Two-Level Preconditioned Conjugate Gradient Methods with Applications to Bubbly Flow Problems

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.dr.ir. J.T. Fokkema,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
maandag 8 september 2008 om 15:00 uur

door

Jok Man TANG

wiskundig ingenieur

geboren te Utrecht.

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

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof.dr.ir. C. Vuik,	Technische Universiteit Delft, promotor
Prof.dr.ir. H. Bijl,	Technische Universiteit Delft
Prof.dr.ir. B.J. Boersma,	Technische Universiteit Delft
Prof.dr. R. Nabben,	Technische Universität Berlin, Duitsland
Prof.dr.ir. C.W. Oosterlee,	Technische Universiteit Delft
Prof.dr.ir. S. Vandewalle,	Katholieke Universiteit Leuven, België
Prof.dr.ir. P. Wesseling,	Technische Universiteit Delft

Keywords: conjugate gradient method, two-level preconditioners, deflation, domain decomposition, multigrid, bubbly flows, Poisson equation with a discontinuous coefficient, singular symmetric positive semi-definite matrices.

TU Delft

The work described in this dissertation was carried out in the section of Numerical Analysis at the Department of Applied Mathematics, Delft Institute of Applied Mathematics, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, The Netherlands.

BRICKS

Part of the research described in this dissertation has been funded by the Dutch BSIK/BRICKS project.

Two-Level Preconditioned Conjugate Gradient Methods with Applications to Bubbly Flow Problems.
Dissertation at Delft University of Technology.
Copyright © 2008 by J.M. Tang

ISBN 978-90-8559-398-0

To my parents,
my mother Lisa Tang-Lam,
and my father Wing-Cheung Tang

Summary

Two-Level Preconditioned Conjugate Gradient Methods with Applications to Bubbly Flow Problems

Jok M. Tang

The Preconditioned Conjugate Gradient (PCG) method is one of the most popular iterative methods for solving large linear systems with a symmetric and positive semi-definite coefficient matrix. However, if the preconditioned coefficient matrix is ill-conditioned, the convergence of the PCG method typically deteriorates. Instead, a two-level PCG method can be used. The corresponding two-level preconditioner usually treats unfavorable eigenvalues of the coefficient matrix effectively, so that the two-level PCG method is expected to converge faster than the original PCG method. Many two-level preconditioners are known in the fields of deflation, multigrid and domain decomposition methods. Several of them are discussed in this thesis, where the main focus is on the deflation method.

We show some theoretical properties of the deflation method, which give insights into the effectiveness of this method. A crucial component of the deflation preconditioner is the choice of projection vectors. Several choices are discussed and examined. We advocate that subdomain projection vectors, which are based on disjoint and piecewise-constant vectors, are among the best choices for a class of problems.

Subsequently, we examine the application of the deflation method to linear systems with singular coefficient matrices. Several mathematically equivalent variants of the original deflation method are proposed to deal with the possible singularity of this coefficient matrix. In addition, two approaches are discussed in order to handle coarse linear systems with a Galerkin matrix, which are involved in each iteration of the deflation method. After the discussion of the implementation and efficiency issues of the deflation method, it is demonstrated that this method is usually faster than the original PCG method.

Moreover, we present a comparison between the deflation method and other well-known two-level PCG methods, among them the balancing-Neumann-Neumann, additive coarse-grid correction, and multigrid methods based on symmetric and nonsymmetric V-cycles. As the parameters of the corresponding two-level preconditioners are abstract, we show that these methods are strongly connected to each other. The comparison is also done where the different two-level PCG methods adopt their typical and optimized set of parameters. Numerical experiments show that some multigrid methods are attractive in addition to the deflation method.

The major application of this thesis is the Poisson equation with a discontinuous coefficient, which is derived from 2-D and 3-D bubbly flow problems. Most of the performed numerical experiments in this thesis are based on this equation. Both stationary and time-dependent experiments are carried out to emphasize the theoretical results. We show that two-level PCG methods are significantly faster than the original PCG method in almost all experiments. Hence, computations involved in bubbly flows can be performed very efficiently using these PCG methods.

Samenvatting

Tweelaags Gepreconditioneerde Geconjugeerde Gradiënten Methoden met Toepassingen in Stromingsproblemen met Bellen

Jok M. Tang

De gepreconditioneerde geconjugeerde gradiënten (PCG) methode is één van de meest populaire iteratieve methoden voor het oplossen van grootschalige lineaire systemen, waarbij de coëfficiëntenmatrix symmetrisch en positief semi-definiet is. Echter, als de gepreconditioneerde coëfficiëntenmatrix slecht geconditioneerd is, dan vertoont de PCG methode langzame convergentie. In plaats hiervan kan de tweelaagse PCG methode gebruikt worden die gebaseerd is op een tweelaagse preconditioner. Deze preconditioner elimineert de effecten van de kleine en grote eigenwaarden van de coëfficiëntenmatrix, waardoor de tweelaagse PCG methode sneller convergeert dan de oorspronkelijke methode. Vele tweelaagse preconditioners zijn bekend in de vakgebieden van deflatie, multirooster en domein decompositie methoden. In dit proefschrift onderzoeken we deze preconditioners nader, waar we ons voornamelijk concentreren op de deflatie methode.

We laten theoretische eigenschappen van de deflatie methode zien, die inzicht geven in de effectiviteit van deze methode. Een cruciale component van de deflatie preconditioner is de keuze van de projectievectoren. Diverse keuzes worden beargumenteerd en onderzocht. We laten zien dat subdomein projectievectoren, die gebaseerd zijn op disjuncte en stuksgewijs constante vectoren, een van de beste keuzes zijn voor een specificieke klasse van problemen.

Vervolgens onderzoeken we de toepassing van de deflatie methode op lineaire systemen waarbij de coëfficiëntenmatrix singulier is. Verscheidene wiskundig equivalente varianten afgeleid van de originele deflatie methode worden behandeld. Deze varianten zijn bestand tegen de mogelijke singulariteit van de coëfficiëntenmatrix. Verder worden twee varianten bekeken die geschikt zijn om kleinere lineaire systemen binnen de deflatie methode op te lossen waarbij de Galerkin matrix betrokken is. Na het behandelen van de implementatie en de efficiëntie van de deflatie methode, laten we zien dat deze methode in de meeste gevallen sneller convergeert dan de originele PCG methode.

Verder presenteren we een vergelijking tussen de deflatie methode en andere bekende tweelaagse PCG methoden, waaronder de gebalanceerde Neumann-Neumann, additief grof-rooster correctie en multirooster methoden gebaseerd op symmetrische en niet-symmetrische V-cycli. Indien de parameters in de te beschouwen tweelaagse preconditioners gelijk zijn, kunnen we aantonen dat de verschillende methoden sterk aan elkaar gerelateerd zijn. De vergelijking is verder ook uitgevoerd, waarbij de tweelaagse PCG methoden hun karakteristieke en geoptimaliseerde verzameling van parameters aannemen. Numerieke experimenten laten zien dat sommige multirooster methoden attractief zijn naast de deflatie methode.

De belangrijkste toepassing in dit proefschrift is de Poisson vergelijking met een discontinue coëfficiënt, hetgeen afgeleid is van 2-D en 3-D twee-fase stromingsproblemen met bellen. De meeste van de uitgevoerde numerieke experimenten zijn gebaseerd op deze vergelijking. Zowel stationaire als tijdsafhankelijke experimenten zijn uitgevoerd om de theoretische resultaten te onderbouwen. We laten zien dat in bijna alle experimenten de tweelaagse PCG methoden significant sneller convergeren dan de originele PCG methode, waardoor de berekeningen voor twee-fase stromingen met bellen efficiënter uitgevoerd kunnen worden.

Acknowledgements

After four years of intensive scientific research, this doctoral thesis is the absolute crowning glory. These words of thanks are, in my opinion, a crucial part of the thesis, as this is the only way to let people who have contributed directly and indirectly come to the front and to thank them in a deserving way. Only with the help and support of these very talented people has my PhD research succeeded and become the finished product that lies in front of you now.

This PhD research is partially sponsored by the BSIK/BRICKS project, which is a six-year scientific research program that is an initiative of the Dutch government. I am thankful to all of the people behind this project for their financial support. In addition, I would like to thank Delft University of Technology for sponsoring a large part of the thesis printing cost.

The bulk of my gratitude goes out to my advisor, Prof. Kees Vuik. During my Applied Mathematics studies, Kees played a prominent role by acting as a motivating and enthusiastic supervisor for various subjects and the MSc project. I am very grateful to him for consequently offering me a position of PhD student. I enjoyed the past four years, in which he functioned as a terrific supervisor, colleague and companion. I have learned a considerable amount from his knowledge, his constructive criticism, and the numerous discussions we had. I am grateful to Kees for the fact that he gave me space and freedom to do my doctoral research, to attend courses, to start cooperations at home and abroad, and to travel the world. It has been a privilege to attend several conferences together. In spite of his busy schedule, he has always been there, as well as given his full 100 per cent interest in my success. I owe Kees for the fact that he has always taken the time and trouble read through my paperwork, exchange mathematical ideas, help get bugs out of program codes, answer my many questions and more. His enthusiasm, patience and obliging manner led to various publications and a higher standard for this thesis. Furthermore, he has inspired and motivated me in both the easy and difficult times, which ensured that I became a much more complete scientist.

I have had a tremendous time with the renowned and prominent numerical analysis group under the guidance of Prof. Piet Wesseling, and subsequently Prof. Kees Vuik. I feel honored to have been part of this group of talented colleagues, namely Fons Daalderop, Duncan van der Heul, Martin van Gijzen, Jos van Kan, Domenico Lahaye,

Prof. Kees Oosterlee, Jennifer Ryan, Guus Segal, Peter Sonneveld, Fred Vermolen, and the many MSc, PhD and postgraduate students of whom the list is too long to recount. I thank you all for the pleasant atmosphere in the department, the many valuable conversations, the hilarious coffee breaks, and, above all, the various amusing outings and dinners outside working hours. I wish our numerical analysis department the very best in the future.

I would like to name Diana Droog, our secretary of the numerical analysis group, separately. She is a true professional and extraordinarily good at her job. These past few years, she has constantly helped me, and assisted me in various ancillary matters, which ensured that I could focus fully on my research instead of on these matters of lesser importance. In addition, she is a fantastic woman who has always brightened up our department. I want to take this opportunity to thank her for all her work, support, and cheerful conversations.

During the whole of my studies and PhD work, I have had the fortune and the privilege to always be able to count on two buddies, namely Sander van Veldhuizen and Jelle Hijmissen. In the past nine years, I have had much fun with and support from them, both on an academic and a social level. Therefore, it is more than self-evident to me to ask them to act as my paranymphs. I want to thank them up front for their assistance and support at the time I will be upholding my doctoral degree.

I affectionately thank the members of the doctoral examination committee for reading through this doctoral thesis as well as providing valuable suggestions and remarks. I realize full well it is not easy to read this bulky work in such a short period of time. Therefore, my utmost respect goes out to them. Moreover, I would like to thank Scott MacLachlan for providing various parts of the thesis with critical comments. Furthermore, I have received useful tips and valuable help concerning these acknowledgements and the propositions belonging to this dissertation from many people, especially from Merel Keijzer, Jennifer Ryan and Fred Vermolen. I am indebted to Kar Fee, who has electronically developed a sketch of the cover of this thesis and who is prepared to act as photographer during the PhD ceremony. Finally, I thank Olof Borgwit for designing the final version of the cover of the dissertation.

In the past four years, I have had the privilege to spend two working visits at TU Berlin in Prof. Reinhard Nabben's group. I am especially indebted to Reinhard for offering me these opportunities. He is a very special and enthusiastic scientist I have learned much from and with whom I have had the honor of working on several papers together. I thank him for the many pleasant conversations and for all the time he invested in me. In addition, Reinhard was a fantastic host during my two visits to Berlin. He made sure that I felt at home in both this metropolis and his department, where his group became a real family to me. For this, I have Veronica Twilling (secretary), Yogi Erlangga, Christian Mense, Elisabeth Ludwig and Sadegh Jokar to thank as well. They have shown me that conducting research is also for a great part connected to friendship and having fun.

I tremendously enjoyed collaborating with Scott MacLachlan. Our numerous discussions and e-mail correspondence have been very useful and productive, resulting in

several publications. Thanks to Scott, I have learned about the field of multigrid as well as the glitches in the mathematical theory in this area. In addition, he has helped me substantially with various papers by providing me with critical commentary on both the contents and the use of English. Moreover, he has contributed significantly to Chapters 7, 9 and 10 of this dissertation, for which I express my sincere gratitude to him.

My former roommate and colleague Sander van der Pijl was of great value and a tower of strength to me in the starting phase of my PhD research. He familiarized me with the world of bubbly flows and the current methods with supplementary program codes. Thanks to Sander, I had a solid base to advance my work in the specific research I was doing. Above all, he was a very social and amicable roommate with a dry wit and a fine technique in slime-volley.

During the last years, I had the pleasure of sharing my office with the many MSc and PhD students. Apart from Sander van der Pijl, Fang Fang was the one with whom I delighted in spending most of the time with in the office. I render my thanks for every nice conversation, her helpfulness and patience over the past three years.

It has been a great virtue to work together with several other professionals I will name shortly. I would like to thank Prof. Bendiks Jan Boersma and Emil Coyajee for their useful help and our interesting discussions. Furthermore, I thank them for their clarifications and for making available their parallel program code, the Tecplot software package, as well as for using their computer cluster for calculations. Moreover, I render my thanks to Tijmen Collignon, Peter Lucas, Allan Gersborg-Hansen (DTU, Denmark), Christophe Vandekerckhove (KU Leuven, Belgium) and Auke Ditzel (MARIN, Wageningen) for the short but effective collaborations in regard to the employment of two-level PCG methods on various applications relevant to them.

I have had the privilege to work with terrific people on various publications. For this, I not only take the opportunity to thank Prof. Kees Vuik, Prof. Reinhard Nabben, and Scott MacLachlan, but I also thank Yogi Erlangga and Elisabeth Ludwig. Additionally, I thank Prof. Piet Wesseling for his many good suggestions and corrections in regard to a number of these papers. I also enjoyed the various pleasant talks Piet and I had during both the time of my graduation and PhD. Lastly, I thank the editors and anonymous referees who, with their suggestions and keen observations, have considerably boosted the standard of the papers, and consequently of this dissertation.

I owe much gratitude to system administrators Eef Hartman and particularly Kees Lemmens. They have always been there for me when computer or network issues arose and when I had pressing questions concerning computers and program codes. In addition, they have taught me the fundamentals of Linux and many software packages.

In the time of my PhD, I had the opportunity to visit several conferences in The Netherlands (of which the Woudschoten conferences were the most splendid) and abroad (such as in China, Greece, Italy, Poland and USA). I have had the privilege to meet many inspiring people during these trips. I specifically want to mention and thank the jury of Burgersdag 2008 (Delft) and the IMACS 2008 conference (Lille, France) for appreciating my work and awarding the best poster and best student paper prize respectively. Namely, I give my thanks to Prof. Robert Beauwens, who handed me the best student paper prize. In doing so, he motivated me to continue my PhD research with much enthusiasm and passion.

The very best foreign adventure has without a doubt been my memorable visit to the group of Prof. Nick Trefethen in Oxford. This was on the invitation of Prof. Gene Golub, who offered me the opportunity to broaden my horizons in the world of numerical mathematics. Gene was a brilliant and perfect numerical mathematician, and, perhaps, the greatest of the past decades. Therefore, he has been my great idol during the time of my doctoral research. It was a privilege to spend a week with him in Oxford, during which time I discovered that Gene is wonderful as a person as well. Despite his boundless fame and renown, he is one of the most self-effacing, humorous, involved and congenial people I have had the privilege to meet during the last few years. The way Gene associated with me was the same as he did with prominent scientists. He was genuinely interested in me, as a scientist but certainly also as a person. The way he motivated and stimulated people is truly brilliant. Furthermore, Gene was an excellent host, who created opportunities for fun activities both within and outside of working hours. He also showed me that the world of numerical mathematics is wider than simple mathematics, as demonstrated by our visits to musicals and to his friends in London. Gene's sudden death at the end of 2007 has not surprisingly been a heavy blow and difficult to grasp. The very greatest numerical mathematician, and above all an amazing person, has evidently left us. Gene, you have occupied a special place in my heart and shown me the beauty of numerical mathematics and life. Despite not always having spoken as much about the research, your energy and enthusiasm have, to a great extent, led to the realization of this thesis. In memory of Gene, we organized the Gene Golub DCSE symposium in Delft within the framework of the worldwide 'Gene-around-the-world' project. With this, I also thank fellow organizers Diana Droog, Martin van Gijzen, Kees Vuik, and Marielba Rojas, who made sure this symposium became a great success.

As my research allowed me to lose myself in it, so did the education I was allowed to provide to students and the courses I could attend in order to broaden my experience. For the various educational tasks, I give thanks to the expertise and help of Henric Corstens, Fons Daalderop, Martin van Gijzen, Kees Vuik and Peter Wilders. For the courses, I specifically thank Merel Keijzer and Evelyn van de Veen for their efforts and extraordinary classes, which have brought my English to a significantly higher level.

The daily lunch break, which was mostly in the aula, was always a cheerful affair and the moment we could settle down every day at the office. For this, I not only openly thank my colleagues from the numerical analysis group, but also the ones from the fifth floor, in particular Jelle Hijmissen and Mirjam ter Brake. Furthermore, I thank the various colleagues of other departments for their conversations outside of the lunch breaks. Pertaining to this, I primarily think of the enjoyable annual DIAM outings, at which colleagues of the entire mathematics department could get to know each other better in an informal way.

In the time of my doctoral research, I delighted in participating at the PhDays, the annual weekend event where PhD students in numerical mathematics from The Netherlands and Flanders assemble to change thoughts and have a lot of fun. After my participation at the successful PhDays 2006 in Bérismenil (Belgium), I had the pleasure to organize both PhDays 2007 in Baarschot and PhDays 2008 in De Haan (Belgium). For this, I humbly thank Arthur van Dam, Yves Frederix, Liesbeth Vanherpe and Sander van Veldhuizen for co-organizing PhDays 2007, and Tijmen Collignon, Katrijn Frederix, Ricardo da Silva, Maria Ugryumova, Joris Vanbiervliet and once more Liesbeth Vanherpe for co-organizing PhDays 2008. It was an honor and very delightful to organize these latest two successful editions.

In conclusion, I would like to thank my dearest friends, acquaintances and family for their support and friendship over the past few years. I would like to specifically thank my parents Lisa and Cheung, who have believed in me unconditionally and have invested in me from my childhood on. Their inexhaustible love and support have given me the energy and strength to complete my PhD after years of effort. I am very proud of them, which makes it unsurprising that I dedicate this thesis to my parents. Additionally, I thank my sister Lora for her support and faith during my entire life.

Last but not least, I owe many thanks to my beloved Siu Mei, who has been my greatest support and anchor over the years. She has constantly encouraged me to continue my doctoral research. Siu Mei has always been there for me and has stood by me the entire time. I have only been able to write this dissertation with her patience, cheerfulness and love. I sincerely thank Siu Mei for this. She is definitely the best thing that has ever happened to me.

Jok Tang
Delft, July 2008

Contents

Summary iv

Samenvatting vii

Acknowledgements ix

1 Introduction 1

1.1 Background . . . 1
1.2 Two-level Preconditioned Conjugate Gradient Methods . . . 3
1.3 Bubbly Flow Problems . . . 3
1.4 Scope of the Thesis . . . 6
1.5 Outline of the Thesis . . . 6
1.6 Notation . . . 9

2 Iterative Methods 11

2.1 Introduction . . . 11
2.2 Basic Iterative Methods . . . 11
2.3 Conjugate Gradient Method . . . 13
2.4 Preconditioned Conjugate Gradient Method . . . 15
2.5 Further Considerations . . . 19
  2.5.1 Preconditioning . . . 19
  2.5.2 Starting Vectors and Termination Criteria . . . 21
2.6 Application to Bubbly Flows . . . 22
2.7 Concluding Remarks . . . 23

3 Deflation Method 25

3.1 Introduction . . . 25
3.2 Preliminaries . . . 25
3.3 Deflated CG Method . . . 28
3.4 Deflated PCG Method . . . 30
3.5 Properties of the Deflation Method . . . 33
  3.5.1 Results for an Eigenvector Deflation Subspace . . . 33

3.5.2 Results for an Arbitrary Deflation Subspace 35
3.5.3 Termination Criteria 38
3.6 Application to Bubbly Flow Problems 39
3.6.1 Results of Numerical Experiments 40
3.7 Concluding Remarks 41

4 Selection of Deflation Vectors 43
4.1 Introduction 43
4.2 Choices of Deflation Vectors 44
4.2.1 Approximated Eigenvector Deflation 44
4.2.2 Recycling Deflation 45
4.2.3 Subdomain Deflation 46
4.2.4 Multigrid and Multilevel Deflation Vectors 48
4.2.5 Discussion of Different Approaches 48
4.3 Application to Bubbly Flows 49
4.3.1 Preliminaries 49
4.3.2 Inexact Eigenvector Deflation 51
4.3.3 Level-Set Deflation Vectors 55
4.3.4 Subdomain Deflation 56
4.3.5 Level-Set-Subdomain Deflation 58
4.3.6 Numerical Experiments 61
4.3.7 Analysis of Small Eigenvalues 64
4.4 Concluding Remarks 67

5 Subdomain Deflation applied to Singular Matrices 69
5.1 Introduction 69
5.2 Preliminaries 71
5.3 Deflation Variants 74
5.4 Theoretical Comparison of Deflation Variants 75
5.4.1 On the Connection of the Singular and Invertible Matrix . . . 76
5.4.2 Comparison of the Deflated Singular and Invertible Matrix . . 77
5.4.3 Comparison of the Preconditioned Deflated Singular and Invertible Matrix 80
5.4.4 Comparison of the Preconditioned Deflated Singular Matrices 82
5.5 Application to Bubbly Flows 83
5.5.1 Results of ICCG and DICCG with Variant 5.2 83
5.5.2 Results of the Comparison between Variants 5.1 and 5.2 . . . 84
5.6 Concluding Remarks 86

6 Comparison of Two-Level PCG Methods – Part I 87
6.1 Introduction 87
6.2 Two-Level PCG Methods 90
6.2.1 Background of the Matrices in Domain Decomposition, Multigrid and Deflation 91

6.2.2 General Linear Systems 91
6.2.3 Definition of the Two-Level PCG Methods 92
6.2.4 Aspects of Two-Level PCG Methods 96
6.3 Theoretical Comparison 99
6.3.1 Spectral Analysis of the Methods 99
6.3.2 Equivalences between the Methods 103
6.4 Numerical Comparison 105
6.4.1 Setup of the Experiments 105
6.4.2 Experiment using Standard Parameters 106
6.4.3 Experiment using Inaccurate Galerkin Solves 106
6.4.4 Experiment using Severe Termination Tolerances 108
6.4.5 Experiment using Perturbed Starting Vectors 112
6.4.6 Further Discussion 114
6.5 Concluding Remarks 115

7 Comparison of Two-Level PCG Methods – Part II 117
7.1 Introduction 117
7.2 Two-Level PCG Methods 118
7.3 Spectral Properties of MG 120
7.3.1 Unit Eigenvalues of the MG-Preconditioned Matrix 120
7.3.2 Positive Definiteness of the MG preconditioner 122
7.4 Comparison of a Special Case of MG and DEF 125
7.5 Effect of Relaxation Parameters 126
7.5.1 Analysis of Scaling Relaxation 127
7.5.2 Optimal Choice of α 128
7.6 Symmetrizing the Smoother 130
7.7 Numerical Experiments 132
7.7.1 1-D Poisson-like Problem 132
7.7.2 2-D Bubbly Flow Problem 134
7.8 Concluding Remarks 135

8 Efficiency and Implementation of the Deflation Method 137
8.1 Introduction 137
8.2 Computations with the Deflation Matrix 139
8.2.1 Construction of AZ 139
8.2.2 Construction of E 139
8.2.3 Calculation of Py and $P^T y$ 140
8.3 Efficient Solution of Galerkin Systems 140
8.3.1 Galerkin Systems within DICCG1 141
8.3.2 Galerkin Systems within DICCG2 141
8.3.3 Comparison of Galerkin Matrices 144
8.3.4 Deflation Properties for a Singular Galerkin Matrix 144
8.4 Stabilization of the Deflation Method 145
8.5 Numerical Experiments 147

8.5.1 Results for the Deflation Method with Efficient Implementation 147
8.5.2 Comparison of DICCG1 and DICCG2 150
8.5.3 Comparison of DICCG2 and ADICCG2 151
8.6 Concluding Remarks 154

9 Comparison of Deflation and Multigrid with Typical Parameters 157
9.1 Introduction 157
9.2 Numerical Methods 159
9.2.1 Deflation Approach 159
9.2.2 Multigrid Approaches 160
9.3 Implementation and Computational Cost 166
9.3.1 Cost of Deflation 166
9.3.2 Cost of Multigrid 167
9.3.3 Singularity of Coefficient Matrix 169
9.3.4 Parallelization 169
9.3.5 Implementation 169
9.4 Numerical Experiments 170
9.4.1 Varying Density Contrasts 170
9.4.2 Varying Bubbly Radii 171
9.4.3 Varying Number of Bubbles 172
9.4.4 Varying Number of Grid Points 173
9.4.5 Difficult Test Problem 174
9.5 Concluding Remarks 175

10 Bubbly Flow Simulations 177
10.1 Introduction 177
10.2 Mathematical Model of the Bubbly Flow 177
10.3 Bubbly Flow Simulations 179
10.3.1 Rising Air Bubble in Water 180
10.3.2 Falling Water Droplet in Air 182
10.3.3 Two Rising and Merging Air Bubbles in Water 183
10.4 Concluding Remarks 185

11 Conclusions 187
11.1 Conclusions 187
11.2 Future Research 189

A Basic Theoretical Results 191

B Determination of Bubbles from the Level-Set Function 199

C More Insights into Deflation applied to Singular Coefficient Matrices 203
C.1 Theoretical Results 203
C.2 Proof of Theorem 5.5 205

D Efficient Implementation of Deflation Operations 209

- D.1 Efficient Construction of S_{AZ} and S_E in 2-D . . . 209
  - D.1.1 Number of Nonzero Entries in AZ . . . 210
  - D.1.2 Treatment of the Different Cases . . . 211
  - D.1.3 Construction of S_{AZ} . . . 212
  - D.1.4 Construction of S_E . . . 212
- D.2 Efficient Construction of S_{AZ} and S_E in 3-D . . . 213
  - D.2.1 Number of Nonzero Entries in AZ . . . 213
  - D.2.2 Matrix S_{AZ} for Eight Blocks . . . 214
  - D.2.3 Matrix S_{AZ} for 27 Blocks . . . 214
  - D.2.4 Matrix S_{AZ} with Variable Number of Blocks . . . 216
  - D.2.5 Construction of S_E . . . 216

E Flop Counts for the Deflation Method 217

- E.1 Deflation Operations . . . 218
- E.2 ICCG, DICCG1 and DICCG2 . . . 220

F Parallel Version of the Deflation Method 223

- F.1 Traditional Parallel Preconditioners . . . 224
- F.2 Parallel Deflation . . . 224

G Two-Level PCG Methods applied to Porous-Media Flows 227

- G.1 Problem Setting . . . 227
- G.2 Experiment using Standard Parameters . . . 228
- G.3 Experiment using Inaccurate Coarse Solves . . . 230
- G.4 Experiment using Severe Termination Tolerances . . . 231
- G.5 Experiment using Perturbed Starting Vectors . . . 234

H DICCG Variants applied to Bubbly Flow Simulations 237

- H.1 Simulation 1: Rising Air Bubble in Water . . . 237
- H.2 Simulation 2: Falling Water Droplet in Air . . . 238

I Comparison of Deflation and Multigrid for a Special Case 243

Bibliography 245

List of Publications 261

Curriculum Vitae 265

List of Figures

1.1 A droplet splash: an example of a two-phase bubbly flow problem. . . 4
1.2 Geometry of some stationary bubbly flows considered in this thesis ($m =$ number of bubbles, $s =$ radius of the bubbles). 4
1.3 A two-phase bubbly flow with the phases air and water. 5

3.1 Deflation subdomains with $k = 3$, which are chosen independently of the density geometry of the bubbly flow. 40
3.2 Norm of the relative residuals during the iterations of ICCG and DICCG$-k$. 42

4.1 Representation of the 2-D subdomains with $k = 3$ in a square domain, Ω, consisting of $n = 64^2$ grid points. 47
4.2 A 2-D example of a bubbly flow problem with $m = 5$ and two different situations for the perturbations, $\{\delta_i\}$. 54
4.3 A 2-D bubbly flow problem with $m = 5$, which illustrates the level-set, subdomain and level-set-subdomain deflation technique. 60
4.4 Eigenvalues of $M^{-1}A$ and $M^{-1}P_{W_S}A$ for S-DICCG, applied to the Poisson problem with $\epsilon = 1$ and $n = 16^2$. 62
4.5 Eigenvalues of both $M^{-1}A$ and $M^{-1}P_{W_L}A$ corresponding to L-DICCG-4, for the Poisson problem with $\epsilon = 10^6$ and $n = 16^2$. 65
4.6 Eigenvalues of $M^{-1}A$ and $M^{-1}P_{W_S}A$ corresponding to S-DICCG, for the Poisson problem with $\epsilon = 10^6$ and $n = 16^2$. 66
4.7 Eigenvalues of both $M^{-1}A$ and $M^{-1}P_{W_{LS}}A$ corresponding to LS-DICCG, for the Poisson problem with $\varepsilon = 10^6$ and $n = 16^2$. 67

5.1 Residuals of ICCG, DICCG-2^3 and DICCG-3^3 with Variant 5.2, for the test case with $n = 32^3$ and $\sigma = 10^{-3}$. 85

6.1 Relative errors during the iterative process, for the bubbly flow problem with $n = 64^2$, and ‘standard’ parameters. 107
6.2 Relative errors during the iterative process, for the bubbly flow problem with parameters $n = 64^2$ and $k = 8^2$, and a perturbed Galerkin matrix inverse, $\widetilde{E}^{-1}$. 109

6.3 Relative errors during the iterative process for the bubbly flow problem with parameters $n = 64^2$, $k = 8^2$, and various termination criterion. . 111
6.4 Relative errors during the iterative process for the bubbly flow problem with $n = 64^2$, $k = 8^2$, and perturbed starting vectors. 113

7.1 Regions where $\kappa_{MG} < \kappa_{DEF}$ (Regions A_1 and A_2) and $\kappa_{DEF} < \kappa_{MG}$ (Regions B_1 and B_2), for arbitrary $M^{-1} = \bar{M}^{-1}$, when Z consists of eigenvectors of $M^{-1}A$. The two condition numbers are equal along the dotted and dotted-dashed lines. 127
7.2 Eigenvalues associated with DEF and MG for the test cases with $k = 20$ as presented in Table 7.3. 134

8.1 Visualization of the results for the test problem with $m = 27$. 149
8.2 Results for the test problem with $m = 27$ for varying grid sizes. ICCG and DICCG1 with both $\psi = 5$ and $\psi = 10$ are presented. 149
8.3 Results for the test problem with $m = 27$ for varying density contrast, ϵ. 150
8.4 CPU time of DICCG1 and DICCG2 for the test problem with $m = 27$ and various grid sizes. 151
8.5 Convergence of the residuals during an inner solve at one iteration of ADICCG2-25^3 (Test Problem 2). The plots are similar for the other outer iterations of the same test case, since one applies the inaccurate solves to the Galerkin matrix, E. 153
8.6 Convergence of the residuals of the outer iterations from DICCG2-25^3 and ADICCG2-25^3 (Test Problem 2). 154
8.7 Residual plots of the outer iterations from ADICCG2-25^3 (Test Problem 1). 155

10.1 Evolution of a rising bubble in water. Parameters l and t denote the time step and actual time, respectively. 181
10.2 Results for ICCG, DICCG-20^3 and BoxMG-CG for the pressure solves during the real-life simulation with a rising air bubble in water. 181
10.3 Evolution of a falling droplet in air. 182
10.4 Results for ICCG, DICCG-20^3 and BoxMG-CG for the pressure solves during the real-life simulation of a falling water droplet in air. 183
10.5 Evolution of two rising air bubbles in water. 184
10.6 Results for DICCG-25^3 and BoxMG-CG for the pressure solve during the real-life simulation with two rising air bubbles in water. ICCG is omitted in these results, because it is not competitive with the other two methods. 184

B.1 A 2-D bubbly flow problem with $m = 3$ showing the application of Algorithm 9. The numbers given in the plots are the corresponding nonzero entries of vector f. 201

D.1 Domain Ω divided into nine subdomains ($k = 9$), so that each subdomain corresponds to exactly one group. 210
D.2 Cases of grid points involved in the groups of S_{AZ}. 210
D.3 Cases of grid points involved in $E := Z^T AZ$, denoted by E1, E2 and E3, whose values in AZ contribute to e_1, e_2 and e_3, respectively. . . . 213
D.4 Treatment of Block 1. 214

G.1 Geometry of the projection vectors and piecewise-constant coefficient in the porous-media flow. 228
G.2 Relative errors during the iterative process, for the porous-media problem with $n = 55^2$, $k = 7$, and 'standard' parameters. 230
G.3 Relative errors during the iterative process for the porous-media problem with $n = 55^2, k = 7$ and $\widetilde{E}^{-1}$, where a perturbation $\psi = 10^{-8}$ is taken. 232
G.4 Relative errors during the iterative process for the porous-media problem with $n = 55^2, k = 7$, and termination tolerance $\delta = 10^{-16}$. 233
G.5 Relative errors in the $A-$norm during the iterative process for the porous-media problem with $n = 55^2, k = 7^2$, and perturbed starting vectors with $\gamma = 10^{-5}$. The plot of the relative errors in the $2-$norm is omitted, since the two plots are approximately the same. 235

H.1 Evolution of the rising bubble in water in the first 250 time steps. . . 238
H.2 Results for ICCG, DICCG1-10^3 and DICCG2-20^3 for the simulation with a rising air bubble in water. 239
H.3 Evolution of the falling droplet in air in the first 250 time steps. . . . 240
H.4 Results for ICCG, DICCG1-10^3 and DICCG2-20^3 for the simulation with a falling water droplet in air. 241

I.1 Function $\lambda_i(2 - \lambda_i)$ for $\lambda_i \in [0, 2]$. 244

List of Tables

1.1 Notation for standard matrices and vectors where $\alpha, \beta, \gamma \in \mathbb{N}$. 9

2.1 Results for ICCG applied to 2-D bubbly flow problems. '# It' means the number of required iterations, and 'CPU' is the corresponding computational time in seconds. 22

2.2 Results for ICCG applied to 3-D bubbly flow problems. 23

3.1 Number of iterations for ICCG and DICCG$-k$ for various number of bubbles, m, and deflation vectors, k. 41

4.1 Results for the Poisson problem with $\epsilon = 1$ and $n = 16^2$. '# It' means the number of required iterations for convergence. 61

4.2 Results for the Poisson problem with $m = 5$, $\epsilon = 10^6$, and varying grid sizes, n. '# It' means the number of required iterations, and 'CPU' is the corresponding computational time in seconds. 62

4.3 Results for the Poisson problem with $m = 5$, $n = 64^2$, and varying density contrast, ϵ. 63

4.4 Results for the Poisson problem with $\epsilon = 10^6$, $n = 64^2$, and varying number of bubbles, m. 64

5.1 Corresponding matrices of the proposed deflation variants. 75

5.2 Number of iterations for ICCG and DICCG (Variant 5.2) to solve the linear system $\bar{A}x = b$ with invertible $\bar{A}$, for the test case with $m = 2^3$, $\epsilon = 10^3$, and $s = 0.05$. 84

5.3 Number of iterations for ICCG and DICCG (Variant 5.2) to solve $\bar{A}x = b$, for the test case with $m = 3^3$, $\epsilon = 10^3$, and $s = 0.05$. 85

5.4 Number of iterations for DICCG (both Variant 5.1 and Variant 5.2) to solve $Ax = b$ (with a singular A) and $\bar{A}x = b$ (with an invertible $\bar{A}$) for $m = 2^3$. 85

6.1 List of methods that are compared in this chapter. The operator of each method can be interpreted as the preconditioner $\mathcal{P}$, given in (6.1) with $\mathcal{A} = A$. Where possible, references to the methods and their implementations are presented in the last column. 96

6.2 Choices of parameters for each method, used in the generalized two-level PCG method as given in Algorithm 7. 97

6.3 Extra computational cost per iteration of the two-level PCG methods compared to PREC. IP = inner products, MVM = matrix-vector multiplications, VU = vector updates and GSS = Galerkin system solves. Note that IP holds for sparse Z and MVM holds for dense Z. 99

6.4 Number of required iterations for convergence and the 2−norm of the relative errors of all methods, for the bubbly flow problem with $n = 64^2$, and 'standard' parameters. 'NC' means no convergence within 250 iterations. 106

6.5 Computational cost within the iterations in terms of number of inner products ('IP'), vector updates ('VU'), Galerkin system solves ('GSS'), and preconditioning step with M^{-1} ('PR'), for the bubbly flow problem with $n = 64^2$, $k = 8^2$, and 'standard' parameters. 108

6.6 Number of required iterations for convergence and the 2−norm of the relative errors of all methods for the bubbly flow problem with parameters $n = 64^2$ and $k = 8^2$, and a perturbed Galerkin matrix inverse, $\widetilde{E}^{-1}$, is used with varying perturbation ψ. 'NC' means no convergence within 250 iterations. 110

6.7 Number of required iterations for convergence and the 2−norm of the relative errors of all methods, for the bubbly flow problem with parameters $n = 64^2$ and $k = 8^2$. Various termination tolerances, δ, are tested. 110

6.8 Number of required iterations for convergence and the 2−norm of the relative errors of some methods, for the bubbly flow problem with $n = 64^2$, $k = 8^2$, and perturbed starting vectors. An asterisk (*) means that an extra uniqueness step is applied in that test case. 114

7.1 List of two-level PCG methods that are compared in this chapter. . . 120

7.2 Test cases corresponding to different regions as presented in Figure 7.1. 132

7.3 Results of the experiment with test cases as presented for the Poisson-like problem in Table 7.2. The results are presented in terms of number of iterations, # It., and condition number, κ. 133

7.4 Number of required iterations for convergence and the 2−norm of the relative errors of 2L-PCG methods, for the bubbly flow problem with $n = 64^2$ and $\bar{M}^{-1} = M^{-1}$. PREC requires 137 iterations and leads to a relative error of 4.6×10^{-7}. 135

7.5 Number of required iterations for convergence and the 2−norm of the relative errors of 2L-PCG methods, for the bubbly flow problem with $n = 64^2$ and $M^{-1} = \bar{M}^{-1} + \bar{M}^{-T} - \bar{M}^{-T} A \bar{M}^{-1}$. PREC requires 137 iterations and leads to a relative error of 4.6×10^{-7}. 135

8.1 Corresponding matrices of the proposed deflation variants in Chapter 5. 140

8.2 Convergence results for ICCG and DICCG1 for all test problems with $n = 100^3$ and $\epsilon = 10^3$. '# It' means the number of required iterations for convergence, and 'CPU' means the total computational time in seconds. 148

8.3 Results for DICCG2 and ADICCG2 to solve $Ax = b$ with $n = 100^3$, corresponding to Test Problem 1. ICCG requires 390 iterations and 37.0 seconds for Test Problem 1 and 543 iterations and 177.6 seconds for Test Problem 2. '# It' = number of iterations of the outer process, 'CPU' = the required computing time (in seconds) including the setup time of the methods, 'NC' = no convergence within 250 iterations. . 152

9.1 Convergence results for the experiment with $n = 64^3$, $m = 2^3$, $s = 0.05$, and varying the contrast, ϵ. 'CPU', '# It.' and 'RES' denote the total computing time, number of iterations or cycles, and the accuracy of the solution measured as the relative norm of the exact residuals, respectively. 171

9.2 Convergence results for the experiment with $n = 64^3$, $m = 2^3$, $\epsilon = 10^3$, and varying the radius of the bubbles, s. 172

9.3 Convergence results for the experiment with $n = 64^3$, $s = 0.05$, $\epsilon = 10^3$, and varying the number of bubbles, m. 173

9.4 Convergence results for the experiment with $m = 2^3$, $s = 0.05$, $\epsilon = 10^3$, and varying the total number of degrees of freedom, n. 174

9.5 Convergence results for the difficult test problem. The following parameters are kept constant: $n = 128^3$, $m = 3^3$, $s = 0.025$, $\epsilon = 10^5$, and $\delta = 10^{-8}$. 175

D.1 Explanation of the variables. 211

D.2 Cases involved in the blocks for eight subdomains in 3-D. 215

D.3 Cases involved in Blocks 1–9 for $k = 27$ in the 3-D case. 215

E.1 Results of flop counts for standard operations. 217

G.1 Number of required iterations for convergence of all proposed methods, for the porous-media problem with 'standard' parameters. The 2−norm of the relative error is approximately the same for all methods in each test case, and, hence, they are omitted in the table. 229

G.2 Total computational cost within the iterations in terms of number of inner products ('IP'), vector updates ('VU'), Galerkin system solves ('GSS'), preconditioning step with M^{-1} ('PR'), for the porous-media problem with $n = 55^2$, $k = 7$, and 'standard' parameters. 229
G.3 Number of required iterations for convergence and the 2−norm of the relative errors of all methods, for the porous-media problem with parameters $n = 55^2$ and $k = 7$. A perturbed Galerkin matrix inverse, $\widetilde{E}^{-1}$, is used with a varying perturbation, ψ. 231
G.4 Number of required iterations for convergence and the 2−norm of the relative errors of all methods, for the porous-media problem with parameters $n = 55^2$ and $k = 7$. Various termination tolerances, δ, are tested. 234
G.5 Number of required iterations for convergence and the 2−norm of the relative errors of some methods, for the porous-media problem with parameters $n = 55^2$, $k = 7$, and perturbed starting vectors. An asterisk (*) means that an extra uniqueness step is applied in that test case. . 234

List of Algorithms

1 Conjugate Gradient (CG) solving $Ax = b$ 14
2 Preconditioned CG (PCG) solving $Ax = b$ (Original Variant) 17
3 Preconditioned CG (PCG) solving $Ax = b$ (Practical Variant) 17
4 Deflated Conjugate Gradient (DCG) solving $Ax = b$ 30
5 Deflated PCG (DPCG) solving $Ax = b$ (Original Variant) 31
6 Deflated PCG (DPCG) solving $Ax = b$ (Practical Variant) 32
7 Generalized Two-Level PCG Method for solving $Ax = b$. 96
8 Computation of Py 138
9 Determination of bubbles from the level-set function in 2-D 200

Chapter 1

Introduction

1.1 Background

The focus of this thesis is on the numerical solution of the linear partial differential equations (PDEs) resulting from the mathematical modeling of physical systems and, in particular, bubbly flows. We assume that these PDEs have already been discretized in a sensible manner through the use of finite differences, finite volumes or finite elements. Our primary focus is on efficient solution of linear systems, of the form

$$Ax = b, \quad A \in \mathbb{R}^{n \times n}, \quad n \in \mathbb{N}, \tag{1.1}$$

that arise from such discretizations, where n is the number of degrees of freedom and is called the dimension of A. In Eq. (1.1), the coefficient matrix, A, is assumed to be real, symmetric, and positive semi-definite (SPSD), i.e.,

$$A = A^T, \quad y^T A y \geq 0 \ \forall y \in \mathbb{R}^n,$$

and has d zero eigenvalues with corresponding linearly independent eigenvectors. If $d > 0$, then A is singular. To guarantee that Eq. (1.1) is consistent, the right-hand side, b, is presumed to be in the range of A, i.e., $b \in \mathcal{R}(A)$ where $\mathcal{R}(A) := \{y \in \mathbb{R}^n : y = Aw \text{ for } w \in \mathbb{R}^n\}$. Thus, the next assumption holds throughout this thesis.

Assumption 1.1. *The coefficient matrix, A, is SPSD and has d zero eigenvalues. Moreover, the linear system (1.1) is consistent.*

The null space of A is defined as $\mathcal{N}(A) := \{w \in \mathbb{R}^n : Aw = \mathbf{0}_n\}$, where $\mathbf{0}_n$ is the all-zero vector with n entries. Then, $\mathcal{N}(A)$ is the orthogonal complement of the column space of A, i.e., $\mathcal{N}(A) = \mathcal{R}(A)^{\perp}$. As a consequence, the linear system (1.1) is only consistent if $b^T w = 0$ is satisfied for all $w \in \mathcal{N}(A)$.

Linear system (1.1) is typically large, sparse, and ill-conditioned. That means that current problems of interest involve millions of degrees of freedom, a fixed number of nonzero entries per row and column of A, and condition number of A, denoted as $\kappa(A)$, approaching infinity as problem size or coefficient ratio in the original PDEs increases, respectively. In this thesis, we denote by $\lambda_i(B)$ (or, shortly, λ_i) the i-th eigenvalue of an arbitrary symmetric matrix, $B \in \mathbb{R}^{n\times n}$, where the set $\{\lambda_i\}$ is always ordered increasingly (unless otherwise stated), i.e., $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. This set, $\{\lambda_i\}$, is called the spectrum of B and is denoted as $\sigma(B)$. If B is SPSD, then its (spectral) condition number is defined as the ratio of the largest and the smallest nonzero eigenvalues, i.e.,

$$\kappa(B) := \frac{\lambda_n}{\lambda_{d+1}}.$$

The linear system (1.1) can be solved using direct methods. Most of these solvers generally involve explicit factorization of (permutations of) A into a product of a lower and an upper triangular matrix. Important advantages of direct solvers are their robustness and general applicability. However, the bottleneck of direct solvers is that the matrix factor is often significantly denser than A. On the one hand, this might lead to an excessive amount of computations and, on the other hand, it might lead to insufficient memory to form and store matrix factors. Therefore, direct methods are typically prohibitively expensive and in some cases impossible, even with the best available computing power.

Instead of direct solvers, iterative methods are more attractive to use to find the solution of (1.1). In this case, both memory requirements and computing time can be reduced, especially if A is large and sparse. Moreover, these methods are mandatory for some numerical discretization methods, where A is not explicitly available. The term 'iterative method' refers to a wide range of techniques that use iterates, or successive approximations, to obtain more accurate solutions to a linear system at each iteration step. Krylov subspace iterative methods, especially the Conjugate Gradient (CG) method of Hestenes and Stiefel, are prominent iterative methods to solve (1.1). In these methods, the closest approximation to the solution of (1.1) is found in a subspace whose size is iteratively increased. The convergence of these methods depends highly on $\kappa(A)$, which again typically grows as the problem size increases. To avoid the increase in iterations, it is common practice to modify the Krylov subspace method in the hopes of reducing the difficulties in solving the given system. If this is applied to CG, then the resulting method is called the preconditioned Conjugate Gradient (PCG) method. In this case, (1.1) is multiplied by a preconditioner, M^{-1}, chosen to reduce the condition number of the iteration matrix from $\kappa(A)$ to $\kappa(M^{-\frac{1}{2}}AM^{-\frac{1}{2}})$, which is equivalent to $\kappa(M^{-1}A)$. The resulting preconditioned system that should be solved reads

$$M^{-1}Ax = M^{-1}b, \tag{1.2}$$

where M is assumed to be symmetric and positive definite (SPD), i.e.,

$$M = M^T, \quad y^T My > 0 \ \forall y \neq \mathbf{0}_n.$$

PCG is more effective than original CG for many problems of interest. When $M^{-1}y$ is easily computed for a given vector, y, the additional cost of the preconditioning in the Krylov iteration can certainly pay off, if it results in a more amenable spectrum of

$M^{-1}A$. That is, if $\kappa(M^{-1}A)$ is significantly less than $\kappa(A)$, or if the spectrum is more clustered than that of the original matrix, we can expect significantly fewer iterations to be needed. However, even with sophisticated preconditioners, such as preconditioners based on incomplete factorizations, $\kappa(M^{-1}A)$ might still become larger as the problem size or coefficient ratio in the original PDEs increases. In this case, PCG may suffer from slow convergence due to the presence of unfavorable eigenvalues in $\sigma(M^{-1}A)$.

1.2 Two-level Preconditioned Conjugate Gradient Methods

In addition to a traditional preconditioner, M^{-1}, a second kind of preconditioner can be incorporated to improve the conditioning of the coefficient matrix even further, so that the resulting approach effectively treats the effect of all unfavorable eigenvalues. This combined preconditioning is known as 'two-level preconditioning', and the resulting iterative method is called a 'two-level PCG (2L-PCG) method'. In this case, CG, in combination with a preconditioner based on a multigrid (MG) method or domain decomposition method (DDM), can be regarded as a 2L-PCG method, since most of these methods rely on preconditioning on two levels. These preconditioners have been known for a long time, dating back at least to the 1930s.

The main focus of this thesis is on the 2L-PCG method whose two-level preconditioner is based on a deflation technique. The resulting method is often called the deflation method and was introduced independently by Nicolaides and Dostal in the 1990s.

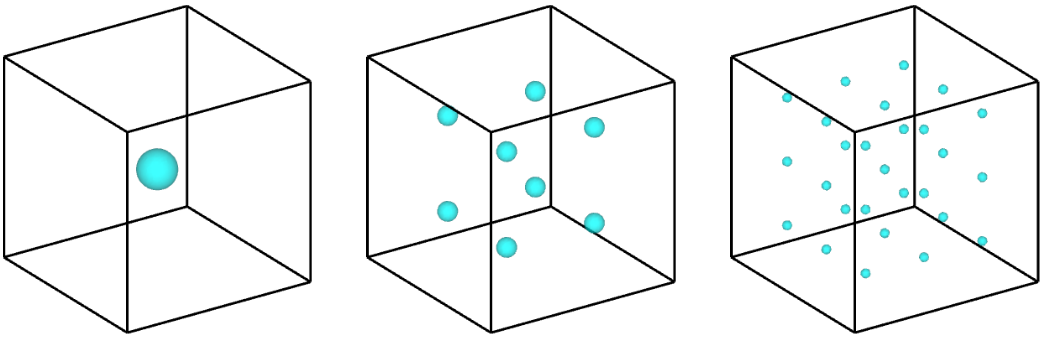
1.3 Bubbly Flow Problems

The main application of this thesis is two-phase bubbly flows, as in Figure 1.1. Computation of these flows is a very active research topic in computational fluid dynamics (CFD). Understanding the dynamics and interaction of bubbles and droplets in a large variety of processes in nature, engineering, and industry are crucial for economically and ecologically optimized design. Bubbly flows occur, for example, in chemical reactors, boiling, fuel injectors, coating, and volcanic eruptions.

Two-phase flows are complicated to simulate, because the geometry of the problem typically varies with time, and the fluids involved have very different material properties. A simple example is that of air bubbles in water, where the densities vary by a factor of about 800. In this thesis, we consider both stationary and time-dependent bubbly flows, where the computational domain is always a unit square or unit cube filled with a fluid to a certain height. The bubbles and droplets in the domain are always chosen such that they are located in a structured way and have equal radius, s, at the starting time. Typical 3-D test problems, considered in this thesis, are depicted in Figure 1.2. 2-D test problems are always based on sections of these 3-D domains. Throughout this thesis, lengths are typically given in centimeters (cm).

Mathematically, bubbly flows are modelled using the Navier-Stokes equations including boundary and interface conditions, which can be approximated numerically

Figure 1.1: A droplet splash: an example of a two-phase bubbly flow problem.



(a) $m = 1$ and $s = 0.1$. (b) $m = 8$ and $s = 0.05$. (c) $m = 27$ and $s = 0.025$.

Figure 1.2: Geometry of some stationary bubbly flows considered in this thesis (m = number of bubbles, s = radius of the bubbles).

using operator-splitting techniques. In these schemes, equations for the velocity and pressure are solved sequentially at each time step. In many popular operator-splitting methods, the pressure correction is formulated implicitly, requiring the solution of a linear system (1.1) at each time step. This system takes the form of a Poisson equation with discontinuous coefficients (also called the 'pressure(-correction) equation') and Neumann boundary conditions, i.e.,

$$\left\{ \begin{array}{rcll} -\nabla \cdot \left(\frac{1}{\rho(\mathbf{x})} \nabla p(\mathbf{x}) \right) & = & f(\mathbf{x}), & \mathbf{x} \in \Omega, \\ \frac{\partial}{\partial \mathbf{n}} p(\mathbf{x}) & = & g(\mathbf{x}), & \mathbf{x} \in \partial\Omega, \end{array} \right. \tag{1.3}$$

where $\Omega, p, \rho, \mathbf{x}$, and $\mathbf{n}$ denote the computational domain, pressure, density, spatial coordinates, and the unit normal vector to the boundary, $\partial\Omega$, respectively. Right-hand sides f and g follow explicitly from the operator-splitting method, where g is such that mass is conserved, leading to a singular but compatible linear system (1.1) [1].

We define n_x, n_y and n_z as the number of degrees of freedom in each spatial direction, so that $n = n_x n_y n_z$. In this thesis, we perform the computations on a uniform Cartesian grid with $n_x = n_y = n_z$. Furthermore, we consider two-phase bubbly

---

[1] For stationary bubbly flow problems, we take $f(\mathbf{x}) = 0$, so that the right-hand side of the linear system, b, only contains components from the boundary conditions. In addition, we take $g(\mathbf{x})$ to be

flows with, for example, air and water.

In this case, ρ is piecewise constant with a relatively large contrast:

$$\rho = \begin{cases} \rho_0 = 1, & \mathbf{x} \in \Lambda_0; \\ \rho_1 = \varepsilon, & \mathbf{x} \in \Lambda_1. \end{cases} \tag{1.4}$$

For flows with water and air, the density contrast, defined as $\epsilon := \frac{\rho_0}{\rho_1} = \varepsilon^{-1}$, is $\epsilon \approx 10^3$, see Figure 1.3. In this case, Λ_0 is water, the main fluid of the flow around the m air bubbles, and Λ_1 is the region inside the bubbles.

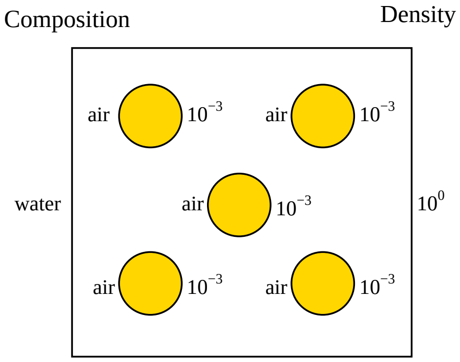


Figure 1.3: A two-phase bubbly flow with the phases air and water.

Solving linear system (1.1), that is a discretization of (1.3), within an operator-splitting approach has long been recognized as a computational bottleneck in fluid-flow simulation, since it typically consumes the bulk of the computing time. Whether finite-difference, finite-element, or finite-volume techniques are used to discretize (1.3), the resulting matrix is sparse, but with a bandwidth of $n_x n_y$, in lexicographical ordering. For a discretization on a cube with 100 degrees of freedom in each direction, this means that A is of dimension $n = 10^6$, with bandwidth $n_x n_y = 10^4$. It is well-known that direct solution techniques without reordering require a number of operations that scale as $n^{\frac{7}{3}}$ for a banded Cholesky decomposition, $\mathcal{O}(10^{14})$ operations in the example above. Thus, here, we consider the solution of (1.1) using iterative techniques. Standard PCG methods are not suitable, since they exhibit a strong sensitivity to the density contrast and grid size. There is a real need for two-level preconditioning in order to accelerate the convergence of the iterative process of PCG. Hence, we apply 2L-PCG methods to solve (1.1).

Next, define $\mathbf{1}_p$ as the all-one vector with p entries. Then, the following assumption holds in our bubbly flow problem, which follows implicitly from the above problem

---

constant at each boundary; we use the following boundary conditions in the 3-D case:

$$\begin{cases} \frac{\partial}{\partial \mathbf{n}} p(\mathbf{x})|_{x=0} &= -\frac{\partial}{\partial \mathbf{n}} p(\mathbf{x})|_{x=1} &= 1; \\ \frac{\partial}{\partial \mathbf{n}} p(\mathbf{x})|_{y=0} &= -\frac{\partial}{\partial \mathbf{n}} p(\mathbf{x})|_{y=1} &= -1; \\ \frac{\partial}{\partial \mathbf{n}} p(\mathbf{x})|_{z=0} &= -\frac{\partial}{\partial \mathbf{n}} p(\mathbf{x})|_{z=1} &= 1. \end{cases}$$

In a similar way, such boundary conditions are chosen in 2-D stationary problems.

setting.

Assumption 1.2. *In bubbly flow problems, we assume that A is a singular M-matrix, and the equations $A\mathbf{1}_n = \mathbf{0}_n$ and $b^T\mathbf{1}_n = 0$ are satisfied.*

A symmetric M-matrix is a square SPSD matrix whose off-diagonal entries are less than or equal to zero. According to [15], $d = 1$ holds for a singular M-matrix, A. In other words, we have $\operatorname{rank} A = n - 1$ and $\dim \mathcal{N}(A) = 1$, where $\operatorname{rank} B$ and $\dim \mathcal{B}$ denote the rank of matrix B and the dimension of subspace $\mathcal{B}$, respectively. Note that Assumption 1.1 follows immediately from Assumption 1.2, since (1.1) is always consistent. For $b = \mathbf{0}_n$, this is trivial, and, for $b \neq \mathbf{0}_n$, $b \perp \mathcal{N}(A) = \operatorname{span}\{\mathbf{1}_n\}$ resulting in the fact that $b \in \mathcal{R}(A)$. Therefore, although A is singular, (1.1) is always consistent and an infinite number of solutions exists. Due to the Neumann boundary conditions, the solution, x, of (1.1) is fixed up to a constant, i.e., if x_1 is a solution then $x_1 + c\mathbf{1}_n$ is also a solution of (1.1), where $c \in \mathbb{R}$ is any constant. This situation presents no real difficulty, since pressure is a relative variable, not an absolute one in the operator-splitting methods.

1.4 Scope of the Thesis

This thesis deals with acceleration of PCG using two-level preconditioning in order to solve linear systems with an SPSD coefficient matrix. The main 2L-PCG method is the deflation method. In the literature, much is known about applying the deflation method to linear systems with invertible coefficient matrices and to problems with fixed and known density fields. In this thesis, we generalize it to linear systems with singular coefficient matrices and to problems where the density field varies or cannot be described explicitly. Moreover, we investigate the efficient implementation of the deflation method and the further improvements of the method. We also compare the deflation method with other well-known 2L-PCG methods by considering both the abstract variants and their optimal variants with their typical parameters. Numerical experiments with bubbly flows are performed to illustrate the theoretical results.

Remark 1.1. *Many theoretical results presented in this thesis are generally applicable and are not restricted to applications of bubbly flows, although these bubbly flows are the main application of this thesis. Therefore, Assumption 1.2 is not demanded in the general discussion, but is only required when the general theoretical results are applied to bubbly flows. In addition, all results that require Assumption 1.2 can also be applied to other fields where this assumption is fulfilled.*

1.5 Outline of the Thesis

The outline of this thesis is as follows.

Chapter 2: Iterative Methods. This chapter is devoted to the introduction of iterative methods, especially the CG and PCG methods. In most introductory books, CG and PCG are derived and analyzed where A is assumed to be invertible, but we give the methods and their concise derivations for general SPSD coefficient matrices. Moreover, some properties of these methods are presented, which are not fully clear in the literature. Finally, the drawbacks of PCG are illustrated using numerical experiments with bubbly flows.

Chapter 3: Deflation Method. In order to improve the convergence of the standard PCG method, the deflation method and its preconditioned variant are introduced in Chapter 3. We give their derivation in detail and present some new theoretical properties. We show, both theoretically and numerically, that the deflation method is expected to be more effective than the original PCG method.

Chapter 4: Selection of Deflation Vectors. The success of the deflation method highly depends on the choice of the so-called 'deflation vectors'. Good approximations of eigenvectors associated with unfavorable eigenvalues are often chosen, which are usually dense and not straightforward to obtain. In addition, the density field is often not known explicitly in our bubbly flow application, which might lead to difficulties for approximating eigenvectors. We analyze this issue in more detail in Chapter 4 and provide some strategies to determine the best deflation vectors for bubbly flow problems. We come up with several suitable choices, whose utility is illustrated in numerical experiments.

Chapter 5: Subdomain Deflation applied to Singular Matrices. Theoretical results for the deflation method are well-known if it is applied to nonsingular coefficient matrices. The application of this method to singular coefficient matrices is more complicated and has not been widely considered in the literature. This issue is further investigated in Chapter 5. We show equivalences between deflation methods applied to singular and invertible coefficient matrices. We come up with several mathematically equivalent variants of the two-level preconditioner corresponding to the deflation method. Numerical experiments are used to show that these variants can be easily applied in practice.

Chapter 6: Comparison of Two-level PCG Methods – Part I. The main focus of this thesis is on the deflation method, whereas other attractive 2L-PCG methods are known in the literature. In Chapter 6, we compare the deflation method with some prominent 2L-PCG methods coming from the fields of deflation, DDM and MG. Both a theoretical and numerical comparison are performed using the abstract forms of these methods. We investigate their spectral properties, equivalences, effectiveness and robustness, and end up with a 2L-PCG method of our choice.

Chapter 7: Comparison of Two-Level PCG Methods – Part II. In Chapter 6, the 2L-PCG method based on the standard multigrid V(1,1)-cycle method is excluded in the comparison, since it has different spectral properties and requires a special theoretical treatment. Chapter 7 examines this method in more detail. We compare the 2L-PCG methods as discussed in Chapter 6, and show that it depends on the chosen parameters which 2L-PCG method is the most effective one.

Chapter 8: Efficiency and Implementation Issues of the Deflation Method. In the previous chapters, we have shown that the deflation method is expected to converge faster than PCG in terms of iteration counts. However, the deflation method needs to be implemented efficiently in order to obtain a fast method with respect to computing time as well. This issue is examined in Chapter 8, where we show how each step of the deflation algorithm can be best implemented for a class of problems. At each iteration of the deflation method, coarse linear systems should be solved, which is usually done by a direct method. If the number of deflation vectors is relatively large, we show that it is more attractive to use an iterative method, so that the resulting method is based on an inner-outer iteration process. This is further detailed and illustrated with numerical experiments in Chapter 8.

Chapter 9: Comparison of Deflation and Multigrid with Typical Parameters. In Chapter 6 and 7, the 2L-PCG methods have been compared in their abstract forms. In this case, the different parameters within these methods can be arbitrary, but are equal for each method, which allows us to perform a general comparison. The comparison can also be carried out with typical parameters in the methods. Each 2L-PCG method then takes its optimized set of parameters that is typical in the field where the method comes from. Chapter 9 is devoted to this comparison. The aim of this chapter is to show which optimized 2L-PCG method is currently the best one to apply for 3-D bubbly flow applications.

Chapter 10: Bubbly Flow Simulations. In the previous chapters, we have shown that 2L-PCG methods are beneficial to use for stationary bubbly flow problems. In Chapter 10, the exact mathematical model for the bubbly flows is formulated, so that real-life time-dependent experiments can be performed. We show that 2L-PCG methods reduce significantly the computations of bubbly flow simulations and are less sensitive to the density field compared with standard PCG methods.

Chapter 11: Conclusions. The main conclusions of the thesis and ideas for future research are presented in Chapter 11.

This thesis is based on the technical reports [87, 132, 134, 136, 137, 140, 141, 144], the proceeding papers [138, 139, 142, 147, 172], and, especially, the journal papers [85, 88, 133, 135, 143, 145, 146, 148]. It is written in such a way that the thesis itself and every chapter are self-contained as much as possible.

1.6 Notation

Throughout this thesis, we use the notation as given in Table 1.1.

Notation	Meaning
I	identity matrix with an appropriate dimension
$\mathbf{e}_{\alpha}^{(\gamma)}$	γ-th column of I with dimension α
$\mathbf{e}_{\alpha,\beta}^{(\gamma)}$	$\alpha \times \beta$ matrix with β identical columns $\mathbf{e}_{\alpha}^{(\gamma)}$
$\mathbf{1}_{\alpha,\beta}$	$\alpha \times \beta$ matrix whose entries are ones
$\mathbf{1}_{\alpha}$	column of $\mathbf{1}_{\alpha,\beta}$
$\mathbf{0}_{\alpha,\beta}$	$\alpha \times \beta$ matrix whose entries are zeros
$\mathbf{0}_{\alpha}$	column of $\mathbf{0}_{\alpha,\beta}$

Table 1.1: Notation for standard matrices and vectors where $\alpha, \beta, \gamma \in \mathbb{N}$.

Chapter 2

Iterative Methods

2.1 Introduction

Recall that the main focus of this thesis is on solving the linear system (see Eq. (1.1))

$$Ax = b, \quad A = [a_{ij}] \in \mathbb{R}^{n \times n}, \tag{2.1}$$

where A is a sparse and SPSD coefficient matrix. We aim at solving (2.1) using Krylov iterative methods, which are examined in this chapter.

We start this chapter by reviewing basic iterative methods. This is followed by presenting the Conjugate Gradient (CG) method, which is a well-known iterative method to solve (2.1). The rate at which CG and general iterative methods converge depends greatly on the spectrum of the coefficient matrix, A. Hence, these methods usually involve a second matrix that transforms A into one with a more favorable spectrum. The resulting method is then called the preconditioned Conjugate Gradient (PCG) method, and is described in Section 2.4. Some further considerations regarding preconditioning, starting vectors and termination criteria of the iterative procedure are discussed in Section 2.5. We conclude this chapter with the application of the solvers to bubbly flow problems in order to illustrate the performance of the PCG methods.

2.2 Basic Iterative Methods

Iterative methods generate a sequence of iterates, $\{x_j\}$, that approximate the exact solution, x. These methods essentially involve matrix A only in the context of matrix-vector multiplications. The starting point of these methods is considering a splitting of A of the form

$$A = M - N, \quad M, N \in \mathbb{R}^{n \times n}, \tag{2.2}$$

where M is assumed to be invertible. If the splitting (2.2) is substituted into Eq. (2.1), we obtain

$$Mx = b + Nx. \tag{2.3}$$

From Eq. (2.3), a basic iterative method can be constructed as follows:

$$Mx_{j+1} = b + Nx_j, \tag{2.4}$$

where the iterate, x_{j+1}, in the $(j+1)$-th step can be determined from the previous iterate, x_j. Eq. (2.4) can be rewritten as

$$x_{j+1} = x_j + M^{-1}r_j, \tag{2.5}$$

where the residual after the j-th iteration is defined as

$$r_j := b - Ax_j,$$

that is a measure of the difference of the iterative and the exact solution of (2.1). For the iteration (2.5) to be practical, it must be relatively easy to solve a linear system with M as the coefficient matrix. For example, $M = \operatorname{diag}(A)$ is used in Jacobi iterations, M consists of the lower-triangular part of A in Gauss-Seidel iterations, and a more complicated M is used in (symmetric) successive over-relaxation ((S)SOR) iterations, that can be derived from Gauss-Seidel iterations by introducing an extrapolation parameter. It has been recognized that these basic iterative methods are impractical, because they converge slowly, need good tuning of some parameters, or require strict conditions for convergence. More basic iteration methods and their analysis can be found in [8, 63, 70, 120, 167].

When the first iterations of (2.5) are developed, one obtains

$$\begin{cases} x_0; & & \\ x_1 & = & x_0 + M^{-1}r_0; \\ x_2 & = & x_0 + 2M^{-1}r_0 - M^{-1}AM^{-1}r_0; \\ x_3 & = & x_0 + 3M^{-1}r_0 - 3M^{-1}AM^{-1}r_0 + \left(M^{-1}A\right)^2 M^{-1}r_0; \\ & \vdots & \end{cases}$$

This yields

$$x_{j+1} \in x_0 + \operatorname{span}\left\{M^{-1}r_j, M^{-1}A(M^{-1}r_j), \ldots, (M^{-1}A)^{j-1}(M^{-1}r_j)\right\}.$$

Subspaces of the form

$$\mathcal{K}_j(A, r_0) := \operatorname{span}\left\{r_0, Ar_0, A^2r_0, \ldots, A^{j-1}r_0\right\}$$

are called Krylov subspaces with dimension j, belonging to A and r_0. Hence, the following holds for basic iterative methods:

$$x_{j+1} \in x_0 + \mathcal{K}_j(M^{-1}A, M^{-1}r_0). \tag{2.6}$$

These methods are also called Krylov(-subspace) methods. From Eq. (2.6), it follows that Krylov methods rely on finding M^{-1} (that is often called a ‘preconditioner’) and a basis for $\mathcal{K}_j$, such that the iterative method converges fast with a reasonable accuracy and efficiency with respect to memory storage and computational time.

Krylov methods can be divided into stationary and nonstationary variants. Methods such as Jacobi, Gauss-Seidel, (S)SOR iterations are stationary methods, since the same operations on the current iteration vectors are performed in each iteration. They are easy to understand and implement, but they are often not effective. On the other hand, nonstationary methods, that have iteration-dependent coefficients, are a relatively recent development. Their analysis is commonly harder to understand, but they can be highly efficient. They rely on forming an orthogonal basis of the Krylov sequence $\left\{r_0, Ar_0, A^2 r_0, \ldots, A^{j-1} r_0\right\}$. The iterates are then constructed by minimizing the residual over the subspace formed. The prototypical method in this class is the (preconditioned) Conjugate Gradient ((P)CG) method, which is described in Section 2.3 and 2.4. This is a popular and effective nonstationary Krylov solver for linear systems with an SPSD coefficient matrix, as the storage for only a limited number of vectors is required. For non-SPSD matrices, the Krylov solvers GMRES [121] and Bi-CGSTAB [160] are popular methods in use, see [120, 160].

2.3 Conjugate Gradient Method

The Conjugate Gradient (CG) method is probably the most prominent iterative method for solving the SPSD linear system (2.1). It is discovered independently by Hestenes and Stiefel, and they jointly published the method in [72], which has become the classical reference on CG. We refer to [8, 61, 63, 86, 117, 120, 161] for more details about this method.

The purpose of CG is to construct a sequence, $\{x_j\}$, that satisfies (2.6), with $M = I$ and the property that

$$\min_{x_j \in \mathcal{K}_j(A, r_0)} ||x_j - x||_A \tag{2.7}$$

holds, where $||w||_A := \sqrt{(w, Aw)}$ is used for any $w \in \mathbb{R}^n$. In other words, the error is minimized in the A-semi-norm (that is often abbreviated as the A-norm, if there is no ambiguity) at each iteration. This minimum is guaranteed to exist in general, only if A is SPSD. Moreover, CG requires that search direction vectors, $\{p_j\}$, are conjugate with respect to A, i.e.,

$$(Ap_i, p_j) = 0, \quad i \neq j, \tag{2.8}$$

hence the name 'Conjugate Gradient method'. It can be shown that (2.8) is equivalent to the fact that the residuals, $\{r_j\}$, form an orthogonal set, i.e.,

$$(r_i, r_j) = 0, \quad i \neq j. \tag{2.9}$$

Now, the CG method proceeds as follows. The $(j+1)$-th iterate is updated via the search direction:

$$x_{j+1} = x_j + \alpha_j p_j, \tag{2.10}$$

where $\alpha_j \in \mathbb{R}$. This yields

$$r_{j+1} = r_j - \alpha_j A p_j. \tag{2.11}$$

It can be shown that

$$\alpha_j = \frac{(r_j, r_j)}{(Ap_j, p_j)} \tag{2.12}$$

minimizes $||x_j - x||_A$ over all possible choices of α_j and ensures that Eq. (2.9) is satisfied. The search directions are updated using the residuals:

$$p_{j+1} = r_{j+1} + \beta_j p_j, \tag{2.13}$$

where $\beta_j \in \mathbb{R}$ equal to

$$\beta_j = \frac{(r_{j+1}, r_{j+1})}{(r_j, r_j)} \tag{2.14}$$

ensures that Eq. (2.8) is satisfied. In fact, it can be shown that Eqs. (2.12) and (2.14) make p_{j+1} conjugate to all previous search directions, $\{p_i : i = 1, \ldots, j\}$, and r_{j+1} orthogonal to all previous residuals, $\{r_i : i = 1, \ldots, j\}$.

The above derivation leads to Algorithm 1, see below. This is essentially the form of the CG algorithm that appeared in [72].

Algorithm 1 Conjugate Gradient (CG) solving $Ax = b$

```
Select $x_0$. Compute $r_0 := b - Ax_0$ and set $p_0 := r_0$.
for $j := 0, 1, \ldots,$ until convergence do
    $w_j := Ap_j$
    $\alpha_j := \frac{(r_j, r_j)}{(w_j, p_j)}$
    $x_{j+1} := x_j + \alpha_j p_j$
    $r_{j+1} := r_j - \alpha_j w_j$
    $\beta_j := \frac{(r_{j+1}, r_{j+1})}{(r_j, r_j)}$
    $p_{j+1} := r_{j+1} + \beta_j p_j$
end for
$x_{\mathrm{it}} := x_{j+1}$
```

Remark 2.1.

- *The iterative solution of $Ax = b$ is denoted by x_{it} in Algorithm 1 to distinguish it from the exact solution, x.*

- *It is straightforward to derive CG from the Lanczos algorithm for solving symmetric eigensystems and vice versa. The relationship can be exploited to obtain relevant information about the eigensystem of A. We refer to [8, 63, 120] for more details.*

- *It can be proven that the CG algorithm indeed minimizes the error in the solution vector over the Krylov subspace in the A-norm, as presented in (2.7).*

- *The equality* $(r_j, r_j) = 0$ *only happens if the solution,* x*, is already found. Moreover, since* $b \in \mathcal{R}(A)$ *and* $Ax_{j+1} \in \mathcal{R}(A)$*, the residual,* r_{j+1}*, is also in the range of* A*, i.e.,* $r_{j+1} \in \mathcal{R}(A)$*. Because* p_{j+1} *is a linear combination of the residuals,* $\{r_1, \ldots, r_j\}$*, it is in the range of* A *as well, so that* $Ap_{j+1} \neq \mathbf{0}_n$ *if* $p_{j+1} \neq \mathbf{0}_n$*. Therefore,* $(Ap_j, p_j) = 0$ *never happens, since* A *is SPSD and* $r_j \neq \mathbf{0}_n$ *(see [63, Lemma 10.2.1]). Hence, breakdown of CG (even for a singular coefficient matrix) only occurs if it is already converged.*

Due to the SPSD property of A, the algorithm can be formulated such that the memory space is needed for only four vectors and one matrix. Each iteration requires the computations of two inner products, one matrix-vector multiplication and four vector updates. Note that a basis for the Krylov subspace does not need to be stored, and the algorithm only uses short recurrences.

Because the residuals, $\{r_j\}$, are orthogonal to each other, it follows that CG certainly converges within n iterations in exact arithmetic. Moreover, the convergence rate is bounded as a function of the condition number of matrix A, $\kappa(A)$, see Lemma 2.1.

Lemma 2.1. *Let* A *and* x *be the coefficient matrix and the solution vector as in Eq. (2.1), respectively. Let* $\{x_i\}$ *be the sequence of CG-generated iterates. After* $j+1$ *iterations of CG, the error is bounded by*

$$||x - x_{j+1}||_A \leq 2||x - x_0||_A \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^{j+1}. \tag{2.15}$$

Proof. The proof is almost identical to the proof of [105, Thm. 3.3]. □

Remark 2.2.

- *From Inequality (2.15), it follows that the convergence of CG does not depend on zero eigenvalues, see also [77].*

- *The accuracy of* $\{x_j\}$ *is often much better than that (2.15) predicts, due to a good clustering of the eigenvalues of* A *or a favorable choice of starting vectors, see [159].*

From Lemma 2.1, a heuristic rule can be formulated: a faster convergence of CG is expected for a smaller κ. In the ideal case, we should have $\kappa(A) \approx 1$. In the next section, we show that the linear system (2.1) can be converted into a related linear system such that the new coefficient matrix, $\widetilde{A}$, is closer to the identity, i.e., $\kappa(\widetilde{A})$ approaches 1.

2.4 Preconditioned Conjugate Gradient Method

The previous section has been concluded by observing that CG is effective if the coefficient matrix is well-conditioned or has a favorable clustering of eigenvalues. Both the efficiency and robustness of CG can be improved by using so-called ‘preconditioning’.

Preconditioning is simply a means of transforming the original linear system (2.1) into one which has the same solution, but which is likely to be easier to solve with CG. In other words, instead of solving (2.1), we solve the transformed preconditioned linear system,

$$\widetilde{A}\tilde{x} = \tilde{b}, \tag{2.16}$$

where

$$\widetilde{A} := M^{-\frac{1}{2}} A M^{-\frac{1}{2}}, \quad \tilde{x} := M^{\frac{1}{2}} x, \quad \tilde{b} := M^{-\frac{1}{2}} b. \tag{2.17}$$

Matrix M^{-1} is called the preconditioner, as in basic iterative methods. It is required that M is SPD, so that $M^{-\frac{1}{2}}$ exists and $(M^{-\frac{1}{2}})^T = M^{-\frac{1}{2}}$ holds. Subsequently, Lemma 2.2 shows that (2.16) satisfies Assumption 1.1.

Lemma 2.2. *Let the linear system (2.16) with coefficient matrix $\widetilde{A}$ be given. Then, Eq. (2.16) satisfies Assumption 1.1, i.e.,*

- *$\widetilde{A}$ is SPSD;*
- *$\widetilde{A}$ has d zero eigenvalues;*
- *Eq. (2.16) is consistent.*

Proof. The fact that $\widetilde{A}$ is SPSD follows immediately from Lemma A.2 by substituting $B := A$ and $C = M^{-\frac{1}{2}}$.

Moreover, note that

$$Av_i = \mathbf{0}_n \quad \Leftrightarrow \quad M^{-1}Av_i = \mathbf{0}_n, \quad i = 1, \ldots, d,$$

where $v_1, \ldots, v_d$ are the eigenvectors associated with the d zero eigenvalues of A. Combining this fact with the equality $\sigma(\widetilde{A}) = \sigma(M^{-1}A)$ (Lemma A.1), it follows that both A and $\widetilde{A}$ have d zero eigenvalues.

In order to prove that Eq. (2.16) is consistent, it suffices to show that $\tilde{b} \notin \mathcal{N}(\widetilde{A})$ for $\tilde{b} \neq \mathbf{0}_n$. Define $\tilde{y} := M^{\frac{1}{2}} y$ for any $y \in \mathcal{N}(A)$. Then,

$$\widetilde{A}\tilde{y} = M^{-\frac{1}{2}} A y = M^{-\frac{1}{2}} \mathbf{0}_n = \mathbf{0}_n,$$

and, for $\tilde{b} \neq \mathbf{0}_n$,

$$\tilde{b}^T \tilde{y} = (M^{-\frac{1}{2}} b)^T M^{\frac{1}{2}} y = b^T y = 0,$$

since $b \notin \mathcal{N}(A)$ by hypothesis. □

Remark 2.3. *If $y \in \mathcal{N}(A)$, then $\tilde{y} := M^{\frac{1}{2}} y \in \mathcal{N}(\widetilde{A})$.*

Algorithm 1 can now be applied to the linear system (2.16). This results in the preconditioned Conjugate Gradient (PCG) method, see Algorithm 2. The iterate, $\tilde{x}_{j+1}$, can be regarded as an approximation of the solution, $\tilde{x}$, and $\tilde{r}_{j+1} = \tilde{b} - \widetilde{A}\tilde{x}_{j+1}$ can be interpreted as the preconditioned residual.

Algorithm 2 Preconditioned CG (PCG) solving $Ax = b$ (Original Variant)

```
Select $\tilde{x}_0$. Compute $\tilde{r}_0 := \tilde{b} - \widetilde{A}\tilde{x}_0$ and set $\tilde{p}_0 := \tilde{r}_0$.
for $j := 0, 1, \ldots,$ until convergence do
    $\tilde{w}_j := \widetilde{A}\tilde{p}_j$
    $\alpha_j := \frac{(\tilde{r}_j, \tilde{r}_j)}{(\tilde{p}_j, \tilde{w}_j)}$
    $\tilde{x}_{j+1} := \tilde{x}_j + \alpha_j \tilde{p}_j$
    $\tilde{r}_{j+1} := \tilde{r}_j - \alpha_j \tilde{w}_j$
    $\beta_j := \frac{(\tilde{r}_{j+1}, \tilde{r}_{j+1})}{(\tilde{r}_j, \tilde{r}_j)}$
    $\tilde{p}_{j+1} := \tilde{r}_{j+1} + \beta_j \tilde{p}_j$
end for
$x_{\mathrm{it}} := M^{-\frac{1}{2}} \tilde{x}_{j+1}$
```

Subsequently, Algorithm 2 can be simplified using the following substitutions:

$$\left\{ \begin{array}{rcl} \tilde{x}_{j+1} & = & M^{\frac{1}{2}} x_{j+1}; \\ \tilde{r}_{j+1} & = & M^{-\frac{1}{2}} r_{j+1}; \\ \tilde{p}_{j+1} & = & M^{\frac{1}{2}} p_{j+1}; \\ \tilde{w}_{j+1} & = & M^{\frac{1}{2}} w_{j+1}. \end{array} \right. \tag{2.18}$$

This yields Algorithm 3, see below.

Algorithm 3 Preconditioned CG (PCG) solving $Ax = b$ (Practical Variant)

```
Select $x_0$. Compute $r_0 := b - Ax_0$,
solve $My_0 = r_0$, and set $p_0 := y_0$.
for $j := 0, 1, \ldots,$ until convergence do
    $w_j := Ap_j$
    $\alpha_j := \frac{(r_j, y_j)}{(p_j, w_j)}$
    $x_{j+1} := x_j + \alpha_j p_j$
    $r_{j+1} := r_j - \alpha_j w_j$
    Solve $My_{j+1} = r_{j+1}$
    $\beta_j := \frac{(r_{j+1}, y_{j+1})}{(r_j, y_j)}$
    $p_{j+1} := y_{j+1} + \beta_j p_j$
end for
$x_{\mathrm{it}} := x_{j+1}$
```

Remark 2.4.

- *From Algorithm 3, it follows that it is not required to determine* $M^{\frac{1}{2}}$ *or its inverse explicitly.*

- *Compared to Algorithm 1, an additional linear system,* $My_{j+1} = r_{j+1}$*, has to be solved at each iteration. Moreover, an extra matrix,* M*, and an extra vector,* y_j*, should be stored in memory.*

- *In Algorithm 3,* (p_j, Ap_j) *and* (r_j, y_j) *do not vanish, since* M *is SPD and both* $(\tilde{p}_j, \widetilde{A}\tilde{p}_j)$ *and* $(\tilde{r}_j, \tilde{r}_j)$ *are nonzero if the (preconditioned) solution is not yet found (cf. Remark 2.1).*

Because Algorithm 2 can be rewritten as Algorithm 3, the preconditioned linear system (2.16) is often denoted by

$$M^{-1}Ax = M^{-1}b, \quad M = [m_{ij}] = \mathbb{R}^{n \times n}. \tag{2.19}$$

Moreover, the next lemmas show some properties of PCG.

Lemma 2.3. *Let* A *and* M^{-1} *be given as in (2.19). Let* $\{r_i\}$ *and* $\{p_i\}$ *be sequences of residuals and search directions satisfying Eq. (2.18), respectively. Then, the following equality holds for PCG:*

$$(M^{-1}r_i, r_j) = (Ap_i, p_j) = 0, \quad i \neq j. \tag{2.20}$$

Proof. Note first that

$$(\tilde{r}_i, \tilde{r}_j) = (M^{-\frac{1}{2}}r_i, M^{-\frac{1}{2}}r_j) = (M^{-1}r_i, r_j),$$

and

$$(\widetilde{A}\tilde{p}_i, \tilde{p}_j) = (M^{-\frac{1}{2}}AM^{-\frac{1}{2}}M^{\frac{1}{2}}p_i, M^{\frac{1}{2}}p_j) = (M^{-\frac{1}{2}}Ap_i, M^{\frac{1}{2}}p_j) = (Ap_i, p_j),$$

using Eq. (2.18). Since $(\tilde{r}_i, \tilde{r}_j) = (\widetilde{A}\tilde{p}_i, \tilde{p}_j) = 0$ holds for $i \neq j$ (see Eqs. (2.8) and (2.9)), the lemma now follows immediately. □

Lemma 2.4. *Let* A *and* M^{-1} *be given as in (2.19). Let* $\{x_i\}$ *be a sequence of PCG-generated iterates. Then,* x_{j+1} *satisfies the following inequality:*

$$||x - x_{j+1}||_A \leq 2||x - x_0||_A \left(\frac{\sqrt{\kappa\left(M^{-1}A\right)} - 1}{\sqrt{\kappa\left(M^{-1}A\right)} + 1} \right)^{j+1}. \tag{2.21}$$

Proof. We have (see Eq. (2.15))

$$||\tilde{x} - \tilde{x}_{j+1}||_{\widetilde{A}} \leq 2||\tilde{x} - \tilde{x}_0||_{\widetilde{A}} \left(\frac{\sqrt{\kappa\left(\widetilde{A}\right)} - 1}{\sqrt{\kappa\left(\widetilde{A}\right)} + 1} \right)^{j+1}. \tag{2.22}$$

The lemma follows from (2.22), Lemma A.1 and the fact that

$$\begin{aligned}
||\tilde{x} - \tilde{x}_i||_{\widetilde{A}} &= (\tilde{x} - \tilde{x}_i)^T \widetilde{A} (\tilde{x} - \tilde{x}_i) \\
&= \left(M^{\frac{1}{2}} x - M^{\frac{1}{2}} x_i\right)^T M^{-\frac{1}{2}} A M^{-\frac{1}{2}} \left(M^{\frac{1}{2}} x - M^{\frac{1}{2}} x_i\right) \\
&= (x - x_i)^T M^{\frac{1}{2}} M^{-\frac{1}{2}} A M^{-\frac{1}{2}} M^{\frac{1}{2}} (x - x_i) \\
&= (x - x_i)^T A (x - x_i) \\
&= ||x - x_i||_A,
\end{aligned}$$

with $i = 0, 1, \ldots, j+1$. ☐

According to Lemma 2.3, both CG and PCG minimize the error in the solution vector over the Krylov subspace in the A-norm, rather than in the $\widetilde{A}$-norm, see also [8, Sect. 11.2]. Furthermore, it can be noticed from Lemma 2.4 that a fast convergence of PCG relies on the choice of M^{-1}. In the ideal case, we should have $\kappa(\widetilde{A}) = \kappa\left(M^{-1}A\right) \approx 1$.

2.5 Further Considerations

For the PCG method, there are still a few issues left that need further discussion, such as the choice of the preconditioner, starting vector, and termination criterion of the iterative process. These issues are considered in this section.

2.5.1 Preconditioning

Algorithm 3 is only efficient if the preconditioner, M^{-1}, satisfies the following requirements:

(i) M is easy to construct;

(ii) the linear system $My_{j+1} = r_{j+1}$ should be solvable at low cost;

(iii) the eigenvalues of $\widetilde{A}$ should be clustered (around 1).

In other words, a good preconditioner improves the convergence of the iterative method, sufficiently to overcome the extra cost of both constructing and applying the preconditioner. The most standard preconditioner is the Jacobi preconditioner, defined as $M = \mathrm{diag}(A)$. In [158], it is shown that this choice minimizes $\kappa(M^{-1}A)$, if the preconditioner is restricted to a diagonal matrix. Block versions of the Jacobi preconditioner can be derived by a partitioning of the variables. If the index set, $\mathcal{S} = \{1, 2, \ldots, n\}$, is partitioned as $\mathcal{S} = \cup_q \mathcal{S}_q$ with disjoint sets, $\{\mathcal{S}_q\}$, then

$$m_{ij} = \begin{cases} a_{ij}, & \text{if } i \text{ and } j \text{ are in the same index subset;} \\ 0, & \text{otherwise.} \end{cases}$$

The resulting preconditioner is now a block-diagonal matrix, known as the block-Jacobi preconditioner.

Another simple way of defining a preconditioner is to perform an incomplete factorization of A. For example, the incomplete Cholesky decomposition without fill-in [97], known as IC(0), is commonly used. The resulting PCG method is often called ICCG. In this approach, we have $M = LL^T$, where $L = [l_{ij}] \in \mathbb{R}^{n \times n}$ is a lower-triangular matrix having the same sparsity pattern as A, and is close to the lower-triangular matrix associated with the exact Cholesky decomposition. More specifically, the entries of the incomplete Cholesky factor, L, should satisfy the following conditions:

$$\begin{cases} l_{ij} = 0, & \text{if } \ a_{ij} = 0; \\ (LL^T)_{ij} = a_{ij}, & \text{if } \ a_{ij} \neq 0. \end{cases}$$

If A is an SPSD matrix with $a_{i,j} \leq 0$ for all $i \neq j$, such an L always exists, see [77]. Accordingly, IC(0) can always be constructed for our bubbly flows problem, as A is an SPSD M-matrix, see Assumption 1.2.

Remark 2.5.

- *The general algorithm for constructing L can be found in [63, 97]. This algorithm can be reduced by accounting for the exact nonzero pattern of A.*
- *In practice, matrices A and L are stored in an appropriate data structure to save memory storage and obtain an efficient method.*

There are various variants of ICCG known in the literature, such as MICCG [9], RICCG [67] and ILUM [119]. More general matrix-based preconditioners can be found in, e.g., [120]. Another type of preconditioners is operator-based, that exploits properties of the physical problems from which the linear system arises. For example, if we aim at solving the linear system derived from Eq. (1.3), then the preconditioner can be based on the same equation but with a constant ρ (that is the Poisson equation with a constant coefficient), or techniques as described in [32]. The difficulty of finding effective preconditioners is well-recognized, and the development of such preconditioners is a major issue in the current active research.

Remark 2.6.

