] Alcouffe, R. E., Brandt, A., Dendy, J. E., and Painter, J. W. The multigrid method for the diffusion equation with strongly discontinuous coefficients. *SIAM J. Sci. Stat. Comput. 2* (1981), 430–454.

[6] Amestoy, P. R., Duff, I. S., Koster, J., and L'Excellent, J.-Y. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications 23*, 1 (2001), 15–41.

[7] Amestoy, P. R., Duff, I. S., and L'Excellent, J.-Y. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Comput. Methods Appl. Mech. Eng. 184* (2000), 501–520.

[8] Arbenz, P., van Lenthe, G. H., Mennel, U., Müller, R., and Sala, M. A scalable multi-level preconditioner for matrix-free μ-finite element analysis of human bone structures. *Internat. J. Numer. Methods Engrg. 73*, 7 (2008), 927–947.

[9] Axelsson, O. *On preconditioning and convergence acceleration in sparse matrix problems.* CERN (Series) ; 74-10., 1974.

[10] Baker, A. H., Kolev, T. V., and Yang, U. M. Improving algebraic multigrid interpolation operators for linear elasticity problems. *Numer. Linear Algebra Appl. 17*, 2-3 (2010), 495–517.

[11] Bathe, K. J. *Finite Element Procedures*, 2 revised ed. Prentice Hall, June 1995.

[12] Bollhöfer, M., and Saad, Y. Multilevel preconditioners constructed from inverse-based ilus. *SIAM J. Sci. Comput. 27*, 5 (2006), 1627–1650.

[13] Bonet, J., and Wood, R. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Wiley, 1999.

[14] Brandt, A., Brannick, J., Kahl, K., and Livshits, I. Bootstrap AMG. *SIAM Journal on Scientific Computing 33*, 2 (2011), 612–632.

[15] Brandt, A., McCormick, S. F., and Ruge, J. W. Algebraic multigrid (AMG) for sparse matrix equations. In *Sparsity and Its Applications*, D. J. Evans, Ed. Cambridge University Press, Cambridge, 1984.

[16] Brannick, J., and Zikatanov, L. Algebraic multigrid methods based on compatible relaxation and energy minimization. In *Domain decomposition methods in science and engineering XVI*, vol. 55 of *Lect. Notes Comput. Sci. Eng.* Springer, Berlin, 2007, pp. 15–26.

[17] Brezina, M., Tong, C., and Becker, R. Parallel algebraic multigrids for structural mechanics. *SIAM J. Sci. Comput. 27*, 5 (2006), 1534–1554.

[18] Briggs, W. L., Henson, V. E., and McCormick, S. F. *A Multigrid Tutorial.* SIAM Books, Philadelphia, 2000. Second edition.

[19] CAPA3D. Capa-3d computer aided pavement analysis. http://www.capa-3d.org, 2009.

[20] Chen, W. *Constitutive Equations for Engineering Materials.* Elsevier, 1994.

[21] Clees, T. *AMG Strategies for PDE Systems with Applications in Industrial Semiconductor Simulation.* PhD thesis, Universität zu Köln, Köln, Germany, 2005.

[22] Coleman, B. D., and Noll, W. The thermodynamics of elastic materials with heat conduction and viscosity. *Archive for Rational Mechanics and Analysis 13*, 1 (Dec. 1963), 167–178.

[23] Collop, A., Scarpas, A., Kasbergen, C., and de Bondt, A. Development of an elasto-visco-plastic constitutive model for asphalt c.a. In *Proceedings of the 2003 International Conference on Computational, Experimental and Engineering Sciences (ICCES 03)* (2003), Tech Science Press.

[24] Davis, T. *Direct Methods for Sparse Linear Systems (Fundamentals of Algorithms)*, illustrated edition ed. Society for Industrial and Applied Mathematic, Sept. 2006.

[25] Demmel, J., Gilbert, J., and Li, X. SuperLU users' guide. Tech. Rep. CSD-97-944, 1997.

[26] Desai, C. *Numerical Methods and Constitutive Modelling in Geomechanics*. Springer, 1991.

