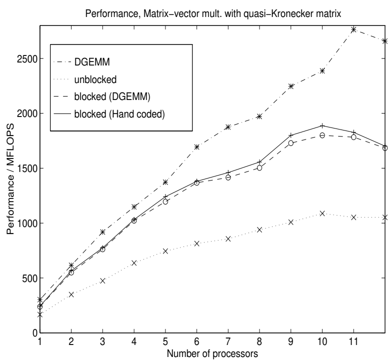
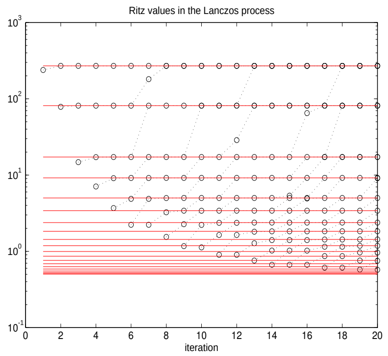
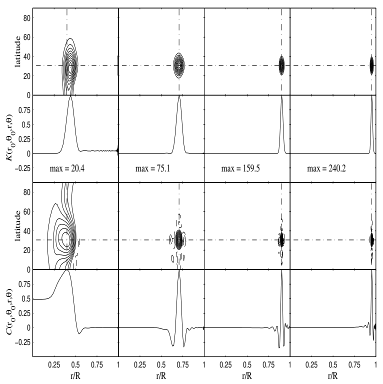
Efficient algorithms for helioseismic inversion

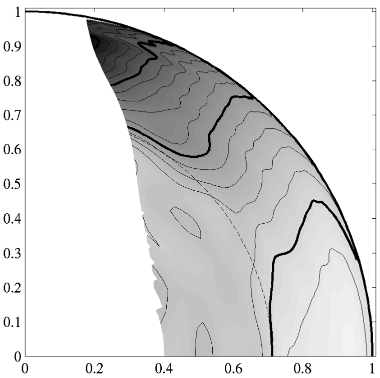
Rasmus Munk Larsen





$$F \otimes G$$

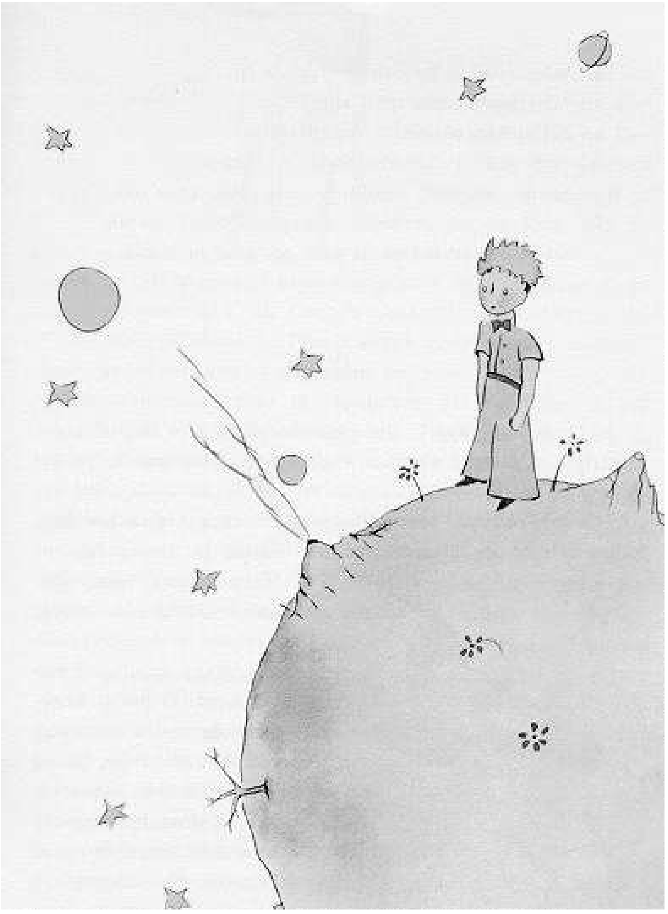




1998

Contents

Preface



Come away, O human child!
To the waters and the wild
With a faery, hand in hand,
For the world's more full of weeping than you can understand.
— W. B. Yeats, from "The Stolen Child"

-C'est le temps que tu as perdu pour ta rose
qui fait ta rose si importante.
-C'est le temps que j'ai perdu pour ma rose...
fit le petit prince, afin de se souvenir.
— A. de Saint-Exupéry, from "Le Petit Prince"

When once the Sun sinks in the west,
And dewdrops pearl the evening's breast;
Almost as pale as moonbeams are,
Or its companionable star,
The evening primrose opes anew
Its delicate blossoms to the dew
And, hermite-like, shunning the light,
Wastes its fair bloom upon the night;
Who, blindfold to its fond caresses,
knows not the beauty he possesses.
Thus it blooms on while night is by;
When day looks out with open eye,
'Bashed at the gaze it cannot shun,
It faints and withers and is gone.
— John Clare, "The Evening Primrose"

Acknowledgements

Das Wandern ist des Müllers Lust,
Das Wandern!
Das Muß ein schlechter Müller sein,
Dem niemals fiel das Wandern ein,
Das Wandern.

— W. Müller, from "Die Schöne Müllerin"

During the four years of my Ph.D. project I have been very fortunate to have been given the opportunity of working at many different institutions and with a number of people from various areas of science. I started out in the Department of Computer Science at Aarhus University, went on to spend a year at the Department for Mathematical Modelling (IMM) at the technical university (DTU) in Lyngby, where I worked with people from both IMM and UNI•C. When returning to Århus, I (at least formally) occupied an office in the astronomy department before going off to spend 7 months in JILA at the University of Colorado in Boulder. During this stay I was very fortunate to be invited to spend the first in a series of very inspiring visits to the Solar Oscillation Investigations (SOI) group at Stanford University.

Of the many people I wish to thank, I am first and foremost deeply indebted to my scientific advisors Jørgen Christensen-Dalsgaard, Ole Caprani, Ole Østerby and Per Christian Hansen.

It tells a lot about Jørgens' heroic character that he was brave enough to take this wacky computer scientist "on board", and in the four years as my primary advisor he never failed to share my enthusiasm and a vision that the intermarriage between inverse helioseismology, numerical analysis and computer science would be a fruitful one (hopefully this thesis will not convince him of the opposite). By allowing me to "wandern" (roam) into many areas with no or little relation to helioseismic inversion, such as singing, and by believing even in my most ludicrous ideas, he did not make finishing my Ph.D. any easier, but undoubtedly helped me becoming a better and more independent scientist. Apart from being very patient and putting up with me in many ways, I can never thank him enough for introducing me to the helioseismology community or for dragging me along to meetings in Boulder (giving me Niwot's curse in the process) – something that has had profound impact on my scientific as well as my personal development. Let me finally give him the ultimately flattering compliment in my book (although completely superfluous since he is not on my Ph.D. committee): His taste in music is almost as good as mine. By the way: Jørgen has made me realize that knowledge of politics can be important. In particular, he and Jesper Schou have trained me well in the traditional Danish art of sweeping things under carpets.

Without the help and encouragement of Ole Caprani, who acted as my personal manager and took over the practical side of my (scientific) life in the last difficult months of writing up and (necessary) social isolation, this thesis would simply not have been. The extent of Ole's help and self sacrifice is most clearly illustrated in the occasional suppression of his natural sarcasm when trying to cheer me up. As much as I thank him for this, I must also warn him that the days when I let him beat me at tennis are over. Thanks to Ole and Monna for the great post-slaughter celebrations with the TA's from the "Arkitektur" course at your house, and to Anders for a couple of nice doubles matches of tennis. By the way, beating your old man is really no problem – just go for his backhand kid!

My warmest thanks and respect go to Ole Østerby for being a dear friend and for fighting – almost single-handedly – the good struggle for numerical analysis at DAIMI. When the first two years of computer science studies did not live up to my expectations, Ole helped rekindle my motivation in his courses on numerical analysis and numerical linear algebra, and laid the foundation for what I have learned ever since. Apart from being one of nature's gentlemen ("ejer hjertets dannelse" in Danish, according to my dictionary(!)), Ole should also have credit for having spent many an afternoon in his office, listening patiently to my crazy and often rather vague ideas for new algorithms, and for sharing memories from his and Ruth's own "American experience".

One of Jørgen's greatest achievements as (actually before he became) my advisor was to introduce me to Professor Per Christian Hansen, who has been my other great scientific mentor and idol. Per has taught me all or most of what I know about regularization algorithms, not to mention scientific writing, and his great insight into numerical algorithms and the clarity of his scientific presentations, oral or written, has been a great source of inspiration to me. Let me express my hopes for fruitful future collaboration on some of our many shared research interests.

From Aarhus University I would also like to thank Peter Møller-Nielsen for distracting me from my writing on many occasions although he was under strict orders not to, and my good friend Stig Larsen for spending many a late night with me, either at work or in a cozy café over a good Cognac and for keeping in touch while I was in Boulder. Jakob Vinther and I are both experts in the noble art of self-pity and have indulged in this exercise over many a meal or beer while discussing physics; may we do so many times in the future! Also, please forgive all the quotations in German in this thesis, but "Deh aerr joh bahre Mathemathiek"! Thanks to Kian for being a great officemate, for introducing me to various cocktails and for insistingly trying to explain stuff to me – I am not sure I ever understood any of his stuff. Also thanks to Torben Amtoft, with whom I have shared an office (or rather: worked in shifts) this last year, for many a good chat about career prospects etc. and for keeping me up to date on the latest results in the Danish soccer league. Thanks to Carsten Butz, Paola Quaglia – (spineless?) "night crawlers" like myself – Flemming Friche Rodler, Dan Witzner and Thomas Stidsen for many a good discussion over a cup of coffee. Thanks to Bo Holm Jakobsen for organizing some very inspiring meetings in the Danish Interdisciplinary Inversion Group, for suggesting a number of references to inverse geoseismology, and for making me realize that mathematical operators have feelings too, e.g., they can be sanguine.

Many thanks to the secretaries at DAIMI and Annette Skovgaard in the astronomy department for helping with all sorts of practical questions. Special thanks to Charlotte Møller Nielsen for being a great help in fixing the paper work for my travel reimbursements etc. and for many a cozy talk when we were both working late. Thanks to Janne whose smiles and cheerfulness have raised my spirits many times.

Thanks to Astrid, Christina, Hanne, Jan, Niels, Marianne, Thea, Peter, Ola, Stig, Ole and Thomas for good company and the wonderful cakes.

My stay at IMM was one of the best periods of my Ph.D., thanks to the good atmosphere and the inspiring company of and cozy lunches with Allan, Dorthe, Hans, Kaj, Ole, $3 \times$ Per and (occasionally) Stig and last but not least Oliekongen! (My first impression: There is a place in Denmark where there is great expertise in and where they take numerical analysis serious?! Wow! Surprise! Cool!) I still regret that I was too shy (partly due to an extremely impertinent interruption on a similar occasion at DAIMI) to sing you a Schubert lied at the Christmas party, but there is still time I hope! Thanks to Lars and Claus for good company on the not so sunny beaches at Las Palmas, Gran Canaria. Thanks to EPOS and Stig Skelboe for a great weekend and a béarnaise sauce I will not easily forget!

From UNI•C I would like to thank Bo for being a cheerful guy in general, and for not calling the police every time I set off the burglar alarm in particular. Thanks to Martin for many a good chat and for enlightening me on the subject of Wagner's operas, and last but not least to Vivi with whom I really enjoyed sharing an office, and owe many thanks (and an apology) for her, as yet unrewarded, work on parallel implementation of block Householder reorthogonalization, among other things. Thanks to George for many a chat over lunch and for giving me a (slightly biased?) account of the Greek – Turkish question and for explaining the universal importance of buzuki-players.

Thanks to Maiken for being a great companion and friend before and during my stay in Copenhagen and for putting up with me during another period of extreme stress – good luck with your old sheep skeletons and new mathematician!

Thanks to Frank Pijpers for many interesting discussions and e-mail exchanges, both while in Århus and in Boulder. Thanks to Sarbani Basu with whom I shared an office at the astronomy department in Århus for useful discussions, for politely not letting her surprise of my complete ignorance of solar physics show too much, and for being the only physicist I know of, who swears by "Matrix Computations" by Golub and Van Loan instead of "Numerical Recipes" – that is obviously a sign of great wisdom and explains her fine scientific results. Thanks to Søren Frandsen for taking my sometimes rather opinionated views (not that they have changed one bit) on the use (abuse) of "Hansen" with relative calm.

Thanks to Rachel Howe, Mike Thompson and Douglas Gough for various discussions of inversion results and for not giving me too hard a time for presenting "interesting results" caused by more or less (usually less) exotic bugs in my code! Well, Mike still has a chance, which he will undoubtedly take advantage of (as he should!). Thanks to Rachel for helping me with various IDL routines for plotting turning point radii etc.

From my time in Boulder I would like to thank JILA and in particular Juri Toomre for letting me stay there for almost 7 months – you can only love a place and its people, where they install you in an office on the ninth floor with a view of the mountains, and do their best to make your feel at home from the day of your arrival! Although he is probably not aware, I admire Juri tremendously for being such a (in lack of a better, wider word) warm man. Somehow he just seems right and in harmony there in his lovely house in the delightful (another good "Juri word") foothills of the Rockies (in "the waters and the wild") with his family, his granite tables, his good taste in wine (thanks for introducing me to Italian Pinot Gris), his salad(!) and his anecdotes; in short: an epicurean soul akin to my own. Hopefully I will be able to repay his invitations sometime. I also admire Juris' great patience in the face of my stubborn determination (or inability) not to learn about solar physics (maybe I did not quite succeed in this, but I came close).

Many thanks to Julian Elliott for putting up with me while sharing a flat, and especially for dragging me up various peaks and for sharing some spectacular moments in the mountains, skiing and hiking. As an example, I clearly remember our spring hike to the Mountain of the Holy Cross, where the mountains were so intensely beautiful, lush and bursting with life and color that it cannot be expressed in plain prose. Thanks for making a number of very thought-provoking comments on the subject of public schools spawned by an essay by George Orwell, which I lent you – they made a lasting impression as you can see.

Thanks to Steve Tobias and Nic Brummell for many a great round of lunch, spoofing, cappuccino, british slang, teasing W. on the subject of tennis rackets (or was it originally cramp-on's?), and watching the wildlife on The Hill. Thanks to Steve and Emma for dragging me through my first and (so far only) 10k road-race (for some reason my Danish friends start to laugh hysterically whenever I mention this) and for a beautiful (but slightly longer than expected) hike.

Thanks to Artyom, wizard of the JILA machine shop and inventor of the self-propelling-polka-dotted-thermo-nuclear gravimeter and a great driver, for persistently trying to teach me two noble arts mastered by (and explaining the high birth and mortality rates of) all male inhabitants of Sct. Petersburg: To drink vodka and (in consequence) to score. I am afraid I was a poor student of the latter (although I know of a few people, who will probably give me a hard time trying to convince me of the opposite) – mainly, I think, because I could not make myself use what (surprisingly?) turned out to be the ultimate pick-up line with Boulder women: "Hello! I am a Russian scientist.". Good luck in your endeavors involving all of the above my dear friend!

Thanks to Deborah Haber for patiently explaining the intricacies in fitting ring-diagrams and the construction of cat-walks and for great birthday parties and flamenco, and to Mark Rast, Paul Charbonneau and other colleagues at JILA and HAO for many good discussions.

Thanks to my friend Henrik (another true epicure) for a memorable road-trip to the national parks of Utah and Arizona and to my beloved San Juan mountains in the August of 1997 and for sharing many a good bottle of red wine while listening to classical music and (nerds will be nerds) debating important issues such as Unix versus Windows, LaTeX versus Word, and C versus Visual Basic in his cozy and tastefully furnished apartment.

Finally I come to Cheri Morrow, a very dear friend, pioneering helioseismologist and fellow Boulderian epicure. First of all I would like to thank Cheri for teaching me the syntax and semantics of the word epicure; life becomes so much simpler when you know what you are, and in particular makes it easier to identify your kinsmen. Secondly, thanks for patiently explaining that a herd of domestic feline quadruped carnivores is not a herd but a flock! Thanks for encouraging, supporting and caring even in my darkest hour. In this connection I wish to thank Rebecca Folsom for forever after being an unbeatable number one on my internal blues charts: Rebecca is the bluuueziest Boulder babe around!

The people of the SOI-MDI team at Stanford University I would like to thank for inviting me to visit on several occasions, for making me feel welcome and also for organizing the SOHO workshops and team meetings, from which I learned a great deal by attending. First of all I would like to thank to Alexander G. Kosovichev for believing in my ideas for new numerical techniques for 2D inversions (something which I hope is justified by the results in this thesis and inversions of MDI data reported in the papers listed in Appendix G) and for inviting me to Stanford in March 1997. Thanks to Sasha and the other members of the SOI-MDI team, my visits to Stanford have been the most inspiring and productive periods of my Ph.D. due to the very stimulating people and environment in general. I am deeply indebted to Phil Scherrer for making these visits possible through the financial support I received, and

I look forward to going back to Stanford to work. Thanks to Margie for taking good care of me and for booking motel rooms etc. Thanks to Sasha and Jesper Schou for their help and for countless discussions, suggestions and explanations of varies aspects of inversions, software and numerical techniques; also thanks to Rakesh Nigam for trying to explain the mysteries of line asymmetries to a non-physicist like myself and Aaron Birch for taking me to play volleyball (although I could not get out of bed for a week afterwards, and had to call the room service just to go to work). Thanks to Rick Bogart and his family for taking me to dinner and introducing me to the "Stanford Theatre experience", which I also look forward to repeating. Thanks to Amara, Nadege, Terry, Jesper, Brian and Peter for taking me hiking. Thanks to the SOI-MDI team for letting me abuse their computer system and for only protesting mildly when I did. Many sysops could learn from Brians motto which I believe goes approximately like "Computers are made to be used (with loadfactors up to the number of processors but no more)". Finally many thanks to Sasha for introducing me to Gene Golub, and to Gene for many inspiring discussions and for explaining a number of new methods to me, which have been of great use in my work. Also, I wish to thank him for the invitation to a most enjoyable Thanksgiving dinner at his house and for discussions over lunch, often accompanied by Erik Boman, and for introducing me to Michael Saunders. Thanks to Annett for introducing me to the restaurants and to so many nice people in Palo Alto, for showing me around San Francisco and for being a great friend. Thanks for many a late night chat with Amara and good luck in Heidelberg!

Last but not least we come to the phenomenon "Jesper Schou". Hmm what to say? Scientific big brother, stunt programmer (and driver), mountaineer, fellow jyde and very dear friend. For a guy who programs in a way that would have made me kick him out of Ole Caprani's programming course when I was a TA, he has done pretty well in ways of scientific computing – decrypting my way through his 2D inversion code(!) gave me important inspiration to the Kronecker product formulation. He has taught me much of what I know about Fortran archaeology (not to mention various computational details of helioseismic inversions), has introduced me to mountaineering on a trip to beautiful Mount Sneffels, where he took a couple of small detours of a few hundres miles just to make sure I got addicted to the grandeur of the Colorado countryside. When at Stanford, he has been my chauffeur, guide, advisor and taken me on some memorable hikes to Big Basin state park, Yosemite etc. Thanks for many good discussions at work and over dinner, for many e-mails and for never ceasing to encourage and inspire me during the last two and a half years.

All this roaming around, trying to pick up whatever I was capable of learning from the people I met and places I visited, has only been possible because I have had a number of people and institutions supporting me financially and otherwise. First of all I would like to thank the Faculty of Science and the Department of Computer Science at Aarhus University for supplying me with a Ph.D. grant and for supporting my travels. Many thanks also to IMM, the EPOS project and in particular professor Kaj Madsen for supporting me on several occasions, and to UNI•C for supplying me with office facilities during my stay in Lyngby and for access to their computing facilities. In the USA I am indebted to Professor Philip Scherrer from the SOI group at Stanford University and to JILA and Professor Juri Toomre for covering my expenses in connection with numerous visits to Stanford and Boulder and for hosting me while working in their respective groups. Last but not least thanks to Teoretisk Astrofysik Center, Igor Novikov and Per Rex Christensen for supporting me financially the last two months while finishing my thesis. Finally I would like to thank Gene Golub, Per Christian Hansen, Jørgen Christensen-Dalsgaard, Juri Toomre, Phil Scherrer and Jesper Schou for their letters of support in connection with my application to SNF.

Finally and most important I would like to thank my parents Kirsten Munk and Erik Larsen for the love and emotional support you have always given me, for being great friends, and for ultimately being the ones to thank for what I have achieved in music, science and life in general. I apologize for choosing a line of work that has (and probably will in the future) brought me so far from Jels and Denmark, especially at a time where my brother Frederik was also studying abroad. Thanks to Frederik and Lisbeth for giving me some great moments musically as well as socially. Special thanks to my sister Anne Christine, Hans and (this time very) last but absolutely most importantly thanks to Anton for being the most beautiful sweet little boy and without comparison the best nephew in the world (no wonder with those genes).

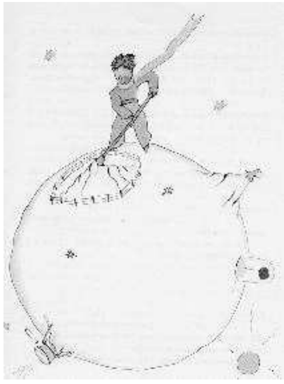
Part I

Efficient Algorithms for Helioseismic Inversion: Mathematical modeling, algorithm design and implementation

Writings from the Postponed Papers of an Itinerant Numerical Analyst

Chapter 1

Introduction



Over hill, over dale,
Thorough bush, thorough brier,
Over park, over pale,
Thorough flood, thorough fire,
I do wander everywhere,
...

— W. Shakespeare

The instruments used for scientific exploration vary significantly from discipline to discipline: researchers working with data from astronomical telescopes, nuclear magnetic resonans scanners, seismographs and scientific satellites observing the Sun would perhaps seem to have very little in common. One phenomenon, however, is common to all disciplines: the ever-increasing use of computers for processing experimental data, testing and simulating mathematical models and for visualizing results.

Computers are particularly important as modelling tools for complex physical systems where it is impossible, or too expensive, to observe or make experiments with the real physical system. The study of the dynamics and evolution of the Earth's climate, formation and evolution of stars and galaxies, physical properties of new complex materials or exotic quantum-mechanical systems in high-energy physics, are all examples of scientific or engineering problems that cannot easily be realized in the laboratory and where the study of mathematical and statistical models via computer experiments plays an essential role. Constructing the model is a process specific to each particular scientific discipline, but the mathematical and statistical problems that have to be solved – a set of differential equations, an optimization problem, an eigenvalue problem or a Markov-chain Monte-Carlo simulation – are often similar and can be treated using the same numerical algorithms.

Although researchers from opposite corners of the scientific arena may use different tools, they all have the same demands: As the amount of experimental data grows and the complexity of the mathematical and statistical models increases, the results produced by the computer must still be correct, the time it takes to produce them should not increase too much, and they should be visualized in an intelligible format.

When the computer is used to perform these calculations, a great variety of tools are at our disposal: programming languages (C, FORTRAN, ...), the use of numerical software libraries (BLAS, LAPACK, Numerical Recipes, ...) and mathematical software tools with graphical capabilities (Mathematica, Matlab, IDL, ...). The techniques used to implement a particular model/algorithm depend on, e.g., the complexity/size of the problem and the architecture of the computer (memory hierarchies/cache, serial vs. parallel, shared versus distributed memory), but also on the *execution environment* (e.g., batch-queue policies, disk quotas) on the system to be used and the *computing traditions* of the particular scientific community.

In our work we have come to realize that in scientific computing, the successful application of computers and numerical techniques in general does not depend on the programming language, visualization software or the type of computer used. What is important when building solutions for complex applications is the ability to recognize and isolate, in the mathematical modelling phase, frequently occuring (sub-) problems that can be solved using standard algorithms. A similar argument can probably be made for many other disciplines, but in scientific computing it is often the case that a given problem may give rise to a computational formulation, which is known to be handled efficiently by an algorithm originally developed for a problem from a completely different application area. In their original abstract formulation, the two problems would most likely look completely unrelated due to the use of different notation or mathematical formalisms. However, when turning the abstract mathematical formulation into a computer implementation, the result nearly always consists of a number of steps that can be identified as belonging to a small set of standard problems from, e.g., numerical linear algebra, non-linear optimization and solution of differential equations. Furthermore, many of these standard algorithms build on a small set of *computational kernels*, such as matrix-factorizations, Fourier transforms and simpler matrix and vectors operations. The philosophy behind modern numerical software libraries, such as BLAS and LAPACK (or ScaLAPACK on distributed memory parallel machines), is to provide robust and efficient implementations of the computational kernels and, in turn, standard algorithms. The advantage is that by expressing an algorithm in terms of these computational kernels, a lot of tedious fine tuning of the code becomes unnecessary.

In other words, this methodology leads in a number of steps more or less directly from a *mathematical model* over the design of a *numerical algorithm* to an *efficient implementation* on modern high-performance computers, where on each of the three levels "standard building blocks" can be identified, and more importantly, have equivalents on the level immediately below.

Sometimes, breaking a complex application into a series of simple problem can help identify which standard algorithm, possibly developed in a completely different application area, can be applied directly. The example we have in mind is the algorithms for helioseismic inversion described in this thesis, where we discovered that standard algorithms for manipulating Kronecker products can be applied in combination with iterative algorithms for solving ill-conditioned least-squares problems to reduce the cost of analyzing the large helioseismic datasets substantially. On the other hand, the use of standard algorithms is usually not what makes a significant improvement. For the helioseismic inversions, we started out treating the problem as a general ill-conditioned least squares problem, and it was believed that the main improvements would come from

1. applying state-of-the-art (standard) algorithms based on Householder bidiagonalization

such as described in Appendix B, and

2. implementing these algorithms on massively parallel computers.

As it turned out, the largest improvement in computational complexity was obtained from insight into the structure of the underlying inverse problem, which was a prerequisite for discovering the Kronecker-product formulation described in Chapter 3 and Appendix C. Designing and implementing the resulting algorithms to make efficient use of the memory hierarchy and parallelizing it, has of course further reduced the execution time, but these improvements are fairly small compared to the effect of the improved algorithm.