- *It might happen that $My_{j+1} = r_{j+1}$ cannot be solved accurately. In this case, this solve can be regarded as $y_{j+1} = \mathcal{M}(r_{j+1})$, where $\mathcal{M}$ is a nonlinear mapping from $\mathbb{R}^n$ to $\mathbb{R}^n$. In order to preserve the optimal convergence property of PCG, one can perform a full orthogonalization of the search direction vectors, which might be extended by truncation and restart strategies. This leads to methods based on GMRES, such as the Flexible PCG method [109]. However, it is possible to use the original PCG method with inexact preconditioning, since the convergence rate of the outer PCG process can be maintained up to a certain accuracy for the inner solve, $y_{j+1} = \mathcal{M}(r_{j+1})$, see, e.g., [62, 64].*

- *CG is usually straightforward to parallelize, while PCG might have difficulties due to the choice of the preconditioner, see [4, 44, 161]. For example, the IC(0) preconditioner is cumbersome to parallelize, in contrast to the (block-)Jacobi preconditioner. Hence, extra attention should be paid to the preconditioner,* M^{-1}*, in a parallel environment.*

2.5.2 Starting Vectors and Termination Criteria

In general, there is no restriction for choosing the starting vector, x_0, in the (P)CG method. The convergence rate of the iterative process hardly depends on it, unless $x - x_0$ is already conjugate to some of the eigenvectors of $M^{-1}A$. Common choices for x_0 are the zero vector, the random vector, and a rough estimate of x.

In Algorithm 1, 2 and 3, 'until convergence' means that the iterative process should be terminated if the error, $||x - x_k||_A$, is sufficiently small. Because this error term is not available, it is customary to terminate if the (preconditioned) residual falls below a specified value. This leads to the following widely used termination criteria:

$$\frac{||r_{j+1}||_2}{||r_0||_2} < \delta, \tag{2.23}$$

and

$$\frac{||y_{j+1}||_2}{||y_0||_2} = \frac{||M^{-1}r_{j+1}||_2}{||M^{-1}r_0||_2} < \delta, \tag{2.24}$$

where r_{j+1} and y_{j+1} represent the original and preconditioned residuals at iteration $j+1$, respectively. The tolerance, $\delta > 0$, determines the accuracy of the solution and is a user-supplied parameter.

Remark 2.7.

- *Termination criterion (2.23) does not depend on the preconditioner. Therefore, this criterion is suitable if PCG methods with different* M *have to be compared.*

- *If* $M^{-1}A \approx I$*, then*

  $$\frac{||y_{j+1}||_2}{||y_0||_2} = \frac{||M^{-1}(b - Ax_{j+1})||_2}{||M^{-1}(b - Ax_0)||_2} = \frac{||M^{-1}A(x - Ax_{j+1})||_2}{||M^{-1}A(x - x_0)||_2} \approx \frac{||x - x_{j+1}||_2}{||x - x_0||_2},$$

  so that termination criterion (2.24) relies on 'real' relative errors.

- *Both criteria (2.23) and (2.24) have the drawback that they strongly depend on the starting vector,* x_0*. A relatively large and inaccurate* x_0 *leads to an inaccurate solution, while* x_0 *close to the solution might result in a too stringent termination criterion. More details on various termination criteria used in practice can be found in, e.g., [10].*

2.6 Application to Bubbly Flows

In this section, we present the performance of PCG in our main application of bubbly flow problems. The computations are performed on a serial Pentium 4 (2.80 GHz) computer with a memory capacity of 1GB. Moreover, the code is compiled with FORTRAN g77 on LINUX.

Both 2-D and 3-D variants of the problem setting, as given in Figure 1.2 of Section 1.3, are considered, where the radius of the bubbles is $s = 0.1$. The number of bubbles, m, the grid size, n, and the density contrast, θ, are varied in the experiments. We adopt ICCG (that is, PCG with the IC(0) preconditioner) to solve the resulting linear system. PCG with Jacobi and Block-Jacobi preconditioners is considered in [137], and is less efficient compared to ICCG. We choose for a random starting vector, and the termination criterion is based on (2.24) with tolerance $\delta = 10^{-8}$.

The results of the experiment are given in terms of the total computing time and the number of required iterations for convergence of ICCG, see Table 2.1 and 2.2. The accuracy of the solutions is also checked. They are omitted in the results, since they are of the same order.

(a) $n = 100^2$, $\theta = 10^3$ and various number of bubbles, m.

m	# It.	CPU
0	109	0.1
1	128	0.1
9	247	0.3

(b) $m = 9$, $\theta = 10^3$ and various grid sizes, n.

n	# It.	CPU
100^2	247	0.3
250^2	466	3.6
500^2	1027	34.4

(c) $m = 9$, $n = 100^2$ and various density contrasts, θ.

θ	# It.	CPU
10^3	247	0.3
10^6	352	0.3
10^8	381	0.4

Table 2.1: Results for ICCG applied to 2-D bubbly flow problems. '# It' means the number of required iterations, and 'CPU' is the corresponding computational time in seconds.

From Table 2.1 and 2.2, it can be readily observed that 3-D problems take more iterations and computing time to solve compared to 2-D problems, because the degrees of freedom are larger in the 3-D case. We see that the convergence of ICCG deteriorates when $M^{-1}A$ becomes more ill-conditioned; this is the case when

- the domain consists of more bubbles;
- the degrees of freedom are increased;
- the density contrast grows.

Hence, ICCG is not a scalable and robust method.

(a) $n = 100^3$, $\theta = 10^3$ and various number of bubbles, m.

m	# It.	CPU
0	170	25.2
1	211	31.1
8	291	43.0
27	310	46.0

(b) $m = 27$, $\theta = 10^3$ and various grid sizes, n.

n	# It.	CPU
50^3	199	3.6
100^3	310	46.0
120^3	363	90.5

(c) $m = 27$, $n = 100^2$ and various density contrasts, θ.

θ	# It.	CPU
10^3	310	46.0
10^6	503	71.8
10^8	532	77.5

Table 2.2: Results for ICCG applied to 3-D bubbly flow problems.

2.7 Concluding Remarks

In this chapter, we review basic and Krylov iterative methods. The (P)CG is a popular Krylov iterative method, which is discussed in more detail. Some theoretical properties that are not fully clear in the literature are derived and explained.

Numerical experiments show that ICCG (that is PCG with the IC(0) preconditioner) is not effective to deal with sophisticated bubbly flows. Consequently, there is a need for an alternative of PCG, so that the convergence of its iterative process is more robust and scalable with respect to the number of bubbles, the grid size, and the density contrast. We deal with this issue in the remainder of this thesis.

Chapter 3

Deflation Method

3.1 Introduction

The linear system of our primary interest is (see Eq. (2.1))

$$Ax = b, \quad A \in \mathbb{R}^{n \times n}, \tag{3.1}$$

where the SPSD coefficient matrix, A, has d zero eigenvalues, and $b \in \mathcal{R}(A)$ holds. As discussed in Chapter 2, PCG is a popular method to solve (3.1), and the resulting preconditioned linear system to be considered is (see Eq. (2.19))

$$M^{-1}Ax = M^{-1}b, \tag{3.2}$$

where M^{-1} is an SPD preconditioner. The spectrum of $M^{-1}A$, $\sigma\left(M^{-1}A\right)$, often consists of unfavorable eigenvalues that deteriorate the convergence of PCG and makes PCG less robust, see also [175]. In this chapter, we describe the so-called deflation method that effectively treats these eigenvalues, so that the convergence can be significantly improved, and a more robust and scalable method can be obtained.

The deflation method applied to CG is independently proposed by Nicolaides [108] and Dostal [40]. It is further exploited in several papers, among them are [56, 58, 82, 93, 94, 99, 103, 104, 122, 173]. Below, we first describe the deflation method and its preconditioned variant following [173], where we account for the possible singularity of the coefficient matrix, A. We derive and discuss these methods including their theoretical properties. Moreover, the effectiveness of the deflation method is illustrated in bubbly flow applications.

3.2 Preliminaries

This section presents some preliminaries that are required to describe the deflation method. We start with Definition 3.1.

Definition 3.1. *Let A be an SPSD coefficient matrix as given in (3.1). Suppose that $Z \in \mathbb{R}^{n \times k}$, with full rank and $k < n - d$, is given. Then, we define the invertible Galerkin matrix, $E \in \mathbb{R}^{k \times k}$, the correction matrix, $Q \in \mathbb{R}^{n \times n}$, and the deflation matrix, $P \in \mathbb{R}^{n \times n}$, as follows:*

$$P := I - AQ, \quad Q := ZE^{-1}Z^T, \quad E := Z^T AZ. \tag{3.3}$$

Remark 3.1.

- *The Galerkin matrix, E, is also known as the coarse matrix. In addition, linear system $Ey_2 = y_1$ is often called the Galerkin or coarse system.*
- *The matrices as defined in Definition 3.1 can be easily generalized for a non-SPSD coefficient matrix, A. We refer to [47, 48, 171] for more details.*

In Eq. (3.3), Z is the so-called 'deflation-subspace matrix' whose k columns are called the 'deflation vectors' or 'projection vectors'. These vectors remain unspecified for the moment, but they are chosen in such a way that E is nonsingular. In other words, the following assumption is always fulfilled in this chapter (and in most of the upcoming chapters).

Assumption 3.1. *Z is chosen such that $\mathcal{N}(A) \nsubseteq \mathcal{R}(Z)$, so that E is nonsingular.*

The fact that E is nonsingular if $\mathcal{N}(A) \nsubseteq \mathcal{R}(Z)$ follows from the next lemma.

Lemma 3.1. *Let A, Z and E be as given in Definition 3.1, where Z satisfies Assumption 3.1. If $\mathcal{N}(A) \nsubseteq \mathcal{R}(Z)$, then E is nonsingular.*

Proof. Note first that $\mathcal{N}(A) \nsubseteq \mathcal{R}(Z)$ yields

$$Z\bar{y} \notin \mathcal{N}(A) \ \ \forall \bar{y} \in \mathbb{R}^n. \tag{3.4}$$

Since A is SPSD, we have

$$y^T Ay > 0, \quad y \notin \mathcal{N}(A).$$

In particular, we can take $y = Z\bar{y}$ and substitute this into the latter expression, giving us

$$(Z\bar{y})^T A(Z\bar{y}) = \bar{y}^T Z^T AZ\bar{y} = \bar{y}^T E\bar{y} > 0, \quad Z\bar{y} \notin \mathcal{N}(A) \ \ \forall \bar{y} \in \mathbb{R}^n. \tag{3.5}$$

Combining (3.4) and (3.5) leads to the fact that E is nonsingular. ☐

If $\mathcal{N}(A) \subseteq \mathcal{R}(Z)$, then E would be singular. In this case, the Moore-Penrose generalized inverse (also known as the pseudo-inverse) should be used rather than the real inverse. This is further considered in Chapter 8.

From Eq. (3.3), some results can be readily obtained, see Lemma 3.2.

Lemma 3.2. *Let A, Z, E, Q and P be as given in Definition 3.1, where Z satisfies Assumption 3.1. Let x and b be the solution and right-hand side of (3.1), respectively. Then, the following equalities hold:*

(a) $E^T = E$;

(b) $Q^T = Q = QAQ$;

(c) $QAZ = Z$;

(d) $PAQ = \mathbf{0}_{n,n}$;

(e) $P^2 = P$;

(f) $AP^T = PA$;

(g) $(I - P^T)x = Qb$.

Proof.

(a) $E^T = (Z^T AZ)^T = Z^T AZ = E$;

(b) $Q^T = (ZE^{-1}Z^T)^T = ZE^{-1}Z^T = Q$ using (a), and
$QAQ = ZE^{-1}Z^T AZE^{-1}Z^T = ZE^{-1}EE^{-1}Z^T = ZE^{-1}Z^T = Q$;

(c) $QAZ = ZE^{-1}Z^T AZ = ZE^{-1}E = Z$;

(d) $PAQ = (I - AQ)AQ = AQ - AQAQ = AQ - AQ = \mathbf{0}_{n,n}$ using (b);

(e) $P^2 = (I - AQ)^2 = I - 2AQ + AQAQ = I - 2AQ + AQ = I - AQ = P$, using again (b);

(f) $AP^T = A(I - QA) = A - AQA = (I - AQ)A = PA$;

(g) $(I - P^T)x = QAx = Qb$, using Eq. (3.1).

□

Remark 3.2.

- *In contrast to P, matrices E and Q are symmetric (Lemma 3.2(a) and (b)).*
- *P is a projector (Lemma 3.2(e)).*
- *Although x is unknown, $(I - P^T)x$ can be computed beforehand using Lemma 3.2(g).*

The next lemmas are frequently used in this thesis.

Lemma 3.3. *Let P, A and Z be as given in Definition 3.1, where Z satisfies Assumption 3.1. Then, the following equalities hold:*

(a) $PAZ = \mathbf{0}_{n,k}$;

(b) $P^T Z = \mathbf{0}_{n,k}$.

Proof. Using Lemma 3.2(c), we obtain

(a) $PAZ = (I - AQ)AZ = AZ - AQAZ = AZ - AZ = \mathbf{0}_{n,k}$;

(b) $P^T Z = (I - QA)Z = Z - QAZ = Z - Z = \mathbf{0}_{n,k}$.

□

Remark 3.3.

- *PA has $k + d$ zero eigenvalues (Lemma 3.3(a)), since $\mathcal{N}(PA) = \mathcal{R}(Z) \oplus \mathcal{N}(A)$ and $\mathcal{N}(A) \cap \mathcal{R}(Z) = \emptyset$ (Assumption 3.1).*
- *The deflation matrix, P, has only zero and unit eigenvalues, so that P is positive semi-definite. This follows from the facts that $PAZ = \mathbf{0}_{n,k}$ (Lemma 3.3(a)) and $P^2Y = PY$ for full rank $Y \in \mathbb{R}^{n\times(n-k-d)}$ satisfying $\mathcal{R}(Y) = \mathcal{R}(AZ)^\perp$ (Lemma 3.2(e)).*

Lemma 3.4. *Suppose that A and P are given as in Definition 3.1, where Z satisfies Assumption 3.1. Then, PA is SPSD.*

Proof. Note first that

$$PA = P^2A = PAP^T,$$

using Lemma 3.2(e) and (f). Then, the lemma follows immediately via Lemma A.2 by substituting $B := A$ and $C := P^T$. □

3.3 Deflated CG Method

In this section, the Deflated CG (DCG) method is introduced.

The original linear system (3.1) can be solved by employing the splitting

$$x = (I - P^T)x + P^T x. \tag{3.6}$$

In Eq. (3.6), $(I - P^T)x$ can be computed immediately from Lemma 3.2(g). Hence, only $P^T x$ should be computed in (3.6) in order to find x. We can write

$$\begin{aligned} x = (I - P^T)x + P^T x \quad &\Leftrightarrow \quad x = Qb + P^T x \\ &\Leftrightarrow \quad Ax = AQb + AP^T x \\ &\Leftrightarrow \quad b = AQb + PAx \\ &\Leftrightarrow \quad Pb = PAx, \end{aligned} \tag{3.7}$$

where we have used Lemma 3.2(f). Note that x at the end of Expression (3.7) is not necessarily a solution of the original linear system (3.1), since it might consist of components of the null space of PA, $\mathcal{N}(PA)$. Therefore, this 'deflated' solution is denoted as $\hat{x}$ rather than x. We now can solve the deflated system,

$$PA\hat{x} = Pb, \tag{3.8}$$

using CG. Solutions $\hat{x}$ and x are related to each other by Lemma 3.5.

Lemma 3.5. *Let P be as given in Definition 3.1, where Z satisfies Assumption 3.1. Suppose that x and $\hat{x}$ are solutions of (3.1) and (3.8), respectively. Then, $P^T\hat{x} = P^T x$ holds.*

Proof. Decompose $\hat{x}$ as

$$\hat{x} = x + y,$$

where $y \in \mathcal{R}(Z) \subset \mathcal{N}(PA)$ (Lemma 3.3(a)). This yields

$$P^T\hat{x} = P^T x + P^T y = P^T x,$$

since $P^T y = \mathbf{0}_n$ due to Lemma 3.3(b). □

From Lemma 3.5, solution x can be easily obtained from $\hat{x}$. This is summarized in the next corollary.

Corollary 3.1. *Let P and Q be as given in Definition 3.1, where Z satisfies Assumption 3.1. Suppose that b is the right-hand side of (3.1). Then, solution x of (3.1) can be expressed as*

$$x = Qb + P^T\hat{x}, \tag{3.9}$$

where $\hat{x}$ is a solution of (3.8).

Proof. This follows immediately from Eqs. (3.6), (3.7), (3.8) and Lemma 3.5. □

Remark 3.4.

- *Since PA is SPSD (Lemma 3.4), this can be interpreted as the new coefficient matrix of the linear system.*
- *The deflated linear system (3.8) is obviously singular. It can only be solved as long as it is consistent, i.e., as long as $Pb = PA\hat{x}$ for some $\hat{x}$, see also [77]. Since $b \in \mathcal{R}(A)$ holds, we also have $Pb \in \mathcal{R}(PA)$. Hence (3.8) is a consistent system.*

The resulting DCG algorithm is presented in Algorithm 4. It can be observed that it is almost equal to the original CG method (cf. Algorithm 1).

Remark 3.5.

- *If $P = I$, Algorithm 1 is readily obtained from Algorithm 4.*
- *$\{\hat{r}_j\}$ is the set of deflated residuals satisfying $\hat{r}_j = Pr_j = P(b - A\hat{x}_j)$. The hats on r_{j+1}, w_j and x_{j+1} emphasize that they are deflated versions of the same parameters in Algorithm 1. Implicitly, the other parameters are also deflated versions, but the hats are neglected here for convenience.*

Algorithm 4 Deflated Conjugate Gradient (DCG) solving $Ax = b$

```
Select $x_0$. Compute $r_0 := (b - Ax_0)$, set $\hat{r}_0 := Pr_0$ and $p_0 := \hat{r}_0$.
for $j := 0, 1, \ldots,$ until convergence do
    $\hat{w}_j := PAp_j$
    $\alpha_j := \frac{(\hat{r}_j, \hat{r}_j)}{(\hat{w}_j, p_j)}$
    $\hat{x}_{j+1} := \hat{x}_j + \alpha_j p_j$
    $\hat{r}_{j+1} := \hat{r}_j - \alpha_j \hat{w}_j$
    $\beta_j := \frac{(\hat{r}_{j+1}, \hat{r}_{j+1})}{(\hat{r}_j, \hat{r}_j)}$
    $p_{j+1} := \hat{r}_{j+1} + \beta_j p_j$
end for
$x_{\text{it}} := Qb + P^T \hat{x}_{j+1}$
```

- *Note that $p_{j+1} \notin \mathcal{R}(Z)$, since p_{j+1} is a linear combination of the deflated residuals, $\{\hat{r}_i\}$ with $i = 0, \ldots, j+1$, and each $\hat{r}_i$ satisfies*

  $$(\hat{r}_i, y) = \hat{r}_i^T y = (Pr_i)^T y = r_i^T P^T y = r_i^T \mathbf{0}_n = 0, \quad \forall y \in \mathcal{R}(Z),$$

  using Lemma 3.3(b). Therefore, the inner products, $(\hat{w}_j, p_j)$ and $(\hat{r}_j, \hat{r}_j)$, can only vanish if the deflated solution, $\hat{x}$, has already been found (cf. Remark 2.1).

3.4 Deflated PCG Method

The deflated system (3.8) can also be solved by using an SPD preconditioner, M^{-1}. In this case, we solve

$$\widetilde{P}\widetilde{A}\hat{\tilde{x}} = \widetilde{P}\tilde{b}, \tag{3.10}$$

with (cf. Eq. (2.17))

$$\widetilde{A} := M^{-\frac{1}{2}} A M^{-\frac{1}{2}}, \quad \hat{\tilde{x}} := M^{\frac{1}{2}} \hat{x}, \quad \tilde{b} := M^{-\frac{1}{2}} b,$$

and

$$\widetilde{P} := I - \widetilde{A}\widetilde{Q}, \quad \widetilde{Q} := \widetilde{Z}\widetilde{E}^{-1}\widetilde{Z}^T, \quad \widetilde{E} := \widetilde{Z}^T \widetilde{A}\widetilde{Z}, \tag{3.11}$$

where $\widetilde{Z} \in \mathbb{R}^{n \times k}$ can be interpreted as a preconditioned deflation-subspace matrix. The resulting method is called the Deflated PCG (DPCG) method. When Algorithm 4 is applied to (3.10), we end up with Algorithm 5 (cf. Algorithm 2) that describes the DPCG method.

Remark 3.6. *All properties and results given in Section 3.3 hold in particular for Eq. (3.10) and Algorithm 5.*

Next, Lemma 3.6 is required for the further analysis of DPCG.

Lemma 3.6. *Let P and M^{-1} be as given in Definition 3.1, where Z satisfies Assumption 3.1. Suppose that $\widetilde{P}$ and $\widetilde{Z}$ are defined as in (3.11). Let $\hat{r}_{j+1}$ and $\hat{\tilde{r}}_{j+1}$ be residuals from Algorithms 4 and 5, respectively. Then, the following equalities hold:*

Algorithm 5 Deflated PCG (DPCG) solving $Ax = b$ (Original Variant)

```
Select $\tilde{x}_0$. Compute $\tilde{r}_0 := (\tilde{b} - \widetilde{A}\tilde{x}_0)$, set $\hat{\tilde{r}}_0 := \widetilde{P}\tilde{r}_0$, and $\tilde{p}_0 := \hat{\tilde{r}}_0$.
for $j := 0, 1, \ldots,$ until convergence do
    $\hat{\tilde{w}}_j := \widetilde{P}\widetilde{A}\tilde{p}_j$
    $\alpha_j := \frac{(\hat{\tilde{r}}_j, \hat{\tilde{r}}_j)}{(\tilde{p}_j, \hat{\tilde{w}}_j)}$
    $\hat{\tilde{x}}_{j+1} := \hat{\tilde{x}}_j + \alpha_j \tilde{p}_j$
    $\hat{\tilde{r}}_{j+1} := \hat{\tilde{r}}_j - \alpha_j \hat{\tilde{w}}_j$
    $\beta_j := \frac{(\hat{\tilde{r}}_{j+1}, \hat{\tilde{r}}_{j+1})}{(\hat{\tilde{r}}_j, \hat{\tilde{r}}_j)}$
    $\tilde{p}_{j+1} := \hat{\tilde{r}}_{j+1} + \beta_j \tilde{p}_j$
end for
$\tilde{x}_{j+1} := \widetilde{Q}\tilde{b} + \widetilde{P}^T \hat{\tilde{x}}_{j+1}$
$x_{\mathrm{it}} := M^{-\frac{1}{2}}\tilde{x}_{j+1}$
```

(a) $\widetilde{P} = M^{-\frac{1}{2}} P M^{\frac{1}{2}}$ *with* $Z = M^{-\frac{1}{2}}\widetilde{Z}$*;*

(b) $\hat{\tilde{r}}_{j+1} = M^{-\frac{1}{2}}\hat{r}_{j+1}$.

Proof. The lemma follows immediately from

$$\begin{aligned}
\widetilde{P} &= I - \widetilde{A}\widetilde{Z}\widetilde{E}^{-1}\widetilde{Z}^T \\
&= I - M^{-\frac{1}{2}}AM^{-\frac{1}{2}}\widetilde{Z}(\widetilde{Z}^T M^{-\frac{1}{2}}AM^{-\frac{1}{2}}\widetilde{Z})^{-1}\widetilde{Z}^T \\
&= I - M^{-\frac{1}{2}}AZ(Z^TAZ)^{-1}(M^{\frac{1}{2}}Z)^T \\
&= M^{-\frac{1}{2}}\left(I - AZ(Z^TAZ)^{-1}Z^T\right)M^{\frac{1}{2}} \\
&= M^{-\frac{1}{2}}PM^{\frac{1}{2}},
\end{aligned} \tag{3.12}$$

so that we also have

$$\begin{aligned}
\hat{\tilde{r}}_{j+1} &= \widetilde{P}(\tilde{b} - \widetilde{A}\tilde{x}_{j+1}) \\
&= \widetilde{P}M^{-\frac{1}{2}}(b - Ax_{j+1}) \\
&= \widetilde{P}M^{-\frac{1}{2}}r_{j+1} \\
&= M^{-\frac{1}{2}}Pr_{j+1} \\
&= M^{-\frac{1}{2}}\hat{r}_{j+1}.
\end{aligned}$$

□

Analogously to the PCG method, DPCG can be rewritten by using new variables satisfying (cf. (2.18))

$$\left\{\begin{aligned}
\hat{\tilde{x}}_{j+1} &= M^{\frac{1}{2}}\hat{x}_{j+1}; \\
\hat{\tilde{r}}_{j+1} &= M^{-\frac{1}{2}}\hat{r}_{j+1}; \\
\tilde{p}_{j+1} &= M^{\frac{1}{2}}p_{j+1}; \\
\hat{\tilde{w}}_{j+1} &= M^{\frac{1}{2}}\hat{w}_{j+1},
\end{aligned}\right. \tag{3.13}$$

where we have used Lemma 3.6. Substituting Expressions (3.13) into Algorithm 5 gives us Algorithm 6 (cf. Algorithm 3).

Algorithm 6 Deflated PCG (DPCG) solving $Ax = b$ (Practical Variant)

```
 1: Select $x_0$. Compute $r_0 := b - Ax_0$ and $\hat{r}_0 = Pr_0$,
    solve $My_0 = \hat{r}_0$ and set $p_0 := y_0$.
 2: for $j := 0, \ldots,$ until convergence do
 3:     $\hat{w}_j := PAp_j$
 4:     $\alpha_j := \frac{(\hat{r}_j, y_j)}{(p_j, \hat{w}_j)}$
 5:     $\hat{x}_{j+1} := \hat{x}_j + \alpha_j p_j$
 6:     $\hat{r}_{j+1} := \hat{r}_j - \alpha_j \hat{w}_j$
 7:     Solve $My_{j+1} = \hat{r}_{j+1}$
 8:     $\beta_j := \frac{(\hat{r}_{j+1}, y_{j+1})}{(\hat{r}_j, y_j)}$
 9:     $p_{j+1} := y_{j+1} + \beta_j p_j$
10: end for
11: $x_{\text{it}} := Qb + P^T x_{j+1}$
```

From Algorithm 6, it follows that it is not required to determine $\widetilde{P}$ or $M^{\frac{1}{2}}$ explicitly. As a result, linear system (3.10) is often denoted by

$$M^{-1}PA\hat{x} = M^{-1}Pb. \tag{3.14}$$

Remark 3.7.

- *A deflation technique applied to a preconditioned system (i.e., Eq. (3.10)) is equivalent to preconditioning of a deflated system (i.e., Eq. (3.14)).*

- *All known properties and results for PCG also hold for DPCG, where PA can be interpreted as the coefficient matrix A in Eq. (2.19) and Algorithm 3. Moreover, if $P = I$ is taken, Algorithm 6 is reduced to Algorithm 3.*

- *DPCG could also be derived in a different way, so that the resulting linear system is*

  $$P^T M^{-1} Ax = P^T M^{-1} b, \tag{3.15}$$

  rather than Eq. (3.14), where the last step of Algorithm 6 (Line 11) is carried out before the iteration process starts, see [82, 93, 94, 108, 122]. More details about this variant can be found in Chapter 6.

Similar results as Lemma 2.3 and 2.4 hold for DPCG, see below.

Lemma 3.7. *Suppose that A, M^{-1} and P are given as in Definition 3.1, where Z satisfies Assumption 3.1. Let $\{\hat{r}_i\}$ and $\{p_i\}$ be sequences of residuals and search directions as generated by Algorithm 6, respectively. Then, the following equality holds for DPCG:*

$$(M^{-1}\hat{r}_i, \hat{r}_j) = (PAp_i, p_j) = 0, \quad i \neq j. \tag{3.16}$$

Proof. The proof is similar to the proof of Lemma 2.3. ☐

Lemma 3.8. *Suppose that A, M^{-1} and P are given as in Definition 3.1, where Z satisfies Assumption 3.1. Let x be the solution of Eq. (3.1) and $\{\hat{x}_i\}$ be the sequence of solutions generated by Algorithm 6. Then, the $(j+1)$-th iterate of DPCG, $\hat{x}_{j+1}$, satisfies the next inequality:*

$$||\hat{x} - \hat{x}_{j+1}||_A \leq 2||\hat{x} - \hat{x}_0||_A \left(\frac{\sqrt{\kappa\left(M^{-1}PA\right)} - 1}{\sqrt{\kappa\left(M^{-1}PA\right)} + 1} \right)^{j+1}. \tag{3.17}$$

Proof. We have (see Eq. (2.21))

$$||\hat{x} - \hat{x}_{j+1}||_{PA} \leq 2||\hat{x} - \hat{x}_0||_{PA} \left(\frac{\sqrt{\kappa\left(M^{-1}PA\right)} - 1}{\sqrt{\kappa\left(M^{-1}PA\right)} + 1} \right)^{j+1}. \tag{3.18}$$

Moreover, note that $PA(P^Ty) = P^2Ay = PAy$ by applying Lemma 3.2. Hence, y is a solution of Eq. (3.8) if and only if P^Ty is also a solution of (3.8). Therefore,

$$\begin{aligned}
||\hat{x} - \hat{x}_{j+1}||_{PA}^2 &= (\hat{x} - \hat{x}_{j+1})^T PA(\hat{x} - \hat{x}_{j+1}) \\
&= (\hat{x} - \hat{x}_{j+1})^T PAP^T(\hat{x} - \hat{x}_{j+1}) \\
&= (P^T\hat{x} - P^T\hat{x}_{j+1})^T A(P^T\hat{x} - P^T\hat{x}_{j+1}) \\
&= (\hat{x} - \hat{x}_{j+1})^T A(\hat{x} - \hat{x}_{j+1}) \\
&= ||\hat{x} - \hat{x}_{j+1}||_A^2.
\end{aligned}$$

Substituting $||\hat{x} - \hat{x}_{j+1}||_{PA} = ||\hat{x} - \hat{x}_{j+1}||_A$ into (3.18) leads to (3.17). ☐

Lemma 3.7 implies that, for the DPCG method, the search directions, $\{p_i\}$, are conjugate with respect to PA, while the deflated residuals, $\{\hat{r}_i\}$, are orthogonal in the M^{-1}-norm. In addition, the convergence of DPCG highly depends on $\kappa\left(M^{-1}PA\right)$ according to Lemma 3.8.

3.5 Properties of the Deflation Method

In this section, we derive some theoretical properties of the DPCG method, where Z is first assumed to consist of eigenvectors, and, thereafter, Z is arbitrary. If we restrict ourselves to linear systems with an invertible coefficient matrix, then more properties of this method can be found in [56, 82, 103–105, 122, 173].

3.5.1 Results for an Eigenvector Deflation Subspace

Theorem 3.1 (cf. [103, Thm. 2.5]) shows that using eigenvectors as deflation vectors can be effective in order to obtain a small $\kappa\left(M^{-1}PA\right)$.

Theorem 3.1. *Suppose that A, M^{-1} and P are given as in Definition 3.1. Let $M^{-1}A$ have eigenvalues $\{\lambda_i\}$ with corresponding orthonormal eigenvectors $\{v_i\}$. If $Z :=$*

$[v_{d+1}\ v_{d+2}\ \cdots\ v_{d+k}]$, *then*

$$\sigma(M^{-1}PA) = \{0, \ldots, 0, \lambda_{d+k+1}, \ldots, \lambda_n\}.$$

Proof. We first prove that

$$\widetilde{P}\widetilde{A}\tilde{v}_i = \begin{cases} 0, & i = 1, \ldots, d+k; \\ \lambda_i, \tilde{v}_i & i = d+k+1, \ldots, n, \end{cases} \tag{3.19}$$

where $\{\tilde{v}_i\}$ is the set of orthonormal eigenvectors corresponding to the eigenvalues of $\widetilde{P}\widetilde{A}$. Note that, according to Lemma A.1, $M^{-1}A$ and $\widetilde{A}$ have the same eigenvalues, $\{\lambda_i\}$, but corresponding to different eigenvectors (i.e, $\{\tilde{v}_i\} \neq \{v_i\}$). Define $\Lambda = \mathrm{diag}(\lambda_{d+1}, \ldots, \lambda_{d+k})$ and $\widetilde{Z} := [\tilde{v}_{d+1}\ \cdots\ \tilde{v}_{d+k}]$, giving us $\widetilde{A}\widetilde{Z} = \widetilde{Z}\Lambda$. Since the eigenvectors are orthonormal (i.e., $\tilde{v}_i^T\tilde{v}_j = \delta_{ij}$, where δ_{ij} denotes the Kronecker delta), we have $\widetilde{Z}^T\widetilde{Z} = I$. Then, we derive

$$\widetilde{E} = \widetilde{Z}^T\widetilde{A}\widetilde{Z} = \widetilde{Z}^T\widetilde{Z}\Lambda = \Lambda,$$

yielding

$$\widetilde{P} = I - \widetilde{A}\widetilde{Z}^T\widetilde{E}^{-1}\widetilde{Z}^T = I - \widetilde{Z}^T\Lambda\Lambda^{-1}\widetilde{Z}^T = I - \widetilde{Z}\widetilde{Z}^T.$$

Moreover, we have

$$\widetilde{Z}\widetilde{Z}^T\tilde{v}_i = \begin{cases} \widetilde{Z}\mathbf{e}_k^{(i)} & = & \tilde{v}_i, & i = d+1, \ldots, d+k; \\ \widetilde{Z}\mathbf{0}_k & = & \mathbf{0}_n, & i = d+k+1, \ldots, n. \end{cases}$$

This results in

$$\widetilde{P}\widetilde{A}\tilde{v}_i = \lambda_i\widetilde{P}\tilde{v}_i = \lambda_i\tilde{v}_i - \lambda_i\widetilde{Z}\widetilde{Z}^T\tilde{v}_i = \begin{cases} & & 0, & i = 1, \ldots, d; \\ \lambda_i\tilde{v}_i - \lambda_i\tilde{v}_i & = & 0, & i = d+1, \ldots, d+k; \\ \lambda_i\tilde{v}_i - 0 & = & \lambda_i\tilde{v}_i, & i = d+k+1, \ldots, n, \end{cases}$$

which proves Eq. (3.19).

Subsequently, Eq. (3.19) can be transformed into $M^{-1}PAv_i = \lambda v_i$, since

$$\begin{aligned} \widetilde{P}\widetilde{A}\tilde{v}_i = \lambda\tilde{v}_i \quad & \Leftrightarrow \quad M^{-\frac{1}{2}}PM^{\frac{1}{2}}\left(M^{-\frac{1}{2}}AM^{-\frac{1}{2}}\right)\tilde{v}_i = \lambda_i\tilde{v}_i \\ & \Leftrightarrow \quad M^{-\frac{1}{2}}PAM^{-\frac{1}{2}}\tilde{v}_i = \lambda_i\tilde{v}_i \\ & \Leftrightarrow \quad M^{-\frac{1}{2}}PAv_i = \lambda_i M^{\frac{1}{2}}v_i \\ & \Leftrightarrow \quad M^{-1}PAv_i = \lambda v_i, \end{aligned}$$

where $Z := M^{-\frac{1}{2}}\widetilde{Z}$, $v_i := M^{-\frac{1}{2}}\tilde{v}_i$ and Lemma 3.6(a) are used. Indeed, λ_i is an eigenvalue of both $M^{-1}PA$ and $\widetilde{P}\widetilde{A}$. Since

$$\begin{aligned}\widetilde{A}\tilde{v}_i = \lambda_i \tilde{v}_i \quad &\Leftrightarrow \quad M^{-\frac{1}{2}}AM^{-\frac{1}{2}}\tilde{v}_i = \lambda_i \tilde{v}_i \\ &\Leftrightarrow \quad M^{-\frac{1}{2}}AM^{-\frac{1}{2}}M^{\frac{1}{2}}v_i = \lambda_i M^{\frac{1}{2}}v_i \\ &\Leftrightarrow \quad M^{-1}Av_i = \lambda_i v_i,\end{aligned}$$

we obtain that $\{v_i\}$ is the set of eigenvectors of $M^{-1}A$, that can be scaled such that they are orthonormal. ☐

Corollary 3.2. *Suppose that we have the same setting as in Theorem 3.1. Then,* $\kappa\left(M^{-1}PA\right) \leq \kappa\left(M^{-1}A\right)$.

Proof. $\kappa\left(M^{-1}PA\right) = \frac{\lambda_n}{\lambda_{d+k+1}} \leq \frac{\lambda_n}{\lambda_{d+1}} = \kappa\left(M^{-1}A\right)$. ☐

From Lemma 3.8 and Corollary 3.2, we obtain that DPCG with eigenvectors as deflation vectors is expected to converge faster than PCG. The resulting method is sometimes called 'eigenvector deflation' or 'spectral deflation'.

Remark 3.8. *Eigenvectors corresponding to the smallest nonzero eigenvalues of A are used as deflation vectors in Theorem 3.1, since M^{-1} often treats the largest eigenvalues of A effectively. In this case, the deflation method is fast in convergence if P acts as a complementary part of the preconditioning by projecting the smallest eigenvalues to zero. However, in general, eigenvectors associated with the largest eigenvalues of A, or a combination of these two approaches, can also be used as deflation vectors in order to reduce $\kappa\left(M^{-1}PA\right)$, see, e.g., [82] where two-fold deflation techniques are introduced based on this idea.*

3.5.2 Results for an Arbitrary Deflation Subspace

Eigenvector deflation can be very effective, but, unfortunately, eigenvectors are usually expensive to compute in practice. In addition, eigenvectors are often dense, leading to a possibly expensive deflation matrix, P. Ideally, Z should consist of sparse and good approximations of eigenvectors, which is further examined in Chapter 4. Moreover, it is also common to choose algebraic vectors as columns in Z (see also Chapter 4). In this case, it is not necessarily guaranteed that these vectors are good approximations of the unfavorable eigenvectors. Hence, the properties as described in Section 3.5.1 are not valid anymore. Instead, we show some properties of the DPCG method, where Z is arbitrary. Note that these properties hold in particular for Z consisting of eigenvectors.

Comparison of Deflated Coefficient Matrices

Define

$$P_i := I - AQ_i, \quad Q_i := Z_i E_i^{-1} Z_i^T, \quad E_i := Z_i^T A Z_i, \quad Z_i \in \mathbb{R}^{n \times k_i}, \tag{3.20}$$

for $k_i < n - d$ and $i = 1, 2, \ldots, k$, where each Z_i satisfies Assumption 3.1. It is convenient to adopt this notation with subscripts by comparing deflation matrices with different deflation subspaces. We start with Theorem 3.2 and 3.3, which are generalizations of [103, Lemma 2.9 and Theorem 2.12].

Theorem 3.2. *Let Z_i and P_i be defined as in (3.20) with $i = 1, 2$ and $k_1 = k_2 = k$. If $\mathcal{R}(Z_1) = \mathcal{R}(Z_2)$, then $M^{-1}P_1A = M^{-1}P_2A$, and, in particular, $Q_1 = Q_2$.*

Proof. The proof is identical to the proof for the case that A is invertible, see [103, Lemma 2.9]. ☐

Theorem 3.3. *Suppose that A and M^{-1} are given as in Definition 3.1. Let Z_i and P_i be defined as in (3.20) with $i = 1, 2$. If $\mathcal{R}(Z_1) \subseteq \mathcal{R}(Z_2)$, then*

$$\kappa(M^{-1}P_1A) \geq \kappa(M^{-1}P_2A). \tag{3.21}$$

Proof. In Section 3.4, it has been shown that $M^{-1}P_iA = \widetilde{P}_i\widetilde{A}$ for $i = 1, 2$, with

$$\begin{cases} \widetilde{A} & := & M^{-\frac{1}{2}}AM^{-\frac{1}{2}}, \\ \widetilde{P}_i & := & I - \widetilde{A}\widetilde{Q}_i, \quad \widetilde{Q}_i := \widetilde{Z}_i\widetilde{E}_i^{-1}\widetilde{Z}_i^T, \quad \widetilde{E}_i := \widetilde{Z}_i^T\widetilde{A}\widetilde{Z}_i, \quad Z_i := M^{-\frac{1}{2}}\widetilde{Z}_i. \end{cases}$$

Moreover, if $\mathcal{R}(Z_1) \subseteq \mathcal{R}(Z_2)$ (i.e., $\mathcal{R}(M^{-\frac{1}{2}}\widetilde{Z}_1) \subseteq \mathcal{R}(M^{-\frac{1}{2}}\widetilde{Z}_2)$), then also $\mathcal{R}(\widetilde{Z}_1) \subseteq \mathcal{R}(\widetilde{Z}_2)$, using Lemma A.13. Now, it suffices to prove that

$$\begin{cases} \lambda_n(\widetilde{P}_1\widetilde{A}) & \geq & \lambda_n(\widetilde{P}_2\widetilde{A}); \\ \lambda_{k_1+d+1}(\widetilde{P}_1\widetilde{A}) & \leq & \lambda_{k_2+d+1}(\widetilde{P}_2\widetilde{A}), \end{cases} \tag{3.22}$$

since this implies $\kappa(\widetilde{P}_1\widetilde{A}) \geq \kappa(\widetilde{P}_2\widetilde{A})$ for $\mathcal{R}(\widetilde{Z}_1) \subseteq \mathcal{R}(\widetilde{Z}_2)$; hence, the lemma follows. Note first that $(\widetilde{P}_1 - \widetilde{P}_2)\widetilde{A}$ is positive semi-definite, which can be easily proven by applying the same procedure as in the proof of [103, Lemma 2.8]. By combining this fact with [103, Lemma 2.2], the first inequality of (3.22) can be obtained. The proof of the second inequality of (3.22) is exactly the same as for the case that $\widetilde{A}$ is invertible, see the proof of [103, Thm. 2.10]. ☐

Corollary 3.3. *Suppose that A and M^{-1} are given as in Definition 3.1. Let P_i be defined as in (3.20). Define $Z_i := [z_1 \; z_2 \; \cdots \; z_i]$ for $i = 1, 2, \ldots, k$. Then,*

$$\lambda_{d+2}(M^{-1}P_1A) \leq \lambda_{d+3}(M^{-1}P_2A) \leq \ldots \leq \lambda_{d+k+1}(M^{-1}P_kA),$$

and

$$\lambda_n(M^{-1}P_1A) \geq \lambda_n(M^{-1}P_2A) \geq \ldots \geq \lambda_n(M^{-1}P_kA).$$

This yields

$$\kappa(M^{-1}P_1A) \geq \kappa(M^{-1}P_2A) \geq \ldots \geq \kappa(M^{-1}P_kA) = \kappa(M^{-1}PA).$$

Proof. Note that $\mathcal{R}(Z_{i-1}) \subseteq \mathcal{R}(Z_i)$ holds for all $i = 2, \ldots, k$. Then, the corollary follows from Theorem 3.3. ☐

Theorem 3.2 implies that P is determined by the space spanned by the columns of Z rather than the actual columns. This has direct consequences for constructing the deflation vectors, see Chapter 4. Furthermore, Theorem 3.3 and Corollary 3.3 show that the condition number of $M^{-1}PA$ becomes more favorable by increasing the number of (arbitrary) vectors in Z; hence, a better convergence of the iterative process is expected, although more work is needed to solve the Galerkin system at each iteration.

Comparison of Deflated and Original Coefficient Matrices

Here, we prove that the condition number of PA is always below that of A for all choices of Z, see Theorem 3.4.

Theorem 3.4. *Suppose that A and P are given as in Definition 3.1. For any full-rank Z, the following inequality holds:*

$$\kappa(PA) \leq \kappa(A). \tag{3.23}$$

Proof. It suffices to show that

$$\left\{ \begin{array}{rcl} \lambda_{d+1}(A) & \leq & \lambda_{d+k+1}(PA); \\ \lambda_n(A) & \geq & \lambda_n(PA), \end{array} \right.$$

for all Z with rank $Z = k$.

The proof of $\lambda_n(PA) < \lambda_n(A)$ is as follows. Note that

$$A - PA = AQA,$$

which is symmetric, because of $(AQA)^T = AQA$. Moreover,

$$(AQ)^2 = AQAQ = AQ,$$

can be derived from Lemma 3.2(b), so that AQ is a projector. Therefore, AQA is SPSD, where we have also used Lemma A.3 by taking $S := AQ$ and $R := A$. Then, $\lambda_i(A) \geq \lambda_i(PA)$, for all $i = 1, \ldots, n$, follows from Lemma A.4. Thus, we particularly have

$$\lambda_n(A) \geq \lambda_n(PA).$$

Next, we show that $\lambda_{d+1}(A) \leq \lambda_{d+k+1}(PA)$. Due to Corollary 3.3, it suffices to prove $\lambda_{d+1}(A) \leq \lambda_{d+2}(P_1A)$, where Z_1 consists of just one deflation vector, so that

$$P_1A = \left(I - AZ_1E_1^{-1}Z^T\right)A = \left(I - \gamma Azz^T\right)A = A - \gamma Azz^TA, \tag{3.24}$$

with $z := Z_1 \in \mathbb{R}^n$ and $\gamma := E_1^{-1} \in \mathbb{R}$. The inequality $Az \neq \mathbf{0}_n$ is always satisfied, since $\mathcal{N}(A) \not\subseteq \mathcal{R}(Z)$ (otherwise Assumption 1.2 cannot be satisfied). Eq. (3.24) implies

$$P_1A = A - R, \quad R := \gamma Azz^TA,$$

so that R is symmetric. From Lemma A.12, we have

$$\text{rank } R = \text{rank } \gamma A z z^T A = \text{rank } A z z^T A = \text{rank } z z^T = 1, \quad A z \neq \mathbf{0}_n.$$

Hence, the conditions of Lemma A.5 are satisfied. By taking $B := A$ and $C := -R$ in that lemma, $\lambda_i(A) \leq \lambda_{i+1}(P_1 A)$ is obtained for $i = 1, 2, \ldots, n$. In particular, we have

$$\lambda_{d+1}(A) \leq \lambda_{d+2}(P_1 A),$$

which completes the proof. □

Subsequently, we prove that Theorem 3.4 can be generalized using an SPD preconditioner, M^{-1}, see Theorem 3.5.

Theorem 3.5. *Let P be given as in Definition 3.1. Then, the following inequality holds:*

$$\kappa(M^{-1} P A) \leq \kappa(M^{-1} A), \tag{3.25}$$

for any A, M^{-1} and Z as given in Definition 3.1.

Proof. In Section 3.4, it has been shown that $M^{-1} P A = \widetilde{P}\widetilde{A}$, with

$$\begin{cases} \widetilde{A} & := & M^{-\frac{1}{2}} A M^{-\frac{1}{2}}; \\ \widetilde{P} & := & I - \widetilde{A}\widetilde{Q}, \quad \widetilde{Q} := \widetilde{Z}\widetilde{E}^{-1}\widetilde{Z}^T, \quad \widetilde{E} := \widetilde{Z}^T \widetilde{A} \widetilde{Z}, \quad Z := M^{-\frac{1}{2}} \widetilde{Z}. \end{cases}$$

Combining Lemma A.1 and Theorem 3.4, we obtain

$$\kappa(M^{-1} P A) = \kappa(\widetilde{P}\widetilde{A}) \leq \kappa(\widetilde{A}) = \kappa(M^{-1} A).$$

□

Theorem 3.5 shows that $M^{-1} P A$ is always better conditioned than $M^{-1} A$. Therefore, the convergence of DPCG is expected to be equal or faster than the original PCG method for each full-rank matrix Z and SPD matrix M^{-1}.

3.5.3 Termination Criteria

In many numerical experiments performed in this thesis, PCG and DPCG are compared. For a fair comparison, equivalent termination criteria of the methods are essential. For this purpose, we need the following lemma.

Lemma 3.9. *Let r_i and $\hat{r}_i$ be residuals from Algorithms 3 and 6, respectively. Then, $\hat{r}_i = r_i$ holds for $i = 0, 1, \ldots$.*

Proof. Using Lemma 3.2 and Corollary 3.1, we derive

$$\begin{aligned}\hat{r}_i &= P(b - A\tilde{x}_i)\\ &= Pb - AP^T\hat{x}_i\\ &= b - A(Qb + P^T x_i)\\ &= b - Ax_i\\ &= r_i.\end{aligned}$$

□

From Lemma 3.9, we obtain that the deflated residuals are identical to the original residuals, although they might differ in practice due to round-off errors. Consequently, equivalent termination criteria of PCG and DPCG can be derived, so that we obtain (cf. Eq. (2.23))

$$\frac{||r_{j+1}||_2}{||r_0||_2} < \delta \quad \Leftrightarrow \quad \frac{||\hat{r}_{j+1}||_2}{||\hat{r}_0||_2} < \delta, \tag{3.26}$$

and (cf. Eq. (2.24))

$$\frac{||M^{-1}r_{j+1}||_2}{||M^{-1}r_0||_2} < \delta \quad \Leftrightarrow \quad \frac{||M^{-1}\hat{r}_{j+1}||_2}{||M^{-1}\hat{r}_0||_2} < \delta, \tag{3.27}$$

for a specified termination tolerance, $\delta > 0$. Both criteria, (3.26) and (3.27), are used throughout this thesis. A deeper discussion about the termination criterion of DPCG can be found in [173, Sect. 4].

3.6 Application to Bubbly Flow Problems

The 2-D and 3-D variants of the bubbly flows, as given in Section 1.3 (see Figure 1.2), are considered in this section. In Section 2.6, we have seen that $M^{-1}A$ is very ill-conditioned for sophisticated bubbly flows, where M^{-1} is the IC(0) preconditioner. In this case, ICCG shows slow convergence. In this section, the convergence is accelerated using the deflation method. The resulting method is called DICCG, that is DPCG with the IC(0) preconditioner. We often denote DICCG with k deflation vectors as DICCG$-k$.

The most simple choice for the deflation subspace is the subspace spanned by structured and uniform subdomains, which are chosen independently of the bubbly flow geometry, see Figure 3.1. Mathematically, this is defined as follows. Let the open domain, Ω, be divided into subdomains, Ω_j, $j = 1, 2, \ldots, q+1$, such that $\overline{\Omega} = \cup_{j=1}^{q+1}\overline{\Omega}_j$ and $\Omega_i \cap \Omega_j = \emptyset$ for all $i \neq j$. In addition, q is always chosen such that $q+1$ is a divisor of n. The discretized domain and subdomains are denoted by Ω_h and Ω_{h_j}, respectively. Then, for each Ω_{h_j} with $j = 1, 2, \ldots, q+1$, we introduce a deflation vector, z_j, as follows:

$$(z_j)_i := \begin{cases} 0, & x_i \in \Omega_h \setminus \Omega_{h_j};\\ 1, & x_i \in \Omega_{h_j}, \end{cases} \tag{3.28}$$

where x_i is a grid point in the discretized domain, Ω_h. Then, for $q > 1$, we define $Z := [z_1\ z_2\ \cdots\ z_q]$, so that $k = q$. Hence, Z consists of disjunct orthogonal piecewise-constant vectors and satisfies $\mathbf{1}_n = \mathcal{N}(A) \not\subseteq \mathcal{R}(Z)$, which implies nonsingularity of E. A deeper discussion on subdomain deflation is presented in Chapter 4.

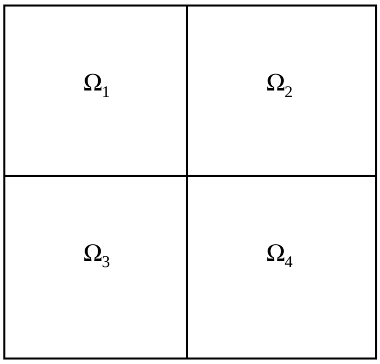


Figure 3.1: Deflation subdomains with $k = 3$, which are chosen independently of the density geometry of the bubbly flow.

The efficiency of DICCG depends on the implementation of this method. We deal with this issue in Chapter 8. For the time being, we report the results by considering the number of iterations. In the numerical experiments, the number of bubbles, m, is varied, whereas the grid sizes ($n = 100^2$ in 2-D and $n = 100^3$ in 3-D) and density contrast ($\epsilon = 10^3$) are fixed. The stopping criterion is based on (3.27) with $\delta = 10^{-8}$.

3.6.1 Results of Numerical Experiments

The results of both 2-D and 3-D experiments are presented in Table 3.1. The accuracy of the solutions are of the same order (see [140]), and, therefore, they are not included in the table.

Considering the results in Table 3.1, we see that DICCG−k always requires fewer iterations compared to ICCG. This confirms Theorem 3.5. It can be observed that, for larger k, DICCG−k requires fewer iterations than for smaller k, which is as expected from Theorem 3.3. The gain factor of DICCG−k for large k can be more than 10, but we note that each iteration becomes more expensive in this case, see Chapter 8 for details. Moreover, an increase of the number of bubbles often leads to a worse performance of ICCG and DICCG−k becomes more superior to ICCG. Additionally, DICCG−k is less sensitive to m for larger k.

Subsequently, the relative residuals of both ICCG and DICCG based on (3.27) are depicted for two test cases in Figure 3.2. It can be noticed that the behavior of the residuals of ICCG is irregular, due to possible unfavorable eigenvectors caused by the presence of the bubbles. For DICCG−k, we can see that a larger k leads to a smoother behavior of the residuals; hence, a faster convergence of the iterative process. The success of the method is two-fold: a larger k leads to a deflation subspace that approximates more eigenvectors corresponding to the unfavorable eigenvalues and each

(a) 2-D experiments with $n = 100^2$ and $\theta = 10^3$.

Method	$m = 0$	$m = 1$	$m = 9$
ICCG	109	128	247
DICCG$-(5^2 - 1)$	49	51	70
DICCG$-(10^2 - 1)$	32	34	44
DICCG$-(20^2 - 1)$	21	22	27
DICCG$-(25^2 - 1)$	19	20	23
DICCG$-(50^2 - 1)$	12	13	14

(b) 3-D experiments with $n = 100^3$ and $\theta = 10^3$.

Method	$m = 0$	$m = 1$	$m = 8$	$m = 27$
ICCG	170	211	291	310
DICCG$-(2^3 - 1)$	109	206	160	275
DICCG$-(5^3 - 1)$	56	58	72	97
DICCG$-(10^3 - 1)$	35	36	36	60
DICCG$-(20^3 - 1)$	22	25	22	31

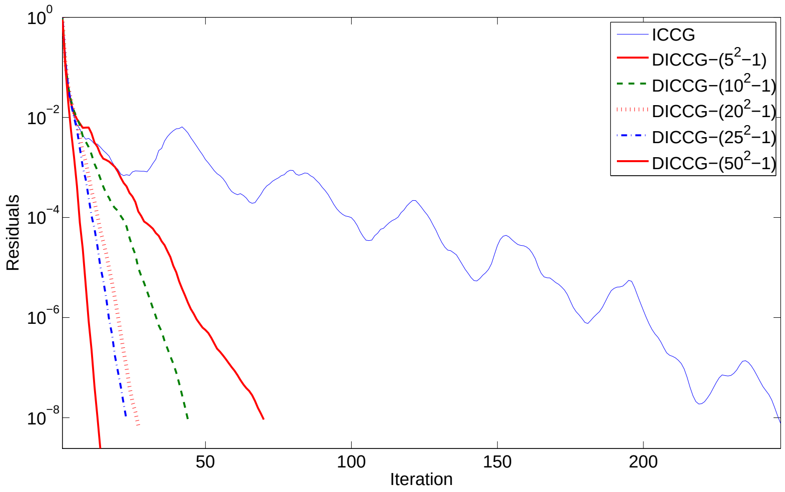
Table 3.1: Number of iterations for ICCG and DICCG$-k$ for various number of bubbles, m, and deflation vectors, k.

of these approximations is more accurate.

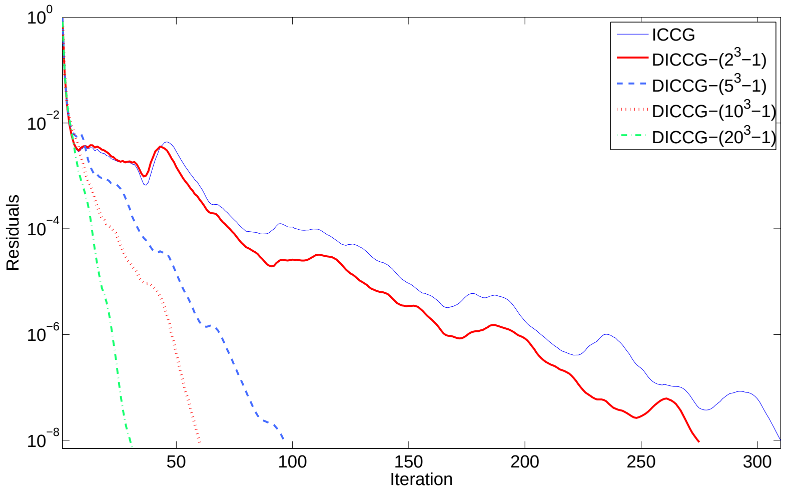
3.7 Concluding Remarks

In this chapter, we describe a deflation method applied to linear systems with a singular coefficient matrix. Thereafter, new theoretical properties are derived and the deflation theory applied to invertible coefficient matrices is generalized. Numerical experiments show that the deflation method with subdomain deflation vectors are very effective for bubbly flow problems.

There are several open issues left, which are treated in the next chapters. We explain the effectiveness of subdomain deflation vectors, and compare them with other common choices. Moreover, we deal with the implementation of the deflation method to obtain an efficient solver. The optimal choice of the number of deflation vectors is investigated. In addition, the application of deflation to singular coefficient matrices is examined. We also relate and compare the deflation method to other two-level PCG methods in order to determine the optimal method.



(a) 2-D experiment with 9 bubbles.



(b) 3-D experiment with 27 bubbles.

Figure 3.2: Norm of the relative residuals during the iterations of ICCG and DICCG$-k$.

Chapter 4

Selection of Deflation Vectors

4.1 Introduction

The underlying idea of applying the deflation method is to effectively treat (extremely) unfavorable eigenvalues that delay the convergence of the PCG method. Recall that the deflation matrix is defined as (see Definition 3.1)

$$P := I - AQ, \quad Q := ZE^{-1}Z^T, \quad E := Z^T AZ, \tag{4.1}$$

with a full-rank deflation-subspace matrix, $Z \in \mathbb{R}^{n \times k}$, consisting of deflation vectors (also known as projection vectors). As mentioned in Chapter 3, the success of the deflation method highly depends on the choice of Z. In the ideal case with respect to convergence, Z should consist of eigenvectors associated with the most unfavorable (often the smallest) eigenvalues of $M^{-1}A$, see Theorem 3.1. These eigenvalues do not play a role anymore in the convergence behavior, so that a faster convergence of the iterative process can be expected. However, the computation of these eigenvectors can be very expensive, and, in addition, these dense vectors might be inefficient in use, since they require much memory and expensive computations with P. Therefore, we intend to find sparse deflation vectors that approximate the unfavorable eigenspace, so that Theorem 3.1 still holds to a certain extent. Additionally, with respect to implementation, it would be favorable to have deflation vectors such that the resulting deflation method is easily parallelizable, and is straightforward to implement in an existing PCG code. In summary, the deflation method should satisfy the next requirements in the ideal case:

- the deflation-subspace matrix, Z, is sparse;
- the deflation vectors approximate the eigenspace corresponding to the unfavorable eigenvalues;
- the cost of constructing deflation vectors is relatively low;
- the method has favorable parallel properties;

- the approach can be easily implemented in an existing PCG code.

Remark 4.1. *The best strategy to choose Z strongly depends on the application, the wishes of the user and the available information about the solution or (the behavior of) unfavorable eigenvectors. There is no optimal choice that always leads to the best results for all applications. One of the main focuses in this chapter is constructing a strategy to find optimal deflation vectors for bubbly flow problems, that might work for various other applications as well.*

This chapter is organized as follows. In Section 4.2, some strategies for choosing deflation vectors known in the literature are reviewed and discussed. Subsequently, Section 4.3 is devoted to finding and analyzing the optimal strategy to choose Z for bubbly flow applications. Finally, the concluding remarks are presented in Section 4.4.

4.2 Choices of Deflation Vectors

In the literature of deflation, MG and DDM, several techniques are known to choose deflation vectors. Each field has its typical strategy to find the optimal choice. Below, we describe and discuss the approaches based on approximated eigenvector, recycling, subdomain, and multigrid deflation vectors. Most of the other alternatives known in the literature are related to these approaches.

4.2.1 Approximated Eigenvector Deflation

The deflation technique based on approximated eigenvectors is a popular approach, see, e.g., [25, 27, 122, 173].

In [173], an effective scheme is proposed based on physical deflation in which the deflation vectors are derived from the solutions of the original PDEs on specific subdomains. The resulting Z is sparse, whereas the corresponding vectors span the unfavorable eigenspace.

A general framework based on Flexible GMRES (FGMRES) is described in [27]. This framework includes techniques that are used to enhance the robustness of Krylov subspace methods, such as the deflated GMRES method, suggested in [80, 99], that aims at enhancing convergence by modifying the spectrum of the original matrix. The approach uses Ritz values and relies on solving generalized eigenvalue problems with much lower dimensions than A itself. If these dimensions become large, this approach is only successful to efficiently solve SPD systems with multiple right-hand sides of the form

$$Ax^{(i)} = b^{(i)}, \quad i = 1, 2, \ldots, \tag{4.2}$$

see also [122, Sect. 5]. In addition, the deflation matrix, P, obtained by the proposed deflation method is not sparse, resulting in possibly expensive computations with P, and large memory requirements. In addition, formulating and solving generalized eigenvalue problems are more straightforward for the classical GMRES algorithm than for CG, making this approach somewhat less suitable for CG-like methods. Finally, the approach seems unlikely to be helpful in realistic situations, because, for very large linear systems, removing a small number of eigenvalues out of many of them close to zero might have a limited effect on the solver, see [27].

A deflation technique applied to basic iterative methods based on Eq. (2.4), such as Gauss-Seidel or Jacobi iterations, is proposed in [25]. This deflation technique relies on computing so-called orthogonalized difference vectors and determining Schur vectors of a matrix with lower dimensions. It provides a distinct advantage for ill-conditioned systems, where the underlying scheme would either diverge or converge very slowly. Several numerical experiments in [25] demonstrate the efficiency of the method. However, for linear systems where the basic iterative scheme is already converging reasonably well, the accelerated convergence provided by deflation is not worth by considering the required extra work, see [25, Sect. 6].

In general, it can be observed that deflation based on approximated eigenvectors might be effective, but some additional efforts are needed to find these approximated eigenvectors, and sufficient memory should be available to store them. For relatively large problems, solving (generalized) eigenvalue problems may take a considerable time, especially in PCG methods, since PCG is based on short-term recurrences. Additionally, the number of approximated eigenvectors should be sufficiently small in order to restrict the extra work considering P and to reduce memory requirements, since these eigenvectors are usually dense. However, this is not always possible, since the spectrum might consist of many unfavorable eigenvalues in realistic problems.

4.2.2 Recycling Deflation

Related to approximated eigenvector deflation is solution and recycling deflation, see, e.g., [31, 112].

Solution deflation, proposed in [31], applies a subspace-projection extrapolation scheme for the starting vector generation of linear systems from implicit time integration schemes. The scheme yields optimal linear combinations from multiple available starting vectors. Similarly to eigenvector deflation, spectral components of the exact solution contained therein are optimally resolved which reduces the condition number. Suppose $\{x^{(1)}, x^{(2)}, \ldots, x^{(q-1)}\}$ is the set of solutions of the linear systems at time steps $l = 1, 2, \ldots, q-1$. Then, the deflation-subspace matrix can be defined as

$$Z = [x^{(1)} \; x^{(2)} \; \cdots \; x^{(q-1)}]. \tag{4.3}$$

Although numerical experiments in [31] emphasize the improved convergence of CG combined with solution deflation, we note that this approach has the drawback that Z is dense in general, and, additionally, it is not guaranteed that $\mathcal{R}(Z)$ indeed consists of any relevant spectral components.

Another approach is described in [112], where deflation vectors are based on recycling information of (previous) Krylov iterations in GMRES-like methods with relatively short-term recurrences. The resulting method is based on GMRES with deflated restarting vectors (i.e., GMRES-DR [100]), and GCR with a so-called optimal truncation (i.e., GCROT [35]). Recycling deflation is successful, if a sequence of linear systems (4.2), or even a sequence of the form

$$A^{(i)}x^{(i)} = b^{(i)}, \quad i = 1, 2, \ldots, \tag{4.4}$$

has to be solved. Note that we indeed have such a sequence (4.4) in bubbly flow simulations, see also Chapter 10. Like most of the approximate deflation methods, the recycling deflation approach requires a significant setup time to find the deflation vectors, especially for large grid sizes. Additionally, those vectors are usually dense, resulting in possible implementation and memory difficulties.

4.2.3 Subdomain Deflation

Another variant of deflation is subdomain deflation, where the deflation vectors are chosen in an algebraic way, see [92, 108, 170, 175] and Section 3.6. The computational domain is divided into several subdomains, where each subdomain corresponds to one or more deflation vectors. The resulting approach is often called ‘subdomain deflation’, and it is strongly related to approaches known in DDM, see, e.g., [126].

In [175], subdomain deflation is applied to the diffusion equation as given in Eq. (1.3). Assume that the computational domain, Ω, is divided into several subdomains, Ω_j, where each Ω_j corresponds to one deflation vector, consisting of ones for grid points in the interior of the discretized subdomain, Ω_{h_j}, and zeros for other grid points. Then, subdomain deflation is effective, if each subdomain, Ω_j, corresponds to exactly one constant part of the coefficient, ρ. In this case, the subspace spanned by the deflation vectors is proved to be almost equal to the eigenspace associated with the smallest eigenvalues. In addition, this approach converges as fast as the physical deflation approach [173] that has been described in Section 4.2.1, but it is more efficient due to a sparser structure of Z.

A detailed treatment of interface points of the subdomains, $\{\Omega_j\}$, can be found in [170]. It is shown that the subdomain technique is most successful if there is no overlap between subdomains. In this case, we have piecewise-constant, disjoint, and orthogonal deflation vectors.

Example 4.1. *Suppose that we have a 1-D computational domain, Ω, consisting of the grid points $x_1, \ldots, x_6$, that is divided into two subdomains such that $\Omega_{h_1} = \{x_1, x_2, x_3\}$ and $\Omega_{h_2} = \{x_4, x_5, x_6\}$. Then, we obtain*

$$Z = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}^T. \tag{4.5}$$

Each row of Z consists of exactly one nonzero, and the rows are clearly orthogonal, disjoint, and piecewise-constant.

Example 4.2. *A graphical representation of 2-D subdomains based on a grid size $n = 64^2$ can be found in Figure 4.1. Similar to Example 4.1, the corresponding deflation vectors can be obtained.*

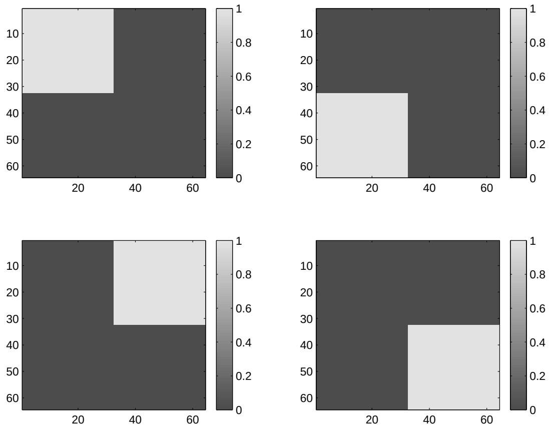


Figure 4.1: Representation of the 2-D subdomains with $k = 3$ in a square domain, Ω, consisting of $n = 64^2$ grid points.

Remark 4.2. *The underlying idea of choosing piecewise-constant deflation vectors is to approximate eigenvectors belonging to the smallest eigenvalues. Since the eigenvectors often represent components of the solution that is not necessarily linear, these piecewise-constant deflation vectors might only give a rough approximation. This motivates several authors (e.g., [54, 169]), to augment the deflation subspace with piecewise-linear (or even higher-order) subdomain deflation vectors. Following Example 4.1, we obtain (cf. Eq. (4.5))*

$$Z = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 2 & 3 \end{bmatrix}^T. \tag{4.6}$$

Hence, we have two deflation vectors per subdomain in 1-D. Generalization to 2-D and 3-D is straightforward for piecewise-constant deflation vectors, in contrast to piecewise-linear deflation vectors. In 2-D, one might take three vectors per subdomain: one constant and two piecewise-linear vectors in each spatial direction. Likewise, one can use four vectors per subdomain in 3-D. If the number of subdomains is large, then this would lead to a relatively large number of piecewise-linear deflation vectors, making the deflation technique more expensive than piecewise-constant subdomain deflation. In addition, the implementation of piecewise-linear vectors can be performed less efficient than piecewise-constant vectors, see Remark 4.3. Nevertheless, the employment of piecewise-linear deflation vectors may accelerate significantly the convergence of the iterative method, especially if the unfavorable eigenvectors have a linear form.

Remark 4.3.

- *In Example 4.1 and 4.2 and Remark 4.2, we have assumed that $\mathcal{N}(A) \not\subseteq \mathcal{R}(Z)$, otherwise Assumption 3.1 cannot be fulfilled. However, if, for instance, $\mathcal{N}(A) = \mathbf{1}_n$, then Z should be adapted to obey $\mathcal{N}(A) \not\subseteq \mathcal{R}(Z)$. This can be easily done by* *deleting one piecewise-constant deflation vector, while the resulting null space of PA remains the same. More details can be found in Section 4.3 and Chapter 5.*

- *According to Theorem 3.2, each column of Z in Example 4.1 and 4.2 and Remark 4.2 can be rescaled, while the resulting deflation matrices remain the same.*

- *The deflation method with piecewise-constant deflation vectors can be implemented very efficiently if A has some favorable properties, see Chapter 8.*

4.2.4 Multigrid and Multilevel Deflation Vectors

In the field of multigrid and multilevel methods, matrices Z and Z^T are known as the prolongation and restriction matrix, respectively, whereas a computation with P can be interpreted as a coarse-grid correction. A basic choice for Z (and corresponding Z^T) is a sparse matrix, given by

$$Z = \begin{bmatrix} \frac{1}{2} & 1 & \frac{1}{2} & 0 & \cdots & & & & \varnothing \\ 0 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & \cdots & & \\ & & & & & \ddots & & & \\ \varnothing & & & & \cdots & 0 & \frac{1}{2} & 1 & \frac{1}{2} \end{bmatrix}^T . \tag{4.7}$$

Many other prolongation and restriction matrices are known in the multigrid literature, see [126, 151, 178]. The columns of Z should approximate the slow-varying eigenvectors, often corresponding to small eigenvalues, in order to obtain an effective method. As also observed in (4.7), the number of deflation vectors is generally large. In this case, the resulting method requires a rather different approach to be efficient, compared to the methods as discussed so far. We focus on this issue in Chapter 9, and, in the meantime, this class of multigrid deflation vectors is not further considered.

4.2.5 Discussion of Different Approaches

Our approach of choice is subdomain deflation, because of the following facts.

- The resulting deflation-subspace matrix, Z, is sparse: each row consists of only one nonzero.

- The number of deflation vectors is relatively small: $k \ll n$.

- The deflation vectors appear to approximate the eigenspace associated with the unfavorable eigenvalues, resulting in faster convergence of the iterative process: this is explained for specific problem settings in [170, 175].

- The deflation vectors can be easily found: these vectors correspond to subdomains, which are straightforward to obtain.

- The approach is well-parallelizable: the deflation vectors are disjoint, so that it has excellent parallel properties (see Appendix F).

- The approach can easily be implemented in an existing PCG code: only a few additional steps should be incorporated in the PCG code, see Chapter 8.

As discussed above, not all of these criteria are satisfied for approximate eigenvector, solution recycling, and multigrid deflation. These criteria seem to be fulfilled for subdomain deflation, so that our main focus is on this approach.

There are still several difficulties left if one applies subdomain deflation vectors in, for example, bubbly flow applications.

- Subdomain deflation is often applied to linear systems with a nonsingular coefficient matrix, while our coefficient matrix of interest is singular.
- Subdomain deflation is used for problems with a fixed coefficient matrix, where the (density) coefficient in the original PDEs is often described explicitly. In our bubbly flow applications, the density field is given implicitly and the coefficient matrix varies in time.

In the next section, we examine whether subdomain deflation can still be applied taking the above difficulties into account. In addition, subdomain deflation vectors are applied successfully to several other problems, but a theoretical proof is still lacking. In [170], it is shown that the unfavorable eigenspace and the eigenspace spanned by the subdomain deflation vectors are almost the same, but the proof seems not to be fully correct and the connection to the corresponding spectra is not completely obvious. Ultimately, we have to show that the most unfavorable eigenvalues are effectively treated by using subdomain deflation vectors. This is further analyzed in the next section, where the main application is bubbly flows.

4.3 Application to Bubbly Flows

In this section, we adopt the problem setting as described in Section 1.3 and examine the optimal strategy to choose the deflation vectors for bubbly flow problems. For the sake of convenience, we restrict ourselves to ICCG and DICCG, so that the following assumption holds throughout this section.

Assumption 4.1. *M^{-1} is the IC(0) preconditioner based on a SPSD coefficient matrix, A.*

4.3.1 Preliminaries

Recall that $m \in \mathbb{N}$ denotes the number of bubbles in domain Ω, where Λ_0 and Λ_1 are the high- and low-density phases, respectively. In addition, Assumption 1.2 holds throughout this whole section. If the m bubbles are numbered, then we define $\Phi_i \subset \Omega$ as the domain corresponding to the i-th bubble, including its interface that may lie in Λ_0, for $i = 1, 2, \ldots, m$. Hence, we have

$$\Lambda_1 \subseteq \cup_{i=1}^{m} \Phi_i \quad \text{and} \quad \cap_{i=1}^{m} \Phi_i = \emptyset.$$

The discretized domain and the corresponding grid points are denoted by Ω_h and $\{x_i\}$, respectively. Moreover, Λ_{h_0}, Λ_{h_1} and Φ_{h_i} are the discretized variants of Λ_0, Λ_1 and Φ_i, respectively. For each $i = 1, \ldots, m$, we introduce the characteristic vector, $\phi_i \in \mathbb{R}^n$, associated with the i-th bubble, where each entry, $(\phi_i)_j$, is defined as follows:

$$(\phi_i)_j = \begin{cases} 1, & \text{if } \; x_j \in \Phi_{h_i}; \\ 0, & \text{elsewhere.} \end{cases}$$

Notice that the set of vectors $\{\phi_i\}_{i=1,\ldots,m}$ is linearly independent.

For $i \geq 2$, the eigenvalues of $M^{-1}A$, $\{\lambda_i\}$, appear to be of order 1, except for a few eigenvalues that are of order ε. The number of these $\mathcal{O}(\varepsilon)-$eigenvalues depends on the number of bubbles, m, see Proposition 4.1.

Proposition 4.1. *Let A and M^{-1} satisfy Assumptions 1.2 and 4.1, respectively. Suppose that $1 < m < n$. Then, the eigenvalues of $M^{-1}A$, $\{\lambda_i\}$, satisfy*

$$\lambda_i = \begin{cases} 0, & \textit{for } \; i = 1; \\ \mathcal{O}(\varepsilon), & \textit{for } \; i = 2, \ldots, m; \\ \mathcal{O}(1), & \textit{for } \; i = m+1, \ldots, n. \end{cases}$$

Moreover, for $i = 2, \ldots, m$, each eigenvector, v_i, corresponding to λ_i is constant in Λ_1.

Hence, $M^{-1}A$ has exactly $m - 1$ eigenvalues of $\mathcal{O}(\varepsilon)$, if there are m bubbles in Ω. Note that Proposition 4.1 still holds if bubbles touch the boundaries. Similar results are proven in literature, see, e.g., [174, Thm. 2.2], where the coefficient matrix is invertible and applications are given to steady porous-media flows.

Moreover, from Proposition 4.1, we obviously have that $M^{-1}A$ is ill-conditioned when $\varepsilon \ll 1$ and $m > 1$. This results in the fact that ICCG converges slowly. The deflation method could be adopted in order to effectively treat the $\mathcal{O}(\varepsilon)-$eigenvalues, resulting in a more effective method. We sometimes add the deflation-subspace matrix as a subscript to P to stress the choice of this deflation-subspace. For example, we have

$$P_Z := I - AQ_Z, \quad Q_Z := ZE_Z^{-1}Z^T, \quad E_Z := Z^T AZ. \tag{4.8}$$

Recall that Theorem 3.5 ensures that $M^{-1}P_Z A$ has a more favorable spectrum than $M^{-1}A$. Hence, the following corollary follows.

Corollary 4.1. *Let A and M^{-1} satisfy Assumptions 1.2 and 4.1, respectively. Suppose that P_Z is defined as in Eq. (4.8). Let $0 = \lambda_1 < \lambda_2 \leq \ldots \leq \lambda_n$ be the eigenvalues of $M^{-1}A$, and $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_n$ be the eigenvalues of $M^{-1}P_Z A$.*

- *If $\mathbf{1}_n \in \mathcal{R}(Z)$, then*

  $$\sigma(M^{-1}P_Z A) = \{0, \ldots, 0, \mu_{k+1}, \ldots, \mu_n\}, \tag{4.9}$$

  with $\lambda_1 \leq \mu_i \leq \lambda_n$ for $i = k+1, \ldots, n$.

- *If* $\mathbf{1}_n \notin \mathcal{R}(Z)$*, then*

$$\sigma(M^{-1}P_Z A) = \{0, \dots, 0, \mu_{k+2}, \dots, \mu_n\}, \tag{4.10}$$

with $\lambda_1 \leq \mu_i \leq \lambda_n$ *for* $i = k+2, \dots, n$*.*

Note that the deflation subspace is larger for $\mathbf{1}_n \notin \mathcal{R}(Z)$, than for $\mathbf{1}_n \in \mathcal{R}(Z)$. Of course, it depends on the deflation-subspace matrix, Z, in which way the eigenvalues, $\{\mu_i\}$, are distributed exactly; therefore, the success of DICCG is related to the choice of Z. Recall that Theorem 3.1 shows that the most straightforward choice for the columns of Z is the set of eigenvectors corresponding to the $\mathcal{O}(\varepsilon)-$eigenvalues, so that they are eliminated from the spectrum of $M^{-1}P_Z A$. As a consequence, the next corollary holds.

Corollary 4.2. *Let* A *and* M^{-1} *obey Assumptions 1.2 and 4.1, respectively. Suppose that* P_V *is the deflation matrix, where* V *denotes its deflation-subspace matrix. If* $V := [v_1 \; v_2 \; \cdots \; v_k]$ *consists of eigenvectors corresponding to all* $\mathcal{O}(\varepsilon)-$*eigenvalues of* $M^{-1}A$*, then* $\sigma(M^{-1}P_V A)$ *only consists of zeros and* $\mathcal{O}(1)-$*eigenvalues.*

Hence, the application of eigenvectors associated with $\mathcal{O}(\varepsilon)-$eigenvalues as deflation vectors is a good strategy to improve the convergence of the iterative process. In this case, it is sufficient to take $k = m$, when all $\mathcal{O}(\varepsilon)-$eigenvalues should be eliminated. Moreover, due to Assumption 1.2, v_1 is the constant eigenvector corresponding to the zero eigenvalue. As a consequence, v_1 may be omitted in V, so that $k = m-1$ deflation vectors would even be sufficient for the elimination of all $\mathcal{O}(\varepsilon)-$eigenvalues. Although the resulting deflation method can be very effective, this method based on a dense matrix V might be inefficient in use. In the next subsections, a perturbation analysis is carried out, resulting in deflation methods with appropriate choices for Z.

4.3.2 Inexact Eigenvector Deflation

Here, we analyze the deflation technique whose deflation vectors are based on inexact eigenvectors corresponding to the smallest eigenvalues of $M^{-1}A$.

Define $\bar{V} := [\bar{v}_1 \; \bar{v}_2 \; \cdots \; \bar{v}_k]$, where each $\bar{v}_i$ is an approximation of the exact eigenvector of $M^{-1}A$, v_i, i.e.,

$$\bar{v}_i := v_i + \delta_i, \quad \delta_i \in \mathbb{R}^n, \quad i = 1, 2, \dots, k, \tag{4.11}$$

where $k \leq m$. As mentioned earlier, it is desirable to construct a deflation method with $\bar{V}$ as deflation-subspace matrix that has the same favorable features as the deflation method based on V. At least, the resulting spectrum of $M^{-1}P_{\bar{V}}A$ with $k = m$ should not contain any eigenvalue of $\mathcal{O}(\varepsilon)$ anymore. In this case, each δ_i has to be chosen in such a way that the eigenvalues of the resulting matrix, $M^{-1}P_{\bar{V}}A$, satisfy

$$\mathcal{O}(\varepsilon) \ll \bar{\lambda}_i \leq \lambda_n, \quad \text{for } i = k+1, \dots, m, \tag{4.12}$$

where $\bar{\lambda}_i$ is an eigenvalue of $M^{-1}P_{\bar{V}}A$, and $\mathbf{1}_n \in \mathcal{R}(\bar{V})$ is assumed. However, as far as we know, no explicit results are given in the literature concerning the way in which v_i can be perturbed such that (4.12) is satisfied. In the remainder of this subsection, we give some propositions and theoretical results, which eventually result in heuristic rules for choosing δ_i. These are partly based on numerical experiments described in [141].

Define $\nu_i \in \mathbb{R}^n$ as the vector with the entries of $\bar{v}_i$, that correspond to the bubbles of phase Λ_1 including the interfaces, and let the other entries be zero, i.e., for each i, the entries of ν_i are defined by

$$(\nu_i)_j = \begin{cases} (\bar{v}_i)_j, & \text{if } \ x_j \in \Lambda_{h_1} \text{ or } x_j \in \partial\Lambda_{h_1}; \\ 0, & \text{otherwise,} \end{cases}$$

where $\partial\Lambda_{h_1}$ denotes the interfaces corresponding to Λ_{h_1}. Similarly to ν_i, we define $\tilde{\mathbf{1}}_n \in \mathbb{R}^n$ as follows:

$$(\tilde{\mathbf{1}}_n)_j = \begin{cases} 1, & \text{if } \ x_j \in \Lambda_{h_1} \text{ or } x_j \in \partial\Lambda_{h_1}; \\ 0, & \text{otherwise.} \end{cases}$$

Moreover, we suppose that the perturbations, $\{\delta_i\}$, satisfy Assumption 4.2.

Assumption 4.2. *Let A satisty Assumption 1.2. Then, each perturbation, $\delta_i \in \mathbb{R}^n$, with $i = 1, 2, \ldots, k$, is chosen such that*

1. $||A\delta_i||_2 = \mathcal{O}(\varepsilon)$ *for each* δ_i*;*
2. *entries corresponding to at least one bubble in Λ_1 are nonzero in $\bar{v}_i$;*
3. *the set $\{\tilde{\mathbf{1}}_n, \nu_1, \ldots, \nu_k\}$ is linearly independent.*

The first condition of Assumption 4.2 means that the norm of the perturbation is small, after premultiplication with A. The second condition says that it is not allowed to choose a perturbation, δ_i, in such a way that all entries of $\bar{v}_i$ corresponding to the bubbles are zero. The final condition means that each δ_i should be chosen in such a way that each vector of $\{\tilde{\mathbf{1}}_n, \nu_1, \ldots, \nu_k\}$ does not correspond to the same bubbles.

Example 4.3. *In our bubbly flow applications, a perturbation, $\delta_i \in \mathbb{R}^n$, satisfies Assumption 4.2 in, for instance, the following two cases:*

- *choose arbitrary entries for δ_i that corresponds to the high-density phase, Λ_0;*
- *choose arbitrary but identical entries in δ_i that corresponds to a complete bubble including its interface in the low-density phase, Λ_1, such that each δ_i corresponds to a different bubble.*

Next, let $V_\varepsilon \in \mathbb{R}^{n\times k}$ with $k \leq m-1$ be defined as a matrix consisting of columns being eigenvectors of $M^{-1}A$ corresponding to $\mathcal{O}(\varepsilon)-$eigenvalues. In addition, define $\bar{V}_\varepsilon \in \mathbb{R}^{n\times k}$ as a perturbation of V_ε, such that each δ_i obeys Assumption 4.2, i.e., each column of $\bar{V}_\varepsilon$ is the sum of the corresponding column of V_ε and δ_i satisfying Assumption 4.2 (cf. Eq. (4.11)). Then, it appears that Assumption 4.3 is always fulfilled in our experiments.

Assumption 4.3. *Let A fulfill Assumption 1.2. Suppose that $P_{\bar{V}_\varepsilon}$ and P_{V_ε} are deflation matrices, where $\bar{V}_\varepsilon$ and V_ε denote their deflation-subspace matrix, respectively. Let R be an $n \times n$ matrix. Then, $P_{\bar{V}_\varepsilon} A = P_{V_\varepsilon} A + R$ with $||Q||_2 = \mathcal{O}(\varepsilon)$.*

Furthermore, we define

$$(\tilde{\nu}_i)_j := \begin{cases} 1, & \text{if } \ (\nu_i)_j \neq 0; \\ 0, & \text{otherwise.} \end{cases}$$

Then, $\eta \in \{0, 1, \ldots, m\}$ denotes the maximum number of independent characteristic vectors, ϕ_{h_i}, such that

$$\phi_{h_i} \in \ \text{span}\ \{\tilde{\mathbf{1}}_n, \tilde{\nu}_1, \ldots, \tilde{\nu}_k\}.$$

In other words, η is the number of bubbles which are effectively captured by $\{\tilde{\nu}_i\}$. For instance, $\eta = k$ means that the number of deflation vectors is equal to the number of vectors corresponding to separate bubbles that can be constructed from the deflation subspace. Now, Theorem 4.1 follows easily.

Theorem 4.1. *Let A, $P_{\bar{V}_\varepsilon}$ and P_{V_ε} satisfy Assumption 4.3. Suppose that $\eta = k$. Then, for $j = 1, \ldots, n$, we have*

$$|\lambda_j(P_{\bar{V}_\varepsilon} A) - \lambda_j(P_{V_\varepsilon} A)| \leq \gamma, \quad \gamma = \mathcal{O}(\varepsilon). \tag{4.13}$$

Proof. Since $P_{\bar{V}_\varepsilon} A = P_{V_\varepsilon} A + R$ with $||R||_2 = \mathcal{O}(\varepsilon)$, the theorem follows immediately from Lemma A.10(iii). ☐

As a consequence of Theorem 4.1, perturbations that meet Assumption 4.2 do not significantly influence the spectrum of the deflated matrix, $P_{V_\varepsilon} A$. If, for instance, $P_{V_\varepsilon} A$ does not contain $\mathcal{O}(\varepsilon)-$eigenvalues, then neither does $P_{\bar{V}_\varepsilon} A$.

Unfortunately, Theorem 4.1 cannot be generalized to preconditioned deflated matrices, in contrast to what is claimed in the literature, see, e.g., [170]. In other words, (4.13) does not hold in general, if $M^{-1} P_{\bar{V}} A$ and $M^{-1} P_V A$ are substituted into $P_{\bar{V}} A$ and $P_V A$, respectively. Counterexamples can be easily found using numerical experiments. It turns out that the preconditioned variant of Theorem 4.1 only holds for the smallest eigenvalues, see Proposition 4.2.

Proposition 4.2. *Let A and M^{-1} fulfill Assumptions 1.2 and 4.1, respectively. Let $P_{\bar{V}_\varepsilon}$ and P_{V_ε} fulfill Assumption 4.3. Let $k \in \{1, 2, \ldots, m-1\}$ be given. Choose each $\delta_i \in \mathbb{R}^n$ such that Assumption 4.2 is fulfilled. Suppose that $\eta = k$ and $\mathbf{1}_n \notin \mathcal{R}(V_{\bar{\varepsilon}})$. Then,*

$$|\lambda_j(M^{-1} P_{\bar{V}_\varepsilon} A) - \lambda_j(M^{-1} P_{V_\varepsilon} A)| \leq \gamma, \quad \gamma = \mathcal{O}(\varepsilon),$$

for all $j = 1, \ldots, m$.

According to Proposition 4.2, $\mathcal{O}(\varepsilon)-$eigenvalues of $M^{-1} P_{V_\varepsilon} A$ are not significantly influenced by these perturbations, if each perturbation, δ_i, is chosen such that Assumption 4.2 is satisfied. However, Proposition 4.2 does not say anything about the other eigenvalues of $M^{-1} P_{V_\varepsilon} A$. Fortunately, it can be observed that the number of

$\mathcal{O}(\varepsilon)-$eigenvalues is equal for $M^{-1}P_{V_\varepsilon}A$ and $M^{-1}P_{\bar{V}_\varepsilon}A$, if $\eta = k$. In addition, a similar result follows for $\eta < k$. These results are stated in Conjecture 4.1.

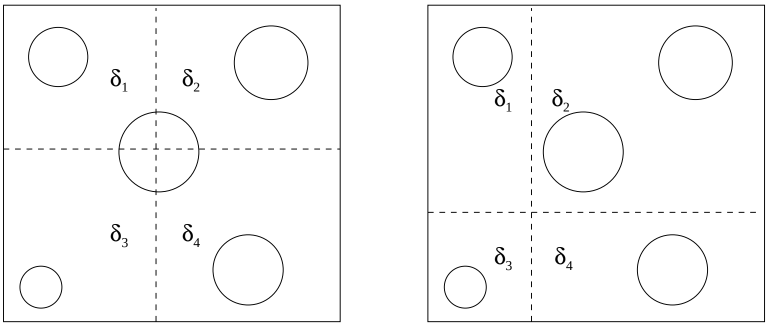
Conjecture 4.1. *Let A, M^{-1}, $P_{\bar{V}_\varepsilon}$ and P_{V_ε} be as given in Proposition 4.2. Let $k \in \{1, 2, \ldots, m-1\}$ be given and suppose that $\eta = k$ holds. Choose each $\delta_i \in \mathbb{R}^n$ such that Assumption 4.2 is fulfilled. Then, the number of $\mathcal{O}(\varepsilon)-$eigenvalues of $M^{-1}P_{\bar{V}_\varepsilon}A$ is equal to*

$$\begin{cases} m-\eta-1, & \text{if } \eta < m-1; \\ 0, & \text{if } \eta \geq m-1. \end{cases}$$

Moreover, if $\eta \geq k$, then the number of $\mathcal{O}(\varepsilon)-$eigenvalues of both $M^{-1}P_{\bar{V}_\varepsilon}A$ and $M^{-1}P_{V_\varepsilon}A$ is the same.

As a special case of Conjecture 4.1, we have that both $M^{-1}P_{\bar{V}_\varepsilon}A$ and $M^{-1}P_{V_\varepsilon}A$ do not contain any $\mathcal{O}(\varepsilon)-$eigenvalue, if $k = m-1$ and each δ_i has nonzero entries associated with at least one bubble. Example 4.4 shows another application of the conjecture.