[27] Douglas, C. C., Haase, G., and Langer, U. *A Tutorial on Elliptic Pde Solvers and Their Parallelization (Software, Environments, and Tools)*, illustrated edition ed. Society for Industrial and Applied Mathematic, Apr. 2003.

[28] Drescher, A., Kringos, N., and Scarpas, T. On the behavior of a parallel elasto-visco-plastic model for asphaltic materials. *Mechanics of Materials* (October 2009).

[29] Erkens, S. *Asphalt Concrete Response: Determination, Modelling, and, Predicition.* PhD thesis, TU Delft, Delft, the Netherlands, 2002.

[30] Faires, D. J., and Burden, R. L. *Numerical Methods*. PWS-Kent Pub.Co., Boston, 1993.

[31] Falgout, R., and Yang, U. hypre: a library of high performance preconditioners. In *Computational Science - ICCS 2002: International Conference, Amsterdam, The Netherlands, April 21-24, 2002. Proceedings, Part III*, no. 2331 in Lecture Notes in Computer Science. Springer-Verlag, 2002, pp. 632–641.

[32] Frank, J., and Vuik, C. On the construction of deflation-based preconditioners. *SIAM J. Sci. Comput. 23*, 2 (2001), 442–462.

[33] Gee, M., Siefert, C., Hu, J., Tuminaro, R., and Sala, M. ML 5.0 smoothed aggregation user's guide. Tech. Rep. SAND2006-2649, Sandia National Laboratories, 2006.

[34] Golub, G. H., and Van Loan, C. F. *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences).* The Johns Hopkins University Press, Baltimore, October 1996.

[35] Graham, I. G., and Scheichl, R. Robust domain decomposition algorithms for multiscale pdes. *Numerical Methods for Partial Differential Equations 23* (2007), 859–878.

[36] Grama, A., Karypis, G., Kumar, V., and Gupta, A. *Introduction to Parallel Computing (2nd Edition).* Addison Wesley, January 2003.

[37] Griebel, M., Oeltz, D., and Schweitzer, M. A. An algebraic multigrid method for linear elasticity. *SIAM J. Sci. Comput. 25*, 2 (2003), 385–407 (electronic).

[38] Gupta, R., Vuik, C., and Lemmens, C. Gpu implementation of deflated preconditioned conjugate gradient. Tech. Rep. 10-20, TU Delft, 2010.

[39] Henson, V., and Yang, U. BoomerAMG: a parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics 41* (2002), 155–177.

[40] Heroux, M. A., Bartlett, R. A., Howle, V. E., Hoekstra, R. J., Hu, J. J., Kolda, T. G., Lehoucq, R. B., Long, K. R., Pawlowski, R. P., Phipps, E. T., Salinger, A. G., Thornquist, H. K., Tuminaro, R. S., Willenbring, J. M., Williams, A., and Stanley, K. S. An overview of the trilinos project. *ACM Trans. Math. Softw. 31*, 3 (2005), 397–423.

[41] Hestenes, M. R., and Stiefel, E. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards 49* (Dec 1952), 409–436.

[42] Holzapfel, G. *Nonlinear Solid Mechanics: A Continuum Approach for Engineering.* Cambridge University Press, 2000.

[43] Householder, A. *Principles of Numerical Analysis.* New York: McGraw-Hill, 1953.

[44] Jönsthövel, T. B. Preconditioned conjugate gradient method enhanced by deflation of rigid body modes applied to composite materials. In *18th world IMACS congress and MODSIM09 international congress on modeling and simulation* (2009), Modeling and simulation society of Australia and New- Zealand., pp. 33–33.

[45] Jönsthövel, T. B., Liu, X., Scarpas, A., and Vuik, C. Parallel direct solver for linear systems resulting from constitutive modeling of pavement. In *Performance modeling and evaluation of pavement systems and materials* (2009), Geotechnical Special Publication, pp. 90–95.

[46] Jönsthövel, T. B., van Gijzen, M. B., Kasbergen, C., and Scarpas, A. Preconditioned conjugate gradient method enhanced by deflation of rigid body modes applied to composite materials. *CMES-computer modeling in engineering & sciences* (2009), 97–118.