If the experience from this thesis project should be summarized in a single sentence (manifesto) it could be: "A successful large-scale numerical application requires physics and mathematics to develop and analyze the model, numerical analysis to develop solution methods, and computer science to develop an efficient implementation." This close interconnection between the areas of *mathematical modeling* and the development of reliable *numerical algorithms* that can be turned into efficient *implementations on large-scale modern computer architectures* could be called "the spirit of Scientific Computing", and has been the guiding principle in the investigations on which this thesis is built. Hopefully the result is "successful" enough to be useful in future analyzes of large helioseismic datasets.

1.1 Overview

The purpose of this thesis is to describe efficient and reliable algorithms for solving large-scale discrete ill-posed problems. This type of calculations constitute the main computational and numerical challenge for the study of inverse problems in a variety of disciplines including astronomy, geophysics and engineering. In this thesis we describe in particular methods for helioseismic inversion, and address a number of both theoretical and practical aspects of algorithm development and implementation. A general overview of regularization algorithms for solving discrete ill-posed problems is given in Chapter 2.

The main goal of this work has been to develop and implement algorithms that deal with the computational and numerical challenges represented by the vast amounts of data from the so-called GONG experiment and the SOI/MDI instrument on board the SOHO satellite. In Chapter 3 we give an introduction to helioseismology including the fundamental theory of solar oscillations and a description of the 2-dimensional helioseismic rotational inverse problem, which has been the motivating problem in most of the investigations reported in this thesis.

During the project we have discovered that the key technique to handle the very large linear least squares problems, which arise when the inverse problem must be solved numerically, is to exploit the special structure of the matrices involved; this structure can be derived from the algebraic form of the modekernels as explained in Chapter 3.

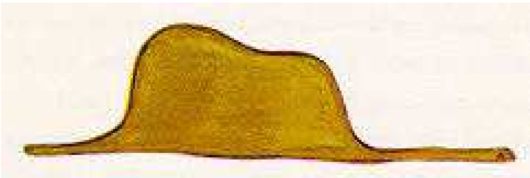
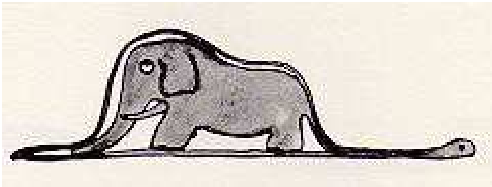
In Chapter 4 we describe how the discovery of the special matrix structure can be combined with the use of iterative regularization algorithms, which are good at taking advantage of the matrix structure. Using this approach we have produced computer programs that implement the optimally localized averages inversion methods, and are able to handle the large inversions using a fraction of the computer resources that were originally thought necessary. In particular, we have found hybrid algorithms based on Lanczos bidiagonalization to be particularly efficient and flexible with respect to finding a good regularization parameter.

However, the use of full reorthogonalization in the Lanczos algorithms could be prohibitively expensive for even larger datasets or when applying the algorithms in other application areas. As an example, it is reported in Chapter 4 that the reorthogonalization already represents 75% of the work in the inversion of the MDI datasets. This has lead us to investigate the algorithm for Lanczos bidiagonalization with partial reorthogonalization described in Appendix A. Our preliminary experiments indicate that this algorithm could reduce the cost of the reorthogonalization by a factor of 2-3. Numerical experiments with test matrices from the Harwell-Boeing collection also show that Lanczos bidiagonalization algorithm with partial reorthogonalization is also very efficient for large sparse SVD calculation, when compared to current state-of-the-art algorithms from ARPACK (Lehoucq et al. [135]) based on implicitly restarted Lanczos or the original symmetric Lanczos algorithm with partial reorthogonalization (e.g. Simon [182, 207]).

The algorithms and programs developed during this thesis project have been used in helioseismic investigations conducted with data from MDI and other experiments including GONG, LOWL and BiSON (see [175, 126, 176, 39, 37]). With the latest helioseismic datasets, the properties of solar oscillation eigenmodes can be determined with unprecedented accuracy, which has made it possible to perform detailed seismic studies of the dynamics of the solar interior, including the study of near-surface flows, differential rotation and momentum transport in and just beneath the convective envelope, which is important for understanding, among other things, the mechanisms behind the Sun's magnetic field, and its variability with the solar cycle. In Chapter 5 we discuss some of the computational aspects particular to the helioseismic inversions. Then we summarize some of the results obtained, and describe how we have tried to assess the reliability of the inversions.

Chapter 2

Algorithms for discrete ill-posed problems

 $\stackrel{?}{\Longrightarrow}$ 

Les grandes personnes ne comprennent jamais rien toutes seules, et c'est fatigant.
— A. Saint-Exupéry, from "Le Petit Prince"

In this chapter we give an overview of algorithms for solving discrete ill-posed problems, which arise in the study of linear inverse problems in a variety of scientific disciplines. Many inverse problems involve the solution of a Fredholm equation of the first kind

$$b(y) = \int_0^1 K(x,y)\, f(x)\, \mathrm{d}x \ , \quad 0 \le y \le 1 \ ,$$

where the function f is to be determined given b and K. In most applications the function b represents some measured quantity, and in practice b is known only at a finite set of point $y_1, y_2, \ldots, y_m$ and moreover b is often contaminated by noise. Hence we may write the above equation in the following generic form

$$b_i = \int_0^1 K_i(x)\, f(x)\, \mathrm{d}x + \epsilon_i \ , \qquad i = 1, \ldots, m \ , \tag{2.1}$$

where the m measurements b_i represent noisy values of m functionals with *kernels* K_i on an unknown function f, and ϵ_i represent the noise. We will assume throughout that the errors ϵ_i are unbiased and Gaussian with covariance matrix $\mathbf{E}$.

Before we discuss the numerical solution, let us make a few general remarks about a number of factors that combine to make the analysis of (2.1) a highly non-trivial task. First of all, the solution to the inverse problem is inherently non-unique, since we are trying to infer a continuous function f from a finite set of measurements. This means that mathematically there are infinitely many solutions f that satisfy Equation (2.1), and this can only be resolved by imposing additional constraints on the solution derived from a priory knowledge (see [118]). In practice, the problem of the non-uniqueness is always "solved" when the problem is discretized since this forces computed solution to be on a particular functional form described by a finite number of parameters. The functional form depends on the discretization as well as the regularization method (in methods such as in the optimally localized averages described below, it is not easy to describe what the assumed functional form is). Another problem is that the inverse problem is ill-conditioned such that solution is very sensitive to noise in the data. This ill-conditioning carries over to the discretized problem, which usually involves solving a linear least-squares problem or a system of linear equations with a very ill-conditioned coefficient matrix. A third challenge is the use of modern automated data acquisition techniques which has led to an explosion in the volume of experimental data available. However, due to the ill-conditioning, standard algorithms for large-scale problems such as sparse LU-factorization and preconditioned conjugate iterations, which have been studied intensely, cannot be applied directly. Finally, the complex observation techniques and subsequent data reduction may, if not modeled carefully, introduce additional systematic errors and error in the data.

To compute a numerical approximation f_{est} to the solution f, it is necessary to discretize Equation (2.1). The two main classes of discretization methods are the *quadrature method* where the equation is discretized by approximating the integrals involved using some quadrature rule, and the *Galerkin method* where the functions are expanded on a chosen set of basis functions (see e.g. [206]). In this presentation we restrict ourselves to consider the quadrature method where the integral in Equation (2.1) is approximated by means of a quadrature rule with weights w_j on the grid $x_1, x_2, \ldots, x_n$, i.e.,

$$\int_0^1 K_i(x)\,\mathrm{d}x \approx \sum_{j=1}^{n} w_j K_i(x_j)\,.$$

Equation (2.1) is thus replaced by the matrix equation

$$\mathbf{A}\,\mathbf{x} = \mathbf{b}\;, \tag{2.2}$$

where $\mathbf{A} \in \mathbb{R}^{m\times n}$, $(\mathbf{A})_{ij} = w_j K_i(x_j)$, $\mathbf{x} = (f(x_1), f(x_2)\ldots, f(x_n))^T$ and $\mathbf{b} = (b_1, b_2\ldots, b_m)^T$. One often refers to $\mathbf{A}$ as the *kernel matrix*. In practice it is mostly the case that $m \geq n$, which we shall assume throughout. For the helioseismic 2-dimensional rotational inversion problem, which will be described in Chapter 3, the number n of quadrature points is of order 10^4 while the number m of kernels (measurements) is of order 10^4–10^6 depending on the dataset used. Hence, Equation (2.2) is overdetermined and is replaced by the least-squares problem

$$\min\;\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2\;. \tag{2.3}$$

When the errors ϵ_i in the measurements are described by the symmetric positive definite error-covariance matrix $\mathbf{E} \neq \mathbf{I}$, the solution $\mathbf{x}$ to Equation (2.3) does not provide the best unbiased linear estimate of the true solution, which instead must be sought as the solution to

$$\min(\mathbf{A}\mathbf{x} - \mathbf{b})^T\mathbf{E}^{-1}(\mathbf{A}\mathbf{x} - \mathbf{b})\;. \tag{2.4}$$

Often, it is convenient to incorporate the statistical information by working in terms of the transformed variables $\hat{\mathbf{A}} = \mathbf{G}^{-T}\mathbf{A}$, $\hat{\mathbf{b}} = \mathbf{G}^{-T}\mathbf{b}$, where $\mathbf{G}$ is the Cholesky factor of $\mathbf{E}$ such that $\mathbf{E} = \mathbf{G}^T\mathbf{G}$; the error-covariance matrix of the transformed data is simply the identity. The solution to Equation (2.4) can now be found by solving

$$\min\;\|\hat{\mathbf{A}}\mathbf{x} - \hat{\mathbf{b}}\|_2\;. \tag{2.5}$$

When the data-errors are assumed to be independent this simplifies to scaling the problem by the estimated standard deviations σ_i of the errors ϵ_i, such that the transformed variables become

$$(\hat{\mathbf{A}})_{ij} = w_j \sigma_i^{-1} K_i(x_j) \ , \quad (\hat{\mathbf{b}})_i = b_i/\sigma_i \ .$$

In this simple setting it is easy to see that the scaling suppresses the influence of data with large errors on the solution by reducing their contribution to the squared residual by a factor of σ_i^2. In the following we will assume that this preprocessing of the data have been performed before any of the regularization algorithm discussed below are applied to solve the inverse problem, and we will write $\mathbf{A}$ instead of $\hat{\mathbf{A}}$ and $\mathbf{b}$ instead of $\hat{\mathbf{b}}$ throughout.

Having sketched the fundamental computational problem involved in analyzing linear inverse problems, we come to the major challenge, which is the main topic of this thesis, namely how to solve Equation (2.5) in a way that is (hopefully) both computationally efficient, numerically stable and which can be implemented to take advantage of modern computer architectures.

As discussed in the introduction, the achievement of all three goals in complicated by the numerical properties of problems arising from the discretization of Fredholm integral equations: The resulting linear least-squares problem in Equation (2.3) often has a very ill-conditioned coefficient matrix, i.e., the condition number $\kappa(\mathbf{A})$ of $\mathbf{A}$ is large, and the solution $\mathbf{x}$ is therefore very sensitive to perturbations in the right-hand-side caused by either noise in the measurements or by round-off errors that occur in the solution process. This can be illustrated by perturbation analysis for the solution to Equation (2.3): Let $\mathbf{x}$ and $\tilde{\mathbf{x}}$ denote the solution to $\min \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$ and the perturbed problem $\min \|(\mathbf{A} + \delta\mathbf{A})\tilde{\mathbf{x}} - (\mathbf{b} + \delta\mathbf{b})\|_2$. If $\mathbf{A}$ has full rank and the elements in $\delta\mathbf{A}$ are so small that

$$\eta \equiv \kappa(\mathbf{A}) \frac{\|\delta\mathbf{A}\|_2}{\|\mathbf{A}\|_2} < 1$$

then $(\mathbf{A} + \delta\mathbf{A})$ also has full rank and

$$\frac{\|\tilde{\mathbf{x}} - \mathbf{x}\|_2}{\|\mathbf{x}\|_2} \leq \frac{\kappa(\mathbf{A})}{1-\eta} \left(\frac{\|\delta\mathbf{A}\|_2}{\|\mathbf{A}\|_2} + \frac{\|\delta\mathbf{b}\|_2}{\|\mathbf{A}\mathbf{x}\|_2} + \eta \frac{\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2}{\|\mathbf{A}\mathbf{x}\|_2} \right) \ , \tag{2.6}$$

see, e.g., [25] for a proof. The noise term, $\delta\mathbf{b}$, typically includes the noise in the given measurements and errors introduced by earlier data processing steps used to produce $b_1, b_2, \ldots, b_m$. The model-error term, $\delta\mathbf{A}$, represent uncertainties in the mathematical model used to describe the underlying physical problem. In the following we will assume that $\delta\mathbf{A}$ is negligible compared to $\delta\mathbf{b}$, and that the kernels $K_1, K_2, \ldots, K_m$ are known exactly. If that is not the case, regularization methods based on total least squares can be used to incorporate the uncertainty in $\mathbf{A}$, see e.g. [74]. In helioseismic rotational inversion this is a good approximation, since it is known (from the study of another inverse problem) that the models of the radial structure of the Sun from which the kernels are calculated are accurate to within a fraction of a percent (cf. [20]), while the uncertainties in $\mathbf{b}$ are one to two orders of magnitude larger.

The ill-conditioning of $\mathbf{A}$ is *not* a result of the discretization but is a property inherent in the continuous inverse problem, i.e., that Equation (2.1) does not contain enough information for us to recover the function f stably. This does not mean that the problem cannot be solved, but standard algorithms for solving linear least squares problems are not applicable to these *ill-posed problems* in their original form. Instead additional a priory information about the solution, e.g., that it is smooth, must be incorporated in order to stabilize the problem such that a single useful solution can be computed. This is called *regularization* (cf. [195]). A number of different regularization methods for solving discrete ill-posed problems have been proposed. Commonly used methods include *Tikhonov regularization*, various methods based on the singular value decomposition (SVD) of $\mathbf{A}$ and *mollifier methods* such as the *optimally localized averages* (Backus-Gilbert) method. Most of these methods are well-studied, and they are used to solve a variety of inverse problems. Below we briefly describe the regularization methods and discuss which numerical algorithms are used to implement them and how suited they are for implementation on modern high-performance computers.

2.1 Regularization methods

The singular-value decomposition (SVD, cf. [83]) is a very valuable tool for analyzing and solving ill-conditioned problems, see [105]. Let $\mathbf{A}$ be a rectangular $m \times n$ matrix with $m \geq n$, then the SVD of $\mathbf{A}$ is of the form

$$\mathbf{A} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^T = \sum_{i=1}^{n} \sigma_i\,\mathbf{u}_i\,\mathbf{v}_i^T \ ,$$

where the matrices $\mathbf{U} \in \mathbb{R}^{m\times m}$ and $\mathbf{V} \in \mathbb{R}^{n\times n}$ are orthogonal and

$$\mathbf{\Sigma} = \begin{array}{c} n \\ m-n \end{array} \begin{bmatrix} \mathbf{\Sigma}_1 \\ 0 \end{bmatrix} ,$$

where $\mathbf{\Sigma}_1 = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$. The distribution of the singular values $\sigma_1, \sigma_2, \ldots, \sigma_n$ gives valuable information about the conditioning of $\mathbf{A}$. Generally for discrete ill-posed problems such as those arising from the discretization of Equation (2.1) the singular values σ_i decay gradually to zero and the singular vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ tend to be more and more oscillatory as the index i increases, which is also the case for helioseismic rotational inversion, see [105, 46]. To see why this is a problem consider the solution to Equation (2.3): If we write it in terms of the SVD of $\mathbf{A}$ we get

$$\mathbf{x} = \sum_{i=1}^{n} \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i \ . \tag{2.7}$$

In order for the original integral equation to have a solution of finite norm, it must satisfy the Picard criterion. In this case the system in Equation (2.3) will satisfy the corresponding "discrete Picard criterion" (see [101]) which means that the Fourier coefficients $\mathbf{u}_i^T\mathbf{b}_{\mathrm{exact}}$ of the noise-free right-hand side must go to zero at least as fast as the singular values. However, in the presence of noise or rounding errors, the coefficients $\mathbf{u}_i^T\mathbf{b}$ will generally not decay to zero, but level off at some value ϵ determined by the noise. The result is that the solution in Equation (2.7) is completely dominated by terms with $\epsilon/\sigma_i \gg 1$, corresponding to the noise components being magnified by division with small σ_i. This usually results in meaningless and wildly oscillating solutions dominated by noise.

2.1.1 SVD-based regularization

In order to stabilize the solution the SVD-based regularization methods apply a "filter" to the expansion in Equation (2.7) to either dampen or simply discard the terms corresponding to $\sigma_i < \epsilon$,

$$\mathbf{x} = \sum_{i=1}^{n} f_i \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i \ . \tag{2.8}$$

The effect of the regularization is determined by the choice of *filter factors* f_i.

Equation (2.8) is a very general formulation that applies to a wide variety of regularization methods. It is possible to characterize many different regularization methods in terms of their filter factors. The most extreme form is *truncated SVD* where the expansion in Equation (2.7) is simply truncated to exclude components in $\mathbf{x}$ corresponding to small σ_i. This corresponds to filter factors f_i given by

$$f_i = \begin{cases} 1 & \text{if } 1 \leq i \leq k \\ 0 & \text{otherwise} \end{cases} , \tag{2.9}$$

where k is the truncation parameter that effectively determines the amount of regularization applied to the problem. Notice that this corresponds to projecting the system in Equation (2.3) onto the subspaces $\text{span}(v_1, \ldots, v_k)$ and $\text{span}(u_1, \ldots, u_k)$ where the operator $\mathbf{A}$ is well-conditioned.

2.1.2 Tikhonov regularization

An alternative to the projecting above is to simply replace $\mathbf{A}$ with a different operator that, in some sense, is close to $\mathbf{A}$ and which is less ill-conditioned. This approach is less drastic than simply truncating the expansion in Equation (2.7), and could be effected by choosing a set of filter coefficients f_i that fall off gradually with i. This type of method includes the widely used Tikhonov regularization method [195], which corresponds to the filter factors

$$f_i = \frac{\sigma_i}{\sigma_i^2 + \lambda^2} , \quad 1 \leq i \leq n \ .$$

It is easy to show that this choice of filter factors corresponds to replacing Equation (2.3) with the problem

$$\min \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 \ + \ \lambda^2 \|\mathbf{x}\|_2^2 \ ,$$

where a solution is sought that both fits the data and has a small 2-norm. Depending on the application, the 2-norm may not be the proper norm to measure the "goodness" of the solution, and a more general formulation is to replace Equation (2.3) with

$$\min \ \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 \ + \ \lambda^2 \|\mathbf{L}\mathbf{x}\|_2^2 \ , \tag{2.10}$$

or equivalently

$$\min \left\| \begin{pmatrix} \mathbf{A} \\ \lambda \mathbf{L} \end{pmatrix} \mathbf{x} - \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix} \right\|_2 \ . \tag{2.11}$$

The choice of $\mathbf{L}$ determines a (semi-) norm of $\mathbf{x}$ that should be minimized while simultaneously trying to fit the data. In this respect, the term $\|\mathbf{L}\mathbf{x}\|_2$ measures a property of f that we want to minimize to incorporate our a priori knowledge of the solution, which is added to make the problem well-conditioned. One example of such a matrix is

$$\mathbf{L} = \begin{pmatrix} 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \end{pmatrix} \in \ \mathbb{R}^{(n-2)\times n} \ ,$$

which for a simple discretization where $\mathbf{x}$ contain samples of f on a equidistant grid would be an approximation to the second derivative operator, i.e., $\|\mathbf{L}\mathbf{x}\|_2^2 \propto \int_0^1 f''(x)^2 \mathrm{d}x$. This choice of $\mathbf{L}$ forces the solution f to be smooth and could be used if we have reason to believe that this is a property of the true solution to the inverse problem. This choice is often used in helioseismic inversions because it is reasonable to assume that the rotation in the Sun is a relatively smoothly varying function of radius and latitude.

The parameter λ is called the *regularization parameter* and it controls the amount of regularization applied to the solution by determining the relative weighting of the two terms in Equation (2.10). Choosing a good value for λ is a problem in statistics that is a complicated further if iterative algorithms are used to implement the regularization methods (see below).

2.1.3 Mollifier methods – optimally localized averages

The mollifier or optimally localized averages (OLA) methods have been used successfully in seismology [149], helioseismology [48] and other areas that involve inverse problems [54]. The mollifier methods, of which the Backus-Gilbert (BG) technique [10] is probably the best known, are based on the idea that given the problem defined by Equation (2.1) we can estimate the value of the solution in a given point x_0 as a linear combination of the measurements

$$f_{\mathrm{est}}(x_0) = \sum_{i=1}^{m} c_i(x_0)\, b_i \ . \tag{2.12}$$

If f denotes the exact solution to Equation (2.1), then it is easy to see from Equation (2.12) that the value of $f_{\mathrm{est}}(x_0)$ is given by

$$f_{\mathrm{est}}(x_0) = \int_0^1 \sum_{i=1}^{m} c_i(x_0)\, K_i(x)\, f(x)\, \mathrm{d}x \ + \sum_{j=1}^{m} c_i(x_0)\, \epsilon_i \ . \tag{2.13}$$

The function

$$\mathcal{K}(x_0, x) \equiv \sum_{i=1}^{m} c_i(x_0)\, K_i(x) \tag{2.14}$$

is called the *averaging kernel* and it shows how the expectation value of $f_{\mathrm{est}}(x_0)$ is a weighted average of the exact solution f. For $f_{\mathrm{est}}(x_0)$ to be meaningful, the function $\mathcal{K}(x_0, x)$ should be peaked around $x = x_0$ and be normalized such that

$$\int_0^1 \mathcal{K}(x_0, x)\, \mathrm{d}x = 1 \ . \tag{2.15}$$

Hence, the averaging kernel $\mathcal{K}(x_0, x)$ often plays a role which is at least as important as the computed estimate $f_{\text{est}}(x_0)$, since it determines the resolving power of the method at x_0. It is also possible to define averaging kernels for other regularization methods such as Tikhonov regularization, and they are often used for comparing different techniques for a particular application; see [48, 175].

The second term on the right-hand side in Equation (2.13) represents the noise propagated from the data to the solution estimate. From the assumption that the errors ϵ_i are unbiased and Gaussian with covariance matrix $\mathbf{E}$ it follows that the expectation value of the noise term is zero with variance

$$\sigma^2(x_0) = \sum_{i=1}^{m}\sum_{j=1}^{m} c_i(x_0)\,\mathbf{E}_{ij}\,c_j(x_0)\ . \tag{2.16}$$

A number of different methods have been proposed for computing the *inversion coefficients* $c_1(x_0), c_2(x_0), \ldots, c_m(x_0)$ such that the averaging kernels $\mathcal{K}(x_0, x)$ have the desired properties. In the BG method the peakedness of $\mathcal{K}(x_0, x)$ is controlled by specifying a weighting function $\mathcal{J}(x_0, x)$ that is zero at x_0 and grows away from x_0 and thus penalizes solutions with non-local contributions to the solution estimate. This leads to a constrained linear least-squares problem

$$\begin{aligned} &\min \int_0^1 [\mathcal{J}(x, x_0)\,\mathcal{K}(x, x_0)]^2\,\mathrm{d}x \;+\; \lambda^2\sigma^2(x_0) \\ &\text{subject to } \int_0^1 \mathcal{K}(x, x_0)\,\mathrm{d}x = 1\ , \end{aligned} \tag{2.17}$$

which must be solved to find the inversion coefficients $c_1(x_0)$, $c_2(x_0)$, $\ldots$, $c_m(x_0)$. The regularization parameter λ is used to trade off between a well-localized $\mathcal{K}$ and a small variance in the propagated error.

In helioseismology this method produces well-localized averaging kernels with little structure away from x_0, see [48] and Chapter 5. Unfortunately, the method is very expensive, since the coefficient matrix in the discretized version of Equation (2.17) depends on x_0, and hence $\mathcal{O}(mn^2)$ floating-point operations are required for each value of x_0.

The so-called *subtractive optimally localized averages* (SOLA) method, see [136, 158], is a more computationally efficient variant of the BG method. The idea in SOLA is to minimize the squared difference between the averaging kernel and some chosen target function $\mathcal{T}(x_0, x)$ that has the desired form. This leads to the minimization problem

$$\begin{aligned} &\min \int_0^1 [\mathcal{K}(x_0, x) - \mathcal{T}(x_0, x)]^2\,\mathrm{d}x \;+\; \lambda^2\sigma^2(x_0) \\ &\text{subject to } \int_0^1 \mathcal{K}(x, x_0)\,\mathrm{d}x = 1\ . \end{aligned} \tag{2.18}$$

It turns out that when Equation (2.18) is discretized, the matrix of the resulting least-squares problem does not depend on x_0 as in the BG method. Instead, it must be solved for a set of different right-hand sides, one for every target position x_0. To distinguish between the two variants, the original BG formulation in Equation (2.17) is sometimes called MOLA, which stands for *multiplicative optimally localized averages.*

In Chapter 4 we return to the OLA methods and describe an algorithm for implementing both SOLA and MOLA efficiently, which has been used for analyzing a number helioseismic datasets, as described in Chapter 5 and Appendices D and G.

2.2 Direct algorithms

The numerical algorithms used to implement the various regularization methods can be divided roughly into two categories: *direct* and *iterative* algorithms. As can be seen from the previous paragraphs, the main computational burden in most methods lies in solving the large-scale regularized least-squares problem

$$\min \left\| \begin{pmatrix} \mathbf{A} \\ \lambda \mathbf{L} \end{pmatrix} \mathbf{x} - \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix} \right\|_2 .$$

This is a classical problem in numerical linear algebra and standard (direct) algorithms for solving it will be briefly discussed below. (If $\mathbf{L} \neq \mathbf{I}$ then the problem must first be transformed to *standard form*, see [105, Section 2.3].) The most commonly used algorithms are:

SVD:

A standard approach is to compute the SVD (or the generalized SVD (GSVD) when $\mathbf{L} \neq \mathbf{I}$, see [83, 100]) of $\mathbf{A}$, and form the regularized solution directly from the expression in Equation (2.8). This furthermore has the advantage that the SVD often gives a lot of insight into the numerical properties of the problem, see, e.g., [46].