Example 4.4. *Consider a 2-D bubbly flow problem with $m = 5$, see Figure 4.2. In this case, the spectrum of $M^{-1}A$ contains four $\mathcal{O}(\varepsilon)-$eigenvalues. The corresponding eigenvectors are taken as deflation vectors. Figure 4.2 presents two situations, where Ω is divided into four (deflation) subdomains, Ω_i, each corresponding to one perturbation vector, δ_i, whose entries are constant in this subdomain and zero elsewhere. In the case of Figure 4.2(a), none of the perturbations satisfy Assumption 4.2. Therefore, all four $\mathcal{O}(\varepsilon)-$eigenvalues of $M^{-1}A$ remain in the spectrum of $M^{-1}P_{\bar{V}_\varepsilon}A$. However, in the case of Figure 4.2(b), all perturbations meet Assumption 4.2, but obviously $\eta = 3$. According to Conjecture 4.1, the spectrum of $M^{-1}P_{\bar{V}_\varepsilon}A$ consists of exactly one $\mathcal{O}(\varepsilon)-$eigenvalue.*



(a) Wrong choice of subdomains: the middle bubble is not captured by one subdomain ($\eta = 0$).

(b) Good choice of subdomains: each bubble is in the interior of a subdomain ($\eta = 3$).

Figure 4.2: A 2-D example of a bubbly flow problem with $m = 5$ and two different situations for the perturbations, $\{\delta_i\}$.

Remark 4.4.

- *From Conjecture 4.1, we obtain the unexpected result that a good strategy for choosing an appropriate deflation vector,* $\bar{v}_i = v_i + \delta_i$*, is related to* $A\delta_i$*, rather than to* $M^{-1}A\delta_i$*.*

- *We refer to [123] for related results concerning the choice of deflation vectors. In that paper, two-level overlapping domain decomposition preconditioners with coarse spaces are studied by smoothed aggregation in iterative solvers for finite element discretizations of elliptic problems. Furthermore, similar observations as presented in this section are proven using functional analysis. It is a topic of future research to extend the theory given in [123] to the deflation strategies here.*

- *Conjecture 4.1 might be proven using ideas given in recent papers [5,81]. In these papers, theoretical bounds for eigenvalue approximations are presented, using so-called principal angles between subspaces spanned by eigenvectors associated with these (perturbed) eigenvalues. This is also left for future research.*

4.3.3 Level-Set Deflation Vectors

In the previous subsection, we have seen that exact eigenvectors are not required to eliminate the smallest nonzero eigenvalues from the spectrum of $M^{-1}P_Z A$. Conjecture 4.1 can serve as a guideline for approximating eigenvectors corresponding to $\mathcal{O}(\varepsilon)-$eigenvalues. This leads to strategies for choosing effective deflation vectors. We start with the so-called level-set deflation method that is described below.

By combining the results obtained in Example 4.3 and Conjecture 4.1, it can be concluded that eigenvectors, $\{v_i\}$, associated with the $\mathcal{O}(\varepsilon)-$eigenvalues are still well-approximated, if

- all entries of v_i corresponding to a bubble of Λ_1 including its interface $\partial\Lambda_1$ are scaled by a constant. Therefore, the value 1 can be chosen for the associated entries of the perturbed eigenvector $\bar{v}_i$, as it follows from Assumption 4.1 that all entries in a bubble are constant;

- the entries of v_i corresponding to the high-density phase Λ_0 can be perturbed arbitrarily. To obtain sparse perturbed eigenvectors, $\{\bar{v}_i\}$, these entries of v_i should be perturbed such that they become zero. In other words, $(\bar{v}_i)_j = 0$, if $x_j \in \Lambda_{h_0}$ and $x_j \notin \partial\Lambda_{h_1}$.

Hence, each v_i can be approximated well by a sparse $\bar{v}_i$ such that only the entries corresponding to bubbles are nonzero. From Conjecture 4.1, we also find that, if $\eta \geq m-1$, then all $\mathcal{O}(\varepsilon)-$eigenvalues of $M^{-1}A$ can be eliminated by choosing $k = m-1$. The requirement that $\eta \geq m-1$ is automatically fulfilled if we associate each $\bar{v}_i$ with one unique bubble, so that only the entries corresponding to a single bubble are nonzero. The resulting deflation-subspace matrix with $\{\bar{v}_i\}$ is denoted by

$W_{\text{L}} \in \mathbb{R}^{n \times k}$, and the resulting deflation method is called L-DICCG$-k$, where k is always chosen to be $m-1$. We define $\widetilde{W_{\text{L}}} \in \mathbb{R}^{n \times m}$ as $W_{\text{L}} \in \mathbb{R}^{n \times (m-1)}$ extended with a column associated with the excluded bubble. Later on, $\widetilde{W_{\text{L}}}$ is used to define the level-set-subdomain deflation variant.

Remark 4.5.

- *If some bubbles in Ω are very close to each other, then some grid points, $\{x_i\}$, might belong to the same nonzero entries of several columns of W_L. In this case, row sums of W_L can be larger than one, resulting in nondisjoint columns. This might require a more sophisticated implementation of the method. On the other hand, if disjoint vectors are imposed by choosing zero instead of one at some entries associated with the bubble interface, then the corresponding eigenvectors appear to be approximated badly. This results in slower convergence of the iterative process, see also Section 4.3.7.*

- *If the density field, ρ, is known explicitly, then L-DICCG$-k$ can be simply applied by locating the bubbles in Λ_1, and choosing one for the corresponding entries of the columns of W_L. However, ρ is often given implicitly in many applications, so that the method can only be used if an extra procedure exists that determines the bubbles explicitly. For example, the level-set approach [102, 110] is adopted to describe ρ implicitly in our applications, see, e.g., [143, 154, 156]. In Appendix B, this method is described concisely and an algorithm is presented for determining the bubbles from this level-set function.*

- *The name 'level-set deflation' suggests that this approach is only applicable if the density is described by the level-set function. This is however not the case. The approach can always be applied, as long as the bubbles can be described either implicitly or explicitly. More general names for the approach are 'coefficient-dependent deflation' or 'density deflation'.*

4.3.4 Subdomain Deflation

In bubbly flow problems where Ω contains many bubbles or the density field, ρ, is unknown or too complex, it is more appropriate to apply the deflation technique with subdomain deflation vectors instead of level-set deflation vectors. These subdomain vectors can be constructed without any knowledge of ρ and are described earlier in Sections 3.6 and 4.2.3. We denote the deflation method with subdomain vectors as S-DICCG$-k$, where k is the number of subdomains minus one. Moreover, $W_{\text{S}} \in \mathbb{R}^{n \times k}$ denotes the corresponding deflation-subspace matrix, which is defined in a more mathematical way below (cf. Section 3.6).

Let Ω be divided into open (equal) subdomains Ω_i, $i = 1, 2, \ldots, q+1$, such that $\overline{\Omega} = \cup_{i=1}^{q+1} \overline{\Omega}_i$ and $\Omega_i \cap \Omega_j = \emptyset$ for all $i \neq j$. The discretized subdomains are denoted by

Ω_{h_i}. For each Ω_{h_i}, we introduce a deflation vector, z_i, as follows:

$$(z_i)_j := \begin{cases} 0, & x_j \in \Omega_h \setminus \overline{\Omega}_{h_i}; \\ 1, & x_j \in \Omega_{h_i}. \end{cases}$$

Then, W_{S} is defined by

$$W_{\mathrm{S}} := [z_1 \; z_2 \; \cdots \; z_q], \tag{4.14}$$

so that $k = q$. Hence, W_{S} consists of disjoint (and, hence, orthogonal) piecewise-constant vectors, which is generally not the case for W_{L}. Moreover, note that W_{S} is usually less sparse and consists of more vectors than W_{L}, while the amount of work is $\mathcal{O}(n)$ for the construction of both W_{S} and W_{L}.

We could also extend W_{S} with an extra column, z_{q+1}, yielding

$$\widetilde{W_{\mathrm{S}}} := [z_1 \; z_2 \; \cdots \; z_q \; z_{q+1}],$$

which is used in Section 4.3.5. Note that each subdomain corresponds to one deflation vector, and we have the identity

$$\widetilde{W_{\mathrm{S}}} \mathbf{1}_{q+1} = \mathbf{1}_n. \tag{4.15}$$

Remark 4.6.

- *Eq. (4.15) is a useful property with respect to implementation and some proofs of theoretical results. However, recall that this might give rise to difficulties for approximating eigenvalues associated with the bubbles, especially if bubbles are very close to each other. In order to approximate the corresponding eigenvectors appropriately, some row sums of W_S should be larger than one, whereas this is not possible using the current definition of W_S.*
- *We show in Chapter 5 that the deflation matrix based on W_S and $\widetilde{W_S}$ are identical.*

Because of Conjecture 4.1, S-DICCG$-k$ can only be efficient if each subdomain, Ω_j, contains a part of at most one bubble. Otherwise, one or more $\mathcal{O}(\varepsilon)-$eigenvalues would remain in the spectrum of $M^{-1}P_{W_{\mathrm{S}}}A$. Hence, the efficiency of the deflation method with a fixed setting of subdomains depends on the number and the location of the bubbles in Ω. In order to ensure the efficiency of the method, the number of subdomains, k, should be taken relatively large, compared with the number of bubbles.

We find that subdomain deflation vectors also approximate other eigenvectors corresponding to small eigenvalues of $\mathcal{O}(1)$, since they appear to vanish from the spectrum of $M^{-1}P_{W_{\mathrm{S}}}A$, for sufficiently large k. In Section 4.3.6, we illustrate this in numerical experiments, but we already state this observation in Proposition 4.3.

Proposition 4.3. *Let A and M^{-1} fulfill Assumptions 1.2 and 4.1, respectively. Let W_S be as defined in (4.14). Then, for sufficiently large k, $\mathcal{R}(W_S)$ approximates the eigenspace corresponding to all $\mathcal{O}(\varepsilon)-$ and the smallest $\mathcal{O}(1)-$eigenvalues of $M^{-1}A$.*

Hence, S-DICCG$-k$ is able to eliminate both $\mathcal{O}(\varepsilon)-$ and $\mathcal{O}(1)-$eigenvalues from $M^{-1}A$. This means that, although S-DICCG$-k$ is usually more expensive per iteration than L-DICCG$-k$, the total computational cost can be much less due to faster convergence.

4.3.5 Level-Set-Subdomain Deflation

For a density field having a complex geometry, S-DICCG$-k$ with large k might encounter difficulties to treat all $\mathcal{O}(\varepsilon)-$eigenvalues effectively, although the smallest $\mathcal{O}(1)-$eigenvalues could be eliminated. On the other hand, L-DICCG$-k$ with $k = m-1$ easily deals with the $\mathcal{O}(\varepsilon)-$eigenvalues, while the $\mathcal{O}(1)-$eigenvalues are usually untouched. Therefore, it might be beneficial to combine both approaches. This new deflation variant is called LS-DICCG$-k$, and $W_{\text{LS}} \in \mathbb{R}^{n\times k}$ denotes its corresponding deflation-subspace matrix. The exact form of W_{LS} is defined below.

The most straightforward choice is to take $\widetilde{W}_{\text{LS}} := [\widetilde{W}_L, \widetilde{W}_S]$, so that the level-set-subdomain deflation-subspace matrix, W_{LS}, is equal to

- $\widetilde{W}_{\text{LS}}$, if $\mathbf{1}_n \notin \mathcal{R}(\widetilde{W}_{\text{LS}})$;
- $\widetilde{W}_{\text{LS}}$ without its last column, if $\mathbf{1}_n \in \mathcal{R}(\widetilde{W}_{\text{LS}})$.

In this case, W_{LS} consists of at most $q+m-1$ columns. If both $\widetilde{W}_L$ and $\widetilde{W}_S$ are known a priori, W_{LS} can be constructed immediately. Although the resulting approach might be effective, there are some obvious drawbacks:

- row sums of W_{LS} larger than one are inevitable, which makes the method less suitable for a parallel environment, and its implementation less efficient than the implementations of level-set or subdomain deflation;
- it is not guaranteed that W_{LS} has full rank.

Instead of this above straightforward choice of level-set-subdomain deflation, we choose for an alternative approach. First, we define some simple operations on matrices. The operation $\cup Y \in \mathbb{R}^{r\times 1}$, acting on $Y = [y_{i,j}] \in \mathbb{R}^{r\times s}$, means that a vector is created whose entries are the maximum entries of each row of Y, i.e., we have $(\cup Y)_i = \max_j y_{i,j}$ for each i, requiring $\mathcal{O}(r)$ flops. Moreover, for $Y_1 \in \mathbb{R}^{r\times s_1}$ and $Y_2 \in \mathbb{R}^{r\times s_2}$, the operation $Y_1 \cap Y_2 \in \mathbb{R}^{r\times s_3}$ means that a new matrix (or vector) is created, whose columns are equal to all possible componentwise multiplications between the columns of Y_1 and Y_2 that are nonzero. Note that $s_3 \leq s_1 s_2$ holds and the amount of work for this operation is at most $\mathcal{O}(rs_1s_2)$. We now define

$$\widetilde{W}_{\text{LS}} := [W_1, W_2], \tag{4.16}$$

with

$$W_1 := \widetilde{W}_{\text{S}} \cap (\mathbf{1}_n - \cup\widetilde{W}_{\text{L}}), \quad W_2 := \widetilde{W}_{\text{L}} \cap \widetilde{W}_{\text{S}}. \tag{4.17}$$

Hence, W_1 consists of all subdomain vectors of W_{S} where the entries corresponding to Λ_1 are zero. Moreover, W_2 consists of columns whose entries correspond to the bubbles divided by the subdomains of $\widetilde{W}_{\text{S}}$. Now, the level-set-subdomain deflation-subspace matrix, W_{LS}, is equal to

- $\widetilde{W}_{\text{LS}}$, if $\mathbf{1}_n \notin \mathcal{R}(\widetilde{W}_{\text{LS}})$;
- $\widetilde{W}_{\text{LS}}$ without its last column, if $\mathbf{1}_n \in \mathcal{R}(\widetilde{W}_{\text{LS}})$.

As noted earlier, both W_{LS} and $\widetilde{W}_{\text{LS}}$ lead to the same deflation matrix. Example 4.5 illustrates their construction.

Example 4.5. *Let*

$$\widetilde{W}_S = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}^T, \quad \widetilde{W}_L = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}^T$$

be the deflation-subspace matrices corresponding to S-DICCG-1 and L-DICCG-1, respectively. Then, this yields

$$\cup\widetilde{W}_L = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}^T, \quad \mathbf{1}_n - \cup\widetilde{W}_L = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}^T,$$

resulting in

$$W_1 = \widetilde{W}_S \cap (\mathbf{1}_n - \cup\widetilde{W}_L) = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}^T,$$

and

$$W_2 = \widetilde{W}_L \cap \widetilde{W}_S = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}^T.$$

This implies

$$\widetilde{W}_{LS} = [W_1, W_2] = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}^T.$$

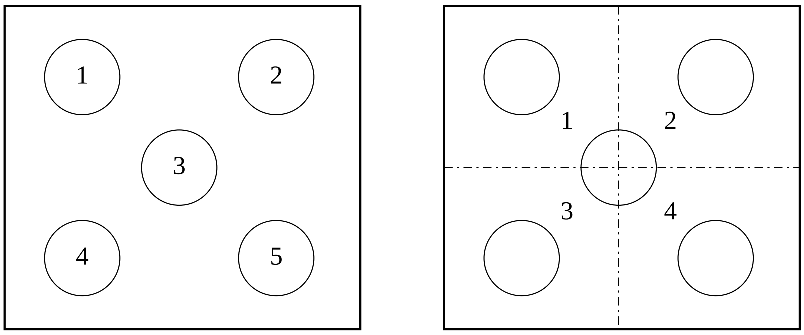
As $\mathbf{1}_n \in \mathcal{R}(\widetilde{W}_{LS})$, the level-set-subdomain deflation-subspace matrix W_{LS} is equal to

$$W_{LS} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T.$$

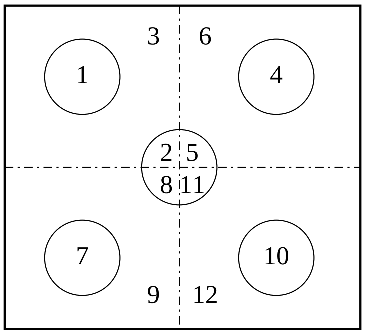
Remark 4.7.

- *If k is general, L-DICCG$-k$, S-DICCG$-k$ and LS-DICCG$-k$ are often denoted by L-DICCG, S-DICCG and LS-DICCG, respectively.*
- *We have*

$$\begin{cases} \mathcal{R}(\widetilde{W}_S) & \subseteq & \mathcal{R}(\widetilde{W}_{LS}); \\ \mathcal{R}(\widetilde{W}_L) & \subseteq & \mathcal{R}(\widetilde{W}_{LS}), \end{cases}$$



(a) L-DICCG−4 ($k = m - 1$). (b) S-DICCG−3 ($k = q = 3$).



(c) LS-DICCG−11 ($k < m(q + 1)$).

Figure 4.3: A 2-D bubbly flow problem with $m = 5$, which illustrates the level-set, subdomain and level-set-subdomain deflation technique.

which means that the deflation subspace of LS-DICCG contains the deflation subspaces of both L-DICCG and S-DICCG.

- *The construction of W_{LS} requires at most $\mathcal{O}(nms)$ flops. In addition, compared with L-DICCG and S-DICCG, LS-DICCG requires more deflation vectors, so an iteration of this hybrid method is more expensive due to the more sophisticated coarse solves. However, since the spectrum of $M^{-1}P_{W_{LS}}A$ is more favorable, convergence can be much faster, resulting in a possibly lower total computational cost of LS-DICCG.*

- *The row sum of W_{LS} is at most one. If $\mathbf{1}_n \notin \mathcal{R}(\widetilde{W}_{LS})$, then we even have $\widetilde{W}_{LS}\mathbf{1}_k = \mathbf{1}_n$. Consequently, LS-DICCG can be easily parallelized and the method can be implemented very efficiently.*

- *Level-set deflation might be combined with other (deflation) techniques to end up with more effective hybrid methods. In fact, level-set deflation is used to remove the effects of the bubbles, so that it could be combined with any effective solver (such as standard multigrid or methods based on fast Fourier transforms) to tackle Poisson problems with a constant coefficient, i.e., problems without bubbles. This is left for future research.*

We end this section with Example 4.6, that illustrates the deflation approaches proposed in this section.

Example 4.6. *Consider a 2-D bubbly flow problem with* $m = 5$. *The associated deflation vectors in L-DICCG−4, S-DICCG−3 and the resulting LS-DICCG−11 are depicted graphically in Figure 4.3.*

4.3.6 Numerical Experiments

After presenting possible choices of deflation vectors applied to bubbly flows, we show their efficiency in 2-D numerical experiments. We test the three deflation approaches L-DICCG, S-DICCG and LS-DICCG, and compare them with ICCG. The computations are performed on a Pentium 4 (2.80 GHz) computer with a memory capacity of 1GB using MATLAB. Since it is easy to use the sparse implementation for matrices and vectors in MATLAB, we are able to measure fairly the computing time that is required for the whole iteration process of the compared methods [1].

First, we consider briefly the test problem without bubbles, that is the Poisson problem with a constant coefficient, so that $\varepsilon = 1$. Next, we treat the test problem with bubbles, where we vary the grid size, n, the density contrast, $\epsilon = \frac{1}{\varepsilon}$, and the number of bubbles, m. The employed geometry of the density field based on $m = 5$ can be found in Figure 1.3. The linear system, $Ax = b$, is solved, where the termination criterion is based on (3.27) with $\delta = 10^{-7}$.

Test Problem with a Constant Coefficient

For this specific test problem without bubbles in the domain, S-DICCG is the only method that can be applied. The results for the problem with $\epsilon = 1$ and $n = 16^2$ are presented in Table 4.1.

Method	# It.
ICCG	23
S-DICCG−3	22
S-DICCG−15	15
S-DICCG−63	10

Table 4.1: Results for the Poisson problem with $\epsilon = 1$ and $n = 16^2$. '# It' means the number of required iterations for convergence.

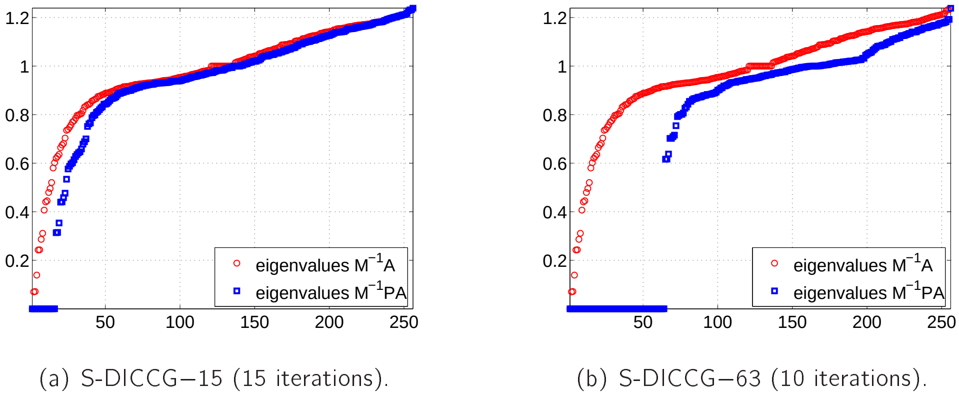
From Table 4.1, it can be noticed that S-DICCG reduces the number of iterations, compared with ICCG. The corresponding eigenvalues of $M^{-1}A$ and $M^{-1}P_{W_S}A$ can be found in Figure 4.4.

From both subplots of Figure 4.4, we observe that small $\mathcal{O}(1)-$eigenvalues of $M^{-1}A$ are eliminated from the spectrum of $M^{-1}P_{W_S}A$ (cf. Proposition 4.3), whereas

---

[1] This is including the computation of AZ and E, but excluding the construction of Z, since it cannot be done efficiently in MATLAB. However, the comparison is still fair, since the computational cost to construct Z is negligible by considering the flop counts given in previous subsections.

the large eigenvalues remain in the spectrum. Increasing the number of deflation vectors results in the elimination of more small eigenvalues. This can be explained by the fact that the corresponding eigenvectors are relatively smooth, so that they can be well-approximated by the subdomain deflation vectors. Other eigenvectors corresponding to larger eigenvalues of $M^{-1}A$ do not have a smooth behavior, and, therefore, these are more difficult to approximate by using these vectors, see [141, Sect. 10.1] for more details.



(a) S-DICCG−15 (15 iterations). (b) S-DICCG−63 (10 iterations).

Figure 4.4: Eigenvalues of $M^{-1}A$ and $M^{-1}P_{W_S}A$ for S-DICCG, applied to the Poisson problem with $\epsilon = 1$ and $n = 16^2$.

Test Problem with Varying Grid Sizes

Next, we perform a numerical experiment for the Poisson problem with $m = 5$, $\epsilon = 10^6$, and varying grid sizes. The convergence results, including the computational cost, can be found in Table 4.2.

		$n = 16^2$		$n = 32^2$		$n = 64^2$	
Deflation Method	k	# It.	CPU	# It.	CPU	# It.	CPU
ICCG	–	39	0.04	82	0.53	159	10.92
S-DICCG−k	3	37	0.12	80	0.67	194	14.01
	15	36	0.07	97	0.80	193	13.82
	63	19	0.11	16	0.20	26	2.14
L-DICCG−k	4	17	0.09	37	0.37	75	6.17
LS-DICCG−k	11	14	0.07	30	0.29	54	4.08
	35	10	0.08	21	0.32	40	3.05
	83	–	–	15	0.20	25	2.05

Table 4.2: Results for the Poisson problem with $m = 5$, $\epsilon = 10^6$, and varying grid sizes, n. '# It' means the number of required iterations, and 'CPU' is the corresponding computational time in seconds.

For all grid sizes, it can be observed that S-DICCG−63 is very efficient, compared with ICCG. This is in contrast to S-DICCG−3 and S-DICCG−15, whose performance is comparable to ICCG. The explanation is that Assumption 4.2 is fulfilled only for

$k = 63$, and, according to Conjecture 4.1, the spectrum associated with S-DICCG−63 does not contain $\mathcal{O}(\varepsilon)$−eigenvalues, see also Section 4.3.7. For the other two cases, S-DICCG−3 and S-DICCG−15, some deflation subdomains consist of parts of several bubbles, and, therefore, the corresponding deflation vectors do not satisfy Assumption 4.2. Hence, the number of $\mathcal{O}(\varepsilon)$−eigenvalues remains the same after applying subdomain deflation. Furthermore, note that ICCG requires significantly fewer iterations than S-DICCG−3 and S-DICCG−15 in the case of $n = 64^2$. This is caused by the fact that the corresponding residuals show erratic behavior with relatively large bumps, so that a small round-off error during the iteration process can lead to significant differences in convergence, see [141, Sect. 10.3] for more details.

From Table 4.2, we observe that L-DICCG reduces significantly the number of iterations. It is an efficient method, since it requires only four deflation vectors. We find that LS-DICCG performs very well in all cases, but S-DICCG and LS-DICCG become comparable for sufficiently large k.

Remark 4.8. *If there are some limitations with respect to the number of deflation vectors due to memory capacity, then LS-DICCG would converge faster than S-DICCG. Suppose that only $k < 50$ deflation vectors can be kept in memory, than the fastest method is LS-DICCG−35 according to Table 4.2.*

Test Problem with Varying Density Contrasts

We fix $m = 5$ and $n = 64^2$, whereas the density contrast, ϵ, is varied in the next numerical experiment. The results of this experiment are presented in Table 4.3.

		$\epsilon = 10^3$		$\epsilon = 10^6$	
Deflation Method	k	# It.	CPU	# It.	CPU
ICCG	–	118	8.12	159	10.92
S-DICCG−k	3	134	9.79	194	14.01
	15	131	9.60	193	13.82
	63	26	2.31	26	2.14
L-DICCG−k	4	74	5.98	75	6.17
LS-DICCG−k	11	54	4.05	54	4.08
	35	40	3.08	40	3.05
	83	25	2.46	25	2.41

Table 4.3: Results for the Poisson problem with $m = 5$, $n = 64^2$, and varying density contrast, ϵ.

From Table 4.3, we see that ICCG requires more iterations and CPU time for larger ϵ, due to the presence of $\mathcal{O}(\varepsilon)$−eigenvalues in the corresponding spectrum. This observation does not hold for L-DICCG and LS-DICCG, which is a favorable feature of these methods, and it confirms the theory given in the previous section. For sufficiently large k, it can be noticed that S-DICCG is also insensitive to ϵ. Furthermore, it can again be observed that S-DICCG−3 and S-DICCG−15 converge more slowly than ICCG, whereas S-DICCG−63 is faster in this experiment.

Test Problem with Varying Number of Bubbles

We consider the Poisson problem with $\epsilon = 10^6$, $n = 64^2$, and a varying number of bubbles, m. The results of this experiment can be found in Table 4.4.

We observe that ICCG needs more iterations for larger m. This can be explained by Proposition 4.1, which states that an increase of m leads to more $\mathcal{O}(\varepsilon)-$eigenvalues. For L-DICCG, LS-DICCG, and S-DICCG with sufficiently large k, we see that their performance depends less on m, which is a favorable feature of these deflation approaches.

Notice that L-DICCG-0 is undefined, so this method cannot be applied for $m = 1$. Furthermore, L-DICCG converges in fewer iterations for increasing $m > 1$. Finally, S-DICCG converges again slower than ICCG for $k \leq 15$.

	$m = 1$			$m = 2$			$m = 5$		
Deflation Method	k	# It.	CPU	k	# It.	CPU	k	# It.	CPU
ICCG	–	89	6.13	–	104	7.20	–	159	10.92
S-DICCG$-k$	3	96	7.39	3	69	5.13	3	194	14.01
	15	52	3.97	15	64	4.79	15	193	13.82
	63	26	2.14	63	27	2.16	63	26	2.14
L-DICCG$-k$	0	–	–	1	79	5.79	4	75	6.17
LS-DICCG$-k$	7	67	5.30	6	65	5.11	11	54	4.08
	19	41	3.14	24	42	3.22	35	40	3.05
	67	26	2.50	72	26	2.11	83	25	2.05

Table 4.4: Results for the Poisson problem with $\epsilon = 10^6$, $n = 64^2$, and varying number of bubbles, m.

In contrast to ICCG, all approaches of the deflation method (except for S-DICCG with relatively small k) hardly depend on m. This implies that, for problems with an increasing number of bubbles, the deflation method becomes more and more superior to ICCG.

4.3.7 Analysis of Small Eigenvalues

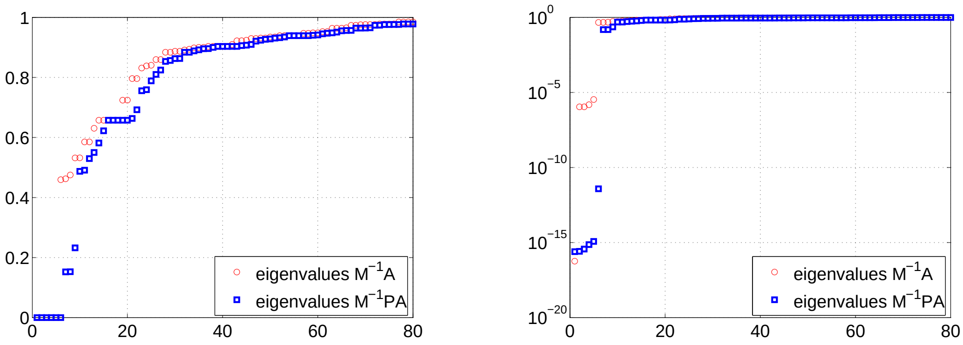
In this subsection, we present some spectral information corresponding to the deflation approaches. These are based on the numerical experiments described in Section 4.3.6.

Level-Set Deflation

In Section 4.3.6, we have noticed that L-DICCG-4 reduces significantly the number of iterations, compared with ICCG. Figure 4.5 shows the corresponding spectra. Because we concentrate on small eigenvalues, only the 80 smallest eigenvalues of each spectrum are presented.

First, it can be noticed in Figure 4.5(a), that $\mathcal{O}(1)-$eigenvalues are approximately the same for ICCG and L-DICCG-4. In Figure 4.5(b), we see that all $\mathcal{O}(10^{-6})-$eigenvalues are removed from $M^{-1}P_{W_L}A$. However, eigenvalues in the vicinity of 0.2 appear, see Figure 4.5(a).

As noticed in Sections 4.3.2 and 4.3.3, interfaces of bubbles should contribute to the deflation vectors. If these interfaces are excluded in the level-set deflation vectors, then the convergence of L-DICCG−4 is significantly slower. In this case, it appears that $\mathcal{O}(10^{-6})-$eigenvalues are effectively eliminated, but with the drawback that eigenvalues between ε and 1 appear.



(a) Normal scale: L-DICCG−4.

(b) Logarithmic scale: L-DICCG−4.

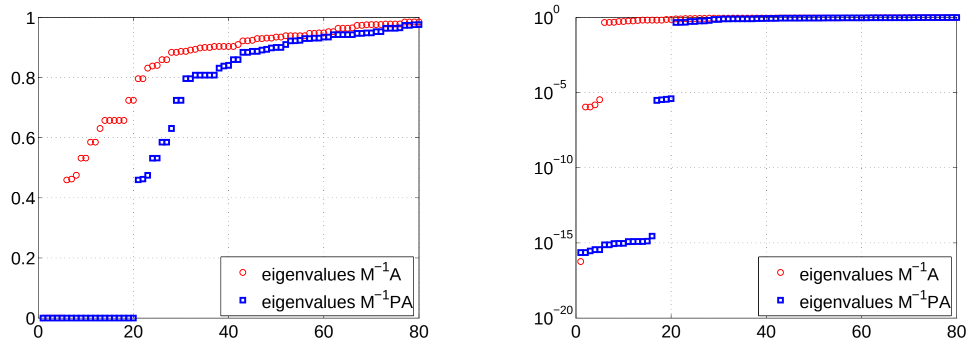
Figure 4.5: Eigenvalues of both $M^{-1}A$ and $M^{-1}P_{W_L}A$ corresponding to L-DICCG−4, for the Poisson problem with $\epsilon = 10^6$ and $n = 16^2$.

Subdomain Deflation

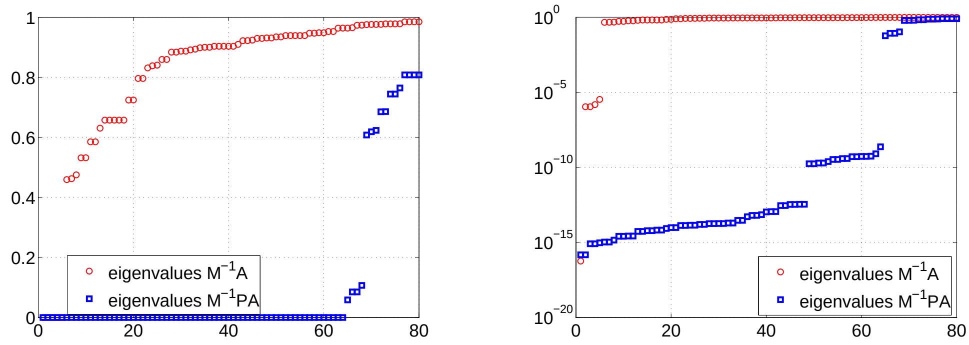
In Section 4.3.6, we have seen that S-DICCG−15 does not give any improvement of the convergence, whereas S-DICCG−63 is very efficient, compared with ICCG. This can be understood by considering their spectral plots, see Figure 4.6 where the 80 smallest eigenvalues of each spectrum are depicted.

It appears that values below 10^{-8} can be interpreted as zero eigenvalues in Figure 4.6. Then, in the case of S-DICCG−15 (see Figure 4.6(b)), it can be observed that none of the $\mathcal{O}(10^{-6})-$eigenvalues of $M^{-1}A$ are eliminated after deflation, since they remain in the spectrum of $M^{-1}P_{W_S}A$. Moreover, S-DICCG−63 converges very fast, because the $\mathcal{O}(10^{-6})-$eigenvalues vanish from the spectrum, see Figure 4.6(d). Apparently, only for sufficiently large k, each deflation subdomain consists of a part of at most one bubble. Hence, the smallest eigenvalues can be eliminated, which confirms Conjecture 4.1.

With respect to the small $\mathcal{O}(1)-$eigenvalues, we observe in Figure 4.6(a) that they are approximately the same for ICCG and S-DICCG−15. Moreover, for the case of S-DICCG−63 (Figure 4.6(c)), it can be seen that the smallest $\mathcal{O}(1)-$eigenvalues do not appear in the spectrum of $M^{-1}P_{W_S}A$, which is similar to the case of L-DICCG−4 (cf. Figure 4.5(a)). However, some other small eigenvalues around 0.1 can be noticed in Figure 4.6(c). Roughly speaking, the eliminated $\mathcal{O}(10^{-6})-$eigenvalues give rise to small eigenvalues of order 10^{-1}. Apparently, the eigenvectors associated with $\mathcal{O}(10^{-6})-$eigenvalues are not approximated accurately enough by the subdomain deflation vectors, even if we increase k. This might be caused by the fact that, by definition,



(a) Normal scale: S-DICCG−15. (b) Logarithmic scale: S-DICCG−15.



(c) Normal scale: S-DICCG−63. (d) Logarithmic scale: S-DICCG−63.

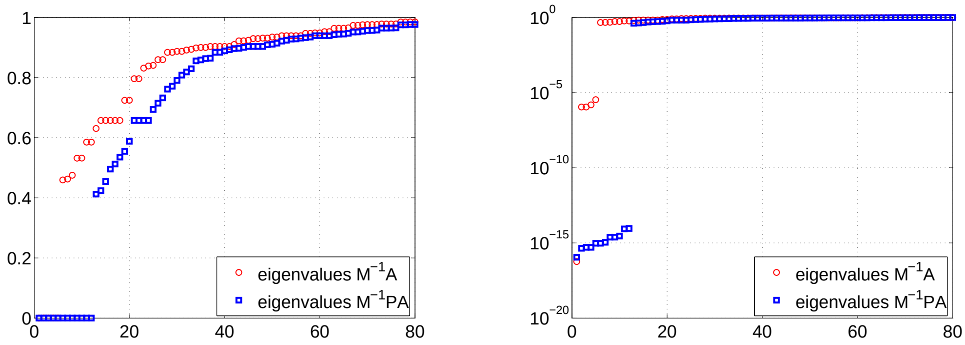
Figure 4.6: Eigenvalues of $M^{-1}A$ and $M^{-1}P_{W_S}A$ corresponding to S-DICCG, for the Poisson problem with $\epsilon = 10^6$ and $n = 16^2$.

the subdomain deflation vectors have the unfavorable property that they are disjoint. This can be remedied by using LS-DICCG instead of S-DICCG, see Section 4.3.7.

Level-Set-Subdomain Deflation

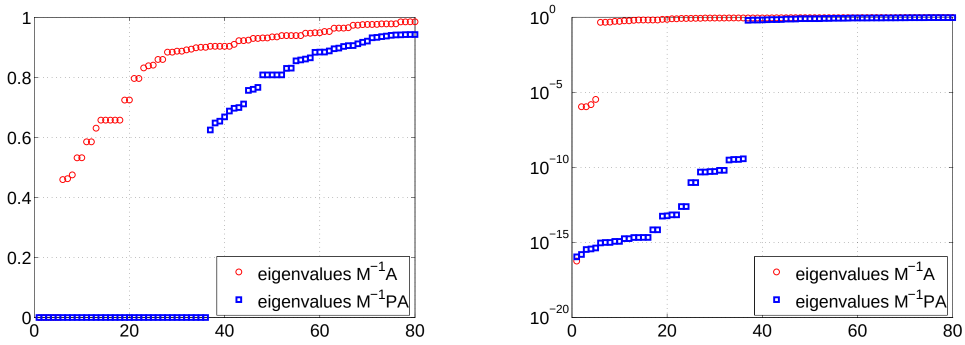
As observed in Section 4.3.6, LS-DICCG performs very well for all k. The related spectral plots can be found in Figure 4.7.

From Figure 4.7(a) and 4.7(b), we see that only $\mathcal{O}(10^{-6})-$eigenvalues disappear and all $\mathcal{O}(1)-$eigenvalues remain in the spectrum in the case of LS-DICCG−11. In Figure 4.7(c) and 4.7(d), it can be observed that both $\mathcal{O}(10^{-6})-$ and the smallest $\mathcal{O}(1)-$eigenvalues do not appear in the spectrum corresponding to LS-DICCG−35. More importantly, in contrast to the cases of S-DICCG and L-DICCG, the elimination of $\mathcal{O}(10^{-6})-$eigenvalues by LS-DICCG does not give rise to new eigenvalues between ε and 1. This is a favorable feature of level-set-subdomain deflation.



(a) Normal scale: LS-DICCG−11.

(b) Logarithmic scale: LS-DICCG−11.



(c) Normal scale: LS-DICCG−35.

(d) Logarithmic scale: LS-DICCG−35.

Figure 4.7: Eigenvalues of both $M^{-1}A$ and $M^{-1}P_{W_{LS}}A$ corresponding to LS-DICCG, for the Poisson problem with $\varepsilon = 10^6$ and $n = 16^2$.

4.4 Concluding Remarks

Some strategies for choosing deflation vectors are reviewed in this chapter: approximate eigenvector, recycling, subdomain and multigrid deflation vectors. Each of them has its own advantages and drawbacks. The most favorable choice strongly depends on many aspects, such as the problem setting, specific application, a priori knowledge of (spectral) information, linear systems to be solved, used Krylov solver, and maximum number of allowed deflation vectors. Hence, there is no ultimate strategy that always performs best for all cases, although we advocate that subdomain deflation is often the most appropriate choice.

In the second part of this chapter, we present some spectral analysis to deflation with inexact eigenvectors, which leads to strategies for choosing the best deflation vectors in bubbly flow applications. The main result is that eigenvectors corresponding to the smallest eigenvalues can be perturbed in such a way that they become sparse, which motivates the use of subdomain deflation. Based on this result, two other deflation approaches are introduced and discussed. The first approach is the level-set deflation method, where the sparse deflation vectors are based on the geometry of the density field. The second approach, which is the level-set-subdomain deflation method, combines original subdomain and level-set deflation, and has the advantages of both approaches.

In the numerical experiments, we compare the proposed deflation approaches for bubbly flows. In most test cases, all of them perform very well compared with ICCG. In addition, they are insensitive to large density contrasts and the number of bubbles. Subdomain deflation is only efficient for a sufficiently large number of subdomains. In this case, not only the smallest eigenvalues corresponding to the bubbles are eliminated, but also other small eigenvalues. Moreover, level-set deflation eliminates the smallest eigenvalues corresponding to bubbles at low cost, but leaves the other eigenvalues more or less untouched. For both of these methods, the elimination of the smallest eigenvalues may result in a spectrum that consists of eigenvalues which are obviously smaller than those of the main cluster. It appears that level-set-subdomain deflation does not have this drawback. Therefore, it is an efficient method, although the work per iteration and the work to create the deflation vectors can be significantly larger than for the other two approaches. However, we note that, if the number of deflation vectors is sufficiently large, then the difference in performance between subdomain and level-set-subdomain deflation is small. In addition, subdomain deflation has the important advantages that it can be used as a blackbox method without any knowledge of the density field, and can be implemented and parallelized in a straightforward way. Hence, subdomain deflation is our method of choice and it is frequently used in the remainder of this thesis.

Finally, a topic for future research is improving the proposed deflation approaches. Since level-set deflation is used to treat the bubbles effectively, it might be possible to combine it with any effective solver (such as standard multigrid or methods based on fast Fourier transforms) for the standard Poisson problem with a constant coefficient in order to obtain a powerful method, that could effectively deal with bubbly flow problems.

Chapter 5

Subdomain Deflation applied to Singular Matrices

5.1 Introduction

In Chapter 3, new theoretical results have been presented for the deflation method applied to singular coefficient matrices. In this chapter, we deal with the issue of deflation and singularity in more detail. Although many results presented here can be generalized to deflation with a general Z, we restrict ourselves to subdomain deflation for convenience, see Definition 5.1.

Definition 5.1. *Let the open domain, Ω, be divided into subdomains, Ω_j, $j = 1, 2, \ldots, q+1$, such that $\overline{\Omega} = \cup_{j=1}^{q+1} \overline{\Omega}_j$ and $\Omega_i \cap \Omega_j = \emptyset$ for all $i \neq j$. The discretized domain and subdomains are denoted by Ω_h and Ω_{h_j}, respectively. Then, for each Ω_{h_j} with $j = 1, 2, \ldots, q+1$, a deflation vector, z_j, is defined as follows:*

$$(z_j)_i := \begin{cases} 0, & x_i \in \Omega_h \setminus \Omega_{h_j}; \\ 1, & x_i \in \Omega_{h_j}, \end{cases} \tag{5.1}$$

where x_i is a grid point in the discretized domain, Ω_h. Then, for $1 < j \leq q$, the deflation-subspace matrices are defined as

$$\begin{cases} Z_j & := & \left[z_1 \ \cdots \ z_j\right]; \\ \widehat{Z}_j & := & \left[Z_{j-1}, \ z_0\right], \end{cases}$$

where $z_0 = \mathbf{1}_n$. In addition, we define $Z_1 := z_1$ and $\widehat{Z}_1 := z_0$.

By construction, subdomain deflation vectors are disjoint and sparse. Moreover, Property 5.1 can be derived from Definition 5.1.

Property 5.1. *Let Z_j, $\widehat{Z}_j$ and z_j be as given in Definition 5.1. Then, the following statements hold:*

(i) $Z_k \mathbf{1}_k = \mathbf{1}_n$;

(ii) $Z_{k-1}\mathbf{1}_{k-1} \neq \widehat{Z}_k \mathbf{1}_k \neq \mathbf{1}_n$;

(iii) $Z_{k-1}^T \mathbf{e}_n^{(n)} = \mathbf{0}_{k-1}$;

(iv) $\widehat{Z}_{k-1}^T \mathbf{e}_n^{(n)} = \mathbf{1}_n$;

(v) $\widehat{Z}_k \mathbf{e}_k^{(k)} = z_0$;

(vi) $Z_j \mathbf{e}_j^{(i)} = z_i, \quad 1 < j \leq k, \quad 1 \leq i \leq j$;

(vii) $\widehat{Z}_j \mathbf{e}_j^{(i)} = z_i, \quad 1 < j < k, \quad 1 \leq i \leq j$.

Subscripts corresponding to matrices will be omitted, if we deal with general matrices without specified dimensions.

In all previous chapters, we have performed the analysis and computations based on a singular coefficient matrix, A, from the linear system (see Eq. (1.1))

$$Ax = b, \quad A \in \mathbb{R}^{n \times n}. \tag{5.2}$$

However, in many CFD packages, one imposes an invertible A, denoted by $\bar{A}$, see also [17,77,113]. This makes the solution, x, of (5.2) unique, which might be advantageous in computations:

- direct solution methods, such as Gaussian elimination, might have some difficulties to solve (5.2) with a singular A;
- linear system (5.2) might be inconsistent as a result of rounding errors, whereas the linear system with $\bar{A}$ is always consistent;
- the deflation technique requires an invertible matrix $E = Z^T A Z$. This is guaranteed for any full-rank $Z = [z_1 \ \cdots \ z_k]$ if $\bar{A}$ is used.

One common way to force invertibility of A is to replace its last entry, $a_{n,n}$, by $\bar{a}_{n,n} = (1+\sigma)a_{n,n}$ with $\sigma > 0$. In fact, a Dirichlet boundary condition is imposed at one point of the domain, Ω. This modification results in an invertible linear system,

$$\bar{A}x = b, \quad \bar{A} = [\bar{a}_{i,j}] \in \mathbb{R}^{n \times n}, \tag{5.3}$$

where $\bar{A}$ is SPD. In practice, it appears that the condition number, κ, is relatively large, especially if σ is close to 0, see Lemma 5.1(ii). Hence, solving (5.3) with the CG method typically shows slow convergence, see also [77, Sect. 4] and [113, Sect. 6.7]. Similarly, this fact holds for the ICCG method as well, see Section 5.5.1. In this chapter, a comparative study is performed on the deflation methods based on (5.2) and (5.3).

We have assumed in the previous chapters that Z does not consist of components of the null space of A, $\mathcal{N}(A)$, in order to ensure that E is nonsingular (see

Assumption 3.1). However, $\mathcal{N}(A)$ is not always known a priori, and, additionally, it is sometimes not practical to exclude some components from Z. In this chapter, we investigate this issue in more detail. It is derived that there exists a strong connection between deflation methods based on a singular and nonsingular matrix E.

The main questions that are answered in this chapter are:

- can the deflation method based on a singular E always be transformed into a method with a nonsingular E, while A is still singular?
- is it possible to transform the deflation method based on a singular A into a method where A is nonsingular?

We start with some notations, definitions and preliminary results in Section 5.2. Section 5.3 is devoted to the introduction of the deflation variants that are compared in this chapter. The theoretical comparison of these variants is performed in Section 5.4. Some results of numerical experiments are presented in Section 5.5. We end this chapter with some concluding remarks in Section 5.6.

Throughout this chapter, Assumption 1.2 holds. Recall that this assumption is always fulfilled in bubbly flow applications, but it could also be applied to other fields.

5.2 Preliminaries

We start this section by presenting the definition of a nonsingular coefficient matrix, $\bar{A}$, based on A.

Definition 5.2. *Let A satisfy Assumption 1.2. Suppose that $\sigma > 0$ is given. Then, the coefficients of $\bar{A} = [\bar{a}_{ij}]$ are defined as*

$$\bar{a}_{i,j} = \begin{cases} (1+\sigma)a_{i,j}, & \text{if } i = j = n; \\ a_{i,j}, & \text{otherwise.} \end{cases} \tag{5.4}$$

The next two properties follow immediately, where we use Lemma A.9.

Property 5.2. *Let $\bar{A}$ be as given in Definition 5.2. Then, $\bar{A}$ is invertible and SPD.*

Property 5.3. *Let $\bar{A}$ be as given in Definition 5.2. Then, it satisfies $\bar{A}\mathbf{1}_{n,n} = \sigma a_{n,n}\mathbf{e}^{(n)}_{n,n}$. In particular, $\bar{A}\mathbf{1}_n = \sigma a_{n,n}\mathbf{e}^{(n)}_n$.*

It can be easily shown that forcing invertibility of A automatically leads to a worse condition number, see Lemma 5.1.

Lemma 5.1. *Let A and $\bar{A}$ be as in Assumption 1.2 and Definition 5.2, respectively. Then,*

(i) $\kappa(\bar{A}) \geq \kappa(A)$, *for all* $\sigma > 0$;

(ii) $\lim_{\sigma \to 0} \kappa(\bar{A}) = \infty$.

Proof. (i) Note that we can write

$$\bar{A} = A + \tau c c^T, \quad c = \mathbf{e}_n^{(n)}, \quad \tau = \sigma a_{n,n}.$$

Lemma A.6 can now be applied by taking $B := \bar{A}$ and $C := A$. From Eq. (A.1), we then obtain

$$\lambda_i(A) \leq \lambda_i(\bar{A}) \leq \lambda_{i+1}(A), \quad i = 1, 2, \ldots, n-1.$$

Therefore,

$$\lambda_1(A) < \lambda_1(\bar{A}) \leq \lambda_2(A), \tag{5.5}$$

using Property 5.2. Furthermore, from Eq. (A.2), we derive

$$\lambda_n(\bar{A}) \geq \lambda_n(A). \tag{5.6}$$

By combining Eqs. (5.5) and (5.6),

$$\kappa(\bar{A}) = \frac{\lambda_n(\bar{A})}{\lambda_1(\bar{A})} \geq \frac{\lambda_n(A)}{\lambda_2(A)} = \kappa(A)$$

follows.

(ii) Taking $B := A$ and $G := \bar{A} - B$ in Lemma A.10(ii) leads to

$$G = \sigma a_{n,n} \mathbf{e}_n^{(n)} \left(\mathbf{e}_n^{(n)}\right)^T,$$

yielding

$$\lambda_1(G) = \ldots = \lambda_{n-1}(G) = 0, \quad \lambda_n(G) = \sigma a_{n,n}.$$

As a result of Lemma A.10(ii), we obtain

$$\lambda_i(A) \leq \lambda_i(\bar{A}) \leq \lambda_i(A) + \sigma a_{n,n}, \quad i = 1, 2, \ldots, n,$$

so, in particular,

$$0 < \lambda_1(\bar{A}) \leq \sigma a_{n,n}, \quad \lambda_n(A) \leq \lambda_n(\bar{A}) \leq \lambda_n(A) + \sigma a_{n,n}.$$

This implies

$$\lim_{\sigma \to 0} \kappa(\bar{A}) = \lim_{\sigma \to 0} \frac{\lambda_n(\bar{A})}{\lambda_1(\bar{A})} \geq \lim_{\sigma \to 0} \frac{\lambda_n(A)}{\sigma a_{n,n}} = \infty.$$

□

Remark 5.1. *Lemma 5.1 seems to be generalizable for preconditioned coefficient matrices, although a proof is lacking.*

Let $B \in \mathbb{R}^{n \times n}$ be an arbitrary matrix. Then, $B^+ \in \mathbb{R}^{n \times n}$ denotes the pseudo-inverse (also called Moore-Penrose generalized inverse) of B, if it satisfies all following conditions:

$$\begin{cases} BB^+B &= B; \\ B^+BB^+ &= B^+; \\ (BB^+)^T &= BB^+; \\ (B^+B)^T &= B^+B. \end{cases} \tag{5.7}$$

Obviously, if B is nonsingular, then the pseudo-inverse and the inverse coincide, i.e., $B^+ = B^{-1}$. In general, the pseudo-inverse always exists and is unique: for any B, there is precisely one B^+ that satisfies (5.7). We refer to, e.g., [14, 73] for more details on the pseudo-inverse.

Remark 5.2. *The pseudo-inverse of a symmetric (and possibly singular) matrix, A, can be computed explicitly by first writing*

$$A = V\Lambda V^T, \quad V = [v_1 \; v_2 \; \cdots \; v_n], \quad \Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_n),$$

with diagonal matrix Λ consisting of the eigenvalues of A, and V being an orthonormal matrix that consists of the corresponding eigenvectors. Now, the pseudo-inverse of A can be determined via

$$A^+ := V\Lambda^+ V^T,$$

where Λ^+ is a diagonal matrix with the reciprocals of each nonzero entry on the diagonal of Λ, and leaving the zeros in place. For example, $\Lambda = diag(0, 1, 2, 3)$ gives $\Lambda^+ = diag\left(0, 1, \frac{1}{2}, \frac{1}{3}\right)$. This implies that a solution of the consistent linear system, $Ax = b$, is given by

$$x = A^+ b = V\Lambda^+ V^T b.$$

Using pseudo-inverses, the next definition and corresponding corollaries can be given.

Definition 5.3. *Let A and $\bar{A}$ be as in Assumption 1.2 and Definition 5.2, respectively. Let Z_i and $\widehat{Z}_i$ be as given in Definition 5.1. Then, the deflation matrices are defined as*

$$\begin{cases} P_i &:= I - AQ_i, & Q_i &:= Z_i E_i^+ Z_i^T, & E_i &:= Z_i^T A Z_i; \\ \bar{P}_i &:= I - \bar{A}\bar{Q}_i, & \bar{Q}_i &:= Z_i \bar{E}_i^{-1} Z_i^T, & \bar{E}_i &:= Z_i^T \bar{A} Z_i; \\ \widehat{P}_i &:= I - \bar{A}\widehat{Q}_i, & \widehat{Q}_i &:= \widehat{Z}_i \widehat{E}_i^{-1} \widehat{Z}_i^T, & \widehat{E}_i &:= \widehat{Z}_i^T \bar{A} \widehat{Z}_i. \end{cases}$$

Both $\bar{P}_i$ and $\widehat{P}_i$ are based on the invertible coefficient matrix, $\bar{A}$, which only differ from the deflation-subspace matrices. Hence, $\bar{E}_i$ and $\widehat{E}_i$ are both nonsingular, so that $\bar{E}_i^{-1}$ and $\widehat{E}_i^{-1}$ exist. On the other hand, P_i is based on the singular coefficient matrix, A. Since the resulting E_k is singular due to Corollary 5.2, its pseudo-inverse has to be used in P_i.

Corollary 5.1. *Let $\widehat{P}_k$ and $\bar{P}_k$ be as given in Definition 5.3. Then, $\widehat{P}_k = \bar{P}_k$ holds.*

Proof. This follows immediately from Theorem 3.2, since $\mathcal{R}(Z_k) = \mathcal{R}(\widehat{Z}_k)$. □

Corollary 5.2. *Let E_i be as given in Definition 5.3. Then,*

- *E_{k-1} is nonsingular;*
- *E_k is singular.*

Proof. (i) Since $\mathcal{N}(A) \not\subseteq \mathcal{R}(Z_{k-1})$ is satisfied, E_{k-1} is nonsingular, due to Lemma 3.1.

(ii) Using Property 5.1(i), we have

$$E_k \mathbf{1}_k = Z_k^T A Z_k \mathbf{1}_k = Z_k^T A \mathbf{1}_n = Z_k^T \mathbf{0}_n = \mathbf{0}_k, \tag{5.8}$$

so that E_k is singular. □

Remark 5.3. *In Chapter 3, some properties of the deflation method have been derived using deflation matrix P_k, where E_k is assumed to be invertible. These properties hold in particular for the deflation methods based on $\bar{P}_k$ and $\widehat{P}_k$.*

5.3 Deflation Variants

In this section, we present three deflation variants based on Definitions 5.2 and 5.3.

Variant 5.1. *Solve $\bar{x}$ from*

$$M^{-1} P_{k-1} A \bar{x} = M^{-1} P_{k-1} b. \tag{5.9}$$

Variant 5.2. *Solve $\bar{x}$ from*

$$\bar{M}^{-1} \bar{P}_k \bar{A} \bar{x} = \bar{M}^{-1} \bar{P}_k b. \tag{5.10}$$

Variant 5.3. *Solve $\bar{x}$ from*

$$M^{-1} P_k A \bar{x} = M^{-1} P_k b. \tag{5.11}$$

Variant 5.1 is the common deflation method that has been used in prior chapters, with the only difference that $k-1$ instead of k deflation vectors are adopted. Variant 5.2 is based on the nonsingular coefficient matrix, $\bar{A}$, so that this variant is always well-defined. Finally, Variant 5.3 is basically identical to the original DPCG for invertible coefficient matrices (see, e.g., [56, 103]), since the original coefficient matrix and all k deflation vectors are used in this variant. Hence, it is the most natural generalization of DPCG for singular systems, although additional efforts are required to generalize all known results for invertible coefficient matrices or singular coefficient matrices with a nonsingular Galerkin matrix, E. However, this can be circumvented, because we show in Section 5.4.4 that Variant 5.3 is (almost) identical to the other two variants.

Remark 5.4.

- *For Variants 5.1 and 5.2, it is common to solve the Galerkin systems associated with E_{k-1} or $\bar{E}_k$ in a direct way, if k is relatively small. However, these Galerkin systems could also be solved iteratively, which can be more efficient if k is relatively large. On the other hand, although E_k^{-1} does not exist, it may be possible* *to apply a direct method for solving the corresponding Galerkin systems. Extra care is then needed by applying, e.g., Gaussian elimination or the band-Cholesky decomposition, to handle the singularity of E_k and to generate a solution up to $\mathcal{N}(E_k)$. However, in this thesis, we restrict ourselves to solve the Galerkin systems in Variant 5.3 iteratively, so that the pseudo-inverse is not explicitly required. This does not cause any problems, as long as these Galerkin systems are consistent, see Chapter 8.*

- *There are more deflation variants known in the literature, which deal with singular coefficient matrices. For example, a variant can be based on Variant 5.3, where one entry of Z_k is perturbed such that E_k becomes nonsingular. Special care should be taken to choose this perturbation appropriately, see [168]. Another variant, that is frequently applied in the multigrid field, is also related to Variant 5.3. If we restrict ourselves to two-grid methods, then E_k is perturbed such that it becomes nonsingular. In other words, the corresponding deflation matrix is*

  $$P_i := I - AQ_i, \quad Q_i := Z_i \hat{E}_i^{-1} Z_i^T, \quad E_i := Z_i^T A Z_i, \quad \hat{E}_i \approx E_i. \tag{5.12}$$

  $\hat{E}_i$ could be obtained in the same way as $\hat{A}$ (cf. Definition 5.2). The associated Galerkin systems can now be solved with a direct method, see also Chapter 9. However, if an iterative method is used, then a slower convergence would be expected, compared to the convergence for solving the Galerkin systems in Variant 5.3. This follows from the fact that $\hat{E}_i$ is more ill-conditioned than the original E_i (cf. Lemma 5.1(i)).

For the sake of convenience, the corresponding matrices of the three deflation variants are summarized in Table 5.1. These variants are compared in the next section in order to determine the most effective variant.

	Matrices				
Variant	Coefficient	Deflation-subspace	Galerkin	Correction	Deflation
5.1	A	Z_{k-1}	E_{k-1}	Q_{k-1}	P_{k-1}
5.2	$\bar{A}$	Z_k	$\bar{E}_k$	$\bar{Q}_k$	$\bar{P}_k$
5.3	A	Z_k	E_k	Q_k	P_k

Table 5.1: Corresponding matrices of the proposed deflation variants.

5.4 Theoretical Comparison of Deflation Variants

In this section, we first show that the condition number of $\bar{A}$ is reduced to the condition number of A by a simple deflation technique. Thereafter, we prove that even the matrices $\bar{M}^{-1}\bar{P}_k\bar{A}$ and $M^{-1}P_{k-1}A$ corresponding to Variants 5.1 and 5.2, respectively, are (almost) equal. Finally, it is also shown that $M^{-1}P_{k-1}A$ from Variant 5.1 and

$M^{-1}P_kA$ from Variant 5.3 are equal. As a consequence, Variants 5.1, 5.2 and 5.3 are based on approximately identical preconditioned-deflated coefficient matrix, so that they would theoretically lead to the same convergence results.

5.4.1 On the Connection of the Singular and Invertible Matrix

Recall that $\bar{P}_1$ is the deflation matrix with one constant deflation vector based on $\bar{A}$. In this subsection, we show that the deflated matrix, $\bar{P}_1\bar{A}$, is identical to the original singular matrix, A. We start with Lemma 5.2, that shows that $\bar{P}_1$ is the identity matrix except for the last row.

Lemma 5.2. *Let $\bar{P}_1$ be as given in Definition 5.3. Then, $\bar{P}_1 = I - \mathbf{e}_{n,n}^{(n)}$.*

Proof. For $k = 1$, we have

$$\bar{P}_1 = I - \bar{A}z_0\bar{E}^{-1}z_0^T, \quad \bar{E}^{-1} = \left(z_0^T\bar{A}z_0\right)^{-1} = \frac{1}{\sigma a_{n,n}},$$

using Property 5.3. Hence,

$$\bar{P}_1 = I - \frac{\bar{A}\mathbf{1}_{n,n}}{\sigma a_{n,n}} = I - \mathbf{e}_{n,n}^{(n)}.$$

□

As a consequence, $\bar{P}_1$ has the properties that the last column is $\mathbf{0}_n$, and the matrix consists of only the values 0, 1 and -1. Next, by applying Lemma 5.2, we obtain the following theorem.

Theorem 5.1. *Let A and $\bar{A}$ be as in Assumption 1.2 and Definition 5.2, respectively. Let $\bar{P}_1$ be as given in Definition 5.3. Then, $\bar{P}_1\bar{A} = A$ holds.*

Proof. Due to Lemma 5.2, $\bar{P}_1\bar{A} = A$ holds for all rows except the last one. The analysis of the last row of $\bar{P}_1\bar{A}$, which is $(\mathbf{e}_n^{(n)} - \mathbf{1}_n)^T\bar{A}$, is as follows. Since $\mathbf{1}_n^TA = \mathbf{0}_n^T$ and $(\mathbf{e}_n^{(n)} - \mathbf{1}_n)^T\bar{A} = (\mathbf{e}_n^{(n)} - \mathbf{1}_n)^TA$ hold, this yields

$$(\mathbf{e}_n^{(n)} - \mathbf{1}_n)^T\bar{A} = \left(\mathbf{e}_n^{(n)}\right)^T A.$$

Hence, the last row of $\bar{P}_1\bar{A}$ and A is also equal, which proves the theorem. □

Theorem 5.1 implies that, after premultiplying with the deflation matrix with $k = 1$, the invertible coefficient matrix, $\bar{A}$, becomes equal to the original singular coefficient matrix, A. Apparently, only one eigenvalue of $\bar{A}$ depends on the value of σ, which corresponds to the constant eigenvector. This eigenvector is eliminated effectively by $\bar{P}_1$, so that $\bar{P}_1\bar{A}$ and A are equal. Moreover, according to the proof of Theorem 5.1, the results for this deflation technique are independent of the entries of the last row of $\bar{A}$.

5.4.2 Comparison of the Deflated Singular and Invertible Matrix

Theorem 5.2 is the main result of this subsection. In order to prove this theorem, a set of lemmas is required, which are stated below. The most important lemmas are Lemma 5.3 and Lemma 5.6, which show that deflation matrix $\bar{P}_k$ is invariant by right-multiplication by $\bar{P}_1$, and that deflated matrices $\bar{P}_k A$ and $P_{k-1}A$ are identical.

Lemma 5.3. *Let $\bar{P}_i$ be as given in Definition 5.3. $\bar{P}_k \bar{P}_1 = \bar{P}_k$ holds.*

Proof. Note first that the last column of $\bar{E}^{-1}$ is equal to $\frac{1}{\sigma a_{n,n}}\mathbf{1}_k$ by using

$$Z_k^T \bar{A} Z_k \mathbf{1}_k = \sigma a_{n,n} Z_k^T \mathbf{e}_n^{(n)} = \sigma a_{n,n} \mathbf{e}_k^{(k)}$$

and Lemma A.11. Then, for all σ, the last column of $\bar{A}\bar{Q}_k$ is exactly $\mathbf{e}_n^{(n)}$, since we have

$$\begin{aligned} \left(\bar{A}\bar{Q}_k\right)_{1:n,n} &= \left(\bar{A} Z_k \bar{E}^{-1} Z_k^T\right)_{1:n,n} = \textstyle\sum_{p=1}^{k} \left(\bar{A} Z_k\right)_{1:n,p} \left(\bar{E}^{-1} Z_k^T\right)_{p,n} \\ &= \frac{1}{\sigma a_{n,n}} \textstyle\sum_{p=1}^{k} (\bar{A} Z_k)_{1:n,p} = \frac{1}{\sigma a_{n,n}} \bar{A}\mathbf{1}_n = \frac{1}{\sigma a_{n,n}} \sigma a_{n,n} \mathbf{e}_n^{(n)} = \mathbf{e}_n^{(n)}, \end{aligned}$$

for all $i = 1, 2, \ldots, n$, where Definition 5.1 and Property 5.1 are used. Therefore, the last column of $\bar{P}_k = I - \bar{A}\bar{Q}_k$ is $\mathbf{0}_n$. Using the latter fact combined with Property 5.3, we obtain $\bar{P}_k \bar{A} \mathbf{1}_n = \mathbf{0}_n$. Hence, this implies

$$\bar{P}_k \bar{P}_1 = \bar{P}_k \left(I - \alpha \bar{A} \mathbf{1}_n\right) = \bar{P}_k - \alpha \bar{P}_k \bar{A} \mathbf{1}_n = \bar{P}_k.$$

□

Lemma 5.4. *Let $\widehat{Z}_k$ and $\bar{A}$ be as given in Definitions 5.1 and 5.2, respectively. Then, there exists a matrix $\bar{Y} := [z_{k+1}\ z_{k+2}\ \cdots\ z_n] \in \mathbb{R}^{(n-k)\times n}$ such that*

- *$X := \left[\bar{Y},\ \widehat{Z}_k\right]$ is invertible;*
- *$\widehat{Z}_k^T \bar{A} \bar{Y} = \mathbf{0}_{k,n-k}$ holds.*

Proof. It is always possible to find a full-rank matrix, $\bar{Y}$, such that

$$\mathcal{R}(X) = \mathcal{R}(\bar{Y}) \oplus \mathcal{R}(\widehat{Z}_k),$$

where $\mathcal{R}(\bar{Y})$ is the orthogonal complement of $\mathcal{R}(\widehat{Z}_k)$, see Lemma A.15. Then, by definition (see Definition A.2), $X := \left[\bar{Y},\ \widehat{Z}_k\right]$ is an invertible matrix. Furthermore, by Definition A.2, we have (cf. Eq. (A.11))

$$\mathcal{R}(\bar{Y}) = \left\{ y \in \mathbb{R}^n \mid \langle w, y \rangle_{\bar{A}} = 0 \quad \forall w \in \mathcal{R}(\widehat{Z}_k) \right\}.$$

In particular, for all $w \in \mathcal{R}(\widehat{Z}_k)$ and $y \in \mathcal{R}(\bar{Y})$, we have $\langle w, y \rangle_{\bar{A}} = w^T \bar{A} y = 0$, which yields $\widehat{Z}_k^T \bar{A} \bar{Y} = \mathbf{0}_{k,n-k}$. □

Lemma 5.5. *Let A and $\bar{A}$ be as in Assumption 1.2 and Definition 5.2, respectively. Suppose that z_0 is as given in Definitions 5.1. Let P_{k-1} and $\widehat{P}_k$ be as given in Definition 5.3. Then, the following equalities hold:*

(i) $\left(P_{k-1} - \widehat{P}_k - \mathbf{e}_n^{(n)} z_0^T\right) \bar{A} z_0 = \mathbf{0}_n$;

(ii) $\left(P_{k-1} - \widehat{P}_k\right) \bar{A} Z_{k-1} = \mathbf{0}_{n,k-1}$.

Proof. *(i)* Note first that

$$Z_{k-1}^T \bar{A} z_0 = \mathbf{0}_n, \quad \widehat{Q}_k \bar{A} z_0 = \widehat{Z}_k \widehat{E}_k^{-1} \left(\widehat{Z}_k^T \bar{A} z_0\right) = \widehat{Z}_k \mathbf{e}_k^{(k)} = z_0, \tag{5.13}$$

using Property 5.1. Combining Eq. (5.13) with Property 5.3 yields

$$\begin{aligned} \left(\bar{A}\widehat{Q}_k - AQ_{k-1}\right) \bar{A} z_0 &= \bar{A}\widehat{Q}_k \bar{A} z_0 - AZ_{k-1} E_{k-1}^{-1} \left(Z_{k-1}^T \bar{A} z_0\right) \\ &= \bar{A} z_0 = \sigma a_{n,n} \mathbf{e}_n^{(n)}. \end{aligned} \tag{5.14}$$

Moreover, we derive

$$\mathbf{e}_n^{(n)} \mathbf{1}_n^T \bar{A} z_0 = \sigma a_{n,n} \mathbf{e}_{n,n}^{(n)} \mathbf{e}_n^{(n)} = \sigma a_{n,n} \mathbf{e}_n^{(n)}, \tag{5.15}$$

using the facts that $\mathbf{e}_n^{(n)} \mathbf{1}_n^T = \mathbf{e}_{n,n}^{(n)}$ and $\mathbf{e}_{n,n}^{(n)} \mathbf{e}_n^{(n)} = \mathbf{e}_n^{(n)}$. Equalizing Eqs. (5.14) and (5.15) results in $\left(P_{k-1} - \widehat{P}_k - \mathbf{e}_n^{(n)} z_0^T\right) \bar{A} z_0 = \mathbf{0}_n$.

(ii) Note first that the following identities hold for $i \leq k-1$:

- $Q_{k-1} A z_i = Z_{k-1} E_{k-1}^{-1} \left(Z_{k-1}^T A z_i\right) = Z_{k-1} \mathbf{e}_{k-1}^{(i)} = z_i$;
- $\widehat{Q}_k \bar{A} z_i = \widehat{Z}_k \bar{E}_k^{-1} \left(\widehat{Z}_k^T \bar{A} z_i\right) = \widehat{Z}_k \mathbf{e}_k^{(i)} = z_i$;
- $\bar{A} z_i = A z_i$,

applying Property 5.1. As a consequence,

$$\left(P_{k-1} - \widehat{P}_k\right) \bar{A} z_i = \bar{A} \widehat{Q}_k \bar{A} z_i - A Q_{k-1} A z_i = \bar{A} z_i - A z_i = \mathbf{0}_n,$$

for $i = 1, 2, \ldots, k-1$. This yields $\left(P_{k-1} - \widehat{P}_k\right) \bar{A} Z_{k-1} = \mathbf{0}_{n,k-1}$. □

Lemma 5.6. *Let A satisfy Assumption 1.2. Let $\bar{P}_k$ and P_{k-1} be as given in Definition 5.3. Then, $\bar{P}_k A = P_{k-1} A$ holds.*

Proof. It is sufficient to prove that $\left(P_{k-1} - \bar{P}_k\right) A = \mathbf{0}_{n,n}$. Since $A\mathbf{1}_n = \mathbf{0}_n$ (Assumption 1.2) holds, this implies that we have to show that each row of $P_{k-1} - \bar{P}_k$ contains the same entries. In other words, after defining

$$B := [\varrho_1 \ \varrho_2 \ \ldots \ \varrho_n]^T \mathbf{1}_n^T,$$

it suffices to prove that there exist some parameters, $\varrho_i \in \mathbb{R},\ i = 1, 2, \ldots, n$, and an invertible matrix, $C \in \mathbb{R}^{n \times n}$, such that

$$\left(P_{k-1} - \widehat{P}_k - B\right) C = \mathbf{0}_{n,n} \tag{5.16}$$

is satisfied. Then, this would yield $P_{k-1} - \bar{P}_k = B$, since $\widehat{P}_k = \bar{P}_k$ holds (Corollary 5.1).

The proof is as follows. Take

$$C := \bar{A}\left[\widehat{Z}_k,\ \bar{Y}\right] = \bar{A}\left[Z_{k-1},\ z_0,\ \bar{Y}\right],$$

where $\bar{Y} = [z_{k+1}\ z_{k+2}\ \cdots\ z_n]$ has the properties that the set,

$$\{z_i :\ i = 0, 1, \ldots, n,\ i \neq k\},$$

is linearly independent and

$$\widehat{Z}_k^T \bar{A}\bar{Y} = \mathbf{0}_{n-k}^T \tag{5.17}$$

is satisfied. Using Lemma 5.4, such a matrix, $\bar{Y}$, can always be constructed. Note that, from Eq. (5.17), we particularly obtain

$$z_0^T \bar{A}\bar{Y} = \mathbf{0}_{n-k}^T, \quad Z_{k-1}^T \bar{A}\bar{Y} = \mathbf{0}_{k-1}. \tag{5.18}$$

Next, the following equalities hold:

- $\left(P_{k-1} - \widehat{P}_k\right) \bar{A}\bar{Y} = \mathbf{0}_{n,n-k}$, by combining Eqs. (5.17) and (5.18);
- $\left(P_{k-1} - \widehat{P}_k\right) \bar{A}Z_{k-1} = \mathbf{0}_{n,k-1}$ (Lemma 5.5(ii));
- $B\left[\bar{A}Z_{k-1},\ \bar{A}\bar{Y}\right] = \mathbf{0}_{n,n}$, since Eq. (5.18) holds and $z_0^T A Z_{k-1} = \mathbf{0}_{k-1}^T$ follows from Properties 5.1 and 5.3.

Combining these latter results gives us

$$\left(P_{k-1} - \widehat{P}_k - B\right)\left[\bar{A}Z_{k-1},\ \bar{A}\bar{Y}\right] = \mathbf{0}_{n,n-1}, \tag{5.19}$$

for all ϱ_i. Moreover,

$$\left(P_{k-1} - \widehat{P}_k - B\right) \bar{A}z_0 = \mathbf{0}_n \tag{5.20}$$

holds due to Lemma 5.5(i), by taking $\varrho_1 \ldots = \varrho_{n-1} = 0$ and $\varrho_n = 1$. Hence, combining Eqs. (5.19) and (5.20) yields

$$\left(P_{k-1} - \widehat{P}_k - B\right) C = \left(P_{k-1} - \widehat{P}_k - B\right)\left[\bar{A}Z_{k-1},\ \bar{A}\widehat{z}_0,\ \bar{A}\bar{Y}\right] = \mathbf{0}_{n,n},$$

with $\varrho_1 \ldots = \varrho_{n-1} = 0$ and $\varrho_n = 1$, which completes the proof of the lemma. □

Finally, Theorem 5.2 shows that the deflated singular matrices based on A and $\bar{A}$ are equal, which is a rather unexpected result. The consequence of the theorem is that Variants 5.1 and 5.2 have the same expected convergence rate.

Theorem 5.2. *Let A and $\bar{A}$ be as in Assumption 1.2 and Definition 5.2, respectively. Let P_{k-1} and $\bar{P}_k$ be as given in Definition 5.3. Then, $\bar{P}_k\bar{A} = P_{k-1}A$ holds for all $\sigma > 0$ and $k > 1$.*

Proof. The following equalities hold:

- $\bar{P}_1\bar{A} = A$ (Theorem 5.1);
- $\bar{P}_k\bar{P}_1 = \bar{P}_k$ (Lemma 5.3);
- $\bar{P}_kA = P_{k-1}A$ (Lemma 5.6),

which are valid for all $\sigma > 0$ and $k \geq 1$. Hence,

$$\bar{P}_k\bar{A} = \bar{P}_k\bar{P}_1\bar{A} = \bar{P}_kA = P_{k-1}A.$$

□

5.4.3 Comparison of the Preconditioned Deflated Singular and Invertible Matrix

In this subsection, we restrict ourselves to the standard incomplete Cholesky (IC(0)) preconditioners, see the next definition.

Assumption 5.1. *Suppose that A satisfies Assumption 1.2 and let $\bar{A}$ be given as Definition 5.2. Then, M^{-1} and $\bar{M}^{-1}$ are the IC(0) preconditioners based on A and $\bar{A}$, respectively.*

From Theorem 5.2, the equality $\bar{P}_k\bar{A} = P_{k-1}A$ holds. This implies that the preconditioned variant of this equality also holds, see the next corollary.

Corollary 5.3. *Let A and $\bar{A}$ be as in Assumption 1.2 and Definition 5.2, respectively. Let $\bar{P}_k$ and P_{k-1} be as in Definition 5.3. Moreover, let M^{-1} be as in Assumption 5.1. Then, $M^{-1}\bar{P}_k\bar{A} = M^{-1}P_{k-1}A$.*

However, if A is not known explicitly, then M^{-1} could be difficult to determine, whereas $\bar{M}^{-1}$ could be readily obtained from $\bar{A}$. This might be inconvenient, because of the fact that $\bar{M}^{-1}\bar{P}_k\bar{A} \neq M^{-1}P_{k-1}A$. However, we show in this subsection that

$$\lim_{\sigma\to 0} \kappa(\bar{M}^{-1}\bar{P}_k\bar{A}) = \kappa(M^{-1}P_{k-1}A) \tag{5.21}$$

holds. First, we deal with the comparison of the condition numbers of $M^{-1}A$ and $\bar{M}^{-1}A$, and, thereafter, we generalize these results to $M^{-1}P_{k-1}A$ and $\bar{M}^{-1}\bar{P}_k\bar{A}$.

An algorithm for computing the IC(0) preconditioner can be found in, e.g., [63, Sect. 10.3.2]. Recall that the IC(0) preconditioner is formed by $M = LL^T$, where L is a lower-triangular matrix. Analogously, $\bar{M} = \bar{L}\bar{L}^T$ can be computed from $\bar{A}$. According to the algorithm described in [63], the IC(0) preconditioners of A and $\bar{A}$ are the same except the last entry, since only the last entry of L and $\bar{L}$ differs, i.e.,

$$\bar{M} - M = \gamma \mathbf{e}_n^{(n)} \left(\mathbf{e}_n^{(n)}\right)^T, \quad \gamma \in \mathbb{R}. \tag{5.22}$$

If we denote $M = [m_{i,j}]$ and $\bar{M} = [\bar{m}_{i,j}]$, then we have $m_{n,n} = a_{n,n}$ and $\bar{m}_{n,n} = \bar{a}_{n,n}$ by definition. Consequently,

$$\gamma = \bar{m}_{n,n} - m_{n,n} = \bar{a}_{n,n} - a_{n,n} = \sigma a_{n,n}.$$

This implies

$$\lim_{\sigma \to 0} \gamma = \lim_{\sigma \to 0} \sigma a_{n,n} = 0. \tag{5.23}$$

Now, we can prove that the condition numbers of $M^{-1}A$ and $\bar{M}^{-1}A$ are the same for $\sigma \to 0$, see Theorem 5.3.

Theorem 5.3. *Suppose that A satisfies Assumption 1.2. Let M^{-1} and $\bar{M}^{-1}$ be as given in Assumption 5.1. Then, the following identity holds:*

$$\lim_{\sigma \to 0} \kappa(\bar{M}^{-1}A) = \kappa(M^{-1}A).$$

Proof. The eigenproblems of $M^{-1}A$ and $\bar{M}^{-1}A$ are given by

$$M^{-1}Av = \lambda v, \quad \bar{M}^{-1}Aw = \mu w, \quad v, w \in \mathbb{R}^n, \quad \lambda, \mu \in \mathbb{R}. \tag{5.24}$$

These eigenproblems can be rewritten as generalized eigenproblems,

$$(A - \lambda M)v = 0, \quad (A - \mu \bar{M})w = 0.$$

Due to Eq. (5.22), we have $M + R_M = \bar{M}$, with the symmetric perturbation matrix, R_M, given by

$$R_M = \xi \mathbf{e}_n^{(n)} \left(\mathbf{e}_n^{(n)}\right)^T, \quad \xi \in \mathbb{R}.$$

This yields

$$||R_M||_2 = \max\{\ |\lambda_1(R_M)|\ ,\ |\lambda_n(R_M)|\ \} = \xi.$$

We note that R_M satisfies $||R_M||_2^2 < c(A, M)$, where $c(A, M)$ denotes the Crawford number (see Eq. (A.3)). This is due to the fact that there exists a parameter, $\sigma_0 > 0$, such that $\xi^2 < c(A, M)$ is satisfied for all $\sigma < \sigma_0$, since $c(A, M)$ does obviously not depend on σ.

Lemma A.8 can now be applied, since the conditions of this lemma are satisfied. Note that $\lim_{\sigma \to 0}\ \xi = 0$ follows from Eq. (5.23). This implies

$$\lim_{\sigma \to 0} \frac{\xi}{c(A, M)} = \frac{1}{c(A, M)} \lim_{\sigma \to 0} \xi = 0,$$

so, in particular,

$$\lim_{\sigma \to 0} \arctan\left(\frac{\xi}{c(A,M)}\right) = 0. \tag{5.25}$$

Now, the eigenvalues of (5.24) are related by Eq. (A.4) of Lemma A.8, i.e.,

$$|\arctan(\lambda_i) - \arctan(\mu_i)| \le \arctan\left(\frac{||R_M||_2}{c(A,M)}\right). \tag{5.26}$$

By combining Eqs. (5.25) and (5.26), we obtain

$$\lim_{\sigma \to 0} \arctan(\lambda_i) = \arctan(\mu_i),$$

resulting in $\lim_{\sigma \to 0} \lambda_i = \mu_i$, since the arctan-operator is bijective and continuous. Hence, the theorem follows immediately. □

Next, we compare the condition numbers of $M^{-1}P_kA$ and $\bar{M}^{-1}\bar{P}_k\bar{A}$. Recall that both A and $P_{k-1}A$ are SPSD matrices, so that we can substitute $P_{k-1}A$ into A in Theorem 5.3. Since $\bar{P}_k\bar{A} = P_{k-1}A$ follows from Theorem 5.2, this gives us the next theorem.

Theorem 5.4. *Suppose that A satisfies Assumption 1.2. Let P_{k-1} and $\bar{P}_k$ be as given in Definition 5.3. Moreover, let M^{-1} and $\bar{M}^{-1}$ satisfy Assumption 5.1. Then,*

$$\lim_{\sigma \to 0} \kappa(\bar{M}^{-1}\bar{P}_k\bar{A}) = \kappa(M^{-1}P_{k-1}A).$$

Theorem 5.4 states that, although $\bar{M}^{-1}\bar{P}_k\bar{A}$ and $M^{-1}P_{k-1}A$ differ, their condition number are almost identical for a sufficiently small perturbation, σ. As a result, Variants 5.1 and 5.2 are expected to have a similar convergence rate.

5.4.4 Comparison of the Preconditioned Deflated Singular Matrices

Here, we prove that the preconditioned deflated matrices, $M^{-1}P_{k-1}A$ and $M^{-1}P_kA$, corresponding to Variants 5.1 and 5.3 are equal. This main result is presented in Theorem 5.5.

Theorem 5.5. *Suppose that A satisfies Assumption 1.2. Let P_i and M^{-1} be as in Definition 5.3 and Assumption 5.1, respectively. Then, the following identity holds:*

$$M^{-1}P_kA = M^{-1}P_{k-1}A. \tag{5.27}$$

Proof. The proof can be found in Appendix C. □

According to Theorem 5.5, the deflated-preconditioned matrices based on $k-1$ and k deflation vectors are the same, so that Variants 5.1 and 5.3 are mathematically equivalent.

Remark 5.5.

- *From Theorem 5.5, it can be observed that it is possible to base computations with deflation matrices on the real inverse rather than the pseudo-inverse of E. This theorem can even be generalized to general singular coefficient matrices and deflation vectors, see Appendix C for details. This research is still ongoing during writing this thesis, see [85].*

- *By combining Theorems 5.3 and 5.5, we conclude that all proposed deflation variants are equivalent for sufficiently small σ, so the effectiveness of these variants is approximately the same. It depends on the efficiency of implementation of the variants in order to decide which variant is the best one in practice, see Chapter 8.*

5.5 Application to Bubbly Flows

In this section, we illustrate the theoretical results, as presented in the previous section, with bubbly flow experiments. The 3-D variants of the bubbly flows with $\epsilon = 10^3$, $m = 2^3$ and $s = 0.05$, as given in Figure 1.2 of Section 1.3, are considered. Both the resulting linear systems, $Ax = b$ and $\bar{A}x = b$, are ill-conditioned, due to the presence of bubbles. These linear systems are solved using ICCG and DICCG$-k$ (i.e., DPCG with the IC(0) preconditioner and k deflation vectors). The termination criterion is based on (3.27) with $\delta = 10^{-8}$, and the deflation-subspace matrix, Z, consists of subdomain deflation vectors as defined in Section 4.2.3. We vary the perturbation parameter, σ, and the number of deflation vectors, k, in our experiments.

Note that numerical experiments with Variant 5.1 have already been performed in Chapters 3 and 4. In Section 5.5.1, we present the results for Variant 5.2. Thereafter, Section 5.5.2 is devoted to the comparison of Variants 5.1 and 5.2. Variant 5.3 is excluded in these experiments, since it requires special care for computations with E_k^+. This topic is further investigated in Chapter 8.

5.5.1 Results of ICCG and DICCG with Variant 5.2

The results of ICCG and DICCG with Variant 5.2 can be found in Table 5.2. Note that both methods are based on $\bar{A}x = b$ with an invertible coefficient matrix, $\bar{A}$. In the case of ICCG, the results of solving the original linear system, $Ax = b$, with a singular matrix, A, are added for comparison.

From Table 5.2, it can be observed that the number of iterations for DICCG is independent of σ (as long as it is sufficiently small), as expected from Theorem 5.3. Confirming Theorem 5.1, we see that the required number of iterations for ICCG is equal to the number for DICCG-1, when the problem with arbitrary $\sigma > 0$ is solved. Moreover, we notice that increasing the number of deflation vectors, k, leads to a nonincreasing number of iterations for DICCG (cf. Theorem 3.3).

(a) ICCG.

n	$\sigma = 0$	$\sigma = 10^{-1}$	$\sigma = 10^{-3}$
32^3	100	138	147
64^3	118	195	195

(b) DICCG$-k$ (Variant 5.2).

	$n = 32^3$		$n = 64^3$	
k	$\sigma = 10^{-1}$	$\sigma = 10^{-3}$	$\sigma = 10^{-1}$	$\sigma = 10^{-3}$
1	100	100	118	118
2^3	66	66	126	126
4^3	66	66	131	131
8^3	28	28	106	106

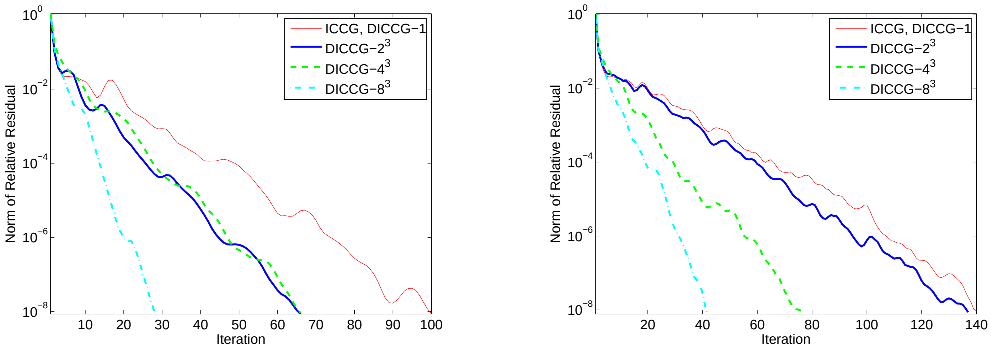
Table 5.2: Number of iterations for ICCG and DICCG (Variant 5.2) to solve the linear system $\bar{A}x = b$ with invertible $\bar{A}$, for the test case with $m = 2^3$, $\epsilon = 10^3$, and $s = 0.05$.

In Figure 5.1(a), the residuals of ICCG and DICCG can be found for the test case of $n = 32^3$ and $\sigma = 10^{-3}$. From this figure, it can be observed that ICCG shows an erratic convergence behavior, while DICCG converges almost monotonically. Apparently, the space spanned by eigenvectors corresponding to the small eigenvalues are well-approximated by the space spanned by the subdomain deflation vectors (cf. the results in Chapter 4). Moreover, we observe that the residuals of DICCG-2^3 and DICCG-4^3 almost coincide. This might be caused by the fact that some unfavorable eigenvectors of $M^{-1}A$ are not treated effectively by both 2^3 and 4^3 subdomain deflation vectors, and it is related to the geometry of the density field. When we take $m = 3^3$ bubbles, the results with $k = 4^3$ are much better than for $k = 2^3$, see Table 5.3. In addition, in Figure 5.1(b), the residuals of ICCG and DICCG are presented for the test case with $n = 32^3$ and $\sigma = 10^{-3}$. Now, the residuals of DICCG-4^3 decrease almost monotonically, whereas the residuals of both ICCG and DICCG-2^3 are still erratic. In this case, eigenvectors associated with small eigenvalues are worse approximated by the deflation vectors, compared to the case with $m = 2^3$ bubbles (cf. Figure 5.1(a)). This is caused by the position of the bubbles with respect to the subdomains, and the increased number of bubbles that is more complicated to treat with a relatively small number of deflation vectors.

5.5.2 Results of the Comparison between Variants 5.1 and 5.2

This subsection deals with a numerical comparison of Variants 5.1 and 5.2. The same setting as in the previous subsection is used. The results are presented in Table 5.4. Recall that Variant 5.1 adopts the deflation method with $k - 1$ instead of k deflation vectors, so that DICCG-1 is not defined in this case.

From Table 5.4, we observe immediately that the results for Variant 5.2 (Table 5.4(a)) are the same as those for Variant 5.1 (Table 5.4(b)). Indeed, the two different variants with a singular and invertible coefficient matrix seem to be mathematically equivalent, which confirms Theorem 5.4.



(a) $m = 2^3$ bubbles.

(b) $m = 3^3$ bubbles.

Figure 5.1: Residuals of ICCG, DICCG−2^3 and DICCG−3^3 with Variant 5.2, for the test case with $n = 32^3$ and $\sigma = 10^{-3}$.

(a) ICCG.

n	$\sigma = 0$	$\sigma = 10^{-1}$	$\sigma = 10^{-3}$
32^3	140	206	233
64^3	246	246	362

(b) DICCG−k (Variant 5.2).

	$n = 32^3$		$n = 64^3$	
k	$\sigma = 10^{-1}$	$\sigma = 10^{-3}$	$\sigma = 10^{-1}$	$\sigma = 10^{-3}$
1	140	140	246	246
2^3	137	137	197	197
4^3	76	76	131	131
8^3	42	42	59	59

Table 5.3: Number of iterations for ICCG and DICCG (Variant 5.2) to solve $\bar{A}x = b$, for the test case with $m = 3^3$, $\epsilon = 10^3$, and $s = 0.05$.