[47] Jönsthövel, T. B., van Gijzen, M. B., MacLachlan, S., Scarpas, A., and Vuik, C. Comparison of the deflated preconditioned conjugate gradient method and algebraic multigrid for composite materials. *Computational Mechanics* (2011).

[48] Jönsthövel, T. B., van Gijzen, M. B., and Vuik, C. On the use of rigid body modes in the deflated preconditioned conjugate gradient method. *SIAM Journal on Scientific Computing (SISC)* (2011).

[49] Kaasschieter, E. F. Preconditioned conjugate gradients for solving singular systems. *J. Comput. Appl. Math. 24*, 1-2 (1988), 265–275.

[50] Kaplan, W. *Advanced Calculus, 4th edition.* Addison-Wesley, 1991.

[51] Karer, E., and Kraus, J. K. Algebraic multigrid for finite element elasticity equations: determination of nodal dependence via edge-matrices and two-level convergence. *Internat. J. Numer. Methods Engrg. 83*, 5 (2010), 642–670.

[52] Karypis, G., and Kumar, V. A coarse-grain parallel formulation of multilevel k-way graph partitioning algorithm. In *PARALLEL PROCESSING FOR SCIENTIFIC COMPUTING. SIAM* (1997).

[53] Klawonn, A., and Widlund, O. A domain decomposition method with lagrange multipliers and inexact solvers for linear elasticity. *SIAM J. Sci. Comput. 22* (2000), 1199–1219.

[54] Klawonn, A., Widlund, O., and Maksymilian, D. Dual-primal feti methods for three-dimensional elliptic problems with heterogeneous coefficients. *SIAM J. Numer. Anal. 40* (2002), 159–179.

[55] Kringos, N. *Modeling of combined physical-mechanical moisture induced damage in asphaltic mixes.* TU Delft, 2007.

[56] Li, X., and Demmel, J. W. Superlu dist: A scalable distributed-memory sparse direct solver for unsymmetric linear systems. *ACM Trans. Mathematical Software 29* (2003), 110–140.

[57] Lu, T.-T., and Shiou, S.-H. Inverses of 2 x 2 block matrices,. *Computers and Mathematics with Applications 43*, 1-2 (2002), 119 – 129.

[58] Lurie, A. *Nonlinear Theory of Elasticity.* North-Holland, 1990.

[59] MacLachlan, S., Manteuffel, T., and McCormick, S. Adaptive reduction-based AMG. *Numer. Linear Alg. Appl. 13* (2006), 599–620.

[60] MacLachlan, S. P., Tang, J. M., and Vuik, C. Fast and robust solvers for pressure-correction in bubbly flow problems. *J. Comput. Phys. 227*, 23 (2008), 9742–9761.

[61] Malvern, L. *Introduction to the Mechanics of a Continuous Medium*. Prentice Hall, 1977.

[62] Meijerink, J. A., and van der Vorst, H. A. Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems. *Journal of Computational Physics 44*, 1 (1981), 134–155.

[63] Moler, C. B. MATLAB user's guide. Tech. rep., University of New Mexico. Dept. of Computer Science, Nov. 1980. This describes use of Classic Matlab, the prototype for the very-much expanded professional Matlab from The MathWorks. Classic Matlab is no longer available.

[64] Newmark, N. M. A method of computation for structural dynamics. *Journal of Engineering Mechanics, ASCE.* (1959), 67–94.

[65] Nicolaides, R. A. On some theoretical and practical aspects of multigrid methods. *Math. Comp. 33* (1979), 933–952.

[66] Nicolaides, R. A. Deflation of conjugate gradients with applications to boundary value problems. *SIAM J. Numer. Anal. 24*, 2 (1987), 355–365.

[67] Olson, L. N., Schroder, J. B., and Tuminaro, R. S. A general interpolation strategy for algebraic multigrid using energy minimization. *SIAM Journal on Scientific Computing 33*, 2 (2011), 966–991.

[68] Polizzi, E., and Sameh, A. H. A parallel hybrid banded system solver: the spike algorithm abstract, 2005.

[69] Reid, J. The use of conjugate gradients for systems of linear equations possessing Òproperty aÓ. *SIAM J. Numer. Anal.* (1972), 325–332.

[70] Ruge, J. Amg for problems of elasticity. *Applied Mathematics and Computation 19*, 1-4 (1986), 293 – 309.