Bidiagonalization:

The most efficient direct implementation of Tikhonov regularization is the bidiagonalization algorithm by Eldén [70] The algorithm consists of the following steps:

1. Compute the bidiagonalization of matrix $\mathbf{A}$.

$$\mathbf{A} = \mathbf{U} \begin{pmatrix} \mathbf{B} \\ \mathbf{0} \end{pmatrix} \mathbf{V}^T ,$$

   where

$$\mathbf{U} \in \mathbb{R}^{m\times m} , \quad \mathbf{B} \in \mathbb{R}^{n\times n} , \quad \mathbf{V} \in \mathbb{R}^{n\times n} ,$$

   $\mathbf{U}$ and $\mathbf{V}$ are orthogonal and $\mathbf{B}$ is upper bidiagonal.

2. Find $\mathbf{c} = \mathbf{U}^T\mathbf{b}$.
3. Solve

$$\min \left\| \begin{pmatrix} \mathbf{B} \\ \lambda \mathbf{I} \end{pmatrix} \mathbf{y} - \begin{pmatrix} \mathbf{c} \\ \mathbf{0} \end{pmatrix} \right\|_2 . \tag{2.19}$$

   This can be solved for $\mathbf{y}$ by eliminating the λ's under the main diagonal using $2n$ Givens rotations.

4. Compute the solution $\mathbf{x} = \mathbf{V}\mathbf{y}$.

For computational details see [70] and Appendix B.

This algorithm has the advantage that the most expensive step in the algorithm, the bidiagonalization of $\mathbf{A}$, is independent of the value of λ. This means that if λ is changed the new solution can be computed cheaply, which is essential when a parameter-choice method is used to find the optimal value of λ, see [70] and Chapter 4.

Normal equations:

1. Compute the lower triangular portion of $\mathbf{Q} = (\mathbf{A}^T\mathbf{A} + \lambda^2\mathbf{L}^T\mathbf{L})$.
2. Find $\mathbf{d} = \mathbf{A}^T\mathbf{b}$
3. Compute the Cholesky factorization $\mathbf{Q} = \mathbf{G}^T\mathbf{G}$.
4. Solve $\mathbf{G}^T\mathbf{y} = \mathbf{d}$ and $\mathbf{G}\mathbf{x} = \mathbf{y}$.

For further details see [83, Section 5.3]. In applications where $m \gg n$ it is important that $\mathbf{Q}$ can be set up without forming $\mathbf{A}$, since this reduces the amount of memory needed from mn to n^2 words. However, from step 1 we notice that $\mathbf{Q}$ depends on λ. Therefore the algorithm is only useful when the optimal value of λ is known in advance (or can be estimated in a very few steps). A more serious problem is that information may be lost in forming the normal equations for an ill-conditioned system, see [25, Section 2.1].

In connection with OLA methods, solving the normal equations is often prohibitively (and unnecessarily) expensive, since $\mathbf{Q}$ will have dimensions $m \times m$ and therefore step 1 involves $\mathcal{O}(m^3)$ operations.

In the table below we have listed the number of floating-point operations (flops) required by each of the three algorithms above when $\mathbf{A}$ is $m \times n$ and $m \geq n$ and $\mathbf{L} = \mathbf{I}$:

Algorithm	Flop Count
SVD	$2mn^2 + 11n^3$
Bidiagonalization	$2mn^2 + 2n^2$
Normal equations	$mn^2 + n^3/3$

We refer to Appendix B for a description of OLA implementations based on the three algorithms above and detailed flop counts.

Large-scale problems very often lead to sparse or structured (e.g., Toeplitz) matrices, and for many large-scale problems the use of algorithms based on standard implementations found in software libraries such as LAPACK (see [4]), may not be applicable because they require the entire matrix to be stored explicitly. One way to solve this problem is to use a so-called "out of core" version of the direct algorithm, which stores the matrix in secondary storage and copies blocks of the matrix in and out memory as they are needed – in other words to include another layer of the memory hierarchy. These methods, however, are usually difficult to program and the speed of the I/O-system may severely limit performance.

Another problem with direct algorithms that compute a factorization of the matrix is that they most often work by applying a series of transformations to the matrix, such that it eventually is on, e.g., banded or triangular form. This can be a problem since the transformations usually destroy any sparsity or structure in $\mathbf{A}$.

2.3 Iterative algorithms

For many large-scale problems iterative methods can usually exploit the sparsity or structure better that direct methods. Iterative methods are also competitive to direct methods when the matrix is dense and matrix-vector products can be computed efficiently such that the matrix itself is never actually computed and stored. The main reason is that most iterative methods are based on iteration schemes accessing the matrix only through matrix-vector multiplications with $\mathbf{A}$ or $\mathbf{A}^T$. This can be exploited in different ways depending on the application:

Structure and sparsity: The fact that iterative algorithms only access $\mathbf{A}$ through matrix multiplication ensures that the sparsity or structure of $\mathbf{A}$ is never altered. When computing the matrix-vector multiplication with $\mathbf{A}$ or $\mathbf{A}^T$ it is possible to exploit this fact. Examples:

1. If $\mathbf{A}$ is sparse then a matrix-vector multiplication requires $2N_z$ floating-point operations, where N_z is the number of non-zeroes in $\mathbf{A}$, and N_z words of storage are needed to store $\mathbf{A}$.
2. If $\mathbf{A}$ is an $n \times n$ Toeplitz matrix then a matrix-vector multiplication requires $15\,p \log_2 p$ floating point operations, where p is the smallest power of 2 larger than n, and $2n$ words of storage are needed to store $\mathbf{A}$. See [71, 99].
3. If $\mathbf{A} = \mathbf{B} \otimes \mathbf{C}$ is an $n^2 \times n^2$ Kronecker product (where $\mathbf{B}$ and $\mathbf{C}$ are $n \times n$) then a matrix-vector multiplication requires $4n^3$ (rather than $2n^4$ had $\mathbf{A}$ been formed explicitly) floating-point operations, and $2n^2$ words of storage are needed to store $\mathbf{B}$ and $\mathbf{C}$; see, e.g., [73, 25], Appendix C and Section 3.3.

Computation of A "on the fly": In some applications it can be more efficient to generate the entries (or blocks of entries) in $\mathbf{A}$ "on the fly" in each matrix-vector multiplication rather than to store the full matrix explicitly in memory. Using this technique the computation $\mathbf{y} \leftarrow \mathbf{A}\mathbf{x}$ could look like this:

```
for i = 1 to M do
    y_i = 0
    for j = 1 to N do
        compute a_ij
        y_i = y_i + a_ij x_j
    enddo
enddo
```

The extra work involved in computing $\mathbf{A}$ may be more than compensated for if $\mathbf{A}$ can be generated using data residing in main memory or maybe even in the cache, see, e.g., [200].

Indirect formulation: If the matrix $\mathbf{A}$ represent some continuous operator, it may be possible to compute the result of the operator acting on a vector $\mathbf{x}$ without using the matrix representation of $\mathbf{A}$; see, e.g., [205].

2.3.1 Lanczos and conjugate-gradient based methods

The issues raised above are not in any way particular to the solution of ill-posed problems, but apply quite generally to applications where iterative algorithms are used for solving problems involving large systems of linear equations or linear least-squares problems. But there is an aspect of iterative algorithms based on the well-known conjugate-gradient or the mathematically equivalent Lanczos algorithm, which is very interesting in connection with the solution of discrete ill-posed problems: These algorithms have an intrinsic regularizing effect when applied directly to the discrete ill-posed problem in Equation (2.3). If an iterative method generates approximations $\mathbf{x}^{(k)}, \quad k = 1, 2, \ldots$ it is often the case that $\mathbf{x}^{(k)}$ initially picks up the SVD-components corresponding to the largest singular values in the solution given by

Equation (2.7). Hence by stopping the iteration long before $\mathbf{x}^{(k)}$ converges to the formal solution of Equation (2.3), which is usually dominated by noise, we get a regularized solution. The iteration number k plays the role of a regularization parameter, and the computation of $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots$ corresponds to sweeping through a range of different regularization parameters; see e.g. the discussion in Appendix B. Methods based on the *Lanczos algorithm*, including the mathematically equivalent *conjugate gradient* method, have the abovementioned properties and many iterative regularization methods based on the Lanczos algorithm have been suggested (e.g., [144, 24, 202, 99, 103, 85, 32]). For a general description of the Lanczos algorithm, its many applications and the connection to the conjugate gradient method see the books by Parlett [152], Cullum & Willoughby [56] and Golub and Van Loan [83, Chapter 9–10].

In its original form the Lanczos algorithm was developed for solving the symmetric eigenvalue problem $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$, and has proved to be especially efficient when a few eigenvalues and eigenvectors of a large sparse matrix are needed. Variants of the algorithm can also be used for computing the SVD and solving linear equations or least-squares problems, see [57, 151, 146]. The most important variant for solving discrete ill-posed problems is the *Lanczos bidiagonalization*, which (in exact arithmetic) computes two sets of orthogonal Lanczos vectors $\{\mathbf{u}_i\}$ and $\{\mathbf{v}_i\}$ such that after k steps and with

$$\mathbf{U}_k = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{k+1}), \qquad \mathbf{V}_k = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k),$$

we obtain a $(k+1) \times k$ bidiagonal matrix $\mathbf{B}_k$ satisfying

$$\mathbf{A} = \mathbf{U}_k \mathbf{B}_k \mathbf{V}_k^T \ . \tag{2.20}$$

The largest singular values of $\mathbf{B}_k$ are good approximations to the largest singular values of $\mathbf{A}$, and this property we can exploit in different ways in trying to solve the original ill-posed problem:

Approximation to the SVD of A: Given the bidiagonalization in Equation (2.20) we can compute an approximation to the SVD of $\mathbf{A}$ by simply computing the SVD of $\mathbf{B}_k$

$$\mathbf{B}_k = \mathbf{P}_k \mathbf{\Sigma}_k \mathbf{Q}_k^T \ .$$

This can be computed cheaply because $\mathbf{B}_k$ is bidiagonal. Now approximations to the singular values of $\mathbf{A}$ are given in $\mathbf{\Sigma}_k$ and moreover we have that the columns of $\mathbf{U}_k\mathbf{P}_k$ and $\mathbf{V}_k\mathbf{Q}_k$ are approximations to the left and right singular vectors of $\mathbf{A}$ respectively. Using these quantities we could, e.g., compute an approximation to the truncated SVD solution in Equation (2.9); see Appendix E.

Solution to the regularized least-squares problem: Alternatively we can compute an approximate Tikhonov solution using the bidiagonalization algorithm described in Section 2.2 with the full bidiagonalization replaced by the Lanczos bidiagonalization from Equation (2.20), i.e., Equation (2.19) becomes

$$\min \left\| \begin{pmatrix} \mathbf{B}_k \\ \lambda\mathbf{I} \end{pmatrix} \mathbf{y}^{(k)} - \begin{pmatrix} \mathbf{U}_k^T\mathbf{b} \\ \mathbf{0} \end{pmatrix} \right\|_2 ,$$

from which we get the kth solution as $\mathbf{x}^{(k)} = \mathbf{V}_k\mathbf{y}^{(k)}$.

In finite precision this procedure is made more difficult due to loss of orthogonality among the Lanczos vectors, i.e., the columns of $\mathbf{U}_k$ and $\mathbf{V}_k$. Some of these problems can be "cured" (as I will briefly discuss below), and since the Lanczos algorithm and equivalently the conjugate gradient algorithm *do* have very attractive convergence properties their behavior in finite precision has been studied intensely in many different contexts; see e.g., [145, 56, 94, 26]. These effects are also discussed in detail in Appendices A and E.

The effects of finite-precision arithmetic in the Lanczos process is of most concern when it is used for computing singular values or when solving least-squares problems with multiple right-hand sides, such as in the OLA algorithm described in Chapter 4 and Appendix B. The most obvious (and most expensive) solution is to enforce orthogonality among the Lanczos vectors by re-orthogonalizing them in each iteration. This can be done by either QR factorization or by use of the modified Gram-Schmidt (MGS) algorithm. This approach is very efficient but unfortunately also very expensive. If k iterations are performed, the cost of full reorthogonalization is $\mathcal{O}(4k^2(m+n))$ floating-point operations and moreover all Lanczos vectors have to be saved which makes the method very expensive for large problems, when many iterations are required to obtain a satisfactory solution. This work can be reduced significantly by using *partial reorthogonalization* by Simon [182, 183] or *selective reorthogonalization* by Parlett and Scott [155], which have been used by a number of authors, see [22, 71] and have proven to be among the most efficient sparse eigenvalue algorithms. These algorithms were developed for the symmetric (classical) Lanczos process, and do not take the special structure of the Lanczos bidiagonalization into account. In Appendix A we present such an algorithm, and experiments performed in Matlab with a variety of matrices from the Harwell-Boeing collection show that the number of floating-point operation used by this is reduced by a factor of 3-4 compared with the original partial reorthogonalization algorithm, when used for sparse SVD calculations.

2.3.2 Parameter-choice methods for iterative regularization

The choice of regularization parameters is a challenging point in turning iterative methods into robust and generally applicable methods for solving large-scale ill-posed problem. Most of the heuristics such as Morozov's discrepancy principle, the Gfrerer/Raus-method, generalized cross-validation and the L-curve criterion, which are commonly used for choosing the regularization parameter, can be computed in a straightforward fashion when direct SVD-based algorithms are being used, see [105], but unfortunately they are not easy to implement when using iterative algorithms. However, since the computational core of the abovementioned heuristics can all be expressed as a minimization problem involving matrix moments, it is possible to use approximation techniques based on Lanczos bidiagonalization and the theory of Gauss quadrature, see [85] and references therein. These can be used to compute rigorous bounds on the parameters produced by the various heuristics in combination with a direct method. We believe that this kind of algorithms is a step towards combining the efficiency of iterative regularization with the reliable determination of the regularization parameters known from direct methods. These issues are discussed in more detail in Chapters 4 and 5.

2.3.3 Many right-hand sides

Little attention has been paid to developing efficient iterative algorithms for solving linear least-squares with many right-hand sides, which arise in, e.g., the SOLA inversion method.

Traditional sparse least-squares solvers such as LSQR or CGLS are inefficient when the number of right-hand sides is large, since their numerical stability hinges on the use of the right-hand side as the starting vector for the Lanczos process, and must therefore solve the systems independently one at a time.

One of the few algorithms described in the literature providing a suitable compromise, is the hybrid algorithm by Björck [24] based on Lanczos bidiagonalization. A variant of this algorithm, which avoid computing the SVD, but solves the regularized least-squares problem directly as described in the previous section, has been used for the investigations reported in Appendix C. This algorithm, which is described in detail in Chapter 4, still suffers from a problem: When multiple right-hand sides are present, full reorthogonalization must be used to preserve numerical stability and with the increasing size of the helioseismic datasets this will soon become too expensive. To solve this problem we intend to investigate two alternative Krylov subspace algorithms:

1. Lanczos bidiagonalization with partial reorthogonalization. One option is to use Björcks algorithm combined with the partial reorthogonalization scheme described in Appendix A to reduce the cost of keeping the generated Krylov basis orthogonal. An advantage of this approach is the possibility to combine it with the parameter choice methods mentioned in Section 2.3.2.

2. Galerkin-projection algorithm. One way to avoid reorthogonalization altogether is to use a generalization of a method based on an extension of the conjugate-gradient algorithm, developed for solving square linear systems with many right-hand, see [36]. Briefly, the method generates a Krylov subspace spanned by a set of direction-vectors obtained by solving one of the systems, called the *seed system*, by the CGLS algorithm. Then it projects the residuals of the other systems orthogonally onto the generated Krylov subspace to get the approximate solutions. The projections can be computed cheaply and involve only a single inner product and a two vector updates per right-hand-side per iteration, which in addition can be executed completely in parallel. With respect to solving ill-posed problems, it is interesting that a generalization of the analysis by Chan and Wan [36] shows that the approximate solutions to the non-seed systems are spanned by the singular vectors corresponding to the largest singular values. Hence the method has an intrinsic regularizing effect, which is confirmed by our preliminary investigations. Much work is still needed to turn this, potentially very efficient, approach into a robust algorithm. These issues are discussed in more detail in Chapter 4 and Appendix A.

2.4 Implementation on high-performance computers

In this section we will give some general remarks about the state-of-the-art of numerical software on high-performance computers before we proceed to discuss how the direct and iterative algorithms for solving discrete ill-posed problems can be implemented efficiently. Results for parallel implementation of three different algorithms for 2-dimensional helioseismic inversion based on

- Lanczos bidiagonalization,
- the LSQR algorithm and
- Eldén's bidiagonalization algorithm from Section 2.2

are discussed in Chapter 4 and in Appendices C and E.

2.4.1 Designing numerical software for high-performance computers

The design and implementation of algorithms for high-performance computers will always be influenced by machine-specific parameters such as the relative speed of different instructions, the number of registers, memory latency and bandwidth, cache size, communication bandwidth and latency; squeezing the maximal performance out of any given computer requires that one pays attention to a lot of tedious details – often a non-trivial and very time-consuming task. The three main issues to address are the efficient use of

1. pipelined floating-point units (vector- and superscalar processors),
2. memory hierarchy (registers, cache, main memory), and
3. parallelism (multiple processors).

Techniques for dealing with the first two problems, sometimes called *serial code optimization*, are well-known and involve:

- the use of improved software technology: advanced optimizing compilers, improved programming environments including debugging, profiling and hardware monitor tools.
- A large number of software libraries have been written over the years giving the user access to a large set of robust computational kernels tuned to achieve maximum performance on a given machine. A good example is the Basic Linear Algebra Subroutines or BLAS, which is a set of subroutines that implement the most frequently used primitives in linear algebra, such as vector updates $\mathbf{y} \leftarrow \alpha\mathbf{x}+\mathbf{y}$ and inner products $\alpha \leftarrow \mathbf{x}^T\mathbf{y}$. The BLAS has become a de facto standard and every modern supercomputer comes with a set of highly tuned BLAS routines, see [133, 134, 64, 65]. Apart from providing a set of optimized kernels, the BLAS has also helped in standardizing numerical software and making it more modular. In particular this has made it easier to obtain a reasonable performance when porting a numerical application to a new platform, without the need for hand-tuning large portions of the code. In fact, numerical software libraries now exist on most high-performance platforms for the most common numerical techniques used in scientific computing, such as numerical linear algebra, methods for solving partial and ordinary differential equations, linear and non-linear optimization, quadrature, fast Fourier transforms, see [63, 4, 140, 117, 41].

Pipelined floating-point processors

Many large scientific applications are very rich in linear algebra operations such as inner products or SAXPY's on large matrices and vectors. These operations form a very regular sequence of floating point operations. This regularity has been exploited in so-called pipelined floating-point units which can be used to provide high-level operations that operate on entire *vectors*, i.e., arrays of floating-point numbers, doing, e.g., component-wise addition or multiplication of two vectors. Originally developed for large serial supercomputer, *vector processors* such as the machines from CRAY, pipelined floating-point units are now found in all superscalar microprocessors such as ALPHA, MIPS, Pentium, IBM PowerPC/Power2 and SPARC found in large PC's and workstations today. Moreover, most processors of today contain multiple functional units typically allowing the processor to execute several vector instructions, integer arithmetic and load/store operations in parallel – another technology inherited from the traditional supercomputer.

Maybe the single most important factor in the large increase in the performance of modern superscalar microprocessers is the developments in compiler technology that have occurred in parallel with the development of the superscalar processors. It is crucial that instructions be scheduled carefully in order to exploit the *instruction-level parallelism* introduced by pipelining and multiple functional units.

A good example of the historical influence of vector-processors on numerical software is seen in the development and widespread use of the BLAS, which provides a set of computational primitives that relate closely to the hardware primitives provided by vector processors. By using the BLAS becomes relatively easy to formulate algorithms that "vectorize by definition" and therefore achieve a good performance across a wide range of different machines. The development and wide-spread use of, e.g., the LINPACK library has fully demonstrated the usefulness of this approach (cf. [63, 67]). With modern compiler technology, however, it is rarely necessary to use hand-tuned BLAS routines written in assembler-language to achieve good performance.

Memory hierarchies

In most of today's computers the rate at which operands have to be fed to the processor significantly exceeds that at which they can be transported to and from the main memory. This has lead to a widespread use of multilayered memory hierarchies, which usually consist of registers, cache, main memory and disk storage. The intermediate levels are designed to disguise the speed gap by inserting small and fast (and expensive) buffer-memories between the processor(s) and the lower levels in the hierarchy. If computations can be arranged such that the processor is be kept busy operating on data in the highest (fastest) level while the operands needed in the future are being loaded from one of the lower levels, the speed gap can almost hidden. Thus for a hierarchical memory to provide any gains in computational performance a large amount of *data reuse* is crucial. This introduces a new factor to consider in the analysis of algorithm complexity, namely the cost of moving data between the different levels of the memory hierarchy. Therefore the number of data transports between layers in the memory hierarchy, e.g., caused by *cache misses*, is in many cases *the* most important performance measure to consider when designing a new algorithm for or implementing a well known algorithm on a new machine [79, 17]. Table 2.1, which is based on figure 1.16 in [109] Section 1.7, lists typical values of the key figures for the memory hierarchy of a large workstation or a small server in 1998. The development of memory hierarchies has been

Table 2.1: Levels of the memory hierarchy.

Level	Registers	Cache	Main memory	Disk storage
Typical Size	< 1 KB	< 4 MB	< 4 GB	> 10 GB
Implementation Technology	Custom CMOS	On- or off-chip CMOS SRAM	CMOS DRAM	Magnetic disk
Access time (cycles)	< 1	1–15	40–100	$5 \cdot 10^5$–$5 \cdot 10^6$
Bandwidth (MB/sec)	4000-32,000	800–5000	400–2000	4–32
Managed by	Compiler	Hardware	Operating system	Operating system

largely technology driven – motivated by the high cost of fast memory – and the impact on

Table 2.2: Memory references and operation count for typical operations from the BLAS.

Level	Operation	Definition	Floating point operations	Memory references	Ratio
1	SAXPY	$y_i = \alpha x_i + y_i, i = 1, \ldots, n$	$2n$	$3n + 1$	$2/3$
2	Matrix-vector mult	$y_i = \sum_{j=1}^{n} A_{ij} x_j + y_i$	$2n^2$	$n^2 + 3n$	2
3	Matrix-matrix mult	$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj} + C_{ij}$	$2n^3$	$4n^2$	$n/2$

the design of algorithms has been quite radical. Much effort has been put into the development of algorithms and compiler techniques to improve the locality of reference in programs. In the most important technique called *blocking* (e.g. [79, 4]), computations are restructured such that memory accesses are clustered both in space and time, and this is typically achieved by reformulating the algorithm to make it operate on blocks of data at a time. A good example is matrix multiplication, which is blocked by transforming the computation into a version that operates on submatrices; see [83], Section 1.4.9. The success of this approach is explained by Table 2.2: In an unblocked (point) matrix-matrix multiply $C \leftarrow A * B + C$ the inner-most loop typically updates a column of C using a SAXPY operation $y_i = \alpha x_i + y_i$. This implementation is very efficient on a vector-processor, because it ensures that the pipelined floating-point units are used efficiently operations are performed on long vectors. Moreover, the operation $y_i = \alpha x_i + y_i$ can be performed in a single clock-cycle using the technique called *chaining*, and it ensures that bank conflicts are unlikely to occur; see e.g., [79, 67]. A similar technique is used in modern superscalar processors, where a result can be *forwarded* from the multiplication to the addition unit to implement a floating-point *multiply-and-add* instruction that can calculate $y_i = \alpha x_i + y_i$ in a single clock-cycle.

On a machine with a memory hierarchy, however, a SAXPY-based implementation is rather inefficient due to the low ratio 2/3 of floating-point operations to the number of memory references of the SAXPY operation as shown in Table 2.2. By formulating the matrix multiply in terms of multiplications of $n \times n$ submatrices, this ratio can be increased by choosing the *block size* n as high as possible subject to the constraint that three submatrices C_{ij}, A_{ik} and B_{kj} should fit in the cache. Here we are essentially exploiting the fact that a matrix multiply performs $\mathcal{O}(n^3)$ operations on $\mathcal{O}(n^2)$ data – the "surface to volume effect" for the ratio of operations to data movements. As an example, a machine with a 1Mb cache could hold $2^20/8$ double precision floating-point numbers in the cache; to fit C_{ij}, A_{ik} and B_{kj} in the cache (assuming they are square) n should be smaller than 209. This is obviously a simplified description; in practice the situation is complicated by the cache being divided in to *cache lines* (typically 128 bytes) and that different schemes for mapping a given memory address into the cache are in use. Often it is useful to construct a model these various effects (see e.g., Appendix E) to analyze these effects and to understand how to improve the performance of a given implementation; this is sometimes called *performance modelling*. For further details on memory hierarchy architecture and performance modelling we refer to [109, 79, 199]. Again the development of the BLAS over the years illustrates the influence of machine architecture. The Level 1 BLAS was introduced 3 years after the introduction of the CRAY-1 in 1976. In 1986 the BLAS-2 was proposed because it had become necessary to optimize at the matrix-vector level to save memory bandwidth that was becoming a scarce resource with machines likes the CRAY-2 (the CRAY-2 had more functional units than the CRAY-1 but only the same number of ports to memory). At the same time vector-machines with cache memory such as the IBM 3090VF came into use, and with this change in machine architecture created the need for a Level 3 BLAS, which was proposed shortly after in 1987. The level 3 BLAS is a collection of linear algebra subroutines that perform $\mathcal{O}(n^3)$ operations on $\mathcal{O}(n^2)$ data; see [65, 123]. The algorithms implemented in the LAPACK library, which is used on most serial and shared memory multiprocessor machines today for computations involving dense matrices, use block-algorithms written in terms of calls to the level 3 BLAS subroutines to achieve good performance (e.g. [4, 5]).