(a) Variant 5.1 (based on A).

k	$n = 32^3$	$n = 64^3$
2^3	66	126
4^3	66	131

(b) Variant 5.2 (based on $\bar{A}$).

	$n = 32^3$		$n = 64^3$	
k	$\sigma = 10^{-1}$	$\sigma = 10^{-3}$	$\sigma = 10^{-1}$	$\sigma = 10^{-3}$
2^3	66	66	126	126
4^3	66	66	131	131

Table 5.4: Number of iterations for DICCG (both Variant 5.1 and Variant 5.2) to solve $Ax = b$ (with a singular A) and $\bar{A}x = b$ (with an invertible $\bar{A}$) for $m = 2^3$.

5.6 Concluding Remarks

In this chapter, we present three different deflation variants, which can deal with the singularity of the coefficient matrix, A. In Variant 5.2, an invertible coefficient matrix, $\bar{A}$, is used instead of the singular matrix A, while the solution of the linear system remains the same. Invertibility of the matrix gives several advantages for the iterative solver. The drawback, however, is that the condition number of $\bar{A}$ becomes worse compared to that of A. We show that this difficulty can be completely remedied by applying the deflation technique with just one deflation vector. Moreover, Variants 5.1 and 5.3 are based on the original singular matrix, A. In Variant 5.1, the deflation-subspace matrix is chosen such that the real inverse of the resulting Galerkin matrix always exists. This variant is basically the deflation method as considered in prior chapters. Moreover, the deflation matrix in Variant 5.3 relies on the natural choice of deflation vectors and the pseudo-inverse of the Galerkin matrix. This variant is most related to the general deflation method applied to invertible coefficient matrices.

The proposed deflation variants are analyzed and compared. We show that the corresponding preconditioned-deflated coefficient matrices are the same, for certain deflation vectors and small perturbations of A to construct $\bar{A}$. Hence, the convergence behavior of the deflation variants are expected to be comparable. It depends on the implementation and practical wishes of the user which variant can be best used. Results of numerical experiments confirm the theoretical results, and show the good performance and equivalences of the deflation variants. Variant 5.3 is not considered yet due to its implementation complexity. This variant is further analyzed in Chapter 8.

Chapter 6

Comparison of Two-Level PCG Methods – Part I

6.1 Introduction

In the previous chapters, the deflation method has been analyzed extensively. This method was originally used by Nicolaides [108] and Dostal [40] to accelerate the convergence of PCG, and several contributions were made since then, including [45, 82, 122, 173]. Following [108], the convergence of PCG can be improved if the components of the residual associated with the smallest eigenvalues are no longer present during the iteration process. The corresponding preconditioner of the deflation method consists of a combination of a traditional single-level preconditioner, M^{-1}, and a second-level preconditioner, P. In this case, the deflation preconditioner can be regarded as a two-level preconditioner, and the resulting method can be interpreted as a two-level PCG (2L-PCG) method, see also Section 1.2.

In addition to the traditional preconditioner, a second kind of preconditioner is incorporated in each 2L-PCG method to improve the conditioning of the coefficient matrix, so that the resulting approach effectively treats the effect of both small and large eigenvalues. Besides the field of deflation, a PCG method in combination with a preconditioner based on multigrid (MG) or domain decomposition method (DDM), can be seen as a 2L-PCG method, since most of these methods rely on preconditioning on two levels. Probably the simplest form of 2L-PCG is CG combined with a two-grid method. In this case, together with the fine-grid linear system from which the approximate solution of the original PDEs is computed, a coarse-grid system is built based on a predefined coarse grid. From MG point of view, the (second-level) coarse-grid system is used to reduce the slow-varying, low-frequency components of the error, which could not be effectively reduced on the (first-level) fine grid. These low-frequency components of the error are associated with the small eigenvalues of the coefficient matrix. The high-frequency components are, however, effectively handled on the fine grid. The latter is associated with the large eigenvalues of the coefficient matrix.

In order to attain a further reduction of the error, however, it is required that the slow-varying components of the error on the fine grid are well approximated on the coarse grid. While the two-grid method on its own is a good method for some class of problems, a better convergence bound can be obtained if it is used as a preconditioner for PCG.

The two-grid preconditioning has been known for a long time, dating back at least to the 1930s. Its potential was first exploited by Fedorenko and Bakhalov in the 1960s, and later by Brandt [21] and Hackbusch [69], which paved the way to the birth of MG methods. We refer to [151, 178] and references therein for more details. In the two-grid method, the second-level problem is derived from the systematic coarsening of the underlying, predefined fine grid, and, henceforth, has a geometrical relationship with the fine grid, see [126, 150]. MG methods can be useful for solving problems with high coefficient ratios, see, e.g., [164–166]. On the other hand, a more general two-level method is obtained if the second-level problem is built by using only the coefficient matrix. This generalization results in the so-called algebraic multigrid (AMG) method. AMG can be very effective for elliptic problems on unstructured grids, even with high coefficient ratios [30, 118]. Similar observations can also be made for DDM. As in the case of MG, DDM by its own is an effective method for some class of problems, and the convergence can be further improved if it is combined with CG. Abstract balancing Neumann-Neumann (BNN) methods [89–91] are well-known examples in this field. Other examples of DDM, which are useful for solving problems with high-coefficient ratios, can be found in [126, 150].

At first glance, 2L-PCG methods from deflation, DDM and MG seem to be different. For example, in deflation, eigenvectors or eigenvector approximations associated with the unfavorable eigenvalues are often used as projection vectors. In contrast, MG or DDM use special projection vectors, which represent interpolations between the fine-grid and coarse-grid subspace. Surprisingly, from algebraic/abstract point of view, the 2L-PCG methods from the three fields are quite comparable or even equivalent. For example, the deflation operator is the same as the multigrid operator if no pre- and post-smoothing are performed, i.e., the deflation operator is the same as the coarse-grid correction operator in MG. While deflation is quite successful within Krylov methods, using only coarse-grid correction does not lead to a successful MG method [150]. This motivates us to approach the 2L-PCG methods from an abstract point of view. In this chapter, we introduce a generalized formulation for deflation, MG and DDM, resulting in a unified theory.

In [103–105], theoretical comparisons are presented for the deflation, BNN and additive coarse-grid correction (AD) methods. It is proven, using spectral analysis among other techniques, that the deflation method is expected to yield faster convergence compared to the other two methods. For certain starting vectors, deflation and BNN even produce the same iterates. Although these methods seem to be comparable, deflation is not always robust, as observed in the limited numerical experiments provided in [103–105]. The residuals may stagnate or even diverge during the iteration process, if the required accuracy is (too) high. The AD and BNN preconditioners are more robust, but they also have drawbacks: BNN is more expensive to apply, and AD is slower to converge. It is known in the literature, see [89, 150], that the implementation of the robust BNN method can be made less expensive, so that the total amount of work is comparable to deflation and AD. In this case, new 2L-PCG methods can be defined and they can be interpreted as reduced variants of the original BNN method. However, not much is known about the robustness of these reduced variants, as most theoretical results apply to only the original method. One of the subjects of this chapter is to deal with this issue in detail. More recent papers about the robustness of 2L-PCG methods can be further found in, e.g., [16, 59, 65].

In this chapter, the two-level PCG methods are compared theoretically by investigating the corresponding spectral properties, their numerical implementations, and equivalences. Thereafter, the main focus is on numerical experiments, where the 2L-PCG methods are tested for their convergence properties and robustness. The effect of different implementations is analyzed, and the results are related to the theory. Note that, in [103–105], the comparisons of deflation, BNN and AD are mainly based on theoretical aspects, whereas only a limited numerical comparison is presented. This chapter focuses equally on theoretical and numerical aspects of these of 2L-PCG methods. The following questions are answered in this chapter:

- what is the relation and equivalences between the two-level PCG methods?
- which two-level PCG methods can be applied, if one uses inaccurate coarse solvers, severe termination criteria or perturbed starting vectors?
- is there a two-level preconditioner that is as robust as BNN and as cheap and fast as deflation?

Similar to the 2L-PCG methods considered in this chapter, there are some other variants known as augmented subspace CG [46], deflated Lanczos method [122], and the Odir and Omin version of CG combined with extra vectors [6]. We refer to [122, 124] for a discussion and comparison of these methods. In the overview paper [124], more details and references are also given about Krylov subspace methods with respect to inexact computations and their equivalences. Another comparison of 2L-PCG methods is carried out in [59], where methods known as Init-CG, Def-CG, Proj-CG and SLRU are compared. The aim of that paper is to obtain an optimal solver, that exploits accurate spectral information about the coefficient matrix in an efficient way. In contrast to that paper, our comparison of 2L-PCG methods is done without any spectral information of the coefficient matrix, A. Recently, novel additive and multiplicative two-level preconditioners applied to general linear systems are considered in [26]. These spectral preconditioners are based on multigrid ideas, and can be well analyzed for special choices of the restriction and prolongation operators.

This chapter is organized as follows. In Section 6.2, we introduce and discuss two-level PCG methods. Section 6.3 is devoted to the theoretical comparison of these methods. Subsequently, the numerical comparison of the two-level PCG methods is carried out in Section 6.4. Finally, some concluding remarks are given in Section 6.5.

Remark 6.1. *In this chapter, we restrict ourselves to a nonsingular coefficient matrix, A, for convenience. However, all main results are generalizable for A that is singular, see [85].*

6.2 Two-Level PCG Methods

In this section, two-level PCG methods are defined and motivated, but we start with some terminology (cf. Definition 3.1), and a preliminary result (cf. Lemma 3.2).

Definition 6.1. *Suppose that an SPD coefficient matrix, $A \in \mathbb{R}^{n\times n}$, and a deflation subspace matrix, $Z \in \mathbb{R}^{n\times k}$, with full rank and $k < n$ are given. Then, we define the invertible Galerkin matrix, $E \in \mathbb{R}^{k\times k}$, the correction matrix, $Q \in \mathbb{R}^{n\times n}$, and the deflation matrix, $P \in \mathbb{R}^{n\times n}$, as follows:*

$$P := I - AQ, \quad Q := ZE^{-1}Z^T, \quad E := Z^T AZ.$$

Lemma 6.1. *Let $A \in \mathbb{R}^{n\times n}$ and $Z \in \mathbb{R}^{n\times k}$ be given. Suppose that Q and P are given as in Definition 6.1. Then, the following equalities hold:*

(a) $P = P^2$;

(b) $PA = AP^T$;

(c) $P^T Z = \mathbf{0}_{n,k}, \;\; P^T Q = \mathbf{0}_{n,n}$;

(d) $PAZ = \mathbf{0}_{n,k}, \; PAQ = \mathbf{0}_{n,n}$;

(e) $QA = I - P^T, \;\; QAZ = Z, \;\; QAQ = Q$;

(f) $Q^T = Q$.

Remark 6.2.

- *E is SPD for any full-rank Z, since A is SPD.*
- *In this chapter, $k \ll n$ does not necessarily hold. But if $k \ll n$ does hold, then E is a matrix with small dimensions, so that it can be easily computed and factored.*

From an abstract point of view, all two-level preconditioners of the methods consist of an arbitrary M^{-1}, combined with one or more matrices P and Q. In the next subsection, we give an explanation of the choices for these matrices in the different fields. Nevertheless, from our point of view, matrices M^{-1} and Z are arbitrary (but fixed) for each 2L-PCG method. In this way, the abstract setting allows us to compare the methods in terms of operators, although they have their roots in different fields.

6.2.1 Background of the Matrices in Domain Decomposition, Multigrid and Deflation

In the 2L-PCG methods used in DDM, such as the BNN and (two-level) additive Schwarz methods, the single-level preconditioner, M^{-1}, consists of local exact or inexact solves on subdomains. Moreover, Z describes a prolongation (or interpolation) operator, while Z^T is a restriction operator based on the subdomains. In this case, E is called the coarse-grid (or Galerkin) matrix. In order to speed up the convergence of the additive Schwarz method, a coarse-grid correction matrix, Q, can be added, which is a so-called additive coarse-grid correction. Finally, P can be interpreted as a subspace correction, in which each subdomain is agglomerated into a single cell. More details can be found in [126, 150].

In the MG approach, Z and Z^T are also the prolongation and restriction operators, respectively, where typical MG grid-transfer operators allow interpolation between neighboring subdomains. E and Q are again the coarse-grid (or Galerkin) and coarse-grid correction matrices, respectively, corresponding to the Galerkin approach. Matrix P can be interpreted as the algebraic form of the coarse-grid correction step in MG, where linear systems with E are usually solved recursively. In the context of MG methods, M^{-1} should work as a smoother (also known as relaxation method) that eliminates the high-frequency errors in the residuals and often corresponds to Jacobi or Gauss-Seidel iterations. Before or after the smoothing step(s), a coarse-grid correction, P, is applied to remove the slow frequencies in the residuals. We refer to [69, 151, 178] for more details.

As discussed in the previous chapter, M^{-1} is often a traditional preconditioner, such as an Incomplete Cholesky factorization, in deflation methods. Furthermore, the deflation-subspace matrix, Z, consists of so-called deflation vectors, which are used in the deflation matrix, P. In this case, the column space of Z builds the deflation subspace, i.e., the space to be projected out of the residuals. It often consists of eigenvectors, approximations of eigenvectors, or piecewise-constant vectors, which are strongly related to DDM. If one chooses eigenvectors, the corresponding eigenvalues would be shifted to zero in the spectrum of the deflated matrix. This fact has motivated the name ‘deflation method’. In the literature, the deflation two-level preconditioner is also known as the spectral preconditioner, see, e.g., [59]. Usually, systems with E are solved directly, using, e.g., a Cholesky decomposition.

6.2.2 General Linear Systems

The general linear system, which is the basis for two-level PCG methods, is

$$\mathcal{P}\mathcal{A}\mathbf{x} = \mathbf{b}, \quad \mathcal{P}, \mathcal{A} \in \mathbb{R}^{n \times n}. \tag{6.1}$$

In the standard (single-level) PCG method, $\mathbf{x} = x$ is the solution of the original linear system, $Ax = b$, $\mathcal{A} = A$ is the SPD coefficient matrix, $\mathcal{P} = M_{\text{PREC}}^{-1}$ represents a traditional SPD preconditioner, and $\mathbf{b} = M_{\text{PREC}}^{-1}b$ is the right-hand side, see also [63, 97].

We denote this method by 'Traditional PCG' (PREC).

Next, $\mathcal{A}$ may also be a combination of A and P, such that $\mathcal{A}$ is SP(S)D, while $\mathcal{P}$ remains a traditional preconditioner. Note that this does not cause difficulties for the CG process, since it is robust for SPSD matrices as long as the linear system is consistent (cf. Chapter 1), see [77]. Furthermore, instead of choosing one traditional preconditioner for $\mathcal{P}$, we can combine different single-level and second-level preconditioners in an additive or multiplicative way, which is illustrated below.

The additive combination of two SPD preconditioners, C_1 and C_2, leads to $\mathcal{P}_{a_2}$, given by

$$\mathcal{P}_{a_2} := C_1 + C_2, \tag{6.2}$$

which is also SPD. Of course, the summation of the preconditioners can be done with different weights for C_1 and C_2. Moreover, (6.2) can be easily generalized to $\mathcal{P}_{a_i}$ for more SPD preconditioners, $C_1, C_2, \ldots, C_i$.

The multiplicative combination of preconditioners can be explained by considering the stationary iterative methods induced by the preconditioner. Assuming that C_1 and C_2 are two SPD preconditioners, we can combine

$$\left\{ \begin{array}{lcl} x^{i+\frac{1}{2}} & = & x^i + C_1(b - Ax^i); \\ x^{i+1} & = & x^{i+\frac{1}{2}} + C_2(b - Ax^{i+\frac{1}{2}}), \end{array} \right. \tag{6.3}$$

to obtain $x^{i+1} = x^i + \mathcal{P}_{m_2}(b - Ax^i)$, with

$$\mathcal{P}_{m_2} := C_1 + C_2 - C_2AC_1, \tag{6.4}$$

which can be interpreted as the multiplicative operator consisting of two preconditioners. Subsequently, C_1 and C_2 could again be combined with another SPD preconditioner, C_3, in a multiplicative way, yielding

$$\mathcal{P}_{m_3} = C_1 + C_2 + C_3 - C_2AC_1 - C_3AC_2 - C_3AC_1 + C_3AC_2AC_1. \tag{6.5}$$

This can also be generalized to $\mathcal{P}_{m_i}$ for $C_1, C_2, \ldots, C_i$.

6.2.3 Definition of the Two-Level PCG Methods

The two-level PCG methods that are considered in this chapter are presented and motivated below.

Additive Method

If one substitutes a traditional preconditioner, $C_1 := M^{-1}$, and a coarse-grid correction matrix, $C_2 := Q$, into the additive combination given in (6.2), this yields

$$\mathcal{P}_{\mathrm{AD}} = M^{-1} + Q. \tag{6.6}$$

Using the additive Schwarz preconditioner for M^{-1}, the abstract form (6.6) includes the additive coarse-grid correction preconditioner [19]. The BPS preconditioner, introduced by Bramble, Pasciak and Schatz in [19], can be written as (6.6). This is further analyzed in, e.g., [41, 42, 111]. If the multiplicative Schwarz preconditioner is taken as M^{-1}, we obtain the Hybrid-2 preconditioner [150, p. 47]. In the MG language, $\mathcal{P}_{\text{AD}}$ is sometimes called an additive multigrid preconditioner, see [11]. In this chapter, the resulting method associated with $\mathcal{P}_{\text{AD}}$ is called 'Additive Coarse-Grid Correction' (AD).

Deflation Methods

The deflation technique is exploited in several papers, amongst them are [56, 58, 82, 93, 94, 99, 103, 104, 108, 122, 173]. Some differences in the formulations can be observed in these papers, while they are basically mathematically equivalent. One of these formulations of the deflation method is presented in Chapter 3. This method is called 'Deflation Variant 1' (DEF1) in this chapter.

An alternative way to describe the deflation technique is to start with an arbitrary vector, $\bar{x}$, and choose $x_0 := Qb + P^T\bar{x}$. Then, the solution of $Ax = b$ can be constructed from the deflated system

$$AP^Ty = r_0, \quad r_0 := b - Ax_0. \tag{6.7}$$

The nonunique solution, y, is then used to obtain $\bar{y} := P^Ty$. It can be shown that $x = x_0 + \bar{y}$ is the unique solution of $Ax = b$. Similarly, deflated system (6.7) can also be solved with a single-level preconditioner, M^{-1}, leading to

$$M^{-1}AP^Ty = M^{-1}r_0, \quad r_0 := b - Ax_0. \tag{6.8}$$

Similar to the procedure for the unpreconditioned case, x can be found from the nonuniquely determined solution, y, of (6.8). This leads to an algorithm that is based on the projection operator P^TM^{-1}, rather than $M^{-1}P$ as in DEF1, see [82, 108, 122]. Hence, we solve

$$P^TM^{-1}Ax = P^TM^{-1}b, \tag{6.9}$$

where the iterates, $\{x_i\}$, within the algorithm are uniquely determined as long as $x_0 := Qb + P^T\bar{x}$ is used. We treat this in more detail in Section 6.3.2. The resulting method is denoted by 'Deflation Variant 2' (DEF2). Observe that Eq. (6.9) cannot be written in the form of (6.1) with an SPD operator $\mathcal{P}$ and an SPSD matrix $\mathcal{A}$. Fortunately, in Section 6.3.2, it is shown that (6.9) is equivalent to a linear system that is in the form of (6.1).

Remark 6.3. *The main difference between DEF1 and DEF2 is their flipped operators. In addition, if we define the 'uniqueness'-operation as computing $w := Qb + P^T\tilde{w}$, for a given vector $\tilde{w}$, this operation is carried out at the end of the iteration process in DEF1, so that an arbitrarily chosen starting vector, x_0, can be used. On the other hand, this operation is applied prior to the iteration process in DEF2, which can be interpreted as adopting a special starting vector. As a consequence, they have different robustness properties with respect to starting vectors, see Section 6.4.5.*

Adapted Deflation Methods

If one applies $C_1 := Q$ and $C_2 := M^{-1}$ in a multiplicative combination as given in (6.4), then this yields

$$\mathcal{P}_{\text{A-DEF1}} = M^{-1}P + Q, \tag{6.10}$$

see [134] for more details. In the MG language, this operator results from a nonsymmetric multigrid V(1,0)-cycle iteration scheme, where one first applies a coarse-grid correction, followed by a smoothing step. Note that, although Q and M^{-1} are SPD preconditioners, (6.10) is a nonsymmetric operator, and, even more, it is not symmetric with respect to the inner product induced by A. In addition, $\mathcal{P}_{\text{A-DEF1}}$ can also be interpreted as an adapted deflation preconditioner, since $M^{-1}P$ from DEF1 is combined in an additive way with a coarse-grid correction, Q. Hence, the resulting method corresponding to $\mathcal{P}_{\text{A-DEF1}}$ is denoted by the 'Adapted Deflation Variant 1' (A-DEF1).

Subsequently, we can also reverse the order of Q and M^{-1} (i.e., $C_1 := M^{-1}$ and $C_2 := Q$) in (6.4), giving us

$$\mathcal{P}_{\text{A-DEF2}} = P^T M^{-1} + Q. \tag{6.11}$$

Using an additive Schwarz preconditioner for M^{-1}, $\mathcal{P}_{\text{A-DEF2}}$ is the two-level Hybrid-II Schwarz preconditioner [126, p. 48]. In MG methods, $\mathcal{P}_{\text{A-DEF2}}$ is the nonsymmetric multigrid V(0,1)-cycle preconditioner, where M^{-1} is used as a smoother. Similar to A-DEF1, $\mathcal{P}_{\text{A-DEF2}}$ is nonsymmetric. Fortunately, we see in Section 6.3.2 that A-DEF2 is equivalent to a method based on a symmetric operator. As in the case of $\mathcal{P}_{\text{A-DEF1}}$, the operator $\mathcal{P}_{\text{A-DEF2}}$ can also be regarded as an adapted deflation preconditioner, since $P^T M^{-1}$ from DEF2 is combined with Q, in an additive way. Accordingly, the resulting method is denoted by the 'Adapted Deflation Variant 2' (A-DEF2) method.

Abstract Balancing Methods

The operators $\mathcal{P}_{\text{A-DEF1}}$ and $\mathcal{P}_{\text{A-DEF2}}$ can be symmetrized by using the multiplicative combination of three preconditioners. If one substitutes $C_1 := Q$, $C_2 := M^{-1}$ and $C_3 := Q$ into (6.5), we obtain

$$\mathcal{P}_{\text{BNN}} = P^T M^{-1} P + Q.$$

The operator $\mathcal{P}_{\text{BNN}}$ is a well-known operator in DDM. In combination with an additive Schwarz preconditioner for M^{-1}, and after some scaling and special choices of Z, the operator $\mathcal{P}_{\text{BNN}}$ is known as the Balancing-Neumann-Neumann preconditioner, introduced in [89], and further analyzed in, e.g., [43, 90, 91, 114, 150]. In the abstract form, $\mathcal{P}_{\text{BNN}}$ is called the Hybrid-1 preconditioner [150, p. 34]. Here, we call it 'Abstract Balancing Neumann-Neumann' (BNN).

Of course, $\mathcal{P}_{\text{A-DEF1}}$ and $\mathcal{P}_{\text{A-DEF2}}$ could also be symmetrized by using twice M^{-1} instead of Q (i.e., $C_1 := M^{-1}, C_2 := Q$ and $C_3 := M^{-1}$) in Eq. (6.5). This results in the well-known symmetric multigrid V(1,1)-cycle iteration scheme, where a pre-smoothing step is followed by a coarse-grid correction and ended with a post-smoothing step. The resulting preconditioner is then explicitly given by

$$\mathcal{P} = M^{-1}P + P^T M^{-1} + Q - M^{-1}PAM^{-1}. \tag{6.12}$$

Note that this operator also follows by combining the A-DEF1 and A-DEF2 operators in a multiplicative way. In (6.12), a structural difference can be observed between BNN and the multigrid $V(1,1)$-cycle iteration. As mentioned before, in MG, M^{-1} is the smoothing operator, and the coarse-grid system typically has half of the order of the original system per direction. Hence, smoothing is cheap compared to solving the coarse-grid system. In this case, symmetrizing with another smoothing step is natural. In DDM, M^{-1} contains all local solves of the subdomain systems, while the dimension of the Galerkin system is typically much smaller than the dimension of the original system. Hence, a symmetrization with a coarse-grid solve is inexpensive in DDM. Except for special choices of the restriction and prolongation operator, see, e.g., [26], it is generally difficult to analyze the spectra of the system preconditioned by (6.12) in comparison with the other methods described in this chapter. Therefore, we do not include this preconditioner in our comparison, but we focus on this issue in Chapter 7.

Moreover, we also consider two variants of BNN. In the first variant, we omit the term Q from $\mathcal{P}_{\text{BNN}}$, giving us

$$\mathcal{P}_{\text{R-BNN1}} \quad = \quad P^T M^{-1} P,$$

which remains a symmetric operator. To our knowledge, $\mathcal{P}_{\text{R-BNN1}}$ is unknown in the literature, and this is the first time that its properties are analyzed. The corresponding method is called 'Reduced BNN Variant 1' (R-BNN1). Next, in the second variant of BNN, we omit both the P and Q terms from $\mathcal{P}_{\text{BNN}}$, resulting in

$$\mathcal{P}_{\text{R-BNN2}} = P^T M^{-1}, \tag{6.13}$$

and this method is denoted by 'Reduced BNN Variant 2' (R-BNN2). Notice that the operators of both R-BNN2 and DEF2 are equal, i.e.,

$$\mathcal{P}_{\text{DEF2}} = \mathcal{P}_{\text{R-BNN2}} = P^T M^{-1},$$

where only the implementation is different, see Section 6.2.4. In fact, the implementation of DEF2 is equivalent to the approach as applied in, e.g., [122], where the deflation method is derived by combining a deflated Lanczos procedure and the standard CG algorithm. On the other hand, R-BNN2 is the approach where deflation is incorporated into the CG algorithm in a direct way [82], and it is also the approach where a hybrid variant is employed in DDM [150]. Finally, as mentioned earlier, $P^T M^{-1}$ is a nonsymmetric preconditioner, but it is shown in Section 6.3.2 that both $\mathcal{P}_{\text{R-BNN1}}$ and $\mathcal{P}_{\text{R-BNN2}}$ are equivalent to $\mathcal{P}_{\text{BNN}}$ for certain starting vectors. Consequently, we classify these methods as variants of the original BNN method, rather than as variants of deflation methods.

6.2.4 Aspects of Two-Level PCG Methods

For the sake of completeness, the 2L-PCG methods that are considered in this chapter are given in Table 6.1. More details about the methods can be found in the references, given in the last column of this table. Subsequently, the implementation and the computational cost of these methods are considered in this subsection.

Name	Method	Operator	References
PREC	Traditional PCG	M^{-1}	[63, 97]
AD	Additive Coarse-Grid Correction	$M^{-1} + Q$	[19, 126, 150]
DEF1	Deflation Variant 1	$M^{-1}P$	[173]
DEF2	Deflation Variant 2	P^TM^{-1}	[82, 108, 122]
A-DEF1	Adapted Deflation Variant 1	$M^{-1}P + Q$	[126, 151, 178]
A-DEF2	Adapted Deflation Variant 2	$P^TM^{-1} + Q$	[126, 151, 178]
BNN	Abstract Balancing	$P^TM^{-1}P + Q$	[89]
R-BNN1	Reduced Balancing Variant 1	$P^TM^{-1}P$	–
R-BNN2	Reduced Balancing Variant 2	P^TM^{-1}	[89, 150]

Table 6.1: List of methods that are compared in this chapter. The operator of each method can be interpreted as the preconditioner $\mathcal{P}$, given in (6.1) with $\mathcal{A} = A$. Where possible, references to the methods and their implementations are presented in the last column.

Implementation Issues

The implementation of the 2L-PCG methods given in Table 6.1 can be presented in one algorithm, resulting in a generalized 2L-PCG method, see Algorithm 7. For each method, the corresponding matrices, $\mathcal{M}_i$, and vectors, $\mathcal{V}_{\text{start}}$ and $\mathcal{V}_{\text{end}}$, are presented in Table 6.2. For more details, we refer to [134].

Algorithm 7 Generalized Two-Level PCG Method for solving $Ax = b$.

```
Select arbitrary $\bar{x}$ and $\mathcal{V}_{\text{start}}, \mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{V}_{\text{end}}$ from Table 6.2
Set $x_0 := \mathcal{V}_{\text{start}}$, and compute $r_0 := b - Ax_0$, $y_0 := \mathcal{M}_1 r_0$, $p_0 := \mathcal{M}_2 y_0$
for $j := 0, 1, \ldots,$ until convergence do
    $w_j := \mathcal{M}_3 A p_j$
    $\alpha_j := \frac{(r_j, y_j)}{(p_j, w_j)}$
    $x_{j+1} := x_j + \alpha_j p_j$
    $r_{j+1} := r_j - \alpha_j w_j$
    $y_{j+1} := \mathcal{M}_1 r_{j+1}$
    $\beta_j := \frac{(r_{j+1}, y_{j+1})}{(r_j, y_j)}$
    $p_{j+1} := \mathcal{M}_2 y_{j+1} + \beta_j p_j$
end for
$x_{\text{it}} := \mathcal{V}_{\text{end}}$
```

From Algorithm 7 and Table 6.2, it can be observed that one or more preconditioning and projection operations are carried out in the steps where the matrices $\mathcal{M}_i$,

Method	$\mathcal{V}_{\text{start}}$	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{V}_{\text{end}}$
PREC	$\bar{x}$	M^{-1}	I	I	x_{j+1}
AD	$\bar{x}$	$M^{-1}+Q$	I	I	x_{j+1}
DEF1	$\bar{x}$	M^{-1}	I	P	$Qb+P^T x_{j+1}$
DEF2	$Qb+P^T\bar{x}$	M^{-1}	P^T	I	x_{j+1}
A-DEF1	$\bar{x}$	$M^{-1}P+Q$	I	I	x_{j+1}
A-DEF2	$Qb+P^T\bar{x}$	$P^T M^{-1}+Q$	I	I	x_{j+1}
BNN	$\bar{x}$	$P^T M^{-1}P+Q$	I	I	x_{j+1}
R-BNN1	$Qb+P^T\bar{x}$	$P^T M^{-1}P$	I	I	x_{j+1}
R-BNN2	$Qb+P^T\bar{x}$	$P^T M^{-1}$	I	I	x_{j+1}

Table 6.2: Choices of parameters for each method, used in the generalized two-level PCG method as given in Algorithm 7.

with $i = 1, 2, 3$, are involved. For most 2L-PCG methods, these steps are combined to obtain the preconditioned/projected residuals, $\{y_i\}$. DEF2 is the only method where a projection step is applied to the search directions, $\{p_i\}$. Likewise, DEF1 is the only method where the projection is performed to create w_j. In this case, $r_{j+1} = P(b - Ax_{j+1})$ should hold, while $r_{j+1} = b - Ax_{j+1}$ is satisfied for the other methods. As discussed in Section 3.5.3, termination criterion (2.23) based on $\{r_i\}$ can be used to compare the 2L-PCG methods in a fair way.

Remark 6.4.

- *Note that Algorithms 3 (PCG) and 6 (DPCG) are particular choices of Algorithm 7.*

- *Notice that we use the same arbitrary starting vector, $\bar{x}$, in each method, but the actual starting vector, $\mathcal{V}_{start}$, may differ for each method. Likewise, it can also be noticed that the ending vector, $\mathcal{V}_{end}$, is the same for all methods, except for DEF1.*

- *A 2L-PCG method is guaranteed to converge if $\mathcal{P}$, as given in (6.1), is SPD or can be transformed into an SPD matrix, see, e.g., [51] for more details. This is obviously the case for PREC, AD, DEF1 and BNN. It can be shown that DEF2, A-DEF2, R-BNN1 and R-BNN2 also rely on appropriate operators, where $\mathcal{V}_{start} = Qb + P^T\bar{x}$ plays an important role in this derivation, see Theorem 6.4. A-DEF1 is the only method which does not have an SPD operator and cannot be decomposed or transformed into an SPD operator, $\mathcal{P}$. Therefore, it is not guaranteed that A-DEF1 always works, but it performs rather satisfactorily for most of the test cases considered in Section 6.4.*

Computational Cost

The computational cost of each method depends not only on the choices of M^{-1} and Z, but also on the implementation and the storage of the matrices. It is easy to see that, for each iteration, PREC requires 1 matrix-vector multiplication (MVM), 2 inner products (IP), 3 vector updates (VU) and 1 preconditioning step.

Note that AZ and E should be computed and stored beforehand, so that only one MVM with A is required in each iteration of the 2L-PCG methods. Moreover, we distinguish between two cases considering Z and AZ:

- Z is sufficiently sparse, so that Z and AZ can be stored in approximately two vectors;

- Z is dense, so that Z and AZ are full matrices.

The first case, which is the best case in terms of efficiency, occurs often in DDM, where the columns of Z correspond to subdomains, while the second (and worst) case occurs, for example, in approximated eigenvector deflation methods. Of course, there are many relevant cases where Z and AZ cannot be stored within two vectors, while dense storage of these matrices is not necessary; however, this is not considered in this chapter for convenience. For each 2L-PCG method, we give the extra computational cost per iteration above that of PREC, see Table 6.3. In the table, the number of operations of the form Py and Qy, for a given vector, y, per iteration is also provided. Note that, if both Py and Qy should be computed for the same vector, y, such as in A-DEF1 and BNN, then Qy can be determined efficiently, since it only requires one IP if Z is sparse, or one MVM if Z is dense.

From Table 6.3, it can be seen that AD is obviously the cheapest method per iteration, while BNN and R-BNN1 are the most expensive 2L-PCG methods, since two operations with P and P^T are involved. With respect to the implementation, this implies that AD only needs two inner/matrix-vector products and one Galerkin system solves extra compared to PREC, while both BNN and R-BNN1 require obviously more inner/matrix-vector products, Galerkin system solves and additional vector updates. Finally, we observe that using a 2L-PCG method is only efficient if Z is sparse, or if the number of projection vectors is relatively small in the case of a dense matrix, Z.

Remark 6.5.

- *The given computational cost in Table 6.3 is based on the resulting abstract operators and implementation as presented in Algorithm 7. As mentioned in Sections 6.2.2 and 6.2.3, the methods have different origins with their own specific and optimal implementation, so that the amount of work for each method can be less as suggested in Table 6.3.*

- *We emphasize that the parameters of the 2L-PCG methods that are compared can be arbitrary, so that the comparison between these methods is based on their abstract versions. This means that the results of the comparison are valid for any full-rank matrix* Z *and SPD matrices* A *and* M^{-1}.

- *In Chapter 8, the efficiency and implementation of DEF1 and A-DEF2 for a specific choice of* Z *are examined in more detail.*

	Theory		Implementation		
Method	Py, P^Ty	Qy	IP / MVM	VU	GSS
AD	0	1	2	0	1
DEF1	1	0	2	1	1
DEF2	1	0	2	1	1
A-DEF1	1	1	3	1	1
A-DEF2	1	1	4	1	2
BNN	2	1	5	2	2
R-BNN1	2	0	4	2	2
R-BNN2	1	0	2	1	1

Table 6.3: Extra computational cost per iteration of the two-level PCG methods compared to PREC. IP = inner products, MVM = matrix-vector multiplications, VU = vector updates and GSS = Galerkin system solves. Note that IP holds for sparse Z and MVM holds for dense Z.

We note that efficiency and implementation issues have not been taken into consideration so far in this thesis. The aim of this chapter is to deal with those issues in more detail. We show that a good implementation of the deflation method is essential in order to obtain a powerful and efficient method.

6.3 Theoretical Comparison

In this section, a comparison of eigenvalue distributions corresponding to the operators of the 2L-PCG methods is carried out, and, thereafter, some equivalence relations between the methods are derived. Although some parts of the results are closely related to results known in the literature [103, 104, 150], we include them here in order to make this chapter self-contained.

6.3.1 Spectral Analysis of the Methods

We start this subsection with a definition.

Definition 6.2. *Suppose that arbitrary matrices $C, D \in \mathbb{R}^{n\times n}$ have the following spectra:*

$$\sigma(C) := \{\lambda_1, \lambda_2, \ldots, \lambda_n\}, \quad \sigma(D) := \{\mu_1, \mu_2, \ldots, \mu_n\},$$

respectively. Then, the addition of two sets, $\sigma(C)$ and $\sigma(D)$, is defined as

$$\sigma(C) + \sigma(D) := \{\mu_1 + \lambda_1, \mu_2 + \lambda_2, \ldots, \mu_n + \lambda_n\}.$$

In Section 3.5.2, we have shown that

$$\kappa\left(M^{-1}PA\right) \leq \kappa\left(M^{-1}A\right),$$

for any SPD matrices A and M^{-1}, and any full-rank Z. This means that the two-level preconditioned matrix corresponding to DEF1 is better conditioned than that of PREC. It follows from the analysis below that the two-level preconditioned matrix corresponding to PREC is always worse conditioned compared to the other 2L-PCG methods.

In [103, 104], it is shown that the condition number of DEF1 is not worse than that of both AD and BNN, i.e.,

$$\left\{ \begin{array}{lcl} \kappa\left(M^{-1}PA\right) & \leq & \kappa\left(M^{-1}A+QA\right); \\ \kappa\left(M^{-1}PA\right) & \leq & \kappa\left(P^TM^{-1}PA+QA\right), \end{array} \right. \tag{6.14}$$

for all full-rank Z and SPD matrices A and M^{-1}.

Remark 6.6. *Inequalities such as (6.14) cannot be derived between between AD and BNN. One would expect the condition number associated with BNN to be below that associated with AD, but this is not always the case, see [105] for a counterexample.*

In addition to the comparisons of AD, DEF1 and BNN performed in [103–105], more relations between the eigenvalue distribution of these and other 2L-PCG methods are presented below. We first show in Theorem 6.1 that DEF1, DEF2, R-BNN1 and R-BNN2 have identical spectra, and that the same is true for BNN, A-DEF1 and A-DEF2.

Theorem 6.1. *Suppose that $A, M^{-1} \in \mathbb{R}^{n\times n}$ are SPD. Let Q and P be as given in Definition 6.1. Then, the following two statements hold:*

- $\sigma\left(M^{-1}PA\right) = \sigma\left(P^TM^{-1}A\right) = \sigma\left(P^TM^{-1}PA\right)$;
- $\sigma\left((P^TM^{-1}P+Q)A\right) = \sigma\left((M^{-1}P+Q)A\right) = \sigma\left((P^TM^{-1}+Q)A\right)$.

Proof. Using Lemma A.1 and Lemma 6.1, we obtain immediately

$$\sigma\left(M^{-1}PA\right) = \sigma\left(AM^{-1}P\right) = \sigma\left(P^TM^{-1}A\right),$$

and

$$\begin{array}{rcl} \sigma\left(M^{-1}PA\right) & = & \sigma\left(M^{-1}P^2A\right) \\ & = & \sigma\left(M^{-1}PAP^T\right) \\ & = & \sigma\left(P^TM^{-1}PA\right), \end{array}$$

which proves the first statement. Moreover, we also have that

$$\begin{array}{rcl} \sigma\left(P^TM^{-1}PA+QA\right) & = & \sigma\left(P^TM^{-1}PA-P^T+I\right) \\ & = & \sigma\left((M^{-1}PA-I)P^T\right)+\sigma(I) \\ & = & \sigma\left(M^{-1}P^2A-P^T\right)+\sigma(I) \\ & = & \sigma\left(M^{-1}PA+QA\right), \end{array}$$

and, likewise,

$$\begin{array}{rcl} \sigma\left(P^TM^{-1}A+QA\right) & = & \sigma\left(P^TM^{-1}A-P^T\right)+\sigma(I) \\ & = & \sigma\left(AM^{-1}P-P\right)+\sigma(I) \\ & = & \sigma\left(PAM^{-1}P-P\right)+\sigma(I) \\ & = & \sigma\left(P^TM^{-1}AP^T-P^T\right)+\sigma(I) \\ & = & \sigma\left(P^TM^{-1}PA+QA\right), \end{array}$$

which completes the proof of the second statement. $\square$

As a consequence of Theorem 6.1, DEF1, DEF2, R-BNN1 and R-BNN2 can be interpreted as one class of 2L-PCG methods having the same spectral properties, whereas BNN, A-DEF1 and A-DEF2 lead to another class of 2L-PCG methods. These two classes can be related to each other by [104, Thm. 2.8], which states that if $\sigma(M^{-1}PA) = \{0, \ldots, 0, \mu_{k+1}, \ldots, \mu_n\}$ is given, then $\sigma(P^T M^{-1}PA+QA) = \{1, \ldots, 1, \mu_{k+1}, \ldots, \mu_n\}$. We can show that the reverse statement also holds. These results are given in Theorem 6.2.

Theorem 6.2. *Suppose that $A, M^{-1} \in \mathbb{R}^{n\times n}$ are SPD. Let Q and P be as in Definition 6.1. Let the spectra of DEF1 and BNN be given by*

$$\sigma(M^{-1}PA) = \{\lambda_1, \ldots, \lambda_n\}, \quad \sigma(P^T M^{-1}PA + QA) = \{\mu_1, \ldots, \mu_n\},$$

respectively. Then, the eigenvalues within these spectra can be ordered such that the following statements hold:

$$\begin{cases} \lambda_i = 0, \ \mu_i = 1, & \text{for } i = 1, \ldots, k; \\ \lambda_i = \mu_i, & \text{for } i = k+1, \ldots, n. \end{cases}$$

Proof. Using Lemma 6.1, we have

$$(P^T M^{-1}P + Q)AZ = Z, \quad M^{-1}PAZ = \mathbf{0}_{n,k}.$$

As a consequence, the columns of Z are the eigenvectors corresponding to the eigenvalues of BNN and DEF1 that are equal to 1 and 0, respectively. Due to [104, Thm. 2.8], it suffices to show that if

$$\sigma(P^T M^{-1}PA + QA) = \{1, \ldots, 1, \mu_{k+1}, \ldots, \mu_n\}$$

holds, then this implies

$$\sigma(M^{-1}PA) = \{0, \ldots, 0, \mu_{k+1}, \ldots, \mu_n\}.$$

The proof is as follows.

Consider the eigenvalues, $\{\mu_i\}$, and corresponding eigenvectors, $\{v_i\}$, with $i = k+1, \ldots, n$ of BNN, i.e., $(P^T M^{-1}P + Q)Av_i = \mu_i v_i$, which implies

$$P^T(P^T M^{-1}P + Q)Av_i = \mu_i P^T v_i. \tag{6.15}$$

Applying Lemma 6.1, we have

$$(P^T)^2 M^{-1}PA + P^T QA = P^T M^{-1}PAP^T.$$

Using the latter expression, Eq. (6.15) can be rewritten as

$$P^T M^{-1} P A w_i = \mu_i w_i,$$

with $w_i := P^T v_i$. Note that $P^T y = \mathbf{0}_n$ if $y \in \mathcal{R}(Z)$, due to Lemma 6.1. However, $w_i \neq \mathbf{0}_n$, since $v_i \notin \mathcal{R}(Z)$ for $i = k+1, \ldots, n$. Hence, μ_i is an eigenvalue of $P^T M^{-1} P A$ as well. Lemma 6.1 implies

$$\sigma\left(M^{-1} P A\right) = \sigma\left(P^T M^{-1} P A\right),$$

so that μ_i is also an eigenvalue of DEF1. □

Due to Theorem 6.2, both DEF1 and BNN provide almost the same spectra with the same clustering. The zero eigenvalues of DEF1 are replaced by unit eigenvalues in the case of BNN.

Remark 6.7. *If $1 \in [\mu_{k+1}, \mu_n]$, then the condition numbers of BNN and DEF1 are identical. On the other hand, if $1 \notin [\mu_{k+1}, \mu_n]$, then DEF1 has a more favorable condition number compared to BNN, see also [104]. In this latter case, if j iterations of CG achieve a suitable reduction in the error using DEF1, more than j iterations of CG might be required to optimally eliminate all errors associated with eigenvalue* 1.

Next, Theorem 6.3 relates all methods in terms of their spectra and provides a strong connection between the two classes as given in Theorem 6.1.

Theorem 6.3. *Let the spectrum of DEF1, DEF2, R-BNN1 or R-BNN2 be given by*

$$\{0, \ldots, 0, \lambda_{k+1}, \ldots, \lambda_n\},$$

satisfying $\lambda_{k+1} \leq \lambda_{k+2} \leq \ldots \leq \lambda_n$. Let the spectrum of BNN, A-DEF1 or A-DEF2 be

$$\{1, \ldots, 1, \mu_{k+1}, \ldots, \mu_n\},$$

with $\mu_{k+1} \leq \mu_{k+2} \leq \ldots \leq \mu_n$. Then, $\lambda_i = \mu_i$ for all $i = k+1, \ldots, n$.

Proof. The theorem follows immediately from Theorem 6.1 and 6.2. □

From Theorem 6.3, it can be concluded that all 2L-PCG methods have almost the same clusters of eigenvalues. Therefore, we expect that the convergence of all methods are similar, see Section 6.4.2 for some test cases. Moreover, the zeros in the spectrum of the first class (consisting of DEF1, DEF2, R-BNN1 or R-BNN2) might become nearly zero, due to round-off errors or the approximate solution of Galerkin systems in the operator. This gives an unfavorable spectrum, resulting in slow convergence of the method. This phenomenon does not appear in the case of BNN, A-DEF1 or A-DEF2. Small perturbations in those 2L-PCG methods lead to small changes in their spectra and condition numbers. Theoretically, this can be analyzed using Z consisting of eigenvectors, see [103, Sect. 3], but, in general, it is difficult to examine for general Z. This issue is further illustrated in Sections 6.4.3 and 6.4.4 using numerical experiments.

6.3.2 Equivalences between the Methods

In this subsection, we show that DEF2, A-DEF2, R-BNN1 and R-BNN2 produce identical iterates in exact arithmetic. More importantly, we prove that these 2L-PCG methods are mathematically equivalent to the more expensive BNN method for certain starting vectors. First, Lemma 6.2 shows that some steps in the BNN implementation can be reduced, see also [89] and [150, Sect. 2.5.2].

Lemma 6.2. *Let Q and P be as given in Definition 6.1. Suppose that $\mathcal{V}_{start} = Qb + P^T\bar{x}$ instead of $\mathcal{V}_{start} = \bar{x}$ is used in BNN, where $\bar{x} \in \mathbb{R}^n$ is an arbitrary vector. Then, this implies that*

- $Qr_{j+1} = \mathbf{0}_n$;
- $Pr_{j+1} = r_{j+1}$,

for all $j = -1, 0, 1, \ldots,$ in the BNN implementation of Algorithm 7.

Proof. Both statements can be proven by induction.

For the first statement, the proof is as follows. It can be verified that $Qr_0 = \mathbf{0}_n$ and $QAp_0 = \mathbf{0}_n$. By the inductive hypothesis, $Qr_j = \mathbf{0}_n$ and $QAp_j = \mathbf{0}_n$ hold. Then, for the inductive step, we obtain $Qr_{j+1} = \mathbf{0}_n$ and $QAp_{j+1} = \mathbf{0}_n$, since $Qr_{j+1} = Qr_j - \alpha_j QAp_j = \mathbf{0}_n$, and

$$\begin{array}{rcl} QAp_{j+1} & = & QAy_{j+1} + \beta_j QAp_j \\ & = & QAP^T M^{-1} Pr_{j+1} + QAQr_{j+1} \\ & = & \mathbf{0}_n, \end{array}$$

where we have used Lemma 6.1.

Next, for the second statement, $Pr_0 = r_0$ and $PAp_0 = Ap_0$ can be easily verified. Assume that $Pr_j = r_j$ and $PAp_j = Ap_j$. Then, both $Pr_{j+1} = r_{j+1}$ and $PAp_{j+1} = Ap_{j+1}$ hold, because

$$\begin{array}{rcl} Pr_{j+1} & = & Pr_j - \alpha_j PAp_j \\ & = & r_j - \alpha_j Ap_j \\ & = & r_{j+1}, \end{array}$$

and

$$\begin{array}{rcl} PAp_{j+1} & = & PAy_{j+1} + \beta_j PAp_j \\ & = & PAP^T M^{-1} Pr_{j+1} + \beta_j Ap_j \\ & = & AP^T M^{-1} r_{j+1} + \beta_j Ap_j \\ & = & AP^T M^{-1} Pr_{j+1} + \beta_j Ap_j \\ & = & A(y_{j+1} + \beta_j p_j) \\ & = & Ap_{j+1}, \end{array}$$

where we have applied the result of the first statement. □

Subsequently, we provide a more detailed comparison between BNN and DEF1 in terms of errors in the A−norm, see Lemma 6.3. In fact, it is a generalization of [104, Thm. 3.4 and 3.5], where we now apply an arbitrary starting vector, $\bar{x}$, instead of the zero starting vector.

Lemma 6.3. *Suppose that* $A \in \mathbb{R}^{n \times n}$ *is SPD. Let* Q *and* P *be as given in Definition 6.1. Let* $(x_{j+1})_{\textit{DEF1}}$ *and* $(x_{j+1})_{\textit{BNN}}$ *denote iterate* x_{j+1} *of BNN and DEF1 as provided by Algorithm 7, respectively. Then, these iterates satisfy*

$$\begin{cases} ||x - (x_{j+1})_{\textit{DEF1}}||_A \leq ||x - (x_{j+1})_{\textit{BNN}}||_A, & \textit{if } (x_0)_{\textit{DEF1}} = (x_0)_{\textit{BNN}}; \\ (x_{j+1})_{\textit{DEF1}} = (x_{j+1})_{\textit{BNN}}, & \textit{if } (x_0)_{\textit{DEF1}} = \bar{x} \textit{ and } (x_0)_{\textit{BNN}} = Qb + P^T \bar{x}. \end{cases}$$

Proof. The proof is analogous to the proofs as given in [104, Thm. 3.4 and 3.5]. ☐

From Lemma 6.3, we conclude that the errors of the iterates built by DEF1 are never larger than those of BNN in the $A-$norm. Additionally, DEF1 and BNN produce the same iterates in exact arithmetic, if $\mathcal{V}_{\text{start}} = Qb + P^T \bar{x}$ is used in BNN.

Next, Lemma 6.2 and 6.3 can now be combined to obtain the following important result.

Theorem 6.4. *Let* Q *and* P *be as given in Definition 6.1. Let* $\bar{x} \in \mathbb{R}^n$ *be an arbitrary vector. Then, the following methods produce exactly the same iterates,* $\{x_{j+1}\}$, *in exact arithmetic:*

- *BNN with* $\mathcal{V}_{\textit{start}} = Qb + P^T \bar{x}$*;*
- *DEF2, A-DEF2, R-BNN1 and R-BNN2 (with* $\mathcal{V}_{\textit{start}} = Qb + P^T \bar{x}$*);*
- *DEF1 (with* $\mathcal{V}_{\textit{start}} = \bar{x}$*) whose iterates are based on* $x_{j+1} = Qb + P^T x_{j+1}$.

Proof. The theorem follows immediately from Lemma 6.2 and 6.3. ☐

As a result of Theorem 6.4, if $\mathcal{V}_{\text{start}} = Qb + P^T \bar{x}$ is used, then BNN is mathematically equivalent to R-BNN1, R-BNN2, A-DEF2 and DEF2, since they produce identical iterates. They even produce the same iterates as DEF1, if each iterate of DEF1, x_{j+1}, is transformed into $Qb + P^T x_{j+1}$. In Section 6.4.2, we show that the methods as given in Theorem 6.4 indeed lead to almost identical results with respect to convergence behavior.

Remark 6.8.

- *Another consequence of Theorem 6.4 is that the corresponding operators for DEF2, A-DEF2, R-BNN1 and R-BNN2 are all appropriate in a certain subspace, although they are not symmetric. Hence, a CG process in combination with these operators should, in theory, work properly.*
- *The results as presented in Theorem 6.4 might not be valid anymore in the computations, if the round-off errors are too large. Therefore, although BNN, DEF2, A-DEF2, R-BNN1 and R-BNN2 give exactly the same iterates, all involved 2L-PCG methods except for BNN may lead to inaccurate solutions and may suffer from nonrobustness in numerical experiments, see also Section 6.4.5. In this case, the omitted projection and correction steps of the BNN algorithm, as suggested in Lemma 6.2, are important to maintain the robustness of the method.*

6.4 Numerical Comparison

In this section, a numerical comparison of the 2L-PCG methods is performed using 2-D bubbly flows with $m = 5$, $\epsilon = 10^3$ and $s = 0.05$ as described in Section 1.3. In order to ensure that the obtaining results are no artifacts, the experiments are also carried out using 2-D Poisson problems with a constant coefficient and 2-D porous-media flows, see Appendix G and [134, Sect. 4].

The IC(0) preconditioner is chosen as M^{-1}, but it seems that other traditional SPD preconditioners could also be used instead, leading to similar results, see [137, 173]. Moreover, $k = q + 1$ subdomain deflation vectors are taken as projection vectors (cf. Section 3.6) based on Variant 5.2 (see Section 5.3). We remark that the projection vectors are not restricted to choices that are common in DDM and deflation. Typical MG projection vectors could also be taken, see [48] and Chapter 9.

6.4.1 Setup of the Experiments

We start with a numerical experiment using standard parameters, which means that an appropriate termination criterion, exact computation of E^{-1}, and exactly computed starting vectors are used. Subsequently, numerical experiments are performed with inexact E^{-1}, severe termination tolerances, and perturbed starting vectors, respectively.

The results for each method are presented in two ways. Firstly, the results are summarized in a table, presenting the number of iterations and the standard norm of the relative errors (i.e., $\frac{||x_{\mathrm{it}}-x||_2}{||x||_2}$ with the iterated solution, x_{it}). Secondly, the results are presented graphically by showing the relative errors in the $A-$norm (i.e., $\frac{||x_j-x||_A}{||x||_A}$ with x_j denoting the $j-$th iterate) during the iteration processes. We recall that each 2L-PCG method optimizes the error in the $A-$norm, rather than in the (two-level) preconditioned $A-$norm (see Section 2.4), so that it is natural to report the errors in the $A-$norm in the experiments. Moreover, the errors are also measured in the $2-$norm, since it may be a more relevant and useful measure of the error, and it appears that there are significant differences between these two measures. Finally, for each test case, the iterative process of each method is terminated if the maximum allowed number of iterations (chosen to be equal to 250) is reached, or if the norm of the relative residual falls below a tolerance, $\delta > 0$, see (2.23). As mentioned in Section 6.2.4, this termination criterion leads to a fair comparison of the 2L-PCG methods.

Remark 6.9. *As mentioned in Section 6.2.1, the choice of parameters, Z, M^{-1} and the direct solver for E^{-1}, are the same for each 2L-PCG method. This allows us to compare these methods fairly. However, in practice, the 2L-PCG methods are derived from different fields, where typical choices associated with these fields are made for these parameters. In Chapter 9, we compare the 2L-PCG methods with their typical parameters.*

6.4.2 Experiment using Standard Parameters

In the first numerical experiment, standard parameters are used with stopping tolerance $\delta = 10^{-10}$, an exact Galerkin matrix inverse, E^{-1}, and an unperturbed starting vector, $\mathcal{V}_{\text{start}}$. The results of the experiment can be found in Table 6.4 and Figure 6.1.

	$k = 2^2$		$k = 4^2$		$k = 8^2$	
Method	# It.	$\frac{\|x_{\text{it}}-x\|_2}{\|x\|_2}$	# It.	$\frac{\|x_{\text{it}}-x\|_2}{\|x\|_2}$	# It.	$\frac{\|x_{\text{it}}-x\|_2}{\|x\|_2}$
PREC	137	4.6×10^{-7}	137	4.6×10^{-7}	137	1.8×10^{-7}
AD	161	1.1×10^{-8}	163	8.4×10^{-9}	60	1.1×10^{-8}
DEF1	149	1.5×10^{-8}	144	3.1×10^{-8}	42	1.8×10^{-8}
DEF2	149	1.5×10^{-8}	144	3.1×10^{-8}	42	1.8×10^{-8}
A-DEF1	239	3.5×10^{-7}	NC	9.0×10^{-6}	48	1.5×10^{-9}
A-DEF2	149	1.5×10^{-8}	144	3.1×10^{-8}	42	1.1×10^{-8}
BNN	149	1.5×10^{-8}	144	3.1×10^{-8}	42	1.1×10^{-8}
R-BNN1	149	1.5×10^{-8}	144	3.1×10^{-8}	42	1.1×10^{-8}
R-BNN2	149	1.5×10^{-8}	144	3.1×10^{-8}	42	1.1×10^{-8}

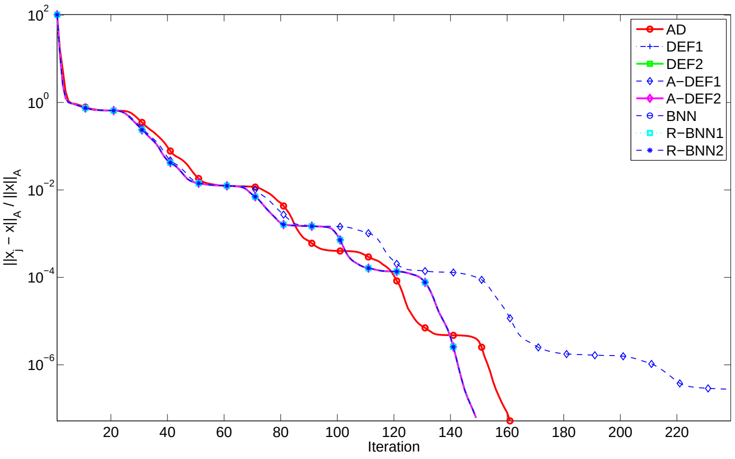
Table 6.4: Number of required iterations for convergence and the 2−norm of the relative errors of all methods, for the bubbly flow problem with $n = 64^2$, and 'standard' parameters. 'NC' means no convergence within 250 iterations.

By considering Table 6.4 and Figure 6.1, we observe that all methods perform the same, except for PREC, AD and A-DEF1. A-DEF1 has difficulties to converge, especially for the cases with $k = 2^2$ and $k = 4^2$. This is not surprising, since it cannot be shown that it is an appropriate preconditioner, see Section 6.2.4. In addition, the number of projection vectors is apparently too low to approximate the eigenvectors corresponding to the small eigenvalues, which is the result of the presence of the bubbles. Therefore, we hardly see any improvements by comparing all 2L-PCG methods to PREC in the case of $k = 2^2$ and $k = 4^2$. It is unexpected that PREC requires fewer iterations in these cases, but we observe that the corresponding solution is somewhat less accurate than the others. Moreover, we remark that AD performs obviously worse, compared to the other 2L-PCG methods.

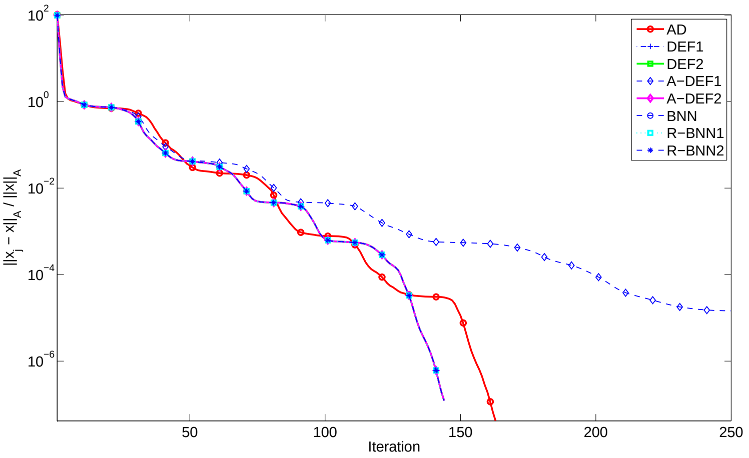
The total computational cost of the methods in this experiment is presented in Table 6.5. We restrict ourselves to the test case with $k = 8^2$, since analogous results are obtained for the other test cases. It depends on the exact implementation of the methods to determine which 2L-PCG method requires the lowest computational cost.

6.4.3 Experiment using Inaccurate Galerkin Solves

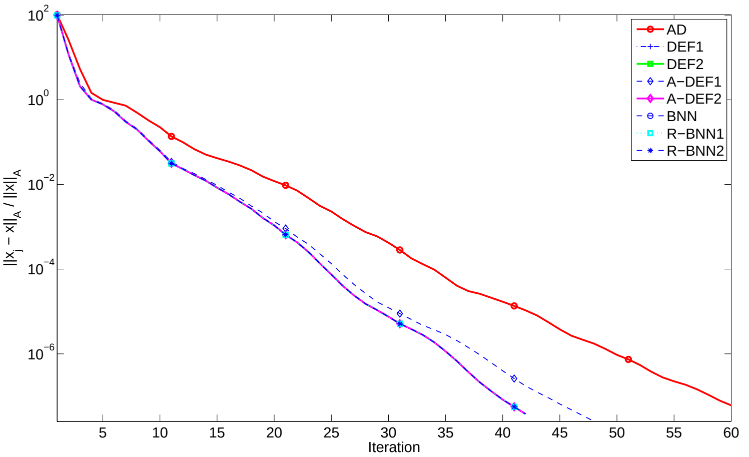
For problems with a relatively large number of projection vectors, it might be expensive to find an accurate solution of the Galerkin system, $Ey_2 = y_1$, by a direct solver at each iteration of the 2L-PCG methods. Instead, only an approximate solution, $\tilde{y}_2$, can be determined, using, for example, approximate solvers based on SSOR or ILUT preconditioners, recursive MG methods or nested iterations, such as a standard (Krylov) iterative solver with a low accuracy. In this case, $\tilde{y}_2$ can be interpreted as $\widetilde{E}^{-1}y_1$, where $\widetilde{E}$ is an inexact matrix based on E. This motivates our next experiment, using $\widetilde{E}^{-1}$



(a) $k = 2^2$.



(b) $k = 4^2$.



(c) $k = 8^2$.

Figure 6.1: Relative errors during the iterative process, for the bubbly flow problem with $n = 64^2$, and 'standard' parameters.

Method	IP	VU	GSS	PR
PREC	137	411	0	137
AD	180	180	42	42
DEF1	126	168	42	42
DEF2	126	168	42	42
A-DEF1	192	192	48	48
A-DEF2	210	168	84	42
BNN	252	210	84	42
R-BNN1	210	210	84	42
R-BNN2	126	168	42	42

Table 6.5: Computational cost within the iterations in terms of number of inner products ('IP'), vector updates ('VU'), Galerkin system solves ('GSS'), and preconditioning step with M^{-1} ('PR'), for the bubbly flow problem with $n = 64^2$, $k = 8^2$, and 'standard' parameters.

defined as

$$\widetilde{E}^{-1} := (I + \psi R) E^{-1} (I + \psi R), \quad \psi > 0, \tag{6.16}$$

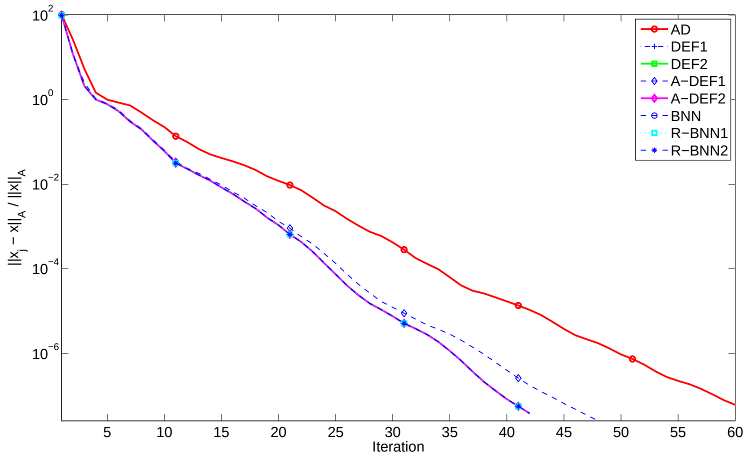
where $R \in \mathbb{R}^{k \times k}$ is a symmetric random matrix with entries from the interval $[-0.5, 0.5]$, see also [103, Sect. 3] for more details. Note that theory, as derived in Section 6.3.2, is not valid for any $\psi > 0$, but we will see that some of those theoretical results are still confirmed for relatively large ψ. The sensitivity of the 2L-PCG methods to this inaccurate solve with various values of ψ are investigated, and the results are related to Theorem 6.16. Note that the results for PREC are not influenced by this adaptation of E^{-1}. They are only included for reference.

Remark 6.10. *Eq. (6.16) does not reflect the way that inexact Galerkin solves typically enter 2L-PCG methods, but it does provide us with good insights into approximate Galerkin solves applied to these methods. Additionally, the approximation of E^{-1} can be quantified explicitly using Eq. (6.16). Experiments with Galerkin solves that are done iteratively (i.e., nested iterations) can be found in Chapter 8. In that chapter, it is shown that it is reasonable to apply (6.16), since they give similar results as in this subsection. Moreover, it turns out that the original PCG rather than a flexible variant can still be used in these experiments, as long as the inner stopping tolerance is sufficiently small. More details about inexact Krylov subspace methods can also be found in [124].*

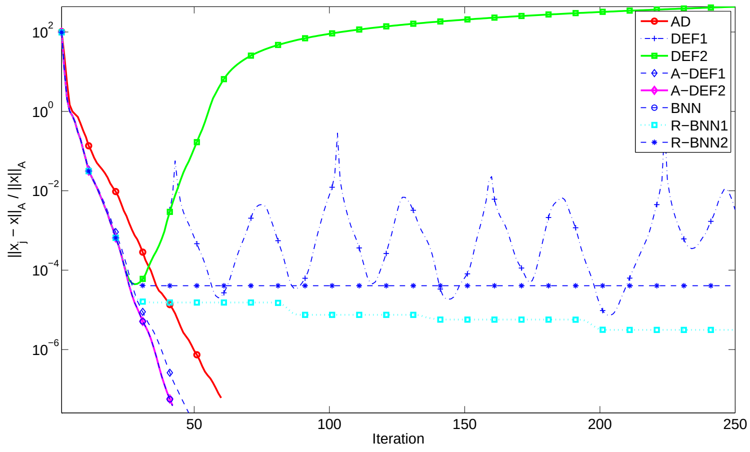
The results of the experiment can be found in Table 6.6 and Figure 6.2. We observe that the most robust 2L-PCG methods are AD, BNN, A-DEF1 and A-DEF2, since they are largely sensitive to perturbations in E^{-1}. On the other hand, DEF1, DEF2, R-BNN1 and R-BNN2 are obviously the worst methods, as expected, since the zero eigenvalues of the corresponding systems become small nearly-zero eigenvalues due to the perturbation, ψ (cf. Section 6.3.1).

6.4.4 Experiment using Severe Termination Tolerances

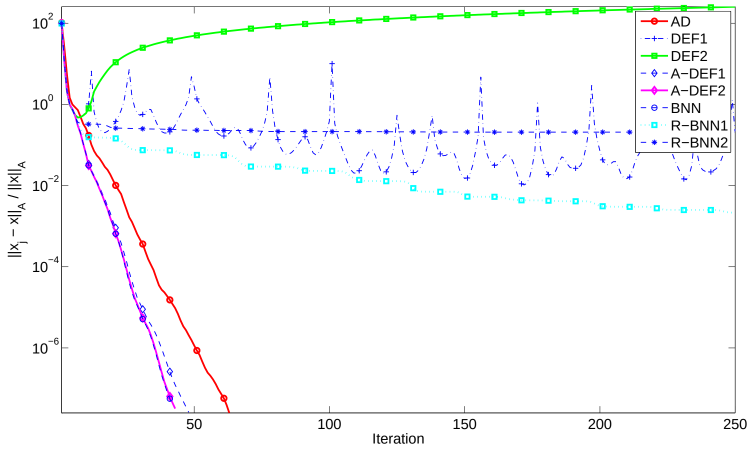
In practice, two-level PCG methods are sometimes compared with a too strict termination criterion. Such a comparison can be unfair, as certain 2L-PCG methods are



(a) $\psi = 10^{-12}$.



(b) $\psi = 10^{-8}$.



(c) $\psi = 10^{-4}$.

Figure 6.2: Relative errors during the iterative process, for the bubbly flow problem with parameters $n = 64^2$ and $k = 8^2$, and a perturbed Galerkin matrix inverse, $\widetilde{E}^{-1}$.

	$\psi = 10^{-12}$		$\psi = 10^{-8}$		$\psi = 10^{-4}$	
Method	# It.	$\frac{\|\|x_{\text{it}}-x\|\|_2}{\|\|x\|\|_2}$	# It.	$\frac{\|\|x_{\text{it}}-x\|\|_2}{\|\|x\|\|_2}$	# It.	$\frac{\|\|x_{\text{it}}-x\|\|_2}{\|\|x\|\|_2}$
PREC	137	4.6×10^{-7}	137	4.6×10^{-7}	137	4.6×10^{-7}
AD	60	2.3×10^{-8}	60	2.3×10^{-8}	63	7.8×10^{-9}
DEF1	42	1.2×10^{-8}	NC	8.3×10^{-4}	NC	9.2×10^{-2}
DEF2	42	1.2×10^{-8}	NC	$3.9 \times 10^{+2}$	NC	$2.2 \times 10^{+2}$
A-DEF1	48	8.8×10^{-9}	48	8.8×10^{-9}	48	8.5×10^{-9}
A-DEF2	42	1.1×10^{-8}	42	1.1×10^{-8}	43	8.2×10^{-9}
BNN	42	1.1×10^{-8}	42	1.1×10^{-8}	42	1.1×10^{-8}
R-BNN1	42	1.1×10^{-8}	NC	4.1×10^{-7}	NC	1.7×10^{-4}
R-BNN2	42	1.2×10^{-8}	NC	3.7×10^{-5}	NC	1.5×10^{-1}

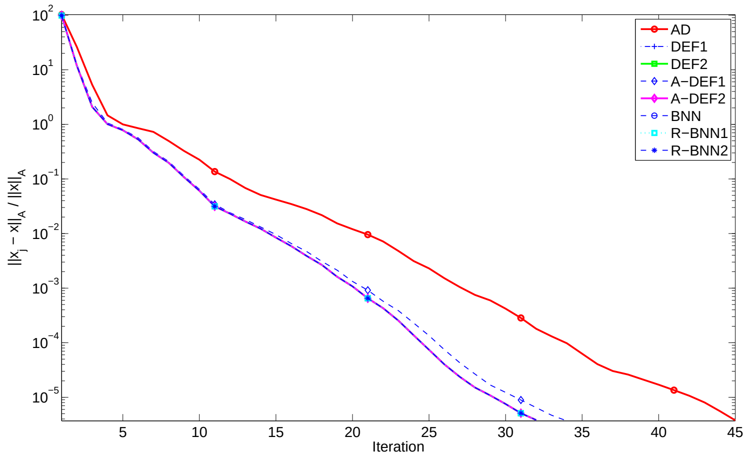
Table 6.6: Number of required iterations for convergence and the 2−norm of the relative errors of all methods for the bubbly flow problem with parameters $n = 64^2$ and $k = 8^2$, and a perturbed Galerkin matrix inverse, $\widetilde{E}^{-1}$, is used with varying perturbation ψ. 'NC' means no convergence within 250 iterations.

sensitive to severe termination criteria, see, e.g., [65]. We investigate this by performing a numerical experiment with various values of the tolerance, δ. Note that, for a relatively small δ, this may lead to a too severe termination criterion with respect to machine precision. However, the aims of this experiment are to test the sensitivity of the 2L-PCG methods to δ, and to investigate the maximum accuracy that can be reached, rather than to perform realistic experiments.

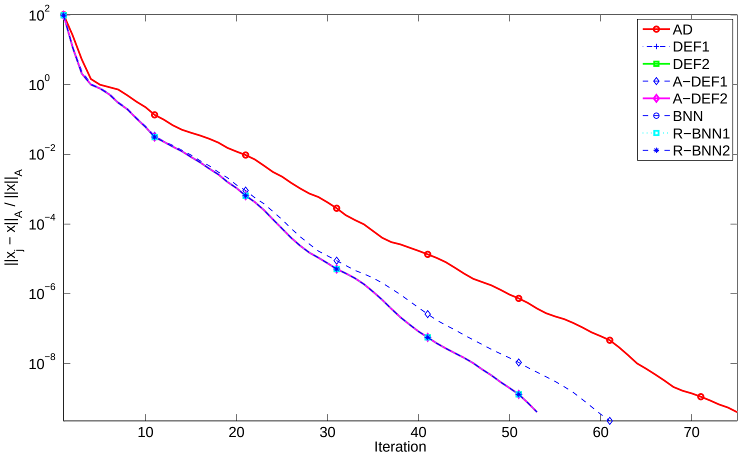
	$\delta = 10^{-8}$		$\delta = 10^{-12}$		$\delta = 10^{-16}$	
Method	# It.	$\frac{\|\|x_{\text{it}}-x\|\|_2}{\|\|x\|\|_2}$	# It.	$\frac{\|\|x_{\text{it}}-x\|\|_2}{\|\|x\|\|_2}$	# It.	$\frac{\|\|x_{\text{it}}-x\|\|_2}{\|\|x\|\|_2}$
PREC	122	6.6×10^{-6}	162	1.8×10^{-10}	179	2.1×10^{-13}
AD	45	1.4×10^{-6}	75	1.1×10^{-10}	178	4.8×10^{-13}
DEF1	32	9.4×10^{-7}	53	1.5×10^{-10}	NC	5.7×10^{-6}
DEF2	32	9.4×10^{-7}	53	1.5×10^{-10}	NC	6.3×10^{-7}
A-DEF1	34	9.7×10^{-7}	61	8.2×10^{-11}	233	2.2×10^{-13}
A-DEF2	32	9.4×10^{-7}	53	1.5×10^{-10}	133	8.3×10^{-13}
BNN	32	9.4×10^{-7}	53	1.5×10^{-10}	133	5.9×10^{-13}
R-BNN1	32	9.4×10^{-7}	53	1.5×10^{-10}	NC	2.2×10^{-12}
R-BNN2	32	9.4×10^{-7}	53	1.5×10^{-10}	NC	8.1×10^{-9}

Table 6.7: Number of required iterations for convergence and the 2−norm of the relative errors of all methods, for the bubbly flow problem with parameters $n = 64^2$ and $k = 8^2$. Various termination tolerances, δ, are tested.

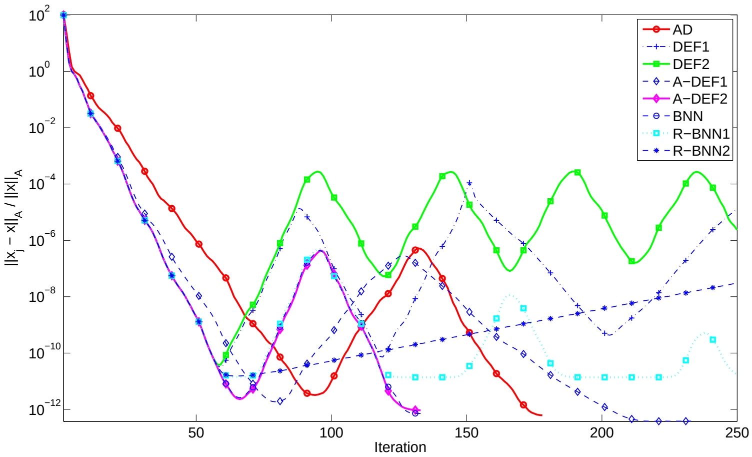
The results of the experiment are presented in Table 6.7 and Figure 6.3. It can be seen that all methods perform well, even in the case of a relatively strict termination criterion (i.e., $\delta = 10^{-12}$). PREC also converges in all cases, but not within 250 iterations. Note, moreover, that it does not give an accurate solution if δ is chosen too large. For $\delta < 10^{-12}$, DEF1, DEF2, R-BNN1 and R-BNN2 show difficulties, since they do not converge appropriately and may even diverge. This is in contrast to PREC, AD, BNN, A-DEF1 and A-DEF2, which give good convergence results for $\delta = 10^{-16}$. Therefore, these 2L-PCG methods can be characterized as robust methods



(a) $\delta = 10^{-8}$.



(b) $\delta = 10^{-12}$.



(c) $\delta = 10^{-16}$.

Figure 6.3: Relative errors during the iterative process for the bubbly flow problem with parameters $n = 64^2$, $k = 8^2$, and various termination criterion.

with respect to termination criteria.

Some of the nonconverging methods might eventually give the solution after the maximum number of iterations, but such a solution takes too much computing time; hence, the result is useless. Moreover, the nonconverging behavior is caused by round-off errors, resulting in, for example, a lack of orthogonality of the residuals with respect to Z, see Section 6.4.6.

6.4.5 Experiment using Perturbed Starting Vectors

In Section 6.3.2, it is proven that BNN with $\mathcal{V}_{\text{start}} = Qb + P^T\bar{x}$ gives exactly the same iterates as DEF2, A-DEF2, R-BNN1 and R-BNN2, in exact arithmetic. In this case, the resulting operators are well-defined and they should perform appropriately. In our next experiment, we perturb $\mathcal{V}_{\text{start}}$ in DEF2, A-DEF2, R-BNN1 and R-BNN2, and examine whether this influences the convergence results. The motivation of this experiment is the same as for the experiment carried out in Section 6.4.3; for relatively large problems, it can be complicated to determine $\mathcal{V}_{\text{start}}$ accurately, due to, for example, the inaccurate computation of Galerkin solves. It is important to note that if we use approximate starting vectors, then there is no longer any equivalence between BNN and its reduced methods, as provided in the results of Section 6.3.2. In this case, it is interesting to see how these methods perform in practice.

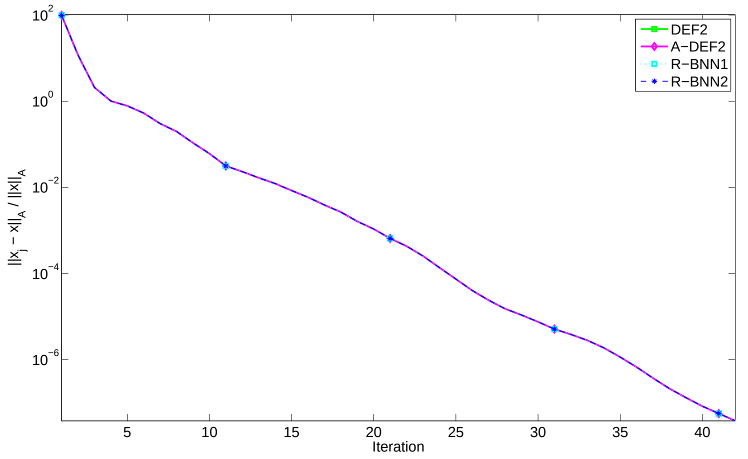
The perturbed $\mathcal{V}_{\text{start}}$, denoted by $\mathcal{W}_{\text{start}}$, is defined as a componentwise multiplication of a random vector and $\mathcal{V}_{\text{start}}$, i.e., each entry of $\mathcal{W}_{\text{start}}$ is defined as

$$(\mathcal{W}_{\text{start}})_i := (1 + \gamma (v_0)_i)\,(\mathcal{V}_{\text{start}})_i, \quad i = 1, 2, \ldots, n, \tag{6.17}$$

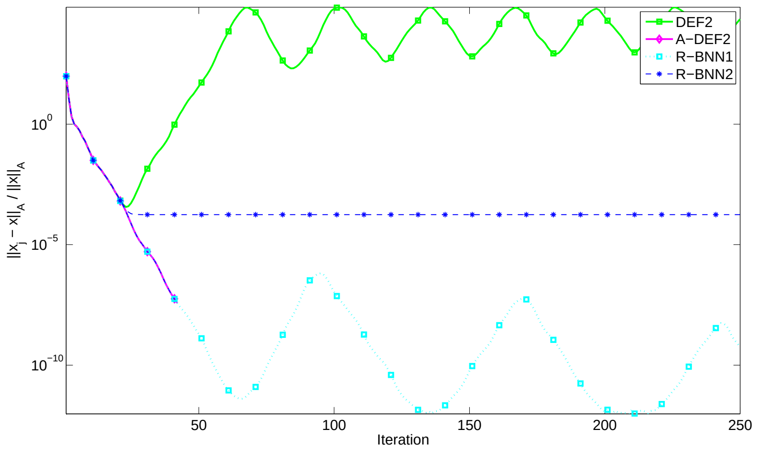
where $\gamma \geq 0$ gives control over the accuracy of the starting vector, and v_0 is a random vector with entries from the interval $[-0.5, 0.5]$, taken to give each entry of $\mathcal{V}_{\text{start}}$ a different perturbation. As in the experiment performed in Section 6.4.3, the choice of $\mathcal{W}_{\text{start}}$ does not reflect the way in which starting vectors are perturbed in practice, but it provides us with some valuable insights where the perturbation can be quantified in an easy way. Furthermore, note that if DEF2, R-BNN1 or R-BNN2 converge using $\mathcal{W}_{\text{start}}$, then we may obtain a nonunique solution, since the corresponding operator is singular. Therefore, as in the case of DEF1, we should apply the 'uniqueness' step (see Remark 6.3) at the end of the iteration process. Note that this procedure is not required for A-DEF2, because this method corresponds to a nonsingular operator.

We perform the numerical experiment using $\mathcal{W}_{\text{start}}$ for different γ. The results can be found in Table 6.8 and Figure 6.4. Here, we use asterisks to stress that an extra uniqueness step is applied in the specific method. Moreover, notice that PREC, AD, DEF1 and BNN are not included in this experiment, since they apply an arbitrary vector, $\mathcal{V}_{\text{start}} = \bar{x}$, by definition.

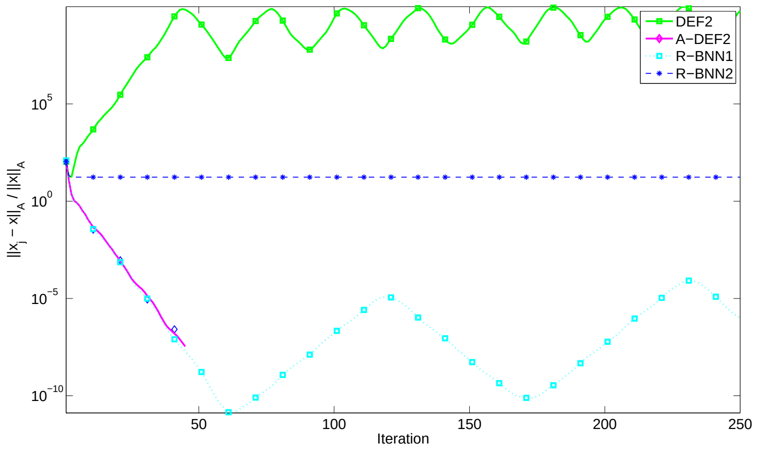
From the results, it can be noticed that all involved methods converge appropriately for $\gamma = 10^{-10}$. For $\gamma \geq 10^{-5}$, DEF2, R-BNN1 and R-BNN2 fail to converge (with respect to the residuals), although R-BNN1 is already converged and the current stopping criterion is apparently unreliable for this method in this experiment. The most



(a) $\gamma = 0$.



(b) $\gamma = 10^{-5}$.



(c) $\gamma = 10^{-10}$.

Figure 6.4: Relative errors during the iterative process for the bubbly flow problem with $n = 64^2$, $k = 8^2$, and perturbed starting vectors.

	$\gamma = 10^{0}$		$\gamma = 10^{-5}$		$\gamma = 10^{-10}$	
Method	# It.	$\frac{\lVert x_{\mathrm{it}}-x\rVert_2}{\lVert x\rVert_2}$	# It.	$\frac{\lVert x_{\mathrm{it}}-x\rVert_2}{\lVert x\rVert_2}$	# It.	$\frac{\lVert x_{\mathrm{it}}-x\rVert_2}{\lVert x\rVert_2}$
DEF2	42	1.1×10^{-8}	NC	$1.4 \times 10^{+4}$	NC	$2.7 \times 10^{+9}$
A-DEF2	42	1.1×10^{-8}	42	1.2×10^{-8}	45	1.2×10^{-8}
R-BNN1	42	1.1×10^{-8}	NC	$1.4 \times 10^{-10}*$	NC	$3.6 \times 10^{-7}*$
R-BNN2	42	1.1×10^{-8}	NC	$1.8 \times 10^{-5}*$	NC	$1.1 \times 10^{+0}*$

Table 6.8: Number of required iterations for convergence and the 2−norm of the relative errors of some methods, for the bubbly flow problem with $n = 64^2$, $k = 8^2$, and perturbed starting vectors. An asterisk (*) means that an extra uniqueness step is applied in that test case.

robust method is, obviously, A-DEF2. This method is completely insensitive to the perturbation, γ. This experiment illustrates that the 'reduced' variants of BNN have different robustness properties with respect to perturbations in starting vectors.

6.4.6 Further Discussion

The theoretical results given in Section 6.3 only hold in exact arithmetic and under the assumptions required to prove them. However, from a numerical point of view, we have observed that some of these assumptions are necessary, whereas others are only sufficient for certain two-level PCG methods. The numerical results confirm the theoretical fact that all 2L-PCG methods perform approximately the same, although A-DEF1 shows problems in some test cases. This is understood by the fact that A-DEF1 corresponds to a non-SPSD operator, as also discussed in Section 6.2.4.

If the dimension of the Galerkin matrix, E, becomes large, it is favorable to solve the corresponding systems iteratively, with a low accuracy. In this case, we see that DEF1, DEF2, R-BNN1 and R-BNN2 show difficulties in convergence. It can be observed that the errors during the iterative process of DEF2 explode, whereas DEF1 converges slowly to the solution, but in an erratic way. The most robust methods are AD, BNN, A-DEF1 and A-DEF2.

If A is ill-conditioned and the tolerance of the termination criterion, chosen by the user, becomes too severe, it is advantageous that the 2L-PCG method would still work appropriately. However, we observe that DEF1, DEF2, A-DEF1, R-BNN1 and R-BNN2 cannot deal with too strict tolerances. This is in contrast to AD, BNN, A-DEF2, which remain robust in all test cases.

In theory, BNN gives the same iterates as DEF2, A-DEF2, R-BNN1 and R-BNN2, for certain starting vectors. In addition to the fact that these 'reduced' variants, except A-DEF2, are not able to deal with inaccurate Galerkin solves, some of them are also sensitive to perturbations of the starting vector. In contrast to the other methods, A-DEF2 is independent of these perturbations. This can be of great importance, if one uses multigrid-like subdomains, where the number of subdomains, k, is very large, and the starting vector cannot be obtained accurately.

In the numerical experiments, we observe that several methods show divergence, stagnation or erratic behavior of the errors during the iterative process. This may be caused by the fact that the residuals gradually lose orthogonality with respect to the columns of Z, see also [122]. It can easily be shown that

$$Z^T r_j = \mathbf{0}_k, \quad j = 0, 1, \ldots, \tag{6.18}$$

should hold for DEF1, DEF2, A-DEF2, R-BNN1 and R-BNN2. However, it appears that (6.18) is not always satisfied in the experiments. A remedy to recover this orthogonality in the badly converging methods is described in, e.g., [122]. If we define the 'reorthogonalization' matrix, $W \in \mathbb{R}^{n \times n}$, as

$$W := I - Z(Z^T Z)^{-1} Z^T, \tag{6.19}$$

then W is orthogonal to Z, i.e.,

$$Z^T W = Z^T - Z^T Z(Z^T Z)^{-1} Z^T = \mathbf{0}_{k,n}. \tag{6.20}$$

Now, orthogonality of the residuals, $\{r_j\}$, can be preserved by premultiplying r_j by W right after r_j is computed in the algorithm:

$$r_j := W r_j, \quad j = 0, 1, \ldots. \tag{6.21}$$

As a consequence, these adapted residuals satisfy (6.18), due to (6.20).

Remark 6.11.

- *Eq. (6.18) is not valid for AD, A-DEF1 and BNN. In the case of AD and BNN, this is not a problem, because they appear to be extremely robust in most test cases. This is in contrast to A-DEF1, which is not robust in several test cases, since it is not an appropriate preconditioner, see Section 6.2.4. The nonrobustness of this projector cannot be resolved using the reorthogonalization strategy.*
- *The reorthogonalization operator (6.21) is relatively cheap, provided that Z is sparse.*

In the numerical experiments of [134, Sect. 4.6], we show that the adapted versions of the methods, including the reorthogonalization strategy, converge better in terms of the residuals. Unfortunately, it appears that accurate solutions could not be obtained using this approach. To preserve the relation, $r_j = b - Ax_j$, each iterate, x_j, should be adapted via

$$x_j := x_j - A^{-1} Z(Z^T Z)^{-1} Z^T r_j, \quad j = 0, 1, \ldots. \tag{6.22}$$

However, it is clear that (6.22) is not useful to apply due to the presence of A^{-1} in that expression. Consequently, it is unlikely that, in practice, the 2L-PCG methods would benefit from the reorthogonalization strategy.

6.5 Concluding Remarks

In this chapter, we consider the abstract forms of several two-level PCG methods, listed in Table 6.1, which originated from the fields of deflation, domain decomposition and multigrid. A comparison of these methods is carried out by investigating their theoretical and numerical aspects.

Theoretically, DEF1 is the best method [103–105]. We see that all two-level PCG methods, except for PREC and AD, have comparable eigenvalue distributions. Two classes of two-level PCG methods can be distinguished, each having the same spectral properties. The first class consists of DEF1, DEF2, R-BNN1 and R-BNN2, and the second class includes BNN, A-DEF1 and A-DEF2. Although the differences are surprisingly marginal, and, therefore, similar convergence behaviors are expected, we derive that the associated spectrum of the methods of the first class is possibly more favorable than those of the second class.

In numerical experiments with realistic termination criteria and relatively small perturbations in the starting vector and Galerkin solves, it is observed that all 2L-PCG methods always converge faster than PREC. More importantly, all 2L-PCG methods show approximately the same convergence behavior, although the residuals of AD has sometimes a nonmonotonical convergence behavior. Both DEF1 and DEF2 are sensitive to sufficiently large perturbations in the Galerkin solves or too strict termination criterion. In contrast to DEF1, DEF2 also has the drawbacks that it cannot deal with perturbed starting vectors and that the method diverges when the convergence deteriorates. The errors are usually bounded in DEF1, when this method does not converge.