[71] Ruge, J. W., and Stüben, K. Algebraic multigrid (AMG). In *Multigrid Methods*, S. F. McCormick, Ed., vol. 3 of *Frontiers in Applied Mathematics*. SIAM, Philadelphia, PA, 1987, pp. 73–130.

[72] Saad, Y. *Iterative Methods for Sparse Linear Systems, Second Edition.* Society for Industrial and Applied Mathematics, Philadelphia, April 2003.

[73] SandiaLabs. Cubit geometry and mesh generation toolkit. http://cubit.sandia.gov/, 2009.

[74] Scarpas, A. *Mechanics based computational platform for pavement engineering.* PhD thesis, TU Delft, Delft, the Netherlands, 2004.

[75] Schellekens, J. *Computational strategies for composite structures.* PhD thesis, TU Delft, Delft, the Netherlands, 1992.

[76] Schenk, O., Gärtner, K., Fichtner, W., and Stricker, A. Pardiso: a high-performance serial and parallel sparse linear solver in semiconductor device simulation. *Future Generation Computer Systems 18*, 1 (2001), 69–78.

[77] Simpleware. http://www.simpleware.com, 2009.

[78] Sluys, L., and de Borst, R. *Computational methods in non-linear solid mechanics.* TU Delft, 2001.

[79] Tang, J. *Two-Level Preconditioned Conjugate Gradient Methods with Applications to Bubbly Flow Problems.* PhD Thesis TU Delft, 2008.

[80] Tang, J., MacLachlan, S., Nabben, R., and Vuik, C. A comparison of two-level preconditioners based on multigrid and deflation. *SIAM. J. Matrix Anal. and Appl. 31* (2010), 1715–1739.

[81] Tang, J., Nabben, R., Vuik, C., and Erlangga, Y. Comparison of two-level preconditioners derived from deflation, domain decomposition and multigrid methods. *Journal of Scientific Computing 39* (2009), 340–370.

[82] Trottenberg, U., Oosterlee, C. W., and Schuller, A. *Multigrid.* Academic Press, December 2000.

[83] Van der Sluis, A., and Van der Vorst, H. The rate of convergence of conjugate gradients. *Numer. Math. 48*, 5 (1986), 543–560.

[84] van der Vorst, H. *Iterative Krylov Methods for Large Linear Systems.* Cambridge Universuty Press, 2003.

[85] Vanék, P., Mandel, J., and Brezina, M. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing 56* (1996), 179–196.

[86] Vuik, C. *Numerical methods for large algebraic systems*. TU Delft, 2004.

[87] Vuik, C., Jönsthövel, T. B., Kasbergen, C., and Scarpas, A. Preconditioned conjugate gradient method enhanced by deflation of rigid body modes applied to composite materials. In *Advances in contact mechanics: a tribute to Prof. J.J. Kalker* (2008), TU Delft, pp. 35–36.

[88] Wesseling, P. *An Introduction to Multigrid Methods*. John Wiley & Sons, Chichester, 1992.

[89] Wriggers, P., and Boersma, A. A parallel algebraic multigrid solver for problems in solid mechanics discretisized by finite elements. *Computers & Structures 69*, 1 (1998), 129 – 137.

[90] Zienkiewicz, O. C., and Taylor, R. L. *The Finite Element Method for Solid and Structural Mechanics, Sixth Edition*, 6 ed. Butterworth-Heinemann, Sept. 2005.