Parallel computers

The architectural developments in serial computers mentioned in the previous section have not changed the basic structure or semantics of a program. Moreover most of the chances have been made more or less transparent to the users by the developments in compiler technology, and computational primitives (such as the BLAS routines) that hide the architectural dependences in low-level software libraries. As a contrast to this users of todays parallel computers need to be aware of the architectural details of their particular machine from the moment they start to develop a new algorithm, if they want to achieve good performance. Taking advantage of parallelism often requires the development of new algorithms and cannot be done entirely by providing new compilers and new software libraries. This is not an effect of parallel computers being a technology in its infancy (parallel computers have existed as long as any of the techniques mentioned earlier in this section). It is rather a consequence of the fact that the introduction of parallel computing represents a much more fundamental change of paradigm in the way we describe, develop and reason about programs. Not only is it difficult to achieve good performance on a parallel computer, but simply ensuring program correctness is a vastly more complex job for both the programmer and the compiler than it is on a serial machine. This is the primary reason why software for parallel supercomputers has not yet reached the same level of maturity that we have seen for e.g., serial supercomputers [139, 109, 199]. Having said this, it is also true that recently certain standards are beginning to emerge in different areas of parallel scientific computing. The advent of common standards for message-passing software like PVM and MPI and the general acceptance of High Performance Fortran (HPF) as a standard for data-parallel programming is a sign that the scientific computing community is moving towards a consensus about the software for parallel computers. In addition, much work has been done in recent years to produce a set of basic computational primitives for matrix computations, similar BLAS and LAPACK, but for distributed memory parallel computers. The results are the Parallel BLAS (PBLAS), Basic Linear Algebra Communication Subroutines (BLACS) and ScaLAPACK [42, 68, 41], which together provide part of the functionality of LAPACK on parallel architectures. In addition, a number of other libraries have appeared, mainly for solving sparse systems of linear equations, which typically arise in the solution of partial differential equations, e.g., in large-scale computational fluid dynamics simulations (e.g., [30]). Examples include Aztec [115]

BlockSolve95 [119] PETSc [12] and P_SPARSLIB [170] (the webpage [69] maintained by Jack Dongarra at the University of Tennesee, provides an excellent overview of freely available software for linear algebra on the web).

2.4.2 Implementing regularization algorithms

Direct algorithms

The direct regularization algorithms described in Section 2.2, are all based on matrix factorizations (SVD, bidiagonalization and Cholesky) of either the kernel matrix or some matrix computed from the kernel matrix. These factorizations, which are some of the standard algorithms in numerical linear algebra, can be implemented efficiently on practically all of todays parallel high-performance computers; see [40, 59, 66]. This is done using so-called *partitioned* or *block* algorithms, where (as described above) the computations of the original sequential algorithm are restructured such as to make it operate on blocks of data at a time. These block operations can be formulated in terms of Level-3 BLAS (matrix-matrix) operations where the amount of reuse of data in the cache is high. Optimized and even parallelized subroutines can be found in software libraries on most high-performance computers.

Iterative algorithms

Direct methods, which are mainly suited for solving small to medium-sized problems, where $\mathbf{A}$ is dense and small enough to be stored in main memory. However, as described in Section 2.3 large-scale problems often lead to sparse or structured matrices that can be treated more efficiently using iterative algorithms. Iterative algorithms are in some sense serial by nature, since the computations in the kth iteration usually involve recursions referring to the values of variables computed in the previous steps. This can be seen in, e.g., the Lanczos bidiagonalization algorithms shown in Section 4.4. Therefore it is difficult to parallelize across steps, but instead an iterative algorithm must be implemented by parallelizing the operations within each iteration. These operations are typically

Inner products:	$\alpha \leftarrow \mathbf{x}^T\mathbf{y}$
Vector update:	$\mathbf{y} \leftarrow \alpha\mathbf{x} + \mathbf{y}$
Matrix-vector multiplications:	$\mathbf{y} \leftarrow \mathbf{A}\mathbf{x}$
	$\mathbf{y} \leftarrow \mathbf{A}^T\mathbf{x}$
Preconditioning:	$\texttt{solve}\ \mathbf{M}\mathbf{x} = \mathbf{y}$

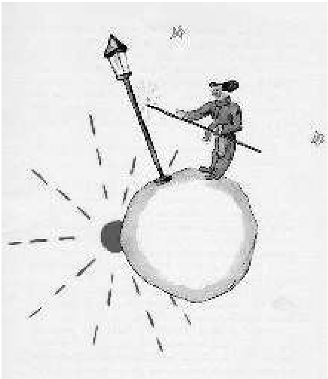
The vector updates are trivial to parallelize. If, say, the vectors are distributed over the processors the updates can simply be done locally, but the global reduction involved in computing the inner products and broadcasting the result involves communication. Much effort has been put into the reordering of the operations in conjugate gradient and Lanczos type algorithms to improve parallel and cache performance. An important technique is to reorder the steps in the iteration such that communication in the accumulation of the inner products can be overlapped with local computations in the matrix-vector multiplication or local vector updates, see [188, 9, 199, 59] and Appendix E. This is sometimes tricky to do without changing the numerical properties of the algorithm, see the discussion in [59, Section 8.2]. Parallelizing the preconditioner is also highly application dependent. General preconditioners such as Jacobi, Gauss-Seidel or SSOR can be parallelized using the coloring schemes described in, e.g. [199, 75]. Preconditioners based on incomplete factorizations (cf. [169]), on the other hand, often involve the solution of triangular systems of equations which are not easy to parallelize.

The main computational effort usually lies in computing the matrix-vector products with $\mathbf{A}$ and $\mathbf{A}^T$. How these operations should be implemented in parallel depends entirely on the properties (e.g. structure or sparsity) of matrix $\mathbf{A}$. See, e.g., [200] and Appendices E and C.

When implementing the LSQR algorithm on a parallel computer, it is necessary to compute matrix-vector multiplications with both $\mathbf{A}$ and $\mathbf{A}^T$. In Appendix E we investigate how the Lanczos bidiagonalization can be reordered such that the multiplications with $\mathbf{A}$ and $\mathbf{A}^T$ can be overlapped. If $\mathbf{A}$ is generated "on-the-fly" this means that $\mathbf{A}$ has to be generated only once, or, alternatively, if $\mathbf{A}$ is stored in memory then the number of cache-misses is halved. The overlapping introduces a number of extra synchronization points in the algorithm which lowers the achieved speedup on the machine, but at the same time reduces the execution time almost by a factor of 2 for large problems due to the reduction in the number of cache-misses. This is a demonstration of the fact that speedup is not necessarily a good performance measure for assessing the performance of a parallel algorithm.

Chapter 3

Helioseismic rotational inversion



In this chapter we give a short introduction to the area of helioseismology, where the global normal modes of oscillation observed on the solar surface are used to study the solar interior in the same way as seismic waves are used to probe the conditions deep under ground on Earth [191]. We will outline the basic theory of stellar oscillations and describe how the properties of observed oscillation modes can be used to infer the solar rotation rate as a function of latitude and distance from the center of the Sun in a substantial fraction of the solar interior. This process known as *helioseismic rotational inversion* gives rise to a linear inverse problem and is treated using the techniques described in the previous chapter. Here we describe the inverse problem and show how the algebraic form of the kernels in the Fredholm integral equation involved can be used to reduce the computational complexity of the solution process. We will only touch upon a few of the topics most relevant to the work presented in this thesis. However, a number of reviews on helioseismology can be found in the literature. For the non-specialist, we would like to recommend the Science special issue on helioseismology [179], which features a series of reviews of many areas of helioseismology. More in-depth treatments of the subject can be found in, e.g., Cox et al. [53], Gough [88] and Roca Cortes [165].

3.1 Introduction to Helioseismology

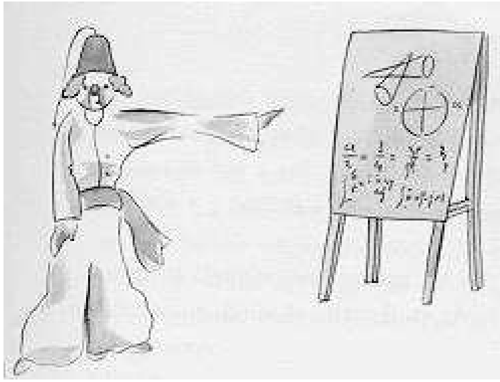
The study of stellar structure and evolution has always played a central part in astrophysics and has been fundamental to our understanding of the structure and history of the galaxies and hence the universe. Perhaps the most well-known example is the discovery that synthesis of all chemical elements (apart from hydrogen and helium), including the chemical elements that planetary systems and eventually humans consist of, has taken place in the interior of stars. A remarkable fact is that most of this knowledge has been obtained from theoretical modeling. The reason for this success is the relative simplicity of stellar matter. A simple stellar model, which is in accord with many observed properties of stars, can in fact be derived from very fundamental physical principles. Although this type of modeling has brought us a long way (e.g. [43]) in understanding the overall nature of the processes governing the life of a star, there are still many open questions, which cannot be answered without observational evidence.

This poses a serious challenge since we cannot (for obvious reasons) probe the interior of stars, not even the Sun, directly. This is unlike (geo-) seismology where predictions (at least in principle) can be checked by simply drilling a hole to measure the "true solution" of the inverse problem. One way to probe indirectly the interior of the Sun is by measuring *solar oscillations*. Fortunately (if you are a helioseismologist), and somewhat surprisingly (unless you are a helioseismologist), the structure of the interior of stars is known with much better accuracy than that of the Earth, where the complex stratification of different materials with poorly determined mechanical properties introduces large uncertainties and non-linearities in the models used to formulate inverse problems [150, 192]. Even when the inversion can be formulated as a linear problem such as in Chapter 2, the norm of $\delta\mathbf{A}$, as well as correlations between poorly determined model parameters, can be non-trivial and has to be taken into account. In contrast, the inverse problems for the Sun, which can be modeled as a relatively uniform and almost spherically symmetrical ball of ionized gas with a well-known equation of state, can be treated using linear perturbation theory around an equilibrium state.

One of the goals of helioseismology, which are discussed, e.g., by Gough et al. [90], is to study the Sun's differential rotation. From observations of sunspots on the solar surface, it has been known for centuries that the Sun rotates differentially with a rotation period of about 25 days at the equator increasing smoothly to about one month at the poles. With the helioseismic datasets from experiments such as MDI and GONG, the solar rotation rate has been determined not only at the surface but also in a substantial fraction of the interior (see Schou et al. [175] and Thompson et al. [194]). This is important for studying the mechanisms that build and rebuild the magnetic field during the 11-year period of the solar sunspot cycle. These magnetohydrodynamical mechanisms are not understood but are believed to depend strongly on the rotation profile in the convection zone, where rotation has been found to be almost constant on radii, and in the shear layer called the tachocline immediately beneath the convection zone, which forms a transition layer to the deep interior where rotation seems to almost uniform (cf. [39]).

Young stars are often observed to rotate rapidly, and another matter of interest is the manner in which the Sun has been spun down to its current slow rotation rate. This is believed to have occurred through a gradual loss of angular momentum of the Sun's outer parts via magnetic coupling to the solar wind. It is uncertain, however, whether the spin-down has affected only the outer parts and whether the Sun could possibly have maintained a rapidly rotating core (cf. Chaplin et al. [37])

3.1.1 Theory of solar oscillations



Sea-nymphs hourly ring his knell
Hark! now I hear them,–Ding-dong, bell.

— W. Shakespeare, from "The Tempest"

Stars with periodic variations in luminosity due to oscillations have been known for centuries. One important example is the so-called Cepheids, observations of which are used to measure indirectly the distances to other galaxies. By using an approximate relation between the absolute luminosity L and the period of oscillation of such stars known from stellar models, the distance d can be calculated from the relation $l = L/4\pi d^2$ by measuring the relative luminosity l. The reason why we can observe the oscillations of the Cepheids as variations in their apparent luminosity is that the pulsations are spherically symmetric oscillations of large amplitude where the whole star contracts and expands. It is believed that many stars pulsate in a much more complicated way where many modes of oscillations including overtones and non-radial modes are continuously excited by processes in the stellar interior. One problem in observing these oscillations is that we cannot resolve the surface of stars other than the Sun, which means that we can only expect to observe a few non-radial modes. Still, a number of interesting questions can be answered using this, seemingly, sparse information, and *asteroseismology* has become very active area of research in recent years, see [29].

Unlike its distant relatives, our own star, the Sun, is close enough that hundreds of thousands of oscillation modes can observed on the solar surface. These oscillations are excited by stochastic processes driven by the turbulent motion in the convection zone, which constitutes roughly the outer 30% of the Sun by radius. These stochastically driven oscillations keep the Sun ringing in a large number of normal modes roughly in the frequency range of 1000–5000 μHz, which corresponds to a period around 5 minutes. Since the amplitudes are very small (< 15 cm/s), the oscillations can be modelled as perturbations around a time-independent equilibrium state. Combined with the assumption of the Sun being spherically symmetric this leads to a significant simplification: When written in spherical polar coordinates (r, θ, ϕ), where r is radial distance from the center of the Sun, θ is co-latitude and ϕ is longitude, the spatial variation of a mode separates in radius and (θ, ϕ). The angular dependence can be expressed as a spherical harmonic

$$Y_l^m(\theta, \phi) \equiv P_l^m(\cos\theta)\exp(im\phi) \ , \tag{3.1}$$

where P_l^m is an associated Legendre function normalized such that

$$\int_{-1}^{1} [P_l^m(u)]^2 \mathrm{d}u = 1 \ . \tag{3.2}$$

With this separation of variables, the general oscillation equations simplify to a set of (one-dimensional) ordinary differential equations that, given a model of the equilibrium state of the Sun, can be solved to obtain the radial and horizontal displacement eigenfunctions of the mode. These are often written as $\xi_{nl}(r)$ and $L^{-1}\eta_{nl}(r)$, where $L \equiv \sqrt{l(l+1)}$. The physical displacement at a given point can now be written

$$\begin{aligned}\delta\mathbf{r}(r,\theta,\phi,t) &= \sqrt{4\pi}Re\Big\{\Big[\xi_{nl}(r)Y_l^m(\theta,\phi)\mathbf{a}_r + \\ &\quad L^{-1}\eta_{nl}(r)\Big(\frac{\partial Y_l^m}{\partial\theta}\mathbf{a}_\theta + \frac{1}{\sin\theta}\frac{\partial Y_l^m}{\partial\phi}\mathbf{a}_\phi\Big)\Big]\exp(-i\omega_{nlm}t)\Big\} ,\end{aligned} \tag{3.3}$$

where $\mathbf{a}_r$, $\mathbf{a}_\theta$ and $\mathbf{a}_\phi$ are unit vectors in the r-, θ- and ϕ-directions. The behavior of a given mode is characterized by the integers l, m and n, where l is the degree of the spherical harmonic (the number of wavelengths around the solar circumference is roughly given by L); the azimuthal order m is the number of nodes along the equator and $-l \leq m \leq l$; n is the radial order, i.e. the number of nodes in radius of the displacement eigenfunction.

What is observed in practice is the line-of-sight component of the velocity on surface, which can be written as

$$v_{nlm}^{\mathrm{obs}}(\theta,\phi,t) = \sin\theta\,\cos\phi\,A_{nlm}(t)\cos[\omega_{nlm}t - m\phi - \delta_{nlm}(t)]P_l^m(\cos\theta) , \tag{3.4}$$

where the factor $\sin\theta\cos\phi$ comes from the projection onto the line of sight. Notice that Equation (3.4) also describes a wave traveling around the Sun with angular velocity m/ω_{nlm}. In a spherically symmetric star the frequency ω_{nlm} of the mode only depends on the wave numbers n and l. As we shall see below, deviations from spherical symmetry such as rotation, lifts this $(2l+1)$-fold degeneracy and causes the mode frequency, as well as the amplitude $A_{nlm}(t)$ and the phase $\delta_{nlm}(t)$, to depend on m. Due to the random way in which a given mode is repeatedly excited and damped, the phase $\delta_{nlm}(t)$ and amplitude $A_{nlm}(t)$ also generally depend on time. The modes are believed to be excited by the convective motions close to the solar surface, but are generally damped and decay exponentially, i.e., $A_{nlm}(t) \propto \exp(-\eta t)$. This means that modes do not show up as single line profiles in a power spectrum, but rather as Lorentzians

$$P(\omega) = \frac{P_0}{(\omega-\omega_0)^2+\eta^2}$$

with half width at half maximum given by the damping parameter η.

It turns out that there are three fundamentally different eigensolutions to the oscillation equations, namely the so-called f-, g- and p-modes. The p-modes, or *acoustic modes*, are essentially sound waves where the restoring force is pressure, and are those observed in the greatest number on the Sun. The p-modes can also be described as global standing waves created by the interference of propagating sound waves trapped in a cavity between the bottom of the photosphere (the "surface" of the Sun) and an inner so-called *turning point* r_t, as illustrated in Figure 3.1. The depth of the cavity depends on the degree and frequency of the mode though the relation

$$\frac{c(r_t)}{r_t} = \frac{\omega_{nlm}}{L}$$

where $c(r_t)$ is the local sound speed. As we shall see below, the frequency difference between two modes is a weighted average of the rotation in the region of the Sun where the corresponding difference between the two oscillations has an appreciable amplitude, e.g., the area between the two inner circles in figure 3.1. This is actually the idea expressed more formally in the OLA methods described in Section 2.1.3, where a linear combination of frequencies is

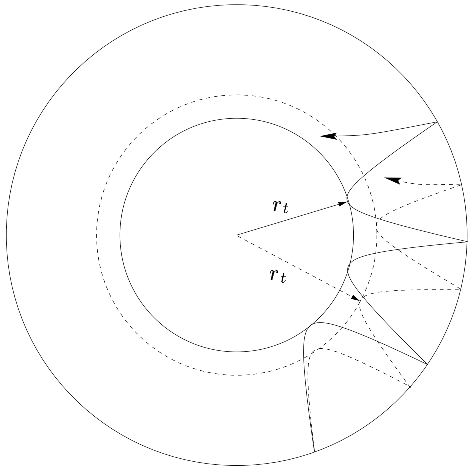


Figure 3.1: *A sketch of two p-modes with different lower turning point radii.*

sought such that corresponding superposition of oscillations is only sensitive to the properties in a small localized region of the Sun.

The modes are also restricted in latitude essentially to the region between co-latitudes $\theta_t \equiv \sin^{-1}|m|/L$ and $(\pi - \sin^{-1}|m|/L)$, such that so-called *sectoral modes* with $m = \pm l$ are concentrated in a narrow band around equator, while *tesseral modes* with $0 < m < l$ modes can extend almost all the way to the poles. One way to illustrate this is to plot for each mode the coordinates of the turning point in radius and latitude on a cut through the Sun. This is done in Figure 3.2 for all p-modes with $l < 200$, $1 \leq m \leq l$ and frequencies below approximately 5mHz (for higher frequencies the modes are no longer reflected at the surface but start to dissipate out into the solar atmosphere). As can be seen in the plot, most modes are trapped relatively close to the surface while only a few p-modes, especially those with low l, penetrate to the core. As illustrated in the discussion about modekernels in Section 3.2.1 below, making inferences about conditions in the deep interior is made even more difficult by the fact that the sensitivity of the modes frequencies to, e.g., the effect of rotation decreases strongly with depth. Another interesting feature of Figure 3.2 is the empty strip along the equator, which simply shows that even the sectoral modes do not live exactly on the equator, but have turning point turning point of $\theta_t \approx \sin^{-1}[l/(l+1/2)]$ which only approaches but never reaches $\pi/2$ as l increases (the beautiful patterns are a result of the discrete nature of the modeset – numerology in the Sun!)

The f-modes are *surface gravity modes*, where the restoring force is gravity. The f-modes, which can be depicted as the waves on the surface of a lake, are usually given a value of $n = 0$ and have properties similar to those of p-modes, as illustrated in Figure 3.4. As the name says, the f-modes are trapped close to the surface, but can be observed up to very high l and thus provides detailed information about, e.g., near-surface flows, see [121].

For g-modes or *internal gravity modes* the restoring force is also gravity, but unlike f- and p-modes they are trapped in the core of the Sun. The g-modes cannot propagate in the convectively unstable outer 30% of the Sun and are therefore expected to have very small amplitude at the surface. In fact, no certain detection of g-modes has been made to date, although this would greatly aid the study of the rotation and structure of the core.

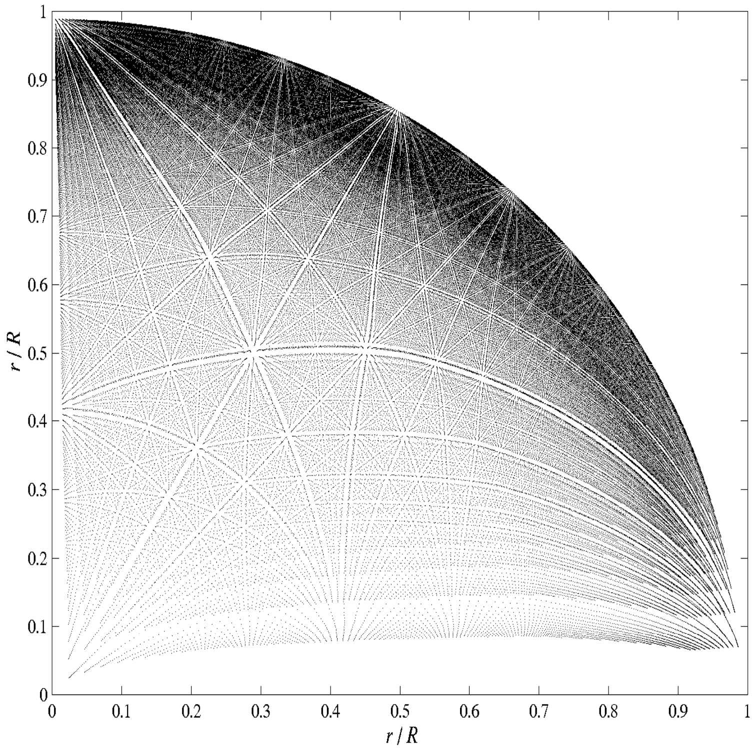
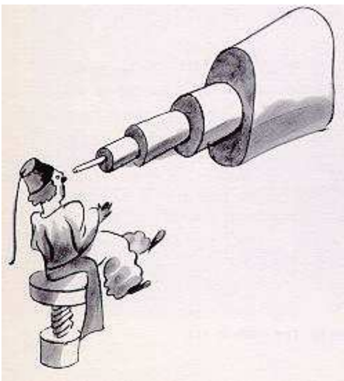


Figure 3.2: *Plot of* (r_t, θ_t) *for all p-modes with* $l \leq 200$ *and* $1 \leq m \leq l$ *and* $\omega_{nlm} < 5mHz$.

3.1.2 Observations



Drei Sonnen sah ich am Himmel stehn,
hab' lang und fest sie angesehn;
und sie auch standen da so stier,
als wollten sie nicht weg von mir.

— Wilhelm Müller, from "Winterreise"

Solar oscillations can be observed with basically two different techniques – either by measuring the temporal variations in luminosity across the solar disk, or by using Doppler observations of the velocity field on the solar surface. As an example, the MDI instrument on board the SOHO spacecraft uses a pair of Michelson interferometers to record images of the Doppler velocity, intensity and magnetic field on the solar disk by measuring the modulations of a solar spectral absorption line. The images are recorded at an interval of 1 minute by means of a 1024×1024 charge coupled device (CCD) (see [171, 122] for details). The observed Doppler velocity is the superposition of modes such as given in Equation (3.4), and can be written in

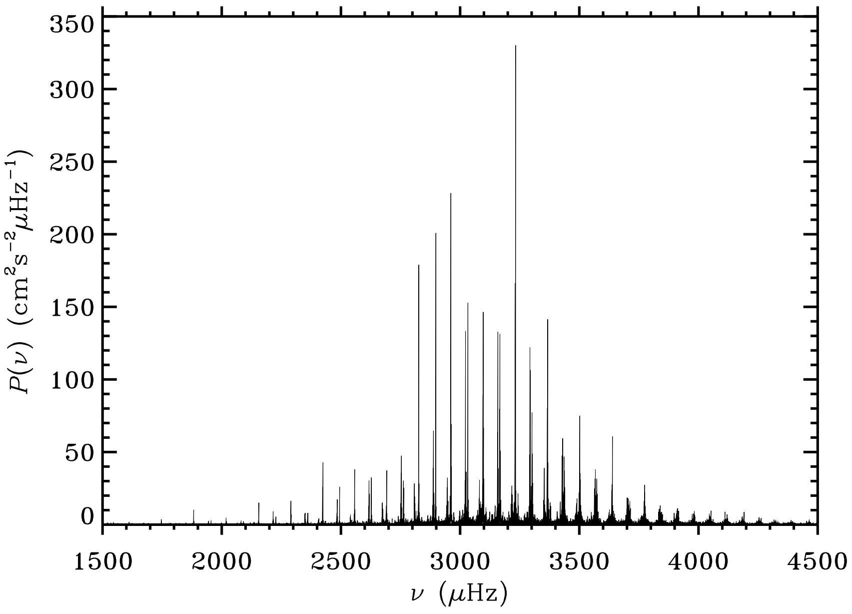


Figure 3.3: *Power spectrum of the solar oscillations in the 5-minute band, obtained from Doppler observations in light integrated over the disk of the Sun. The figure is reproduced from [44].*

the form

$$V_D(\theta, \phi, t) = \sin\theta\cos\phi \sum_{nlm} A_{nlm}(t)\cos[\omega_{nlm}t - m\phi - \delta_{nlm}(t)]P_l^m(\cos\theta) \ . \qquad (3.5)$$

The oscillations are superimposed on a background of noise generated by the turbulent convection in the surface layers, but since the noise is essentially random its importance decreases with increasingly long time-series, and the oscillations stand out clearly in a power-spectrum. In Figure 3.3 is shown the power-spectrum of solar p-modes measured by Doppler observations in light integrated over the disk of the Sun. The observations are from the BiSON network (cf. Elsworth et al. [72]).

To be useful for helioseismic analysis, the individual mode frequencies ω_{nlm} must somehow be determined from the velocity signal V_D. This turns out to be a highly complicated and computationally demanding process. Here we will just outline the main steps; for a general introduction to the observational aspects of helioseismology see Brown [28]; a recent review by Schou [177] gives an overview of the current state-of-the-art in determining mode parameters of medium- and high-degree oscillations.