We deduce that, for certain starting vectors, the expensive operator of BNN can be reduced to simpler and cheaper operators, which are used in DEF2, A-DEF2, R-BNN1 and R-BNN2. Hence, some 2L-PCG methods of the two spectral classes are mathematically equivalent in exact arithmetic. However, these reduced variants, except for A-DEF2, are not robust in the numerical experiments, when applying inaccurate Galerkin solves, strict stopping tolerances or perturbed starting vectors. In fact, one should realize that the reduced variants of BNN, except A-DEF2, are as not robust as DEF1 or DEF2.

By examining all theoretical and numerical aspects, we conclude that BNN and A-DEF2 are the best 2L-PCG methods in the sense of robustness. However, two deflation matrices are involved in BNN, making the method expensive to use. On the contrary, only one deflation matrix is involved in A-DEF2, so that it is attractive to apply. Hence, A-DEF2 seems to be the best and most robust method, considering the theory, numerical experiments, and the computational cost.

If robustness is not an crucial issue in experiments, then the deflation method (DEF1 or DEF2) is a very efficient method (and often faster than other methods), see also Chapter 8 where DEF1 and A-DEF2 are compared in more detail. Finally, the two-level PCG method based on the multigrid V(1,1)-cycle preconditioner is excluded in the comparison presented in this chapter, but it is related to the other methods in the next chapter.

Chapter 7

Comparison of Two-Level PCG Methods – Part II

7.1 Introduction

In the previous chapter, we have compared several two-level PCG methods originated from different fields. In that comparison, we have not included the two-level PCG method with a preconditioner based on a multigrid V(1,1)-cycle (denoted by the MG method in this chapter), since it has very different spectral properties and requires a specific theoretical treatment, because of the more general choice for the traditional preconditioner allowed within MG. The aim of this chapter is to fill this gap. We focus on the comparison between abstract balancing Neumann-Neumann (BNN), deflation (DEF), and multigrid V(1,1)-cycle (MG) preconditioners. DEF is equal to the DEF1 method from Chapter 6.

Of course, the MG method [23, 69, 151, 178] and its properties [20, 53, 68, 96, 107] are well-known. Our intention is not to reproduce these results (although some known results needed for the comparison are briefly reviewed), but to compare and connect MG to the other 2L-PCG methods. Intuitively, we expect MG to have better convergence properties than the other 2L-PCG methods, when the MG smoother (also known as the MG relaxation) is chosen to be equal to M^{-1}, since it is the only 2L-PCG method with two applications of the traditional preconditioners (in the pre- and post-smoothing steps), in addition to a single coarse-grid correction step within one iteration. DEF, on the other hand, has optimal convergence properties in terms of its spectral properties compared with the other 2L-PCG methods (except MG), see the previous chapter. Therefore, it is sufficient for the comparison to show that MG has more favorable spectral properties than DEF, if MG is indeed superior to DEF. Hence, we often base the analysis on the comparison of DEF and MG in this chapter. However, the comparison between MG and BNN is, in some cases, easier to perform, so BNN is used in the analysis as well.

Some spectral analysis for MG is carried out in [26]. In that paper, projection vectors are based on exact eigenvectors of $M^{-1}A$ and more pre- and post-smoothing steps are allowed per iteration. The resulting two-level preconditioner is called a 'multiplicative two-grid spectral preconditioner'. It is shown that this preconditioner can be effective for many practical applications, where sequences of linear systems have to be solved. In this chapter, we restrict ourselves to the standard multigrid V(1,1)-cycle preconditioner, while eigenvectors are sometimes used to illustrate the theoretical results. Moreover, we note that while the condition number of preconditioned systems is an imperfect indicator of the convergence properties of CG, it is the only analysis tool available with sufficient generality to compare the techniques considered here.

This chapter is organized as follows. In Section 7.2, DEF, BNN and MG are described concisely. Then, some spectral properties of MG are presented in Section 7.3. Thereafter, in Section 7.4, MG and DEF are compared by investigating their spectral properties using special choices of parameters; it is shown there that MG can be less effective than DEF. In Section 7.5, we show that MG is superior to DEF for more sophisticated preconditioners. Subsequently, Section 7.6 is devoted to the comparison of MG, BNN and DEF with the same cost per iteration. For special choices of preconditioners, we show that they are almost spectrally equivalent. Section 7.7 is devoted to some numerical experiments in order to illustrate the theoretical results. Some concluding remarks are presented in Section 7.8.

7.2 Two-Level PCG Methods

In this section, the 2L-PCG methods are described that will be examined to solve the linear system, $Ax = b$, where A is assumed to be SPD. We remark again that most results presented in this chapter are generalizable to linear systems where A is SPSD. The following definition (cf. Definition 6.1) is assumed to hold throughout this chapter.

Definition 7.1. *Suppose that an SPD coefficient matrix, $A \in \mathbb{R}^{n\times n}$, and a deflation-subspace matrix, $Z \in \mathbb{R}^{n\times k}$, with full rank and $k < n$ are given. Then, we define the invertible Galerkin matrix, $E \in \mathbb{R}^{k\times k}$, the correction matrix, $Q \in \mathbb{R}^{n\times n}$, and the deflation matrix, $P \in \mathbb{R}^{n\times n}$, as follows:*

$$P := I - AQ, \quad Q := ZE^{-1}Z^T, \quad E := Z^TAZ.$$

In addition, $\bar{M}^{-1} \in \mathbb{R}^{n\times n}$ is an arbitrary preconditioning matrix and $M^{-1} \in \mathbb{R}^{n\times n}$ is an SPD preconditioning matrix.

Remark 7.1. *The difference between M^{-1} and $\bar{M}^{-1}$ is that M^{-1} is assumed to be symmetric, positive definite and nonsingular, whereas $\bar{M}^{-1}$ might be nonsymmetric, singular, or even indefinite, so that it is basically the pseudo-inverse of $\bar{M}$. Preconditioner M^{-1} is applied in deflation-like methods, whereas the more general preconditioner, $\bar{M}^{-1}$, is applied solely in multigrid methods, where a general smoothing operator is allowable.*

The deflation method (DEF) is already described in the previous chapters. Recall that its two-level preconditioner is

$$\mathcal{P}_{\text{DEF}} = M^{-1}P. \tag{7.1}$$

In order to derive the BNN and MG preconditioners, we consider again the multiplicative combination of preconditioners, see Section 6.2.2. Recall that the multiplicative operator consisting of three preconditioners is given by (see Eq. (6.5))

$$\mathcal{P}_{m_3} = C_1 + C_2 + C_3 - C_2AC_1 - C_3AC_2 - C_3AC_1 + C_3AC_2AC_1. \tag{7.2}$$

It has already been derived that if one substitutes $C_1 := Q$, $C_2 := M^{-1}$ and $C_3 := Q$ into (7.2), we obtain

$$\mathcal{P}_{\text{BNN}} = P^TM^{-1}P + Q, \tag{7.3}$$

which is the two-level preconditioner corresponding to the abstract balancing Neumann-Neumann (BNN) method. We have shown that BNN has the same spectral properties as the 2L-PCG methods based on multigrid V(0,1)- and V(1,0)-cycle preconditioners (see Theorem 6.1).

On the other hand, we could also use $\bar{M}^{-1}$ twice instead of Q, i.e., $C_1 := \bar{M}^{-T}, C_2 := Q$ and $C_3 := \bar{M}^{-1}$ in (7.2). We use the general preconditioner, $\bar{M}^{-1}$, instead of M^{-1}, because $\bar{M}^{-1}$ is not required to be symmetric nor invertible to define $\mathcal{P}_{m_3}$. The resulting two-level preconditioner, well-known as the multigrid V(1,1)-cycle preconditioner, is then explicitly given by (see Eq. (6.12))

$$\mathcal{P}_{\text{MG}} = \bar{M}^{-T}P + P^T\bar{M}^{-1} + Q - \bar{M}^{-T}PA\bar{M}^{-1}. \tag{7.4}$$

The latter expression for $\mathcal{P}_{\text{MG}}$ also follows from the error-propagation operator:

$$V := (I - \mathcal{P}_{\text{MG}}A) = (I - \bar{M}^{-T}A)P^T(I - \bar{M}^{-1}A), \tag{7.5}$$

which is often written as

$$V := S^*P^TS, \quad S := I - \bar{M}^{-1}A, \tag{7.6}$$

where $S^* := I - \bar{M}^{-T}A$ denotes the adjoint of S with respect to the A-inner product. Recall that matrices S and S^* are known as the pre- and post-smoothers, respectively, and P^T is the coarse-grid correction operation. The resulting two-level PCG method with $\mathcal{P}_{\text{MG}}$ is called MG, see [23, 69, 151, 178] for more details.

Note that $\mathcal{P}_{\text{MG}}$ is obviously symmetric, but it is not necessarily positive semi-definite, see Section 7.3.2. Next, it can be observed that the two-level preconditioner corresponding to DEF is included as a term in the two-level preconditioner of MG if $\bar{M}^{-1} = M^{-1}$ is taken (cf. Eqs. (7.1) and (7.4)). Hence, we might expect that MG is always more effective than DEF. For common choices of M^{-1}, $\bar{M}^{-1}$ and Z, this is indeed the case, see Section 7.7.2. However, Section 7.4 shows that this is not true in all cases.

To summarize, the abbreviations and the two-level preconditioners corresponding to the proposed 2L-PCG methods are presented in Table 7.1.

Name	Method	Two-level preconditioner, $\mathcal{P}$
PREC	Traditional PCG	M^{-1}
DEF	Deflation	$M^{-1}P$
BNN	Abstract Balancing	$P^T M^{-1} P + Q$
MG	Multigrid V(1,1)-cycle	$\bar{M}^{-1}P + P^T \bar{M}^{-1} + Q - \bar{M}^{-1}PA\bar{M}^{-1}$

Table 7.1: List of two-level PCG methods that are compared in this chapter.

Remark 7.2.

- *Eq. (7.4) is only used for the analysis of MG, but is never implemented using this explicit form as the action of $\mathcal{P}_{MG}$ can be computed with only a single multiplication, each involving $\bar{M}^{-1}$, $\bar{M}^{-T}$, and Q.*
- *We emphasize that the parameters of the two-level PCG methods that will be compared can be arbitrary, so that the comparison between these methods is based on their abstract versions. This means that the results of the comparison are valid for any full-rank matrix Z, SPD matrices A, M^{-1}, and matrix $\bar{M}^{-1}$.*

7.3 Spectral Properties of MG

In this section, we present some results related to the spectral properties of the MG method. We first prove a result analogous to [104, Thm. 2.5], demonstrating that the MG preconditioner also clusters a number of eigenvalues at 1. Thereafter, we discuss necessary and sufficient conditions for the MG preconditioner to be SPD. Note that while these are natural concerns from a preconditioning point of view, these questions are not commonly considered for MG methods, which are often applied as stationary iterations and not used as preconditioners in all cases, unlike DEF.

First, we present some notation in Definition 7.2.

Definition 7.2. *Let A and B be an SPD and arbitrary matrix, respectively. Define $||B||_A := ||A^{\frac{1}{2}}BA^{-\frac{1}{2}}||_2$. Then,*

- *if B is SPD, then the SPD square root of B is denoted by $B^{\frac{1}{2}}$;*
- *if $||B||_A < 1$, B is called convergent in the A-norm (or A-norm convergent).*

7.3.1 Unit Eigenvalues of the MG-Preconditioned Matrix

In Chapters 3 and 6, we have seen that, for a SPD matrix A, DEF corresponds to a two-level preconditioned coefficient matrix that has exactly k zero eigenvalues, whereas the matrix associated with BNN has at least k unit eigenvalues. Theorem 7.1 shows that the two-level preconditioned matrix corresponding to MG also has at least k unit eigenvalues.

Theorem 7.1. *Let $\mathcal{P}_{MG}$ and S be as given in (7.4) and (7.6), respectively. Suppose that*

$$\dim \mathcal{N}(S) = \bar{m}, \qquad \bar{m} \in \mathbb{N}. \tag{7.7}$$

Then, $\mathcal{P}_{MG}A$ has one as an eigenvalue, with geometric multiplicity at least k and at most $k + 2\bar{m}$.

Proof. In the following, we use the factorization of $I - \mathcal{P}_{\mathrm{MG}}A = S^{*}P^{T}S$ as given in Eqs. (7.5) and (7.6). Note first that $\dim \mathcal{N}(S^{*}) = \dim \mathcal{N}(S) = \bar{m}$, see also Lemma A.14.

Considering Eq. (7.6), there are three ways for a vector, $v \neq \mathbf{0}_n$, to be in $\mathcal{N}(I - \mathcal{P}_{\mathrm{MG}}A)$:

(i) $v \in \mathcal{N}(S)$, so that $Sv = \mathbf{0}_n$;

(ii) $Sv \in \mathcal{N}(P^T)$, yielding $P^T S v = \mathbf{0}_n$;

(iii) $P^T S v \in \mathcal{N}(S^{*})$, so that $S^{*}P^{T}Sv = \mathbf{0}_n$.

We treat each case separately.

(i) The geometric multiplicity of the zero eigenvalue of $I - \mathcal{P}_{\mathrm{MG}}A$ must be at least $\bar{m}$, due to Eq. (7.7). This accounts exactly for all contributions to $\mathcal{N}(I - \mathcal{P}_{\mathrm{MG}}A)$ from null space vectors of the first type.

(ii) Counting the geometric multiplicity of vectors of the second type is only slightly more complicated. The fundamental theorem of linear algebra (see Theorem A.3) gives an orthogonal decomposition of $\mathbb{R}^n$ as

$$\mathbb{R}^n = \mathcal{R}(S) \oplus \mathcal{N}\left(S^T\right). \tag{7.8}$$

Since $\dim \mathcal{R}(S) = n - \bar{m}$, it must be the case that

$$\dim \mathcal{N}\left(S^T\right) = \bar{m}. \tag{7.9}$$

Now, consider the intersection of $\mathcal{R}(Z)$ with subspaces $\mathcal{R}(S)$ and $\mathcal{N}\left(S^T\right)$:

$$\mathcal{Z}_1 := \mathcal{R}(Z) \cap \mathcal{R}(S), \quad \mathcal{Z}_2 := \mathcal{R}(Z) \cap \mathcal{N}\left(S^T\right),$$

and let $\dim \mathcal{Z}_1 = k_1$ and $\dim \mathcal{Z}_2 = k_2$. Note that necessarily $k_1 + k_2 = k$, and that k_2 is no bigger than $\bar{m}$, because of (7.9). Since $\mathcal{N}(P^T) = \mathcal{R}(Z)$, we have $\dim \mathcal{N}(S) = k_1$, which is the contribution to the dimension of the null space by vectors of the second type. Since $k_1 + k_2 = k$ for $k_2 \leq \bar{m}$, the total dimension of the null space arising from vectors of the first and second type must satisfy $k \leq k_1 + \bar{m} \leq k + \bar{m}$.

(iii) Similarly, we can determine the dimension of the null space of the third type. Note first that (cf. Eq. (7.8))

$$\mathbb{R}^n = \mathcal{R}\left(P^T S\right) \oplus \mathcal{N}\left(S^T P\right).$$

Let $\mathcal{M} := \mathcal{N}(S^*)$ and $\mathcal{M}_1 = \mathcal{M} \cap \mathcal{R}\left(P^T S\right)$. Then, the number of unit eigenvalues of the third type is

$$\bar{m}_1 = \dim \mathcal{M}_1 \leq \dim \mathcal{M} = \bar{m}.$$

Thus, $\dim \mathcal{N}(\mathcal{P}_{\rm MG} A) = \bar{m} + k_1 + \bar{m}_1$, which can be bounded by

$$k \leq \bar{m} + k_1 + m_1 \leq k + 2\bar{m}.$$

Since counting the geometric multiplicity of zero eigenvalues of $I - \mathcal{P}_{\rm MG} A$ is trivially equal to the geometric multiplicity of unit eigenvalues of $\mathcal{P}_{\rm MG} A$ (see Lemma A.1(b)), the proof is complete. □

Remark 7.3.

- *$\mathcal{P}_{MG} A$ has at least k unit eigenvalues, even if S is singular.*

- *If zero is not an eigenvalue of S, then it is also not an eigenvalue of S^* (which is similar to S^T). Thus, Theorem 7.1 then says that $\mathcal{P}_{MG} A$ has exactly k unit eigenvalues.*

- *Since $\bar{M}^{-1}$ is nonsymmetric, the geometric and algebraic multiplicity of the zero eigenvalue of S (or, equivalently, the unit eigenvalues of $\bar{M}^{-1} A$) should be distinguished, since they might differ.* [1]

- *In a similar manner as Theorem 7.1, it can be shown that $\mathcal{P}_{BNN} A$ has at least k and at most $2k + \bar{m}$ unit eigenvalues.*

7.3.2 Positive Definiteness of the MG preconditioner

Recall that a 2L-PCG method is guaranteed to converge if $\mathcal{P}$, as given in (6.1), is SPD or can be transformed into an SPD matrix. This is certainly satisfied for BNN and DEF, see the previous chapter. Here, we examine this issue for MG. It is obvious that $\mathcal{P}_{\rm MG}$ (and, therefore, also $\mathcal{P}_{\rm MG} A$) is not positive definite for all choices of Z and $\bar{M}^{-1}$, as in the next example.

Example 7.1. *Suppose that $\bar{M}^{-1} = I$ and $Z = [v_1 \cdots v_k]$, where $\{v_i\}$ is the set of orthonormal eigenvectors corresponding to the eigenvalues of A, $\{\lambda_i\}$. Then,*

$$\mathcal{P}_{MG} = P + P^T + Q - PA = 2I - 2ZZ^T + Z\Lambda^{-1}Z^T - A + ZZ^TA, \tag{7.10}$$

where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_k)$. Multiplying (7.10) by v_i gives us

$$\mathcal{P}_{MG} v_i = 2v_i - 2ZZ^T v_i + Z\Lambda^{-1}Z^T v_i - \lambda_i v_i + \lambda_i ZZ^T v_i.$$

---

[1] A simple example is Gauss-Seidel for the 1-D Poisson problem with homogeneous Dirichlet boundary conditions. Take $A = \mathrm{tridiag}(-1, 2, -1)$ and M to be the lower-triangular part of A. Then, S has eigenvalue $\frac{1}{2}$ with algebraic multiplicity $\frac{n}{2}$, assuming that n is even. Since there is only one eigenvector corresponding to this eigenvalue, the geometric multiplicity is 1.

This implies

$$\mathcal{P}_{MG} v_i = \begin{cases} \frac{1}{\lambda_i} v_i, & \text{for } i = 1, \dots, k; \\ (2 - \lambda_i) v_i, & \text{for } i = k+1, \dots, n. \end{cases} \tag{7.11}$$

Hence, the spectrum of $\mathcal{P}_{MG}$ *is given by*

$$\left\{ \frac{1}{\lambda_1}, \dots, \frac{1}{\lambda_k}, 2 - \lambda_{k+1}, \dots, 2 - \lambda_n \right\}.$$

In this case, $\mathcal{P}_{MG}$*, is SPD if and only if* $\lambda_n < 2$.

Example 7.1 shows that $\mathcal{P}_{\text{MG}}$ can be indefinite for some choices of Z and $\bar{M}^{-1}$. This highlights an important difference between MG and DEF. Indeed, many preconditioners, M^{-1}, that make sense with DEF lead to indefinite $\mathcal{P}_{\text{MG}}$, while choices of $\bar{M}^{-1}$ that lead to $\mathcal{P}_{\text{MG}}$, which is SPD, might give nonsymmetric operators for $\mathcal{P}_{\text{DEF}}$. Next, a necessary and sufficient condition for $\mathcal{P}_{\text{MG}}$ to be SPD is given in Theorem 7.2.

Theorem 7.2. *Let* $\bar{M}^{-1}$ *and* Z *be as defined in Definition 7.1. Let* $\mathcal{P}_{MG}$ *be as given in (7.4). A necessary and sufficient condition for* $\mathcal{P}_{MG}$ *to be SPD is that* Z *and* $\bar{M}^{-1}$ *satisfy*

$$\min_{w:\ w \perp AZy\ \forall y} w^T \left(\bar{M}^{-1} + \bar{M}^{-T} - \bar{M}^{-1} A \bar{M}^{-T} \right) w > 0. \tag{7.12}$$

Proof. By definition, $\mathcal{P}_{\text{MG}}$ is positive definite if and only if $u^T \mathcal{P}_{\text{MG}} u > 0$ for all vectors $u \neq \mathbf{0}_n$. Taking $u := A^{\frac{1}{2}} y$, this means that $\mathcal{P}_{\text{MG}}$ is SPD if and only if $y^T A^{\frac{1}{2}} \mathcal{P}_{\text{MG}} A^{\frac{1}{2}} y > 0$, for all y, or that $A^{\frac{1}{2}} \mathcal{P}_{\text{MG}} A^{\frac{1}{2}}$ is positive definite. Moreover, $A^{\frac{1}{2}} \mathcal{P}_{\text{MG}} A^{\frac{1}{2}}$ is symmetric and, so, it is SPD if and only if its smallest eigenvalue is greater than 0. This, in turn, is equivalent to requiring that $I - A^{\frac{1}{2}} \mathcal{P}_{\text{MG}} A^{\frac{1}{2}}$ has largest eigenvalue less than 1. But $I - A^{\frac{1}{2}} \mathcal{P}_{\text{MG}} A^{\frac{1}{2}}$ is a similarity transformation of V (see Eq. (7.6)),

$$A^{\frac{1}{2}} V A^{-\frac{1}{2}} = I - A^{\frac{1}{2}} \mathcal{P}_{\text{MG}} A^{\frac{1}{2}},$$

which can be written as $A^{\frac{1}{2}} V A^{-\frac{1}{2}} = (R\widetilde{S})^T (R\widetilde{S})$, for

$$R := I - A^{\frac{1}{2}} Q A^{\frac{1}{2}}, \quad \widetilde{S} := I - A^{\frac{1}{2}} \bar{M}^{-1} A^{\frac{1}{2}}.$$

Note that the eigenvalues of $(R\widetilde{S})^T (R\widetilde{S})$ are the singular values squared of $R\widetilde{S}$ (see, e.g., [63]), which are also the eigenvalues of $(R\widetilde{S})(R\widetilde{S})^T = R\widetilde{S}\widetilde{S}^T R$. So, the largest eigenvalue of $A^{\frac{1}{2}} V A^{-\frac{1}{2}}$ is less than 1 if and only if the largest eigenvalue of $R\widetilde{S}\widetilde{S}^T R$ is less than one. This happens if and only if

$$\frac{u^T R (\widetilde{S}\widetilde{S}^T) R u}{u^T u} < 1, \quad \forall u \neq \mathbf{0}_n. \tag{7.13}$$

To maximize this ratio, we write $u = A^{\frac{1}{2}} Z y_1 + R y_2$, and note that R is the $L2$-orthogonal projection onto the orthogonal complement of the range of $A^{\frac{1}{2}} Z$. Then,

$$u^T R (\widetilde{S}\widetilde{S}^T) R u = y_2^T R (\widetilde{S}\widetilde{S}^T) R y_2, \quad u^T u = y_1^T Z^T A Z y_1 + y_2^T R^2 y_2.$$

So, maximizing the ratio over all choices of y_1 means choosing $y_1 = \mathbf{0}_n$, so that the denominator of (7.13) is as small as possible. Therefore,

$$\frac{u^T R\widetilde{S}\widetilde{S}^T Ru}{u^T u} < 1 \quad \forall u \neq \mathbf{0}_n \quad \Leftrightarrow \quad \frac{y_2^T R\widetilde{S}\widetilde{S}^T Ry_2}{y_2^T R^2 y_2} < 1 \quad \forall y_2 \neq \mathbf{0}_n. \tag{7.14}$$

Thus, if the ratio on the right of (7.14) is bounded below 1 for all y_2, so must be the ratio in Eq. (7.13). But, if the ratio in (7.13) is bounded below 1 for all u, then it is bounded for $u = Ry_2$, which gives the bound at the right-hand side of (7.14).

Equivalently, we can maximize the ratio of Eq. (7.14) over $\mathcal{R}(R) = \mathcal{R}(A^{\frac{1}{2}}Z)^{\perp}$. So, the largest eigenvalue of $R\widetilde{S}\widetilde{S}^T R$ is less than 1 if and only if

$$\max_{x : x \perp A^{\frac{1}{2}}Zy \forall y} \frac{x^T \widetilde{S}\widetilde{S}^T x}{x^T x} < 1. \tag{7.15}$$

By computation, we have

$$\widetilde{S}\widetilde{S}^T = I - A^{\frac{1}{2}}\left(\bar{M}^{-1} + \bar{M}^{-T} - \bar{M}^{-1}A\bar{M}^{-T}\right)A^{\frac{1}{2}}.$$

Therefore, the bound (7.15) is equivalent to requiring

$$\min_{x : x \perp A^{\frac{1}{2}}Zy \forall y} \frac{x^T A^{\frac{1}{2}}\left(\bar{M}^{-1} + \bar{M}^{-T} - \bar{M}^{-1}A\bar{M}^{-T}\right)A^{\frac{1}{2}}x}{x^T x} > 0.$$

Taking $w = A^{\frac{1}{2}}x$, this is, in turn, equivalent to

$$\min_{w : w \perp AZy \forall y} w^T\left(\bar{M}^{-1} + \bar{M}^{-T} - \bar{M}^{-1}A\bar{M}^{-T}\right)w > 0,$$

because $w^T A^{-1} w > 0$ for all w. □

Thus, a necessary and sufficient condition for $\mathcal{P}_{\text{MG}}$ to be SPD is given by (7.12). Intuitively, we expect the spectral properties of $\mathcal{P}_{\text{MG}}$ to reflect those of $\bar{M}^{-1}$, with some account for the coarse-grid correction. Eq. (7.12) is particularly interesting in comparison with Theorem 7.3, which gives a necessary and sufficient condition for M^{-1} to define a convergent smoother, see also [57, 180].

Theorem 7.3. *Let $\bar{M}^{-1}$ and Z be as defined in Definition 7.1. Let S be as given in (7.6). A necessary and sufficient condition for S to be convergent in the A-norm is*

$$\min_{w} w^T(\bar{M}^{-1} + \bar{M}^{-T} - \bar{M}^{-1}A\bar{M}^{-T})w > 0. \tag{7.16}$$

Proof. See [57, 180]. □

Theorem 7.3 amounts to the condition

$$\|S\|_A < 1 \;\Leftrightarrow\; \lambda_{\min}(\bar{M} + \bar{M}^T - A) > 0,$$

that can also be found, for example, in [180, Thm. 5.3]. On the other hand, Theorem 7.2 gives

$$\min_{w: w \perp AZy \forall y} w^T \widetilde{M}^{-1} w > 0 \;\Leftrightarrow\; \min_{v: v = \bar{M}^{-T} w, w \perp AZy \forall y} v^T (\bar{M} + \bar{M}^T - A) v > 0,$$

where

$$\widetilde{M}^{-1} := \bar{M}^{-1} + \bar{M}^{-T} - \bar{M}^{-T} A \bar{M}^{-1}. \tag{7.17}$$

Necessarily,

$$\min_{v: v = \bar{M}^{-T} w, w \perp AZy \forall y} v^T (\bar{M} + \bar{M}^T - A) v > \min_y y^T \widetilde{M}^{-1} y = \lambda_{\min}(\bar{M} + \bar{M}^T - A) > 0,$$

so the condition for $\mathcal{P}_{\text{MG}}$ to be SPD is weaker than the condition for a convergent S in the A-norm. In other words, the A-norm convergence of S implies both convergence of $I - \mathcal{P}_{\text{MG}} A$, and that $\mathcal{P}_{\text{MG}}$ is SPD. However, $\mathcal{P}_{\text{MG}}$ can be SPD even if $||S||_A \geq 1$, so long as coarse-grid correction effectively treats amplified modes.

7.4 Comparison of a Special Case of MG and DEF

Here, we show that abstract preconditioners in the MG framework do not always lead to better conditioned systems than DEF. Such problems can even be found in the case of $M^{-1} = \bar{M}^{-1} = I$, see Appendix I. In this section, we show that this can be generalized to arbitrary M^{-1}, but requiring that $\bar{M}^{-1} = M^{-1}$ and Z consisting of eigenvectors of $M^{-1}A$. We start with some spectral bounds on MG and DEF under these assumptions. Thereafter, we perform a comparison between the condition numbers for MG and DEF.

Theorem 7.4 shows the eigenvalue distribution of $\mathcal{P}_{\text{MG}} A$ and $\mathcal{P}_{\text{DEF}} A$, if Z consists of eigenvectors of $M^{-1}A$.

Theorem 7.4. *Suppose that $M^{-1} = \bar{M}^{-1}$ is arbitrary and $\{\lambda_i\}$ is the set of eigenvalues of $M^{-1}A$ with corresponding eigenvectors $\{v_i\}$. Let Z be decomposed of $v_1, \ldots, v_k$. Suppose that MG is convergent, so that $0 \leq \lambda_j \leq 2$ holds for $k < j \leq n$. Furthermore, suppose that the eigenvalues are ordered so that $0 < \lambda_{k+1} \leq \lambda_j \leq \lambda_n \leq 2$ for all $k < j \leq n$. Let $\mathcal{P}_{DEF}$ and $\mathcal{P}_{MG}$ be as given in (7.1) and (7.4), respectively. Then,*

(i) $\mathcal{P}_{MG} A$ has the following eigenvalues:

$$\begin{cases} 1, & \text{for } i = 1, \ldots, k; \\ \lambda_i (2 - \lambda_i), & \text{for } i = k+1, \ldots, n, \end{cases} \tag{7.18}$$

(ii) $\mathcal{P}_{DEF} A$ has the following eigenvalues:

$$\begin{cases} 0, & \text{for } i = 1, \ldots, k; \\ \lambda_i, & \text{for } i = k+1, \ldots, n. \end{cases} \tag{7.19}$$

Proof. The proof follows from [26, Prop. 2] and [173, Sect. 4]. □

Hence, it depends on eigenvalues λ_{k+1} and λ_n of $M^{-1}A$ whether κ_{MG} or κ_{DEF} is more favorable, since

$$\kappa_{\mathrm{MG}} = \frac{1}{\min\{\lambda_{k+1}(2-\lambda_{k+1}), \lambda_n(2-\lambda_n)\}}, \quad \kappa_{\mathrm{DEF}} = \frac{\lambda_n}{\lambda_{k+1}}, \tag{7.20}$$

for any $M^{-1} = \bar{M}^{-1}$ and Z consisting of eigenvectors of $M^{-1}A$. So, for some choices of Z and M^{-1}, MG yields a larger condition number than DEF.

We discuss Figure 7.1 from which the best method can be easily determined for given λ_{k+1} and λ_n. Note first that if $\lambda_{k+1} = \lambda_n$, then $\mathcal{P}_{\mathrm{MG}}$ consists of at most two different eigenvalues, 1 and $\lambda_n(2-\lambda_n)$. In addition, if $\lambda_{k+1} = 2-\lambda_n$, then $\kappa_{\mathrm{MG}} = [\lambda_{k+1}(2-\lambda_{k+1})]^{-1} = [\lambda_n(2-\lambda_n)]^{-1}$. Next, the region corresponding to $0 < \lambda_{k+1} \leq \lambda_n \leq 2$ is naturally partitioned into two subdomains, along the line where $\lambda_{k+1}(2-\lambda_{k+1}) = \lambda_n(2-\lambda_n)$, which occurs when $\lambda_{k+1} = 2-\lambda_n$:

- if $\lambda_{k+1}(2-\lambda_{k+1}) \leq \lambda_n(2-\lambda_n)$, then $\kappa_{\mathrm{MG}} = [\lambda_{k+1}(2-\lambda_{k+1})]^{-1}$. Thus, $\kappa_{\mathrm{MG}} < \kappa_{\mathrm{DEF}}$ if and only if
$$\lambda_{k+1} \leq 2 - \frac{1}{\lambda_n};$$
- if $\lambda_{k+1}(2-\lambda_{k+1}) \geq \lambda_n(2-\lambda_n)$, then $\kappa_{\mathrm{MG}} = [\lambda_n(2-\lambda_n)]^{-1}$. Thus, $\kappa_{\mathrm{MG}} < \kappa_{\mathrm{DEF}}$ if and only if
$$\lambda_{k+1} \leq \lambda_n^2(2-\lambda_n).$$

Figure 7.1 depicts these regions graphically. For any given λ_{k+1} and λ_n, the method with smallest condition number follows immediately from this figure. Example 7.2 gives some consequences of Figure 7.1.

Example 7.2.

(a) *If $\sigma(M^{-1}A) \subseteq (0, 0.5]$, then we deal with Region B_1, and, hence, $\kappa_{DEF} \leq \kappa_{MG}$.*

(b) *If $\sigma(M^{-1}A) \subseteq (0, 2)$ with $\lambda_{k+1} \approx 2-\lambda_n$, then we deal with either Region A_1 or A_2, and $\kappa_{DEF} > \kappa_{MG}$ holds.*

Case (a) says that if M^{-1} is a 'bad' smoother (no eigenvalues of S are less than $\frac{1}{2}$), then MG is expected to converge worse than DEF. On the other hand, Case (b) implies that if M^{-1} is a 'good' smoother (all eigenvalues that need to be handled by relaxation are done so with eigenvalues of S bounded in a neighborhood of the origin), then MG converges better than DEF.

7.5 Effect of Relaxation Parameters

While DEF may have a smaller condition number than MG for some choices of M^{-1} and Z, MG has an added relaxation parameter that is often very important. We illustrate this here by considering $M^{-1} = \bar{M}^{-1} = \alpha I$ for an optimized choice of α. Such a choice of relaxation scheme within MG is commonly known as Richardson relaxation.

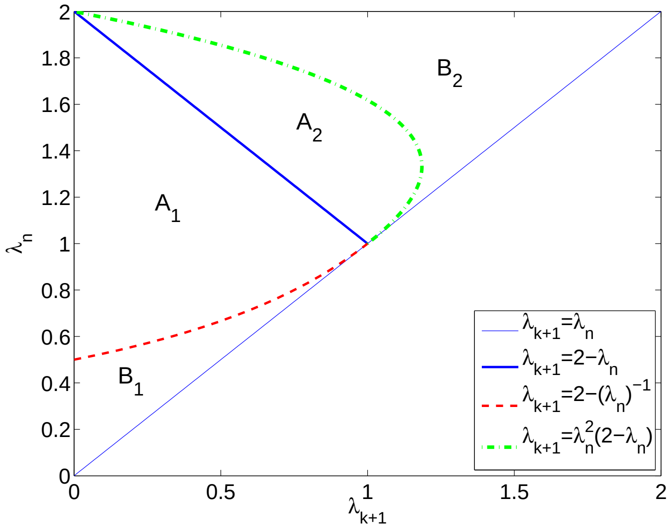


Figure 7.1: Regions where $\kappa_{MG} < \kappa_{DEF}$ (Regions A_1 and A_2) and $\kappa_{DEF} < \kappa_{MG}$ (Regions B_1 and B_2), for arbitrary $M^{-1} = \bar{M}^{-1}$, when Z consists of eigenvectors of $M^{-1}A$. The two condition numbers are equal along the dotted and dotted-dashed lines.

7.5.1 Analysis of Scaling Relaxation

Instead of considering the original linear system, $Ax = b$, we now consider the scaled linear system,

$$\alpha A x = \alpha b, \quad \alpha > 0, \tag{7.21}$$

with $M^{-1} = \bar{M}^{-1} = I$. A subscript, α, is added to the notation for operators and matrices, if they are for (7.21). So, P_α and $\mathcal{P}_{\text{MG},\alpha}$ denote the deflation matrix and MG preconditioner based on (7.21), respectively.

Solving the scaled linear system (7.21) with $M^{-1} = \bar{M}^{-1} = I$ is equivalent to solving the preconditioned linear system, $M^{-1}Ax = M^{-1}b$, with $M^{-1} = \bar{M}^{-1} = \alpha I$. The parameter, α, can then be regarded as a parameter of the relaxation instead of the linear system. The relaxation processes are rescaled, whereas there is no net effect on coarse-grid correction. Therefore, DEF is scaling invariant, i.e.,

$$\kappa_{\text{DEF},\alpha} = \frac{\lambda_n(M^{-1}P_\alpha \alpha A)}{\lambda_{k+1}(M^{-1}P_\alpha \alpha A)} = \frac{\lambda_n(M^{-1}PA)}{\lambda_{k+1}(M^{-1}PA)} = \kappa_{\text{DEF}}.$$

In contrast, MG is not scaling invariant, and the positive-definiteness property of $\mathcal{P}_{\text{MG},\alpha}$ depends strongly on α, since it is well-known that Richardson relaxation is convergent if

$$0 < \alpha < \frac{2}{||A||_2}, \tag{7.22}$$

see, e.g., [180]. For multigrid, we typically try to choose α close to $\frac{1}{||A||_2}$, which guarantees that the slow-to-converge modes of relaxation are only those associated with the small eigenvalues of A. A better choice of α is possible if we make assumptions on how the eigenvectors of A associated with small eigenvalues are treated by coarse-grid correction. It is also possible to get an explicit expression for the optimal α, see the next subsection.

7.5.2 Optimal Choice of α

The best value of α depends on Z, so the optimal α, denoted by α_{opt}, can only be determined if the choice of Z is fixed. In this case, the job of relaxation is specifically to reduce errors that are conjugate to the range of Z. The best choice of α is the one that minimizes the 'spectral radius' of relaxation over the complement of the range of interpolation, i.e.,

$$\min_{w, y^T Z^T A w = 0 \, \forall y} \frac{|w^T (I - \alpha A) w|}{w^T w}.$$

If we restrict ourselves to Z consisting of eigenvectors of A, parameter α_{opt} is easily determined such that it gives the most favorable condition number for MG, see the next theorem.

Theorem 7.5. *Suppose that $M^{-1} = \bar{M}^{-1} = \alpha I$ and $\{\lambda_i\}$ is the increasingly-sorted set of eigenvalues of $M^{-1}A$ with corresponding eigenvectors $\{v_i\}$. Let Z be decomposed of k orthonormal eigenvectors from $\{v_i\}$. Moreover, let $\mathcal{P}_{MG}$ be as given in (7.4) such that $\mathcal{P}_{MG}A$ is SPD. Then, $\kappa(\mathcal{P}_{MG,\alpha}A)$ is minimized for*

$$\alpha_{opt} = \frac{2}{\lambda_{k+1} + \lambda_n}. \tag{7.23}$$

Proof. Note first that, by choosing $\bar{M}^{-1} = M^{-1} = \alpha I$, the error-propagation operator for MG, V, can be written as (cf. Eq. (7.10)).

$$V = I - \mathcal{P}_{\mathrm{MG}}A = (I - \alpha A)P^T(I - \alpha A) = 2\alpha I + Z\Lambda^{-1}Z^T - 2\alpha ZZ^T - \alpha^2 A + \alpha^2 Z\Lambda Z^T.$$

So, applying $\mathcal{P}_{\mathrm{MG}}$ to an eigenvector, v_i, of A gives (cf. Eq. (7.11))

$$\mathcal{P}_{\mathrm{MG}} v_i = \begin{cases} \frac{1}{\lambda_i} v_i, & \text{for } i = 1, \ldots, k; \\ \alpha(2 - \alpha\lambda_i), & \text{for } i = k+1, \ldots, n. \end{cases}$$

Thus, $\mathcal{P}_{\mathrm{MG}}A$ has eigenvalue 1 with algebraic multiplicity k, and $n-k$ eigenvalues of the form $\alpha\lambda_i(2 - \alpha\lambda_i)$, for $i = k+1, \ldots, n$.

Let $\{\sigma_i\}$ be the set of eigenvalues of $\mathcal{P}_{\mathrm{MG}}A$, which are positive and sorted increasingly, so that its condition number is given by $\frac{\sigma_n}{\sigma_1}$. By assumption, $\alpha\lambda_i(2 - \alpha\lambda_i) > 0$ for all $i = k+1, \ldots, n$ and, by calculation, $\alpha\lambda_i(2 - \alpha\lambda_i) < 1$ for all α and λ_i. Thus,

$$\sigma_1 = \min_{i \in [k+1, n]} \{\alpha\lambda_i(2 - \alpha\lambda_i)\}, \quad \sigma_n = 1.$$

Since the function $f(\lambda) := \alpha\lambda(2 - \alpha\lambda)$ is concave down, we have

$$\min_{i \in [k+1, n]} \{\alpha\lambda_i(2 - \alpha\lambda_i)\} = \min\{\alpha\lambda_{k+1}(2 - \alpha\lambda_{k+1}), \alpha\lambda_n(2 - \alpha\lambda_n)\}. \tag{7.24}$$

Subsequently, we want to maximize this minimum eigenvalue,

$$\max_{\alpha} \min \left\{ \alpha\lambda_{k+1}(2-\alpha\lambda_{k+1}), \alpha\lambda_n(2-\alpha\lambda_n) \right\}.$$

This is achieved when we choose α so that

$$\alpha\lambda_{k+1}(2-\alpha\lambda_{k+1}) = \alpha\lambda_n(2-\alpha\lambda_n),$$

which occurs when $\alpha = \frac{2}{\lambda_{k+1}+\lambda_n}$. ☐

Corollary 7.1. *Let the conditions of Theorem 7.5 be satisfied. Then,* $\kappa_{MG} \leq \kappa_{DEF}$.

Proof. If the optimal weighting parameter, α_{opt}, is substituted into (7.24), then the smallest eigenvalue of $\mathcal{P}_{\text{MG}}A$ is equal to

$$\frac{4\lambda_{k+1}\lambda_n}{(\lambda_{k+1}+\lambda_n)^2}. \tag{7.25}$$

As a consequence, the condition number of $\mathcal{P}_{\text{MG}}A$ is given by

$$\kappa_{\text{MG}} = \frac{(\lambda_{k+1}+\lambda_n)^2}{4\lambda_{k+1}\lambda_n}. \tag{7.26}$$

Finally, $\kappa_{\text{MG}} \leq \kappa_{\text{DEF}}$ follows from the fact that

$$\frac{(\lambda_{k+1}+\lambda_n)^2}{4\lambda_{k+1}\lambda_n} \leq \frac{\lambda_n}{\lambda_{k+1}} \quad \Leftrightarrow \quad (\lambda_{k+1}+\lambda_n)^2 \leq (2\lambda_n)^2,$$

which is always true, since $\lambda_{k+1} \leq \lambda_n$. ☐

Remark 7.4.

- *The condition numbers corresponding to MG and DEF are the same if the spectrum of A is 'flat' (i.e., if* $\lambda_{k+1} = \lambda_n$*). But, using the optimized parameter,* α_{opt}*, in MG, it gives a more favorable condition number than DEF.*
- *In Section 7.4, it is shown that* $\kappa_{MG} \geq \kappa_{DEF}$ *can happen in general. However, according to Theorem 7.5, these examples can never be constructed if* α_{opt} *is used.*
- *In practice, approximations to* α *are fairly easy to compute, although the exact eigenvalue distribution is usually unknown. Gershgorin circle theorem (see, e.g., [63, Sect. 8.1.2])) gives us estimates of both* λ_1 *and* λ_n*, which can be used to approximate* λ_{k+1}.
- *An optimal weighting parameter,* α_{opt}*, can also be considered for general preconditioners,* $\bar{M}^{-1}$*; however, it is often much more difficult to express* α_{opt} *explicitly, as it depends on the spectral properties of* $\bar{M}^{-1}A$*, which may not be known. In general, the optimal choice of* α *is such that relaxation converges as quickly as possible on the modes that are not being treated by the coarse-grid correction*

phase. Thus, if the spectral picture of $\bar{M}^{-1}A$ is known well-enough to approximate the eigenvalues corresponding to λ_{k+1} and λ_n, a similar choice of α_{opt} as in Eq. (7.25) may be possible.

7.6 Symmetrizing the Smoother

In the previous section, we have seen that MG can be expected to converge in fewer iterations than DEF for specific choices of M^{-1}, $\bar{M}^{-1}$ and Z. However, the fact that MG requires fewer iterations than DEF for many preconditioners does not mean that it is more efficient, since each iteration of MG is more expensive, due to the choice of two smoothing steps. In order to make a fairer comparison between DEF and MG, we now consider DEF using the preconditioning version of the symmetrized smoother:

$$S^*S = (I - \bar{M}^{-T}A)(I - \bar{M}^{-1}A) = I - \widetilde{M}^{-1}A, \tag{7.27}$$

with

$$\widetilde{M}^{-1} := \bar{M}^{-1} + \bar{M}^{-T} - \bar{M}^{-T}A\bar{M}^{-1}. \tag{7.28}$$

Note that $\widetilde{M}^{-1}$, as defined here, is the same as in Eq. (7.17). Then, we use

$$M^{-1} := \widetilde{M}^{-1} \tag{7.29}$$

as the preconditioner in DEF, since this choice allows implementation in such a way that each iteration of BNN, DEF and MG has similar cost. In this section, we compare the spectra associated with MG, BNN and DEF using (7.29). For general Z and $\bar{M}^{-1}$ such that $\widetilde{M}^{-1}$ is SPD, we show that BNN and DEF, both with preconditioner $\widetilde{M}^{-1}$, and MG yield the same eigenvalues for those modes that are not treated by the coarse-grid correction, see Theorem 7.6.

Theorem 7.6. *Let $\bar{M}^{-1}$ be as given in Definition 7.1 such that $\mathcal{P}_{MG}$ is SPD. In addition, let $M^{-1} = \widetilde{M}^{-1}$ be as defined in (7.28) such that $\mathcal{P}_{BNN}$ is SPD. Then, the eigenvalues of $\mathcal{P}_{MG}A$ and $\mathcal{P}_{BNN}A$ are equal.*

Proof. We show the equivalence of κ_{MG} and κ_{BNN} by examining the extreme eigenvalues of their error-propagation forms,

$$\left\{ \begin{array}{rcl} I - \mathcal{P}_{\mathrm{MG}}A & = & S^*P^TS; \\ I - \mathcal{P}_{\mathrm{BNN}}A & = & P^T(I - \widetilde{M}^{-1}A)P^T. \end{array} \right.$$

We examine both methods by making the same similarity transformation,

$$I - \mathcal{P}A \to A^{\frac{1}{2}}(I - \mathcal{P}A)A^{-\frac{1}{2}}.$$

This allows us to make use of the fact that $I - A^{\frac{1}{2}}QA^{\frac{1}{2}}$ is an orthogonal projection in the L^2-inner product. Computing the similarity transformed systems, we have

$$\begin{cases} A^{\frac{1}{2}}(I - \mathcal{P}_{\text{MG}}A)A^{-\frac{1}{2}} &= (I - A^{\frac{1}{2}}\bar{M}^{-T}A^{\frac{1}{2}})(I - A^{\frac{1}{2}}QA^{\frac{1}{2}})(I - A^{\frac{1}{2}}\bar{M}^{-1}A^{\frac{1}{2}}); \\ A^{\frac{1}{2}}(I - \mathcal{P}_{\text{BNN}}A)A^{-\frac{1}{2}} &= (I - A^{\frac{1}{2}}QA^{\frac{1}{2}})(I - A^{\frac{1}{2}}\widetilde{M}^{-1}A^{\frac{1}{2}})(I - A^{\frac{1}{2}}QA^{\frac{1}{2}}). \end{cases}$$

By defining $C := (I - A^{\frac{1}{2}}QA^{\frac{1}{2}})(I - A^{\frac{1}{2}}M^{-1}A^{\frac{1}{2}})$, we can rewrite the latter expressions as

$$\begin{cases} A^{\frac{1}{2}}(I - \mathcal{P}_{\text{MG}}A)A^{-\frac{1}{2}} &= C^TC; \\ A^{\frac{1}{2}}(I - \mathcal{P}_{\text{BNN}}A)A^{-\frac{1}{2}} &= CC^T, \end{cases}$$

where the following equalities are used:

$$\begin{cases} (I - A^{\frac{1}{2}}QA^{\frac{1}{2}})^2 &= I - A^{\frac{1}{2}}QA^{\frac{1}{2}}; \\ (I - A^{\frac{1}{2}}QA^{\frac{1}{2}})^T &= I - A^{\frac{1}{2}}QA^{\frac{1}{2}}; \\ (I - A^{\frac{1}{2}}\bar{M}^{-1}A^{\frac{1}{2}})^T &= I - A^{\frac{1}{2}}\bar{M}^{-T}A^{\frac{1}{2}}; \\ I - A^{\frac{1}{2}}\widetilde{M}^{-1}A^{\frac{1}{2}} &= (I - A^{\frac{1}{2}}\bar{M}^{-T}A^{\frac{1}{2}})(I - A^{\frac{1}{2}}\bar{M}^{-1}A^{\frac{1}{2}}). \end{cases}$$

Since $A^{\frac{1}{2}}(I-\mathcal{P}_{\text{MG}}A)A^{-\frac{1}{2}}$ and $A^{\frac{1}{2}}(I-\mathcal{P}_{\text{BNN}}A)A^{-\frac{1}{2}}$ are similar to $I-\mathcal{P}_{\text{MG}}A$ and $I-\mathcal{P}_{\text{BNN}}A$, respectively, and, $\sigma(C^TC) = \sigma(CC^T)$ (see Lemma A.1), we obtain

$$\sigma(I - \mathcal{P}_{\text{MG}}A) = \sigma(C^TC) = \sigma(I - \mathcal{P}_{\text{BNN}}A),$$

and the theorem follows immediately. ☐

From Theorem 7.6, we obtain that MG and BNN with $\widetilde{M}^{-1}$ give exactly the same condition number. This also implies that the condition number of MG is surprisingly not smaller than the condition number of DEF, see the next corollary.

Corollary 7.2. *Let $\bar{M}^{-1}$ and $M^{-1} = \widetilde{M}^{-1}$ be as in Theorem 7.6 such that $\mathcal{P}_{DEF}$ is SPD. Then,*

$$\begin{cases} \kappa_{MG} &= \kappa_{BNN}; \\ \kappa_{DEF} &\leq \kappa_{MG}, \end{cases}$$

where $\kappa_{MG}, \kappa_{BNN}$ and κ_{DEF} are the condition numbers corresponding to MG, BNN and DEF, respectively.

Proof. The corollary follows from Theorem 7.6 and [104, Thm. 2.7]. ☐

Remark 7.5.

- *Ordering the smoothers in the opposite way might lead to a different definition of $\widetilde{M}^{-1}$; this, in turn, could change the eigenvalues of MG and BNN, although an analogous result to Theorem 7.6 still holds for the consistent choice of S and $\widetilde{M}^{-1}$.*

- *Corollary 7.2 shows that BNN, DEF and MG are expected to show comparable convergence behavior for special choices of traditional preconditioners. We note that this result is only valid in exact arithmetic. If coarse-grid systems are solved* *inaccurately, DEF might have convergence difficulties, while BNN and MG are less sensitive to it, see the previous chapter.*

7.7 Numerical Experiments

In this section, we present the results of some numerical experiments, where PREC and the 2L-PCG methods are compared. The starting vector for each iterative method is arbitrary and the termination criterion of the iterative process is based on (2.23) with $\delta = 10^{-8}$. We start with a 1-D Poisson-like problem to illustrate the theory obtained in Section 7.4. Then, we consider the same 2-D bubbly flow problem as in Section 6.4 to show the performance of DEF, BNN and MG in a more realistic setting. We stress that these examples are chosen to highlight the presented theory and not to present the efficiency of the solvers; in practice, very different choices of $\bar{M}^{-1}$, M^{-1} and Z are used for each method, see Chapter 9.

7.7.1 1-D Poisson-like Problem

Several 1-D Poisson-like problems are considered, with the matrix

$$A = \begin{bmatrix} \beta & \gamma & & \emptyset \\ \gamma & \beta & \ddots & \\ & \ddots & \ddots & \gamma \\ \emptyset & & \gamma & \beta \end{bmatrix}, \quad \beta, \gamma \in \mathbb{R}, \tag{7.30}$$

where we vary the constants β and γ so that each test case corresponds to a different region as shown in Figure 7.1, see Table 7.2. In addition, we choose $\bar{M}^{-1} = M^{-1} = I$ and Z consisting of eigenvectors corresponding to the smallest eigenvalues of A. Right-hand side, b, is chosen randomly. We take $n = 100$ (other values of n lead to approximately the same results), and the number of projection vectors, k, is varied. The results of the experiment can be found in Table 7.3.

Problem	β	γ	Range of λ_i	Region	Expected Fastest Method
(T1)	1.5	−0.125	[1.25, 1.75]	B2	DEF
(T2)	1	−0.05	[0.9, 1.1]	A1 / A2	MG
(T3)	0.25	−0.1	[0.05, 0.45]	B1	DEF
(T4)	1.25	−0.125	[1.0, 1.5]	A1 / A2	MG/DEF

Table 7.2: Test cases corresponding to different regions as presented in Figure 7.1.

From Table 7.3(a), it can be seen that DEF yields a smaller condition number and is faster than MG for specific choices of β and γ. On the other hand, as observed in Table 7.3(b), β and γ can also be chosen such that MG yields a smaller condition number and is faster than DEF.

Since the condition number associated with DEF is always below that of MG in the case as presented in Table 7.3(c), DEF is expected to be faster than MG. However,

(a) $\beta = 1.5, \ \gamma = -0.125$.

	$k = 2$		$k = 20$		$k = 60$	
Method	# It.	κ	# It.	κ	# It.	κ
PREC	11	1.4	11	1.4	11	1.4
DEF	11	1.4	10	1.3	8	1.1
BNN	11	1.7	10	1.7	8	1.7
MG	15	2.3	15	2.3	12	2.3

(b) $\beta = 1, \ \gamma = -0.05$.

	$k = 2$		$k = 20$		$k = 60$	
Method	# It.	κ	# It.	κ	# It.	κ
PREC	9	1.2	9	1.2	9	1.2
DEF	9	1.2	9	1.2	7	1.1
BNN	9	1.2	9	1.2	7	1.1
MG	5	1.01	5	1.01	5	1.01

(c) $\beta = 0.25, \ \gamma = -0.1$.

	$k = 2$		$k = 20$		$k = 60$	
Method	# It.	κ	# It.	κ	# It.	κ
PREC	34	9.0	34	9.0	34	9.0
DEF	34	8.8	24	4.9	11	1.4
BNN	34	19.6	25	11.0	11	3.2
MG	30	10.1	22	5.7	11	1.9

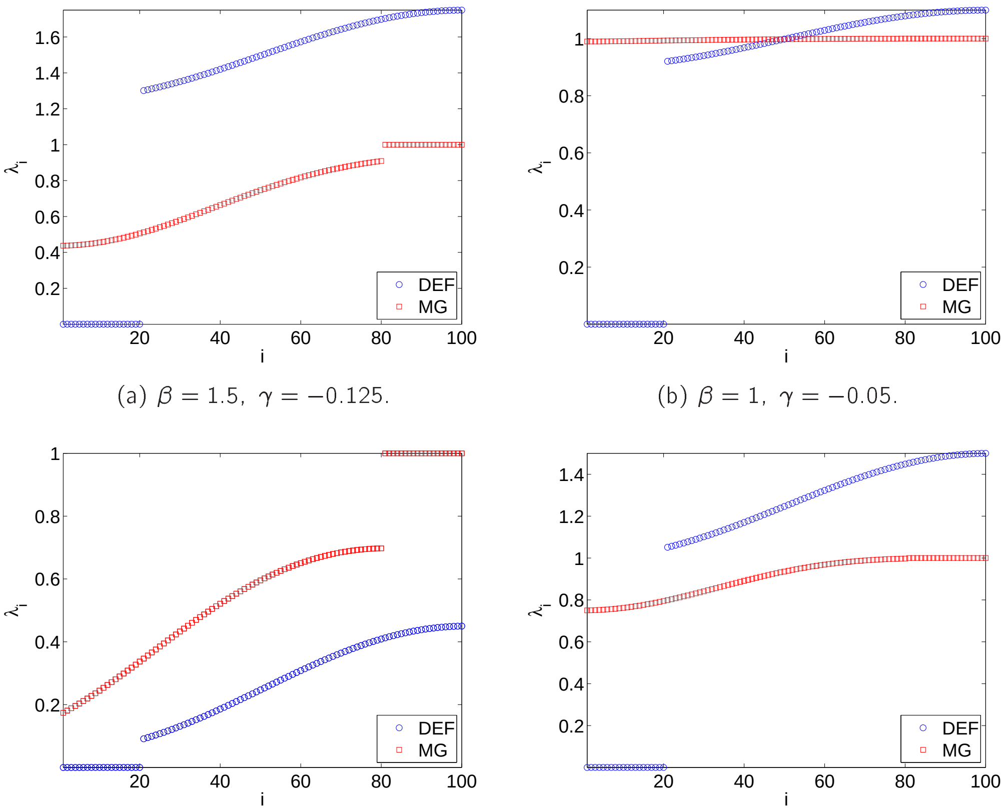
(d) $\beta = 1.25, \ \gamma = -0.125$.

	$k = 2$		$k = 20$		$k = 60$	
Method	# It.	κ	# It.	κ	# It.	κ
PREC	11	1.5	11	1.5	11	1.5
DEF	12	1.5	11	1.4	8	1.1
BNN	12	1.5	11	1.5	8	1.5
MG	10	1.3	10	1.3	9	1.3

Table 7.3: Results of the experiment with test cases as presented for the Poisson-like problem in Table 7.2. The results are presented in terms of number of iterations, # It., and condition number, κ.

that is not the case in this test problem. The two methods converge at the same rate for large k, but MG is faster than DEF for small k. This can be explained by the fact that the spectrum of eigenvalues of MG consists of two clusters, see Figure 7.2(c). If the first cluster of ones is omitted (or is approximated by a Ritz value), then the condition number of the remaining spectrum is favorable when compared to that of DEF. For example, in the case of $k = 2$, we have $\kappa_{\text{MG}} = 7.0$ (instead of $\kappa_{\text{MG}} = 10.1$) when the unit eigenvalues are omitted. Obviously, this would then be the smallest condition number over all of the methods.

Finally, MG has a smaller condition number and is faster than DEF for small k in the case presented in Table 7.3(d). On the other hand, for large k, DEF has a smaller condition number than MG and performs somewhat better than MG. Indeed, the best method depends on λ_{k+1} for this case with specific β and γ.



(a) $\beta = 1.5,\ \gamma = -0.125$.

(b) $\beta = 1,\ \gamma = -0.05$.

(c) $\beta = 0.25,\ \gamma = -0.1$.

(d) $\beta = 1.25,\ \gamma = -0.125$.

Figure 7.2: Eigenvalues associated with DEF and MG for the test cases with $k = 20$ as presented in Table 7.3.

7.7.2 2-D Bubbly Flow Problem

In this section, a numerical comparison of the two-level PCG methods is performed using 2-D bubbly flows with $m = 5$, $\epsilon = 10^3$, and $s = 0.05$. As in Section 6.4, M^{-1} is chosen to be the IC(0) preconditioner and subdomain deflation vectors are taken as projection vectors based on Variant 5.2 (see Section 5.3)..

Experiment with $\bar{M}^{-1} = M^{-1}$

The results with $\bar{M}^{-1} = M^{-1}$ are presented in Table 7.4 (cf. Table 6.4).

From the table, it can be observed that, for all k, DEF and BNN require the same number of iterations, whereas MG is the fastest method in terms of the number of iterations, which is as expected. Recall that this does not necessarily mean that MG is the fastest method with respect to computing time, since each iteration of MG is more expensive than an iteration of DEF.

	$k = 2^2$		$k = 4^2$		$k = 8^2$	
Method	# It.	$\frac{\|\|x_{\mathrm{it}}-x\|\|_2}{\|\|x\|\|_2}$	# It.	$\frac{\|\|x_{\mathrm{it}}-x\|\|_2}{\|\|x\|\|_2}$	# It.	$\frac{\|\|x_{\mathrm{it}}-x\|\|_2}{\|\|x\|\|_2}$
DEF	149	1.5×10^{-8}	144	3.1×10^{-8}	42	1.8×10^{-8}
BNN	149	1.5×10^{-8}	144	3.1×10^{-8}	42	1.1×10^{-8}
MG	86	1.0×10^{-7}	93	6.5×10^{-8}	32	1.9×10^{-8}

Table 7.4: Number of required iterations for convergence and the 2−norm of the relative errors of 2L-PCG methods, for the bubbly flow problem with $n = 64^2$ and $\bar{M}^{-1} = M^{-1}$. PREC requires 137 iterations and leads to a relative error of 4.6×10^{-7}.

Experiment with Symmetrized Smoother

We perform the same experiment as above, but now taking $M^{-1} = \bar{M}^{-1} + \bar{M}^{-T} - \bar{M}^{-T} A \bar{M}^{-1}$, while $\bar{M}^{-1}$ is still the IC(0) preconditioner. In contrast to the previous experiment, the amount of work for each iteration of BNN, MG and DEF is now approximately the same and Theorem 7.6 holds. The results of this experiment are presented in Table 7.5.

	$k = 2^2$		$k = 4^2$		$k = 8^2$	
Method	# It.	$\frac{\|\|x_{\mathrm{it}}-x\|\|_2}{\|\|x\|\|_2}$	# It.	$\frac{\|\|x_{\mathrm{it}}-x\|\|_2}{\|\|x\|\|_2}$	# It.	$\frac{\|\|x_{\mathrm{it}}-x\|\|_2}{\|\|x\|\|_2}$
DEF	87	7.2×10^{-8}	94	1.3×10^{-8}	34	7.6×10^{-9}
BNN	87	7.2×10^{-8}	94	1.3×10^{-8}	34	7.6×10^{-9}
MG	86	1.0×10^{-7}	93	6.5×10^{-8}	32	1.9×10^{-8}

Table 7.5: Number of required iterations for convergence and the 2−norm of the relative errors of 2L-PCG methods, for the bubbly flow problem with $n = 64^2$ and $M^{-1} = \bar{M}^{-1} + \bar{M}^{-T} - \bar{M}^{-T} A \bar{M}^{-1}$. PREC requires 137 iterations and leads to a relative error of 4.6×10^{-7}.

As can be observed in Table 7.5, MG is now comparable with DEF and BNN, as expected from the theory of Section 7.6. All methods require approximately the same number of iterations and lead to the same accuracy.

7.8 Concluding Remarks

We compare two-level PCG methods based on deflation (DEF), balancing Neumann-Neumann (BNN) and multigrid V(1,1)-cycle (MG) preconditioners in their abstract forms, which all consist of combinations of traditional and projection-type preconditioners. When specific choices are made for the algorithmic components, each MG iteration is more expensive than a DEF or BNN iteration, due to the more sophisticated form of the two-level preconditioner. At first glance, we would expect MG to be the most effective method; however, we show that there exist some traditional and projection preconditioners such that DEF is expected to converge faster than MG in exact arithmetic.

If Richardson relaxation is used with an optimal weighting as a traditional preconditioner, then we prove that MG always gives a more favorable condition number than DEF or BNN. For more sophisticated and effective traditional preconditioners, we still expect MG to be superior to DEF and BNN, although the work per iteration of MG remains more than for the other methods.

For special choices of traditional preconditioners, we show that BNN, DEF and MG require the same amount of work per iteration and their spectra only differ in one cluster of eigenvalues around 0 or 1. Hence, these methods are expected to show comparable convergence behavior, assuming that coarse-grid systems are solved accurately.

The gap from the previous chapter is filled by taking the MG preconditioner into account. For certain choices of parameters, this two-level PCG method is strongly related to those as discussed in that chapter. The different methods with their optimized set of parameters are further examined in the upcoming two chapters.

Chapter 8

Efficiency and Implementation of the Deflation Method

8.1 Introduction

The deflation method (also known as DPCG and DEF) has been introduced in Chapter 3, and, subsequently, some aspects of this method have been examined in the subsequent chapters. Efficiency and implementation issues have not been extensively taken into consideration so far in this thesis. The aim of this chapter is to focus on those issues. We show that a good implementation of the deflation method is essential in order to obtain a powerful and efficient method.

We have seen in the prior chapters that increasing the number of projection vectors usually leads to a faster convergence of the iterative process. This does not give a more efficient deflation method in general, since the cost of each iteration becomes higher due to larger Galerkin systems (i.e., linear systems involving the Galerkin matrix, E) that should be solved. Hence, there is always an optimum of the number of projection vectors, regarding the total computing time that is required to find the solution using the deflation method. This optimum depends on many aspects, such as

- sparsity pattern and dimension of the coefficient matrix, A;
- choice of the preconditioner, M^{-1};
- choice and dimensions of the deflation-subspace matrix, Z;
- way of computing the matrix-vector product Py, where P is the deflation matrix and y is an arbitrary vector.

The latter aspect, computing Py, can be divided into several steps, see Algorithm 8. The efficiency of implementing each line of this algorithm influences both the efficiency of the whole deflation method and the optimal number of projection vectors that should be chosen. In general, an optimum cannot be determined beforehand, but it is common that a relatively low number of projection vectors often improves the efficiency.

In this chapter, Assumption 8.1 holds in order to determine the efficiency of the deflation method for one specific problem setting. Numerical experiments are performed to determine the optimal choice of the number of subdomain projection vectors with respect to the total computing time.

Algorithm 8 Computation of Py

```
$y_1 := Z^T y$
Solve $E y_2 = y_1$
$y_3 := (AZ) y_2$
$Py := y - y_3$
```

Assumption 8.1.

- *A is derived after discretization of the Poisson problem that is originated from bubbly flow applications (see Section 1.3), and it consists of 7 nonzero diagonals in the 3-D case;*

- *M^{-1} is the IC(0) preconditioner (see Section 2.5.1), so that the resulting deflation method is DICCG (see Section 3.6);*

- *Z consists of subdomain projection vectors (see Section 4.2.3). In addition, the number of subdomains and deflation vectors is assumed to be equal.*

The deflation method can be regarded as a two-grid method, because a Galerkin system has to be solved at each iteration. If these systems are solved recursively, then we would obtain a method that is very close to multigrid methods, see Chapter 9. In this chapter, we restrict ourselves to Galerkin systems that are solved in either a direct or an iterative way. In the latter case, the resulting deflation method can be interpreted as an inner-outer iteration process, which requires a special treatment. For example, attention should be paid to the stability and termination criteria for both the inner- and outer-iteration process. We examine this issue in this chapter. Moreover, some theoretical results are presented for the deflation method with a singular Galerkin matrix. These insights provide us a better understanding of the efficiency of the deflation method.

Remark 8.1. *Subdomain deflation lends itself for an efficient parallel implementation. This issue is discussed in Appendix F.*

This chapter is organized as follows. In Section 8.2, we show the efficient implementation of the matrix-vector product, Py. Section 8.3 is devoted to the treatment of Galerkin systems and the associated deflation methods. In Section 8.4, we focus on the inner-outer iteration process and its stability properties. Numerical experiments are performed in Section 8.5, and some concluding remarks are presented in Section 8.6.

8.2 Computations with the Deflation Matrix

In order to obtain a fast solver, the deflation method should be implemented efficiently in a program code. In this section, we show how this can be done by considering each step of the computations with the deflation matrix. We restrict the analysis to the 3-D case, since the treatment of the 2-D case is similar. We remark that the main part of this analysis is only valid for 3-D regular grids, and that the floating point operations (flops) counts and their analysis only hold for subdomain deflation, where nonoverlapping identical cubes are used as subdomains. An further discussion of the deflation operations can be found in Appendix D and [140], whereas the flop counts of ICCG and DICCG are analyzed in Appendix E and [140].

8.2.1 Construction of AZ

The matrix-matrix product AZ can be computed efficiently by determining only the nonzero entries. The outcome of this product is stored as a small matrix, denoted by $S_{AZ} \in \mathbb{R}^{\gamma \times 3}$, where $\gamma \in \mathbb{N}$ is the number of nonzero entries of the matrix AZ. The first and second columns of S_{AZ} are filled with the row and column indices of the nonzero entries of AZ, respectively. The third column of S_{AZ} stores the corresponding values of these nonzero entries. The entries of S_{AZ} can be determined efficiently, since Z represents subdomains, $\{\Omega_j\}$, which are nonoverlapping cubes. Moreover, AZ only has nonzero contributions near the interfaces of these cubes, and, hence, it consists of relatively many zeros. So, the few nonzero entries of AZ may be known beforehand.

Example 8.1. *Let* $A \in \mathbb{R}^{4\times 4}$ *and* $Z \in \mathbb{R}^{4\times 2}$ *be matrices, obtained from the 1-D Poisson-like problem, given by*

$$A = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}.$$

Then, this leads immediately to $\gamma = 4$ *and*

$$AZ = \begin{bmatrix} 0 & 0 \\ 1 & -1 \\ -1 & 1 \\ 0 & 0 \end{bmatrix}, \quad S_{AZ} = \begin{bmatrix} 2 & 1 & 1 \\ 3 & 1 & -1 \\ 2 & 2 & -1 \\ 3 & 2 & 1 \end{bmatrix}.$$

Considering the number of flops, it is not difficult to show that constructing S_{AZ} requires $\mathcal{O}(n^{\frac{2}{3}} k^{\frac{1}{3}})$ flops in the 3-D case, see Section E.1.

8.2.2 Construction of E

The Galerkin matrix, $E := Z^T AZ$, can be easily formed during the construction of AZ. Each nonzero entry of AZ makes exactly one contribution to E, by simply adding the value to the corresponding entry of E.

The way of solving the Galerkin system, $Ey_2 = y_1$, determines the best storage of E, see Section 8.3. The matrix corresponding to this efficient storage of E is denoted by S_E. For the time being, the number of flops to solve a Galerkin system is denoted by ϑ.

8.2.3 Calculation of Py and $P^T y$

In contrast to AZ and E, the deflation matrix, P, is not constructed explicitly. Instead, each step of the matrix-vector product Py, as presented in Algorithm 8, is performed separately. In the same way, $P^T y$ can be treated. Both algorithms require $\mathcal{O}(n + \vartheta)$ flops.

Note that Z is not stored explicitly, since the matrix-vector products, $Z^T y$ and Zy_2, can be simply determined from y, requiring $\mathcal{O}(n)$ flops. Furthermore, both $(AZ)y_2$ and $(AZ)^T y$ can also be easily computed, since S_{AZ} is known. Both computations require $\mathcal{O}(n^{\frac{2}{3}} k^{\frac{1}{3}})$ flops in the 3-D case, see Section E.1.

8.3 Efficient Solution of Galerkin Systems

In this section, we demonstrate the strategies to solve Galerkin systems (i.e., Line 2 of Algorithm 8) efficiently. Recall first from Chapter 5 that three deflation variants can be used, see Table 8.1 (which is the same as Table 5.1).

	Matrices				
Variant	Coefficient	Deflation-subspace	Galerkin	Correction	Deflation
5.1	A	Z_{k-1}	E_{k-1}	Q_{k-1}	P_{k-1}
5.2	$\bar{A}$	Z_k	$\bar{E}_k$	$\bar{Q}_k$	$\bar{P}_k$
5.3	A	Z_k	E_k	Q_k	P_k

Table 8.1: Corresponding matrices of the proposed deflation variants in Chapter 5.

We distinguish two main DICCG methods in this chapter, which only differ in the solver of the Galerkin systems, see Definition 8.1.

Definition 8.1.

- *DICCG1−k is defined as DICCG corresponding to any deflation variant of Table 8.1, where each Galerkin system is solved directly.*
- *DICCG2−k is defined as DICCG corresponding to any deflation variant of Table 8.1, where each Galerkin system is solved iteratively.*

Remark 8.2.

- *If there is no ambiguity, we omit the bars on matrices, and subscripts associated with the matrices. In addition, DICCG1−k and DICCG2−k are shortly denoted by DICCG1 and DICCG2, if k is unspecified.*

- *If one applies Variant 5.3 in DICCG1, then extra care is needed to solve the corresponding Galerkin systems. The direct solver should generate a solution up to the null space of the Galerkin matrix.*
- *Any deflation variant, as presented in Table 8.1, can be used for both DICCG1 and DICCG2, since we have shown in Section 5.4 that all variants are (almost) mathematically equivalent. However, for convenience, we restrict ourselves to Variant 5.1 and 5.2 for DICCG1, and Variant 5.1 and 5.3 for DICCG2, in this chapter.*

In this section, we first demonstrate how a Galerkin system, $Ey_2 = y_1$, can be solved efficiently for both DICCG1 and DICCG2. Both DICCG methods require a different treatment. We then compare these methods theoretically. We use k_x that denotes the number of grid points in each spatial direction of a subdomain, i.e., $k_x := \sqrt[3]{\frac{n}{k}}$, assuming that k is a divisor of n.

8.3.1 Galerkin Systems within DICCG1

To solve $Ey_2 = y_1$ with a direct method, we apply the band-Cholesky decomposition [63, Sect. 4.3.5], and, thereafter, band-back/forward substitution [63, Sect. 4.3.2]. In this case, the bandwidth of E is $k^{\frac{2}{3}} + k^{\frac{1}{3}}$, making the decomposition efficient only for relatively small k.

Recall that both E_k and E_{k-1} are invertible, so that their band-Cholesky decompositions exist. Furthermore, constructing the Cholesky decomposition requires $\mathcal{O}(k^{\frac{7}{3}})$ flops, whereas the backward and forward substitutions take $\mathcal{O}(k^{\frac{5}{3}})$ flops.

8.3.2 Galerkin Systems within DICCG2

To find a solution of $Ey_2 = y_1$ in DICCG2, we apply the iterative solver ICCG. This is possible and efficient, since E has the same properties as A. Obviously, E is SPSD and has a similar sparsity pattern to A, because Z is based on nonoverlapping subdomains. Moreover, E is better conditioned than A, see Theorem 8.1.

Theorem 8.1. *Let A, E_{k-1} and E_k be as in Table 8.1. Then, the following inequalities hold:*

$$\kappa(E_{k-1}) \leq \kappa(A), \quad \kappa(E_k) \leq \kappa(A). \tag{8.1}$$

Proof. Note first that both E_{k-1} and E_k have rank $k-1$, as Z has full rank and the algebraic multiplicity of the zero eigenvalue of A is one. In addition, without loss of generality (see Theorem C.2), we rescale Z_k with $\sqrt{\frac{n}{k}}$ such that it satisfies $Z_k^T Z_k = I$. In order to prove the left-hand inequality of (8.1), it suffices to show that

$$\lambda_2(A) \leq \mu_2(E_k) \quad \text{and} \quad \mu_k(E_k) \leq \lambda_n(A), \tag{8.2}$$

where $0 = \mu_1(E_k) < \mu_2(E_k) \leq \ldots \leq \mu_k(E_k)$ and $0 = \lambda_1(A) < \lambda_2(A) \leq \ldots \leq \lambda_n(A)$ are the eigenvalues of E_k and A, respectively.

The inequalities (8.2) can be derived from Theorem A.2 (that is the Courant-Fischer Minimax Theorem). From this theorem, we obtain in particular

$$\lambda_2(A) = \min_{w^T w=1,\ w \perp w_1(A)} w^T A w, \quad \lambda_n(A) = \max_{w^T w=1} w^T A w, \tag{8.3}$$

where $u_1(A)$ is the eigenvector corresponding to $\lambda_1(A)$, see [73, Sect. 4.2] for more details.

Note that $u_1(A) = \mathbf{1}_n$ and $u_1(E) = \mathbf{1}_k$ hold due to Assumption 1.2 and Eq. (5.8). In addition, for $w := Z_k y$, we have

$$\begin{cases} w^T A w & = & (Z_k y)^T A Z_k y & = & y^T E y; \\ (Z_k y)^T (Z_k y) & = & y^T y; \\ (Z_k y)^T \mathbf{1}_n & = & y^T Z_k^T \mathbf{1}_n & = & y^T \mathbf{1}_k, \end{cases}$$

using Property 5.1(i). Hence, this implies

$$\min_{(Z_k y)^T (Z_k y)=1,\ Z_k y \perp \mathbf{1}_n} (Z_k y)^T A (Z_k y) = \min_{y^T y=1,\ y \perp \mathbf{1}_k} y^T E y. \tag{8.4}$$

Now, combining Eqs. (8.3) and (8.4) gives us

$$\lambda_2(A) = \min_{w^T w=1,\ w \perp \mathbf{1}_n} w^T A w \leq \min_{y^T y=1,\ y \perp \mathbf{1}_k} y^T E y = \mu_2(E),$$

which is the left inequality of (8.2). For the right inequality of (8.2), it follows in a similar way that

$$\mu_k(E) = \max_{y^T y=1} y^T E y \leq \max_{w^T w=1} w^T A w = \lambda_n(A),$$

where we have applied

$$\max_{(Z_k y)^T (Z_k y)=1} (Z_k y)^T A (Z_k y) = \max_{y^T y=1} y^T E y.$$

The right-hand inequality of (8.1) can be proven in a similar way as above. □

Next, there is no need to force invertibility of E_k, since ICCG can deal with a singular coefficient matrix. We only have to ensure the consistency of all Galerkin systems during the outer-iteration process of DICCG2, see Theorem 8.2.

Theorem 8.2. *Let DICCG2 be as given in Definition 8.1. Then, all Galerkin systems within DICCG2 are consistent.*

Proof. Recall that E_{k-1} is invertible, so that we can restrict ourselves to $E := E_k$. The Galerkin system, $E y_2 = y_1$, appears three times in Algorithm 6 (Lines 1, 3 and 11), which are treated separately below.

In the matrix-vector product $P r_0$, we have to solve the Galerkin system $E y_2 = Z^T r_0$.

This system is consistent, since it is compatible due to Eq. (5.8) and

$$(Z^T r_0)^T \mathbf{1}_k = r_0^T Z \mathbf{1}_k = r_0^T \mathbf{1}_n = b^T \mathbf{1}_n - x_0^T A \mathbf{1}_n = \mathbf{0}_n - x_0^T \mathbf{0}_n = \mathbf{0}_n. \tag{8.5}$$

Moreover, since

$$(Z^T A p_j)^T \mathbf{1}_k = p_j^T A Z \mathbf{1}_k = p_j^T A \mathbf{1}_n = \mathbf{0}_n, \tag{8.6}$$

the system $E y_2 = Z^T A p_j$ is compatible as well. Hence, $P A p_j$ is consistent. Finally, using the same argument as above, we conclude that $P^T \tilde{x}_{j+1}$ is also consistent, using the fact that $P^T \tilde{x}_{j+1} = \tilde{x}_{j+1} - Z E^+ Z^T A \tilde{x}_{j+1}$. ☐

From Theorem 8.2, we conclude that it is possible to solve each Galerkin system, $E y_2 = y_1$, iteratively. Each of the ICCG steps costs $\mathcal{O}(k)$ flops, and the efficiency of this method depends on the number of required inner ICCG iterations.

Remark 8.3. *Note that the solution of the Galerkin systems in Variant 5.3 is not unique, since it is determined up to a constant vector. If y_2 is a solution of $E_k y_2 = y_1$, then $y_2 + \alpha \mathbf{1}_k$ with $\alpha \in \mathbb{R}$ is also a solution. Fortunately, $y_3 := A Z_k (y_2 + \alpha \mathbf{1}_k)$ is unique, due to the fact that $A Z_k \mathbf{1}_k = A \mathbf{1}_n = \mathbf{0}_n$. Hence, Algorithm 8 gives a unique Py for all deflation variants.*

Recall that, in the case of DICCG2, we have an inner-outer iterative process with DICCG as an outer-iteration process and ICCG as an inner-iteration process, so that we need two different termination criteria. The inner and outer tolerances are denoted by δ_{outer} and δ_{inner}, respectively, which satisfy

$$\delta_{\text{inner}} = \omega \, \delta_{\text{outer}}, \quad \omega > 0. \tag{8.7}$$

For large $\omega \geq 1$, DICCG2 does not converge, as the method is sensitive to inaccurate solves of the Galerkin systems, see also [103, Sect. 3]. However, for small $\omega \ll 1$, the convergence of the inner iterations of DICCG2 is relatively slow; the inner-iteration process may stagnate or even diverge due to a too severe termination tolerance. Therefore, ω should be chosen carefully to obtain an convergent and efficient method. From our numerical experiments with bubbly flows, it appears that

$$\omega = 10^{-2} \tag{8.8}$$

is an appropriate choice, but it usually depends on many factors, see [64, 124] for more details.

Remark 8.4. *For problems with a large grid size or large jumps in the coefficient of the PDEs, it could be advantageous to solve the inner iterations with DICCG, instead of ICCG. The inner iterations could even be solved by a recursive application of DICCG, see, e.g., [48]. This is in analogy with multigrid-like methods, see also [56, Sect. 3].*

8.3.3 Comparison of Galerkin Matrices

Here, we examine the Galerkin matrices in the different deflation methods in order to determine the fastest method.

Note first that the Galerkin matrices, E_{k-1} and E_k, satisfy

$$E_k = \left[\begin{array}{cc} E_{k-1} & \times \\ \times & \times \end{array} \right], \tag{8.9}$$

where × represents some irrelevant entries of E_k. Then, we can show that the eigenvalues of E_k and E_{k-1} interlace, see Theorem 8.3.

Theorem 8.3. *Let E_{k-1} and E_k be as given in Table 8.1. Then, the following inequalities hold:*

$$0 = \lambda_1(E_k) \leq \lambda_1(E_{k-1}) \leq \lambda_2(E_k) \leq \ldots \leq \lambda_{k-1}(E_k) \leq \lambda_{k-1}(E_{k-1}) \leq \lambda_k(E_k).$$

Proof. The theorem follows immediately from the interlacing property (Lemma A.7). □

In contrast to the case that A is invertible, the spectrum of E_{k-1} is not in the range of the nonzero eigenvalues of E_k, i.e., $\kappa(E_{k-1})$ is not smaller than $\kappa(E_k)$. But, in practice, we often see that the largest eigenvalues of both matrices are almost the same, whereas the smallest nonzero eigenvalues differ significantly, i.e., we have

$$\lambda_{k-1}(E_{k-1}) \rightarrow \lambda_k(E_k), \quad \lambda_1(E_{k-1}) \ll \lambda_2(E_k),$$

for large k. This yields

$$\kappa(E_k) \leq \kappa(E_{k-1}).$$

Accordingly, one should iterate with E_k, rather than E_{k-1}, to obtain the fastest expected convergence of the inner-iteration process. In other words, Variant 5.3 is the variant of choice in DICCG2 and, hence, this is used in Section 8.5.3.

8.3.4 Deflation Properties for a Singular Galerkin Matrix

In Section 3.5, we have presented some theoretical results based on deflation matrices whose Galerkin matrix, E, is nonsingular. Under certain circumstances, these results also hold for the case with a singular Galerkin matrix. We start with Assumption 8.2 that is satisfied throughout this subsection.

Assumption 8.2. *Suppose that $Z_k = [Z_{k-1},\ z_k]$ with $Z_{k-1} \in \mathbb{R}^{n \times (k-1)}$ and $z_k \in \mathbb{R}^n$ holds. Then, we assume that $E_k := Z_k^T A Z_k$ is a singular SPSD matrix, whose pseudo-inverse E_k^+ satisfies*

$$E_k^+ = \left[\begin{array}{cc} E_{k-1}^{-1} & \mathbf{0}_{k-1} \\ \mathbf{0}_{k-1}^T & 0 \end{array} \right], \tag{8.10}$$

where $E_{k-1} := Z_{k-1}^T A Z_{k-1}$ is an invertible SPD matrix.

Note that Eq. (8.10) can be deduced from Eq. (8.9) by choosing zeros for ×. In fact, this subsection deals with a particular choice for E_{k-1} and E_k.

From Assumption 8.2, we have that the nonzero eigenvalue distributions of E_k and E_{k-1} are identical, so that Theorem 8.4 follows immediately. This results in the fact that the convergence of the inner solver for Galerkin systems within both DICCG1 and DICCG2 is the same, if they are performed by a PCG method and the Galerkin matrices satisfy Assumption 8.2.

Theorem 8.4. *Let E_{k-1} and E_k satisfy Assumption 8.2. Then, $\kappa(E_{k-1}) = \kappa(E_k)$.*

Moreover, the proof of Theorem 5.5 is straightforward if Assumption 8.2 is satisfied. For completeness, this is presented below.

Theorem 8.5. *Let A and M^{-1} be an SPSD and SPD matrix, respectively. Let P_k and P_{k-1} be as defined in Table 8.1. Suppose that Assumption 8.2 is satisfied. Then, $M^{-1}P_kA = M^{-1}P_{k-1}A$ holds.*

Proof. Using

$$\begin{aligned} Q_k &= Z_k E_k^+ Z_k^T \\ &= [Z_{k-1},\ z_k] \begin{bmatrix} E_{k-1}^{-1} & \mathbf{0}_{k-1} \\ \mathbf{0}_{k-1}^T & 0 \end{bmatrix} [Z_{k-1},\ z_k]^T \\ &= Z_{k-1} E_{k-1}^+ Z_{k-1}^T \\ &= Q_{k-1}, \end{aligned}$$

the theorem follows immediately. □

8.4 Stabilization of the Deflation Method

In practice, the deflation method (DPCG or DEF) might be not robust if the number of projection vectors, k, is relatively large. This is caused by the fact that Galerkin systems involving E_k or E_{k-1} might also become large and cannot be solved accurately. The deflation method can be stabilized by adding a correction matrix in the linear system, i.e., we solve (cf. Eq. (6.11))

$$(P^T M^{-1} + Q)Ax = (P^T M^{-1} + Q)b, \tag{8.11}$$

with starting vector $x_0 = Qb + P^T\tilde{x}_0$ and an arbitrary vector $\tilde{x}_0$. The resulting method is called the adapted deflation method (ADPCG or A-DEF) method, which is equal to the A-DEF2 method as introduced and analyzed in Chapter 6.

Remark 8.5.

- *The operator $P^T M^{-1} + Q$ in Eq. (8.11) cannot be replaced by $M^{-1}P + Q$, because it has been shown in Chapter 6 that the resulting method may suffer from instability.*

- *The solution, x, in (8.11) is the same as the solution of $Ax = b$, since $P^T M^{-1} + Q$ is invertible.*
- *The matrix-vector product Qy for any $y \in \mathbb{R}^n$ can be carried out efficiently in a similar way as Py or $P^T y$ (see Section 8.2.3).*

It has been demonstrated in Chapter 6 that ADPCG can be derived from the well-known balancing Neumann-Neumann (BNN) method [89]. In addition, both methods have more-or-less the same favorable robustness properties, due to the following theorem (which is Theorem 6.3 for an SPSD coefficient matrix, A).

Theorem 8.6. *Let A and M^{-1} be an SPSD and SPD matrix, respectively. Suppose that P is a deflation matrix corresponding to any deflation variant as presented in Table 8.1. Let the spectra of DPCG and ADPCG be given by*

$$\sigma(M^{-1}PA) = \{\lambda_1, \ldots, \lambda_n\}, \quad \sigma(P^T M^{-1} A + QA) = \{\mu_1, \ldots, \mu_n\},$$

respectively. Then, the numbering of the eigenvalues within these spectra can be such that the following statements hold:

$$\begin{cases} \lambda_i = 0, \ \mu_i = 1, & \text{for } i = 1, \ldots, k; \\ \lambda_i = \mu_i, & \text{for } i = k+1, \ldots, n. \end{cases}$$

Proof. The proof is almost the same as the proof of Theorem 6.3, see also [85]. ☐

If Galerkin systems with E_k or E_{k-1} are solved inaccurately, then the zero eigenvalues associated with DPCG become nearly zero, resulting in a method that is not robust. On the other hand, we do not have this phenomenon in the ADPCG method, since the corresponding eigenvalues of $M^{-1}A$ are projected to one instead of zero, see Chapter 6 for more details. It follows that if the Galerkin system, $Ey_2 = y_1$, is solved iteratively, then this can be done with a lower accuracy for ADPCG, compared with DPCG. As discussed in Section 8.3, the Galerkin systems within DPCG should be solved accurately. Following the discussion above, the expectation is that a larger ω in Eq. (8.7) can be taken in the ADPCG method. In the numerical experiments (of Section 8.5.3, we investigate the choice of ω for both DPCG and ADPCG in more detail.

Remark 8.6. *If the Galerkin system, $Ey_2 = y_1$, is solved inaccurately, the resulting operator $P^T M^{-1} + Q$ is varying at each iteration, while a fixed operator is expected in the CG process. In other words, the operation*

$$y_2 = (P^T M^{-1} + Q)^{-1} y_1$$

seen by the outer process turns out to be

$$y_2 = \mathcal{F}(y_1),$$

where $\mathcal{F}$ can be regarded as a nonlinear mapping. If the inner tolerance is too loose, the optimal convergence property of the CG process can only be preserved, if one performs a full orthogonalization of the search direction vectors that can be extended with truncation and restart strategies. This results in GMRES-like methods, such as the Flexible CG method [109]. We also consider this variant in Section 8.5.3. However, we note that it is possible to use the original (D)PCG method with inexact preconditioning, since the convergence rate of the outer CG process can be maintained up to a certain accuracy for the inner iterations, see [62, 64].

8.5 Numerical Experiments

In this section, we perform some 3-D numerical experiments with stationary bubbly flow problems, which illustrate the theoretical results as obtained in the previous sections. Results of the 2-D numerical experiments can be found in [140, Sect. 7]. The numerical results are presented in terms of both the number of iterations and computing time, so that this section is basically an extension of Section 3.6.

We apply the problem setting as given in Section 1.3. Four test problems are considered with $m = 0, 1, 8, 27$ air bubbles. The corresponding geometries of these test problems can be found in Figure 1.2.

In Section 8.5.1, it is shown that the deflation method can indeed be implemented efficiently. We vary the density contrast, ϵ, the number of projection vectors, k, and the grid size, n. First, ICCG and DICCG1 are compared, followed by the comparison of DICCG1 and DICCG2. DICCG1 is based on Variant 5.2, whereas Variant 5.3 is used in DICCG2. Subsequently, we compare DICCG2 and ADICCG2 in Section 8.5.3, where ADICCG2 denotes the ADICCG method in which the Galerkin system, $Ey_2 = y_1$, is solved iteratively using ICCG. We investigate whether DICCG2 can indeed be stabilized without losing efficiency.

For each iterative process, a random starting vector and the termination criterion (2.23) with a tolerance $\delta = 10^{-8}$ are used. As a measure of the accuracy of the solutions, the exact relative residuals are also investigated in the experiments. These results are omitted below, as these residuals are comparable for ICCG and both DICCG methods in all cases.

8.5.1 Results for the Deflation Method with Efficient Implementation

The computations of this subsection are performed on a Pentium 4 (2.80 GHz) computer with a memory capacity of 1GB. The code is compiled with FORTRAN g77 on LINUX.