List of publications

Journal papers

- Jönsthövel, T.B., Oosterlee, CW, Mulder, WA (2006). Improving Multigrid for 3D electro-magnetic diffusion on stretched grids. ECCOMAS CFD 2006. Delft: Tu Delft.
- Jönsthövel, T.B., Liu, X, Scarpas, A, Vuik, C (2009). Parallel direct solver for linear systems resulting from constitutive modeling of pavement. In Ceylan Halil, Liu Xueyan, Gopalakrishnan Kasthurirangan & Huang Likui (Eds.), Performance modeling and evaluation of pavement systems and materials Vol. 195. Geotechnical Special Publication (pp. 90-95). Virginia, USA: American Society of Civil Engineers.
- Jönsthövel, T.B., Gijzen, M.B. van, Vuik, C, Kasbergen, C, Scarpas, A (2009). Preconditioned conjugate gradient method enhanced by deflation of rigid body modes applied to composite materials. CMES-computer modeling in engineering & sciences, 47(2), 97-118.
- Jönsthövel, T.B., Gijzen, M.B. van, MacLachlan, S, Scarpas, A, Vuik, C (2011). Comparison of the Deflated Preconditioned Conjugate Gradient method and Algebraic Multigrid for composite materials. Computational Methods for Interface Mechanical Problems. (SICM)
- Jönsthövel, T.B., Gijzen, M.B. van, Vuik C (2011). On the use of rigid body modes in the deflated preconditioned conjugate gradient method. SIAM Journal on Scientific Computing (SISC). **(accepted)**

Proceedings and relevant talks

- Vuik, C, Gijzen, M.B. van, Scarpas, A, Kasbergen, C, Jönsthövel, T.B. (2008). Numerical methods for algebraic problems from structural mechanics. In T Scarpas, C Vuik & Y Sutjiadi (Eds.), Advances in contact mechanics: a tribute to Prof. J.J. Kalker extented summaries (pp. 35-36). Delft: Tu Delft.

- Woudschoten Conference, Woudschoten, the Netherlands - 2009, 2010 : poster presentation
- Jönsthövel, T.B. (2009). Preconditioned conjugate gradient method enhanced by deflation of rigid body modes applied to composite materials. In R.S. Anderssen, R.D. Braddock & R.S. Anderssen (Eds.), 18th world IMACS congress and MODSIM09 international congress on modeling and simulation (pp. 33-33). Cairns: Modeling and simulation society of Australia and New-Zealand.
- Mini-symposium invited speaker, group of Structural Mechanics, Department of Civil Engineering, University of New South Wales, Sydney, Australia - 2009 : presentation
- GeoHunan, Changsha, China - 2009 : paper and presentation at session "Advanced Constitutive Modeling of Asphaltic Concrete Materials.
- ECCM 2010, Paris, France - 2010 : contribution to presentation "Fast iterative methods for mechanical problems with interfaces" by M.B. van Gijzen.
- ICCES10, Las Vegas, USA - 2010 : paper and presentation at session "Mechanics of Composite Materials and Structures.
- 11th Copper Mountain Conference on Iterative Methods, Copper Mountain, Colorado, USA - 2010 : abstract and presentation at session "Krylov Techniques"
- PMAA10, Basel, Swiss - 2010 : abstract and presentation at session "Miscellaneous A"
- Invited speaker at SIAM Student Chapter, Tufts University, Medford, USA - 2011
- ICIAM 2011, Vancouver - 2011 : abstract and presentation at session 'New Developments in Iterative Solvers for Large Sparse Problems'

Curriculum Vitae

Tom Bernard Jönsthövel was born on the 18th of July 1983, in Dordrecht, the Netherlands. He completed secondary school at Titus Brandsmacollege (Dordrecht) in 2001. From 2001 to 2002 he studied Applied Mathematics at Eindhoven University of Technology (TUE), and from 2002 he continued his studies at Delft University of Technology (TU Delft) where he obtained his Bachelor of Science and Master of Science degree in 2005, respectively 2006. His Master research, "Improving Multigrid for 3D electro-magnetic diffusion on stretched grids.", was carried out at the Shell Research and Development facility in Rijswijk, Netherlands under supervision of Prof. Dr. W.A. Mulder and Prof. Dr. C. Oosterlee.

He did his PhD studies at TU Delft in the group of Numerical Analysis of the Department of Applied Mathematics, where he was supervised by Prof. Dr. C. Vuik, Dr. van Gijzen, and, Dr. A. Scarpas. In the framework of his PhD project he has worked as a researcher at the Schlumberger Research and Development facility in Cambridge, UK, at Tufts University, Boston, USA and at the University of New South Wales, Sydney, Australia.

Besides his PhD studies, he started his own engineering consultancy company, and worked as a professional studio musician, film score composer, and, pianist in a theater tour all around the Netherlands. His major interests are scientific computing, music (composing/performing), traveling and photography.