Spatial separation: Initially modes with different angular dependence, i.e. different values of l and m, are separated. If simultaneous measurements of the velocity field on the entire solar surface were available, the separation could be done simply by noticing that the oscillations on the surface are described by spherical harmonics Y_l^m, and that the spherical harmonics form an orthogonal basis of functions on the unit sphere with respect to the inner product

$$< f, g > = \int_0^\pi \int_0^{2\pi} f^*(\theta,\phi)\, g(\theta,\phi)\, \sin\theta \,\mathrm{d}\theta \,\mathrm{d}\phi \ .$$

Thus, the components corresponding to $l = l_0$ and $m = m_0$ could be isolated by computing the inner product of V_D with $Y_{l_0}^{m_0}$. This can be done relatively efficiently, for all combinations of l_0 and m_0 simultaneously, by remapping the image to a regular grid in (θ, ϕ) and doing a Fourier transform in ϕ and a Legendre transform in θ, see, e.g., Brown [28]. If the entire solar surface could be observed, this would ideally result in a times series $y_{l_0 m_0}$, say, containing the combined amplitude signal for all modes with $l = l_0$, $m = m_0$ and different radial orders n. In practice, however, the observations cover less than one hemisphere with the result that modes with neighboring ls and ms can no longer be separated; instead, nearby modes with $|l - l_0| < 2$ and $|m - m_0| < 2$, say, start to "leak" into the time series for l_0 and m_0 with the result that spurious peaks begin to appear in the power spectrum. In this case the computed time series should rather be written as the sum

$$y_{l_0,m_0}^{\mathrm{obs}} = \sum_{l,m} C_{lm,l_0 m_0}\, y_{lm} \ ,$$

where C is the so-called *leakage matrix* and y_{lm} is the actual time-series of the modes on the Sun. Schou [173] analyzed a large number of instrumental (e.g. blurring due to imperfect point spread function) and other systematic errors that can introduce additional leakage, which further adds to the difficulty of the subsequent spectral analysis.

Spectral analysis / Peak bagging: Once the individual time series for the required range in l and m are known, the mode-frequencies and other parameters can be extracted by first doing a Fourier transform in time of each time-series and then determining the frequencies ω_{nlm} of the individual peaks in the resulting power-spectra; this procedure is used by the GONG project, see Hill et al. [111].

In contrast, the observation from MDI are generally not provided in the form of individual mode frequencies, but instead, for each *nl-multiplet*, their m-dependence is expressed in terms of *splitting coefficients* a_j:

$$\omega_{nlm}/2\pi = \nu_{nl} + \sum_{j=1}^{j_{\max}} a_j(n,l)\mathcal{P}_j^{(l)}(m) \ , \tag{3.6}$$

where $\mathcal{P}_j^{(l)}$ are orthogonal polynomials in m, of degree j. The expansion most commonly used was originally suggested by Ritzvoller and Laveley [164]; see also Schou, Christensen-Dalsgaard and Thompson [178]. The splitting coefficients are determined by fitting the expression in Equation (3.6) to the Fourier transformed time-series in the complex Fourier domain, see Schou [173, 177] and Rhodes et al. [162]. Due to leakage and the stochastic nature of the excitation mechanisms, the fitting is not just a simple non-linear least-squares fit, but involves a maximum-likelihood estimation based on statistical models that take into account correlations between the modes, the properties of the damping and excitations mechanisms (which, e.g., causes the peaks in the power-spectrum to be asymmetric, see Nigam et al. [142]), temporal leaks caused by gaps in the observations, as well as incorporating the leakage from neighboring ls via a pre-calculated leakage matrix. Computationally, the maximum-likelihood estimation is essentially an extremely demanding global optimization problem, which is made computationally tractable through a series of advanced heuristics providing, e.g., good starting guesses for the frequencies and for telling which modes can be safely ignored in the leakage calculations etc. These techniques were developed by Schou [173]

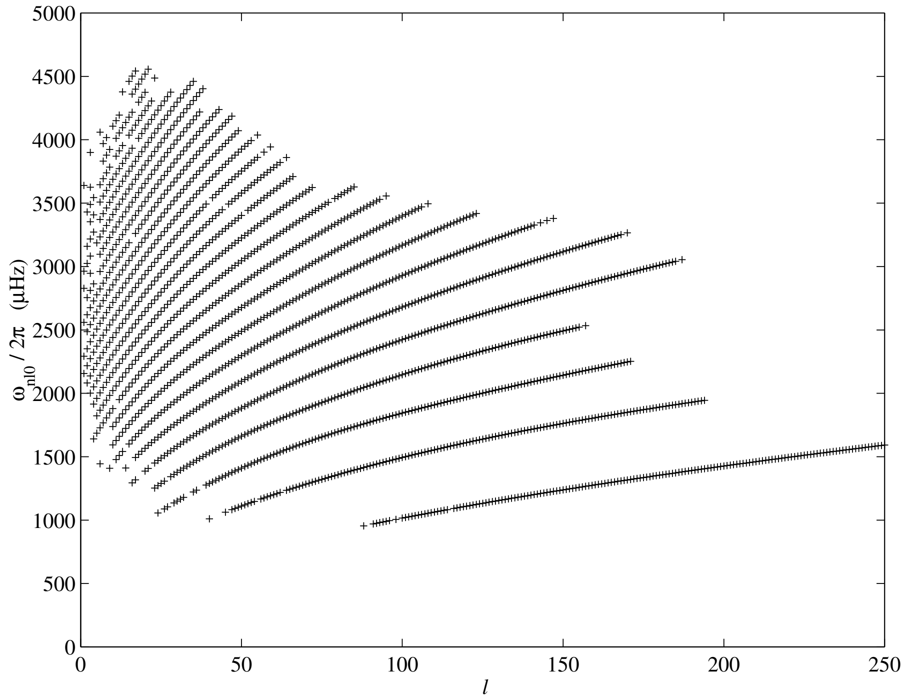


Figure 3.4: *Frequencies and degrees of the p- and f-modes in the MDI 360 day set. Each ridge corresponds to a certain value of* n*. The modes in the lowest ridge are the f-modes with* $n = 0$.

and are used in the MDI peak-bagging algorithm. See also Anderson et al. [6] for an earlier description of the a maximum likelihood estimation procedure, derived by modelling the modes as stochastically driven damped oscillators. The more recent treatment by Appourchaux et al. [7, 8] deals in particular with the maximum-likelihood methods used for fitting modes of low degree, $1 \leq l \leq 4$ say, found in, e.g., the observations from the GOLF and VIRGO instruments mentioned below. At low l, where the total number of modes is small, it is computationally feasible (and necessary due to lower signal to noise ratio) to include a more complete treatment of the leaks than is done in, e.g., the MDI algorithm. With the abovementioned techniques, frequencies for p-modes of degrees up to $l \approx 200$, radial order n up to 28 and frequencies in the range 1–4.5mHz and f-modes with l up to 250 have been determined. The degree and frequencies of the nl-multiplets in the dataset obtained from 360 days of MDI observations are shown in Figure 3.4. To give an idea of the amount of data involved, we mention that this dataset comprises some 2020 multiplets and a total of 33496 splitting coefficients.

For higher l, even the abovementioned technique breaks down, because the peaks for different l begin to overlap and blend to form ridges. There is hope, however, that so-called *ridge fitting* techniques (see Rhodes et al. [163]) will provide mode frequencies of modes with l up to approximately 1000 from MDI observations.

The MDI instrument on board the SOHO spacecraft (see Scherrer et al. [171] and Kosovichev et al. [122]) and to some extent the ground-based GONG network (see Harvey et al. [108]) have provided most of the data for the investigations, where the algorithms described in this thesis have been used; see Chapter 5 and Appendices D and G. Two other instruments for studying solar oscillations are on board the SOHO satellite. These are GOLF (Gabriel et al. [78]) and VIRGO (Frölich et al. [77]), which are mainly designed to observe low degree modes to provide data for studying, e.g., core rotation. Other instruments for observing low degree modes are the ground based LOWL instrument (Tomczyk et al. [196]) and the BiSON network, e.g. Elsworth et al. [72].

3.2 Helioseismic rotational inversion

To illustrate how measurements of solar oscillation frequencies can be used to infer the properties of the solar interior, we have adopted a simplified geometrical description from [44] of the influence of the Sun's rotation. As mentioned above, the frequencies ω_{nlm} are independent of m in a spherically symmetric (non-rotating) star. Now, assume that the Sun rotates with uniform angular velocity Ω and consider an oscillation with frequency ω_{nl0} independent of m in the frame of reference rotating with the star. Let (r', θ', ϕ') denote the coordinates in a coordinate system in this frame, then (r', θ', ϕ') are related to the coordinates (r, θ, ϕ) in an inertial frame of reference by the Galilei transformation

$$(r', \theta', \phi') = (r, \theta, \phi - \Omega t) \ .$$

It follows from (3.4) that the observer in the inertial frame observes a mode with frequency

$$\omega_{nlm} = \omega_{nl0} + m\,\Omega$$

and thus sees a frequency splitting according to m. By measuring the magnitude of the frequency splitting between the two oscillations with orders $\pm m$ – corresponding to two waves traveling in opposite directions around the Sun – the rotation rate Ω can be inferred by simply noting that

$$\omega_{nlm} - \omega_{nl-m} = 2m\,\Omega \ .$$

In the general case, the rotation $\Omega(r, \theta)$ depends on both radius and latitude since the Sun is not a rigid body. In two-dimensional helioseismic rotational inversion, the observed frequency splittings $\Delta\omega_{nlm}$ caused by the rotation are used to make inferences about the rotation rate as a function of radius and latitude. A more complete analysis shows that the change in frequency is approximately a weighted average of the rotation in the region of the Sun where the oscillation has an appreciable amplitude; to first order the rotation is related to the frequency splittings through the integral equation

$$\Delta\omega_{nlm} \equiv \omega_{nlm} - \omega_{nl0} = m \int_0^R \int_0^\pi K_{nlm}(r, \theta)\, \Omega(r, \theta)\, r \, \mathrm{d}r \mathrm{d}\theta \ , \tag{3.7}$$

where R is solar radius. The kernels $K_{nlm}(r, \theta)$ are determined by the oscillation eigenfunctions, which are calculated from a spherically symmetric (non-rotating) solar model, i.e., the frequency shifts are estimated as first order order perturbations to ω_{nl0}, for further details see, e.g., [45]. Equation (3.7) describes the inverse problem that we will be studying in the following.

3.2.1 Kernel structure

The helioseismic data come in two different forms: *individual frequency splittings* $\Delta\omega_{nlm}$ and so-called *splitting coefficients* (see below). In both cases kernels can be defined such that the

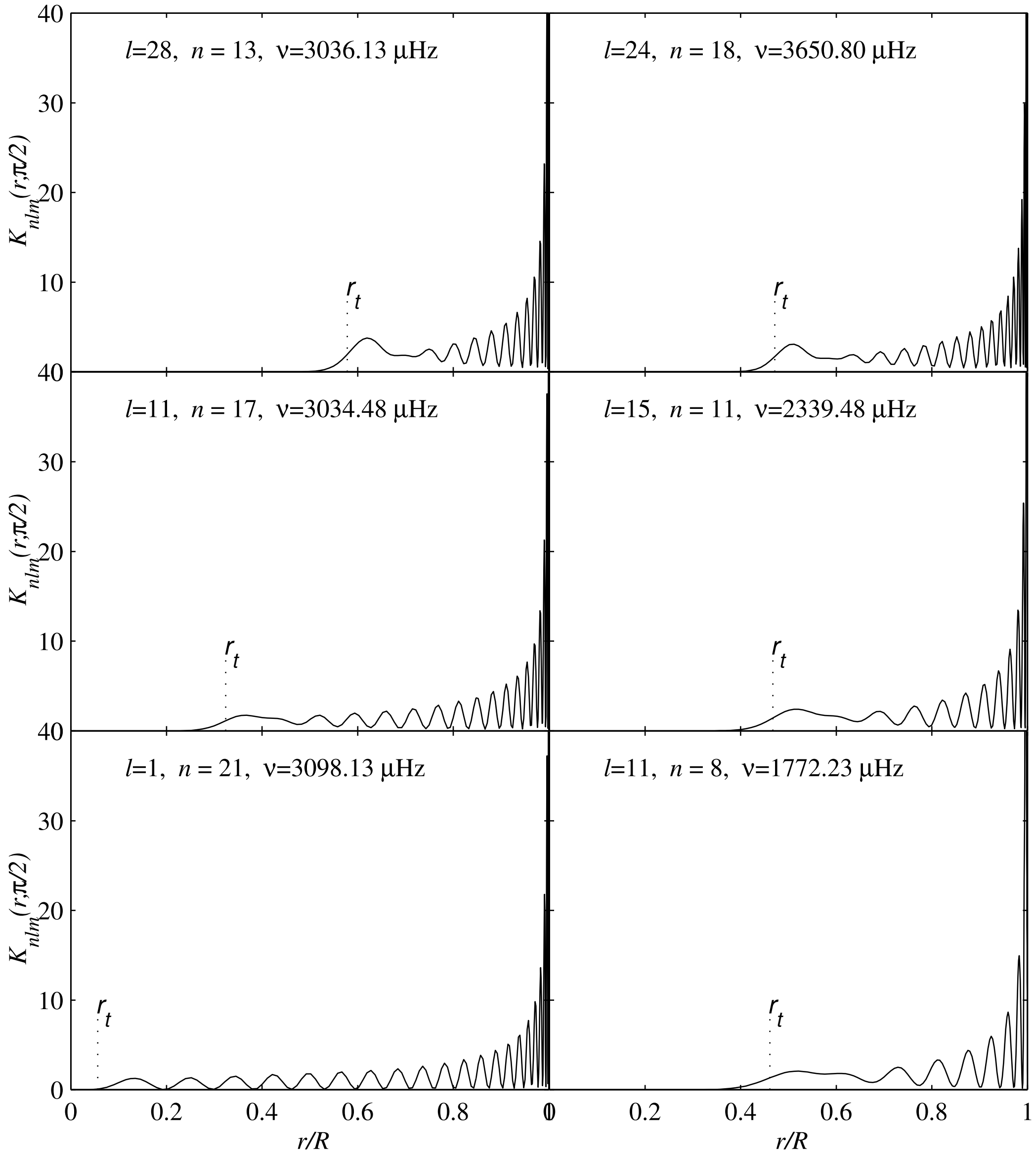


Figure 3.5: *Radial cuts through a few selected modekernels $K_{nlm}(r,\theta)$ in the equatorial plane. Left-hand panels show different modes with approximately the same frequency. Right-hand panels show modes with approximately the same value of ν/L corresponding to $r_t \approx 0.47R$. The vertical lines indicate the turning points.*

linear inverse problem can be written as in Equation (3.7). The properties of the kernels are important for understanding the inverse problem, since they describe the spatial sensitivity of the mode frequencies to rotation. In effect, the kernels together with the weighting of the data defined by the error-covariance estimates, completely determine the resolution that can be achieved by any inversion method. Below we give expressions for the kernels and illustrations of the radial and latitudinal behavior of a few selected modes.

Individual splittings

In the description above we have expressed the inverse problem in terms of frequency splittings $\Delta\omega_{nlm}$ of the individual normal modes. The corresponding kernels K_{nlm} can be written in the form

$$K_{nlm} = F_1^{nl}(r)\, G_1^{lm}(\theta) + F_2^{nl}(r)\, G_2^{lm}(\theta) \tag{3.8}$$

where

$$F_1^{nl}(r) = I_{nl}\left[\xi_{nl}(r)^2 - \frac{2}{L}\eta_{nl}(r)\xi_{nl}(r) + \eta_{nl}(r)^2\right]\rho(r)\, r\ , \tag{3.9}$$

$$F_2^{nl}(r) = I_{nl}\,\frac{\eta_{nl}(r)^2}{L^2}\,\rho(r)\, r\ , \tag{3.10}$$

$$G_1^{lm}(u) = P_l^m(u)^2\,\sin(\theta)\ , \tag{3.11}$$

$$G_2^{lm}(u) = \left[(1-u^2)\left(\frac{\mathrm{d}P_l^m(u)}{\mathrm{d}u}\right)^2 - 2uP_l^m(u)\frac{\mathrm{d}P_l^m(u)}{\mathrm{d}u} + \left(\frac{m^2}{1-u^2} - L^2\right)P_l^m(u)^2\right]\sin(\theta)\ , \tag{3.12}$$

and

$$I_{nl} = \int_0^R \left[\xi_{nl}(r)^2 + \eta_{nl}(r)^2\right]\rho(r)r^2\mathrm{d}r\ ,$$

and $u = \cos\theta$, see, e.g., [45]. Notice that the kernels are even functions in latitude. This means that we can only infer the component of the rotation, which is symmetric across the equator from observed frequency splittings. Any antisymmetric component will, to first order, have no effect on the observed frequencies.

Except for modes with very low l the behavior of the kernels is dominated by the term $F_1^{nl}(r)G_1^{lm}(\theta)$, which is illustrated in Figures 3.5 and 3.6. Figure 3.5 shows a cut through a few selected kernels along the equator and illustrates how the kernels have a large amplitude close to the surface that decreases rapidly with depth down to the turning point r_t, where it goes to zero. The most direct consequence of this behavior for inversions is that with increasing depth it becomes increasingly difficult to construct well-localized averaging kernels. This is enhanced by the nature of the measurement errors, which tend to be larger for deeply penetrating modes, see the right panel of Figure 3.7.

Figure 3.6 shows the latitudinal behavior of the kernels, which oscillate essentially like

$$G_1^{lm}(\theta) \simeq \mathrm{const}\cdot\cos^2[(l+1/2)\theta - \pi/4 + m\pi/2]\ ,$$

from the equator to the turning point θ_t, beyond which they go to zero. It is not obvious from this behavior that in latitude the main difficulty lies in constructing well-localized averages near the rotation axis. This is explained, however, by looking at the data-errors which we have plotted for data from the GONG months 21–23 dataset used in [176]; σ_{nlm} is the estimated standard deviation of the noise in the splittings which are assumed to be statistically independent. As can be seen in the left panel of Figure 3.7 the measurement errors grow dramatically when the turning point of the corresponding mode approaches the rotation axis, and therefore these modes are suppressed in the inversion.

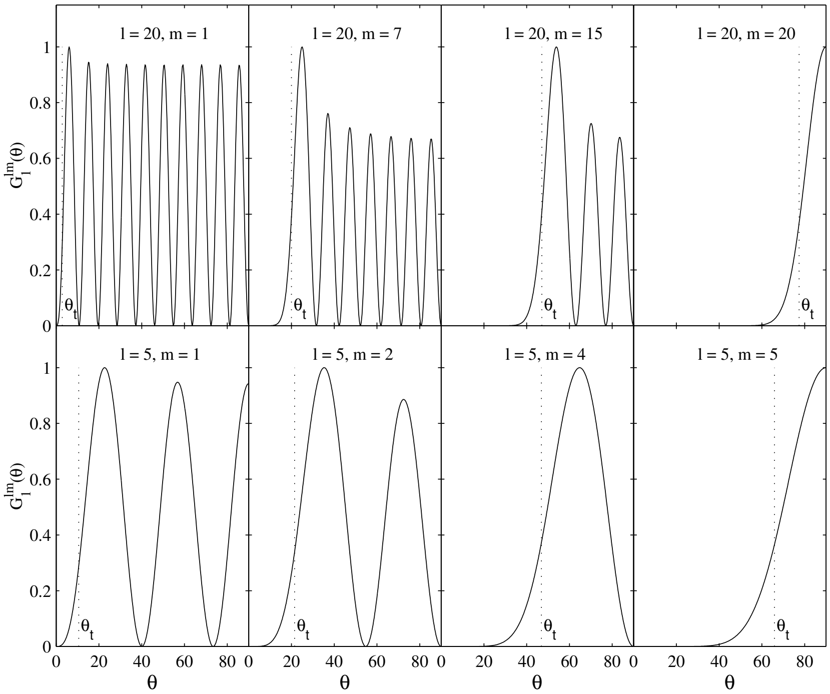


Figure 3.6: *Latitudinal behavior of a few selected modekernels for individual frequency splittings. The functions $G_1^{lm}(u)$ have been scaled to have a maximum absolute value of unity. The dotted vertical lines indicate the turning points.*

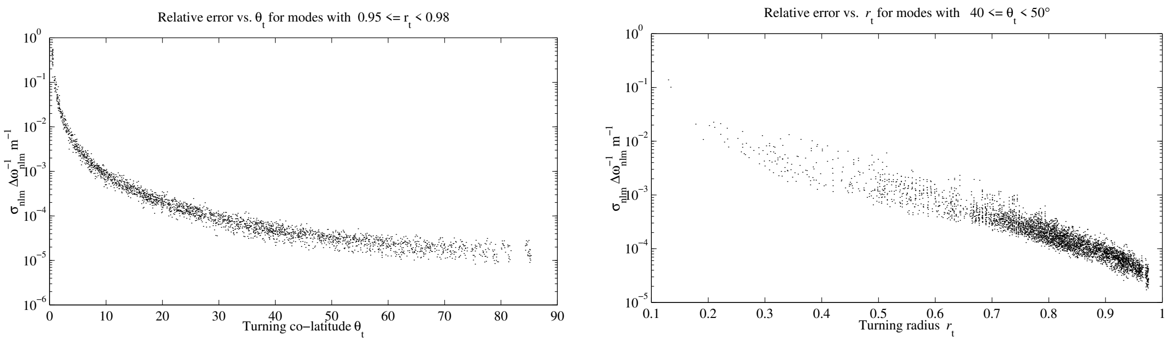


Figure 3.7: *Normalized errors estimates $\sigma_{nlm}/(m\,\Delta\omega_{nlm})$ for observations from GONG months 21–23. Left panel shows errors plotted versus turning point latitude θ_t. Right panel shows errors plotted versus turning point radius r_t.*

Splitting coefficients

As mentioned earlier, helioseismic observations are generally not provided in the form of individual frequency splittings, but instead, e.g., the MDI data are expressed in terms of *splitting coefficients* a_j defined in Equation (3.6). To first order the rotational splitting only contributes to the odd splitting coefficients, which in the following will be indexed by $j = 2s+1$, i.e., $s_{\max} = (j_{\max}-1)/2$.

From Equation (3.6) it can be shown that the splitting coefficients are linear combinations of the individual splittings

$$2\pi a_j(n,l) = \sum_m \gamma_j(l,m)\Delta\omega_{nlm} \ , \tag{3.13}$$

for some coefficients $\gamma_j(m,l)$. This means that the inverse problem in Equation (3.7) can be expressed in terms of the splitting coefficients thus

$$\begin{aligned} 2\pi a_{2s+1}(n,l) &= \int_0^R \int_0^\pi \sum_m \gamma_{2s+1}(l,m) K_{nlm}(r,\theta)\,\Omega(r,\theta)\,r\,\mathrm{d}r\mathrm{d}\theta \\ &\equiv \int_0^R \int_0^\pi K^{(a)}_{nls}(r,\theta)\,\Omega(r,\theta)\,r\,\mathrm{d}r\mathrm{d}\theta \ , \end{aligned} \tag{3.14}$$

where we have defined the kernels $K^{(a)}_{nls}$ corresponding to the splitting coefficients $a_{2s+1}(n,l)$. To set up these kernels we have used the code from [178], where the coefficients $\gamma_j(m,l)$ are extracted from the modified Gram-Schmidt procedure used to set up the polynomials $\mathcal{P}^{(l)}_j$ (see Appendix A in [178]); the corresponding kernels are then computed according to Equation (3.14). Recently Pijpers [157] has derived closed form expressions for the polynomials and the corresponding kernels. When the normalization in Equation (3.2) is used, these expressions for the kernels can be written in the form

$$K^{(a)}_{nls} = F^{nl}_1(r)\,G^{ls}_1(\theta) + F^{nl}_2(r)\,G^{ls}_2(\theta) \tag{3.15}$$

where

$$\begin{aligned} G^{ls}_1(u) &= v_{ls}\left[\frac{(4s+3)}{2(2s+2)(2s+1)}\right]^{1/2}(1-u^2)\,P^1_{2s+1}(u) \ , \qquad (3.16) \\ G^{ls}_2(u) &= -\frac{v_{ls}}{2}\left[(4s+3)(2s+2)(2s+1)\right]^{1/2}(1-u^2)\,P^1_{2s+1}(u) \ , \qquad (3.17) \end{aligned}$$

where the normalization constant v_{ls} is given by

$$v_{ls} = (-1)^s\,\frac{1}{l}\,\frac{(2l+1)!(2s+2)!(l+s+1)!}{s!(s+1)!(l-s-1)!(2l+2s+2)!}$$

The radial behavior of $K^{(a)}_{nls}$ is identical to that of K_{nlm}, while the latitudinal behavior is illustrated in Figure 3.8. The effect of the latitudinal behavior of $K^{(a)}_{nls}$ on inversions is discussed in Appendix D.

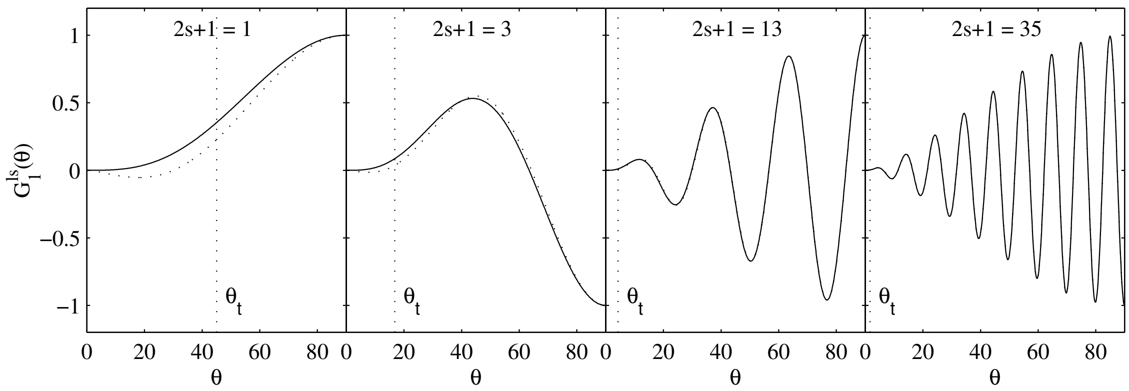


Figure 3.8: *Latitudinal behavior of a few selected a-coefficient modekernels for* $s = 1, 2, 6, 17$. *The functions* $G_1^{ls}(u)$ *have been scaled to have a absolute value of unity at* $\theta = \pi/2$. *The solid line shows* $G_1^{ls}(u)$ *while the dotted lines show the asymptotic expansion discussed in Appendix D. The vertical lines indicate the turning points.*

3.3 Formulation based on Kronecker products

The key to reducing the computational complexity and storage requirements of the problem is the observation that the kernels K_{nlm} (or $K_{nls}^{(a)}$) can be written on the form

$$K(r, \theta) = F_1(r)\, G_1(\theta) \;+\; F_2(r)\, G_2(\theta)\ . \qquad (3.18)$$

We now describe, in terms of K_{nlm}, how this algebraic structure gives rise to a kernel-matrix with *block Kronecker product* structure when the integral in Equation (3.7) is discretized, and discuss some practical considerations involved in dealing with real datasets, when individual modes are missing or when splitting coefficients corresponding to polynomial expansions of varying degree are present in the same dataset.