Results for a fixed grid size and density contrast

The results for all test cases with a grid size $n = 100^3$ and density contrast $\epsilon = 10^3$ are shown in Table 8.2.

	$m = 0$		$m = 1$		$m = 8$		$m = 27$	
Method	# It.	CPU	# It.	CPU	# It.	CPU	# It.	CPU
ICCG	170	25.2	211	31.1	291	43.0	310	46.0
DICCG1-2^3	109	20.2	206	37.5	160	29.1	275	50.4
DICCG1-5^3	56	11.3	58	11.5	72	14.2	97	19.0
DICCG1-10^3	35	8.0	36	8.5	36	8.2	60	13.0
DICCG1-20^3	22	26.5	25	27.6	22	27.2	31	29.3

Table 8.2: Convergence results for ICCG and DICCG1 for all test problems with $n = 100^3$ and $\epsilon = 10^3$. '# It' means the number of required iterations for convergence, and 'CPU' means the total computational time in seconds.

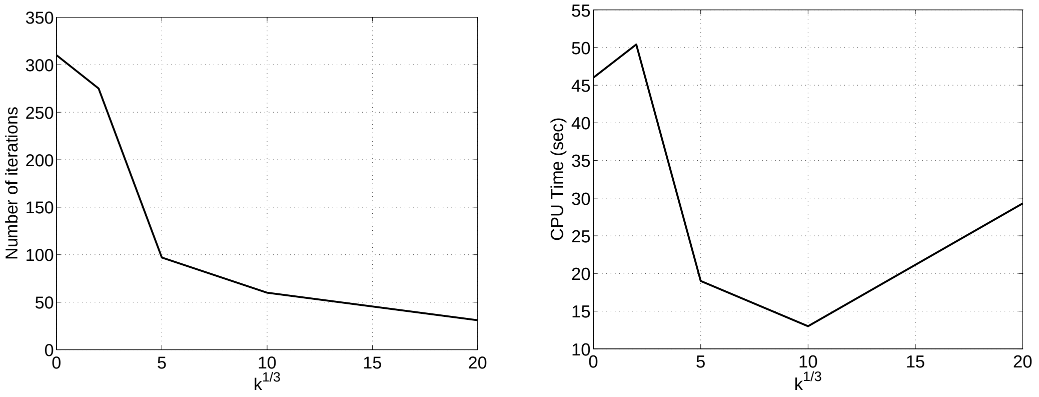
Considering the results in Table 8.2, we see that DICCG1 always requires fewer iterations when compared to ICCG. Recall that DICCG1 requires fewer iterations if k becomes larger (cf. Section 3.6.1). The optimal choice with respect to the CPU time is $k = 10^3$, i.e., DICCG1-10^3 converges most rapidly in all test cases. The improvement in the CPU time is relatively large compared to ICCG. Furthermore, one can notice that, in general, it is not always the case that more bubbles in the problem setting lead to more iterations, and, therefore, more CPU time for both ICCG and DICCG1 to converge. Namely, for DICCG1-2^3 and DICCG1-20^3, we observe that fewer iterations and CPU time are required for the test case with $m = 8$ than for $m = 1$. Finally, notice that, for large k, DICCG1 requires significant CPU time due to the increase of the computational cost for solving Galerkin systems. Hence, DICCG1 does converge with a small number of iterations for $k > 10^3$, but it requires a lot of CPU time for each iteration.

To visualize the results given in Table 8.2, we present those for the test case with $m = 27$ in Figure 8.1. From Subfigure 8.1(a), it can be observed that as small k is increased, the number of iterations of DICCG1 decreases. For large k, the benefit is smaller. Furthermore, after a peak at $k = 2^3$, the required CPU time for DICCG1 decreases until $k = 10^3$. Thereafter, the CPU time increases and DICCG1 is less efficient, see Figure 8.1(b). In Figure 8.1(c), the benefit factor for the number of iterations is depicted for each k. Obviously, the larger k, the larger the profit of DICCG1. For example, DICCG1-20^3 requires almost 10 times fewer iterations than ICCG. Finally, the benefit factor for the CPU time is depicted for each k in Figure 8.1(d). For $k = 10^3$, the maximum benefit factor is achieved. In this case, DICCG1-10^3 is around 3.5 times faster than ICCG.

Results for Varying Grid Sizes and Density Contrasts

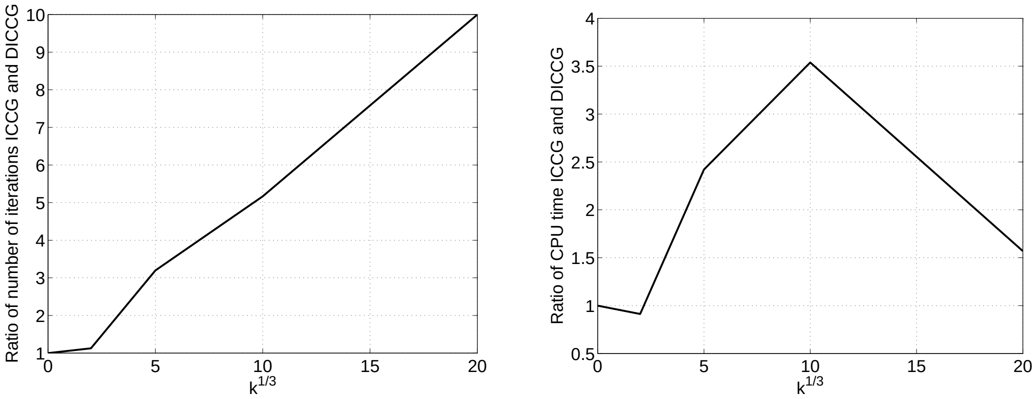
The results for the test problem with 27 bubbles and varying grid sizes are presented in Figure 8.2. Here, we use $\psi := \frac{n_x}{k_x}$ as the ratio of the grid size and the number of deflation vectors, both in one spatial direction. We investigate whether DICCG1 is scalable, i.e., whether the number of iterations of DICCG1 is equal for all k and for a fixed ψ.

From the figure, one observes immediately that for larger grid sizes, the differences



(a) Number of iterations versus $k^{\frac{1}{3}}$.

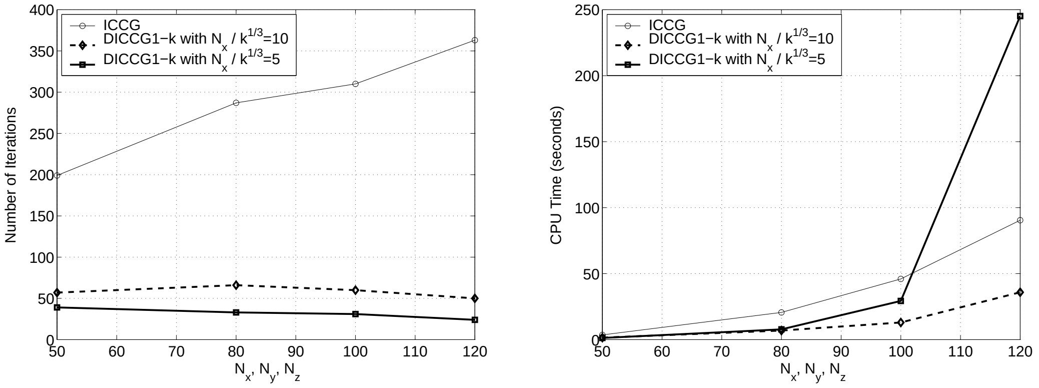
(b) CPU time versus $k^{\frac{1}{3}}$.



(c) Ratio of numbers of ICCG and DICCG1 iterations versus $k^{\frac{1}{3}}$.

(d) Ratio of CPU time required for ICCG and DICCG1 versus $k^{\frac{1}{3}}$.

Figure 8.1: Visualization of the results for the test problem with $m = 27$.



(a) Number of iterations versus grid size per direction.

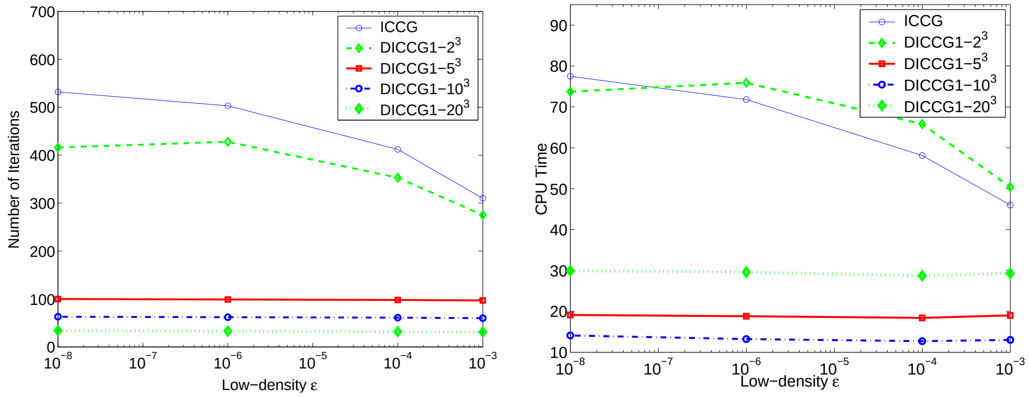
(b) CPU time versus grid size per direction.

Figure 8.2: Results for the test problem with $m = 27$ for varying grid sizes. ICCG and DICCG1 with both $\psi = 5$ and $\psi = 10$ are presented.

in performance between ICCG and DICCG1 also become significantly larger. For instance, in the case of $n = 100^3$, ICCG converges in 275 iterations and 50.4 seconds, while DICCG1-10^3 finds the solution in 60 iterations and in only 13.0 seconds. Moreover, we notice that the number of ICCG iterations grows with the grid sizes, while the number of iterations for DICCG1 for both $\psi = 5$ and $\psi = 10$ remains approximately constant. It seems that, in order to keep the number of iterations constant in DICCG1 as the grid size is increased, the number of deflation vectors must also increase, proportionally to the grid sizes. Moreover, the CPU time required for DICCG1 increases more-or-less quadratically with grid size, which is a consequence of the expensive direct solve of the Galerkin systems. This is in agreement with the theory (cf. Section 8.3.1). In the next subsection, this is remedied by using DICCG2 instead of DICCG1.

Next, after experiments with varying grid sizes, we fix the grid size as $n = 100^3$ and vary the low-density, ϵ (which is $\frac{1}{\epsilon}$). The results are presented in Figure 8.3.

From the figure, we see that DICCG1 for $k > 2^3$ hardly depends on ϵ, while ICCG becomes obviously worse when we choose a smaller ϵ. In other words, DICCG1 with $k > 2^3$ is insensitive to the density contrast in terms of both the number of iterations and the CPU time.



(a) Number of iterations versus low-density ϵ. (b) CPU time versus low-density ϵ.

Figure 8.3: Results for the test problem with $m = 27$ for varying density contrast, ϵ.

8.5.2 Comparison of DICCG1 and DICCG2

In the previous subsection, we have seen that DICCG1 is very efficient as long as $k < 20^3$. From $k = 20^3$, each iteration of DICCG1 is relatively expensive, although only a relatively low number of iterations is needed. The bottleneck is the expensive construction of the banded Cholesky decomposition of E. Direct computations with E can be avoided by using DICCG2, hopefully resulting in a fast solver for large k. In this subsection, a numerical comparison between DICCG1 and DICCG2 is carried out. Note that, since we fix $\delta_{\text{outer}} = 10^{-8}$ for all test cases, the termination tolerance, $\delta_{\text{inner}} = 10^{-10}$, should be adopted for the inner iterations in DICCG2, as mentioned in Section 8.3.2.

Some results for the test problem with $m = 27$ and varying grid sizes can be found in Figure 8.4. Similar results are found for the other test problems. The number of iterations required for both DICCG1 and DICCG2 is more-or-less equal in all test cases, which is in agreement with Theorem 5.5, so these results are omitted for convenience.

We observe in Figure 8.4 that for a relatively small number of deflation vectors, DICCG1 and DICCG2 perform approximately the same. However, for problems with relatively large k, DICCG2 is clearly more efficient. The difference between the two DICCG methods becomes significant at $k = 20^3$. In addition, we observe that, in all cases, DICCG1 achieves its optimum at $k = 10^3$, whereas the optimum of DICCG2 is achieved for $k > 10^3$. Hence, we conclude that DICCG2 is the most efficient method for $k > 10^3$. This conclusion is rather natural, but cannot be drawn beforehand. For example, unforeseen problems solving the Galerkin systems may occur, since the precise efficiency of solving these systems is not known exactly, and the consistency of these systems may be lost due to round-off errors.

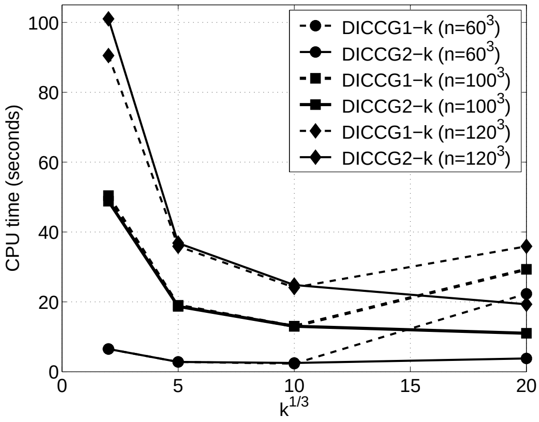


Figure 8.4: CPU time of DICCG1 and DICCG2 for the test problem with $m = 27$ and various grid sizes.

8.5.3 Comparison of DICCG2 and ADICCG2

The computations of this section are performed on an Intel Core 2 Duo (2.66 GHz) computer with a memory capacity of 8GB. The code is compiled with the Intel FORTRAN compiler, ifort, on LINUX. Some discrepancies could be observed between the results of the experiments of this subsection and the previous subsection, due to an updated code and computer environment.

We consider two test problems (cf. Figure 1.2):

- Test Problem 1: $m = 8$ bubbles, radius $s = 0.10$, grid size $n = 100^3$;
- Test Problem 2: $m = 27$ bubbles, radius $s = 0.05$, grid size $n = 150^3$,

where the density contrast is $\epsilon = 10^3$. We examine PCG and the deflation methods DICCG2 and ADICCG2 for different parameters of δ_{inner} and number of deflation vectors, k.

Results of the Experiments

The results of the experiments for the two test problems can be found in Table 8.3.

(a) Test Problem 1.

		$k = 5^3$		$k = 10^3$		$k = 20^3$		$k = 25^3$	
Method	δ_{inner}	# It.	CPU	# It.	CPU	# It.	CPU	# It.	CPU
DICCG2	10^{-10}	151	17.9	66	8.5	32	5.8	28	6.5
	10^{-8}	NC	–	NC	–	NC	–	NC	–
ADICCG2	10^{-10}	140	20.2	60	9.2	30	7.2	27	10.1
	10^{-8}	140	20.1	60	9.1	30	6.7	27	9.4
	10^{-6}	140	20.1	60	9.1	30	6.3	27	8.2
	10^{-4}	141	20.2	60	9.0	29	5.6	29	7.0
	10^{-2}	NC	–	194	28.2	NC	–	NC	–

(b) Test Problem 2.

		$k = 15^3$		$k = 25^3$		$k = 50^3$	
Method	δ_{inner}	# It.	CPU	# It.	CPU	# It.	CPU
DICCG2	10^{-10}	53	24.1	44	25.1	24	82.1
	10^{-8}	NC	–	NC	–	NC	–
ADICCG2	10^{-10}	50	27.6	41	32.5	22	130.4
	10^{-8}	50	27.2	41	30.7	22	116.0
	10^{-6}	50	26.7	42	29.3	22	86.2
	10^{-4}	52	27.4	43	27.0	24	58.2
	10^{-2}	NC	–	NC	–	NC	–

Table 8.3: Results for DICCG2 and ADICCG2 to solve $Ax = b$ with $n = 100^3$, corresponding to Test Problem 1. ICCG requires 390 iterations and 37.0 seconds for Test Problem 1 and 543 iterations and 177.6 seconds for Test Problem 2. '# It' = number of iterations of the outer process, 'CPU' = the required computing time (in seconds) including the setup time of the methods, 'NC' = no convergence within 250 iterations.

From Table 8.3, we see in all test cases that DICCG2 and ADICCG2 are always faster and require fewer iterations compared with ICCG, which confirms Theorem 3.5. Both deflation methods require approximately the same number of iterations for fixed k, which is as expected from Theorem 8.6. It can be observed that increasing the number of projection vectors, k, leads to a reduction of the number of iterations for both DICCG2 and ADICCG2. This is in agreement with Theorem 3.3. In additio, we expect that ADICCG2 is more robust than DICCG2 due to Theorem 8.6. This is indeed the case: for $\delta_{\text{inner}} \leq 10^{-8}$, DICCG2 does not converge anymore, while ADICCG2 still shows convergence, provided that $\delta_{\text{inner}} \leq 10^{-4}$. We notice that the benefit of using a larger δ_{inner} in ADICCG2 can be substantial for large k.

Furthermore, it can be noticed that there is an optimum regarding the computing time for specific k and corresponding δ_{inner}. For Test Problem 1, this is $k = 20^3$ and $\delta_{\text{inner}} = 10^{-10}$ in the case of DICCG2, whereas $k = 20^3$ and $\delta_{\text{inner}} = 10^{-4}$ are the optimal values in the case of ADICCG2. Considering Test Problem 2, the optimal choices are $k = 15^3$ and $\delta_{\text{inner}} = 10^{-10}$ for DICCG2, and $k = 25^3$ and $\delta_{\text{inner}} = 10^{-4}$ for

ADICCG2. Hence, ADICCG2 can be faster than DICCG2 using their optimal δ_{inner}.

Discussion of the Results

From the above results, it can be observed that the optimal values are $\omega = 10^{-2}$ for DICCG2 and $\omega = 10^4$ for ADICCG2 with respect to Eq. (8.7). These still hold if we vary δ_{outer}. Apparently, DICCG2 can deal with nearly zero eigenvalues as long as they are very small, so that they are treated as zero eigenvalues by the method. In addition, ADICCG2 is faster than DICCG2 in some cases, because a larger δ_{inner} can be taken, while the number of outer iterations remain approximately the same. This is rather surprising, because no extra orthogonalization steps considering the search directions or residuals are added to the iterative process in order to preserve the known orthogonality properties of the CG process.

We investigate the inner-outer iterations in more detail. Note first that each outer iteration of ADICCG2 requires two inner solves (i.e., two solves for the Galerkin systems), whereas DICCG2 only needs one (cf. Eqs. (3.14) and (8.11)). Therefore, ADICCG2 can only be more efficient, if each inner solve of this method is performed at least twice as fast as DICCG2, which is the case for sufficiently large δ_{inner}. This is illustrated in Figure 8.5, which shows a typical convergence of the residuals of an inner solve, within an outer iteration of ADICCG2-25^3. It can be observed that ADICCG2-25^3 would only be faster than DICCG2-25^3, if the inner solves are reduced from 142 to at most 71. This means that, in theory, one has to perform the inner solves with an accuracy of approximately $\delta_{\text{inner}} \leq 10^{-5}$. This can indeed be achieved for ADICCG2-25^3, see Table 8.3. Moreover, we remark that if k becomes relatively large, then E would also be very large. Then, it is inevitable to use DICCG2 or ADICCG2 instead of ICCG in order to solve $Ey_2 = y_1$ efficiently. Recall that we would then obtain an iterative method with a multilevel preconditioning, see Remark 8.4.

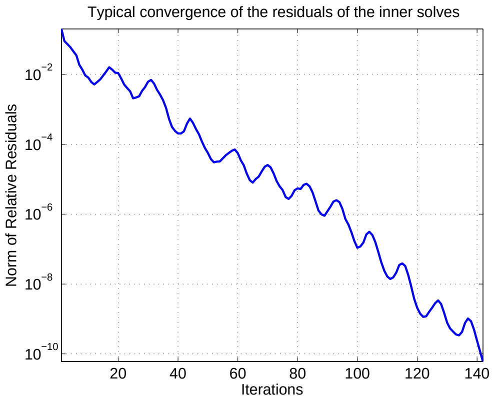
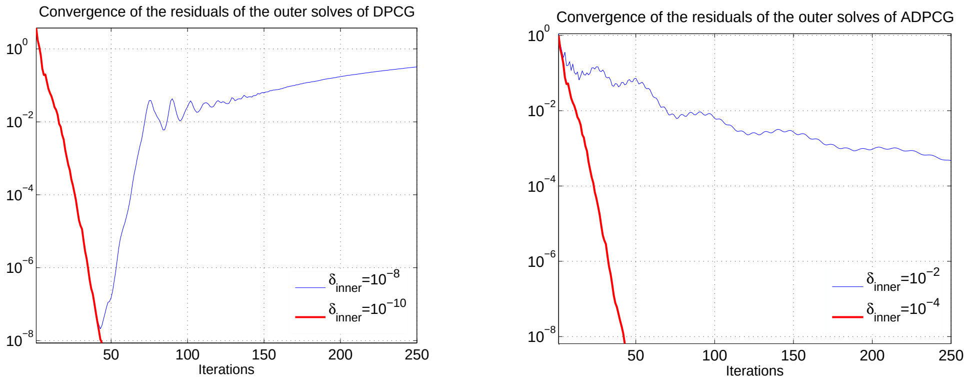


Figure 8.5: Convergence of the residuals during an inner solve at one iteration of ADICCG2-25^3 (Test Problem 2). The plots are similar for the other outer iterations of the same test case, since one applies the inaccurate solves to the Galerkin matrix, E.

Next, we examine the residuals of the outer iterations to see what happens if a method does not converge, see Figure 8.6. From the figure, we can observe that if

DICCG2 shows no convergence, it even diverges. This is in contrast to ADICCG2, whose residuals are still decreasing slowly. This is an extra advantage of ADICCG2. Although it might not be the fastest method, it gives somewhat more robust residuals in case it converges slowly.



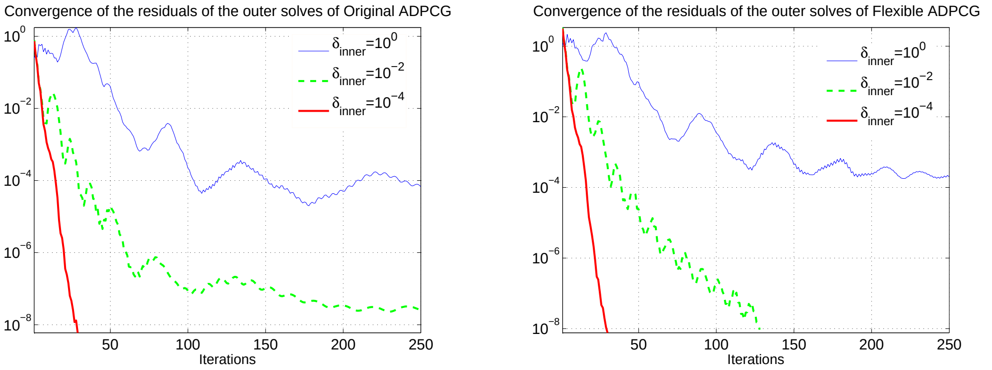
(a) DICCG2. (b) ADICCG2.

Figure 8.6: Convergence of the residuals of the outer iterations from DICCG2—25^3 and ADICCG2—25^3 (Test Problem 2).

The reason that ADICCG2 does not work for $\delta_{\text{inner}} \geq 10^{-4}$ is twofold. On the one hand, solving $Ey_2 = y_1$ with low accuracy can be interpreted as computing $y_2 = E^+ y_1$ with a strongly perturbed matrix E^+. As concluded in Section 8.4, the associated spectrum of ADICCG2 remains the same if E^+ is slightly perturbed. Large perturbations of E^+ can lead to the appearance of relatively small eigenvalues in the spectrum, which cause the slow convergence of the method. On the other hand, as mentioned in Section 8.4, the (D)PCG algorithm cannot deal with strongly varying preconditioners, because orthogonal properties of the residuals and search directions are not guaranteed anymore. This problem might be solved by using flexible (D)PCG instead of (D)PCG, but experiments show that this does not lead to better results. In Figure 8.7, the results can be found for one test case of Test Problem 1, where two variants of ADICCG2 is used: the original adapted deflation method and its flexible variant without restart or truncation strategies, denoted by ‘original ADICCG2’ and ’flexible ADICCG2’, respectively. It can be readily noticed in Figure 8.7 that the flexible variant might lead to a convergent method (see the case with $\delta_{\text{inner}} = 10^{-4}$), but it requires too many iterations to be an effective method. The situation would be even worse, if restart or truncation strategies are added.

8.6 Concluding Remarks

In Chapter 5, it is demonstrated that DICCG can be easily adapted so that it is also applicable to linear systems with a singular coefficient matrix. Additionally, the method is efficient for bubbly flow problems when measured by the required number of iterations. In this chapter, we demonstrate that the computational time is also gained



(a) Original ADICCG2.

(b) Flexible ADICCG2.

Figure 8.7: Residual plots of the outer iterations from ADICCG2-25^3 (Test Problem 1).

by applying DICCG instead of ICCG, if an efficient implementation is used.

We show that the involved Galerkin systems within the deflation method can be solved both directly and iteratively. The resulting DICCG methods are denoted by DICCG1 and DICCG2, which only differ in the implementation of the solvers for the Galerkin systems. A direct solver for these systems is adopted in DICCG1, whereas an iterative solver for the Galerkin systems is applied in DICCG2. Theoretical properties of these Galerkin systems are derived, which are of importance to DICCG2. Furthermore, insights are gained into stabilizing the deflation method, resulting in the ADICCG2 (A-DEF2) method. In this method, the inner iterations can be solved rather inaccurately, while the number of outer iterations remain approximately the same.

Several 3-D numerical experiments based on bubbly flow problems are performed in order to test the efficiency of DICCG1 and DICCG2. For a relatively small number of deflation vectors, DICCG1 performs very well, but DICCG2 is more efficient for a larger grid size and/or number of deflation vectors. Compared with ICCG, both methods significantly reduce the computational cost in all test cases, especially for large problems. Additionally, they are insensitive to density contrasts, while ICCG has difficulties for large contrasts. Furthermore, we show that the DICCG methods are scalable in terms of iterations and CPU time, as long as the number of deflation vectors is chosen proportionally to the grid size. Moreover, numerical experiments illustrate that the DICCG2 can indeed be stabilized without losing much efficiency. The resulting ADICCG2 method can be more efficient than DICCG2 for some test cases. In order to improve the efficiency of the deflation methods considered in this chapter, multigrid-like components could be incorporated. Moreover, these methods can also be compared to well-known multigrid methods based on their typical and optimized parameters, which is the main topic of the next chapter.

Chapter 9

Comparison of Deflation and Multigrid with Typical Parameters

9.1 Introduction

In this chapter, we compare deflation and multigrid methods based on their typical optimized parameters applied to linear systems coming from bubbly flow problems. In the previous chapters, we have seen that the deflation method (DPCG) is a 2L-PCG method that offers one attractive possibility for the efficient solution of the linear system (see Eq. (1.1)),

$$Ax = b, \tag{9.1}$$

which takes the form of a Poisson equation (also called 'pressure-correction equation') with discontinuous coefficients and Neumann boundary conditions, see Eq. (1.3).

Recall from Chapters 6 and 7 that another option for the efficient solution of the pressure-correction equation is the use of multigrid (MG) techniques. We have shown that, algebraically, DPCG and PCG with a MG preconditioner are strongly related to each other. These results are only valid when the same set of parameters are taken in both corresponding algorithms. However, the preferred choices for these components are quite different between the two methods, and, in addition, they have a different meaning and background. The fine-grid smoother or preconditioner is usually chosen to give effective treatment of certain modes of error. A complementary space is defined, in terms of a set of deflation vectors or the range of the multigrid interpolation operator, and an optimal correction over this space is computed. However, while deflation techniques are typically based on a strong fine-scale preconditioner (such as an IC(0) preconditioner) in combination with a coarse-scale correction over a very small space, multigrid techniques typically make use of a rather weak fine-scale smoother (e.g., a Jacobi or Gauss-Seidel iteration) in combination with a coarse-scale correction over a space that is a large fraction of the fine-scale problem size. Furthermore, the treatment of the linear systems associated with the coarse scale are handled differently. Deflation techniques typically solve these systems using a direct or iterative method, whereas a recursive procedure is used in the multigrid approach.

The black box multigrid technique, first introduced in [3], uses geometrically structured coarse grids in combination with an interpolation operator designed to account for the effects of jumps in the diffusion coefficients to achieve fast multigrid convergence in many situations [12, 37, 39]. Algebraic multigrid, or AMG, is also known to be effective for elliptic problems with jumps in their coefficients [118, 129], achieving this efficiency by tailoring both the coarse-grid structure and interpolation operator to account for the jumps in the coefficients. While both of these solvers are applied successfully in many cases, the modelling of bubbly flows provides some unique challenges. In particular, in simulations with bubbles that appear at the finest resolution of the grid, these techniques may encounter difficulties in treating such small-scale effects through adapting the coarse-scale models. In these cases, we find in this chapter that using the above multigrid algorithms as preconditioners within PCG easily restores their optimal convergence behavior at a minimal extra cost.

The use of MG-preconditioned CG within a pressure-correction method is not new. Indeed, multigrid was first considered for use as a preconditioner for discontinuous-coefficient problems very early in its history, see [18, 78]. Under certain symmetry assumptions on relaxation and the coarse-grid operators, Tatabe demonstrated that multigrid always defines a positive-definite preconditioner (regardless of the number of pre- and post-relaxations) and, so, multigrid is acceptable as a preconditioner for PCG [149]. In the fluid dynamics literature, Tatabe's MGCG method has been adopted for the solution of the pressure-correction equation for variable-density flows [115, 130]. Similarly, the MUDPACK software package [1] has also been used for solving these equations [49, 50, 152], but this has been found to not offer robust performance to the large density jumps that appear in realistic simulations [50]. In contrast to many of these techniques, the multigrid algorithms considered here include two important features, Galerkin coarsening and operator-induced interpolation. Galerkin coarsening creates the multigrid coarse-grid operators using matrix products that restrict the fine-grid operators onto the range of interpolation. This greatly simplifies the task of creating a consistent coarse-grid correction process for problems with density variations that are not resolved on the coarse scale. Operator-induced interpolation techniques further improve this approach by building interpolation operators tuned towards capturing the fine-scale modes that are slow to be reduced by simple relaxation on the variable-coefficient problem.

In this chapter, we make a detailed comparison between deflation and multigrid methods with their own typical parameters for bubbly flow problems. While the application of the deflation method in bubbly flows is examined in the previous chapters, the use of advanced multigrid methods for these flow simulations has, to our knowledge, not been previously considered in the literature. As well, the linear system (9.1) that arises in 3-D two-phase flows offers a good opportunity for comparison of these two families of solvers. In contrast to previous theoretical comparisons, as performed in Chapters 6 and 7, we focus here on evaluating each solver using its most advantageous selection of options.

The remainder of this chapter is organized as follows. Section 9.2 presents the details of the deflation and especially the multigrid methods considered here. Some implementation details of the two families of solvers are compared in Section 9.3. Then, Section 9.4 shows a numerical comparison of the different solvers for 3-D stationary bubbly flows. Concluding remarks are presented in Section 9.5.

Remark 9.1. *Domain decomposition methods, such as the balancing Neumann-Neumann method, with their typical and optimal parameters are not considered in this chapter, because these methods are advantageous in especially a parallel environment. Since this chapter is restricted to computations on a sequential computer, DDM is excluded from the comparison.*

9.2 Numerical Methods

We know from Section 1.3 that the solution of the pressure-correction equation within operator-splitting approaches has long been recognized as a computational bottleneck in fluid flow simulation. In the case of single-phase fluids, a common approach to overcoming this bottleneck is the use of multigrid solvers for this equation [151]. Standard geometric multigrid techniques offer optimal-scaling solution properties for the pressure-correction equation in a single-phase fluid. For two-phase fluids, however, large differences in the fluid densities can lead to dramatic deterioration in the multigrid performance. In this chapter, we consider alternate approaches to solving the pressure-correction equation that do not exhibit the same sensitivity to jumps in the material properties. We focus on solving Eq. (9.1), which is a discretization of the pressure-correction equation.

9.2.1 Deflation Approach

Recall that the linear system that is solved in ICCG is (see Eq. (2.19))

$$M^{-1}Ax = M^{-1}b,$$

where M^{-1} denotes the IC(0) preconditioner. To improve the performance of ICCG, we include a second operation in the preconditioner, so that we solve the following system (see Eq. (3.14)):

$$M^{-1}PA\tilde{x} = M^{-1}Pb, \tag{9.2}$$

where $P := I - AZE^{-1}Z^T$ is the deflation matrix with $Z := [z_1 \ z_2 \ \cdots \ z_k]$. The resulting method is called DICCG. In this chapter, the vectors $\{z_i\}$ are subdomain deflation vectors (see Section 4.2.3). If k becomes large, the Galerkin systems involving E become more costly to solve, and, in particular, the use of standard sparse direct solvers may be inefficient. Instead, an iterative solver can be adopted to deal with these Galerkin systems. In this chapter, the Galerkin systems in DICCG are, themselves, solved using ICCG. The resulting method is known as DICCG2, see Section 8.3. As mentioned in that section, as k increases, even standard iterative solution of the coarse-level systems can become quite expensive, and deflation-like techniques should also be incorporated into the coarse-level ICCG algorithm, leading to a recursive multi-level deflation method [48]. This is, however, not considered in the results presented here.

9.2.2 Multigrid Approaches

Since Eq. (9.1) closely resembles the linear system associated with a diffusion equation, another class of techniques to consider is the family of multigrid methods. In the case of single-phase flow, in particular, the pressure-correction equation (see Eq. (1.3)) reduces to the case of a simple Poisson equation with a constant density, for which geometric multigrid methods are known to provide optimal solution techniques [151, 178]. For two-phase fluids, more complicated multigrid techniques are necessary to achieve optimal performance; such techniques are well-known in other fields. Here, we present the details of these methods, and their specialization to solving (9.1).

Just as deflation methods use a correction over a small subspace to account for the deficiencies of a traditional preconditioner, all multigrid methods combine the use of a coarse-scale (or coarse-grid) correction process that is aimed at correcting modes that a fine-scale iteration (or smoothing) is slow to resolve. Different multigrid methods differ in the choices made in these two processes, particularly in the details of how the coarse-scale space is chosen and how that correction is computed. Here, we concentrate on multigrid methods that make use of simple pointwise smoothers, such as the Jacobi or Gauss-Seidel iterations, and consider four different choices for the coarse-grid correction procedure. A brief introduction into classical multigrid techniques is presented in the following subsections; we refer to, e.g., [151, 178] for more details.

Geometric Multigrid

For the case of single-phase flows discretized on a regular grid, the use of geometric multigrid techniques has long been studied (cf. [151, 178]). These methods are based on the realization that, while simple iterations, such as weighted-Jacobi and Gauss-Seidel, do not efficiently resolve all modes of the solution, they do quickly and effectively damp a large subspace of errors for a large class of matrices, including those considered here. In particular, for single-phase flow (constant density), the errors that are not quickly damped by these simple iterations are dominated by so-called ‘smooth’ error modes; errors that vary slowly between neighboring grid points. It is, thus, the job of the coarse-grid correction process to attenuate exactly these modes.

An important difference between the coarse-grid correction processes in multigrid and deflation techniques is the size of the subspace employed for coarse-grid correction. While deflation aims for a correction over a much smaller subspace than the fine-scale problem size, the size of the coarse-grid problem in a multigrid method is specifically chosen to be a relatively large fraction of the fine-grid size; typical coarsening rates are by a factor of 2 or 3 in each dimension. Such slow coarsening is justified by considering the convergence behavior of the complementary stationary iteration; for any fixed reduction tolerance, the number of error modes which are reduced in magnitude by a factor larger than that tolerance (i.e., the number of slowly converging modes) increases with the size of the fine-grid matrix. Thus, to achieve convergence that is truly independent of problem size using classical stationary iterations requires that the size of the coarse-grid problem always remains a fixed fraction of the fine-grid problem size.

Here, we consider a geometric multigrid approach using Gauss-Seidel smoothing. For maximum efficiency, the algorithm is specialized to grids with the number of grid points in each dimension of the form $\alpha 2^{\beta}$, where $\alpha, \beta \in \mathbb{N}$ with typically $\alpha = 1$ or $\alpha = 3$, so that the coarse grid may be chosen by reducing the grid size by a factor of 2 in each direction at all levels. Interpolation is trilinear cell-centered interpolation, while restriction is taken as cell-centered piece-wise constant restriction [98]. Smoothing and residuals on the coarse grid are realized using direct discretization of the fine-scale homogeneous problem on the coarse grid. This is possible because of the assumption of constant-density for the single-phase (incompressible) flow; for the two-phase flows considered here, more sophisticated techniques, described below, are needed to discretize, on the coarse scale, a fine-scale flow that may have phase boundaries that cannot be represented on the coarse grid.

While geometric multigrid techniques often yield the fastest solvers for the constant-coefficient version of the pressure-correction equation, several complications arise in extending these techniques to the case of variable density. The properties of simple smoothing techniques, such as Jacobi or Gauss-Seidel, are highly dependent on the density; in the case of nonconstant density, the dominant errors after smoothing may exhibit sharp transitions and/or cusps, which must be accounted for in the coarsening process. Furthermore, if the variations in density have fine-scale features (as we expect for bubbly flows), it may not be clear how best to represent the equations on the coarser grid, as needed in the multigrid process. In the following subsections, we discuss several approaches for overcoming these obstacles.

Another approach to overcome the above discussed complications is to use the geometric multigrid method as a preconditioner for the PCG iteration. A good solver, such as geometric multigrid, for the constant-coefficient case is expected to make a good preconditioner for the variable density case, so long as the density contrast is not too significant. Results for this approach are reported in Section 9.4 as method GMG-CG. All of the other multigrid approaches that we develop here may be applied both as a standalone solver and as a preconditioner for PCG. This preconditioner is basically the multigrid V(1,1)-cycle preconditioner as discussed in Chapter 7. In what follows, we discuss only the case of these techniques being used as standalone solvers; see Section 9.2.2.

Galerkin Coarsening

While GMG-CG (and geometric multigrid in general) performs well when the density contrast is small, its performance suffers greatly when problems with large density contrasts are considered (as shown in Section 9.4). Improving the multigrid performance requires improvement in one (or both) of the multigrid components, smoothing or coarse-grid correction. In GMG-CG, however, direct smoothing on (1.3) is replaced by smoothing on the constant-coefficient problem as a preconditioner for the variable-density problem of interest. Thus, a first step in improving the performance of GMG-CG would be to replace the smoothing on the homogeneous problem with that on the real problem of interest.

Making the above improvement on the fine scale is simple to implement; both the constant-density and variable-density problems are well-defined on the fine scale, and it is relatively simple to replace smoothing on one with smoothing on the other. On coarse scales, however, we have no direct representation of the fine-scale problem, unless it is possible to represent the variation in the density naturally on the coarse scale. For many flows of interest, this is clearly the case for all coarse scales in a multigrid hierarchy. Thus, we need some indirect way to account for fine-scale variations in the density directly in smoothing on the coarse scales.

There are many possible ways to create a coarse-scale model with a fine-scale density distribution; the problem of numerical homogenization, or upscaling, is studied in many disciplines, see, e.g., [176]. While these techniques focus on defining an effective density coefficient that can be naturally represented on the coarse scale only, we instead focus on the multigrid point of view that the primary purpose of smoothing on the coarse scales is not to represent the flow on those scales but, rather, to compute an appropriate correction to the errors in the fluid pressures on the fine scale.

Using the coarse-grid models to improve a fine-grid approximation to the pressure naturally leads to the question of how good a correction is possible from a coarse grid. Mathematically, we consider a fixed coarse grid and interpolation matrix, Z, that maps from the coarse grid to the fine grid. Asking for the best possible correction from the coarse grid, i.e., the best correction in the range of Z, means that we wish to minimize some norm of the error, e, in our approximation, x_{j+1}, to the solution, x, that satisfies (9.1). Writing the corrected approximation as $\hat{x}_{j+1} = x_{j+1} + Zy$, for some vector y, this means that we wish to minimize

$$\|\hat{e}\| = \|x - (x_{j+1} + Zy)\| = \|e - Zy\|.$$

Both the minimum value and the coarse-grid vector, y, for which the minimum is achieved depend strongly on the norm chosen for the minimization. Choosing the A-norm implies that the optimal choice for y satisfies

$$(Z^T AZ)y = Z^T Ae = Z^T(b - Ax_{j+1}).$$

That is, the best possible coarse-grid correction for a fixed choice of the multigrid interpolation matrix, Z, may be expressed in terms of a Galerkin matrix, $E := Z^T AZ$, and restriction of the fine-grid residual, $r_{j+1} := b - Ax_{j+1}$, using Z^T as the restriction map. This choice of coarse-grid and restriction operators is known in the multigrid literature as Galerkin coarsening [106, 151, 178], because of its close relationship to Galerkin finite elements.

A first generalization of GMG-CG is then to consider the multigrid method with interpolation, Z, given as in GMG-CG, but with smoothing on all levels replaced by Gauss-Seidel smoothing on the finest-grid matrix and its Galerkin restrictions. This technique, which we denote by CCMG, was first proposed for problems similar to (1.3) in [177] and later studied in [79, 163].

The Black Box Multigrid Method

While CCMG offers a great improvement over GMG-PCG in terms of its scalability, its performance still degrades as the density contrast increases, see Table 9.1 in Section 9.4. As CCMG arose through improvements to the smoothing phase in GMG-PCG, we now consider the role of interpolation in the performance of CCMG. Of particular importance in achieving consistent multigrid performance regardless of the density contrast or configuration of the flow is the principle of complementarity; as multigrid methods aim to reduce errors through two distinct processes, smoothing and coarse-grid correction, optimal multigrid performance can only occur when these processes are appropriately complementary.

While we could aim to improve the performance of CCMG by making further improvements to the smoothing routine, such improvements often dramatically increase the cost of the iteration. Instead, we aim to improve the multigrid performance by making a different choice for the interpolation matrix, Z, to better complement the performance of lexicographical Gauss-Seidel smoothing. It has long been recognized that for problems with discontinuous coefficients, such as the pressure-correction equation, the errors left after smoothing are not smooth, as in the case of constant-coefficient problems [3]. Thus, while coarse-grid correction with a fixed interpolation operator, such as those analyzed in [98] and discussed above, may be used to effectively complement smoothing for constant-coefficient problems, they are less appropriate when the problem contains large jumps in its coefficients.

The solution to the problem with large jumps in coefficients, first discussed in [3] and further developed in [37, 38], is to allow the coefficients of the interpolation matrix, Z, to depend on the coefficients of A. Such an operator-induced interpolation is better able to reflect the slow-to-converge errors of smoothing as these errors are, themselves, dependent on the variation in A. The technique of the Black Box Multigrid Method [37], also denoted by BoxMG, defines the coefficients of interpolation to a fine-grid point by combining the entries of the matrix in the rows corresponding to the grid point and its graph neighborhood.

In 3-D, the BoxMG algorithm assumes that the fine-grid matrix comes from the discretization of an equation such as (1.3) on a logically rectangular grid, see [38]. The fine-grid operator is then assumed to have at most a 27-point connectivity structure; for each grid point, connection is only allowed to grid points that reside in grid point neighboring (possibly only at corners) that in which the grid point lies. The coarse grid is constructed by removing every other plane of grid points in each direction, in contrast to the coarsening used in GMG and CCMG, where finite volumes were aggregated in pairs in each direction. Interpolation in BoxMG then falls into four categories: fine-grid points may be themselves coarse-grid points (as the coarse grid is embedded in the fine grid), they may lie on the line segment connecting two coarse-grid points, they may lie in the same plane as four coarse-grid points, at the center of the square defined by these grid points, or they may lie in a plane with no coarse-grid points, at the center of the cube defined by 8 coarse-grid points.

Interpolation of corrections to embedded coarse-grid points is always done using injection, as the errors at these grid points are directly represented on the coarse grid. For fine-grid points lying between two coarse-grid points, the interpolation weights are defined by first adding the matrix entries in the row of A corresponding to the grid point along the planes orthogonal to the connecting line, collapsing the 27-point stencil to a 3-point stencil joining these three grid points. The correction to the grid point is then computed by setting this 3-point stencil to zero, substituting in the injected corrections at the coarse-grid points and solving for the correction at the fine-grid points. A similar approach is used for fine-grid points lying at the center of a square in the plane of four coarse-grid points, however first the four other neighboring grid points in that plane are resolved using the first approach. In this way, the 27-point stencil needs only be collapsed to a 9-point stencil, which can be treated using the previously computed corrections as in the 3-point stencil case. Finally, the correction to the grid point at the center of the cube defined by 8 coarse-grid points may be calculated directly, by satisfying the 27-point stencil at this grid point using the corrections computed at all of its neighbors, see [12].

The Galerkin matrix, E, in BoxMG is again defined using a Galerkin coarsening, with Z defined as described above. It can be verified that, if the fine-grid stencil has its nonzero connections confined to within a 27-point stencil pattern, then the coarse-grid operator also has 27-point connectivity. Thus, BoxMG may be applied recursively to define a full multigrid hierarchy. While there are many ways to use knowledge of the discrete operator, or of the density distribution itself, to define the multigrid hierarchy, the approach taken in BoxMG has been shown to be successful for a wide variety of problems. In [101], it is shown that one of the reasons for the success of BoxMG is that defining interpolation in this way approximately preserves the continuity of the normal flux, $\left(\frac{1}{\rho}\nabla p\right) \cdot \mathbf{n}$, across an interface with a jump in the density. Thus, BoxMG can be thought of as combining effective multigrid principles with useful physical insight in achieving a stable and efficient solution algorithm.

Algebraic Multigrid

With the early papers on BoxMG [3, 12, 37], it was recognized that the combination of operator-induced interpolation and Galerkin coarsening can lead to very robust methods for a wide class of problems. The algebraic multigrid method [22, 118], or AMG, is an algorithm based on a different implementation of the same principles, but which can be applied to an even larger class of problems. In particular, AMG can be applied to problems without a regular grid structure and allows for the choice of unstructured coarse grids regardless of the fine-grid operator structure.

The central idea of AMG is that all components of the coarse-grid correction cycle should be determined by the properties of the fine-grid operator. The first step in coarsening is then to determine whether two grid points that are connected in the fine-grid operator are connected in a significant way, where grid points i and j are said to be connected if $a_{ij} \neq 0$. Each grid point, i, is said to strongly depend on any of its neighboring grid points for which a_{ij} is of similar size as the largest entry in row i. For M-matrices, such as the coefficient matrix of Eq. (9.1), we define the set of grid points that i strongly depends upon as

$$S_i = \left\{ j : -a_{ij} \geq \theta \cdot \max_{J \neq i} \{-a_{iJ}\} \right\},$$

for some suitable θ, $0 < \theta \leq 1$. Once these strong connections are identified, a coarse grid is formed by taking a maximal independent set of the graph created by the set of edges, $\{a_{ij}\}$, where $j \in S_i$.

To define interpolation in AMG, a similar strategy is used to collapse connections between grid points that appear only on the fine grid and define an interpolation operator. Choosing the coarse grid through the maximal independent subset algorithm described above implies that the coarse-grid points are embedded in the fine grid. For any fine-grid point, i, that is not also a coarse-grid point, interpolation can be defined by collapsing the connections from i to other fine-grid points, j, based on their common coarse-grid neighbors. Unlike BoxMG, this is done without using any intuition into the couplings involved; the elimination of these fine-fine connections is a purely algebraic operation. Each fine-fine connection is replaced using a weighted average of the coefficients connecting the fine-grid point, j, to the coarse-grid neighbors of grid point i; see [118, 129] for details.

Because both the choice of the coarse grid and the interpolation operator in AMG are determined based on the fine-grid operator, we expect AMG to have convergence properties similar to, or possibly even better than, those of BoxMG. However, the price paid in AMG for this robustness is the use of completely unstructured matrix and vector data structures, as a result of the unstructured grid hierarchy. Additionally, the cost of the additional operations to compute the coarse grid (and, in fact, a more expensive computation of the interpolation weights) makes AMG an expensive alternative in situations where BoxMG (or CCMG or GMG) is expected to perform well. Nevertheless, AMG is often the method of choice in commercial codes where robustness is considered more important than achieving the smallest possible solution time. Indeed, in CFD, AMG solvers have been recognized as an important tool for achieving efficient solution in a wide variety of flow regimes, see [116].

On the Need for Preconditioning

While the multigrid methods discussed here are typically considered as standalone solvers, it is sometimes useful to also consider them as two-level preconditioners for 2L-PCG. These preconditioners take the form of a typical multigrid V(1,1)-cycle, where they only differ in the choices of the parameters, see Chapter 7. Both BoxMG and

AMG aim to directly treat the fine-scale structure of the density field through the use of operator-dependent interpolation algorithms. It is possible, however, that operator-dependent interpolation alone is not sufficient to yield an optimal solution algorithm in all situations.

It is quite natural for AMG and BoxMG to not interpolate significantly across bubble boundaries, see also [88, Sect. 3.3.5]. A intuitive requirement from this point of view would be to require, for each fine-scale bubble, a sufficient number of coarse-grid points to lie within the bubble, to allow for an accurate computation of a coarse-grid correction for all grid points within the bubble. In a real-life simulation, however, this may not be practical, due to bubbles and droplets that may only be resolved at the size of a single grid point on the fine scale. In Section 9.4, we see that the number of iterations for both BoxMG and AMG without the use of a Krylov wrapper increase as the number of bubbles grows. While such an increase is not dramatic, it can easily be attenuated by the use of these multigrid methods as a preconditioner for 2L-PCG; in this case, the multigrid method gives good convergence for almost all types of errors, and the CG acceleration effectively resolves the few error modes associated with these small bubbles. In Section 9.4, we denote the 2L-PCG methods with CCMG, BoxMG, and AMG preconditioners for 2L-PCG by CCMG-CG, BoxMG-CG, and AMG-CG, respectively.

9.3 Implementation and Computational Cost

In the numerical experiments of Section 9.4, we make use of standard implementations of the methods discussed above, when available. In this section, we discuss the relative costs of these techniques, as well as their scalability for large problem sizes. Relative to CG by itself, or even to ICCG, all of the other methods considered here have a larger cost per iteration. Their utility lies in the significant reduction in iterations possible using a multilevel technique as compared to a single-level method, such as ICCG. Here, we stress the details of the relative costs of a single cycle of these algorithms, as a prelude to the numerical results in Section 9.4.

9.3.1 Cost of Deflation

The computational cost of the deflation method has already been discussed in Chapter 8 and Appendix E. Recall that the setup of the deflation method is rather cheap, since Z may be constructed independently of the problem matrix. Furthermore, it is not necessary to store Z explicitly in memory; AZ and E may be computed beforehand. In the 3-D case, construction of AZ and E can each be done in $\mathcal{O}(n^{\frac{2}{3}}k^{\frac{1}{3}})$ flops. Moreover, DICCG needs only one more step than ICCG at each iteration. The additional cost for the deflation step is $\mathcal{O}(n+\theta)$ flops, where θ is the number of flops required for the inner solves involving E. Using ICCG as inner iterative solver, each inner iteration costs $\mathcal{O}(k)$ flops, and at most $\mathcal{O}(k^{\frac{1}{3}})$ iterations are required to achieve sufficient accuracy in the inner solve, leading to $\theta = \mathcal{O}(k^{\frac{4}{3}})$ operations. Note that, because AZ can be precomputed and is much sparser than A, there are no additional matrix-vector multiplications with A required at each iteration of DICCG.

Remark 9.2. *While, in principle, DICCG may be applied in an unstructured manner, the implementation considered here is based on the assumption of a rectangular tensor-product grid. This allows significant savings in both the storage and the computational cost required by the iteration, as structured matrix data structures may be used in place of the more general (and more costly) storage required by an unstructured implementation, see Chapter 8 and Appendix E. In this sense, DICCG tested here is more comparable to a geometric multigrid approach than to AMG, although the DICCG algorithm could be applied in the same unstructured settings as AMG.*

9.3.2 Cost of Multigrid

The relative costs of the multigrid methods studied here, in principle, increase with the complexity of the algorithm. Geometric multigrid can easily be implemented in a very efficient manner. In fact, because GMG-CG uses smoothing only on the homogeneous problem, it may be implemented in a fully matrix-free manner. The same stencil is applied everywhere on each level, up to boundary conditions, and the simple transfer operators can be implemented again with constant stencils away from the boundaries. Thus, the true computational cost of a single iteration of GMG-CG is much smaller than that of a method with similar number of operations, because of the optimization possible under the assumption of constant coefficients in the operator.

While multigrid methods use much finer coarse grids than typical deflation methods, their recursive treatment of these grids leads to an overall cost per iteration that remains $\mathcal{O}(n)$. Consider, for example, the cost of a single GMG V(1,1)-cycle (that is, the cost of a single preconditioning step in GMG-CG). On each level of the multigrid hierarchy, two smoothing sweeps are performed at the appropriate resolution. On every grid, the cost of these smoothing sweeps is directly proportional to the size of that grid. Thus, $\mathcal{O}(n)$ operations are required for each sweep on the finest grid, $\mathcal{O}(\frac{n}{8})$ operations are required for each sweep on the first coarse grid (which has size $\frac{n}{8}$), and $\mathcal{O}(\frac{n}{64})$ operations per sweep are required on the next coarsest grid, etc. Overall, the total number of operations required to perform two smoothing sweeps on all levels is then bounded by $\frac{16}{7}$ times the cost of a single sweep of smoothing on the finest level, which is $\mathcal{O}(n)$. Similarly, the additional storage requirements for GMG-CG can also be bounded by a small constant times the fine-scale, $\mathcal{O}(n)$ storage requirement for CG.

CCMG adds the cost of structured matrix storage and operations on all levels, as well as that of the Galerkin product in the setup stage of the algorithm. For the numerical results presented in Section 9.4, we use 64-point interpolation and restriction stencils, corresponding to bilinear interpolation and its adjoint for restriction. This results in some growth in the stencil size on coarser grids, but this growth can be easily quantified and still included in a structured-grid matrix data structure. Alternately, lower-order interpolation and restriction may be used, as in [79], to control the growth of the stencil on coarse grids. While these costs somewhat increase the cost of CCMG relative to GMG, the overall cost per cycle for CCMG remains $\mathcal{O}(n)$, as the added storage and computational costs on each level are bounded by a small constant times the number of unknowns on that level.

Here, we focus on an optimal implementation of the CCMG algorithm within the AMG code, with a fixed choice of coarse grids and transfer operators. The implementation does not use the most efficient data structures and the reported CPU times in Section 9.4 are much larger than strictly necessary for CCMG, although we stress that the iteration counts and final residuals are the same for this implementation as they would be for a more efficient one. In practice, the setup costs for CCMG should be cheaper than those for BoxMG, due to the fixed interpolation pattern. However, the cost of a single iteration of CCMG is expected to require more operations than a single iteration of BoxMG, as BoxMG uses a 27-point interpolation stencil, compared to CCMG's 64-point stencil, leading to denser coarse-grid matrices for CCMG when compared with BoxMG. As will be seen in Section 9.4, the iteration counts for CCMG clearly scale less well than those for DICCG, BoxMG, and AMG for the problems considered here, particularly as the density contrast increases (Table 9.1).

In contrast to GMG and CCMG, BoxMG is not based on cell-centered data structures. Instead, BoxMG is, primarily, a node-based code; however, its robustness leads to successful results for our cell-centered discretization as well. The added cost in BoxMG is primarily in the setup phase of the algorithm, where coefficients of the operator on each level are used in determining interpolation to that level. Coarsening in BoxMG is also slightly different than that in CCMG and GMG, as it naturally takes nodal fine grids of $2^{\beta}+1$ grid points in each dimension into nodal coarse grids with $2^{\beta-1}+1$ grid points; however, because of the use of operator-induced interpolation, BoxMG is also able to successfully solve problems with grids of arbitrary sizes, as will be seen in Section 9.4, while maintaining the typical $\mathcal{O}(n)$ complexity per multigrid cycle.

Finally, AMG has the highest cost per iteration of the multigrid (and other) approaches considered here, because of the added cost of its unstructured grid processing and data structures, as well as a significant setup cost. This is to be expected; our choice of a structured-grid discretization naturally suggests that the best efficiency is obtained with a structured-grid solver. Results for AMG are included to answer two interesting questions. First, it is interesting to see how much of a performance loss is seen with these unstructured data structures; results in Section 9.4 suggest that AMG is typically a factor of 10 to 15 times slower than BoxMG, when used as a preconditioner. Secondly, we note that while the CPU-time cost of AMG is significantly greater than that of the multilevel structured-grid codes (indeed, it is sometimes greater than that of simple ICCG), the iteration counts for AMG-CG are quite good (typically comparable to those of BoxMG). This demonstrates the scalability and robustness seen with AMG, while highlighting the advantages of using a structured-grid algorithm when possible.

9.3.3 Singularity of Coefficient Matrix

We recall that the coefficient matrix, A, is singular in (9.1). A nonunique solution, x, always exists, because we know that the system is consistent. However, extra care should be taken in the implementation of the methods we compare. Matrix E is often singular, due to the singularity of A and the construction of Z. In this case, E^{-1} does not exist, and instead the pseudo-inverse, E^{+} should be used in the operator P, see Chapters 5 and 8. The iteration process, where the deflation matrix is based on E^{+}, does not cause any difficulties in DICCG, since the corresponding systems are consistent, see Theorem 8.2.

The multigrid iterations are similarly insensitive to the singularity on all levels but the coarsest, as only iterative approaches are used in reducing errors at all other levels. On the coarsest level, the known form of the null space of E allows a simple perturbation and projection technique to be used in the direct solve of this system; see Remark 5.4 and [39] for details.

9.3.4 Parallelization

While the tests performed in Section 9.4 are all done in a serial computing environment where, because of the efficiency seen in the best approaches, problems of up to 8 million degrees of freedom are easily handled, it is important to stress that this was done for convenience alone and not because of any inherent serial nature of the algorithms considered. Indeed, much effort has been invested in the parallelization of exactly the algorithms considered here. Parallelization of deflation solvers is considered in Appendix F and [56], where it is shown that, with a sensible alignment of the subdomains and processor boundaries, deflation applied to block-IC preconditioners can be implemented with limited increase in cost over that of the parallel matrix-vector multiplies already required by block-incomplete Cholesky PCG.

Parallelization of standard multigrid methods has been considered for many problems and many architectures; see, for example, [52, 76]. Similarly, parallel implementations exist for BoxMG [7] and AMG [71]. Because of the use of pointwise smoothers in the smoothing step, virtually no parallel communication is necessary when block-Jacobi smoothing is used in place of pointwise Gauss-Seidel. In AMG, parallel communication and inherently serial algorithms is a well-studied issue, particularly with respect to the choice of coarse grid [2].

9.3.5 Implementation

While we stress that there is nothing ‘out of the ordinary’ in the multigrid implementations considered here, it must be acknowledged that there have been many less-successful attempts to apply multigrid to these problems. Therein lies the attractiveness of the DICCG method; given an existing code, with existing data structures and single-level preconditioners, it is relatively simple to add an effective deflation step to the existing pressure solver. In contrast, use of the best-available multigrid codes (as considered here) requires some translation of the discrete problem into data structures that are more natural for multigrid treatment. This trade-off is at the root of the questions investigated here. While a greater investment of programming effort may be necessary to implement a robust, efficient multigrid solver, such as BoxMG, this effort appears to pay off in the reduced computing times seen in the following numerical results.

9.4 Numerical Experiments

In this section, we perform some numerical experiments with 3-D stationary bubbly flow problems as described in Section 1.3, where the presented methods in this chapter are compared. The geometry of some test cases can be found in Figure 1.2. The computations are performed on an Intel Core 2 Duo (2.66 GHz) computer with a memory capacity of 8GB. The code is compiled with the Intel FORTRAN compiler, ifort, on LINUX.

The experiments are similar to those in Section 8.5.1. They are be divided into several parts, where some parameters are varied to see how they affect the performance of the methods: n (total number of degrees of freedom), m (number of bubbles), s (radius of each bubble) and ϵ (density contrast). The results of the experiments are presented in terms of the required computing time (CPU), including both the setup and solution time, number of iterations or cycles (# It.), and the obtained accuracy (RES), measured as the relative norm of the residual, $\frac{||b-Ax_{j+1}||_2}{||b-Ax_0||_2}$.

For all 2L-PCG methods, the starting vector is the zero vector (i.e., $x_0 = \mathbf{0}_n$), and the termination criterion is based on (2.23) with $\delta = 10^{-8}$. In theory, the CG-generated residuals as in (2.23) should be equal to the exact residuals in RES, but they might differ in the experiments due to round-off errors, see also [66, Sect. 7.3]. Moreover, DICCG is based on Variant 5.2 (see Section 5.3), where we typically take $k^{\frac{1}{3}} = \frac{1}{8}n^{\frac{1}{3}}$; $k = 4^3, 8^3, 16^3$ are chosen for $n = 32^3, 64^3, 128^3$, respectively. Moreover, in the stationary MG methods, it is common to use the stopping criterion based on the real residuals, (i.e., $\frac{||b-Ax_{j+1}||_2}{||b-Ax_0||} < \delta = 10^{-8}$). Therefore, RES is always below δ for the stationary MG methods, while this is not necessarily the case for the 2L-PCG methods. Finally, we remark that x_{j+1} denotes the approximation of the solution after $j+1$ iterations in 2L-PCG methods, whereas it is the approximation after performing $j+1$ multigrid cycles in stationary MG methods.

9.4.1 Varying Density Contrasts

The results for the test problem with varying contrast, ϵ, are presented in Table 9.1.

A larger ϵ typically corresponds to a linear system whose coefficient matrix is more ill-conditioned. Therefore, most methods need more iterations and computing time as ϵ increases, as shown in Table 9.1. However, the performance of BoxMG-CG and BoxMG appears to be independent of the contrast. In addition, they are the fastest methods in the experiments, followed by DICCG for $\epsilon = 10^3$ or $\epsilon = 10^5$. Moreover, we observe in Table 9.1 that GMG-CG and CCMG are very sensitive to ϵ: the number of iterations grows quickly with increasing ϵ. In fact, for $\epsilon = 10$, we see that GMG-CG is competitive with both BoxMG and DICCG as a solution technique, For larger ϵ, however, the significant increase in number of iterations makes GMG-CG a much less attractive option. For CCMG-CG and AMG-CG, the number of iterations only grows slowly compared with GMG-CG. As mentioned in Section 9.3.2, CCMG-CG is not implemented as efficiently as possible. Nevertheless, we can get an idea of the cost of its optimal implementation by comparing with the performance of BoxMG. CCMG-CG is at least as expensive per iteration as BoxMG, so, in the case of $\epsilon = 10^3$, CCMG-CG would require at least 1.2 sec instead of 4.3 sec and may, therefore, be slightly faster than DICCG. However, for $\epsilon = 10^5$, CCMG-CG would need at least 2.5 sec, so DICCG is faster than CCMG-CG for larger density contrasts.

We also observe in Table 9.1 that the accuracy of DICCG is consistently the worst when compared with the other methods, although the differences are generally quite small. As mentioned in Section 9.3.1, this is caused by the fact that the deflated residuals are used in DICCG, leading to extra round-off errors.

	$\epsilon = 10$			$\epsilon = 10^3$			$\epsilon = 10^5$		
Method	CPU	# It.	RES	CPU	# It.	RES	CPU	# It.	RES
ICCG	3.1	131	0.1E-7	5.8	244	0.1E-8	6.9	289	0.5E-8
DICCG	1.1	35	0.3E-7	1.7	54	0.4E-7	1.9	59	0.4E-7
GMG-CG	1.0	33	0.1E-7	3.9	132	0.9E-8	8.0	267	0.8E-8
CCMG-CG	3.3	10	0.1E-8	4.3	18	0.1E-7	6.7	37	0.4E-8
BoxMG-CG	0.8	12	0.1E-8	0.8	12	0.2E-8	0.8	12	0.2E-8
AMG-CG	8.0	9	0.3E-8	8.9	14	0.1E-8	9.2	16	0.2E-8
CCMG	4.0	17	0.6E-8	11.2	79	0.1E-7	40.9	338	0.1E-7
BoxMG	0.8	17	0.3E-8	0.8	17	0.3E-8	0.8	17	0.3E-8
AMG	8.9	15	0.8E-8	13.0	40	0.1E-7	21.5	92	0.9E-8

Table 9.1: Convergence results for the experiment with $n = 64^3$, $m = 2^3$, $s = 0.05$, and varying the contrast, ϵ. 'CPU', '# It.' and 'RES' denote the total computing time, number of iterations or cycles, and the accuracy of the solution measured as the relative norm of the exact residuals, respectively.

9.4.2 Varying Bubbly Radii

The results for an experiment with varying the radius of the bubbles, s, are given in Table 9.2. The smallest radius is chosen to be $s = 0.01875$, because the bubbles are no longer resolved for $s < \frac{1}{64} = 0.015635$. In general, a smaller radius does not significantly affect the conditioning of the coefficient matrix, but it does change the form of the errors that are difficult to resolve, possibly making them more difficult to approximate.

In Table 9.2, it can be seen that there are changes in convergence behavior of the various methods for different radii. In general, a smaller s leads to a more favorable performance for several of the iterative methods, including ICCG, GMG-CG and CCMG-CG. The other methods do not have a clear relation with respect to s. This even seems to hold for the stationary methods, CCMG, BoxMG and AMG, which we might expect to be sensitive to the size of the bubbles due to the challenges discussed in Section 9.2.2. BoxMG and BoxMG-CG seem to be fully insensitive to s. They are also the fastest methods in this experiment, followed by again DICCG. It is interesting to note that while GMG-CG is very ineffective in the case of large bubbles, its performance improves as the bubbles shrink, and, for the case of $s = 0.01875$, it becomes competitive with BoxMG and DICCG.

	$s = 0.1$			$s = 0.05$		
Method	CPU	# It.	RES	CPU	# It.	RES
ICCG	6.0	250	0.1E-7	5.8	244	0.1E-8
DICCG	1.3	39	0.3E-7	1.7	54	0.4E-7
GMG-CG	4.3	143	0.9E-8	3.9	132	0.9E-8
CCMG-CG	5.1	24	0.4E-8	4.3	18	0.1E-7
BoxMG-CG	0.8	12	0.3E-8	0.8	12	0.2E-8
AMG-CG	8.3	11	0.6E-8	8.9	14	0.1E-8
CCMG	13.0	95	0.8E-8	11.2	79	0.1E-7
BoxMG	0.8	17	0.4E-8	0.8	17	0.3E-8
AMG	10.6	25	0.8E-8	13.0	40	0.1E-7

	$s = 0.025$			$s = 0.01875$		
Method	CPU	# It.	RES	CPU	# It.	RES
ICCG	3.8	159	0.8E-8	4.1	170	0.9E-8
DICCG	1.5	46	0.9E-7	1.5	45	0.8E-7
GMG-CG	2.6	85	0.8E-8	1.3	41	0.4E-8
CCMG-CG	3.6	12	0.5E-8	3.8	14	0.4E-8
BoxMG-CG	0.8	12	0.2E-8	0.8	12	0.1E-8
AMG-CG	8.2	12	0.7E-8	8.8	14	0.3E-8
CCMG	7.0	43	0.7E-8	10.1	69	0.9E-8
BoxMG	0.8	17	0.3E-8	0.8	17	0.3E-8
AMG	10.9	29	0.6E-8	12.8	39	0.7E-8

Table 9.2: Convergence results for the experiment with $n = 64^3$, $m = 2^3$, $\epsilon = 10^3$, and varying the radius of the bubbles, s.

9.4.3 Varying Number of Bubbles

In Table 9.3, we present results demonstrating the performance of the various solvers with variation in the number of bubbles, m. Note that the test case with $m = 0$ corresponds to the Poisson equation with a constant density, i.e., a domain with only one phase. From Proposition 4.1, we know that increasing m leads to the appearance of more large eigenvalues in the original coefficient matrix, A, and small eigenvalues in the IC(0)-preconditioned coefficient matrix, $M^{-1}A$. This results in a more difficult linear system to solve, although both the original and preconditioned coefficient matrices are not necessarily worse conditioned.

It can be seen in Table 9.3 that, for most methods, the convergence worsens with increasing m, as expected. Moreover, GMG-CG is the best method in the case of $m = 0$, but quickly loses efficiency for $m > 0$. The number of iterations required by CCMG also grows rapidly with m, whereas it increases gradually for CCMG-CG, AMG-CG and AMG. A single iteration of these methods, however, is more expensive than one of BoxMG or DICCG, as mentioned in Section 9.3. For $m > 0$, BoxMG and BoxMG-CG are always the fastest methods, followed by DICCG. The performance of BoxMG-CG degrades only a little with increasing m, while the iteration counts for BoxMG increase more substantially. For a sufficiently large number of deflation vectors, k, DICCG would be less sensitive to changes in m, see Section 8.5.1.

	$m = 0$			$m = 1$		
Method	CPU	# It.	RES	CPU	# It.	RES
ICCG	3.0	125	0.8E-8	3.9	163	0.7E-8
DICCG	1.0	31	0.2E-7	1.5	47	0.2E-7
GMG-CG	0.3	8	0.1E-8	3.7	124	0.8E-8
CCMG-CG	3.2	9	0.7E-8	3.7	13	0.7E-8
BoxMG-CG	0.8	12	0.1E-8	0.8	12	0.1E-8
AMG-CG	7.7	7	0.9E-8	8.0	9	0.2E-8
CCMG	4.0	17	0.4E-8	10.0	68	0.9E-8
BoxMG	0.8	17	0.3E-8	0.8	17	0.3E-8
AMG	8.2	11	0.6E-8	11.8	33	0.6E-8

	$m = 2^3$			$m = 3^3$		
Method	CPU	# It.	RES	CPU	# It.	RES
ICCG	5.8	244	0.1E-8	8.2	342	0.8E-8
DICCG	1.7	54	0.4E-7	2.0	60	0.7E-7
GMG-CG	3.9	132	0.9E-8	9.8	329	0.8E-8
CCMG-CG	4.3	18	0.1E-7	6.5	35	0.4E-8
BoxMG-CG	0.8	12	0.2E-8	1.0	15	0.2E-8
AMG-CG	8.9	14	0.1E-8	8.9	14	0.8E-8
CCMG	11.2	79	0.1E-7	27.9	223	0.1E-7
BoxMG	0.8	17	0.3E-8	1.3	29	0.7E-8
AMG	13.0	40	0.1E-7	14.1	45	0.7E-8

Table 9.3: Convergence results for the experiment with $n = 64^3$, $s = 0.05$, $\epsilon = 10^3$, and varying the number of bubbles, m.

9.4.4 Varying Number of Grid Points

Table 9.4 presents results with a varying grid size, n. A larger n leads to coefficient matrices that are more ill-conditioned, as mentioned in Section 9.2. It can be observed in Table 9.4 that only BoxMG, BoxMG-CG and DICCG show perfectly scalable iteration counts with respect to the number of grid points, although the computing times grow relatively quickly. Recall that, for DICCG, more deflation vectors are taken for larger n, which results in a decreasing number of iterations for DICCG. Moreover, observe that BoxMG and BoxMG-CG outperform the other methods both in terms of the number of iterations and the computing time. For larger n, the number of iterations for CCMG-CG and AMG-CG grows very slowly; however, the large cost per iteration combined with the large setup cost for these methods still makes them uncompetitive. As mentioned earlier, a lower bound for the cost of CCMG-CG can be given; in the case of $n = 128^3$, CCMG-CG would require at least 11.1 sec (instead of 38.0 sec for our current implementation), and, therefore, may be competitive with DICCG. Furthermore, while AMG and AMG-CG are competitive in terms of the number of iterations required for convergence, they are clearly much more expensive; this is due to the extra costs associated with the unstructured-grid data structures used within AMG, and the extra setup required based on this assumption, as discussed in Section 9.3.2.

	$n = 32^3$			$n = 64^3$			$n = 128^3$		
Method	CPU	# It.	RES	CPU	# It.	RES	CPU	# It.	RES
ICCG	0.3	112	0.9E-8	5.8	244	0.1E-8	92.3	444	0.9E-8
DICCG	0.2	64	0.8E-7	1.7	54	0.4E-7	11.7	39	0.3E-7
GMG-CG	0.2	81	0.9E-8	3.9	132	0.9E-8	36.0	134	0.8E-8
CCMG-CG	0.4	14	0.3E-8	4.3	18	0.1E-7	38.0	19	0.9E-8
BoxMG-CG	0.1	12	0.2E-8	0.8	12	0.2E-8	7.0	12	0.3E-8
AMG-CG	0.8	10	0.4E-8	8.9	14	0.1E-8	89.0	15	0.2E-8
CCMG	0.8	44	0.1E-7	11.2	79	0.1E-7	99.1	85	0.8E-8
BoxMG	0.1	16	0.8E-8	0.8	17	0.3E-8	7.4	17	0.4E-8
AMG	0.9	20	0.1E-7	13.0	40	0.1E-7	129.2	45	0.8E-8

Table 9.4: Convergence results for the experiment with $m = 2^3$, $s = 0.05$, $\epsilon = 10^3$, and varying the total number of degrees of freedom, n.

In this experiment, the scalability of the methods can be easily observed. Considering the computing time, it can be seen that the required CPU time for both DICCG and BoxMG increase by a factor of approximately 8 when doubling n in each direction. This is quite favorable in comparison with the other methods, which scale with factors of 10 or more. Finally, we observe that ICCG requires significantly more CPU time as n increases, as predicted by the classical theory.

9.4.5 Difficult Test Problem

We end the stationary experiments with a test case where the most difficult parameters (taken based on the previous experiments) are chosen. The results associated with this experiment can be found in Table 9.5, using termination tolerance $\delta = 10^{-8}$. We note that a higher accuracy than $\delta = 10^{-8}$ cannot be reached, due to the accumulation of round-off errors and the effects of finite precision arithmetic.

As in the other experiments, BoxMG-CG and BoxMG perform the best, in terms of both the number of iterations and the computing time. They are again followed by DICCG, which also performs rather well. GMG-CG, CCMG, and AMG all typically fail to converge within the allowed number of iterations for $\delta = 10^{-8}$ (and also for the weaker tolerance $\delta = 10^{-6}$). CCMG-CG and AMG-CG do converge but, as always, are not competitive in terms of true CPU time.

Method	CPU	# It.	RES
ICCG	195.4	942	0.1E-7
DICCG	19.6	65	0.5E-7
GMG-CG	–	> 1000	–
CCMG-CG	114.0	92	0.1E-7
BoxMG-CG	7.4	13	0.4E-8
AMG-CG	107.6	29	0.8E-8
CCMG	–	> 1000	–
BoxMG	7.8	18	0.7E-8
AMG	–	> 1000	–

Table 9.5: Convergence results for the difficult test problem. The following parameters are kept constant: $n = 128^3$, $m = 3^3$, $s = 0.025$, $\epsilon = 10^5$, and $\delta = 10^{-8}$.

While neither AMG nor CCMG converge within the allowed number of iterations as standalone solvers, both perform reasonably well as preconditioners. In fact, both unaccelerated solvers do converge, but very slowly, with AMG converging marginally faster than CCMG. While it may be possible to improve this performance somewhat by using different smoothing schemes, or by changing some of the parameters in the AMG setup stage, this is beyond the scope of the current study.

9.5 Concluding Remarks

After performing an algebraic comparison of multilevel techniques based on their abstract forms in Chapters 6 and 7, we present a comparison of several of these techniques using their typical and optimized parameters, which may be considered as efficient solvers for linear systems from two-phase bubbly flows. In particular, two families of algorithms are considered in this chapter; the DICCG algorithm, based on the principles of deflation for classical PCG techniques, and multigrid algorithms. For the multigrid algorithms, we consider a range of approaches, including standard geometric, robust geometric and algebraic multigrid variants, applied as both standalone solvers and two-level preconditioners for 2L-PCG methods. The solvers are compared on a series of 3-D stationary problems, where it is shown that BoxMG-CG and DICCG are the most stable and efficient techniques. Overall, we demonstrate that solution of the pressure-correction equation within bubbly flow applications can be significantly accelerated using the methods studied here. In the next chapter, we continue on the comparison of BoxMG-CG and DICCG, but applied to time-dependent bubbly flow problems, where the density field varies at each time step. It might be interesting to see whether the multilevel techniques are also efficient for these more realistic test problems.

Chapter 10

Bubbly Flow Simulations

10.1 Introduction

The main application of this thesis is bubbly flows, whose computation is a very active research topic in CFD, see, for instance, [34, 49, 130, 152, 153, 155, 156] and, more recently, [74, 95, 125, 131, 157]. In the previous chapters, we have performed numerical experiments based on stationary bubbly flow problems, i.e., problems in which the bubbles are fixed in the computational domain and do not evolve in time. We have shown that DICCG and BoxMG-CG are efficient methods for solving the corresponding linear systems in stationary problems, see Section 9.4. However, in practice, the density field usually changes in time. Therefore, in this chapter, some numerical experiments (also called simulations) are carried out based on 3-D time-dependent bubbly flow problems, where the density field evolves in time.

The aim of this chapter is to examine whether DICCG and BoxMG are still effective and efficient to solve a sequence of linear systems in simulations. A comparison between DICCG variants for these simulations is carried out in Appendix H.

In Section 10.2, we first describe concisely the mathematical model that is used for the bubbly flow simulations, where we refer [154, Sect. 8.3.2] for more details. Thereafter, the results of the simulations are presented in Section 10.3. Concluding remarks are given in Section 10.4.

10.2 Mathematical Model of the Bubbly Flow

Bubbly flows are mathematically governed by the incompressible Navier-Stokes equations,

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\frac{1}{\rho} \nabla p + \frac{1}{\rho} \nabla \cdot \mu \left(\nabla \mathbf{u} + \nabla \mathbf{u}^T \right) + \mathbf{f}, \tag{10.1}$$

subject to an incompressibility constraint,

$$\nabla \cdot \mathbf{u} = 0, \tag{10.2}$$

where $\mathbf{u} = (u, v, w)^T$ is the velocity vector, and ρ, p, μ and $\mathbf{f}$ are the density, pressure, viscosity and source function (consisting of, for example, gravity and interface tension forces), respectively, which are functions of spatial coordinates and time. In this chapter, we only consider simulations without surface tension forces in order to obtain complicated density fields with many and small bubbles. We assume the density and viscosity are constant within each fluid. At the boundaries of the domain, we impose Dirichlet boundary conditions for the velocity.

Eqs. (10.1) and (10.2) are solved on an equidistant Cartesian grid in a rectangular domain using a pressure-correction method [162]. These equations are discretized using finite differences on a uniform staggered grid with n cells, where the grid points of the pressure variables are located at the cell centers, and the grid points associated with velocity components are located at the cell-face centers.

In the pressure-correction method, a tentative velocity vector, $\mathbf{u}^*$, is first computed from

$$\frac{\mathbf{u}^* - \mathbf{u}^l}{\Delta t} = -\nabla \cdot \mathbf{u}^l \mathbf{u}^l + \frac{1}{\rho} \nabla \cdot \mu \left(\nabla \mathbf{u}^* + (\nabla \mathbf{u}^l)^T \right), \tag{10.3}$$

where $\mathbf{u}^l$ denotes the value of $\mathbf{u}$ at time step l. The resulting system of equations for unknown vector $\mathbf{u}^*$ is solved, for example, using the PCG method. The velocities at the new time step, $l+1$, are computed from

$$\frac{\mathbf{u}^{l+1} - \mathbf{u}^*}{\Delta t} = -\frac{1}{\rho} \mathcal{G} p + \mathbf{f},$$

under the constraint of (10.2). This yields

$$\left\{ \begin{array}{rcl} \mathbf{u}^{l+1} & = & \mathbf{u}^* + \Delta t \left(-\frac{1}{\rho} \mathcal{G} p + \mathbf{f} \right); \\ \mathcal{D} \mathbf{u}^{l+1} & = & 0, \end{array} \right. \tag{10.4}$$

where $\mathcal{D}$ represents the discretization of the divergence operator, and $\mathcal{G}$ is the discrete gradient operator. Finally, Equation (10.4) gives

$$\mathcal{D} \frac{1}{\rho} \mathcal{G} p = \mathcal{D} \left(\frac{1}{\Delta t} \mathbf{u}^* + \mathbf{f} \right), \tag{10.5}$$

which is known as both the pressure-correction equation and the Poisson equation with a discontinuous coefficient (cf. Eq. (1.3)). Eq. (10.5) can again be solved using, for example, the PCG method. However, solving (10.5) requires significantly more computing time than finding the solution of (10.3), since the convergence of the iterative process suffers from the highly discontinuous behavior of the coefficient, ρ, but is not ameliorated by a small Δt. More strongly, solving (10.5) typically consumes the bulk of the computing time for all computations of the bubbly flow, see, e.g., [24, 74, 154]. Further details about the pressure-correction method applied to bubbly flows can be found in [155–157].

Due to the staggered grid, we do not have pressure points at the boundaries of the domain. Explicit pressure boundary conditions are, however, not required in the method, since, in Eq. (10.5), the velocity boundary conditions are naturally included in the discrete divergent operator, $\mathcal{D}$. It follows implicitly that Neumann boundary conditions hold for the pressure. In this case, the pressure is a relative variable, since the differences in pressure and not its absolute values are meaningful in the pressure-correction method.

Eq. (10.5) can be written as a linear system (see Eq. (1.1))

$$Ax = b, \quad A = [a_{ij}] \in \mathbb{R}^{n \times n}, \tag{10.6}$$

for $n = n_x n_y n_z$, and a singular SPSD matrix, A. It appears that $b \in \mathcal{R}(A)$ is always satisfied for (1.1), see [145] for more details. In this case, (1.1) is consistent and the solution, x, is determined up to a constant.

We consider two-phase bubbly flows with air (a low-density phase) and water (a high-density phase). In this case, ρ is piecewise-constant with a density contrast $\epsilon \approx 820$, which is the ratio of the two densities, see Section 1.3. Moreover, the density advection is performed using the mass-conserving level-set method [154, 157].

Remark 10.1.

- *Operator-splitting methods, such as the pressure-correction method, are amongst the oldest numerical schemes for solving the incompressible Navier-Stokes equations, dating back to the original work of Chorin [28, 29]. In the 1980s, this work was extended to second-order convergent methods for the velocities [13, 162].*

- *Many other approaches could be employed for both the solution of the Navier-Stokes equations and the advection of the density field. Artificial compressibility techniques, for example, replace the incompressibility condition by one with a small compressibility term that vanishes when treated appropriately, recovering, in this limit, the original (incompressible) Navier-Stokes equations [28, 84]. While we only consider the standard finite-difference discretization, other approaches are also possible; finite-element discretizations of Navier-Stokes are complicated by the need to satisfy a discrete inf-sup condition to give stable pressure discretizations [55, 60]. While we use an interface-capturing level-set scheme, other approaches include front-tracking techniques [152], the volume-of-fluid and marker-and-cell methods, as well as arbitrary Lagrangian-Eulerian techniques. Lattice Boltzmann techniques may also be used to model incompressible multi-phase flow, with similar considerations arising [75].*

10.3 Bubbly Flow Simulations

In this section, we consider simulations of three ‘real-life’ bubbly flows. In order to obtain sophisticated geometries, these flows are considered without surface tension. The pressure-correction method is adopted to solve the Navier-Stokes equations, as described in Section 10.2. The interface advection is carried out using the mass-conserving level-set method [155–157]. Our main interest in each time step is the pressure solve (10.6), which takes the bulk of the computing time in each simulation, especially when the total degrees of freedom, n, are relatively large. The actual time step, Δt, is restricted by

$$\Delta t \leq \tau := \frac{h\varphi}{2\left(|u|_{\max} + |v|_{\max} + |w|_{\max}\right)},$$

where h is the distance between grid points in one direction and $\varphi = 0.35$ is the CFL number, see [155] for more details. That means that we use an adaptive time stepping procedure by considering the time-step restrictions due to convection of the bubbly flow. At each time step, l, the actual time, t, is advected with an increment, Δt, that obeys

$$\Delta t = \min\left(\tau, \Delta t_{\max}\right),$$

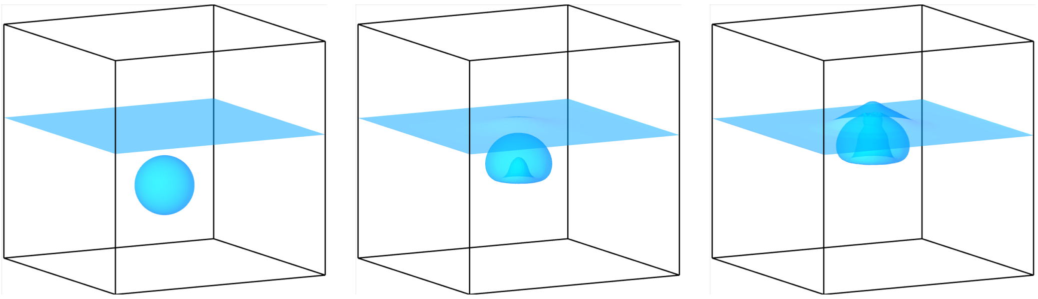
where we choose $\Delta t_{\max} = 0.0005$ sec.

DICCG$-k$ denotes DICCG2 with k projection vectors (see Definition 8.1), based on deflation variant 5.3 (see Table 5.1). Then, at each time step, the resulting linear system (10.6), with n, is solved using both BoxMG-CG and DICCG$-k$, as these are shown to be the most stable and efficient methods for the stationary problems considered in Section 9.4. The number of projection vectors, k, is chosen to minimize the required CPU time. ICCG is used as a benchmark in the experiments. The initial guess for each solve is chosen to be the previous solution, except for the first 10 time steps, where the zero starting vector is used (to avoid problems with achieving a too-strict relative residual reduction when the flow is initialized). The termination criterion of all methods is chosen as in the stationary experiments in Section 9.4. For more details on the physical problems simulated here, see [155, 156].

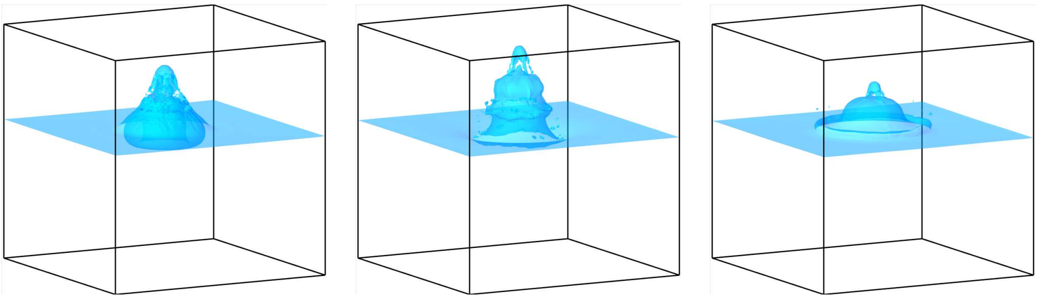
10.3.1 Rising Air Bubble in Water

We first consider a test problem where a cube of 1 cm^3 is filled with water to a height of 0.6 cm. For these experiments, we take the density of water to be 820 times that of air (i.e., $\varepsilon = 1.22 \times 10^{-3}$). At the initial time step, $l = 0$, a spherical air bubble with radius of 0.125 cm is located in the middle of the domain at a height of 0.35 cm. The exact material constants and other relevant conditions for this simulation can be found in [155, Sect. 7.2]. The evolution of the geometry during 500 time steps is given in Figure 10.1. Here, we take a grid with $n = 100^3$; in this case, the optimal value for k in DICCG$-k$ appears to be $k = 20^3$. Results of the experiment can be found in Figure 10.2, showing both the number of iterations and computing time required for each method for the pressure solves at each time step, l.

It can be readily observed from Figure 10.2 that, for each time step, both DICCG and BoxMG-CG converge in fewer iterations and require less computing time than ICCG. Due to the zero starting vector in the first 10 iterations, one can observe a peak in the ICCG cost around these first iterations, whereas this phenomenon cannot be clearly seen for DICCG and BoxMG-CG. Moreover, BoxMG-CG shows better performance than DICCG. We remark that both methods behave smoothly in time and

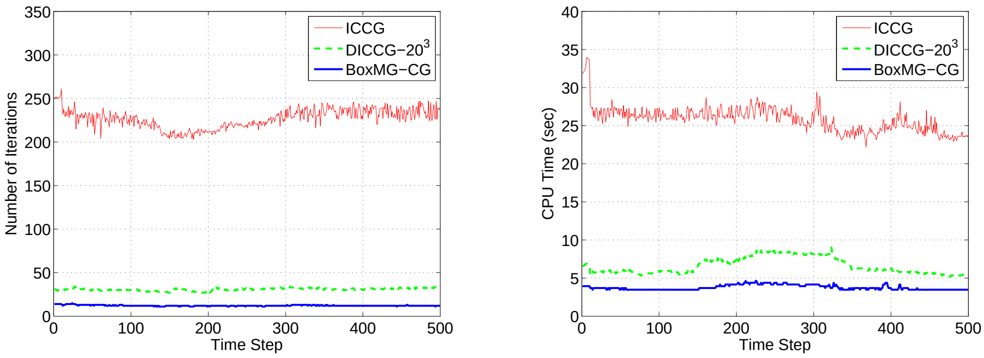


(a) $I = 0$ ($t = 0$ sec). (b) $I = 100$ ($t = 0.013$ sec). (c) $I = 200$ ($t = 0.022$ sec).



(d) $I = 300$ ($t = 0.032$ sec). (e) $I = 400$ ($t = 0.050$ sec). (f) $I = 500$ ($t = 0.064$ sec).

Figure 10.1: Evolution of a rising bubble in water. Parameters I and t denote the time step and actual time, respectively.



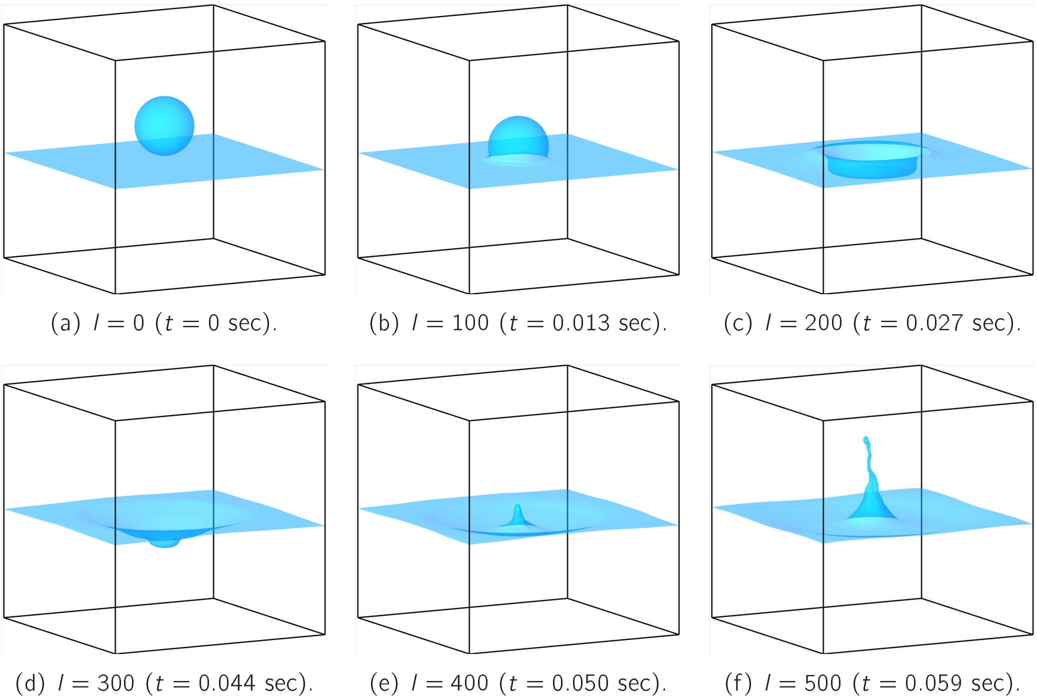
(a) Number of iterations versus time step. (b) CPU time versus time step.

Figure 10.2: Results for ICCG, DICCG-20^3 and BoxMG-CG for the pressure solves during the real-life simulation with a rising air bubble in water.

seem to be more-or-less independent of the (sometimes complicated) geometry of the density field. This is in contrast to ICCG, whose convergence is rather erratic. Only some small outliers can be observed in the convergence of DICCG and BoxMG. For example, a small peak can be seen around $l = 325$ in DICCG, and around $l = 390$ and $l = 410$ in BoxMG-CG. This is likely related to particular changes in the density field at these time steps, but it is difficult to pinpoint the cause, due to the complicated surface dynamics at these time steps. Moreover, it can be seen that, for $l \in [150, 350]$, more iterations are required especially for DICCG, because the geometry is most complicated in this period, due to the interaction of the bubble with the interface and the appearance of many droplets, as can be observed in Figure 10.1.

10.3.2 Falling Water Droplet in Air

In the next simulation, we consider a 1 cm^3 cube filled with water to a height of 0.45 cm. At the initial time step, $l = 0$, a spherical water droplet with radius 0.125 cm is located in the middle of the domain at a height of 0.6 cm. The same material constants and conditions are used as in Section 10.3.1. The evolution of the geometry during the 500 time steps is depicted in Figure 10.3. Again, the grid resolution is $n = 100^3$ and the optimal number of projection vectors is $k = 20^3$.

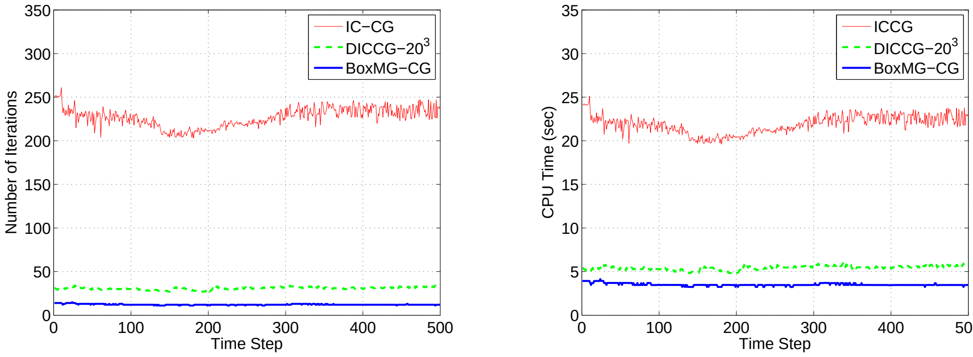


(a) $l = 0$ ($t = 0$ sec). (b) $l = 100$ ($t = 0.013$ sec). (c) $l = 200$ ($t = 0.027$ sec).

(d) $l = 300$ ($t = 0.044$ sec). (e) $l = 400$ ($t = 0.050$ sec). (f) $l = 500$ ($t = 0.059$ sec).

Figure 10.3: Evolution of a falling droplet in air.

The results of the experiment can be found in Figure 10.4. It can again be noticed that ICCG performs worse than both DICCG and BoxMG-CG. BoxMG-CG is always faster than DICCG, although the differences are small in this experiment; the number of iterations and computing time per time step are approximately the same for both methods. We observe a very smooth behavior of the corresponding performance curves, because very few additional bubbles or droplets appear during the simulation. In this experiment, BoxMG and DICCG are more-or-less insensitive to the geometry of the density field, while it can be readily observed that the performance of ICCG does depend on it.



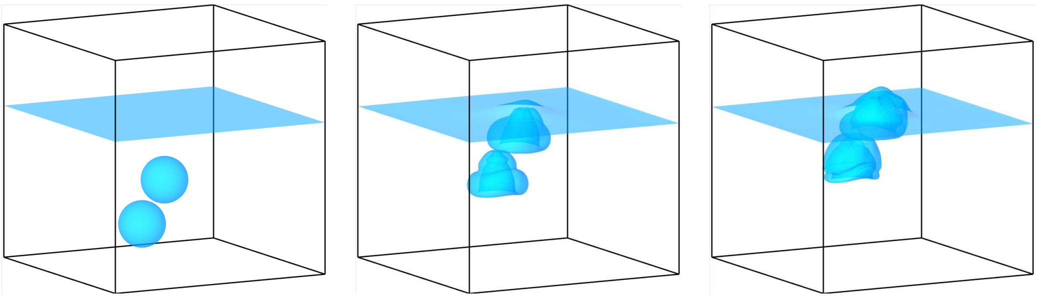
(a) Number of iterations versus time step. (b) CPU time versus time step.

Figure 10.4: Results for ICCG, DICCG-20^3 and BoxMG-CG for the pressure solves during the real-life simulation of a falling water droplet in air.

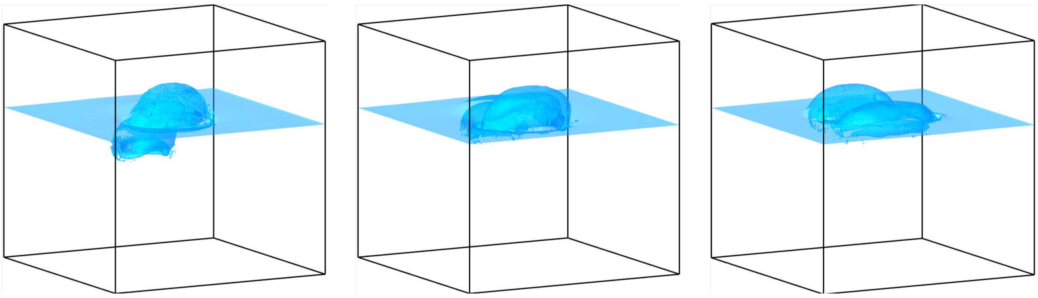
10.3.3 Two Rising and Merging Air Bubbles in Water

In the final simulation, we consider a test problem where a 1 cm^3 cube is filled with water to a height of 0.65 cm. At the initial time step, $l = 0$, two air bubbles of radius 0.1 cm are located with centers at coordinates $(0.5, 0.5, 0.37)$ and $(0.5, 0.3, 0.15)$. The evolution of the geometry during 2500 time steps can be found in Figure 10.5. This test problem is, obviously, harder to solve than the previous two test problems, since there is interaction between the two bubbles at the same time as they interact with the water interface. In addition, we now consider a refined grid with $n = 200^3$, resulting in a strongly ill-conditioned coefficient matrix and making the problem very complicated to solve. DICCG$-k$ with $k = 25^3$ appears to be optimal in terms of the required CPU time for all possible k.

Results are presented in Figure 10.6. ICCG is omitted in these results, since it is extremely slow in this difficult test case, requiring, on average, over 700 iterations and 500 seconds of CPU time per time step during the first 100 time steps, which have relatively simple dynamics. It can be observed that the number of iterations, and, therefore, also the computing time, increases gradually during the simulation for both methods, but especially for DICCG-25^3. This is due to the fact that the geometry of the problem becomes progressively more sophisticated as the simulation proceeds. Apparently, the influence of the projection vectors depends heavily on the time step. This even holds if we increase k. Obviously, BoxMG is always faster than DICCG (especially for $l \in [1000, 2500]$), both with respect to the number of iterations and the computing time.

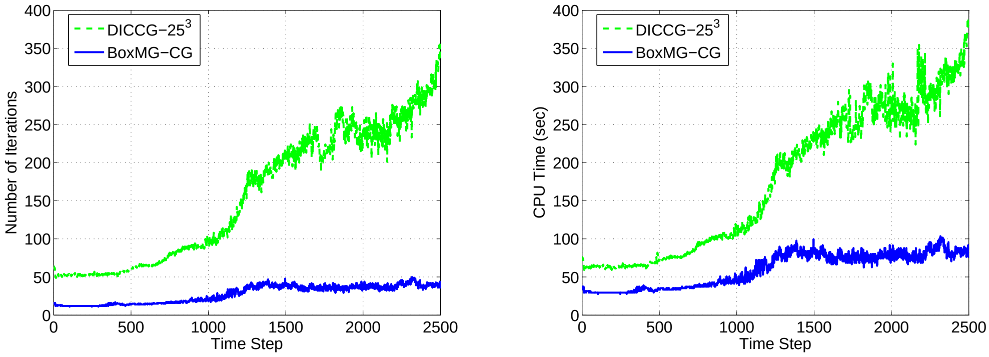


(a) $I = 0$ ($t = 0$ sec). (b) $I = 500$ ($t = 0.025$ sec). (c) $I = 1000$ ($t = 0.035$ sec).



(d) $I = 1500$ ($t = 0.045$ sec). (e) $I = 2000$ ($t = 0.056$ sec). (f) $I = 2500$ ($t = 0.066$ sec).

Figure 10.5: Evolution of two rising air bubbles in water.



(a) Number of iterations versus time step. (b) CPU time versus time step.

Figure 10.6: Results for DICCG−25^3 and BoxMG-CG for the pressure solve during the real-life simulation with two rising air bubbles in water. ICCG is omitted in these results, because it is not competitive with the other two methods.

10.4 Concluding Remarks

In previous chapters, we have seen that some deflation (DICCG) and multigrid (BoxMG) methods are effective to solve stationary bubbly flow problems. In this chapter, the success of DICCG is emphasized in realistic bubbly flow simulations. Compared to ICCG, the benefit of the deflation method is obviously observed in terms of both the number of iterations and the CPU time. Moreover, we conclude that BoxMG-CG performs better than DICCG, especially for relatively large grid sizes. BoxMG-CG is more scalable, and requires fewer iterations and less computing time in all experiments, for all time steps.

BoxMG seems to have a low sensitivity to the density fields, gives accurate solutions and is very robust in all cases. Improvement of DICCG to give performance comparable to BoxMG-CG is a subject for future research. The relatively large coarse grids required by DICCG to achieve good convergence properties suggest that there is a need for a better solver for the coarse linear systems in DICCG in order to make the method more efficient and scalable. Overall, we demonstrate that solution of the pressure-correction equation within bubbly flow simulations can be significantly accelerated using two-level PCG methods.

Chapter 11

Conclusions

The focus of this thesis is on the analysis of two-level preconditioned Conjugate Gradient (PCG) methods in which the deflation method (DPCG or DEF) plays a central role. Most of the performed numerical experiments are based on the Poisson equation with a discontinuous coefficient derived from bubbly flow problems.

11.1 Conclusions

For linear systems with a nonsingular coefficient matrix, it is known that DEF can be very effective and efficient. We show that most of this theory is generalizable to singular coefficient matrices. Three variants of the deflation methods that can deal with these matrices are discussed, where we prove that all of these variants correspond to almost the same deflated-preconditioned coefficient matrices. In fact, these variants are equivalent to the original deflation method, so that DEF is expected to be effective and efficient when it is applied to linear systems with a singular coefficient matrix.

In each iteration of DEF, coarse linear systems, based on the Galerkin matrix, must be solved. We show that this can be done with a direct or iterative method, so that it involves an inner-outer iteration process in the latter case. We examine their efficiency and derive their theoretical properties. The optimal approach depends on the problem setting and grid size. For problems with highly refined grids or many projection vectors, DEF based on inner-outer iterations is the most attractive choice.

The deflation method with fixed subdomain vectors as projection vectors is well-understood if the underlying PDEs use constant coefficients. However, the density coefficient is often varying in time, such as in our bubbly flow simulations. We show that DEF with fixed subdomain projection vectors is still the method of choice for this case, although level-set and level-set-subdomain projection vectors, which depend on the density field and have different implementation properties, could also be attractive in practice. In addition, we show that the projection vectors should always be good approximations of eigenvectors associated with unfavorable eigenvalues of the coefficient matrix. We demonstrate that our choices of projection vectors (subdomain, level-set or level-set-subdomain vectors) are, indeed, good approximations for these eigenvectors in our bubbly flow problems. It depends on the implementation, the geometry of problem, and the maximum number of allowed projection vectors, which of these variants is the most suitable one in practice.

The effectiveness and efficiency of the deflation method are also emphasized in numerical experiments for both stationary and time-dependent bubbly flows. Compared with PCG, DEF significantly reduces the computational cost for most of the test cases, especially for problems with many bubbles or a highly refined grid. Additionally, the deflation method is less sensitive to the contrasts between the phases, and is scalable in terms of iterations and CPU time, as long as the number of projection vectors is chosen to be proportional to the grid size.

In addition to the deflation method, several other two-level PCG methods are well-known in the literature, among them are methods based on additive coarse-grid correction (AD), balancing Neumann-Neumann (BNN), reduced variants of balancing Neumann-Neumann (R-BNN), and multigrid V(1,0)-, V(0,1)- and V(1,1)-cycles. The abstract forms of these methods are compared theoretically and numerically. For certain choices, we obtain the remarkable result that some of these methods are mathematically equivalent. Most of these methods can be divided into two classes, each having the same spectral properties. The differences between the two classes are small, so that similar convergence behaviors are expected. Moreover, we show both theoretically and numerically that the second class (consisting of the two-level PCG methods based on BNN and multigrid V(1,0)-, V(0,1)-cycles) is more robust than the first class (consisting of DEF and R-BNN), although some of the methods from these classes are mathematically equivalent.

We derive that the two-level PCG method with the multigrid V(1,0)-cycle preconditioner is the same as an adapted variant of both DEF and R-BNN. In addition, we advocate that this method is the best method with respect to effectiveness, efficiency and robustness for a class of problems. Additionally, when simple choices are made for the algorithmic components, each iteration of the two-level PCG method based on a multigrid V(1,1)-cycle (MG) is more expensive than a DEF iteration. At first glance, we would expect MG to be the most effective method; however, we show that there exist some parameters such that DEF is expected to converge more rapidly than MG. But, for more realistic choice of parameters, MG is expected to be faster than both DEF and the other 2L-PCG methods given above, although the work per iteration of MG may remain more than for the other methods. For typical choices of parameters, we derive that BNN, DEF and MG require the same amount of work per iteration, and their spectra are almost the same. Hence, these methods are expected to show a comparable convergence behavior while the corresponding cost is similar.

A comparison between DEF and MG is also performed, where the preconditioners are based on their own typical and optimized set of parameters. For the multigrid algorithms, we consider a range of approaches, including standard geometric, robust geometric and algebraic multigrid variants. The solvers are compared on a series of stationary problems in three dimensions, where we demonstrate that DEF and MG based on the Dendy's blackbox multigrid preconditioner (BoxMG-CG) are the most robust and efficient 2L-PCG methods. Large time-dependent bubbly flow simulations are also performed, showing efficient and scalable solution of the pressure-correction equation using these methods. BoxMG-CG is more scalable, and requires fewer iterations and less computing time than DEF.

Overall, we demonstrate that solving the Poisson equation with a discontinuous coefficient within bubbly flow applications can be done more efficiently using some of the two-level PCG methods studied in this thesis.

11.2 Future Research

As shown in this thesis, the deflation method is a fast and efficient method. However, it should be improved to give performance comparable to BoxMG-CG for very large problems. The relatively large coarse grids, required by the deflation method to achieve good convergence properties, suggest that there is a need for a better solver for the linear systems associated with the Galerkin matrix. This is required to make the method more efficient and scalable. An alternative is to use the multilevel (projection-based) Krylov method as proposed in [48], where the Galerkin systems are solved recursively, so that the resulting approach is close to typical multigrid methods.

Moreover, the choice of the traditional preconditioner and the projection vectors could be further improved in the deflation method. Currently, we use the incomplete Cholesky preconditioner, but it might be more favorable to use operator-based preconditioners, based on ideas described in [32]. In addition, we use subdomain projection vectors that are chosen independently of the discontinuous coefficient. We have analyzed that this is an efficient approach, but coefficient-dependent projection vectors (such as the level-set or level-set-subdomain vectors, which are described in this thesis) might be advantageous for relatively large and complicated bubbly flow problems.

Another important issue for future research is the parallelization of the deflation method (and its adapted variant) in this thesis, following the guidelines given in [56]. This is required to cope with very large 3-D bubbly flows problems. A fast traditional preconditioner based on block incomplete Cholesky factorization and first steps to parallelize the deflation operator have already been carried out. In addition, parallel asynchronous iterative methods exhibit properties that are highly favorable in the context of large heterogeneous networks of computers. By combining these methods with deflation–type techniques, sophisticated parallel preconditioners may be constructed that are both efficient and robust. This approach is currently under investigation by Tijmen Collignon with promising preliminary results.

Appendix A

Basic Theoretical Results

In this chapter, we present some fairly basic results related to linear algebra, which are used in this thesis.

Lemma A.1. *Let $B, C \in \mathbb{R}^{n \times n}$ be arbitrary matrices. Then, the following equations hold:*

(a) $\sigma(BC) = \sigma(CB)$;

(b) $\sigma(B + I) = \sigma(B) + \sigma(I)$;

(c) $\sigma(B) = \sigma(B^T)$.

Proof. (a) Let $\lambda \in \mathbb{C}$ and $v \in \mathbb{C}^n$ be an eigenvalue and corresponding eigenvector of BC, respectively. We consider two cases:

- $\lambda \neq 0$: the corresponding v satisfies $BCv \neq \mathbf{0}_n$, so, in particular, we have $Cv \neq \mathbf{0}_n$. Then, the following equations are equivalent:

$$\left\{ \begin{array}{rcl} BCv & = & \lambda v; \\ CBCv & = & \lambda Cv; \\ CBw & = & \lambda w, \end{array} \right.$$

  where $w := Cv \neq \mathbf{0}_n$.

- $\lambda = 0$: we have

$$\det(BC) = \det(CB) = 0,$$

  and, hence, if λ is a zero eigenvalue of BC, then it is also a zero eigenvalue of CB.

In other words, λ is an eigenvalue of both BC and CB.

(b) Let $\lambda \in \mathbb{C}$ and $v \in \mathbb{C}^n$ be an eigenvalue and corresponding eigenvector of $B+I$, respectively. Then, the following equations are equivalent

$$\begin{cases} (B+I)v &= \lambda v; \\ Bv &= (\lambda - 1)v; \\ Bv &= \mu v, \end{cases}$$

where $\mu := \lambda - 1$. In other words, λ is an eigenvalue of $B + I$ if and only if $\lambda - 1$ is an eigenvalue of B.

(c) By definition of determinants, $\det(B - \lambda I) = \det(B^T - \lambda I)$ holds for all $\lambda \in \mathbb{C}$, so that $\sigma(B) = \sigma(B^T)$. $\square$

Lemma A.2. *Let $B \in \mathbb{R}^{n\times n}$ be an SPSD matrix. Let $C \in \mathbb{R}^{n\times n}$ be any matrix. Then, $\widetilde{B} := C^T BC$ is SPSD.*

Proof. Matrix $\widetilde{B}$ is symmetric, since

$$\widetilde{B}^T = (C^T BC)^T = C^T B^T C = C^T BC = \widetilde{B}.$$

Moreover, by definition,

$$u^T Bu \geq 0, \quad \forall u \in \mathbb{R}^n.$$

If we choose, in particular, $u := Cv$, we obtain

$$(Cv)^T B(Cv) = v^T C^T BCv^T = v^T \widetilde{B} v^T \geq 0,$$

which proves that $\widetilde{B}$ is SPSD. $\square$

Lemma A.3. *Let $S \in \mathbb{R}^{n\times n}$ satisfy $S^2 = S$. Let $R \in \mathbb{R}^{n\times n}$ be an SPD matrix such that SR is symmetric. Then, SR is SPD.*

Proof. Note first that

$$SR = S^2 R = S(SR)^T = SR^T S^T = SRS^T,$$

because SR is symmetric and S is a projection matrix. Then, the lemma follows from Lemma A.2 by taking $C := S^T$ and $B := R$. $\square$

Next, for two symmetric matrices, B and C, we write $B \preceq C$ if $B - C$ is PSD.

Lemma A.4 (Thm. 4.3.1 of [73]). *Let $B, C \in \mathbb{R}^{n\times n}$ be SPD matrices with the property that $B \preceq C$. Then,*

$$\lambda_i(B) \geq \lambda_i(C), \quad i = 1, 2, \ldots, n.$$

Lemma A.5 (Thm. 4.3.6 of [73]). *Let $B, C \in \mathbb{R}^{n\times n}$ be symmetric and suppose that B has at most rank s. Then,*

$$\lambda_i(B) \leq \lambda_{i+k}(B + C), \quad i = 1, 2, \ldots, n - s.$$

Subsequently, Lemma A.6 is presented, which is from the perturbation theory for the symmetric eigenvalue problem (see also [179] and [63, Thm. 8.1.8]).

Lemma A.6. *Suppose $B = C + \tau cc^T$ where $B \in \mathbb{R}^{n\times n}$ is symmetric, $c \in \mathbb{R}^n$ has unit 2-norm and $\tau > 0$. Then,*

$$\lambda_i(C) \le \lambda_i(B) \le \lambda_{i+1}(C), \quad i = 1, 2, \ldots, n-1. \tag{A.1}$$

Moreover, there exist $m_1, m_2, \ldots, m_n \ge 0$ such that

$$\lambda_i(B) = \lambda_i(C) + m_i\tau, \quad i = 1, 2, \ldots, n, \tag{A.2}$$

with $m_1 + m_2 + \ldots + m_n = 1$.

The next lemma is known as the interlacing property or the interlacing eigenvalues theorem for bordered matrices (see, e.g., [63, Thm. 8.1.7]).

Lemma A.7 (Interlacing Property)**.** *If $B \in \mathbb{R}^{n\times n}$ is symmetric and $B_s = B(1:s, 1:s)$, then*

$$\lambda_1(B_{s+1}) \le \lambda_1(B_s) \le \lambda_2(B_{s+1}) \le \ldots \le \lambda_s(B_{s+1}) \le \lambda_s(B_s) \le \lambda_{s+1}(B_{s+1}),$$

for $1 \le s \le n-1$.

Next, given an SPSD matrix $F \in \mathbb{R}^{n\times n}$ and an SPD matrix $G \in \mathbb{R}^{n\times n}$, we consider the eigenproblem,

$$G^{-1}Fy = \lambda y,$$

which can be rewritten as

$$(F - \lambda G)y = 0,$$

where λ and y are an eigenvalue and corresponding eigenvector of $G^{-1}F$, respectively. The latter problem is known as the symmetric-definite generalized eigenproblem, and $F - \lambda G$ is called a pencil, see, e.g., [63, Sect. 8.7]. In this case, λ and y are known as a generalized eigenvalue and generalized eigenvector of the pencil $F - \lambda G$, respectively. Moreover, the Crawford number, $c(F, G)$, of the pencil $F - \lambda G$ is defined as

$$c(F, G) := \min_{||y||_2=1} (y^T Fy)^2 + (y^T Gy)^2 > 0. \tag{A.3}$$

The following lemma gives information about the eigenvalues after perturbing matrix G. This lemma is a simplified variant of the original theorem given in [127], see also [63, Sect. 8.7].

Lemma A.8. *Let $F \in \mathbb{R}^{n\times n}$ be an SPSD matrix and $G \in \mathbb{R}^{n\times n}$ be an SPD matrix. Let $F - \lambda_i G$ be the symmetric-definite $n \times n$ pencil with $\lambda_1 \le \lambda_2 \le \ldots \le \lambda_n$. Suppose R_G is a symmetric $n \times n$ matrix that satisfies $||R_G||_2^2 < c(F, G)$. Then, $F - \mu_i(G + R_G)$ is symmetric-definite with $\mu_1 \le \mu_2 \le \ldots \le \mu_n$, satisfying*

$$|\arctan(\lambda_i) - \arctan(\mu_i)| \le \arctan\left(\frac{||R_G||_2}{c(F, G)}\right), \quad i = 1, 2, \ldots, n. \tag{A.4}$$

A matrix, $B = [b_{i,j}] \in \mathbb{R}^{n\times n}$, is irreducibly diagonally dominant if B is irreducible and

$$|b_{jj}| \geq \sum_{i\neq j} |b_{i,j}|, \quad j = 1, \ldots, n,$$

with strict inequality for at least one j. Now, the following lemma, which is [120, Corollary 4.8], can be proven.

Lemma A.9. *If a matrix B is irreducibly diagonal dominant, then it is nonsingular.*

Next, we give some results that characterize eigenvalues in a variational way, see also [73, Section 4.2]. Most of them use the so-called Rayleigh-Ritz ratio given by

$$\frac{y^T B y}{y^T y}, \quad B \in \mathbb{R}^{n\times n},\ y \in \mathbb{R}^n.$$

Theorem A.1 (Rayleigh-Ritz Theorem)**.** *Let $B \in \mathbb{R}^{n\times n}$ be a symmetric matrix. Then,*

$$\lambda_1 y^T y \leq y^T B y \leq \lambda_n y^T y, \quad \forall y \in \mathbb{R}^n,$$

and

$$\begin{cases} \lambda_{\max} & = & \lambda_n & = & \max_{y\neq \mathbf{0}_n} \frac{y^T B y}{y^T y} & = & \max_{y^T y=1} y^T B y; \\ \lambda_{\min} & = & \lambda_n & = & \min_{y\neq \mathbf{0}_n} \frac{y^T B y}{y^T y} & = & \min_{y^T y=1} y^T B y. \end{cases}$$

Theorem A.2 (Courant-Fischer Minimax Theorem)**.** *Let $B \in \mathbb{R}^{n\times n}$ be a symmetric matrix. Suppose that r is a given integer with $1 \leq r \leq n$. Then, for $y \neq \mathbf{0}_n$, we have*

$$\begin{cases} \min_{w_1,\ldots,w_{n-r}\in\mathbb{R}^{n\times n}} & \max_{y\perp w_1,\ldots,w_{n-r}} & \frac{y^T A y}{y^T y} & = & \lambda_r; \\ \max_{w_1,\ldots,w_{r-1}\in\mathbb{R}^{n\times n}} & \min_{y\perp w_1,\ldots,w_{r-1}} & \frac{y^T A y}{y^T y} & = & \lambda_r. \end{cases} \tag{A.5}$$

We remark that if we take $r = n$ and $r = 1$ in the first and second expression of (A.5), respectively, the assertions reduce to Theorem A.1.

Subsequently, the Frobenius norm and p-norm for matrices are defined as

$$||B||_F := \sqrt{\sum_{i,j=1}^{n} b_{i,j}^2}, \quad ||B||_p := \sup_{y\neq 0} \frac{||By||_p}{||y||_p}, \tag{A.6}$$

respectively. In particular, the 2-norm for symmetric matrices is defined as

$$||B||_2 := \sup_{y\neq 0} \frac{||By||_2}{||y||_2} = \max\{\ |\lambda_1(B)|,\ |\lambda_n(B)|\ \}, \tag{A.7}$$

where $\lambda_1 \leq \ldots \leq \lambda_n$. Moreover, we mention well-known properties of the eigenvalues of symmetric matrices, which can be found in [63, Sect. 8.1.2].

Lemma A.10. *Let $B, G \in \mathbb{R}^{n\times n}$ be symmetric matrices. Then,*

(i) $\sum_{i=1}^{n} [\ \lambda_i(B+G) - \lambda_i(B)\ ]^2 \leq ||G||_F^2$;

(ii) $\lambda_i(B) + \lambda_1(G) \leq \lambda_i(B+G) \leq \lambda_i(B) + \lambda_n(G), \quad i = 1, 2, \ldots, n;$

(iii) $|\lambda_i(B+G) - \lambda_i(B)| \leq ||G||_2, \quad i = 1, 2, \ldots, n.$

Lemma A.10(ii) is known as the Wielandt-Hoffman theorem.

If an invertible matrix satisfies some conditions, then some entries of the inverse are known a priori, see the next lemma.

Lemma A.11. *Let $C = [c_{i,j}] \in \mathbb{R}^{n \times n}$ be a symmetric and invertible matrix with the property that*

$$C\mathbf{1}_n = \gamma \mathbf{e}_n^{(n)}, \quad \gamma \neq 0. \tag{A.8}$$

Then, the entries of the last row and last column of $C^{-1} = [c_{i,j}^{-1}]$ have the same value, $\frac{1}{\gamma}$, i.e.,

$$c_{n,j}^{-1} = c_{i,n}^{-1} = \frac{1}{\gamma} \quad \forall\ i,\ j. \tag{A.9}$$

Proof. From Eq. (A.8), we obtain $\gamma C^{-1}\mathbf{e}_n^{(n)} = \mathbf{1}_n$. This yields

$$\gamma c_{i,1}^{-1} = 1 \quad \rightarrow \quad c_{i,1}^{-1} = \frac{1}{\gamma},$$

for all $i = 1, 2, \ldots, n$. Due to the symmetry of C, its inverse is also symmetric and Eq. (A.9) follows. ☐

Lemma A.12. *Suppose that $u = [u_i] \in \mathbb{R}^n$ and $v = [v_i] \in \mathbb{R}^n$. Then, rank $uv^T = 1$.*

Proof. Since

$$uv^T = [u_1 \ \cdots \ u_n]^T [v_1 \ \cdots \ v_n] = [v_1 u \ \ v_2 u \ \ \cdots \ \ v_n u],$$

the columns are multiples of each other, so that rank $uv^T = 1$. ☐

Lemma A.13. *Let matrices $B \in \mathbb{R}^{n \times s_1}$, $D \in \mathbb{R}^{n \times s_2}$ and an invertible matrix, $C \in \mathbb{R}^{n \times n}$, be given. If $\mathcal{R}(B) \subseteq \mathcal{R}(D)$, then $\mathcal{R}(CB) \subseteq \mathcal{R}(CD)$ holds.*

Proof. Denote $B = [b_1 \ \cdots \ b_{s_1}]$ and $D = [d_1 \ \cdots \ d_{s_2}]$, where $\{b_i\}$ and $\{d_i\}$ are sets of vectors. Since $\mathcal{R}(B) \subseteq \mathcal{R}(D)$ holds, we can write

$$b_i = c_1 d_1 + \cdots c_{s_2} d_{s_2}, \quad c_j \in \mathbb{R},$$

for all $i = 1, \ldots, s_1$. So, we also have

$$Cb_i = c_1 Cd_1 + \cdots c_{s_2} Cd_{s_2}, \quad c_j \in \mathbb{R},$$

giving us $\mathcal{R}(CB) \subseteq \mathcal{R}(CD)$. ☐

Subsequently, we present two well-known theorems in the linear algebra (see, e.g., [128]), followed by a consequence of these theorems.

Theorem A.3 (Fundamental Theorem of Linear Algebra)**.** *Let B be a symmetric matrix. Then,*

$$\mathcal{N}(B) = \mathcal{R}(B)^{\perp}, \quad \mathcal{R}(B) = \mathcal{N}(B)^{\perp}.$$

Theorem A.4 (Rank-Nullity Theorem)**.** *For any $B \in \mathbb{R}^{n\times n}$, we have*

$$\operatorname{rank} B + \dim \mathcal{N}(B) = n.$$

Lemma A.14. *Suppose that $S := I - RB$ and $S^* := I - R^T B$, where $B \in \mathbb{R}^{n\times n}$ is SPD and $R \in \mathbb{R}^{n\times n}$ is any matrix. Then,*

$$\dim \mathcal{N}(S) = \dim \mathcal{N}(S^*).$$

Proof. Note first that S^* is similar to S^T, since $S^* = B^{-1} S^T B$. Hence, the eigenvalues of S^* and S^T are the same (including multiplicity), so that

$$\dim \mathcal{N}(S^*) = \dim \mathcal{N}\left(S^T\right).$$

Lemma A.4 says that

$$\dim \mathcal{R}(S) + \dim \mathcal{N}(S) = n.$$

On the other hand, Theorem A.3 gives an orthogonal decomposition of

$$\mathbb{R}^n = \mathcal{R}(S) \oplus \mathcal{N}\left(S^T\right), \tag{A.10}$$

implying that

$$\dim \mathcal{N}\left(S^T\right) = n - \dim \mathcal{R}(S) = \dim \mathcal{N}(S).$$

□

The following standard definitions are related to orthogonal complements and direct sums.

Definition A.1. *Let $\mathcal{H}$ be a vector space with an arbitrary inner product, $\langle \cdot, \cdot \rangle$, and let $\mathcal{Z}$ be a closed subspace of $\mathcal{H}$. Then, the orthogonal complement $\mathcal{Y}$ of $\mathcal{Z}$, also denoted by $\mathcal{Z}^{\perp}$, is defined as*

$$\mathcal{Y} = \{ y \in \mathcal{H} \mid \langle z, y \rangle = 0 \quad \forall z \in \mathcal{Z} \}, \tag{A.11}$$

so that $\mathcal{Z}$ is the subspace orthogonal to $\mathcal{Y}$.

Definition A.2. *Let $\mathcal{X}$ be a vector space. Suppose that $\mathcal{Y}$ and $\mathcal{Z}$ are subspaces of $\mathcal{X}$. Then, $\mathcal{X}$ is said to be the direct sum of $\mathcal{Y}$ and $\mathcal{Z}$, written as*

$$\mathcal{X} = \mathcal{Y} \oplus \mathcal{Z}, \tag{A.12}$$

if each $x \in \mathcal{X}$ has a unique representation,

$$x = y + z, \tag{A.13}$$

where $y \in \mathcal{Y}$ and $z \in \mathcal{Z}$.

In other words, the direct sum of two subspaces, $\mathcal{Y}$ and $\mathcal{Z}$, is the sum of subspaces in which $\mathcal{Y}$ and $\mathcal{Z}$ have only the zero element in common. Using Definitions A.1 and A.2, we can derive Lemma A.15, which is well-known and states that the union of the subspaces $\mathcal{Y}$ and $\mathcal{Z}$ is exactly $\mathcal{H}$ (see, e.g., [83, pp. 146–147]).

Lemma A.15. *Let $\mathcal{H}, \mathcal{Y}$ and $\mathcal{Z}$ be defined as in Definition A.1. Then,*

$$\mathcal{H} = \mathcal{Y} \oplus \mathcal{Z}. \tag{A.14}$$

Note that $\dim \mathcal{Y} + \dim \mathcal{Z} = n$ for $\mathcal{H} := \mathbb{R}^n$. This means that $\mathcal{Y} = \mathbb{R}^{n-s}$ holds as $\mathcal{Z} = \mathbb{R}^s$ with $s < n$ is given.

Appendix B

Determination of Bubbles from the Level-Set Function

The level-set approach [102, 110] can be adopted to describe the density field, ρ, implicitly in many applications, such as two-phase bubbly flow applications, see [143, 154, 156]. In this approach, the interfaces of the bubbles are defined by the zero level-set of a marker function $\Psi(\mathbf{x}, t)$ that is defined as follows:

$$\begin{cases} \Psi = 0, & \text{at the interface;} \\ \Psi > 0, & \text{inside the high-density phase;} \\ \Psi < 0, & \text{elsewhere.} \end{cases}$$

The interface is implicitly advected, by advecting Ψ as if it would be a material property:

$$\frac{\partial \Psi}{\partial t} + \mathbf{u} \cdot \nabla \Psi = 0,$$

where $\mathbf{u}$ is the velocity vector in Ω. Therefore, ρ can be determined at each time step, without having the exact coordinates of the bubbles. For choosing deflation vectors in the deflation method, an extra procedure for determining the bubbles from Ψ should be carried out. For example, Algorithm 9 gives the pseudo-code of an algorithm for determining bubbles from a given level-set function, which can be used for 2-D problems on an equidistant grid [1] [2]. In this algorithm, $\hat{x}_i$ denotes an adjacent grid point of grid point x_i.

In Algorithm 9, three loops are needed to distinguish the bubbles from the rest of the domain and to include their adjacent grid points, requiring $\mathcal{O}(n)$ flops. Note that, in the case of deciding whether a grid point is in a bubble, we simply look at the sign of the corresponding element of Ψ. If the value is positive, the grid point is in the interior of the bubble, if it is negative, then it is outside the bubble, and, otherwise,

---

[1] John Brusche has contributed to the realization of this algorithm.

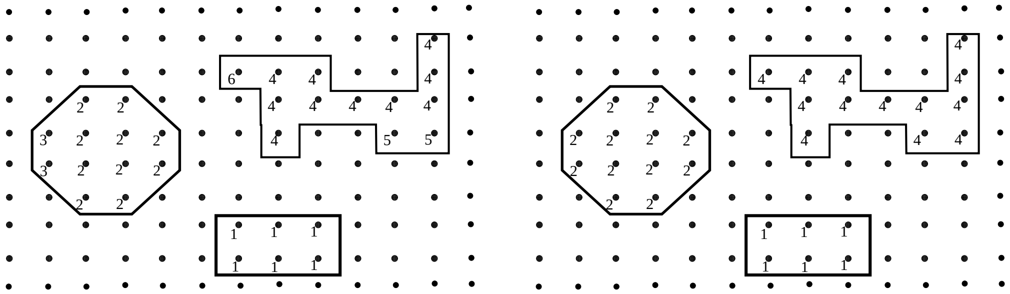
[2] If the computations are performed on an unstructured grid, similar algorithms as Algorithm 9 can be applied using reordering strategies, such as Cuthill McKee's algorithm [33].

Algorithm 9 Determination of bubbles from the level-set function in 2-D

```
Set $j = 1$ and $f = \mathbf{0}_n$;
for $x_1$ to $x_n$ (from left to right and from bottom to top) do
   if $x_i \in \Lambda_{h_1}$ then
      if left and/or bottom $\hat{x}_i \notin \Lambda_{h_1}$ then
         $f_i = j$;
         $j = j + 1$;
      else
         $f_i = \min_{\hat{x}_i} f$;
      end if
   end if
end for
for $x_n$ to $x_1$(from right to the left and from top to bottom) do
   if $x_i \in \Lambda_{h_1}$ then
      if right and/or top $\hat{x}_i \notin \Lambda_{h_1}$ then
         $f_i = j$;
         $j = j + 1$;
      else
         $f_i = \min_{\hat{x}_i} f$;
      end if
   end if
end for
Renumber all $f_i \neq 0$;
for $x_1$ to $x_n$ do
   if $x_i \in \Lambda_{h_1}$ and $\hat{x}_i \notin \Lambda_{h_1}$ then
      $f_{\hat{x}_i} = f_i$;
   end if
end for
```

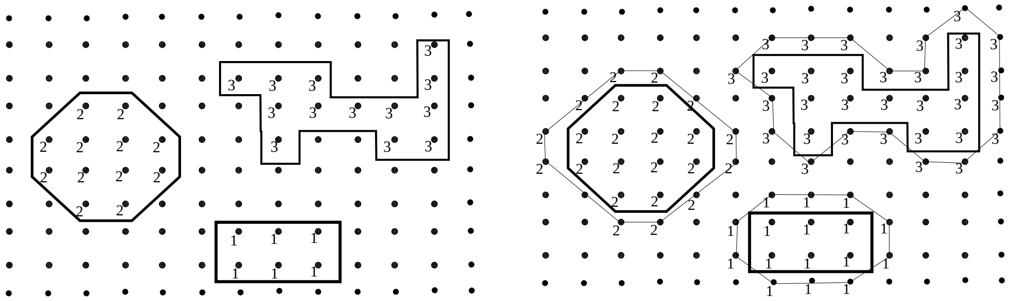
it is on the interface. In this way, it is straightforward to determine the bubbles from the level-set function, and to obtain a code where each deflation vector corresponds to exactly one bubble. The algorithm is further explained in Example B.1.

Example B.1. *A 2-D bubbly flow problem with* $m = 3$ *bubbles is considered, see Figure B.1. In each of the subplots, one can see the intermediate and final results of applying Algorithm 9 to determine each bubble.*



(a) After the first loop (Line 11).

(b) After the second loop (Line 21).



(c) After renumbering (Line 22).

(d) After the algorithm (Line 27).

Figure B.1: A 2-D bubbly flow problem with $m = 3$ showing the application of Algorithm 9. The numbers given in the plots are the corresponding nonzero entries of vector f.

Appendix C

More Insights into Deflation applied to Singular Coefficient Matrices

In addition to Chapter 5, we give some more results and insights into the application of the deflation method applied to linear systems with singular coefficient matrices [1]. The main focus of this appendix is on proving Theorem 5.5.

The following definition holds throughout this appendix.

Definition C.1. *Suppose that an SPD coefficient matrix, $A \in \mathbb{R}^{n\times n}$, is given. Let $Z \in \mathbb{R}^{n\times k}$ be a deflation-subspace matrix with full rank and $k < n$. Let $Z_a \in \mathbb{R}^{n\times k_a}$ be a deflation-subspace matrix with full rank and $k_a \leq k$ satisfying $\mathcal{N}(A) \cap \mathcal{R}(Z_a) = \emptyset$. Then, we define*

$$\left\{ \begin{array}{lllllllll} P & := & I - AQ, & Q & := & ZE^{+}Z^{T}, & E & := & Z^{T}AZ; \\ P_a & := & I - AQ_a, & Q_a & := & Z_aE_a^{-1}Z_a^{T}, & E_a & := & Z_a^{T}AZ_a. \end{array} \right.$$

Note that E can be singular, while E_a is obviously nonsingular (see Section 3.2). In addition, if there does not exist a vector, y, such that $y \subseteq \mathcal{R}(Z)$ for $y \in \mathcal{N}(A)$, then $P = P_a$.

C.1 Theoretical Results

We show that a deflation matrix based on a singular Galerkin matrix can always be reduced to a deflation matrix based on a nonsingular Galerkin matrix.

Theorem C.1. *Let $A \in \mathbb{R}^{n\times n}$ and $Z \in \mathbb{R}^{n\times k}$ be as given in Definition C.1. Then, there exists a matrix $Z_a \in \mathbb{R}^{n\times k_a}$ with $k_a \leq k$ such that E_a is invertible and*

$$Q = Q_a, \quad P = P_a. \tag{C.1}$$

---

[1] This appendix is based on research that is still ongoing, see [85].

Proof. Since E is SPSD, there exists an orthogonal matrix, $U \in \mathbb{R}^{k \times k}$, such that

$$U^T E U = \left[\begin{array}{cc} D_1 & \mathbf{0}_{k_a,(k-k_a)} \\ \mathbf{0}_{(k-k_a),k_a} & \mathbf{0}_{(k-k_a),(k-k_a)} \end{array} \right],$$

where $D_1 \in \mathbb{R}^{k_a \times k_a}$ with $k_a \leq k$ is a nonsingular diagonal matrix. Now, let

$$U = [U_1, U_2], \quad U_1 \in \mathbb{R}^{k \times k_a}, \quad U_2 \in \mathbb{R}^{k \times (k-k_a)}.$$

Then,

$$ZU = [ZU_1, \ ZU_2], \quad (ZU)^T = \left[\begin{array}{c} U_1^T Z^T \\ U_2^T Z^T \end{array} \right]. \tag{C.2}$$

Moreover, note that

$$Q = ZE^+Z^T = ZU(U^T E U)^+ U^T Z^T, \tag{C.3}$$

for each orthogonal matrix, $U \in \mathbb{R}^{k \times k}$. Combining Eqs. (C.2) and (C.3) yields

$$\begin{array}{rcl} Q = ZE^+Z^T & = & ZU(U^T E U)^+ U^T Z^T \\ & = & ZU \left(\left[\begin{array}{c} U_1^T Z^T \\ U_2^T Z^T \end{array} \right] E[ZU_1, ZU_2] \right)^+ U^T Z^T \\ & = & [ZU_1, ZU_2] \left[\begin{array}{cc} D_1^{-1} & \mathbf{0}_{k_a,(k-k_a)} \\ \mathbf{0}_{(k-k_a),k_a} & \mathbf{0}_{(k-k_a),(k-k_a)} \end{array} \right] \left[\begin{array}{c} U_1^T Z^T \\ U_2^T Z^T \end{array} \right], \\ & = & ZU_1 D_1^{-1} U_1^T Z^T \\ & = & ZU_1 (U_1^T E U_1)^{-1} U_1^T Z^T. \end{array}$$

Therefore, for $Z_a := ZU_1 \in \mathbb{R}^{k \times k_a}$, we have

$$Q = ZE^+Z^T = Z_a (Z_a^T A Z_a)^{-1} Z_a^T = Q_a,$$

so that the theorem follows immediately. ☐

Theorem C.1 shows that, for each SPSD coefficient matrix, A, and deflation-subspace matrix, Z, there exists a reduced deflation-subspace matrix, Z_a, such that E_a is nonsingular and deflation matrices P and P_a are equal.

Furthermore, the proof of Theorem C.1 also provides a technique to construct Z_a: choose $Z_a := ZU_1$ with U_1 consisting of eigenvectors of E corresponding to the nonzero eigenvalues. Note that these eigenvectors are orthogonal to the eigenvectors associated with the zero eigenvalues, so that $\mathcal{N}(A) \nsubseteq \mathcal{R}(U_1)$ and Z_a has full rank. Therefore, if $\mathcal{N}(E)$ is known, a basis of its orthogonal complement, say W, can be constructed such that $\mathcal{R}(U_1) = \mathcal{R}(W)$. Hence,

$$\mathcal{R}(ZU_1) = \mathcal{R}(ZW).$$

Thus, by using Theorem 3.2, ZW can also be applied as a reduced deflation-subspace matrix, resulting in Theorem C.2.

Theorem C.2. *Let $A \in \mathbb{R}^{n\times n}$ and $Z \in \mathbb{R}^{n\times k}$ be as given in Definition C.1. Suppose that E has rank k_a. Let $Z_s := ZW$ and Z_a be defined as in the proof of Theorem C.1. Suppose that W consists of basis vectors of the orthogonal complement of $\mathcal{N}(E)$. Then, $Z_s^T AZ_s$ is nonsingular and*

$$Q = Q_a = Z_s(Z_s^T AZ_s)^{-1} Z_s^T.$$

C.2 Proof of Theorem 5.5

Using the results of the previous section, we can prove Theorem 5.5 (that is equal to Theorem C.3), see below.

Theorem C.3. *Let $A \in \mathbb{R}^{n\times n}$, $Z \in \mathbb{R}^{n\times k}$ and $E \in \mathbb{R}^{k\times k}$ be as given in Definition C.1. Suppose that*

$$\begin{aligned} A\mathbf{1}_n &= \mathbf{0}_n; && \text{(C.4)}\\ Z\mathbf{1}_k &= \mathbf{1}_n; && \text{(C.5)}\\ \dim \mathcal{R}(E) &= k-1, && \text{(C.6)} \end{aligned}$$

and $Z_{k-1} = [z_1, \ldots, z_{k-1}]$. Let E_{k-1}, Q_{k-1} and P_{k-1} be as defined in Definition 5.3. Then, E_{k-1} is nonsingular and

$$AQA = AQ_{k-1}A.$$

Hence, (see Eq. (5.27))

$$M^{-1}PA = M^{-1}P_{k-1}A.$$

Proof. From Eqs. (C.4) and (C.5), we obtain $E\mathbf{1}_k = \mathbf{0}_k$, so that $\mathcal{N}(E) = \mathcal{R}(\mathbf{1}_k)$. Next, we choose a basis W of the orthogonal complement of $\mathcal{R}(\mathbf{1}_k)$. We take the vectors $\{w_i \in \mathbb{R}^k : i = 1, \ldots, k-1\}$, where

$$\left\{ \begin{array}{rcl} w_1 &=& \left[1, \frac{-1}{k-1}, \ldots, \frac{-1}{k-1}\right]^T, \\ w_2 &=& \left[\frac{-1}{k-1}, 1, \frac{-1}{k-1}, \ldots, \frac{-1}{k-1}\right]^T, \\ &\vdots& \\ w_{k-1} &=& \left[\frac{-1}{k-1}, \ldots, \frac{-1}{k-1}, 1\right]^T. \end{array} \right.$$

Eq. (C.5) gives us

$$z_1 = \mathbf{1}_n - \sum_{j=2}^{k} z_j,$$

which can be rewritten as

$$\frac{1}{k-1} z_1 = \frac{1}{k-1} \left(\mathbf{1}_n - \sum_{j=2}^{k} z_j \right). \tag{C.7}$$

On the other hand, we obtain

$$Zw_1 = z_1 - \sum_{j=2}^{k} \frac{1}{k-1} z_j. \tag{C.8}$$

Combining Eqs. (C.8) and (C.7) yields

$$Zw_1 = \left(1 + \frac{1}{k-1}\right) z_1 + \frac{1}{k-1} \mathbf{1}_n.$$

Similarly, we obtain

$$Zw_i = \left(1 + \frac{1}{k-1}\right) z_i + \frac{1}{k-1} \mathbf{1}_n, \quad i = 1, \ldots, k-1.$$

Hence,

$$ZW = [\alpha z_1, \ldots, \alpha z_{k-1}] + \frac{1}{k-1} \mathbf{1}_n \mathbf{1}_{n-1}^T, \quad \alpha := 1 + \frac{1}{k-1}. \tag{C.9}$$

Suppose now that

$$\bar{Z} := [\alpha z_1, \ldots, \alpha z_{k-1}].$$

Then, with Eq. (C.4), we obtain

$$W^T E W = \bar{Z}^T A \bar{Z},$$

while Theorem C.2 implies that $\bar{Z}^T A \bar{Z}$ is nonsingular. Using Eqs. (C.4) and (C.9), we obtain $AZW = A\bar{Z}$, so that

$$AZW(W^T E W)^{-1} W^T Z^T A = A\bar{Z}(\bar{Z}^T A \bar{Z})^{-1} \bar{Z}^T A.$$

Since $\mathcal{R}(\bar{Z}) = \mathcal{R}(Z_{k-1})$, Theorem 3.2 yields

$$AZW(W^T E W)^{-1} W^T Z^T A = A Z_{k-1} E_{k-1}^{-1} Z_{k-1}^T A. \tag{C.10}$$

On the other hand, we have

$$AQA = AZW(W^T E W)^{-1} W^T Z^T A \tag{C.11}$$

from Theorem C.2. Combining Eqs. (C.10) and (C.11) yields

$$AQA = AQ_{k-1}A,$$

which also implies

$$M^{-1}PA = M^{-1}P_{k-1}A.$$

□

Remark C.1.

- *From the proof of Theorem C.3, it is important to note that the following inequalities hold:*

  $$Q \neq Q_{k-1}, \quad AQ \neq AQ_{k-1}, \quad QA \neq Q_{k-1}A,$$

  while these would be equalities if E is nonsingular.

- *Several other interesting theoretical results for the deflation technique based on a singular Galerkin matrix can be found in [85].*

Appendix D

Efficient Implementation of Deflation Operations

In this appendix, we demonstrate the efficient computation of AZ and E, so that they can be easily incorporated in the deflation method. One can consult [140] for more details. Recall that $A \in \mathbb{R}^{n \times n}$ is a coefficient matrix, $Z \in \mathbb{R}^{n \times k}$ is the deflation-subspace matrix, and $E := Z^T AZ \in \mathbb{R}^{k \times k}$ is the Galerkin matrix. As discussed in Chapter 8, the nonzeros of these matrices are stored in the matrices S_{AZ} and S_E, respectively, whose exact forms are explained below. Moreover, Assumption D.1 holds throughout this appendix (see Assumption 8.1).

Assumption D.1.

- *A is derived after discretization of the Poisson problem that is originated from bubbly flow problems (see Section 1.3) and consists of 5 and 7 nonzero diagonals in the 2-D and 3-D case, respectively;*
- *Z consists of subdomain deflation vectors (see Section 4.2.3), where the subdomains are squares and cubes in the 2-D and 3-D case, respectively. In addition, the number of subdomains and deflation vectors is assumed to be equal.*

D.1 Efficient Construction of S_{AZ} and S_E in 2-D

Matrix $S_{AZ} \in \mathbb{R}^{\gamma \times 3}$ can be deduced from AZ, where γ is the number of nonzero entries of AZ. The first and second columns of S_{AZ} are the row and column indices of the nonzero entries of AZ, respectively. The third column of S_{AZ} stores their corresponding values.

Each deflation vector in Z corresponds to one subdomain in Ω. If we assume Ω to be a square, then these subdomains can be divided into nine different groups as depicted in Figure D.1. Note that all groups (except the corner groups 1, 3, 7, 9) may consist of more subdomains. For instance, for $k = 25$, Group 5 consists of exactly 16 subdomains, while each of Group 2, 4, 6 and 8 consists of 4 subdomains. Moreover, in Figure D.2, we can see the different cases and the grid points that are involved in the computation of S_{AZ}. In addition, the variables used in this section are explained in the Table D.1.

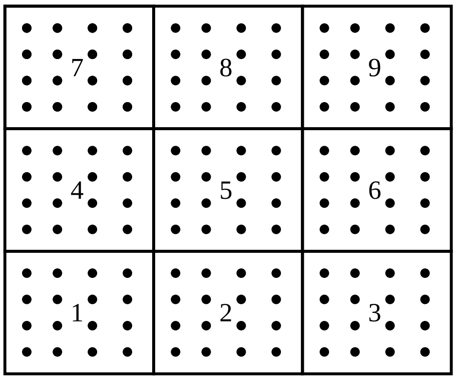


Figure D.1: Domain Ω divided into nine subdomains ($k = 9$), so that each subdomain corresponds to exactly one group.

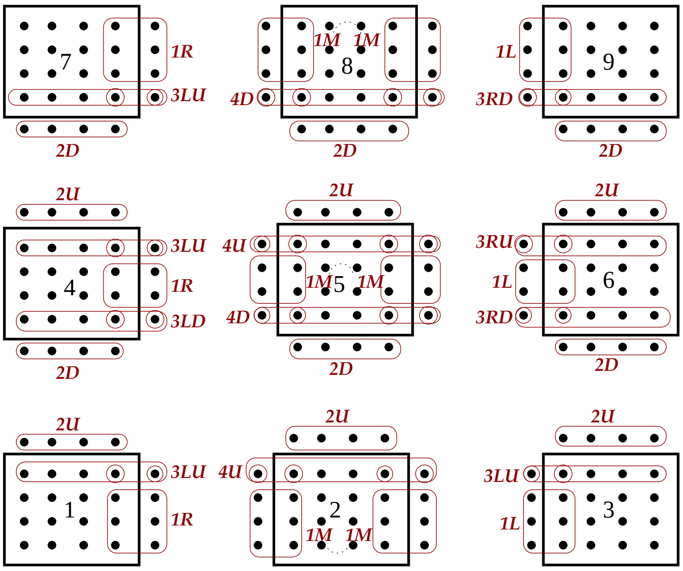


Figure D.2: Cases of grid points involved in the groups of S_{AZ}.

D.1.1 Number of Nonzero Entries in AZ

The number of nonzeros of AZ, γ, can be computed by counting the number of nonzeros for the different kinds of subdomains.

- *Corner Subdomains (Group 1, 3, 7, 9).* Each corner subdomain has nonzero contributions of $4n_b - 1$ grid points, so that $\gamma_c = 4(4n_b - 1)$.

Variable	Meaning
k	Number of subdomains
k_x	Number of subdomains in one direction ($=\sqrt{k}$ in the 2-D case)
n_b	Number of grid points in one direction of a subdomain
γ	Total number of nonzeros in AZ
γ_c	Number of nonzeros in AZ from all corner subdomains
γ_b	Number of nonzeros in AZ from all boundary subdomains
γ_i	Number of nonzeros in AZ from all interior subdomains

Table D.1: Explanation of the variables.

- *Boundary Subdomains (Group 2, 4, 6, 8).* Each boundary subdomain consists of $6n_b - 2$ involving grid points. Because we have $4(k_x - 2)$ boundary subdomains, $\gamma_b = 8(3n_b - 1)(k_x - 2)$ holds.

- *Interior Subdomains (Group 5).* $8n_b - 4$ grid points are involved per interior subdomain. Since there are $(k_x - 2)^2$ interior subdomains, this yields $\gamma_i = 4(2n_b - 1)(k_x - 2)^2$.

Now, γ is given by

$$\begin{aligned} \gamma &= \gamma_c + \gamma_b + \gamma_i \\ &= 4(4n_b - 1) + 8(3n_b - 1)(k_x - 2) + 4(2n_b - 1)(k_x - 2)^2. \end{aligned} \tag{D.1}$$

Obviously, if k is large, then γ_i is the dominant term in Eq. (D.1).

D.1.2 Treatment of the Different Cases

The different cases as presented in Figure D.2 are considered separately.

Case 1 (1R, 1L, 1M)

We distinguish the cases ‘left’ (L), ‘right’ (R) and ‘middle’ (M) variant in Case 1, where we note that the work is twice as much compared to ‘left’ or ‘right’ for variant ‘middle’. For each row of the domain, we add the values of A corresponding to the involved grid points to S_{AZ}, where it is sometimes efficient to use

$$-a_{i,j} = \sum_{k \neq j} a_{i,k}, \tag{D.2}$$

since $A\mathbf{1}_n = \mathbf{0}_n$ holds from Assumption 1.2. For instance, for the case 'left', we add two values of A to S_{AZ} in each row: for the first entry, y, we add the negative value of the corresponding right entry of A, and we add the corresponding left entry of A to S_{AZ} for the second entry, $y + 1$.

Case 2 (2U, 2D)

Two variants ‘up’ (U) and ‘down’ (D) are distinguished for this case. The corresponding entries of S_{AZ} can be easily computed: we add the corresponding bottom entry of A for ‘up’, while the top entry of A is added to S_{AZ} in the case of ‘down’.

Case 3 (3LU, 3LD, 3RU, 3RD, 3MU, 3MD)

This case consists of six different variants. Each variant requires a sequence of operations, since the ‘corner’ points have to be treated differently compared to the ‘boundary’ points. For instance, Variant ‘3LU’ requires the computations of the boundary points, followed by two corner points that should be treated separately, both using again (D.2).

Case 4 (4D, 4U)

We distinguish the variants ‘4D’ and ‘4U’, whose treatment is analogous to the procedure of the variants in Case 3. Instead of two corner points, we now have four corner points that should be handled separately.

D.1.3 Construction of S_{AZ}

The computation of S_{AZ} is straightforward by using the cases as described above. Each subdomain is handled by determining the entries corresponding to each case. If $k = 4$, then four subdomains should be considered with three cases each. In the case of $k > 4$, Group 2 (or 4, 6, 8) in Figure 2 consists of $k_x - 2$ subdomains, while Group 5 consists of $(k_x - 2)^2$ subdomains.

Moreover, recall that if the singular coefficient matrix, A, is made invertible according to Definition 5.2, then S_{AZ} consists of an extra row. Combining the facts $(AZ)_{n,n} = \sigma a_{n,n}$ and $\tilde{a}_{n,n} = (1 + \sigma) a_{n,n}$, we obtain

$$(AZ)_{n,n} = \frac{\sigma}{1+\sigma} \tilde{a}_{n,n}.$$

D.1.4 Construction of S_E

The Galerkin matrix, E, is a relatively small and sparse SPD matrix with the same nonzero pattern as A. The different cases in the computations of S_{AZ} can also be applied to construct S_E. Obviously, each nonzero entry of AZ is used once in order to compute S_E. The geometry of the procedure is given in Figure D.3, where the following remarks can be made.

- E is symmetric, so that only a limited number of nonzero entries of AZ is required to compute S_E.
- S_E is stored efficiently as $S_E := [e_1 \; e_2 \; e_3] \in \mathbb{R}^{k \times 3}$, where e_1 is the main diagonal of E, and e_2 and e_3 are the second and third nonzero subdiagonals of E, respectively. All indicated interior grid points contribute to e_1, while all

right and top grid points next to the interior grid points contribute to e_2 and e_3, respectively. Later on, zero columns can be added between e_2 and e_3, which can be filled with entries coming from the Cholesky decomposition.

- The construction of S_E can be easily implemented in the existing code of the computation of S_{AZ}.

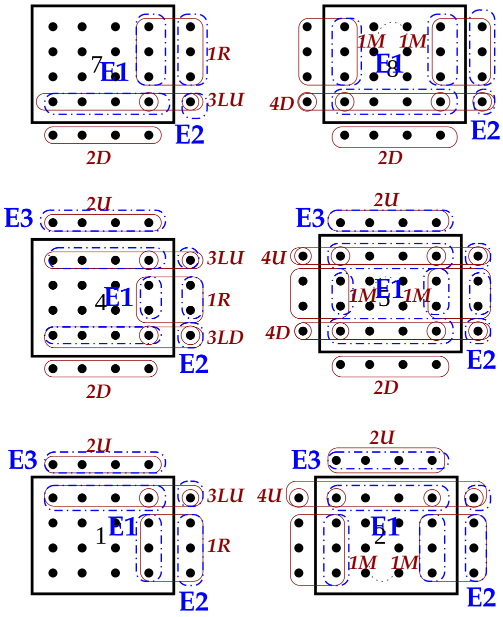


Figure D.3: Cases of grid points involved in $E := Z^T AZ$, denoted by E1, E2 and E3, whose values in AZ contribute to e_1, e_2 and e_3, respectively.

D.2 Efficient Construction of S_{AZ} and S_E in 3-D

The results from the previous section can be generalized to the 3-D case. Each subdomain of Figure D.2 takes the form of a block in this case. These blocks are numbered lexicographically. We demonstrate the efficient computation of S_{AZ} and S_E, where the analysis is based on 9 subdomains followed by 27 and more subdomains.

D.2.1 Number of Nonzero Entries in AZ

Similarly to the 2-D case, the number of nonzero entries of AZ, γ, can be easily computed, see below.

- *Corner Blocks.* It is easy to see that $6n_b^2 - 3n_b + 1$ nonzero entries are involved for each of the eight corner blocks, so that $\gamma_c = 8(6n_b^2 - 3n_b + 1)$.
- *Interior Blocks.* $12n_b^2 - 12n_b + 8$ nonzero entries are involved for each interior block. This implies $\gamma_i = (k_x - 2)^3(12n_b^2 - 12n_b + 8)$, because we have $(k_x - 2)^3$ interior blocks.

- *Boundary Blocks.* We divide the boundary blocks into 'real-boundary' blocks and 'boundary-interior' blocks. Each of the $12(k_x-2)$ real-boundary blocks has $8n_b^2-5n_b+2$ nonzero entries, whereas each of the $6(k_x-2)^2$ boundary-interior blocks requires $10n_b^2-8n_b+4$ entries. Hence, $\gamma_b = 12(k_x-2)8n_b^2-5n_b+2+60(k_x-2)^2n_b^2-8n_b+4$.

The total number γ can now again be computed using $\gamma = \gamma_c+\gamma_i+\gamma_b$. As in the 2-D case, one extra row is required for S_{AZ}, if A is forced to be invertible using Definition 5.2.

D.2.2 Matrix S_{AZ} for Eight Blocks

In the case of eight subdomains, we only need Groups 1, 3, 7 and 9 of Figure D.2, which are sections of corresponding blocks in 3-D. We treat Block 1 extensively. The remaining blocks can be analyzed in a similar way.

Block 1

Block 1 is artificially divided into layers, where each layer corresponds to one position on the z-axis. Note that the layers of $z=1,\ldots,n_b-1$ are identical, see Figure D.4. As a consequence, they are the same as Block 1 in the 2-D case. For layer $z=n_b$, each grid point of the block accounts for an extra entry (at $z=n_b+1$). This requires the introduction of 'Case 6' that encounters for these entries. In addition, Cases 1 and 3 are not required anymore. Finally, for layer $z=n_b+1$, each interior grid point of the block has a contribution that is treated in Case 5.

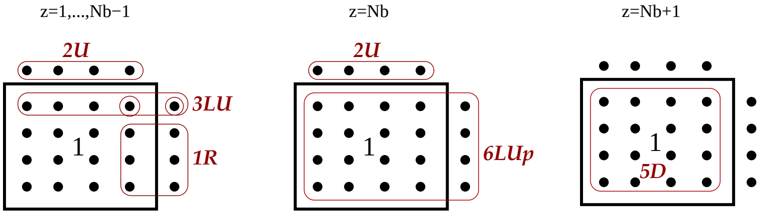


Figure D.4: Treatment of Block 1.

Other Blocks

The remaining blocks can be treated in the same way as Block 1. In Table D.2, we summarize the involved cases for each of these blocks.

D.2.3 Matrix S_{AZ} for 27 Blocks

For $k=27$, the eight blocks from the previous subsection are the eight corner blocks. The remaining 19 blocks can be constructed in a straightforward way. The different cases for each block are considered below.

z-position	Block 1	Block 2	Block 3	Block 4
$1, \ldots, n_b - 1$	1R, 3LU, 2U	1L, 3RU, 2U	2U, 3LD, 1R	2D, 3RD, 1L
n_b	6LUp, 2U	6RUp, 2U	2U, 6LDp	2D, 6RDp
$n_b + 1$	5D	5D	5D	5D

z-position	Block 5	Block 6	Block 7	Block 8
$1, \ldots, n_b - 1$	5U	5U	5U	5U
n_b	6RUn, 2U	6RUn, 2U	2D	2D, 6RDn
$n_b + 1$	1L, 3RU, 2U	1L, 3RU, 2U	6LDn	2D, 3RD, 1L

Table D.2: Cases involved in the blocks for eight subdomains in 3-D.

Blocks 1–9

The treatment of Blocks 1–9 is presented in Table D.3.

z-position	Block 1	Block 2	Block 3
$1, \ldots, n_b - 1$	1R, 3LU, 2U	1M, 4U, 2U	2U, 3LD, 1R
n_b	6LUp, 2U	6MUp, 2U	2U, 6LDp
$n_b + 1$	5D	5D	5D

z-position	Block 4	Block 5	Block 6
$1, \ldots, n_b - 1$	2D, 3LD, 1R, 3LU, 2U	2D, 4D, 1M, 4U, 2U	2D, 3RD, 1L, 3RU, 2U
n_b	2D, 6LMp, 2U	2D, 6MMp, 2U	2D, 6MMp, 2U
$n_b + 1$	5D	5D	5D

z-position	Block 7	Block 8	Block 9
$1, \ldots, n_b - 1$	2U, 3LD, 1R	2D, 4D, 1M	2D, 3RD, 1L
n_b	2U, 6LDp	2D, 6MDp	2D, 6RDP
$n_b + 1$	5D	5D	5D

Table D.3: Cases involved in Blocks 1–9 for $k = 27$ in the 3-D case.

Blocks 10–18

Blocks 10–18 can be constructed from Blocks 1–9. Instead of three, we obviously have five different layers. The last three layers are the same as the block on the bottom of these layers, while the first two blocks follow immediately from the last two blocks. For example, Block 10 consists of

$$z = \begin{cases} n_b : & \text{5U;} \\ n_b + 1 : & \text{6LUn;} \\ n_b + 2, \ldots, 2n_b - 1 : & \text{1R, 3LU, 2U;} \\ 2n_b : & \text{6LUp, 2U;} \\ 2n_b + 1 : & \text{5D.} \end{cases}$$

Therefore, the cases for the last three layers with respect to the z-position (i.e., $z = n_b + 2, \ldots, 2n_b + 1$) are exactly the same as given for Block 1. In addition, the cases associated with the first two layers (i.e., $z = n_b, n_b + 1$) are almost identical to the cases corresponding to the last two cases (i.e., $z = 2n_b, 2n_b + 1$), where 'p' is replaced by 'n' in Case 6. A similar pattern of cases can be derived for Blocks 10–18.

Blocks 19–27

Blocks 19–27 also follow immediately from Blocks 1–9. The different layers of Blocks 1–9 should be reversed and, moreover, 'p' should be replaced by 'n' in Case 6, while 'D' should be replaced by 'U' in Case 5. For instance, Block 20 consists of

$$z = \begin{cases} 2n_b : & \text{5U;} \\ 2n_b + 1 : & \text{6MUn; 2U;} \\ 2n_b + 2, \ldots, 3n_b : & \text{1M, 4U, 2U;} \end{cases}$$

This is exactly the reverse procedure of Block 2, where 5D and 6LUp are now 5U and 6LUn, respectively. In a similar way, the other blocks can be analyzed.

D.2.4 Matrix S_{AZ} with Variable Number of Blocks

The determination of matrix S_{AZ} with a variable number of blocks is a straightforward generalization of the case with 27 blocks as described above. Each of the 27 blocks should now be considered as different classes, which cover all new blocks.

D.2.5 Construction of S_E

Matrix S_E is constructed in the same way as the 2-D case. Instead of $S_E = [e_1 \; e_2 \; e_3]$, we now have $S_E = [e_1 \; e_2 \; e_3 \; e_4]$, where e_4 can be computed similarly to e_2 and e_3.

Appendix **E**

Flop Counts for the Deflation Method

In this appendix, we compare the floating-point operations (flops) of ICCG and DICCG in more detail. The DICCG1 and DICCG2 methods (see Section 8.3), which only differ in the inner-iteration solver, are examined. We restrict ourselves to the 3-D case; a thorough 2-D analysis can be found in [140]. Moreover, the following assumption holds throughout this appendix.

Assumption E.1.

- *$A \in \mathbb{R}^{n\times n}$ consists of 7 nonzero diagonals;*
- *M^{-1} is the IC(0) preconditioner (see Section 2.5.1), so that the resulting deflation method is DICCG;*
- *$Z \in \mathbb{R}^{n\times k}$ consists of subdomain deflation vectors (see Section 4.2.3) where $k \ll n$;*
- *AZ is computed and stored efficiently as S_{AZ} (see Appendix D.2);*
- *$E \in \mathbb{R}^{k\times k}$ has bandwidth $k_x^2 + k_x = k^{\frac{2}{3}} + k^{\frac{1}{3}}$, and it is computed and stored efficiently as S_E (see Appendix D.2).*

Assumption E.1 leads to fairly standard results as given in Table E.1, where $F_\times$ denotes the number of flops required for a specific operation $\times$, and chol(A) is the Cholesky factor, C, that satisfies $A = CC^T$.

Notation	Operation	# Flops
$F_{(y_1,y_2)}$	(y_1, y_2)	$2n$
$F_{y_1+y_2}$	$y_1 + y_2$	n
F_{Ay}	Ay	$13n$
$F_{\mathrm{chol}(A)}$	construct C from A	$12n$
$F_{Ay_2=y_1}$	solve y_2 from $CC^T y_2 = y_1$	$15n$

Table E.1: Results of flop counts for standard operations.

E.1 Deflation Operations

The flop counts for some operations in the deflation method are presented below.

Computation of S_{AZ} and S_E

According to Appendix D, the number of rows of S_{AZ} in the 3-D case is given by $\gamma = \gamma_c + \gamma_i + \gamma_b$, where

$$\begin{cases} \gamma_c &=& 8(6n_b^2 - 3n_b + 1); \\ \gamma_i &=& (k_x - 2)^3(12n_b^2 - 12n_b + 8); \\ \gamma_b &=& 76(k_x - 2)n_b^2 - 5n_b + 2 + 60(k_x - 2)^2 n_b^2 - 8n_b + 4. \end{cases} \tag{E.1}$$

Substituting $n_b = \sqrt[3]{\frac{n}{k}}$ and $k_x = \sqrt[3]{k}$ into (E.1) and rearranging some terms, we obtain

$$\begin{cases} \gamma_c &\approx& 48\left(\frac{n}{k}\right)^{\frac{2}{3}}; \\ \gamma_i &\approx& 12n^{\frac{2}{3}}k^{\frac{1}{3}} + 8k; \\ \gamma_b &\approx& 60n^{\frac{2}{3}}, \end{cases}$$

resulting in

$$\gamma \approx 12n^{\frac{2}{3}}k^{\frac{1}{3}} + 60n^{\frac{2}{3}} + 48\left(\frac{n}{k}\right)^{\frac{2}{3}} + 8k.$$

Recall from Appendix D that S_{AZ} and S_E can be constructed with the same cost. The number of flops to create AZ and E, denoted by F_{AZ} and F_E, is

$$F_E \approx F_{AZ} \approx 12n^{\frac{2}{3}}k^{\frac{1}{3}} + 60n^{\frac{2}{3}} + 48\left(\frac{n}{k}\right)^{\frac{2}{3}} + 8k \approx \mathcal{O}(n^{\frac{2}{3}}k^{\frac{1}{3}}).$$

This latter expression can also be obtained by observing that $\gamma_i \gg \gamma_b + \gamma_c$ for sufficiently large k, so that the contributions of γ_b and γ_c could be neglected. Moreover, the construction of S_{AZ} and S_E in the 3-D case ($\mathcal{O}(n^{\frac{2}{3}}k^{\frac{1}{3}})$ flops) is clearly more expensive than in the 2-D case ($\mathcal{O}(n^{\frac{1}{3}})$ flops).

Computation of $(AZ)^T y_1$, $Z^T y_1$, $(AZ)y_2$ and Zy_2

We easily derive

$$F_{AZy_2} = F_{(AZ)^T y_1} = 2\gamma \approx \mathcal{O}(n^{\frac{2}{3}}k^{\frac{1}{3}}),$$

and

$$F_{Zy_2} = F_{Z^T y_1} = n = \mathcal{O}(n).$$

In contrast to the 2-D case, the difference of cost between the computations of $(AZ)y_1$ and Zy_1 is relatively small.

Computation of Solving $Ey_2 = y_1$

The Galerkin system, $Ey_2 = y_1$, is solved differently in DICCG1 and DICCG2, see below.

- *DICCG1: Solving $Ey_2 = y_1$ directly.* The factor, L, of band-Cholesky decomposition is constructed from E, followed by solving y_2 from $LL^T y_2 = y_1$. Since the bandwidth of E is $k^{\frac{2}{3}} + k^{\frac{1}{3}}$, we obtain (cf. [137, Appendix B])

  $$F_{\mathrm{chol}(E)} = k\left((k^{\frac{2}{3}} + k^{\frac{1}{3}})^2 + 3(k^{\frac{2}{3}} + k^{\frac{1}{3}})\right) + k \approx \mathcal{O}(k^{\frac{7}{3}}),$$

  and

  $$F_{Ey_2=y_1,\mathrm{DICCG1}} = k\left(2(k^{\frac{2}{3}} + k^{\frac{1}{3}}) + 1\right) \approx \mathcal{O}(k^{\frac{5}{3}}).$$

- *DICCG2: Solving $Ey_2 = y_1$ iteratively.* If $Ey_2 = y_1$ is solved using ICCG, it can be easily shown that

  $$F_{Ey_2=y_1,\mathrm{DICCG2}} = 31k + 36kI_{\mathrm{ICCG}},$$

  where I_{ICCG} is the number of inner iterations of ICCG.

Clearly, it depends on k and I_{ICCG} whether DICCG1 or DICCG2 is the most efficient method.

Computations of Py and PAy

Obviously, F_{Py} and F_{PAy} depend on the choice of DICCG1 or DICCG2. We obtain

$$\begin{aligned} F_{Py,\mathrm{DICCG1}} &= F_{Z^T y_1} + F_{Ey_2=y_1,\mathrm{DICCG1}} + F_{(AZ)y_2} + F_{y-y_3} \\ &= n + 2k^{\frac{5}{3}} + 2k^{\frac{4}{3}} + k + 24n^{\frac{2}{3}}k^{\frac{1}{3}} + 120n^{\frac{2}{3}} + 96\left(\tfrac{n}{k}\right)^{\frac{2}{3}} + 16k + n, \end{aligned}$$

and

$$\begin{aligned} F_{Py,\mathrm{DICCG2}} &= F_{Z^T y_1} + F_{Ey_2=y_1,\mathrm{DICCG2}} + F_{(AZ)y_2} + F_{y-y_3} \\ &= n + 31k + 36kI_{\mathrm{ICCG}} + 24n^{\frac{2}{3}}k^{\frac{1}{3}}. \end{aligned}$$

Moreover, we have

$$\begin{aligned} F_{PAy,\mathrm{DICCG1}} &= F_{Py,\mathrm{DICCG1}} + F_{Ay} \\ &= 15n + 24n^{\frac{2}{3}}k^{\frac{1}{3}} + 120n^{\frac{2}{3}} + 96\left(\tfrac{n}{k}\right)^{\frac{2}{3}} + 2k^{\frac{5}{3}} + 2k^{\frac{4}{3}} + 17k, \end{aligned}$$

and

$$\begin{aligned} F_{PAy,\mathrm{DICCG2}} &= F_{Py,\mathrm{DICCG2}} + F_{Ay} \\ &= 15n + 24n^{\frac{2}{3}}k^{\frac{1}{3}} + 120n^{\frac{2}{3}} + 47k + 36kI_{\mathrm{ICCG}} + 96\left(\tfrac{n}{k}\right)^{\frac{2}{3}}. \end{aligned}$$

Hence, both F_{Py} and F_{PAy} for DICCG1 and DICCG2 require $\mathcal{O}(n)$ flops.

E.2 ICCG, DICCG1 and DICCG2

In this section, we first determine the number of flops required before and after the iteration process of ICCG, DICCG1 and DICCG2. Thereafter, we compute the number of flops required within the iteration process. Finally, the total flops of the methods are determined.

Computations outside the Iteration Process

Compared to ICCG, both DICCG1 and DICCG2 methods need some additional work before and after the iteration process, whose number of flops is determined below. $F_{\text{prior, DICCG1}}$ and $F_{\text{prior, DICCG2}}$ denote the extra flops for DICCG1 and DICCG2 prior to the iteration process, whereas $F_{\text{after, DICCG1}}$ and $F_{\text{after, DICCG2}}$ denote the extra flops for DICCG1 and DICCG2 required after the iteration process, respectively. We obtain the following results:

$$\begin{aligned} F_{\text{prior, DICCG1}} &= F_{AZ} + F_E + F_{\text{chol}(E)} \\ &= 24 n^{\frac{2}{3}} k^{\frac{1}{3}} + 120 n^{\frac{2}{3}} + 96 \left(\tfrac{n}{k}\right)^{\frac{2}{3}} + 16k + k^{\frac{7}{3}} + 2k^2 + 4k^{\frac{5}{3}} + 3k^{\frac{4}{3}} + k, \end{aligned}$$

and

$$\begin{aligned} F_{\text{prior, DICCG2}} &= F_{AZ} + F_E \\ &= 24 n^{\frac{2}{3}} k^{\frac{1}{3}} + 120 n^{\frac{2}{3}} + 96 \left(\tfrac{n}{k}\right)^{\frac{2}{3}} + 16k. \end{aligned}$$

Subsequently, in order to compute $F_{\text{after, DICCG1}}$ and $F_{\text{after, DICCG2}}$, we determine the number of flops for the computation of Qy:

$$\begin{aligned} F_{Qy,\text{DICCG1}} &= F_{Z^T y_1} + F_{Ey_2 = y_1,\text{DICCG1}} + F_{Zy_2} \\ &= 2n + 2k^{\frac{5}{3}} + 2k^{\frac{4}{3}} + k, \end{aligned}$$

$$\begin{aligned} F_{Qy,\text{DICCG2}} &= F_{Z^T y_1} + F_{Ey_2 = y_1,\text{DICCG2}} + F_{Zy_2} \\ &= 2n + 31k + 36k l_{\text{ICCG}}. \end{aligned}$$

Hence, this yields

$$\begin{aligned} F_{\text{after, DICCG1}} &= F_{Qy,\text{DICCG1}} + F_{P^T y,\text{DICCG1}} + F_{y_1+y_2} \\ &= 2n + 2k^{\frac{5}{3}} + 2k^{\frac{4}{3}} + k + 2n + 24 n^{\frac{2}{3}} k^{\frac{1}{3}} + 120 n^{\frac{2}{3}} + \\ &\quad 96 \left(\tfrac{n}{k}\right)^{\frac{2}{3}} + 2k^{\frac{5}{3}} + 2k^{\frac{4}{3}} + 17k, \end{aligned}$$

$$\begin{aligned} F_{\text{after, DICCG2}} &= F_{Qy,\text{DICCG2}} + F_{P^T y,\text{DICCG2}} + F_{y_1+y_2} \\ &= 2n + 31k + 36k l_{\text{ICCG}} + 16n + 24 n^{\frac{2}{3}} k^{\frac{1}{3}} + 120 n^{\frac{2}{3}} + \\ &\quad 47k + 36k l_{\text{ICCG}} + 96 \left(\tfrac{n}{k}\right)^{\frac{2}{3}}. \end{aligned}$$

As a result, F_{prior} is $\mathcal{O}(n^{\frac{2}{3}} k^{\frac{1}{3}})$, whereas F_{after} is $\mathcal{O}(n)$ for both DICCG1 and DICCG2.

Furthermore, ICCG, DICCG1 and DICCG2 have several computations in common, whose number of flops is denoted by $F_{\text{common-out}}$. It is easy to see that

$$F_{\text{common-out}} = 39n.$$

Computations within the Iteration Process

We compute the number of flops that is involved in the iteration process of ICCG, DICCG1 and DICCG2. The flops of their common operations,

$$F_{\text{common-in}} = 31n,$$

can be easily derived. Next, the difference between ICCG and DICCG1/DICCG2 within the iteration process is computing $w_j := Ap_j$ and $\bar{w}_j := PAp_j$ (cf. Algorithms 3 and 6). Combining the facts

$$\begin{array}{rcl} F_{PAp,\text{DICCG1}} & = & F_{PAy,\text{DICCG1}} \\ & = & 15n + 24n^{\frac{2}{3}}k^{\frac{1}{3}} + 120n^{\frac{2}{3}} + 96\left(\frac{n}{k}\right)^{\frac{2}{3}} + 2k^{\frac{5}{3}} + 2k^{\frac{4}{3}} + 17k, \end{array}$$

and

$$\begin{array}{rcl} F_{PAp,\text{DICCG2}} & = & F_{PAy,\text{DICCG2}} \\ & = & 15n + 24n^{\frac{2}{3}}k^{\frac{1}{3}} + 120n^{\frac{2}{3}} + 47k + 36kI_{\text{ICCG}} + 96\left(\frac{n}{k}\right)^{\frac{2}{3}}, \end{array}$$

with

$$F_{Ap,\text{ICCG}} = F_{Ay,\text{ICCG}} = 13n,$$

we deduce that both F_{PAp} and F_{Ap} are $\mathcal{O}(n)$.

Total Number of Flops for ICCG, DICCG1 and DICCG2

Using the above results, the total number of flops for ICCG is

$$\begin{array}{rcl} F_{\text{ICCG}} & = & F_{\text{common-out}} + (F_{\text{common-in}} + F_{Ap})I_{\text{ICCG}} \\ & = & 39n + (31n + 13n)I_{\text{ICCG}} \\ & = & 39n + 44nI_{\text{ICCG}}, \end{array}$$

while, for DICCG1 and DICCG2, we obtain

$$\begin{array}{rcl} F_{\text{DICCG1}} & = & F_{\text{prior, DICCG1}} + F_{\text{common-in}} + F_{\text{after, DICCG1}} + \left(F_{\text{common-in}} + F_{PAp,\text{DICCG1}}\right) I_{\text{DICCG1}} \\ & = & 43n + 47n^{\frac{2}{3}}k^{\frac{1}{3}} + 240n^{\frac{2}{3}} + 192\left(\frac{n}{k}\right)^{\frac{2}{3}} + k^{\frac{7}{3}} + 2k^2 + 8k^{\frac{5}{3}} + 7k^{\frac{4}{3}} + 25k + \\ & & \left(46n + 24n^{\frac{2}{3}}k^{\frac{1}{3}} + 120n^{\frac{2}{3}} + 96\left(\frac{n}{k}\right)^{\frac{2}{3}} + 2k^{\frac{5}{3}} + 2k^{\frac{4}{3}} + 17k\right) I_{\text{DICCG1}}, \end{array}$$

and

$$\begin{array}{rcl} F_{\text{DICCG2}} & = & F_{\text{prior, DICCG2}} + F_{\text{common-in}} + F_{\text{after, DICCG2}} + \left(F_{\text{common-in}} + F_{PAp,\text{DICCG2}}\right) I_{\text{DICCG2}} \\ & = & 57n + 58k + 48n^{\frac{2}{3}}k^{\frac{1}{3}} + 240n^{\frac{2}{3}} + 192\left(\frac{n}{k}\right)^{\frac{2}{3}} + 16k + 72kI_{\text{DICCG2}} + \\ & & \left(46n + 24n^{\frac{2}{3}}k^{\frac{1}{3}} + 120n^{\frac{2}{3}} + 47k + 36kI_{\text{ICCG}} + 96\left(\frac{n}{k}\right)^{\frac{2}{3}}\right) I_{\text{DICCG2}}. \end{array}$$

Obviously, it depends on the exact values of k and n, and the number of inner/outer iterations which of the ICCG, DICCG1 and DICCG2 methods is the most efficient one.

Appendix F

Parallel Version of the Deflation Method

Parallel computing is fast becoming an inexpensive alternative to the standard supercomputer approach for solving large linear systems, see, e.g., [4, 36, 44, 120]. The main operations to be parallelized for Krylov iterative methods, in particular the deflation method, are:

(a) matrix-vector multiplications;

(b) vector updates;

(c) dot products;

(d) preconditioning setup and operations;

(e) deflation setup and operations.

The potential bottlenecks are setting up the preconditioner and solving linear systems with the preconditioner (Operation (d)). On the other hand, deflation setup and operations (Operation (e)) can be easily parallelized if subdomain deflation vectors are used, which is explained in Section F.2. In addition, we note that the dot-product operation (c) might be troublesome in computational applications, since all processors must synchronize and perform communication before computations can be continued (at least for parallel synchronous iterative methods).

Since Operation (d) might be rather complicated in the parallel approach, we treat this in more detail. Preconditioners are considered based on Schur-complement and nonoverlapping additive-Schwarz (also known as additive Schwarz with minimum overlap) methods. Equivalently, instead of the traditional DICCG method, we consider the Deflated PCG method with a block-Jacobi preconditioner, M_{BJ}^{-1}, or a block-IC(0) preconditioner, M_{BIC}^{-1}, see Section F.1 and also [137].

F.1 Traditional Parallel Preconditioners

Recall that (see Section 2.5.1)

$$M_{\text{BJ}} = \begin{pmatrix} A_1 & & & \emptyset \\ & A_2 & & \\ & & \ddots & \\ \emptyset & & & A_p \end{pmatrix}, \tag{F.1}$$

where A_i denotes the i-th diagonal block of the coefficient matrix, A. It can be shown that solving $M_{\text{BJ}} y_2 = y_1$ accurately is equivalent to the Schur-complement approach. In practice, each submatrix, A_i, can be relatively large, so that it might be attractive to solve each subsystem of $M_{\text{BJ}} y_2 = y_1$ inaccurately. This latter approach is equivalent to solving linear systems with the additive-Schwarz preconditioner, see also [126]. On the other hand, the incomplete Cholesky (IC(0)) decomposition of the blocks of M_{BJ} can also be used to construct an efficient parallel preconditioner. The resulting preconditioner is called the block-IC preconditioner, denoted by M_{BIC}^{-1}. Since there is no overlap between the blocks in any of the above described preconditioners, the corresponding preconditioning steps are well-parallelizable.

F.2 Parallel Deflation

In this section, we describe concisely in which way Operation (e) can be carried out efficiently in a parallel environment, see [56] for more details. We restrict ourselves to subdomain deflation, where each available processor corresponds to one subdomain and a fixed number of unknowns. For convenience, one subdomain per processor is assumed. The coupling with neighboring subdomains is realized by the use of virtual grid points added to the local grids. In this way, a block-row of the linear system, $Ax = b$, corresponding to the subdomain ordering

$$A = \begin{pmatrix} A_{11} & \cdots & A_{1p} \\ \vdots & & \vdots \\ A_{p1} & \cdots & A_{pp} \end{pmatrix},$$

can be represented locally on one processor: the diagonal block, A_{ii}, represents coupling between local unknowns of subdomain Ω_i, and the off-diagonal blocks of Ω_i represent coupling between local unknowns and the virtual grid points.

For the deflation operations in parallel, we first compute and store successively the matrices E and E^{-1} on each processor, whereas AZ is computed and stored locally. The use of P within the deflation method involves the operation $y_2 = PAy_1$, which consists of the following operations:

- the matrix-vector multiplication $x_1 := Ay_1$, requiring nearest neighbor communications;

- the local contribution to the restriction $x_2 := Z^T x_1$, which should be distributed to all processors;
- a coarse-grid operation, $x_3 := E^{-1} x_2$, that is locally determined;
- $y_2 := I - AZx_3$, which is also determined locally.

The total communication involved in $y_2 = PAy_1$ is a nearest neighbor communication of the length of the interface.

For a five-point discretization of PDEs (i.e., standard 2-D problems), it can be verified that the added iteration expense of deflation is less expensive than an IC(0) factorization, and the resulting parallel method can be implemented efficiently on a distributed memory computer, see [56, Sect. 5]. More research is required to investigate this issue for more-points discretization and 3-D problems.

It is well-known that the overlapping of subdomains in the traditional preconditioner makes the parallel iterative method more-or-less independent of the subdomain grid size, but overlapping is not always easy to implement on top of an existing software package. In order to make a parallel iterative method more robust, one could apply the deflation technique as suggested above. In [56], it is shown that only a slow increase of the number of iterations can be observed when the subdomain grid size is constant and the number of (deflation) subdomains increases. Additionally, for a fixed global grid, the number of iterations even decreases if the number of processors grows, see [56, 137].