3.3.1 Derivation of the block Kronecker product structure

Following the description in Section 2 we discretize Equation (3.7) by representing the kernels K_{nlm} on a regular grid with N_r points $r_1, r_2, \ldots, r_{N_r}$ in radius and N_θ points $\theta_1, \theta_2, \ldots, \theta_{N_\theta}$ in latitude, and use a quadrature rule with weights $w_{ij},\ i = 1, \ldots, N_r,\ j = 1, \ldots, N_\theta$, such that the double integrals are replaced by

$$\int_0^R \int_0^\pi K_{nlm}(r, \theta)\, r\, \mathrm{d}r\mathrm{d}\theta \;\approx\; \sum_{i=1}^{N_r} \sum_{j=1}^{N_\theta} w_{ij} K_{nlm}(r_i, \theta_j) \qquad (3.19)$$

We introduce the matrix $\mathbf{K} \in \mathbb{R}^{M \times N}$ defined as

$$(\mathbf{K})_{k,(i-1)N_r+j} = K_k(r_i, \theta_j) \ , \qquad k = 1, \ldots, M \ , \quad i = 1, \ldots, N_r \ , \quad j = 1, \ldots, N_\theta \ ,$$

where we have used a single index k to enumerate the M different combinations of n, l and m, and $N = N_r N_\theta$ is used to denote the total number of gridpoints. We have furthermore assumed that the gridpoints have been mapped to a row of $\mathbf{K}$ using a simple lexicographical ordering as illustrated in Figure 3.9.

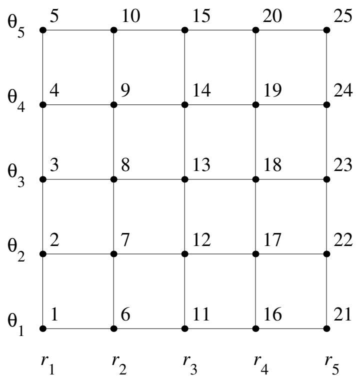


Figure 3.9: *Illustration of a simple lexicographical numbering of the points in a* 5×5 *grid.*

If we substitute the expression for K_{nlm} in Equation (3.18) into the definition of $\mathbf{K}$ we can set $\mathbf{K} \equiv \mathbf{K}_1 + \mathbf{K}_2$ with

$$(\mathbf{K}_1)_{k,(i-1)N_r+j} = F_1^{nl}(r_i)\, G_1^{lm}(\theta_j) \ , \tag{3.20}$$

$$(\mathbf{K}_2)_{k,(i-1)N_r+j} = F_2^{nl}(r_i)\, G_2^{lm}(\theta_j) \ . \tag{3.21}$$

Let us take a closer look at $\mathbf{K}_1$:

$$(\mathbf{K}_1)_{k,(i-1)N_r+j} = F_1^{nl}(r_i)\, G_1^{lm}(\theta_j) \equiv f_i\, g_j \ , \quad i = 1, \ldots, N_r \ , \quad j = 1, \ldots, N_\theta \ .$$

We can describe the structure of the kth row in $\mathbf{K}_1$ as follows

$$\begin{aligned}(\mathbf{K}_1)_{k,:} \;=\; \Big[\, & f_1\, g_1\ f_1\, g_2\ \cdots\ f_1\, g_{N_\theta}\ f_2\, g_1\ f_2\, g_2\ \cdots\ f_2\, g_{N_\theta}\ \cdots \\ & \cdots f_{N_r}\, g_1\ f_{N_r}\, g_2 \cdots\ f_{N_r}\, g_{N_\theta} \,\Big] \ ,\end{aligned}$$

where $f_i = F_1^{nl}(r_i)$ and $g_j = G_1^{lm}(\theta_j)$. By introducing the row-vectors $\mathbf{f} = (f_1, f_2, \ldots, f_{N_r})$ and $\mathbf{g} = (g_1, g_2, \ldots, g_{N_\theta})$ the above can be written as

$$(\mathbf{K}_1)_{k,:} = \;=\; \Big[f_1\, \mathbf{g}\ f_2\, \mathbf{g}\ \cdots\ f_{N_r}\, \mathbf{g} \Big] \tag{3.22}$$

$$= \mathbf{f} \otimes \mathbf{g} \ . \tag{3.23}$$

The symbol "$\otimes$" denotes the *Kronecker product*, which is defined as:

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \ ,$$

i.e., it is the matrix obtained by forming the products of all combinations of elements from matrices $\mathbf{A}$ and $\mathbf{B}$. By representing a row in $\mathbf{K}$ using the Kronecker product, we only need to store the $N_r + N_\theta$ elements in $\mathbf{f}$ and $\mathbf{g}$ instead of the $N_r N_\theta$ elements required if the corresponding row of $\mathbf{K}$ was stored explicitly. In the 2D helioseismic inversions where typically $N_r \approx 100$ and $N_\theta \approx 50$ this alone reduces the amount of storage approximately by a factor of 30.

A further improvement comes from realizing that the columns in $\mathbf{K}$ also have a regular structure that can be described as a Kronecker product. To see this, consider the set of rows in $\mathbf{K}_1$ corresponding to modes with a certain degree l: In this set there will typically be a number of different so-called *multiplets* (nl-combinations) corresponding to different values of n, and for each multiplet we assume (see below) that there are modes for every value of m in the interval $1, 2, \ldots, l$. With the assumption that every multiplet contains all different m-values we notice from Equation (3.20) that if we look down a column of $\mathbf{K}_1$ then every combination of $F_1^{nl}(r_i)$ and $G_1^{lm}(\theta_j)$ appears exactly once. This is precisely the kind of structure that a Kronecker product is used to express!

If we introduce the matrices

$$\begin{aligned} (\mathbf{F}_1^{(l)})_{pi} &= F_1^{pl}(r_i) \ , \quad p = 1, \ldots, \#_n(l) \ , \quad i = 1, \ldots, N_r \ , \\ (\mathbf{G}_1^{(l)})_{qj} &= G_1^{lq}(\theta_j) \ , \quad q = 1, \ldots, l \ , \quad j = 1, \ldots, N_\theta \ , \end{aligned}$$

where $\#_n(l)$ is the function that counts the number of multiplets with degree l, then we can write the portion of $\mathbf{K}_1$ corresponding to modes with degree l as

$$\mathbf{F}_1^{(l)} \otimes \mathbf{G}_1^{(l)} \ .$$

Example: Assume that a modeset contains the following multiplets $(n, l) \in \{(2,3), (5,3), (6,3)\}$ of degree 3, and all modes with $1 \leq m \leq l$, corresponding to the following:

l	n	m
3	2	1
3	2	2
3	2	3
3	5	1
3	5	2
3	5	3
3	6	1
3	6	2
3	6	3

Using the observation above, the $\mathbf{K}_1$-part of the discretized kernels for these 9 modes can be expressed compactly as the Kronecker product of a 3-by-N_r and a 3-by-N_θ matrix:

$$\begin{pmatrix} F_1^{2,3}(r_1) & F_1^{2,3}(r_2) & \cdots & F_1^{2,3}(r_{N_r}) \\ F_1^{5,3}(r_1) & F_1^{5,3}(r_2) & \cdots & F_1^{5,3}(r_{N_r}) \\ F_1^{6,3}(r_1) & F_1^{6,3}(r_2) & \cdots & F_1^{6,3}(r_{N_r}) \end{pmatrix} \otimes \begin{pmatrix} G_1^{3,1}(\theta_1) & G_1^{3,1}(\theta_2) & \cdots & G_1^{3,1}(\theta_{N_\theta}) \\ G_1^{3,2}(\theta_1) & G_1^{3,2}(\theta_2) & \cdots & G_1^{3,2}(\theta_{N_\theta}) \\ G_1^{3,3}(\theta_1) & G_1^{3,3}(\theta_2) & \cdots & G_1^{3,3}(\theta_{N_\theta}) \end{pmatrix}$$

By repeating the construction above for every degree $1, 2, \ldots, l_{\max}$ we can extend this to all of $\mathbf{K}$ and finally get the *block Kronecker product* structure

$$\mathbf{K} = \begin{pmatrix} \mathbf{F}_1^{(1)} \otimes \mathbf{G}_1^{(1)} \\ \mathbf{F}_1^{(2)} \otimes \mathbf{G}_1^{(2)} \\ \vdots \\ \mathbf{F}_1^{(l_{\max})} \otimes \mathbf{G}_1^{(l_{\max})} \end{pmatrix} + \begin{pmatrix} \mathbf{F}_2^{(1)} \otimes \mathbf{G}_2^{(1)} \\ \mathbf{F}_2^{(2)} \otimes \mathbf{G}_2^{(2)} \\ \vdots \\ \mathbf{F}_2^{(l_{\max})} \otimes \mathbf{G}_2^{(l_{\max})} \end{pmatrix} , \qquad (3.24)$$

where $l_{\max}$ is the highest degree present in the modeset.

The analysis in Appendix C shows that by using the block Kronecker product formulation, the amount of storage needed is reduced from $\mathcal{O}(MN)$ to $\mathcal{O}(2[M_{nl}N_r+M_{lm}N_\theta])$, where M_{nl} is the number of different multiplets and M_{lm} is the number of different lm-combinations in the modeset. More importantly, the computational complexity of a matrix-vector multiplication is reduced from $\mathcal{O}(2MN)$ to $\mathcal{O}(4\min(MN_\theta + M_{lm}N,\, MN_r + M_{nl}N))$, which for individual splittings is asymptotically $\mathcal{O}(M\sqrt{N} + \sqrt{M}N)$ when $N_r = \text{const} \cdot N_\theta$. For large modesets, where M and N are at least of order 10^4 this represents a reduction in the number of floating point operations by roughly two orders of magnitude (assuming that all lm-combinations are present, see next paragraph).

3.3.2 Practical considerations

The assumption that for a given nl-multiplet in the dataset, modes with all different value of m from 1 to l will be present, is not generally true in practice. Often certain modes are missing because their frequency could not be determined reliably by the peak-bagging algorithm. However, the assumption often holds for a substantial number of multiplets, and even when this is not the case it is always possible to break the modeset into smaller subsets of multiplets with the same degree that contain the same set of m-values. Datasets consisting of splitting coefficients are more regular in this respects since there are rarely individual coefficients missing from a multiplet. However, when polynomial expansions (cf. Equation 3.6) with different numbers of coefficients are used for different multiplets with the same degree, the corresponding kernels cannot be expressed as a single Kronecker product and it is necessary subdivide the modeset into smaller blocks.

In our implementation we perform a preprocessing step where the modeset is analyzed to determine this partitioning into blocks of multiplets for which the kernels can be expressed as Kronecker products. In Figure 3.10 we show the result of the preprocessing for the MDI 144 day modeset, which consist of splitting coefficients. The smallest possible number of blocks was $l_{\max} = 250$ but when the modeset was subdivided to account for multiplets with different numbers of coefficients in the polynomial expansion, the final number of blocks was 485. The histogram in the right panel shows the distribution of the number of multiplets that were combined in a single block, i.e., the number of rows in the block-matrices $\mathbf{F}_1^{(i)}$ and $\mathbf{F}_2^{(i)}$. The histogram in the right panel of Figure 3.10 shows the distribution of the number of splitting coefficients in a block, i.e., the number of rows in the block-matrices $\mathbf{G}_1^{(i)}$ and $\mathbf{G}_2^{(i)}$. For this dataset most multiplets were analyzed using expansions with 3, 9 or 18 coefficients and the block sizes in the histogram essentially shows the relatively few blocks had to be split due to missing coefficients. For this particular modeset the amount of storage needed to represent the kernels is reduced from approximately 1500 Mbytes to approximately 7 Mbytes. The time to calculate $\mathbf{Kx}$ or $\mathbf{K}^T\mathbf{y}$ is reduced by a factor of 49 by using a subroutine KRONMV based on the Kronecker product formulation instead of the normal Level-2 BLAS matrix-vector routine DGEMV, when measured on a single processor of an SGI Power Challenge. However only approximately a factor of 7 is due to the reduced number of operations. The additional factor of 7 comes from better cache performance derived from the ability to formulate the operations in connection with each Kronecker product as matrix-matrix multiplications. This issue is discussed further below.

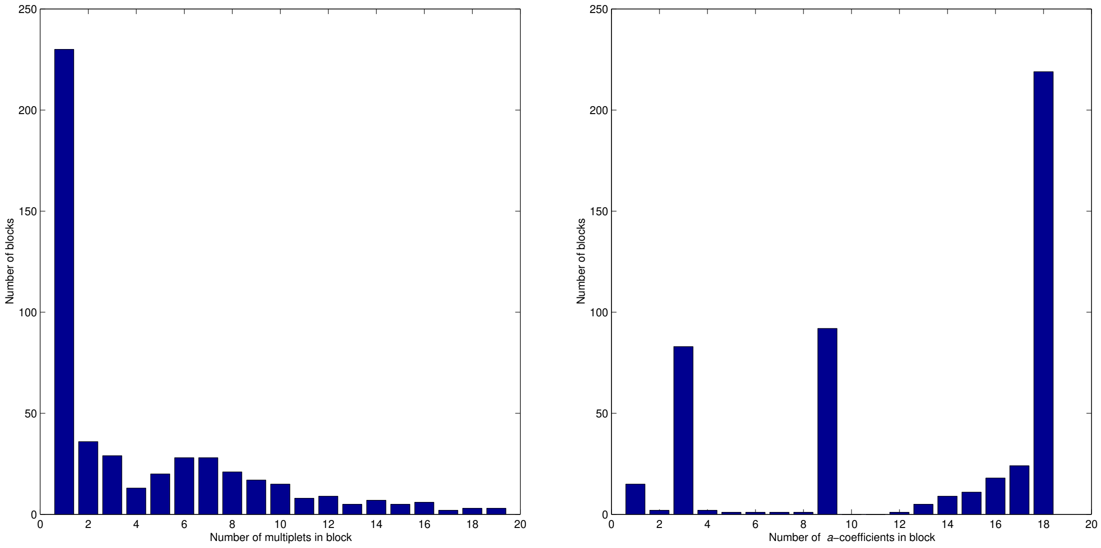


Figure 3.10: *Block sizes in the block Kronecker matrix. Left panel shows the distribution of the number of the number of rows in* $\mathbf{F}_1^{(i)}$ *and* $\mathbf{F}_2^{(i)}$. *Right panel shows the distribution of the number of rows in the* $\mathbf{G}_1^{(i)}$ *and* $\mathbf{G}_2^{(i)}$.

3.3.3 Simplifications for splitting coefficients

In the next section we summarize some of the results that have followed from the analysis above. Before, we would like to elaborate a little on a possible improvement: An analysis similar to the one above can be made for modesets based on splitting coefficients. In fact, a further simplification is possible since according to Equations (3.16) and (3.17) the functions $G_i^{ls}(\theta)$, $i = 1,2$ are the same for *all* modekernels regardless of their degree, apart from the normalization v_{ls} which depends on l. With current datasets, where only $a_1, a_3, \ldots, a_{35}$ are used, this means that 18 functions is all that is needed to represent the latitudinal behavior of the entire set of modekernels. If we define the matrices

$$(\mathbf{G}^{(a)})_{sj} = (1-u_j^2)P_{2s+1}^1(u_j)\ , \quad s = 1,\ldots,s_{\max}\ , \quad j = 1,\ldots,N_\theta\ , \tag{3.25}$$

$$(\mathbf{D}_1^l)_{ss} = v_{ls}\left[\frac{(4s+3)}{2(2s+2)(2s+1)}\right]^{1/2}\ , \quad s = 1,\ldots,s_{\max} \tag{3.26}$$

$$(\mathbf{D}_2^l)_{ss} = -\frac{v_{ls}}{2}\left[(4s+3)(2s+2)(2s+1)\right]^{1/2}\ , \quad s = 1,\ldots,s_{\max}\ , \tag{3.27}$$

where $u_j = \cos\theta_j$, then we can write the discretization of $K_{nlm}^{(a)}(r,\theta)$ as

$$\mathbf{K}^{(a)} = \begin{pmatrix} \mathbf{F}_1^{(1)} \otimes \mathbf{D}_1^{(1)}\mathbf{G}^{(a)} \\ \mathbf{F}_1^{(2)} \otimes \mathbf{D}_1^{(2)}\mathbf{G}^{(a)} \\ \vdots \\ \mathbf{F}_1^{(l_{\max})} \otimes \mathbf{D}_1^{(l_{\max})}\mathbf{G}^{(a)} \end{pmatrix} + \begin{pmatrix} \mathbf{F}_2^{(1)} \otimes \mathbf{D}_2^{(1)}\mathbf{G}^{(a)} \\ \mathbf{F}_2^{(2)} \otimes \mathbf{D}_2^{(2)}\mathbf{G}^{(a)} \\ \vdots \\ \mathbf{F}_2^{(l_{\max})} \otimes \mathbf{D}_2^{(l_{\max})}\mathbf{G}^{(a)} \end{pmatrix}$$

$$= \left[\begin{pmatrix} \mathbf{F}_1^{(1)} \otimes \mathbf{D}_1^{(1)} \\ \mathbf{F}_1^{(2)} \otimes \mathbf{D}_1^{(2)} \\ \vdots \\ \mathbf{F}_1^{(l_{\max})} \otimes \mathbf{D}_1^{(l_{\max})} \end{pmatrix} + \begin{pmatrix} \mathbf{F}_2^{(1)} \otimes \mathbf{D}_2^{(1)} \\ \mathbf{F}_2^{(2)} \otimes \mathbf{D}_2^{(2)} \\ \vdots \\ \mathbf{F}_2^{(l_{\max})} \otimes \mathbf{D}_2^{(l_{\max})} \end{pmatrix} \right] \mathbf{G}^{(a)} \tag{3.28}$$

This expression is the closest one can get to a completely separable problem, but alas, the normalization factors v_{ls} in the diagonal matrices $\mathbf{D}_1^{(l)}$ and $\mathbf{D}_2^{(l)}$ depend differently on l and therefore the two terms in the sum cannot be combined. Had this not been the case, Equation (3.28) could have been written in the form

$$\mathbf{K}^{(a)} = \begin{pmatrix} \hat{\mathbf{F}}^{(1)} \\ \hat{\mathbf{F}}^{(2)} \\ \vdots \\ \hat{\mathbf{F}}^{(l_{\max})} \end{pmatrix} \otimes \mathbf{G}^{(a)} \equiv \mathbf{F}^{(a)} \otimes \mathbf{G}^{(a)} ,$$

and the solution to the least-squares problems involving $\mathbf{K}^{(a)}$ that occur in RLS and OLA inversions could have been solved extremely efficiently using the following property of the Kronecker product:

$$(\mathbf{A} \otimes \mathbf{B})^\dagger = (\mathbf{A}^\dagger \otimes \mathbf{B}^\dagger) ,$$

where $\dagger$ denotes the More-Penrose pseudo-inverse (see, e.g., [25, Section 8.5]).

Still, Equation (3.28) is interesting since it shows that the amount of storage needed can be reduced to

$$\mathcal{O}(2N_r M_{nl} + N_\theta s_{\max} + l_{\max} s_{\max})$$

where M_{nl} is the total number of multiplets, which is typically only a few thousand, and the last term is from the amount of memory needed to store the diagonal normalization matrices $\mathbf{D}_i^{(l)}, l = 1, \ldots, l_{\max}\ i = 1, 2$. For the MDI 144 days set, this would reduce the storage needed for $\mathbf{K}$ to 3.1Mbytes.

We have not implemented this improved version of the Kronecker algorithm yet, since it was discovered only recently – after having properly digested Pijpers' excellent paper [157]. Originally, we started out implementing the more general formulation given above, which applies to both kinds of data and which allows us to use the same datastructures as the program written by Schou (cf. [178]) that is used for setting up $\mathbf{F}_i^{(l)}$ and $\mathbf{G}_i^{(l)}$. The formulation in Equation (3.28) does promise a further reduction in the execution time for datasets consisting of splitting coefficients, since in the current implementation most of the time used by the Kronecker matrix-vector routine is actually spent multiplying by the blocks $\mathbf{G}_i^l$ in Equation (3.24) (see Appendix C), something that would be replaced in the improved formulation by a single multiplication with $\mathbf{G}^{(a)}$ and a simple scaling by each block by the diagonal matrices $\mathbf{D}_i^l$. Whether it is worthwhile to try and reduce the execution time even further is another matter.

3.3.4 Savings from using inversion algorithms based on Kronecker products

The highly regular structure can be exploited when multiplying a vector by a Kronecker product. To show this we first introduce the function $\mathrm{vec}(\cdot) : \mathbb{R}^{m \times n} \to \mathbb{R}^{mn}$ which stacks the

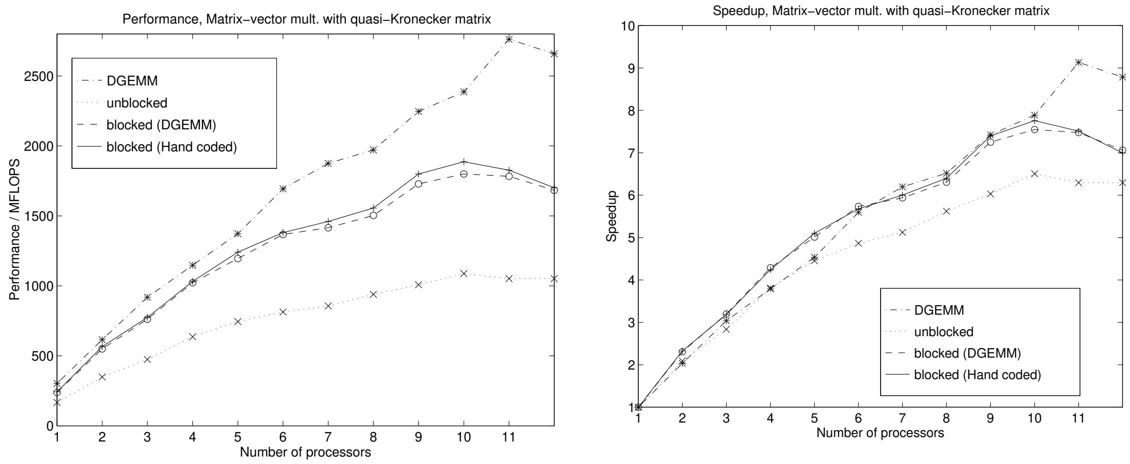


Figure 3.11: *Performance and speedup for different implementations of* KRONMV *compared to* DGEMM. *Left panel shows performance in MFLOPS and right panel shows speedup as a function as the number of processors. In both panels the MDI 144 day dataset was used. The dotted line corresponds to a BLAS-1 (unblocked) version of* KRONMV, *while dashed and solid lines correspond to a BLAS-3 implementation. In the "hand coded" version (solid line) the inner loops in the multiplication of the Kronecker blocks were unrolled manually. The dot-dashed line shows the results for the optimized matrix-matrix multiplication routine* DGEMM *from BLAS provided by Silicon Graphics. To make the comparison meaningful, the values for* DGEMM *were measured by multiplying matrices of a size similar to the blocks in* **K**.

columns of a matrix into a long vector. If $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n) \in \mathbb{R}^{m\times n}$ then we define

$$\mathrm{vec}(\mathbf{A}) = \begin{pmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_n \end{pmatrix} . \tag{3.29}$$

We can now state the following important result: If $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\quad \mathbf{B} \in \mathbb{R}^{p\times q}$, and $\mathbf{x} = \mathrm{vec}(\mathbf{X})$, where $\mathbf{X} \in \mathbb{R}^{q\times n}$, then

$$(\mathbf{A} \otimes \mathbf{B})\,\mathbf{x} = \mathrm{vec}(\mathbf{B}\,\mathbf{X}\,\mathbf{A}^T) \ . \tag{3.30}$$

Consequently, if $\mathbf{A}$ and $\mathbf{B}$ are $n \times n$ matrices the cost of computing $(\mathbf{A} \otimes \mathbf{B})\,\mathbf{x}$ is $2n^4$ while $\mathrm{vec}(\mathbf{B}\,\mathbf{X}\,\mathbf{A}^T)$ requires only $4n^3$ floating point operations.

In Appendix C this property is used to derive a fast algorithm for computing the product of a vector with the kernel-matrix $\mathbf{K}$ or $\mathbf{K}^T$. This algorithm is furthermore very efficient when implemented on modern computer architectures since it can be formulated entirely as Level-3 BLAS operations. This important aspect is illustrated in Figure 3.11. We have implemented the fast matrix-vector multiplication algorithm in a FORTRAN subroutine KRONMV and in the figure the performance and parallel speedup of three versions of KRONMV are compared with the values for the optimized matrix-matrix multiplication routine DGEMM from the optimized BLAS library of the SGI Power Challenge parallel multiprocessor on which the experiments were carried out. The DGEMM routine has been tuned by SGI to obtain the best possible performance on the machine, since this routine is responsible for most of the work in many so-called block algorithms that are used implement the majority of the routines

Table 3.1: Reduction in storage and execution time for matrix-vector multiplication compared with GEMV resulting from the use of KRONMV, when $N_r = 101$ and $N_\theta = 51$.

	M	M_{nl}	M_{lm}	Storage	Operations	$T_{\mathsf{GEMV}}/T_{\mathsf{KRONMV}}$
LOWL	3279	1102	294	33	4.6	29
GONG	4704	1556	793	43	2.7	18
MDI	30648	2036	5255	136	6.9	49
MDIm	150372	2029	31375	202	21.9	—

in modern software libraries for numerical linear algebra such as LAPACK. For this reason, the performance of DGEMM is in most cases the "peak performance" of the machine that one should compare with. As can be seen in the left panel of Figure 3.11, the Level-3 BLAS version of KRONMV achieves approximately 90% of the performance of DGEMM for 1–5 processors, which is a very satisfactory result. With more processors the performance decreases gradually because there is simply not enough work to keep all processors occupied. With more than 10 processors the performance actually begins to decrease due to parallel overhead. The speedup curve also shows that the speedup is almost linear with fewer than 10 processors. The speedup is for each algorithm measured relative to the execution time on single processor processor of the given algorithm. This is the reason why the speedup curve for KRONMV above that of DGEMM although one might expect the opposite from the behavior in the left panel. In Table 3.1, which reproduced from Appendix C, is shown the factors by which the storage requirements and operations are reduced when using KRONMV instead of the usual Level-2 BLAS matrix-vector routine DGEMV. The last column, which shows the measured reduction in execution time measured on a single processor of a SGI Power Challenge, is highly interesting. It shows that the reduction in execution time is far greater than expected from the reduction in the number of operations alone. The additional factor is caused by the superior cache performance of KRONMV, and this is in fact the greater of the two factors for all four datasets. The fast matrix-vector multiplication routine KRONMV forms the basis of a number of iterative implementations of the OLA inversion methods based on Lanczos bidiagonalization. The algorithms are described in detail in Appendix C and in Chapter 4.