Finally, in order to improve the parallel deflation method, one can also apply the deflation technique on the local level. If the block-Jacobi preconditioner is used, then solving $M_{\text{BJ}} y_2 = y_1$ consists of solving subsystems of the form

$$A_i (y_2)_i = (y_1)_i, \quad i = 1, 2, \ldots, p. \tag{F.2}$$

If this is done iteratively, the convergence could be improved by solving the deflated subsystems of the form

$$P_i A_i (y_2)_i = P_i (y_1)_i, \quad i = 1, 2, \ldots, p,$$

instead of (F.2), where each smaller local deflation matrix, P_i, is based on the grid points of the specific subdomain. The overall deflation method can then be interpreted as a twofold deflation method, which is still a topic of current research.

Appendix G

Two-Level PCG Methods applied to Porous-Media Flows

As an extension of Section 6.4, a numerical comparison of two-level PCG methods for 2-D porous-media flows is performed in this appendix. This is a continuation of Section 6.4, where the same comparison has been done for 2-D bubbly flows.

G.1 Problem Setting

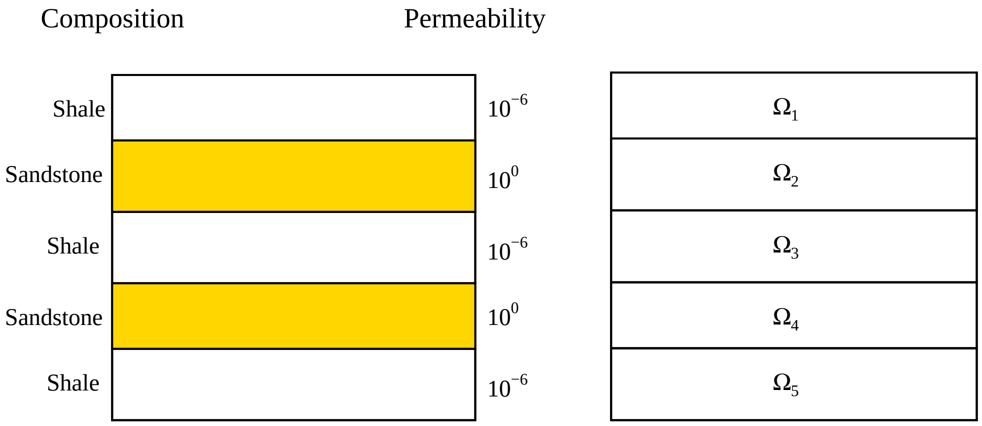
We solve the linear system, $Ax = b$, which is derived after discretization of the Poisson equation with a discontinuous coefficient,

$$-\nabla \cdot (\sigma(\mathbf{x})\nabla p(\mathbf{x})) = \mathbf{0}, \quad \mathbf{x} = (x, y) \in \Omega = (0, 1)^2, \tag{G.1}$$

where p denotes the pressure, and σ is the permeability of the porous-media flow. Except for the discontinuous coefficient, Eq. (G.1) is the same as the Poisson equation in bubbly flow problems (cf. Eq. (1.3)). The exact description of the test problem and the corresponding choice for projection vectors are given below. In addition, the further setup and procedure of the experiment are taken to be the same as in Section 6.4.1.

In the porous-media flow problem, Ω consists of equal shale ($\sigma = 10^{-6}$) and sandstone ($\sigma = 1$) layers with uniform thickness, see Figure G.1(a). We impose a Dirichlet condition on the boundary $y = 1$ and homogeneous Neumann conditions on the other boundaries. The layers are denoted by the disjoint set, $\{\Omega_j,\ j = 1, 2, \ldots, k\}$, such that $\overline{\Omega} = \cup_{j=1}^{k} \overline{\Omega}_j$. The discretized domain and layers are denoted by Ω_h and Ω_{h_j}, respectively.

We choose as preconditioner, M^{-1}, the IC(0) factorization of A. In contrast to the projection vectors used in bubbly flows (see Section 6.4), the projection vectors are now chosen to be strongly related to the geometry of the problem. For each Ω_{h_j},



(a) Piecewise-constant coefficient. (b) Projection vectors.

Figure G.1: Geometry of the projection vectors and piecewise-constant coefficient in the porous-media flow.

a projection vector, z_j, is defined as follows:

$$(z_j)_i := \begin{cases} 0, & x_i \in \Omega_h \setminus \overline{\Omega}_{h_j}; \\ 1, & x_i \in \Omega_{h_j}, \end{cases} \tag{G.2}$$

where x_i is a grid point of Ω_h. In this case, each projection vector corresponds to a unique layer, see also Figure G.1(b). Then, we define $Z := [z_1 \; z_2 \; \cdots \; z_k]$.

G.2 Experiment using Standard Parameters

In the first numerical experiment, standard parameters are used with stopping tolerance $\delta = 10^{-10}$, an exact Galerkin matrix inverse, E^{-1}, and an unperturbed starting vector, $\mathcal{V}_{\text{start}}$.

The results of the experiment are presented in Table G.1 and Figure G.2. The relative errors are omitted, because they are approximately the same. The figure presents only one test case, since a similar behavior is seen for the other test cases. Moreover, for the sake of a better view, the results for PREC are omitted in Figure G.2.

From Table G.1, we observe that PREC needs more iterations to converge when n or k is increased. This only holds partly for the two-level PCG methods. The convergence of the other methods is less sensitive to the number of layers, since the number of projection vectors is chosen to be equal to the number of layers. PREC is obviously the slowest method, and the two-level PCG methods, except for A-DEF1, show approximately the same performance, which confirms the theory (cf. Theorem 6.1 and 6.3). Notice that even AD shows comparable results with the other two-level PCG methods (except A-DEF1), but it can be observed in Figure G.2 that AD shows a very erratic behavior with respect to the errors in the 2−norm (which has not been seen in the bubbly flow experiments). In other words, although AD requires approximately

	$k = 5$		$k = 7$	
Method	$n = 29^2$	$n = 54^2$	$n = 41^2$	$n = 55^2$
PREC	102	174	184	222
AD	59	95	74	90
DEF1	58	94	75	90
DEF2	68	94	75	90
A-DEF1	58	95	86	103
A-DEF2	58	94	75	90
BNN	58	94	75	90
R-BNN1	58	94	75	90
R-BNN2	58	94	75	90

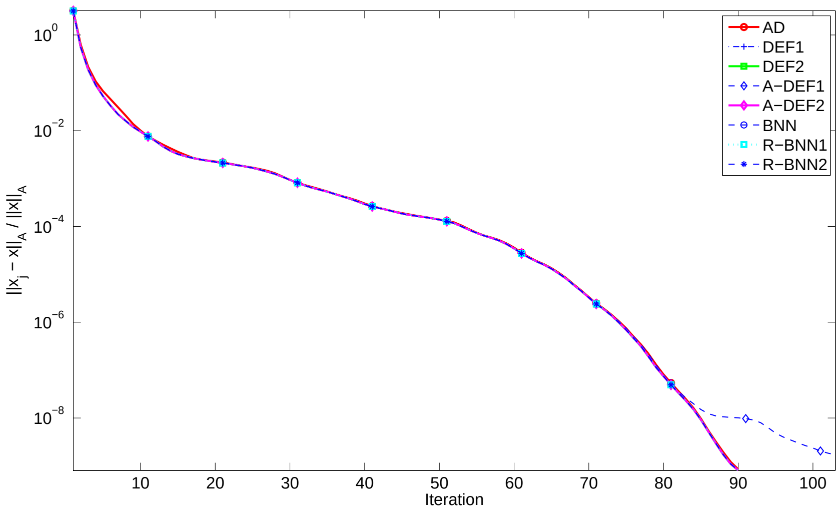
Table G.1: Number of required iterations for convergence of all proposed methods, for the porous-media problem with 'standard' parameters. The 2−norm of the relative error is approximately the same for all methods in each test case, and, hence, they are omitted in the table.

the same number of iterations, the errors measured in the 2−norm are larger, and the iterated solution is less reliable. Furthermore, A-DEF1 is somewhat slower in convergence, especially if the test case becomes more complicated.

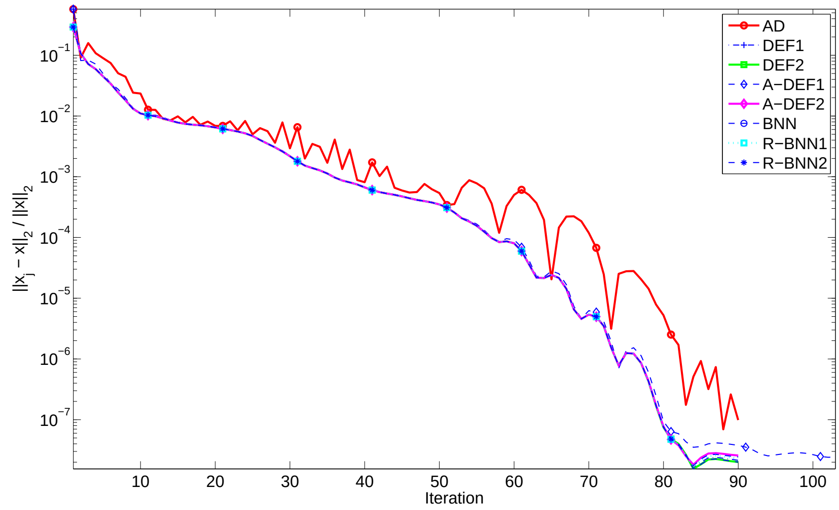
Subsequently, we present the same results in terms of computational cost. We restrict ourselves to the test case with $n = 55^2$ and $k = 7$, see Table G.2. Analogous results are obtained for the other test cases. The total computational cost within the iterations is given, following the analysis carried out in Section 6.2.4. Due to the sparsity of Z, both Z and AZ can be stored as approximately two vectors, resulting in the fact that there is no need to perform extra matrix-vector multiplications in addition to those required by PREC. It depends on the exact implementation of the methods (such as the storage and computation with Z, AZ and E) to determine which two-level PCG method requires the lowest computational cost. For example, if both IP, VU and GSS require the same amount of computing time, then it can be deduced from Table G.2 that BNN is the most expensive method, whereas AD, following by DEF1, DEF2 and R-BNN2, has the lowest computational cost per iteration.

Method	IP	VU	GSS	PR
PREC	222	666	0	222
AD	270	270	90	90
DEF1	270	360	90	90
DEF2	270	360	90	90
A-DEF1	412	412	103	103
A-DEF2	450	360	180	90
BNN	540	450	180	90
R-BNN1	450	450	180	90
R-BNN2	270	360	90	90

Table G.2: Total computational cost within the iterations in terms of number of inner products ('IP'), vector updates ('VU'), Galerkin system solves ('GSS'), preconditioning step with M^{-1} ('PR'), for the porous-media problem with $n = 55^2$, $k = 7$, and 'standard' parameters.



(a) Relative errors in the A−norm.



(b) Relative errors in the 2−norm.

Figure G.2: Relative errors during the iterative process, for the porous-media problem with $n = 55^2$, $k = 7$, and 'standard' parameters.

G.3 Experiment using Inaccurate Coarse Solves

In the next experiment, we solve $Ey_2 = y_1$ inexactly. In this case, $\tilde{y}_2$ can be interpreted as $\widetilde{E}^{-1}y_1$, where $\widetilde{E}^{-1}$ is defined as (see Eq. (6.16))

$$\widetilde{E}^{-1} := (I + \psi R)E^{-1}(I + \psi R), \quad \psi > 0, \tag{G.3}$$

where $R \in \mathbb{R}^{k \times k}$ is a symmetric random matrix with entries from the interval $[-0.5, 0.5]$. The sensitivity of the two-level PCG methods to this inaccurate solve with various values of ψ are investigated, and the results are related to Theorem G.3.

The results of the experiment can be found in Table G.3 and Figure G.3. We observe that the most robust two-level PCG methods are AD, BNN and A-DEF2, since they are largely sensitive to perturbations in E^{-1}. On the other hand, DEF1, DEF2, R-BNN1 and R-BNN2 are obviously the worst methods, as expected, since the zero eigenvalues of the corresponding two-level preconditioned matrices become nearly zero eigenvalues due to the perturbation, ψ (cf. Section 6.3.1). In addition, it can be observed that the errors diverge or stagnate for all test cases with DEF2 and R-BNN2, whereas they remain bounded and tend to converge in the case of DEF1 and R-BNN1.

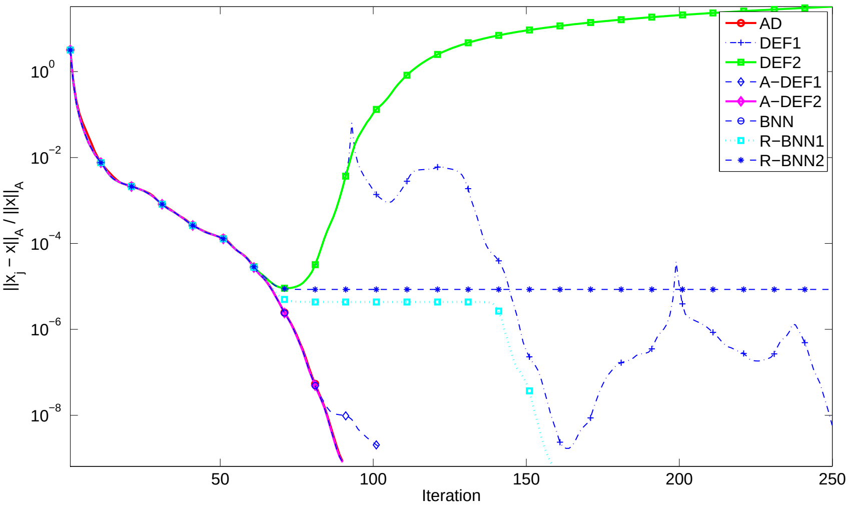
	$\psi = 10^{-12}$		$\psi = 10^{-8}$		$\psi = 10^{-4}$	
Method	# It.	$\frac{\|x_{\text{it}}-x\|_2}{\|x\|_2}$	# It.	$\frac{\|x_{\text{it}}-x\|_2}{\|x\|_2}$	# It.	$\frac{\|x_{\text{it}}-x\|_2}{\|x\|_2}$
PREC	222	2.6×10^{-8}	222	2.6×10^{-8}	222	2.6×10^{-8}
AD	90	1.0×10^{-7}	90	1.4×10^{-7}	92	1.2×10^{-7}
DEF1	90	2.6×10^{-6}	NC	6.8×10^{-7}	178	1.4×10^{-3}
DEF2	90	2.6×10^{-6}	NC	$1.6 \times 10^{+2}$	NC	$2.0 \times 10^{+4}$
A-DEF1	103	2.0×10^{-8}	103	2.2×10^{-8}	120	2.6×10^{-7}
A-DEF2	90	2.2×10^{-8}	90	2.6×10^{-8}	90	2.5×10^{-7}
BNN	90	2.3×10^{-8}	90	2.8×10^{-8}	90	7.1×10^{-8}
R-BNN1	90	6.8×10^{-7}	159	2.2×10^{-8}	213	6.9×10^{-5}
R-BNN2	90	2.6×10^{-6}	NC	2.6×10^{-2}	NC	$1.8 \times 10^{+2}$

Table G.3: Number of required iterations for convergence and the 2−norm of the relative errors of all methods, for the porous-media problem with parameters $n = 55^2$ and $k = 7$. A perturbed Galerkin matrix inverse, $\widetilde{E}^{-1}$, is used with a varying perturbation, ψ.

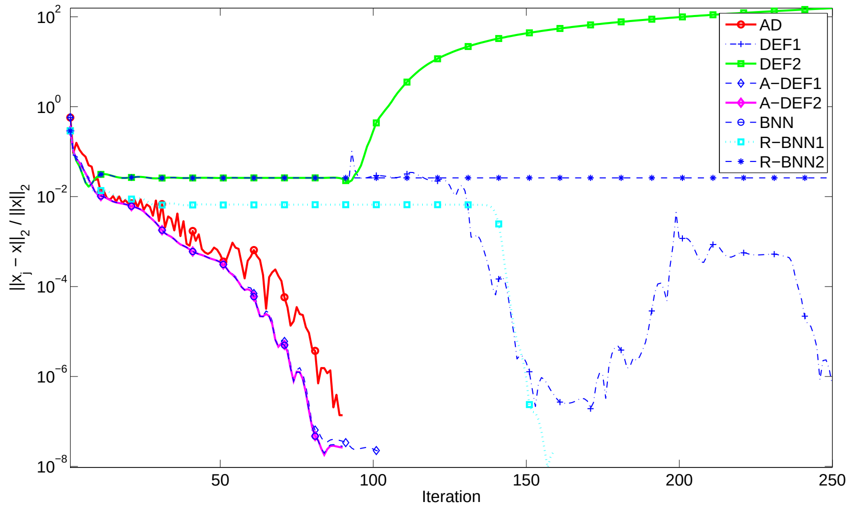
G.4 Experiment using Severe Termination Tolerances

In this section, we perform a numerical experiment with various values of the termination tolerance, δ.

The results of the experiment are presented in Table G.4 and Figure G.4. It can be seen that all methods perform well, even in the case of a relatively strict termination criterion (i.e., $\delta = 10^{-12}$). PREC also converges in all cases, but not within 250 iterations. Note, moreover, that it does not give an accurate solution if δ is chosen too large, see [173]. For $\delta < 10^{-12}$, DEF1, DEF2, R-BNN1 and R-BNN2, show difficulties, since they do not converge appropriately and may even diverge. This is in contrast to PREC, AD, BNN, A-DEF1 and A-DEF2, which give good convergence results for $\delta = 10^{-16}$. Therefore, these two-level PCG methods can be characterized as robust methods with respect to several termination criteria.

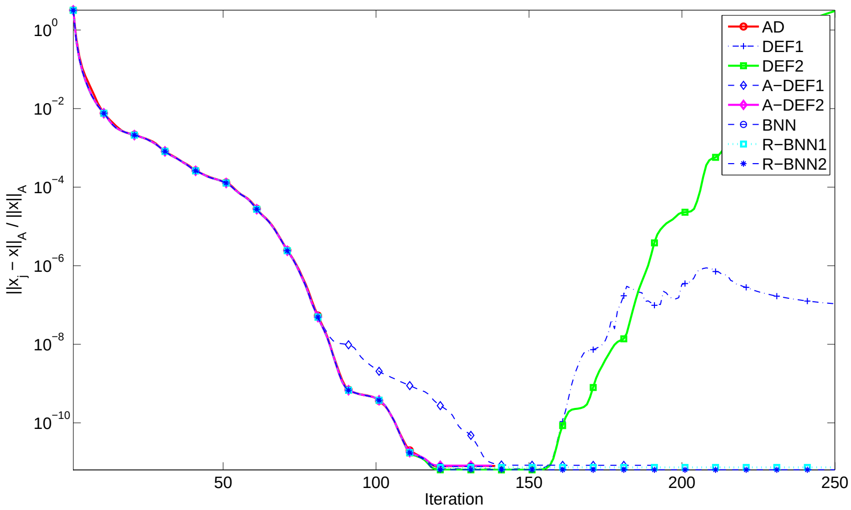


(a) Relative errors in the A−norm.

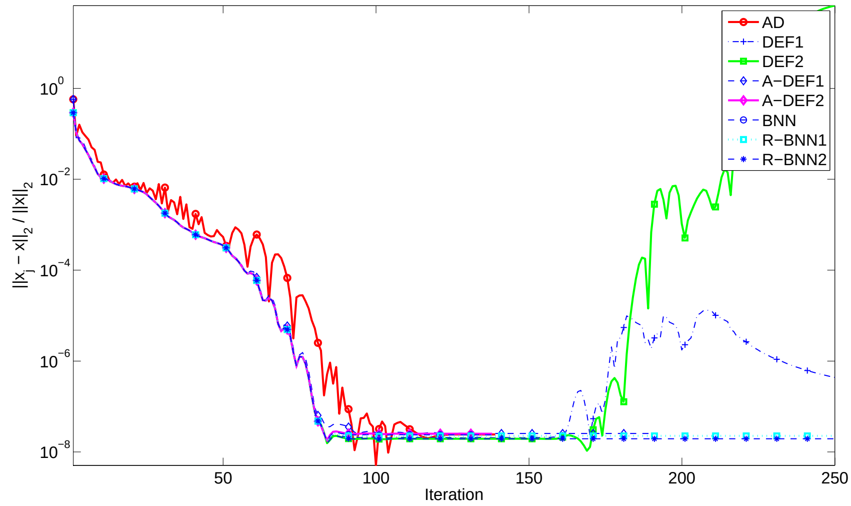


(b) Relative errors in the 2−norm.

Figure G.3: Relative errors during the iterative process for the porous-media problem with $n = 55^2$, $k = 7$ and $\widetilde{E}^{-1}$, where a perturbation $\psi = 10^{-8}$ is taken.



(a) Relative errors in the A−norm.



(b) Relative errors in the 2−norm.

Figure G.4: Relative errors during the iterative process for the porous-media problem with $n = 55^2$, $k = 7$, and termination tolerance $\delta = 10^{-16}$.

	$\delta = 10^{-8}$		$\delta = 10^{-12}$		$\delta = 10^{-16}$	
Method	# It.	$\frac{\|\|x_{\text{it}}-x\|\|_2}{\|\|x\|\|_2}$	# It.	$\frac{\|\|x_{\text{it}}-x\|\|_2}{\|\|x\|\|_2}$	# It.	$\frac{\|\|x_{\text{it}}-x\|\|_2}{\|\|x\|\|_2}$
PREC	134	3.7×10^{-1}	> 250	2.4×10^{-8}	> 250	2.4×10^{-8}
AD	80	5.2×10^{-6}	123	2.4×10^{-8}	139	2.4×10^{-8}
DEF1	80	7.5×10^{-8}	121	2.0×10^{-8}	NC	4.4×10^{-7}
DEF2	80	7.5×10^{-8}	144	1.9×10^{-8}	NC	$6.6 \times 10^{+1}$
A-DEF1	80	9.4×10^{-8}	121	2.5×10^{-8}	190	2.5×10^{-8}
A-DEF2	80	7.7×10^{-8}	121	2.5×10^{-8}	138	2.5×10^{-8}
BNN	80	7.7×10^{-8}	121	2.4×10^{-9}	138	2.4×10^{-8}
R-BNN1	80	7.6×10^{-8}	121	2.3×10^{-8}	NC	2.3×10^{-8}
R-BNN2	80	7.5×10^{-8}	121	1.9×10^{-8}	NC	1.9×10^{-8}

Table G.4: Number of required iterations for convergence and the 2−norm of the relative errors of all methods, for the porous-media problem with parameters $n = 55^2$ and $k = 7$. Various termination tolerances, δ, are tested.

G.5 Experiment using Perturbed Starting Vectors

In Section 6.3.2, we have proven that BNN with $\mathcal{V}_{\text{start}} = Qb + P^T \bar{x}$ gives exactly the same iterates as DEF2, A-DEF2, R-BNN1 and R-BNN2, in exact arithmetic. In our next experiment, we perturb $\mathcal{V}_{\text{start}}$ in DEF2, A-DEF2, R-BNN1 and R-BNN2, and examine whether this influences the convergence results. The perturbed $\mathcal{V}_{\text{start}}$, denoted by $\mathcal{W}_{\text{start}}$, is defined as a componentwise multiplication of a random vector and $\mathcal{V}_{\text{start}}$, i.e., each entry of $\mathcal{W}_{\text{start}}$ is defined as (see Eq. (6.17))

$$(\mathcal{W}_{\text{start}})_i := (1 + \gamma (v_0)_i)\,(\mathcal{V}_{\text{start}})_i, \quad i = 1, 2, \ldots, n,$$

where $\gamma \geq 0$ gives control over the accuracy of the starting vector, and vector v_0 is a random vector with entries from the interval $[-0.5, 0.5]$, taken to give each entry of $\mathcal{V}_{\text{start}}$ a different perturbation.

We perform the numerical experiment using $\mathcal{W}_{\text{start}}$ for different γ. The results can be found in Table G.5 and Figure G.5. Here, we use asterisks to stress that an extra uniqueness step is applied in the specific method. Moreover, notice that PREC, AD, DEF1 and BNN are not included in this experiment, since they apply an arbitrary vector, $\mathcal{V}_{\text{start}} = \bar{x}$, by definition.

	$\gamma = 10^{-10}$		$\gamma = 10^{-5}$		$\gamma = 10^{0}$	
Method	# It.	$\frac{\|\|x_{\text{it}}-x\|\|_2}{\|\|x\|\|_2}$	# It.	$\frac{\|\|x_{\text{it}}-x\|\|_2}{\|\|x\|\|_2}$	# It.	$\frac{\|\|x_{\text{it}}-x\|\|_2}{\|\|x\|\|_2}$
DEF2	90	2.2×10^{-8}	NC	$2.1 \times 10^{+11}$	NC	$3.5 \times 10^{+18}$
A-DEF2	90	2.5×10^{-8}	90	2.5×10^{-8}	90	2.4×10^{-8}
R-BNN1	90	2.5×10^{-8}	NC	$2.5 \times 10^{-8}*$	NC	$1.3 \times 10^{-5}*$
R-BNN2	90	2.0×10^{-8}	NC	$2.9 \times 10^{-6}*$	NC	$2.5 \times 10^{-1}*$

Table G.5: Number of required iterations for convergence and the 2−norm of the relative errors of some methods, for the porous-media problem with parameters $n = 55^2$, $k = 7$, and perturbed starting vectors. An asterisk (*) means that an extra uniqueness step is applied in that test case.

From the results, it can be noticed that all involved methods converge appropriately

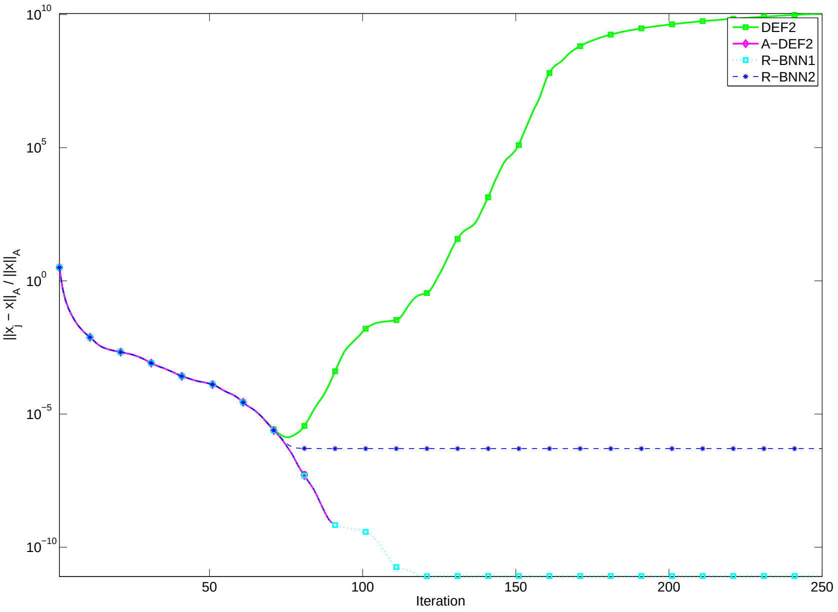


Figure G.5: Relative errors in the $A-$norm during the iterative process for the porous-media problem with $n = 55^2$, $k = 7^2$, and perturbed starting vectors with $\gamma = 10^{-5}$. The plot of the relative errors in the $2-$norm is omitted, since the two plots are approximately the same.

for $\gamma = 10^{-10}$. For $\gamma \geq 10^{-5}$, DEF2, R-BNN1 and R-BNN2 fail to converge. The most robust method is, obviously, A-DEF2. This method seems to be completely insensitive to the perturbation, γ. This experiment shows that the 'reduced' variants of BNN have different robustness properties with respect to perturbations in starting vectors.

Appendix H

DICCG Variants applied to Bubbly Flow Simulations

We perform two 3-D simulations of $l = 250$ time steps in order to test the DICCG1$(-k)$ and DICCG2$(-k)$ methods as given in Definition 8.1. Recall that the difference between the two deflation methods is the inner solver for the Galerkin systems: this is done in a direct way in DICCG1, while ICCG is applied within DICCG2. In this appendix, we show that both deflation variants are applicable to bubbly flow simulations, and the performance of these methods is comparable for relatively small problems.

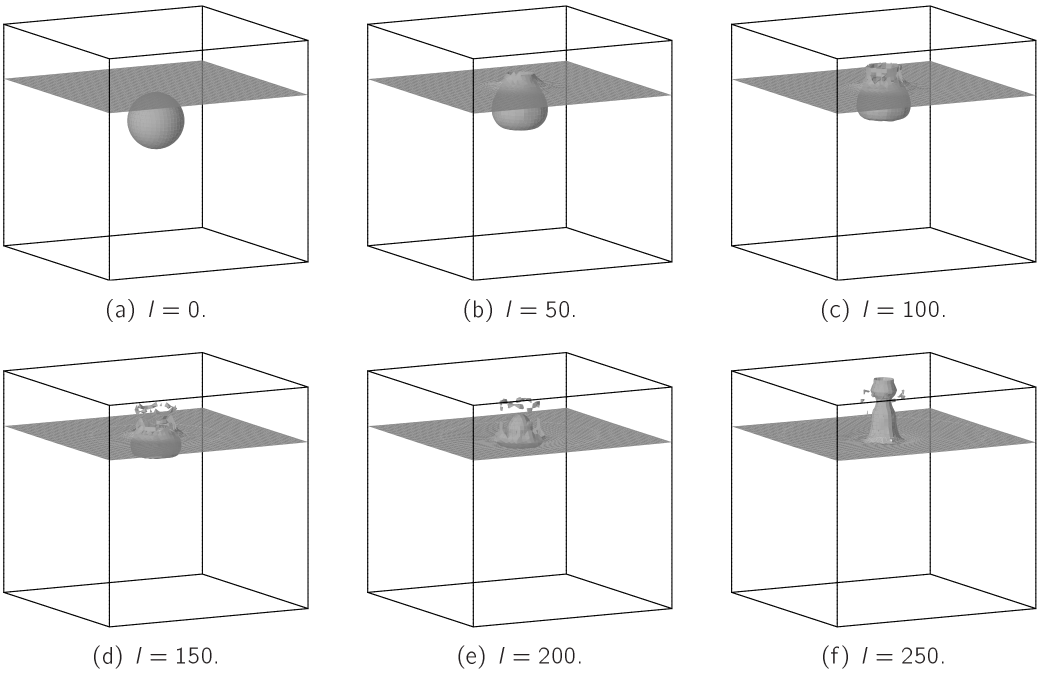
In the first simulation, an air bubble is rising in water, whereas a water droplet is falling in the air in the second simulation. We do not include surface tension in the simulations in order to obtain complicated density geometries. We refer to [154, Sect. 8.3.4] for more details. Similarly to Section 10.3, we concentrate on solving the linear system (10.6) derived from a Poisson problem (10.5) at each time step. We adopt ICCG, DICCG1 and DICCG2 to solve (10.5). We take deflation variant 5.1 in DICCG1 and deflation variant 5.3 in DICCG2 (cf. Table 5.1).

H.1 Simulation 1: Rising Air Bubble in Water

For the first simulation, the starting position of the bubble in the domain and the evolution of its movement for $l \in [0, 250]$ can be found in Figure H.1.

In [154], the Poisson solver is based on ICCG. Here, we compare this method to both DICCG1-10^3 and DICCG2-20^3 for $n = 100^3$. It turns out that these deflation methods are optimal in the sense that they need the lowest computational time to perform the simulation compared to DICCG1 and DICCG2 with different k (cf. Figure 8.4). The results are presented in Figure H.2.

From Figure H.2(a), we notice that the number of iterations is strongly reduced by the deflation method. DICCG1-10^3 and DICCG2-20^3 require at most 60 iterations, while ICCG converges in between 200 and 300 iterations for most time steps. For each l, DICCG2-20^3 requires fewer iterations than DICCG1-10^3, which is in agreement


(a) $l = 0$. (b) $l = 50$. (c) $l = 100$.

(d) $l = 150$. (e) $l = 200$. (f) $l = 250$.

Figure H.1: Evolution of the rising bubble in water in the first 250 time steps.

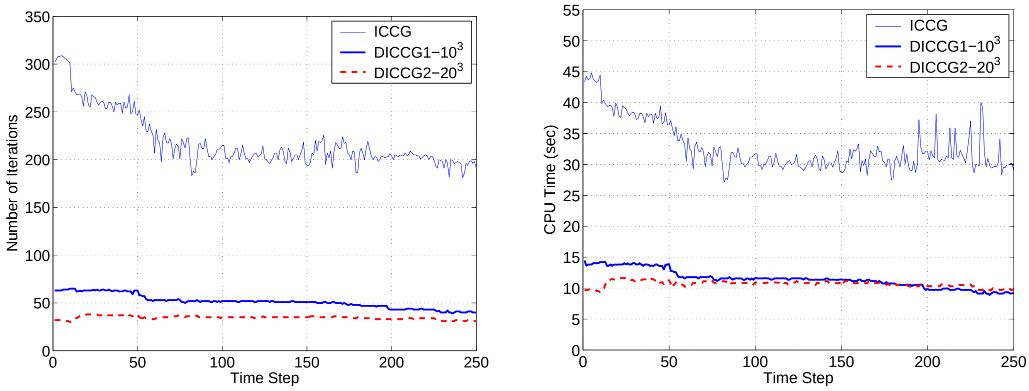
with Theorem 3.3. Moreover, we observe the erratic behavior of ICCG, whereas the deflation methods are less sensitive to the geometries of the bubbles, during the evolution of the simulation. Considering CPU time, DICCG1-10^3 and DICCG2-20^3 also show very good performance, see Figure H.2(b). For most time steps, ICCG requires 25–45 seconds to converge, whereas both deflation methods are comparable and only need around 9–14 seconds. Moreover, in Figure H.2(c), one can find the gain factors for both the ratios of the iterations and the CPU time between ICCG and the two deflation methods, respectively. From this figure, we conclude that DICCG1-10^3 or DICCG2-20^3 need approximately 4–8 times fewer iterations, depending on the time step. More importantly, both deflation methods converge approximately 2–4 times faster than ICCG at all time steps.

We end this subsection with the remark that similar results can be found for other choices of grid sizes. For problems with larger grid sizes, the deflation methods become more favorable, when compared to ICCG.

H.2 Simulation 2: Falling Water Droplet in Air

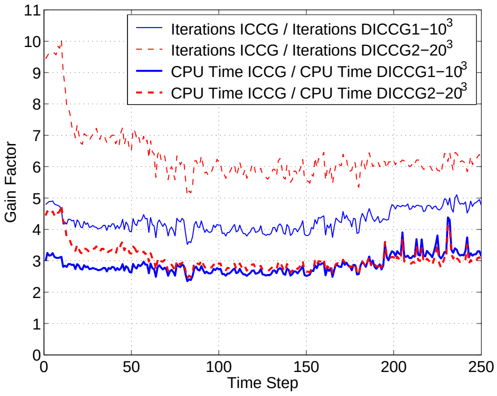
For the second simulation, the starting position of the droplet in the domain and the evolution of its movement can be found in Figure H.3. The results are presented in Figure H.4, where again DICCG1-10^3 and DICCG2-20^3 are adopted.

Similar observations as those from the previous subsection can be drawn from Figure H.4. Obviously, the deflation methods are more efficient, when compared with ICCG, in terms of both number of iterations and required CPU time. We observe that



(a) Number of iterations versus time step.

(b) CPU time versus time step.



(c) Gain factors of DICCG1-10^3 and DICCG2-20^3 with respect to ICCG.

Figure H.2: Results for ICCG, DICCG1-10^3 and DICCG2-20^3 for the simulation with a rising air bubble in water.

both DICCG1-10^3 and DICCG2-20^3 need approximately 3–5 times fewer iterations and they converge more-or-less 2–4 times faster than ICCG. In this test problem, it can be observed that DICCG2-20^2 performs somewhat better than DICCG1-10^3.

Finally, a small jump in the DICCG1-10^3 performance can be noticed around the 205-th time step in Figure H.4. This might be the result of the appearance of a rising droplet, which can be observed in Figures H.3(e) and (f) as well. This jump is not significant in DICCG2-20^3. Apparently, a larger set of deflation vectors effectively treats that droplet.

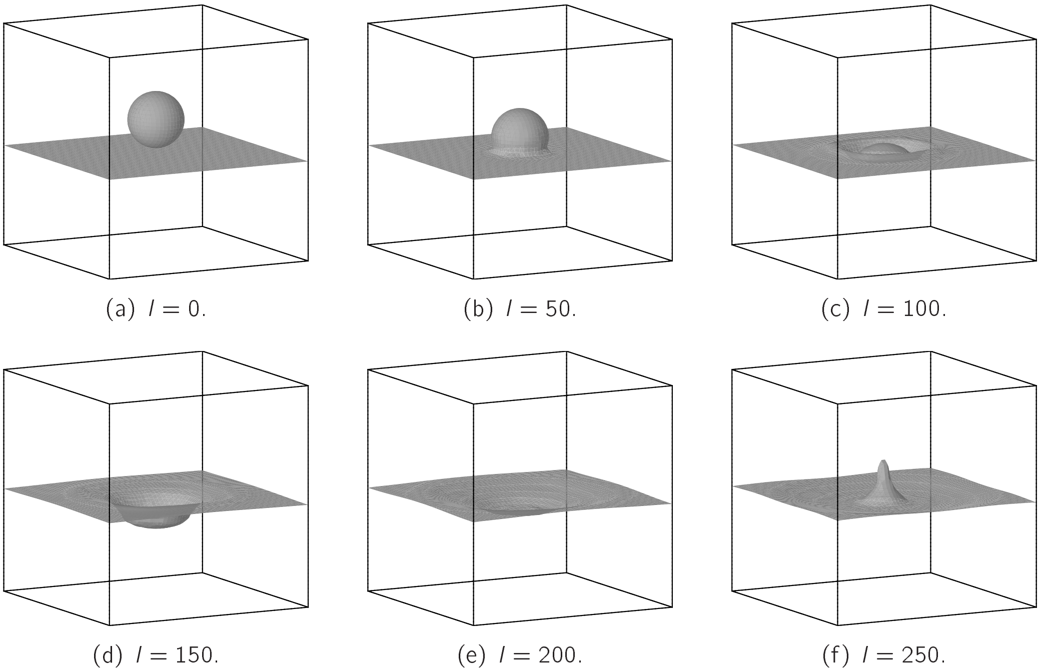
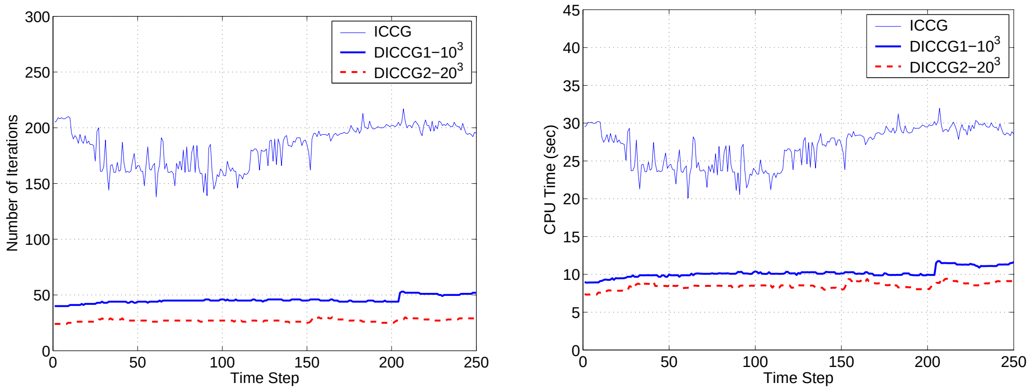
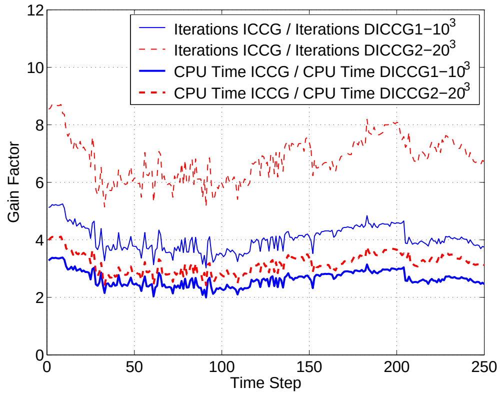

(a) $l = 0$.
(b) $l = 50$.
(c) $l = 100$.
(d) $l = 150$.
(e) $l = 200$.
(f) $l = 250$.

Figure H.3: Evolution of the falling droplet in air in the first 250 time steps.



(a) Number of iterations versus time step.

(b) CPU time versus time step.



(c) Gain factors of DICCG1-10^3 and DICCG2-20^3 with respect to ICC G.

Figure H.4: Results for ICCG, DICCG1-10^3 and DICCG2-20^3 for the simulation with a falling water droplet in air.

Appendix I

Comparison of Deflation and Multigrid for a Special Case

We consider the two-level PCG methods (DEF and MG) based on the following two-level preconditioners (see Eqs. (7.1) and (7.4)):

$$\left\{ \begin{array}{lcl} \mathcal{P}_{\text{DEF}} & = & M^{-1}P; \\ \mathcal{P}_{\text{MG}} & = & \bar{M}^{-T}P + P^T\bar{M}^{-1} + Q - \bar{M}^{-T}PA\bar{M}^{-1}. \end{array} \right.$$

We show that abstract preconditioners in the MG framework do not always lead to better conditioned two-level coefficient matrices compared to DEF. Such problems can even be found in the case of $M^{-1} = \bar{M}^{-1} = I$.

We assume that $Z = [v_1 \cdots v_k]$, where $\{v_i\}$ is the set of orthonormal eigenvectors corresponding to the increasing set of eigenvalues of A, $\{\lambda_i\}$. Then, we know from Example 7.1 that the MG operator is only SPD if $\lambda_i < 2$. Similar to Example 7.1, we obtain

$$\begin{array}{rcl} \mathcal{P}_{\text{MG}}Av_i & = & 2Av_i - 2ZZ^TAv_i + Z\Lambda^{-1}Z^TAv_i - A^2v_i + ZZ^TA^2v_i \\ & = & 2\lambda_i v_i - 2\lambda_i ZZ^Tv_i + \lambda_i Z\Lambda^{-1}Z^Tv_i - \lambda_i^2 v_i + \lambda_i^2 ZZ^Tv_i, \end{array}$$

where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_k)$. This implies

$$\mathcal{P}_{\text{MG}}Av_i = \left\{ \begin{array}{lcll} 2\lambda_i v_i - 2\lambda_i v_i + v_i - \lambda_i^2 v_i + \lambda_i^2 v_i & = & v_i, & \text{for } i = 1, \ldots, k; \\ 2\lambda_i v_i - \lambda_i^2 v_i, & = & \lambda_i(2-\lambda_i)v_i, & \text{for } i = k+1, \ldots, n. \end{array} \right.$$

Hence, if A has eigenvalues $\{\lambda_i\}$, then the spectrum of $\mathcal{P}_{\text{MG}}A$ is given by

$$\{1, \ldots, 1, \lambda_{k+1}(2-\lambda_{k+1}), \ldots, \lambda_n(2-\lambda_n)\}. \tag{I.1}$$

We note that $\lambda_i(2-\lambda_i) \leq 1$ for all $i = k+1, \ldots, n$, because of $0 < \lambda_i < 2$, see

Figure I.1. Accordingly, the condition number of $\mathcal{P}_{\mathrm{MG}}A$ is given by

$$\kappa_{\mathrm{MG}} = \frac{1}{\min\{\lambda_{k+1}(2-\lambda_{k+1}), \lambda_n(2-\lambda_n)\}}.$$

On the other hand, for DEF, we know that (see Section 3.5)

$$\mathcal{P}_{\mathrm{DEF}}Av_i = \begin{cases} 0, & \text{for } i = 1, \ldots, k; \\ \lambda_i v_i, & \text{for } i = k+1, \ldots, n. \end{cases} \tag{I.2}$$

Therefore, (cf. Eq. (3.2))

$$\kappa_{\mathrm{DEF}} = \frac{\lambda_n}{\lambda_{k+1}}.$$

It depends on eigenvalues λ_{k+1} and λ_n of A whether κ_{MG} or κ_{DEF} is more favorable. If $\lambda_{k+1}, \ldots, \lambda_n \to 2$, then obviously $\kappa_{\mathrm{DEF}} < \kappa_{\mathrm{MG}}$. In other words, M^{-1} and Z can be chosen in such a way that MG with an SPD operator is expected to converge slower than DEF, see also Example I.1.

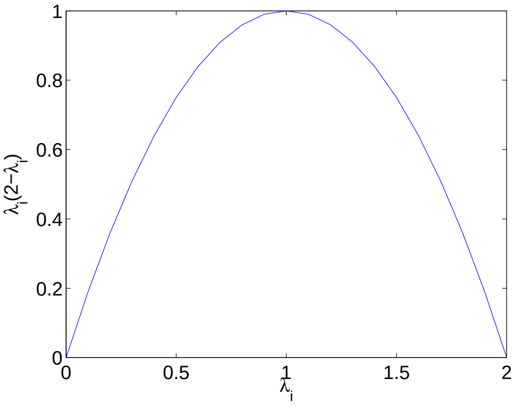


Figure I.1: Function $\lambda_i(2-\lambda_i)$ for $\lambda_i \in [0, 2]$.

Example I.1. *We construct a simple example to show that $\kappa_{MG} < \kappa_{DEF}$ does not hold in general, even if $\mathcal{P}_{MG}$ is SPD.*

Let A be an SPD diagonal matrix given by

$$A = \mathrm{diag}(1, 1.25, 1.5, 1.75).$$

Then, the spectrum of A is $\sigma = (1, 1.25, 1.5, 1.75)$, where the corresponding eigenvectors are columns of I: $I = [v_1\ v_2\ v_3\ v_4]$. Hence, $\mathcal{P}_{MG}$ is SPD.

Choose now $Z = [v_1\ v_2]$ and $M^{-1} = I$. Then, the eigenvalues of $\mathcal{P}_{MG}A$ are given by Eq. (I.1):

$$\sigma_{MG} = \{1, 1, \lambda_3(2-\lambda_3), \lambda_4(2-\lambda_4)\} = \{1, 1, 0.4375, 0.75\},$$

whereas (cf. Eq. (I.2))

$$\sigma_{DEF} = \{0, 0, \lambda_3, \lambda_4\} = \{0, 0, 1.5, 1.75\}.$$

This leads immediately to the condition numbers

$$\kappa_{MG} = \frac{1}{\min\{\lambda_{k+1}(2-\lambda_{k+1}), \lambda_n(2-\lambda_n)\}} = \frac{1}{0.4375} = 2.2857,$$

and

$$\kappa_{DEF} = \frac{\lambda_n}{\lambda_{k+1}} = \frac{1.75}{1.5} = 1.1667,$$

so that $\kappa_{MG} > \kappa_{DEF}$ *obviously holds in this case.*

Example I.2. *It is easy to construct examples showing that* $\kappa_{MG} < \kappa_{DEF}$. *For instance, take*

$$A = \mathrm{diag}(0.5, 0.75, 1.0, 1.25),$$

with the same setting of the parameters of MG and DEF as in Example I.1. Then,

$$\sigma_{MG} = \{1, 1, 1, 0.9375\}, \quad \sigma_{DEF} = \{0, 0, 1.0, 1.25\},$$

giving us

$$\kappa_{MG} = \frac{1}{0.9375} = 1.0667, \quad \kappa_{DEF} = \frac{1.25}{1.0} = 1.25,$$

so that $\kappa_{MG} < \kappa_{DEF}$ *holds in this case.*

Bibliography

[1] J. C. Adams. MUDPACK: Multigrid portable FORTRAN software for the efficient solution of linear elliptic partial differential equations. *Appl. Math. Comput.*, 34(2):113–146, 1989.

[2] D. M. Alber and L. N. Olson. Parallel coarse-grid selection. *Numer. Lin. Alg. Appl.*, 14(8):611–643, 2007.

[3] R. E. Alcouffe, A. Brandt, J. E. Dendy, and J. W. Painter. The multigrid method for the diffusion equation with strongly discontinuous coefficients. *SIAM J. Sci. Stat. Comput.*, 2:430–454, 1981.

[4] P. Arbenz and W. Petersen. *Introduction to Parallel Computing*. Oxford University Press, Oxford, 2004.

[5] M. E. Argentati, A. V. Knyazev, C. C. Paige, and I. Panayotov. Bounds on changes in Ritz values for a perturbed invariant subspace of a Hermitian matrix. *SIAM J. Matrix Anal. Appl.*, 2008. Submitted. Published as a technical report `http://arxiv.org/abs/math/0610498`.

[6] S. F. Ashby, T. A. Manteuffel, and P. E. Saylor. A taxonomy for conjugate gradient methods. *SIAM J. Numer. Anal.*, 27(6):1542–1568, 1990.

[7] T. Austin, M. Berndt, B. K. Bergen, J. E. Dendy, and J. D. Moulton. Parallel, scalable, and robust multigrid on structured grids. T-7 Research Highlight LA-UR-03-9167, Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM, USA, 2003.

[8] O. Axelsson. *Iterative solution methods*. Cambridge University Press, Cambridge, UK, 1994.

[9] O. Axelsson and G. Lindskog. On the eigenvalue distribution of a class of preconditioning methods. *Numer. Math.*, 48(5):479–498, 1986.

[10] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *Templates for the Solution*

of Linear Systems: Building Blocks for Iterative Methods. SIAM, Philadelphia, PA, 1994. Second edition.

[11] P. Bastian, W. Hackbusch, and G. Wittum. Additive and multiplicative multigrid. A comparison. *Comput.*, 60(4):345–364, 1998.

[12] A. Behie and P. A. Forsyth. Multi–grid solution of three–dimensional problems with discontinuous coefficients. *Appl. Math. Comput.*, 13:229–240, 1983.

[13] J. B. Bell, P. Colella, and H. M. Glaz. A second-order projection method for the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 85(2):257–283, 1989.

[14] A. Ben-Israel and T. N. E. Greville. *Generalized Inverses: Theory and Applications*. Springer, New York, 2003. Second Edition.

[15] A. Berman and R. J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. SIAM, Philadelphia, PA, USA, 1994. Corrected republication, with supplement, of work first published in 1979 by Academic Press.

[16] R. Blaheta. Multilevel iterative methods and deflation. In P. Wesseling, E. Onate, and J. Periaux, editors, *European Conference on Computational Fluid Dynamics ECCOMAS CFD 2006*, Delft, 2006. TU Delft.

[17] P. Bochev and R. B. Lehoucq. On the finite element solution of the pure Neumann problem. *SIAM Rev.*, 47(1):50–66, 2005.

[18] D. Braess. On the combination of the multigrid method and conjugate gradients. In *Multigrid methods, II (Cologne, 1985)*, volume 1228 of *Lecture Notes in Math.*, pages 52–64. Springer, Berlin, 1986.

[19] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring. I. *Math. Comput.*, 47(175):103–134, 1986.

[20] J. H. Bramble, J. E. Pasciak, J. Wang, and J. Xu. Convergence estimates for multigrid algorithms without regularity assumptions. *Math. Comp.*, 57:23–45, 1991.

[21] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Math. Comp.*, 31(138):333–390, 1977.

[22] A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic multigrid (AMG) for sparse matrix equations. In D. J. Evans, editor, *Sparsity and Its Applications*, pages 257–284. Cambridge University Press, Cambridge, 1984.

[23] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. SIAM Books, Philadelphia, 2000. Second edition.

[24] B. Bunner and G. Tryggvason. Dynamics of homogeneous bubbly flows. Part 1. Rise velocity and microstructure of the bubbles. *J. Fluid Mech.*, 466:17–52, 2002.

[25] K. Burrage, J. Erhel, B. Pohl, and A. Williams. A deflation technique for linear systems of equations. *SIAM J. Sci. Comput.*, 19(4):1245–1260, 1998.

[26] B. Carpentieri, L. Giraud, and S. Gratton. Additive and multiplicative two-level spectral preconditioning for general linear systems. *SIAM J. Sci. Comput.*, 29(4):1593–1612, 2007.

[27] A. Chapman and Y. Saad. Deflated and augmented Krylov subspace techniques. *Numer. Lin. Alg. Appl.*, 4(1):43–66, 1997.

[28] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Math. Comp.*, 22:745–762, 1968.

[29] A. J. Chorin. On the convergence of discrete approximations to the Navier-Stokes equations. *Math. Comp.*, 23:341–353, 1969.

[30] A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, G. N. Miranda, and J. W. Ruge. Robustness and scalability of algebraic multigrid. *SIAM J. Sci. Comput.*, 21(5):1886–1908, 2000.

[31] M. Clemens, M. Wilke, R. Schuhmann, and T. Weiland. Subspace projection extrapolation scheme for transient field simulations. *IEEE Transactions on Magnetics*, 40(2):934–937, 2004.

[32] P. Concus and G. H. Golub. Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations. *SIAM J. Numer. Anal.*, 10:1103–1120, 1973.

[33] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*, pages 157–172, New York, NY, USA, 1969. ACM Press.

[34] F. S. de Sousa, N. Mangiavacchi, L. G. Nonato, A. Castelo, M. F. Tomé, V. G. Ferreira, J. A. Cuminato, and S. McKee. A front-tracking/front-capturing method for the simulation of 3D multi-fluid flows with free surfaces. *J. Comput. Phys.*, 198(2):469–499, 2004.

[35] E. de Sturler. Truncation strategies for optimal Krylov subspace methods. *SIAM J. Numer. Anal.*, 36(3):864–889, 1999.

[36] J. Demmel, M. Heath, and H. van der Vorst. Parallel numerical linear algebra. In *Acta Numerica 1993*, pages 111–198. Cambridge University Press, Cambridge, UK, 1993.

[37] J. E. Dendy. Black box multigrid. *J. Comput. Phys.*, 48(3):366–386, 1982.

[38] J. E. Dendy. Two multigrid methods for three-dimensional equations with highly discontinuous coefficients. *SIAM J. Sci. Stat. Comput.*, 8:673–685, 1987.

[39] J. E. Dendy. Black box multigrid for periodic and singular problems. *Appl. Math. Comput.*, 25(1, part I):1–10, 1988.

[40] Z. Dostal. Conjugate gradient method with preconditioning by projector. *Int. J. Comput.. Math.*, 23:315–323, 1988.

[41] M. Dryja. An additive Schwarz algorithm for two- and three-dimensional finite element elliptic problems. In T. Chan, R. Glowinski, J. Périaux, and O. Widlund, editors, *Domain Decomposition Methods*, pages 168–172, Philadelphia, PA, 1989. SIAM.

[42] M. Dryja and O. B. Widlund. Towards a unified theory of domain decomposition algorithms for elliptic problems. In T. Chan, R. Glowinski, J. Périaux, and O. Widlund, editors, *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 3–21. SIAM, Philadelphia, PA, 1990.

[43] M. Dryja and O. B. Widlund. Schwarz methods of Neumann-Neumann type for three-dimensional elliptic finite element problems. *Comm. Pure Appl. Math.*, 48(2):121–155, 1995.

[44] I. S. Duff and H. A. van der Vorst. Developments and trends in the parallel solution of linear systems. *Parallel Comput.*, 25(13-14):1931–1970, 1999.

[45] M. Eiermann, O. G. Ernst, and O. Schneider. Analysis of acceleration strategies for restarted minimal residual methods. *J. Comput. Appl. Math.*, 123(1-2):261–292, 2000.

[46] J. Erhel and F. Guyomarc'h. An augmented conjugate gradient method for solving consecutive symmetric positive definite linear systems. *SIAM J. Matrix Anal. Appl.*, 21(4):1279–1299, 2000.

[47] Y. A. Erlangga and R. Nabben. Deflation and balancing preconditioners for Krylov subspace methods applied to nonsymmetric matrices. *SIAM J. Matrix Anal.*, 2008. To appear.

[48] Y. A. Erlangga and R. Nabben. Multilevel projection-based nested Krylov iteration for boundary value problems. *SIAM J. Sci. Comput.*, 30(3):1572–1595, 2008.

[49] A. Esmaeeli and G. Tryggvason. Direct numerical simulations on bubbly flows. Part 1. Low Reynolds number arrays. *J. Fluid Mech.*, 377:313–345, 1998.

[50] A. Esmaeeli and G. Tryggvason. Direct numerical simulations on bubbly flows. Part 2. Moderate Reynolds number arrays. *J. Fluid Mech.*, 385:325–358, 1999.

[51] V. Faber and T. Manteuffel. Necessary and sufficient conditions for the existence of a conjugate gradient method. *SIAM J. Numer. Anal.*, 21:352–362, 1984.

[52] R. D. Falgout and J. E. Jones. Multigrid on massively parallel architectures. In *Multigrid methods, VI (Gent, 1999)*, volume 14 of *Lect. Notes Comput. Sci. Eng.*, pages 101–107. Springer, Berlin, 2000.

[53] R. D. Falgout, P. S. Vassilevski, and L. T. Zikatanov. On two-grid convergence estimates. *Num. Lin. Alg. Appl.*, 12(5–6):471–494, 2005.

[54] P. F. Fischer. An overlapping Schwarz method for spectral element solution of the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 133(1):84–101, 1997.

[55] M. Fortin. Old and new finite elements for incompressible flows. *Int. J. Numer. Methods Fluids*, 1(4):347–364, 1981.

[56] J. Frank and C. Vuik. On the construction of deflation-based preconditioners. *SIAM J. Sci. Comp.*, 23:442–462, 2001.

[57] A. Frommer, R. Nabben, and D. B. Szyld. Convergence of stationary iterative methods for Hermitian semidefinite linear systems and applications to Schwarz methods. *SIAM J. Matrix Anal. Appl.*, 2008. To appear.

[58] H. De Gersem and K. Hameyer. A deflated iterative solver for magnetostatic finite element models with large differences in permeability. *Eur. Phys. J. Appl. Phys.*, 13:45–49, 2000.

[59] L. Giraud, D. Ruiz, and A. Touhami. A comparative study of iterative solvers exploiting spectral information for SPD systems. *SIAM J. Sci. Comput.*, 27(5):1760–1786, 2006.

[60] V. Girault and P.-A. Raviart. *Finite element approximation of the Navier-Stokes equations*, volume 749 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1979.

[61] G. H. Golub and D. P. O'Leary. Some history of the conjugate gradient and Lanczos methods. *SIAM Rev.*, 31(1):50–102, 1989.

[62] G. H. Golub and M. L. Overton. The convergence of inexact Chebyshev and Richardson iterative methods for solving linear systems. *Numer. Math.*, 53(5):571–593, 1988.

[63] G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins Univ. Press, Baltimore, MD, 1996. Third edition.

[64] G. H. Golub and Q. Ye. Inexact preconditioned conjugate gradient method with inner-outer iteration. *SIAM J. Sci. Comput.*, 21(4):1305–1320, 2000.

[65] I. G. Graham and R. Scheichl. Robust domain decomposition algorithms for multiscale PDEs. *Numer. Methods Partial Differ. Eq.*, 23:859–878, 2007.

[66] A. Greenbaum. *Iterative methods for solving linear systems*, volume 17 of *Frontiers in Applied Mathematics*. SIAM, Philadelphia, PA, 1997.

[67] I. Gustafsson. A class of first order factorization methods. *BIT*, 18(2):142–156, 1978.

[68] W. Hackbusch. Convergence of multi–grid iterations applied to difference equations. *Math. Comp.*, 34:425–440, 1980.

[69] W. Hackbusch. *Multigrid Methods and Applications*. Springer-Verlag, Berlin, 1985.

[70] L. A. Hageman and D. M. Young. *Applied iterative methods*. Computer Science and Applied Mathematics. Academic Press, New York, NY, USA, 1981.

[71] V. E. Henson and U. M. Yang. BoomerAMG: a parallel algebraic multigrid solver and preconditioner. *Appl. Numer. Math.*, 41(1):155–177, 2002.

[72] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Stand.*, 49:409–436, 1952.

[73] R. Horn and C. Johnson. *Matrix Analysis*. Cambridge University Press, New York, NY, USA, 1990. USA Edition.

[74] J. Hua and J. Lou. Numerical simulation of bubble rising in viscous liquid. *J. Comput. Phys.*, 222(2):769–795, 2007.

[75] T. Inamuro, T. Ogata, S. Tajima, and N. Konishi. A lattice Boltzmann method for incompressible two-phase flows with large density differences. *J. Comp. Phys.*, 198:628–644, 2004.

[76] J. E. Jones and S. F. McCormick. Parallel multigrid methods. In *Parallel numerical algorithms (Hampton, VA, 1994)*, volume 4 of *ICASE/LaRC Interdiscip. Ser. Sci. Eng.*, pages 203–224. Kluwer Acad. Publ., Dordrecht, 1997.

[77] E. F. Kaasschieter. Preconditioned conjugate gradients for solving singular systems. *J. Comput. Appl. Math.*, 24(1-2):265–275, 1988.

[78] R. Kettler and J. A. Meijerink. A multigrid method and a combined multigrid-conjugate gradient method for elliptic problems with strongly discontinuous coefficients in general domains. Tech. Rep. 604, Shell Oil Company, 1981.

[79] M. Khalil and P. Wesseling. Vertex-centered and cell-centered multigrid for interface problems. *J. Comput. Phys.*, 98:1–20, 1992.

[80] S. A. Kharchenko and A. Yu. Yeremin. Eigenvalue translation based preconditioners for the GMRES(k) method. *Num. Lin. Alg. Appl.*, 2(1):51–77, 1995.

[81] A. V. Knyazev and M. E. Argentati. Rayleigh-Ritz majorization error bounds with applications to FEM and subspace iterations. *SIAM J. Numer. Anal.*, 2008. Submitted. Published as a technical report `http://arxiv.org/abs/math/0701784`.

[82] L. Y. Kolotilina. Twofold deflation preconditioning of linear algebraic systems. I. Theory. *J. Math. Sci.*, 89:1652–1689, 1998.

[83] E. Kreyszig. *Introductory Functional Analysis with Applications*. Wiley, New York, 1989.

[84] D. Kwak, C. Kiris, and J. Dacles-Mariani. An assessment of artificial compressibility and pressure projection methods for incompressible flow simulations. In *Sixteenth International Conference on Numerical Methods in Fluid Dynamics*, volume 515 of *Lecture Notes in Physics*, pages 177–182. Springer, 1998.

[85] E. Ludwig, R. Nabben, and J. M. Tang. Deflation and projection methods applied to positive semi-definite systems. 2008. In preparation.

[86] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, Reading, Massachussets, 1984. Second edition.

[87] S. P. MacLachlan, J. M. Tang, and C. Vuik. Fast and robust solvers for pressure correction in bubbly flow problems. DIAM Report 08-01, Delft University of Technology, Delft, 2008.

[88] S. P. MacLachlan, J. M. Tang, and C. Vuik. Fast and robust solvers for pressure correction in bubbly flow problems. 2008. Submitted.

[89] J. Mandel. Balancing domain decomposition. *Comm. Numer. Meth. Engrg.*, 9:233–241, 1993.

[90] J. Mandel. Hybrid domain decomposition with unstructured subdomains. In A. Quarteroni, Y. A. Kuznetsov, J. Périaux, and O. B. Widlund, editors, *Domain Decomposition Methods in Science and Engineering: The Sixth International Conference on Domain Decomposition*, volume 157 of *Contemporary Mathematics*, pages 103–112. AMS, 1994. Como, Italy, June 15–19, 1992.

[91] J. Mandel and M. Brezina. Balancing domain decomposition for problems with large jumps in coefficients. *Math. Comp.*, 65:1387–1401, 1996.

[92] L. Mansfield. On the use of deflation to improve the convergence of conjugate gradient iteration. *Communs. Appl. Numer. Meth.*, 4:151–156, 1988.

[93] L. Mansfield. On the conjugate gradient solution of the Schur complement system obtained from domain decomposition. *SIAM J. Numer. Anal.*, 27(6):1612–1620, 1990.

[94] L. Mansfield. Damped Jacobi preconditioning and coarse grid deflation for conjugate gradient iteration on parallel computers. *SIAM J. Sci. Stat. Comput.*, 12(6):1314–1323, 1991.

[95] E. Marchandise, P. Geuzaine, N. Chevaugeon, and J. Remacle. A stabilized finite element method using a discontinuous level set approach for the computation of bubble dynamics. *J. Comput. Phys.*, 225(1):949–974, 2007.

[96] S. F. McCormick and J. W. Ruge. Convergence estimates for multigrid algorithms without regularity assumptions. *SIAM J. Numer. Anal.*, 19:924–929, 1982.

[97] J. A. Meijerink and H. A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comp.*, 31(137):148–162, 1977.

[98] M. Mohr and R. Wienands. Cell-centred multigrid revisited. *Comput. Vis. Sci.*, 7(3-4):129–140, 2004.

[99] R. B. Morgan. A restarted GMRES method augmented with eigenvectors. *SIAM J. Matrix Anal. Appl.*, 16(4):1154–1171, 1995.

[100] R. B. Morgan. GMRES with deflated restarting. *SIAM J. Sci. Comput.*, 24(1):20–37, 2002.

[101] J. D. Moulton, J. E. Dendy, and J. M. Hyman. The black box multigrid numerical homogenization algorithm. *J. Comput. Phys.*, 141:1–29, 1998.

[102] W. Mulder, S. Osher, and J. A. Sethian. Computing interface motion in compressible gas dynamics. *J. Comput. Phys.*, 100(2):209–228, 1992.

[103] R. Nabben and C. Vuik. A comparison of Deflation and Coarse Grid Correction applied to porous media flow. *SIAM J. Numer. Anal.*, 42:1631–1647, 2004.

[104] R. Nabben and C. Vuik. A comparison of deflation and the balancing preconditioner. *SIAM J. Sci. Comput.*, 27:1742–1759, 2006.

[105] R. Nabben and C. Vuik. A comparison of abstract versions of deflation, balancing and additive coarse grid correction preconditioners. *Numer. Lin. Alg. Appl.*, 15(4):355–372, 2008.

[106] R. A. Nicolaides. On the l^2 convergence of an algorithm for solving finite element equations. *Math. Comp.*, 31:892–906, 1977.

[107] R. A. Nicolaides. On some theoretical and practical aspects of multigrid methods. *Math. Comp.*, 33:933–952, 1979.

[108] R. A. Nicolaides. Deflation of conjugate gradients with applications to boundary value problems. *SIAM J. Numer. Anal.*, 24(2):355–365, 1987.

[109] Y. Notay. Flexible conjugate gradients. *SIAM J. Sci. Comput.*, 22(4):1444–1460, 2000.

[110] S. Osher and R. P. Fedkiw. Level set methods: an overview and some recent results. *J. Comput. Phys.*, 169(2):463–502, 2001.

[111] A. Padiy, O. Axelsson, and B. Polman. Generalized augmented matrix preconditioning approach and its application to iterative solution of ill-conditioned algebraic systems. *SIAM J. Matrix Anal. Appl.*, 22(3):793–818, 2000.

[112] M. L. Parks, E. de Sturler, D. D. Johnson G. Mackey, and S. Maiti. Recycling Krylov subspaces for sequences of linear systems. *SIAM J. Sci. Comput.*, 28(5):1651–1674, 2006.

[113] S. V. Patankar. *Numerical Heat Transfer and Fluid Flow*. McGraw-Hill, New York, 1980.

[114] L. F. Pavarino and O. B. Widlund. Balancing Neumann-Neumann methods for incompressible Stokes equations. *Comm. Pure Appl. Math.*, 55(3):302–335, 2002.

[115] E. G. Puckett, A. S. Almgren, J. B. Bell, D. L. Marcus, and W. J. Rider. A high-order projection method for tracking fluid interfaces in variable density incompressible flows. *J. Comp. Phys.*, 130:269–282, 1997.

[116] M. Raw. Robustness of coupled algebraic multigrid for the Navier-Stokes equations. Technical Paper 96-0297, AIAA Press, Washington, D.C., 1996.

[117] J. K. Reid. On the method of conjugate gradients for the solution of large sparse linear equations. In J. K. Reid, editor, *Large Sparse Sets of Linear Equations*, pages 231–254. Academic Press, New York, NY, USA, 1971.

[118] J. W. Ruge and K. Stüben. Algebraic multigrid (AMG). In S. F. McCormick, editor, *Multigrid Methods*, volume 3 of *Frontiers in Applied Mathematics*, pages 73–130. SIAM, Philadelphia, PA, 1987.

[119] Y. Saad. ILUM: a multi-elimination ILU preconditioner for general sparse matrices. *SIAM J. Sci. Comput.*, 17(4):830–847, 1996.

[120] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, PA, USA, 2003. Second edition.

[121] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, 1986.

[122] Y. Saad, M. Yeung, J. Erhel, and F. Guyomarc'h. A deflated version of the Conjugate Gradient algorithm. *SIAM J. Sci. Comput.*, 21(5):1909–1926, 2000.

[123] R. Scheichl and E. Vainikko. Additive Schwarz and aggregation-based coarsening for elliptic problems with highly variable coefficients. *Comp.*, 80(4):319–343, 2007.

[124] V. Simoncini and D. B. Szyld. On the occurrence of superlinear convergence of exact and inexact Krylov subspace methods. *SIAM Rev.*, 47(2):247–272, 2005.

[125] R. Singh and W. Shyy. Three-dimensional adaptive cartesian grid method with conservative interface restructuring and reconstruction. *J. Comput. Phys.*, 224(1):150–167, 2007.

[126] B. F. Smith, P. E. Bjørstad, and W. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Cambridge University Press, Cambridge, UK, 1996.

[127] G. W. Stewart. Perturbation bounds for the definite generalized eigenvalue problem. *Lin. Alg. Appl.*, 23:69–85, 1979.

[128] G. Strang. *Introduction to Linear Algebra.* Wellesley-Cambridge Press, Wellesley, MA, 1993.

[129] K. Stüben. An introduction to algebraic multigrid. In U. Trottenberg, C. Oosterlee, and A. Schüller, editors, *Multigrid*, pages 413–528. Academic Press, San Diego, CA, 2001.

[130] M. Sussman and E. G. Puckett. A coupled level set and volume-of-fluid method for computing 3D and axisymmetric incompressible two-phase flows. *J. Comput. Phys.*, 162(2):301–337, 2000.

[131] M. Sussman, K. M. Smith, M. Y. Hussaini, M. Ohta, and R. Zhi-Wei. A sharp interface method for incompressible two-phase flows. *J. Comput. Phys.*, 221(2):469–505, 2007.

[132] J. M. Tang, S. P. MacLachlan, R. Nabben, and C. Vuik. A comparison of two-level preconditioners based on multigrid and deflation. DIAM Report 08-05, Delft University of Technology, Delft, 2006.

[133] J. M. Tang, S. P. MacLachlan, R. Nabben, and C. Vuik. Theoretical comparison of two-level preconditioners based on multigrid and deflation. 2008. Submitted.

[134] J. M. Tang, R. Nabben, C. Vuik, and Y. A. Erlangga. Theoretical and numerical comparison of various projection methods derived from deflation, domain decomposition and multigrid methods. DIAM Report 07-04, Delft University of Technology, Delft, 2007.

[135] J. M. Tang, R. Nabben, C. Vuik, and Y. A. Erlangga. Comparison of two-level pcg methods derived from deflation, domain decomposition and multigrid methods. 2008. Submitted.

[136] J. M. Tang and C. Vuik. On the theory of deflation and singular symmetric positive semi-definite matrices. DIAM Report 05-06, Delft University of Technology, Delft, 2005.

[137] J. M. Tang and C. Vuik. Parallel deflated CG methods applied to moving boundary problems. Literature overview. DIAM Report 05-02, Delft University of Technology, Delft, 2005.

[138] J. M. Tang and C. Vuik. Deflated ICCG method applied to 3-D multi-phase flows. In T.E. Simos, G. Psihoyios, and Ch. Tsitouras, editors, *Extended Abstracts, ICNAAM 2006, International Conference of Numerical Analysis and Applied Mathematics*, pages 323–326, Weinheim, 2006. Wiley.

[139] J. M. Tang and C. Vuik. Deflated ICCG method solving the singular and discontinuous diffusion equation derived from 3-D multi-phase flows. In P. Wesseling, E. Onate, and J. Periaux, editors, *European Conference on Computational Fluid Dynamics ECCOMAS CFD 2006*, Delft, 2006. TU Delft.

[140] J. M. Tang and C. Vuik. An efficient deflation method applied to 2-D and 3-D bubbly flow problems. DIAM Report 06-01, Delft University of Technology, Delft, 2006.

[141] J. M. Tang and C. Vuik. New variants of deflation techniques for bubbly flow problems. DIAM Report 06-14, Delft University of Technology, Delft, 2006.

[142] J. M. Tang and C. Vuik. Acceleration of preconditioned Krylov solvers for bubbly flow problems. In Y. Shi, G. D. van Albada, and J. Dongarra, editors, *Computational Science - ICCS 2007. 7th International Conference, Beijing China, May 27-30, 2007, Proceedings, Part I*, Lecture Notes in Computer Science, Vol. 4487, pages 603–614, Berlin, 2007. Springer.

[143] J. M. Tang and C. Vuik. Efficient deflation methods applied to 3-D bubbly flow problems. *Elec. Trans. Numer. Anal.*, 26:330–349, 2007.

[144] J. M. Tang and C. Vuik. Fast deflation methods with applications to two-phase flows. DIAM Report 07-10, Delft University of Technology, Delft, 2007.

[145] J. M. Tang and C. Vuik. New variants of deflation techniques for bubbly flow problems. *J. Numer. Anal. Indust. Appl. Math.*, 2(3–4):227–249, 2007.

[146] J. M. Tang and C. Vuik. On deflation and symmetric positive semi-definite matrices. *J. Comput. Appl. Math.*, 206(2):603–614, 2007.

[147] J. M. Tang and C. Vuik. Acceleration of preconditioned Krylov solvers for bubbly flow problems. In R. Wyrzykowski, J. Dongarra, K. Karczweski, and J. Wasniewski, editors, *Parallel Processing and Applied Mathematics. 7th International Conference, PPAM 2007. Gdansk, Poland, September 2007. Revised Papers*, Lecture Notes in Computer Science, Vol. 4967, pages 1323–1332, Berlin, 2008. Springer.

[148] J. M. Tang and C. Vuik. Fast deflation methods with applications to two-phase flows. *Int. J. Multisc. Comput. Eng.*, 6(1):13–24, 2008.

[149] O. Tatebe. The multigrid preconditioned conjugate gradient method. In N. D. Melson, T. A. Manteuffel, and S. F. McCormick, editors, *Sixth Copper Mountain Conference on Multigrid Methods*, volume CP 3224, pages 621–634, Hampton, VA, 1993. NASA.

[150] A. Toselli and O. Widlund. *Domain Decomposition Methods - Algorithms and Theory*, volume 34 of *Springer Series in Computational Mathematics*. Springer, Berlin, 2004.

[151] U. Trottenberg, C. W. Oosterlee, and A. Schüller. *Multigrid.* Academic Press, London, 2000.

[152] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y.-J. Jan. A front-tracking method for the computations of multiphase flow. *J. Comput. Phys.*, 169:708–759, 2001.

[153] G. Tryggvason, A. Esmaeeli, J. Lu, and S. Biswas. Direct numerical simulations of gas/liquid multiphase flows. *J. Comput. Phys.*, 38(9):660–681, 2006.

[154] S. P. van der Pijl. Computation of bubbly flows with a mass-conserving level-set method. PhD thesis, Delft University of Technology, 2005.

[155] S. P. van der Pijl, A. Segal, and C. Vuik. Modelling of three-dimensional multiphase flows with a mass-conserving level-set method. DIAM Report 06-10, Delft University of Technology, Delft, 2006.

[156] S. P. van der Pijl, A. Segal, C. Vuik, and P. Wesseling. A mass-conserving Level-Set method for modelling of multi-phase flows. *Int. J. Numer. Methods Fluids*, 47:339–361, 2005.

[157] S. P. van der Pijl, A. Segal, C. Vuik, and P. Wesseling. Computing three-dimensional two-phase flows with a mass-conserving level set method. *Comput. Vis. Sci.*, 2008. To appear.

[158] A. van der Sluis. Condition, equilibration, and pivoting in linear algebraic systems. *Numer. Math.*, 15:74–86, 1970.

[159] A. van der Sluis and H. A. van der Vorst. The rate of convergence of conjugate gradients. *Numer. Math.*, 48(5):543–560, 1986.

[160] H. A. van der Vorst. Bi-CGSTAB: a fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13(2):631–644, 1992.

[161] H. A. van der Vorst. *Iterative Krylov Methods for Large Linear Systems.* Cambridge University Press, Cambridge, 2003.

[162] J. van Kan. A second-order accurate pressure-correction scheme for viscous incompressible flow. *SIAM J. Sci. Stat. Comput.*, 7(3):870–891, 1986.

[163] J. van Kan, C. Vuik, and P. Wesseling. Fast pressure calculation for 2D and 3D time dependent incompressible flow. *Num. Lin. Alg. Appl.*, 7:429–447, 2000.

[164] P. Vaněk. Acceleration of convergence of a two-level algorithm by smooth transfer operators. *Appl. Math.*, 37:265–274, 1992.

[165] P. Vaněk. Fast multigrid solvers. *Appl. Math.*, 40:1–20, 1995.

[166] P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid by smooth aggregation for second and fourth order elliptic problems. *Comput.*, 56:179–196, 1996.

[167] R. S. Varga. *Matrix Iterative Analysis*. Prentice-Hall, Englewood Cliffs, N.J., 1962.

[168] J. Verkaik. Deflated Krylov-Schwarz domain decomposition for the incompressible Navier-Stokes equations on a colocated grid. Msc thesis, Delft University of Technology, 2003. Available on `http://ta.twi.tudelft.nl/nw/users/vuik/numanal/verkaik_afst.pdf`.

[169] J. Verkaik, C. Vuik, B. D. Paarhuis, and A. Twerda. The deflation accelerated Schwarz method for CFD. In V. S. Sunderam, G. D. van Albada, P. M. A. Sloot, and J. J. Dongarra, editors, *Computational Science-ICCS 2005: 5th International Conference, Atlanta, GA, USA, May 22-25, 2005, Proceedings, Part I*, pages 868–875, Berlin, 2005. Springer. Lecture Notes in Computer Science 3514.

[170] F. Vermolen, C. Vuik, and A. Segal. Deflation in preconditioned conjugate gradient methods for finite element problems. In M. Křížek, P. Neittaanmäki, R. Glowinski, and S. Korotov, editors, *Conjugate Gradient and Finite Element Methods*, pages 103–129. Springer, Berlin, 2004.

[171] C. Vuik and J. Frank. Coarse grid acceleration of a parallel block preconditioner. *Fut. Gener. Comput. Syst.*, 17:933–940, 2001.

[172] C. Vuik, R. Nabben, and J. M. Tang. Deflation acceleration for domain decomposition preconditioners. In P. Wesseling, C.W. Oosterlee, and P. Hemker, editors, *Proc. 8th European Multigrid Conference, September 27-30, 2005, Scheveningen, The Netherlands*. Delft University of Technology, 2006.

[173] C. Vuik, A. Segal, and J. A. Meijerink. An efficient preconditioned CG method for the solution of a class of layered problems with extreme contrasts in the coefficients. *J. Comput. Phys.*, 152:385–403, 1999.

[174] C. Vuik, A. Segal, J. A. Meijerink, and G. T. Wijma. The construction of projection vectors for a Deflated ICCG method applied to problems with extreme contrasts in the coefficients. *J. Comput. Phys.*, 172:426–450, 2001.

[175] C. Vuik, A. Segal, L. El Yaakoubi, and E. Dufour. A comparison of various deflation vectors applied to elliptic problems with discontinuous coefficients. *Appl. Numer. Math.*, 41:219–233, 2002.

[176] X.-H. Wen and J. J. Gómez-Hernández. Upscaling hydraulic conductivities in heterogeneous media: An overview. *J. Hydrology*, 183:9–32, 1996.

[177] P. Wesseling. Cell-centered multigrid for interface problems. *J. Comput. Phys.*, 79:85–91, 1988.

[178] P. Wesseling. *An Introduction to Multigrid Methods*. John Wiley & Sons, Chichester, 1992. Corrected Reprint. Philadelphia: R.T. Edwards, Inc., 2004.

[179] J. H. Wilkinson. *The algebraic eigenvalue problem*. Oxford University Press, Inc., New York, NY, USA, 1988.

[180] D. M. Young. *Iterative Solutions of Large Linear Systems*. Academic Press, New York, 1971.

List of Publications

Journal Papers

- J.M. Tang and C. Vuik, 'On Deflation and Symmetric Positive Semi-Definite Matrices', *Journal of Computational and Applied Mathematics*, Vol. **206**, Issue 2, pp. 603–614, 2007.
- J.M. Tang and C. Vuik, 'Efficient Deflation Methods applied to 3-D Bubbly Flow Problems', *Electronic Transactions on Numerical Analysis*, Vol. **26**, pp. 330–349, 2007.
- J.M. Tang and C. Vuik, 'New Variants of Deflation Techniques for Pressure Correction in Bubbly Flow Problems', *Journal of Numerical Analysis, Industrial and Applied Mathematics*, Vol. **2**, Issue 3–4, pp. 227–249, 2007.
- J.M. Tang and C. Vuik, 'Fast Deflation Methods with Applications to Two-Phase Flows, *International Journal for Multiscale Computational Engineering*, Vol. **6**, Issue 1, pp. 13–24, 2008.
- S.P. MacLachlan, J.M. Tang, and C. Vuik, 'Fast and Robust Solvers for Pressure Correction in Bubbly Flow Problems', *submitted* (2007).
- J.M. Tang, R. Nabben, C. Vuik, and Y.A. Erlangga, 'Comparison of Two-Level PCG Methods derived from Deflation, Domain Decomposition and Multigrid Methods', *submitted* (2008).
- J.M. Tang, S.P. MacLachlan, R. Nabben, and C. Vuik, 'A Comparison of Two-Level Preconditioners based on Deflation and Multigrid', *submitted* (2008).
- E. Ludwig, R. Nabben, and J.M. Tang, 'Deflation and Projection Methods applied to Positive Semi-Definite Systems', *in preparation*.

Proceeding Papers

- C. Vuik, R. Nabben, and J.M. Tang, 'Deflation Acceleration for Domain Decomposition Preconditioners', *Proceedings of the 8th European Multigrid Conference on Multigrid, Multilevel and Multiscale Methods, The Hague, The Netherlands, September 27-30, 2005, (Eds: P. Wesseling, C.W. Oosterlee, P. Hemker)*, CDROM ISBN 90-9020969-7.

- J.M. Tang and C. Vuik, 'Deflated ICCG Method solving the Singular and Discontinuous Diffusion Equation derived from 3-D Multi-Phase Flows', *Proceedings of ECCOMAS CFD 2006, Egmond aan Zee, The Netherlands, September 5-8, 2006, (Eds: P. Wesseling, E. Onate, J. Periaux)*, CDROM ISBN 90-9020970-0.

- J.M. Tang and C. Vuik, 'Deflated ICCG Method applied to 3-D Multi-Phase Flow', *Proceedings of International Conference on Numerical Analysis and Applied Mathematics 2006 (ICNAAM-2006), Hersonnisos, Crete, Greece, September 15–19, 2006 (Eds: T.E. Simos, G. Psihoyios, Ch. Tsitouras)*, Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim, pp. 323–326, ISBN 3-527-40743-X.

- J.M. Tang and C. Vuik, 'Acceleration of Preconditioned Krylov Solvers for Bubbly Flow Problems', *Computational Science - ICCS 2007. 7th International Conference, Beijing, China, May 27–30, 2007, Proceedings, Part I (Eds.: Y. Shi, G.D. van Albada, J. Dongarra, P.M.A. Sloot)*, LNCS Vol. 4487, pp. 874–881, 2007.

- J.M. Tang and C. Vuik, 'Acceleration of Preconditioned Krylov Solvers for Bubbly Flow Problems', *Parallel Processing and Applied Mathematics. 7th International Conference, PPAM 2007. Gdansk, Poland, September 2007. Revised Papers (Eds.: R. Wyrzykowski, J. Dongarra, K. Karczweski, J. Wasniewski)*, LNCS Vol. 4967, pp. 1323–1332, 2008.

Technical Reports

- J.M. Tang, 'Parallel Deflated CG Methods applied to Moving Boundary Problems. Literature Overview', DIAM Report **05-02**, Delft University of Technology, 2005.

- J.M. Tang and C. Vuik, 'On the Theory of Deflation and Singular Symmetric Positive Semi-Definite Matrices', DIAM Report **05-06**, Delft University of Technology, 2005.

- J.M. Tang and C. Vuik, 'An Efficient Deflation Method applied to 2-D and 3-D Bubbly Flow Problems', DIAM Report **06-01**, Delft University of Technology, 2006.

- J.M. Tang and C. Vuik, 'New Variants of Deflation Techniques for Bubbly Flow Problems', DIAM Report **06-14**, Delft University of Technology, 2006.

- J.M. Tang, R. Nabben, C. Vuik, and Y.A. Erlangga, 'Theoretical and Numerical Comparison of Various Projection Methods derived from Deflation, Domain Decomposition and Multigrid Methods', DIAM Report **07-04**, Delft University of Technology, 2007.
- J.M. Tang and C. Vuik, 'Fast Deflation Methods with Applications to Two-Phase Flows', DIAM Report **07-10**, Delft University of Technology, 2007.
- S.P. MacLachlan, J.M. Tang, and C. Vuik, 'Fast and Robust Solvers for Pressure Correction in Bubbly Flow Problems', DIAM Report **08-01**, Delft University of Technology, 2008.
- J.M. Tang, S.P. MacLachlan, R. Nabben, and C. Vuik, 'Theoretical Comparison of Two-Level Preconditioners based on Deflation and Multigrid', DIAM Report **08-05**, Delft University of Technology, 2008.

Miscellaneous Contributions

- Jok Tang and Kees Vuik, 'Deflated PCG Method for the Poisson Solver', *Burgersdag 2008*, Delft, The Netherlands, January 10, 2008. Winner of the Best Poster Presentation.
- Jok M. Tang, 'A Generalized Projected CG Method with Applications to Bubbly Flow Problems', *IMACS 2008: 9th IMACS International Symposium on Iterative Methods in Scientific Computing*, Lille, France, March 17-21, 2008. Winner of the Student Paper Competition.

Relevant Talks

- 'Deflated ICCG Methods applied to 3-D Multi-Phase Problems', *9th Copper Mountain Conference on Iterative Methods*, Colorado, USA, April 2006.
- 'Deflated ICCG Method applied to 3-D Multi-Phase Flows', *Euromech Colloquium 479: Numerical Simulation of Multiphase Flows with Deformable Interfaces*, Scheveningen, The Netherlands, August 2006.
- 'Deflated ICCG Methods applied to 3-D Multi-Phase Problems', *ECCOMAS CFD 2006: European Conference on Computational Fluid Dynamics*, Egmond aan Zee, The Netherlands, September 2006.
- 'Deflated ICCG Methods applied to 3-D Multi-Phase Problems', *ICNAAM 2006: International Conference of Numerical Analysis and Applied Mathematics*, Crete, Greece, September 2006.
- 'Deflation Method applied to 3-D Bubbly Flow Problems', *Diplomanden- und Doktorandenseminar Numerische Mathematik WS 2006/07*, Berlin, Germany, November 2006.

- ‘Versnellen van Numerieke Methoden voor de Berekening van Stromingen met Bellen en Bubbels’, *43rd Dutch Mathematical Conference 2007*, Leiden, The Netherlands, April 2007. Selected by a jury for the competition of the Philips Mathematics Prize for PhD-students.
- ‘Acceleration of Preconditioned Krylov Solvers for Bubbly Flow Problems’, *ICCS 2007: International Conference on Computational Science 2007*, Beijing, China, May 2007.
- ‘Deflated PCG Method applied to Bubbly Flow Problems’, *Computing Laboratory Seminar*, Oxford, England, August 2007.
- ‘Acceleration of Preconditioned Krylov Solvers for Bubbly Flow Problems’, *PPAM 2007: Parallel Processing and Applied Mathematics*, Gdańsk, Poland, September 2007.
- ‘Comparison of Projection Methods with Applications to Bubbly Flows’, *Diplomanden- und Doktorandenseminar Numerische Mathematik WS 2007/08*, Berlin, Germany, November 2007.
- ‘A Generalized Two-Level PCG Method’, *Gene Golub DCSE Symposium*, Delft, The Netherlands, February 2008.
- ‘A Generalized Two-Level Preconditioned Conjugate Gradient Method’, *IMACS 2008: 9th IMACS International Symposium on Iterative Methods in Scientific Computing*, Lille, France, March 2008.
- ‘A Generalized Two-Level Preconditioned Conjugate Gradient Method’, *10th Copper Mountain Conference on Iterative Methods*, Colorado, USA, April 2008.
- ‘Two-Level Preconditioned Conjugate Gradient Methods’, *ECCOMAS 2008: 5th. European Congress on Computational Methods in Applied Sciences and Engineering*, Venice, Italy, July 2008.

Curriculum Vitae

Jok Man Tang was born on September 1, 1981, in Utrecht, The Netherlands. He completed secondary school at St. Bonifatius College (Utrecht) in 1999. From 1999 to 2004, he studied Applied Mathematics at Delft University of Technology. He obtained his Bachelor of Science degree in 2003, followed by his Master of Science degree (cum laude) in 2004. His Master's thesis, '*Construction of a Combined Preconditioner for the Helmholtz Problem*', was carried out in the numerical analysis group of Prof. P. Wesseling at Delft University of Technology, in collaboration with Shell International Exploration and Production, under supervision of Prof. C. Vuik and Prof. W. Mulder.

He worked as a PhD student in the numerical analysis group at Delft University of Technology from October 2004 to September 2008. He was supervised by Prof. C. Vuik, and has collaborated with especially Prof. R. Nabben (Technische Universität Berlin, Germany) and S.P. MacLachlan (Tufts University, USA), which have led to several publications. He visited University of Oxford and Technische Universität Berlin several times for his work. In addition, the research has been presented at many international conferences and symposiums.

He was awarded prizes for the *Best Poster Presentation* (Burgersdag 2008, Delft, January 2008) and *Best Student Paper* (9th IMACS International Symposium on Iterative Methods in Scientific Computing, Lille, France, March 2008). Moreover, he has taught linear algebra and was a teaching assistant for differential equations and numerical analysis courses. In addition, he has co-organized PhDays 2007 (Baarschot, April 2007), the Gene Golub DCSE Symposium (Delft, February 2008), and PhDays 2008 (De Haan, Belgium, April–June 2008).