Here we show a few examples of the savings in storage for the MDI 144 day dataset that result from use of the Kronecker formulation for various regularization algorithms that we have used. Table 3.2 lists the storage requirements for $\mathbf{K}$ used, e.g., in the direct bidiagonalization algorithm in Appendix B or SVD based methods. The matrix $\mathbf{K}^T\mathbf{K}$ is the coefficient matrix of the normal equations used to implement RLS (Tikhonov regularization) and $\mathbf{K}\mathbf{K}^T$ is the (large) matrix that would result if the OLA methods were implemented via the normal equations. In the iterative implementations of SOLA or MOLA based on the LSQR algorithm it is only necessary to store the $\mathbf{F}$ and $\mathbf{G}$ matrices (apart from a few vectors), and thus the storage requirements are reduced dramatically. However, as described in Appendices C and A, the most efficient implementation of SOLA uses the Lanczos bidiagonalization with full reorthogonalization, and here it is necessary to save the matrices $\mathbf{U}$ and $\mathbf{V}$ containing the Lanczos vectors, which take up a lot of memory. Still, because the number of Lanczos iterations needed is a factor of 5 smaller than the dimensions of the problem, the total amount of storage is not much higher than for RLS solved via the normal equations. As discussed in Appendix A we are currently investigating a number of algorithms for reducing the cost

Table 3.2: Storage requirements for the MDI 144 day dataset in various formulations.

Matrices	Dimensions	Storage in Mb	Description
$\mathbf{K}$	30648×5151	1500	Full kernel matrix
$\mathbf{K}^T\mathbf{K}$	5051×5051	200	Normal equations, RLS
$\mathbf{K}\mathbf{K}^T$	30648×30648	7200	Normal equations, OLA
$\mathbf{F}_1$, $\mathbf{F}_2$	2036×101	3	Radial part of Kronecker product
$\mathbf{G}_1$, $\mathbf{G}_2$	5255×51	4	Angular part of Kronecker product
$\mathbf{G}^{(a)}$	18×51	0.007	Angular part of Kronecker product (coeff.)
$\mathbf{U}$	5051×1000	40	Left Lanczos vectors, SOLA
$\mathbf{V}$	30648×1000	235	Right Lanczos vectors, SOLA

Table 3.3: Storage and execution time used by three different implementations for computing the 2d SOLA solution in every point of the 101×51 discretization grid.

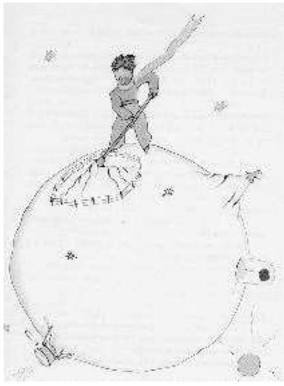
		Storage/Mb			Time/minutes		
	M	Direct	Lanczos	LSQR	Direct	Lanczos	LSQR
LOWL	3279	140	75	11	12.3	2.5	90
GONG	4704	200	90	15	45	3.3	200
MDI	30648	1300	300	26	375	7.5	750
MDIm	150372	6200	1274	73	NA	26	2400

of or completely avoiding the reorthogonalization. The numbers in the table are calculated assuming that matrices are stored in IEEE double precision format which means 8 bytes pr. floating point number.

Table 3.3 shows a comparison of the storage and execution time for three implementations of SOLA. For each dataset, the rotation was estimated for each of the 5151 points in the discretization grid, i.e., a linear least-squares problem of dimension $(M-1) \times N$ was solved for N different right-hand sides. For the large MDI datasets, the iterative method based on Lanczos bidiagonalization with full reorthogonalization (see Chapter 4) is approximately 60 times faster than the direct method (the bidiagonalization algorithm from Appendix B. The table clearly displays the time/space trade-off between the two iterative algorithms; the LSQR-algorithm is slow since it must solve the system for each right-hand independently but only needs to store a few vectors in addition to the $\mathbf{F}$ and $\mathbf{G}$ matrices. In contrast, the Lanczos algorithm, which uses full reorthogonalization, needs much more storage to save the Lanczos vectors, but is much faster since it only needs to compute the Lanczos bidiagonalization once.

Chapter 4

Efficient algorithms for OLA inversions



Hät ich tausend arme zu rühren!
Könnt ich brusend die Räder führen!
Könnt ich wehen durch alle Haine!
Könnt ich drehen alle Steine!
Daß die schöne Müllerin
Merkte meinen treuen Sinn!

— Wilhelm Müller, from “Die Schöne Müllerin”

The method of optimally localized averages of Section 4 has since its introduction by Backus and Gilbert in 1968 [10] been used for solving inverse problems in many areas including astronomy [54], seismology [149], helioseismology [87] and signal processing [136, 137]. The implementations used in these publications, as well as the one given by Press et al. in [161], are all based on a formulation involving the normal equations, which incorporates the unity-norm constraint on the averaging kernel by means of a Lagrange multiplier - an implementation which involves computing the LDL^T factorization of an $(m+1)\times(m+1)$ dense matrix. When it was first considered to use OLA for 2D helioseismic inversions with datasets from the SOHO and GONG projects comprising some 10^5–10^6 frequency splittings, it soon became clear that naively applying this implementation, which requires $O(m^3)$ operations and $O(m^2)$ words of storage, would be prohibitively expensive (e.g. [173, 47]). One suggestion by Christensen-Dalsgaard and Thompson [49] to reduce the amount of data by using the SVD to extract the significant components would partly solve this problem. However, the normal equations approach has another disadvantage since it is necessary to recompute the LDL^T factorization, whenever the regularization parameter is changed. In the paper [129] found in Appendix B, we developed an algorithm for dealing with these issues, and compared it to previous implementations of OLA methods. The original paper only treated the SOLA case, but in this chapter we provide a general algorithmic framework for creating efficient implementations of both MOLA and SOLA, and describe an implementations based on these algorithms.

In Section 4.1 we formulate a generalized OLA method which includes SOLA and MOLA as special cases, and show how it may be implemented using numerical techniques similar to those presented Appendices B and C. This includes a description of the implementation of each of the computational steps involved. We then go on to describe how to compute various heuristics including averaging kernels, solution estimates, formal errors and the error correlation function for the solution. The central steps in the algorithm consists of solving an ordinary regularized least-squares problem. As explained in Appendix B, this regularized least-squares problem can be solved in a variety of ways, and we have experimented with a number of both direct and iterative algorithms for implementing this central and most time-consuming step:

1. A direct algorithm that computes a Householder bidiagonalization that solves the resulting bidiagonal system by means of Eldén's algorithm [70].

2. An iterative algorithm that solve the reduced system using the LSQR algorithm by Paige and Saunders [146, 147].

3. A hybrid algorithm based on Lanczos bidiagonalization which is similar to the algorithms by Björck [24] and O'Leary and Simmons [144].

As reported in Appendix C, it is the hybrid algorithm which is by far the most computationally efficient for 2D helioseismic inversions, when the block Kronecker product structure of the matrices is taken into account. Therefore, and because the first two algorithms are described in great detail in Appendix B, our presentation below goes through the main computational steps of the hybrid method.

The main computational steps of the general algorithm have been implemented to construct a number of SOLA and MOLA codes for 2D helioseismic inversion. These codes have been implemented on a Silicon Graphics Power Challenge, which is a shared memory multiprocessor, see Section 3.3 for a discussion of the achieved performance.

4.1 The general OLA algorithm

We saw in Section 2.1.3 that the central idea in all mollifier methods, including Backus-Gilbert, is to compute pointwise estimates $f_{\mathrm{est}}(x_0)$ of the solution f to the Fredholm equation

$$b_i = \int_0^1 K_i(x)\, f(x)\, \mathrm{d}x + \epsilon_i \ ,$$

as a linear combination of the data b_i

$$f_{\mathrm{est}}(x_0) = \sum_{i=1}^{m} c_i(x_0)\, b_i$$

The goal is to choose the set of inversion coefficients $c_i(x_0)$ such that the corresponding point-spread function or *averaging kernel*

$$\mathcal{K}(x_0, x_0) \equiv \sum_{i=1}^{m} c_i(x_0)\, K_i(x)$$

has an integral of unity and is peaked around x_0. In this case $f_{\mathrm{est}}(x_0)$ can be said, in the absence of noise, to be a weighted average of the true f around x_0, see e.g. [10, 136] and references therein.

The computational problem in the original Backus-Gilbert method and its variations can be summarized in a single constrained minimization problem: Given a weighting $\mathcal{J}(x, x_0)$ and a target function $\mathcal{T}(x, x_0)$ find the solution estimate where the corresponding averaging kernel minimizes the value of the functional

$$\min \int_0^1 \mathcal{J}(x, x_0)^2 \left[\sum_{i=1}^{m} K_i(x)\, c_i(x_0) - \mathcal{T}(x, x_0) \right]^2 \mathrm{d}x + \lambda^2 \sum_{i,j=1}^{m} c_i(x_0)\, \mathbf{E}_{ij}\, c_j(x_0) \ . \tag{4.1}$$

subject to

$$\sum_{i=1}^{m} c_i(x_0) \int_0^1 K_i(x)\, \mathrm{d}x = 1 \ . \tag{4.2}$$

To simplify notation we have collapsed the spatial variables into a single argument x. The familiar expression for the 2-dimensional OLA used in helioseismology is obtained by simply substituting

$$x \equiv (r, \theta) \ , \quad \mathrm{d}x \equiv r\, \mathrm{d}r\, \mathrm{d}\theta$$

where r is fractional solar radius and θ is co-latitude, and by replacing the integral with a double integral over the unit circle in the first quadrant

$$\{(r, \theta)\,|\, 0 \le r \le 1,\ 0 \le \theta \le \pi/2\} \ .$$

The second term in Equation (4.1) is a measure of the variance of the error in f_{est}, and is introduced to control the size of the errors that are propagated from the measurements to the solution. The regularization term is not present in the original paper by Backus and Gilbert [10], but is important in practical inversions since the discretized systems is usually very ill-conditioned. Also, it allows for much higher flexibility in trading off between error propagation and resolution, see e.g. [48, 159]. The two variants MOLA (the original Backus-Gilbert method) and SOLA are contained as special cases in the formulation above; choosing the target function $\mathcal{T}(x, x_0)$ equal to zero yields the MOLA method, while choosing the weight function $\mathcal{J}(x, x_0)$ equal to unity gives the SOLA variant. We are not aware of any attempts to use a combination of a weighting *and* a target function. This could be used for specifying a target function and penalize non-local contributions to f_{est} by weighting the integrated deviations from the target with the distance from the target location x_0.

One of the main arguments used in favor of OLA methods is that the ill-posed part of the computation, the determination of $c_i(x_0)$, $i = 1, \ldots, m$ is carried out on exactly given quantities, namely on $\mathcal{J}(x)$, $\mathcal{T}(x)$ and $K_i(x)$. In practice, however, the linear systems arising from the discretization of Equation (4.1) are usually so ill-conditioned that the regularization term is necessary to avoid numerical instability and to keep the solution from being dominated by round-off. In cases where the kernels $K_i(x)$ are not known exactly, the error in the model can be incorporated using algorithms based on total least squares (see [74, 80]). In helioseismic rotational inversion the radial structure of the kernels is only known approximately from a solar model. However, it is known from the solution of inversions of the solar structure that the errors in kernels are very small (a fraction of a percent, see e.g. [43]) compared to the errors in the frequency splittings, and therefore using total least squares would hardly provide any additional information, although this might deserve closer examination.

In the following we will describe the individual steps in an algorithm for computing the generalized OLA inversion.

4.2 Discretization

To allow a numerical treatment of Equation (4.1) we need to discretize it. The two main classes of discretization methods are the *quadrature method* where the equation is discretized by approximating the integrals involved using some quadrature rule, and the *Galerkin method* where the functions are expanded on a chosen set of basis functions. In this presentation we restrict ourselves to consider the quadrature method, since that is the one used in the helioseismic inversions. However, the algorithms below are quite general and can be used for problems discretized using the Galerkin method.

Without loss of generality we assume that the integrals are approximated using a quadrature rule with points $x_1, x_2, \ldots, x_n$ and weights $w_1, w_2, \ldots, w_n$

$$\int_0^1 f(x)\,\mathrm{d}x \approx \sum_{j=1}^{n} w_j f(x_j) \ .$$

We now define the $m \times n$ matrix $\mathbf{K}$ with elements

$$(\mathbf{K})_{ij} = K_i(x_j) \ , \qquad i = 1, \ldots, m \ , \quad j = 1, \ldots, n \ ,$$

in which the rows contain samples of the kernels. In order to include the weighting function and the integrals in a convenient way we introduce the diagonal $n \times n$ matrices $\mathbf{J}(x_0)$ and $\mathbf{W}$ with elements

$$(\mathbf{J}(x_0))_{jj} = \mathcal{J}(x_j, x_0) \ , \quad (\mathbf{W})_{jj} = w_j \ , \qquad j = 1, \ldots, n \ .$$

Also, we introduce the n-vectors $\mathbf{t}(x_0)$ and $\mathbf{e}$ with elements

$$(\mathbf{t}(x_0))_j = \mathcal{T}(x_j, x_0) \ , \quad (\mathbf{e})_j = 1 \ , \qquad j = 1, \ldots, n$$

and finally the m-vectors $\mathbf{c}(x_0)$ and $\mathbf{b}$ with elements

$$(\mathbf{c}(x_0))_i = c_i(x_0) \ , \quad (\mathbf{b})_i = b_i \ , \qquad i = 1, \ldots, m \ ,$$

which contain the inversion coefficients and the measurements respectively. To simplify notation a bit we shall ignore the dependence on x_0 in the following, whenever possible.

Now the discretized version of the minimization problem in Equation (4.1) along with the constraint in Equation (4.2) can be written as the equality constrained linear least-squares problem

$$\begin{aligned} &\min_{\mathbf{c} \in \mathbf{R}^m} \|\mathbf{W}^{1/2}\,\mathbf{J}\,\left(\mathbf{K}^T\,\mathbf{c} - \mathbf{t}\right)\|_2^2 + \lambda^2\,\mathbf{c}^T\,\mathbf{E}\,\mathbf{c} \\ &\text{subject to } \mathbf{e}^T \mathbf{W} \mathbf{K}^T \mathbf{c} = 1 \ , \end{aligned} \tag{4.3}$$

which may be verified by writing out the first term

$$\begin{aligned} \|\mathbf{W}^{1/2}\,\mathbf{J}\,\left(\mathbf{K}^T\,\mathbf{c} - \mathbf{t}\right)\|_2^2 &= \left(\mathbf{K}^T\,\mathbf{c} - \mathbf{t}\right)^T\,\mathbf{J}\,\mathbf{W}\,\mathbf{J}\,\left(\mathbf{K}^T\,\mathbf{c} - \mathbf{t}\right) \\ &= \sum_{j=1}^{n} w_j\,\mathcal{J}(x_j, x_0)^2 \left[\sum_{i=1}^{m} K_i(x_j)\,c_i(x_0) - \mathcal{T}(x_j, x_0)\right]^2 \\ &\approx \int_0^1 \mathcal{J}(x, x_0)^2 \left[\sum_{i=1}^{m} K_i(x)\,c_i(x_0) - \mathcal{T}(x, x_0)\right]^2 \mathrm{d}x \ , \end{aligned}$$

and the expression on the left-hand side of the constraint

$$\begin{aligned} \mathbf{e}^T\mathbf{W}\mathbf{K}^T\mathbf{c} &= \sum_{j=1}^{n} w_j \sum_{i=1}^{m} K_i(x_j)\, c_i(x_0) \\ &\approx \int_0^1 \mathcal{K}(x,x_0)\, \mathrm{d}x \ . \end{aligned}$$

A convenient way to incorporate the statistical information in $\mathbf{E}$ is to use the transformation of variables defined by

$$\mathbf{x} = \mathbf{G}\,\mathbf{c} \ , \tag{4.4}$$

where $\mathbf{G}$ is the Cholesky factor of $\mathbf{E}$ such that of $\mathbf{E} = \mathbf{G}^T\mathbf{G}$. This corresponds to working in terms of the transformed data $\mathbf{G}^{-T}\mathbf{b}$, which have an error-covariance matrix equal to $\mathbf{G}^{-T}\mathbf{E}\mathbf{G} = \mathbf{I}$. After the change of variables Equation (4.3) can be written in the generic form

$$\min_{\mathbf{x}\in\mathbf{R}^m} \|\mathbf{A}\,\mathbf{x} - \mathbf{b}\|_2^2 + \lambda^2\|\mathbf{x}\|_2^2 \quad \text{subject to} \quad \mathbf{z}^T\,\mathbf{x} = 1 \ , \tag{4.5}$$

where $\mathbf{A} \equiv \mathbf{W}^{1/2}\,\mathbf{J}\,\mathbf{K}^T\,\mathbf{G}^{-1} \in \mathbb{R}^{n\times m}$, $\mathbf{b} \equiv \mathbf{W}^{1/2}\,\mathbf{J}\,\mathbf{t} \in \mathbb{R}^n$ and $\mathbf{z} \equiv \mathbf{G}^{-T}\,\mathbf{K}\,\mathbf{W}\,\mathbf{e} \in \mathbb{R}^m$.

4.3 Elimination of the constraint

To solve the equality constrained least-squares (LSE) problem in Equation (4.5) we first use the so-called *nullspace method* (see, e.g., [25, pp. 189–191]) to transform it into an *unconstrained* problem. However, Expression (4.5) contains the extra regularization term, which means that it is not quite a standard LSE problem. Before we proceed, we would like to argue that even with the regularization term the nullspace method will always compute a meaningful solution. To do this we write Equation (4.5) in the form

$$\min_{\mathbf{x}\in\mathbf{R}^m} \left\| \begin{pmatrix} \mathbf{A} \\ \lambda\mathbf{I} \end{pmatrix} \mathbf{x} - \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix} \right\|_2 \quad \text{subject to} \quad \mathbf{z}^T\,\mathbf{x} = 1 \ .$$

and consider the two cases:

$\lambda \neq 0$: In this case there exists a unique solution solution since

$$\operatorname{rank}(\mathbf{z}^T) = 1 \quad \text{and} \quad \operatorname{null}(\mathbf{z}^T) \cap \operatorname{null}\begin{pmatrix} \mathbf{A} \\ \lambda\mathbf{I} \end{pmatrix} = \{\mathbf{0}\} \ ,$$

i.e. the nullspaces of $(\mathbf{A}^T, \lambda\mathbf{I})^T$ and $\mathbf{z}^T$ intersect only trivially.

$\lambda = 0$: The solution is no longer unique because (since we assume $m > n$)

$$\{\mathbf{0}\} \neq \operatorname{null}(\mathbf{A}) \subset \operatorname{null}(\mathbf{e}^T\,\mathbf{W}^{1/2}\mathbf{J}^{-1}\mathbf{A}) = \operatorname{null}(\mathbf{z}^T) \ ,$$

and the intersection between the nullspaces is therefore nontrivial. This means that if $\mathbf{x}$ is a solution to Equation (4.5) then so is any vector $\mathbf{x} + \mathbf{n}$, where $\mathbf{n}$ is an arbitrary vector from $\{\operatorname{null}(\mathbf{z}^T) \cap \operatorname{null}(\mathbf{A})\}$. In this situation the nullspace method will compute the solution with the smallest norm, which is also the “best” solution to the problem since it corresponds to the set of inversion coefficients $c_i(x_0)$ where the error in the corresponding estimate $f_{\text{est}}(x_0)$ has the smallest variance.

The nullspace method has three stages, which will be described in this and the following two paragraphs:

1. Derive a lower-dimensional unconstrained least squares problem from the given problem.
2. Compute the solution to the lower-dimensional problem.
3. Transform this solution back to the original setting.

The first stage consists in computing a QR factorization of the coefficient matrix in the constraint. In our case this is simply the row-vector $\mathbf{z}^T$, and the QR factorization can be computed using a single *Householder transformation* (e.g. [83, Section 5.1]):

$$\mathbf{H}^T\mathbf{z} = \begin{pmatrix} \alpha^{-1} \\ \mathbf{0} \end{pmatrix}$$

where $\alpha^{-1} = \mathrm{sign}(\mathbf{z}(1))\|\mathbf{z}\|_2$. Let $\mathbf{H}$ be partitioned as

$$\mathbf{H} = \begin{array}{c} \left[\begin{array}{cc} \mathbf{h}_1 & \mathbf{H}_2 \end{array} \right] m \\ \begin{array}{cc} 1 & \quad m-1 \end{array} \end{array} ,$$

then $\mathrm{null}(\mathbf{z}^T) = \mathrm{range}(\mathbf{H}_2)$ and we may write any solution to Equation (4.5) on the form

$$\mathbf{x} = \alpha\,\mathbf{h}_1 + \mathbf{H}_2\,\bar{\mathbf{x}}\ , \quad \bar{\mathbf{x}} \in \mathbb{R}^{m-1}\ , \tag{4.6}$$

since from the definition of $\mathbf{H}$ it follows that

$$\mathbf{z}^T\mathbf{x} = \alpha\,\mathbf{z}^T\,\mathbf{h}_1 + \mathbf{z}^T\,\mathbf{H}_2\,\bar{\mathbf{x}} = 1 + 0\ ,$$

as required for $\mathbf{x}$ to satisfy the constraint. Hence we may simplify Equation (4.5) as follows:

$$\begin{aligned} & \min_{\mathbf{x}\in\mathbb{R}^m} \|\mathbf{A}\,\mathbf{x} - \mathbf{b}\|_2^2 + \lambda^2\|\mathbf{x}\|_2^2 \quad \text{subject to} \quad \mathbf{z}^T\,\mathbf{x} = 1 \\ = \quad & \min_{\bar{\mathbf{x}}\in\mathbb{R}^{m-1}} \|\mathbf{A}\,(\alpha\,\mathbf{h}_1 + \mathbf{H}_2\,\bar{\mathbf{x}}) - \mathbf{b}\|_2^2 + \lambda^2(\|\bar{\mathbf{x}}\|_2^2 + \alpha^2) \\ = \quad & \min_{\bar{\mathbf{x}}\in\mathbb{R}^{m-1}} \|\bar{\mathbf{A}}\bar{\mathbf{x}} - \bar{\mathbf{b}}\|_2^2 + \lambda^2\|\bar{\mathbf{x}}\|_2^2 + (\lambda\alpha)^2\ , \end{aligned} \tag{4.7}$$

where we have defined $\bar{\mathbf{A}} \equiv \mathbf{A}\mathbf{H}_2$ and $\bar{\mathbf{b}} \equiv (\mathbf{b} - \alpha\,\mathbf{A}\,\mathbf{h}_1)$. We have now reduced the original problem to solving the ordinary (i.e. *unconstrained*) regularized least-squares problem in Equation (4.7). We emphasize that calculations involving the Householder transformation $\mathbf{H} = \mathbf{I} + \gamma\mathbf{v}\mathbf{v}^T$ can be implemented efficiently by keeping it in factored form, which means only storing the scalar $\gamma = 1/[\|\mathbf{z}\|_2\,(\|\mathbf{z}\|_2 + |\mathbf{z}(1)|)]$ and the Householder vector $\mathbf{v} = \mathbf{z} + \mathrm{sign}(\mathbf{z}(1))\|\mathbf{z}\|_2\mathbf{e}_1$.

4.4 Solution by Lanczos bidiagonalization

As discussed in the introduction, Equation (4.7) can be solved in a number of ways. Here we describe an algorithm based on the Lanczos bidiagonalization algorithm (LBD) due to Golub and Kahan [81]. Here we use the variant from [146] known as *Bidiag1*, which is the most stable implementation for solving least squares problems. For the rectangular $n \times (m-1)$ matrix $\bar{\mathbf{A}}$ the Lanczos bidiagonalization computes a sequence of *Lanczos vectors* $\mathbf{u}_j \in \mathbb{R}^n$ and $\mathbf{v}_j \in \mathbb{R}^{m-1}$ and scalars α_j and β_j for $j = 1, 2, \ldots, k$ as follows:

1. Choose $\bar{\mathbf{b}}$ as starting vector:
   $\beta_1 = \|\bar{\mathbf{b}}\|_2$, $\mathbf{u}_1 = \bar{\mathbf{b}}/\beta_1$ and let $\mathbf{v}_0 \equiv 0$

2. **for** $j = 1, 2, \ldots, k$ **do**
   $\mathbf{r}_j = \bar{\mathbf{A}}^T \mathbf{u}_j - \beta_j \mathbf{v}_{j-1}$
   $\alpha_j = \|\mathbf{r}_j\|_2$
   $\mathbf{v}_j = \mathbf{r}_j / \alpha_j$
   $\mathbf{p}_j = \bar{\mathbf{A}}\, \mathbf{v}_j - \alpha_j \mathbf{u}_j$
   $\beta_{j+1} = \|\mathbf{p}_j\|_2$
   $\mathbf{u}_{j+1} = \mathbf{p}_j / \beta_{j+1}$
   end

After k steps the $(k+1) \times k$ lower bidiagonal matrix

$$\mathbf{B}_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{pmatrix} . \tag{4.8}$$

has been generated. In exact arithmetic the Lanczos vectors form an orthonormal set such that

$$\begin{aligned} \mathbf{U}_{k+1} &= (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{k+1}) \in \mathbb{R}^{n \times (k+1)} \ , \quad \mathbf{U}_{k+1}^T \mathbf{U}_{k+1} = \mathbf{I}_{k+1} \ , \\ \mathbf{V}_k &= (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k) \in \mathbb{R}^{(m-1) \times k} \ , \qquad \mathbf{V}_k^T \mathbf{V}_k = \mathbf{I}_k \ , \end{aligned} \tag{4.9}$$

where $\mathbf{I}_k$ is the $k \times k$ identity matrix. By construction the columns of $\mathbf{U}_{k+1}$ and $\mathbf{V}_k$ satisfy a set of recurrences which we can write in a compact matrix form as

$$\begin{aligned} &\mathbf{U}_{k+1}(\beta_1 \mathbf{e}_1) = \bar{\mathbf{b}} \ , \\ \bar{\mathbf{A}}\mathbf{V}_k = \mathbf{U}_{k+1}\mathbf{B}_k \ , \qquad &\bar{\mathbf{A}}^T \mathbf{U}_{k+1} = \mathbf{V}_k \mathbf{B}_k^T + \alpha_{k+1} \mathbf{v}_{k+1} \mathbf{e}_{k+1}^T \ . \end{aligned} \tag{4.10}$$

We now seek an approximate solution $\bar{\mathbf{x}}_k$ to Equation (4.7) in span$(\mathbf{V}_k)$, which can be written in the form

$$\bar{\mathbf{x}}_k = \mathbf{V}_k \mathbf{y}_k \ , \qquad \mathbf{y}_k \in \mathbb{R}^k \ . \tag{4.11}$$

By substituting Equations (4.10) and (4.11) in Equation (4.7) and use the orthogonality of $\mathbf{U}_{k+1}$ and $\mathbf{V}_k$ we find that

$$\begin{aligned} \min_{\bar{\mathbf{x}} \in \mathrm{span}(\mathbf{V}_k)} \left\| \begin{pmatrix} \bar{\mathbf{A}} \\ \lambda \mathbf{I} \end{pmatrix} \bar{\mathbf{x}} - \begin{pmatrix} \bar{\mathbf{b}} \\ \mathbf{0} \end{pmatrix} \right\|_2^2 &= \min_{\mathbf{y} \in \mathbf{R}^k} \left\| \begin{pmatrix} \bar{\mathbf{A}}\mathbf{V}_k \\ \lambda \mathbf{V}_k \end{pmatrix} \mathbf{y} - \begin{pmatrix} \bar{\mathbf{b}} \\ \mathbf{0} \end{pmatrix} \right\|_2^2 \\ &= \min_{\mathbf{y} \in \mathbf{R}^k} \left\| \begin{pmatrix} \mathbf{U}_{k+1}\mathbf{B}_k \\ \lambda \mathbf{V}_k \end{pmatrix} \mathbf{y} - \begin{pmatrix} \bar{\mathbf{b}} \\ \mathbf{0} \end{pmatrix} \right\|_2^2 \\ &= \min_{\mathbf{y} \in \mathbf{R}^k} \left\| \begin{pmatrix} \mathbf{B}_k \\ \lambda \mathbf{I}_k \end{pmatrix} \mathbf{y} - \begin{pmatrix} \beta_1 \mathbf{e}_1 \\ \mathbf{0} \end{pmatrix} \right\|_2^2 \end{aligned} \tag{4.12}$$

This means that $\|\bar{\mathbf{A}}\bar{\mathbf{x}} - \bar{\mathbf{b}}\|_2^2 + \lambda^2 \|\bar{\mathbf{x}}\|_2^2$ is minimized over span$(\mathbf{V}_k)$ by choosing $\mathbf{y}_k$ as the solution to the least squares problem in Equation (4.12). It is important that Equation (4.12) can be solved efficiently because it is often necessary to solve it for many different values of the regularization parameter λ in order to find the optimal one. For this purpose we use an implementation of Eldén's algorithm [70] that requires only $O(26k)$ flops. Computational details may be found in [70] and Appendix B. We have made a small addition to Eldén's algorithm such that the residual norm

$$\delta = \left\| \begin{pmatrix} \mathbf{B}_k \\ \lambda \mathbf{I}_k \end{pmatrix} \mathbf{y}_k - \begin{pmatrix} \beta_1 \mathbf{e}_1 \\ \mathbf{0} \end{pmatrix} \right\|_2$$

is accumulated during the algorithm. This is useful since many parameter choice methods, such as the L-curve criterion (see [106]) need to know the value of the residual norm. The cost of $3k$ flops for accumulating the residual norm is cheaper than computing it explicitly after the solution has been found ($6k$ flops).

4.5 Back transformation and calculation of heuristics

The bulk of the computation lies in the reduction to bidiagonal form, using either the Lanczos algorithm just described or the Householder bidiagonalization which we used in Appendix B. Once $\mathbf{y}_k$ has been determined, it is (at least in principle) simple to transform the solution back to the original setting using Eqs. (4.11), (4.6) and the definition of $\mathbf{x}$ in Equation (4.4) to get the inversion coefficients

$$\mathbf{c} = \mathbf{G}^{-1}\mathbf{x} = \mathbf{G}^{-1}\mathbf{H} \begin{pmatrix} \alpha \\ \mathbf{V}_k \mathbf{y}_k \end{pmatrix} \tag{4.13}$$

Once $\mathbf{c}$ is known all the quantities of interest, e.g., solution estimate, error variance and the averaging kernel can be computed from their definitions in Section 4.1. However, unless $\mathbf{c}$ is of specific interest in itself, these quantities can be computed much more efficiently without forming $\mathbf{c}$. It turns out that most of these quantities can be expressed, more or less directly, in terms of the small k-vector $\mathbf{y}_k$. Since k is much smaller than n and m, this represents large savings in storage and work (here we are reaping the fruit of our efforts to use orthogonal transformations throughout the algorithm). This may seem insignificant, since the back-transformation is very cheap compared with the bidiagonalization, but in practice the bidiagonalization is typically computed only once and stored on disk, while the back-transformation is repeated over and over e.g. to experiment with different target functions and different values of λ. In SOLA the back-transformation can be almost as costly as the reduction to bidiagonal form if implemented naively, since it must be done for many right-hand sides, corresponding to different target positions. With the fast back-transformation, the response time of the implementation described below is in fact low enough that the algorithm, in combination with appropriate visualization software, can be used as an *interactive tool* to examine the effect of varying the regularization. Another important point is that by computing the transformed data-vector

$$\bar{\mathbf{b}} \equiv \begin{pmatrix} \bar{b}_1 \\ \bar{\mathbf{b}}_2 \end{pmatrix}, \quad \bar{b}_1 = \mathbf{h}_1^T \mathbf{G}^{-T} \mathbf{b}, \quad \bar{\mathbf{b}}_2 = \mathbf{V}_k^T \mathbf{H}_2^T \mathbf{G}^{-T} \mathbf{b}, \tag{4.14}$$

once and for all (notice that $\bar{\mathbf{b}}$ is independent of x_0), the large $(m-1) \times k$ matrix $\mathbf{V}_k$ can be completely eliminated from the back-transformation, and does not have to be stored on disk. The transformed vector $\bar{\mathbf{b}}$ is computing in the following way: After $\mathbf{H}$ and $\mathbf{G}$ have been determined, $\bar{b}_1$ is calculated and the vector $\mathbf{H}_2^T\mathbf{G}^{-T}\mathbf{b}$ is computed and parsed to the subroutine that performs the LBD. The entries in $\bar{\mathbf{d}}_2$ are then calculated by forming the inner products of $\mathbf{H}_2^T\mathbf{G}^{-T}\mathbf{d}$ and the columns in $\mathbf{V}_k$ as they are generated. After the LBD $\mathbf{V}_k$ is never referenced again, and the memory allocated for storing $\mathbf{V}_k$ can either be returned to the operating system or used for other purposes. In our SOLA implementation, e.g., part of the $\mathbf{V}_k$ array is reused for setting up the discretized target function(s).

Below we have listed the formulas on which our implementation of the back-transformation is built. Their derivations are straightforward and essentially consist in substituting the expression for $\mathbf{c}$ in terms of $\mathbf{x}$ and $\bar{\mathbf{x}}_k$, equations (4.13), (4.11) and (4.6), in the definition of the relevant quantity and subsequently using the orthogonality of $\mathbf{H}$, $\mathbf{U}_k$ and $\mathbf{V}_k$. The final step in the derivation of the formula for the averaging kernel uses the relations in Equation (4.10) describing the Lanczos bidiagonalization to eliminate $\mathbf{V}_k$.

Solution estimate:

$$f_{\text{est}}(x_0) = \mathbf{c}^T\mathbf{b} = \begin{pmatrix} \alpha \\ \mathbf{V}_k\mathbf{y}_k \end{pmatrix}^T \mathbf{H}^T\mathbf{G}^{-T}\mathbf{b} = \alpha\bar{b}_1 + \mathbf{y}_k^T\bar{\mathbf{b}}_2 \tag{4.15}$$

Error variance:

$$\sigma(x_0)^2 = \mathbf{c}^T\,\mathbf{E}\,\mathbf{c} = \|\mathbf{x}\|_2^2 = \alpha^2 + \|\bar{\mathbf{x}}_k\|_2^2 = \alpha^2 + \|\mathbf{y}_k\|_2^2 \tag{4.16}$$

Averaging kernel:

$$\begin{aligned} \mathcal{K}(x,x_0) \approx \mathbf{K}^T\,\mathbf{c} &= \mathbf{J}^{-1}\mathbf{W}^{-1/2}\mathbf{A}(\alpha\mathbf{h}_1 + \mathbf{H}_2\bar{\mathbf{x}}_k) \\ &= \mathbf{J}^{-1}\mathbf{W}^{-1/2}(\alpha\mathbf{A}\mathbf{h}_1 + \bar{\mathbf{A}}\mathbf{V}_k\mathbf{y}_k) \\ &= \mathbf{J}^{-1}\mathbf{W}^{-1/2}(\alpha\mathbf{A}\mathbf{h}_1 + \mathbf{U}_{k+1}\mathbf{B}_k\mathbf{y}_k) \end{aligned} \tag{4.17}$$

Residual norm = Spread (MOLA) or deviation of the averaging kernel from the target (SOLA):

$$\begin{aligned} \int_0^1 \mathcal{J}(x,x_0)^2 \left[\sum_{i=1}^m K_i(x)\,c_i(x_0) - \mathcal{T}(x,x_0)\right]^2 \mathrm{d}x &\approx \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 \\ &= \|\bar{\mathbf{A}}\bar{\mathbf{x}}_k - \bar{\mathbf{b}}\|_2^2 \\ &= \|\mathbf{B}_k\mathbf{y}_k - \beta_1\mathbf{e}_1\|_2^2 \\ &= \delta^2 - \lambda^2\|\mathbf{y}_k\|_2^2 \end{aligned} \tag{4.18}$$

Another important heuristic is the normalized correlation function $C(x_0,x_1)$ for the solution, which describes the correlation of the errors in the solution estimates $f_{\text{est}}(x_0)$ and $f_{\text{est}}(x_1)$. The correlation function is defined as

$$C(x_0,x_1) = \frac{E\left[\left(\sum c_i(x_0)\epsilon_i\right)\left(\sum c_j(x_1)\epsilon_j\right)\right]}{\left\{E\left[\left(\sum c_i(x_0)\epsilon_i\right)^2\right] E\left[\left(\sum c_j(x_1)\epsilon_j\right)^2\right]\right\}^{1/2}}\,, \tag{4.19}$$

where $E[\ldots]$ denotes expectation value. The calculation of $C(x_0,x_1)$ requires a few comments. For unbiased Gaussian errors ϵ_i with error-covariance matrix $\mathbf{E}$ it is straightforward to show that

$$C(x_0,x_1) = \frac{\sum_{ij} c_i(x_0)\,\mathbf{E}_{ij}\,c_j(x_1)}{\left\{\left[\sum_{ij} c_i(x_0)\,\mathbf{E}_{ij}\,c_j(x_0)\right]\left[\sum_{ij} c_i(x_1)\,\mathbf{E}_{ij}\,c_j(x_1)\right]\right\}^{1/2}}$$

and using the same tricks as in the derivations above this may be reduced to

$$
\begin{aligned}
C(x_0, x_1) &= \frac{\mathbf{c}^T(x_0)\,\mathbf{E}\,\mathbf{c}(x_1)}{\left[\mathbf{c}(x_0)^T\,\mathbf{E}\,\mathbf{c}(x_0)\right]^{1/2}\left[\mathbf{c}(x_1)^T\,\mathbf{E}\,\mathbf{c}(x_1)\right]^{1/2}} \\
&= \frac{\mathbf{x}(x_0)^T\,\mathbf{x}(x_1)}{\sigma(x_0)\sigma(x_1)} \\
&= \frac{\alpha^2 + \bar{\mathbf{x}}_k(x_0)^T\,\bar{\mathbf{x}}_k(x_1)}{\sigma(x_0)\sigma(x_1)} \qquad (4.20) \\
&= \frac{\alpha^2 + \mathbf{y}_k(x_0)^T\,\mathbf{y}_k(x_1)}{\sigma(x_0)\sigma(x_1)} \qquad (4.21)
\end{aligned}
$$

Chapter 5

Discussion and conclusion

Jeg sænker min kikkert og ser
at alle de adskilte dele
hænger sammen som et hele
der kan blive til meget mer.
Jeg synger mens solkuglen bunder:
Gid jorden aldrig går under!

— Benny Andersen, from "Oven visse vande"

In this final chapter I will try to summarize the results described in the previous chapters and the papers found in the appendices of this thesis. I have decided to discuss them more or less in the chronological order in which they were produced, partly because it illustrates the evolution (improvement, hopefully) of my own understanding of the various topics, but also because I find it interesting to try and trace how one result lead to the next, in a (not always logical or obvious) chain of small discoveries.

5.1 Discussion

As mentioned in the introduction, I started out by treating the helioseismic inversions as general ill-conditioned least-squares problems, and most of my efforts were concentrated on developing efficient parallel implementations of standard algorithms as described by, e.g., Hanke and Hansen [97]. During this period I investigated, in collaboration with Per Christian Hansen, various direct and iterative regularization methods, the latter especially with the aim to find efficient parallel implementations of Lanczos bidiagonalization (LBD). A motivating factor was that it is quite typical when solving large ill-posed problems, e.g., from inverse helioseismology that the number of iterations needed in a Lanczos-based method is far less than the dimension of the problem (e.g. [202, 104, 95]) – a property which makes these Krylov subspace methods an interesting alternative to direct methods based on, e.g., SVD, bidiagonalization or the normal equations. This had been illustrated by the successful application of Lanczos and conjugate gradient based iterative regularization in related areas such as image reconstruction [98, 99] and signal analysis [71]. These problems also involve the numerical solution of ill-posed problems with large and sparse or structured (e.g. Toeplitz or Hankel) matrices, and indeed I hope that the methods described here will also be applicable in these areas.

Some of the experiments with LBD, including the restructured iteration presented in Appendix E, were aimed at improving the parallel performance of the LBD algorithm. One of the conclusions which can be drawn from this work is that when implementing iterative algorithms, where the central operations are usually Level-2 or Level-1 BLAS operation (matrix-vector or vector-vector), on a shared memory multi-processor like the Convex Exemplar, one cannot rely entirely on traditional performance measures such as the number of arithmetic operations (flops) and speedup. Instead, the performance degradation due to memory hierarchy transports, e.g. cache misses, plays a role that is often more important than parallel efficiency. To do this properly, however, requires a certain degree of insight into the architectural peculiarities of the specific machine. In our case, reducing the number of cache misses and obtaining high speedup rates were mutually conflicting demands. As an example, the super-linear speedup obtained with the simple parallel implementation in Appendix E, was primarily an indication of its poor cache performance. A more general conclusions from these experiments with iterative regularization algorithms on parallel computers is that unless the sequential program takes full advantage of the internal parallelism in the CPU and is not slowed down by memory hierarchy transports, the efforts are usually better spent trying to improve cache performance use of the pipelined floating point units of the CPU (*not to mention trying to find a better algorithm!*), than to try and move the program to a parallel computer. As Boeing's Ken Neves is quoted to have said (in C. M. Pancake's excellent review of parallel computing [148]): "Nobody wants parallelism. What we want is performance". There are a number of additional reasons for this:

- Parallelizing is very time consuming. Writing, debugging and tuning a parallel program takes considerably longer time than writing the corresponding sequential program, and using an automatic parallelizing compiler is simply not a realistic (or at least efficient) option for large real-world applications. Such compilers can do a fairly good job using advanced semantic techniques from program analysis, but it is my experience that for realistic parallel programs, where both FPU, cache and parallel performance must be considered simultaneously, so many assumptions must be made about the program to get a semantically correct translation, that an automatic parallelization cannot compete with a carefully written parallel program. I got a first-hand impression of this, when I wrote part of a parallelizing C compiler for the SGI Power Challenge; I was responsible for designing and implementing a runtime system for coordinating and synchronizing the parallel threads, quite similar to the one used by SGI's own compilers (see [131]).

- It is not unusual that the most efficient algorithm for solving a given problem on a sequential computer is not the one best suited for parallelization. A classic example is the QR algorithm for the symmetric tridiagonal eigenvalue problem, which is difficult to parallelize. For this reason, e.g., the ScaLAPACK library uses the embarrassingly parallel spectral bisection algorithm combined with inverse iterations for computing the eigenvectors. In the investigations for Appendix E we used a our own parallel implementation of a similar (embarrassingly parallel) spectral bisection algorithm for the SVD with success.

- The tools for serial code optimization and debugging, including optimizing compilers and profilers are generally good and easy to use. This is illustrated in Appendix F, where I report on the effect of tuning the spectral bisection algorithm mentioned above on a number of different architectures (see also [130] where we discuss the use of profiling tools on the SGI Power Challenge (in danish)). Similar tools are sometimes provided in some parallel computing environments, but even so, extracting the relevant information to optimize or even debug the communication, say, in a large parallel program is a much more complex task than for a sequential one.

This is not to say that no large-scale application have ever benefited from parallel computer, since there are obviously applications, e.g. large MHD simulations, that are so computationally intensive that they need massive computer power. Rather, I wish to make the point that for a given project one should carefully consider were to spend the finite human resources available, and as a rule, attempt to find a better algorithm and tuning existing sequential programs before trying parallelize a program with poor sequential performance.

Many of the ideas in Appendix E developed to reduce the amount of cache misses were in fact never used in the helioseismic inversions, since after I discovered the block Kronecker product formulation it became clear that the matrix-vector multiplication could be implemented entirely as Level-3 BLAS operations. In this respect the Kronecker formulation seems "almost too good to be true", in the sense that not only does it reduce the computational complexity of the problem, but it also leads to an implementations that performs well on high-performance machines with a memory hierarchy.

As should be quite obvious from this thesis, my interaction with the helioseismology community has been extremely inspiring. The first concrete result was the development of an efficient algorithm for the SOLA mollifier method, which had been reinvented and introduced to the helioseismology community by Pijpers and Thompson in 1992 [158] (the method goes back at least to the 1990 paper by Louis and Mass [136], but similar techniques have apparently been used in signal processing even earlier, see discussion in [136]). For helioseismic inversion it was demonstrated in [158, 159] that SOLA is an inversion method, which in terms of computational efficiency and resolution ability is comparable with other classes of inversion methods such as Tikhonov regularization. In its original implementation, however, it is inefficient when many regularization parameters must be tried, but as opposed to Tikhonov regularization, efficient implementations of SOLA and MOLA were not found in standard texts such as [97]. This lead us to develop the algorithms in Appendix B, which we were careful to formulate such that they could be implemented using standard "building blocks" from numerical linear algebra. The advantage of this approach is that these "building blocks" are found in highly vectorized and parallelized subroutines on today's supercomputers, thus ensuring good performance without the need for a tedious fine-tuning of the code. When used as an interactive tool, where the solution is recomputed for different values of the regularization parameter λ or for different target functions, the improvement over a implementation based on the normal equations is quite dramatic, even when using the algorithm based on Householder bidiagonalization. For sparse or structured matrices we formulated the fundamental iterative LSQR-based method, in which the amount of regularization can either be incorporated explicitly, or controlled by the number of iterations. The results of these investigations resulted in the papers in Appendices B and E, but more importantly formed the basis for developing the algorithms mentioned in Chapter 4.

The algorithms in Appendix B lay the foundation for many of the subsequent results, e.g. generalization to MOLA, and most importantly for incorporating the block Kronecker product structure described in Chapter 3 and Appendix C. After finishing the paper in Appendix B, I started to implement a version of the LSQR-based algorithm for helioseismic

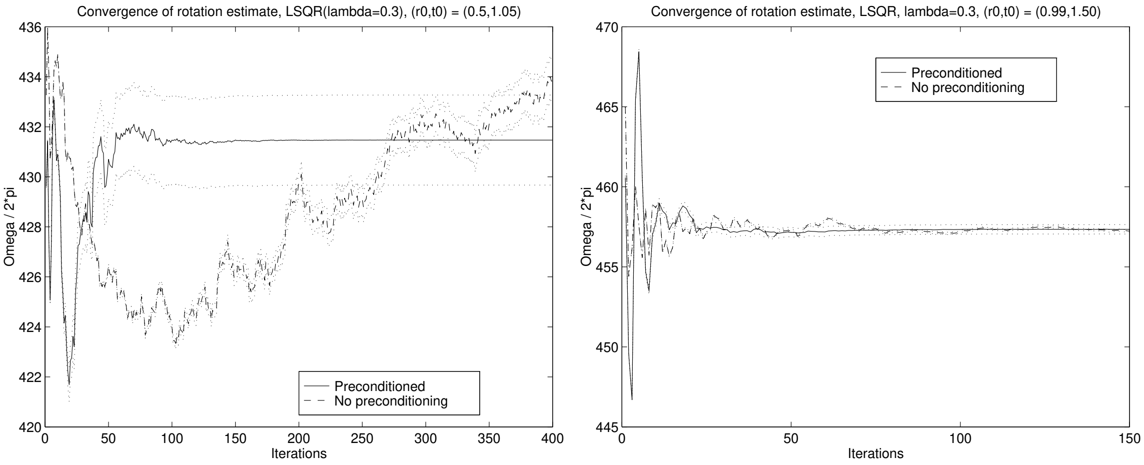


Figure 5.1: *Convergence of the rotation estimate in the LSQR-based iterative SOLA algorithm with and without diagonal preconditioning. Left panel shows the convergence history for target position* $r_0 = 0.5R$, $\theta_0 = 60°$. *Solid lines show the rotation estimate* $\Omega(r_0, \theta_0)$ *and dotted lines show* $\Omega(r_0, \theta_0) \pm \sigma(r_0, \theta_0)$, *where* $\sigma(r_0, \theta_0)$ *is the computed error estimate. Right panel shows the convergence history for target position* $r_0 = 0.99R$, $\theta_0 = 86°$. *In both panels the solution was computed with target widths* $(\Delta r, \Delta\theta) = (0.1, 0.3)$ *and* $\lambda = 0.3$.

inversion. Originally my idea was to exploit the special form of the kernels in a matrix-vector multiplication routine by simply storing discretization of the F_1^{nl}, G_1^{lm}, F_2^{nl} and G_2^{lm} and then generate elements of $\mathbf{K}$ "on the fly" from the definition

$$K_{nlm}(r, \theta) = F_1^{nl}(r)\, G_1^{lm}(\theta) \;+\; F_2^{nl}(r)\, G_2^{lm}(\theta) \tag{5.1}$$

as they where needed. This approach would give the same savings in storage as the Kronecker approach, but does not reduce the computational complexity. The idea to the Kronecker product formulation was partly inspired by the related so-called $\mathbb{R}^1 \otimes \mathbb{R}^1$ methods by Sekii [180] and Pijpers and Thompson [160], which replace the original 2d-problem by a slightly different inverse problem that separates in r and θ. In both methods this is achieved by restricting the inversion coefficients defined in Equation (2.12) to be on the form $\alpha_{nl}\beta_{lm}$. While the original method by Sekii essentially ignores the second term in Equation (5.1) to get a problem on pure Kronecker form, Pijpers and Thompson use a formulation based on the SOLA algorithm to solve an approximate problem where the second term in the kernels is also taken into account.

As discussed in Appendix C, the most efficient way of implementing the iterative SOLA algorithm, where the constrained least-squares problem must be solved for many right-hand sides, is to compute the Lanzcos bidiagonalization with reorthogonalization. This result was the conclusion of a number of experiments, some of them discussed in Appendix C, with different formulations based on LSQR, CGLS and Lanzcos bidiagonalization with full reorthogonalization. We also made a number of experiments with preconditioning LSQR and CGLS to speed up convergence, and used a simple diagonal preconditioner described, e.g., on page 283 in [25]. This corresponds to replacing the coefficient matrix $\bar{\mathbf{A}}$ in Equation (4.7) with $\bar{\mathbf{A}}\mathbf{S}^{-1}$, where

$$\mathbf{S} = \mathrm{diag}(d_1^{1/2}, d_2^{1/2}, \ldots, d_{m-1}^{1/2}), \;\; d_i = \|\bar{\mathbf{a}}_i\|_2^2 \;,$$

and $\bar{\mathbf{a}}_i$ is the ith column of $\bar{\mathbf{A}}$. As illustrated in Figure 5.1, the preconditioning speeds up the convergence quite dramatically, especially for target positions close to the pole and in the deep interior. Unfortunately, the preconditioning also changes the effect of the regularization, such that the weight with which a given mode contributes to the regularization term in Equation (4.1) is changed. The result, which can be seen most clearly in the left panel of Figure 5.1 is, for a given value of the residual norm (target misfit), to increase the random errors propagated to the solution. This lead us to abandon the preconditioning. The reduction in the number of iterations can be quite substantial, however, and we mention this in the hope that another preconditioning scheme can be found that does not change the regularization properties of the method in the way described. An idea that might work, is the algorithm discussed by Vogel in [203], where preconditioning is simply applied directly to the normal equations.

Our use of the Lanczos algorithm with full reorthogonalization for implementing the SOLA inversion method spawned another series of investigations, originally motivated by the desire to reduce the cost of the reorthogonalization. This lead us to consider the use of partial reorthogonalization (cf. [182]), either for computing approximations to the largest singular values and corresponding singular vectors, or to be used directly in the algorithm from Chapter 4. It turned out to be relatively straightforward to specialize Simon's original algorithm (PRO) to take advantage of the special structure of the recurrences in the Lanczos bidiagonalization. This resulted in the sparse SVD algorithms described in Appendix A. Through a series of numerical experiments in Matlab we found that the method is robust and often 2-3 times faster in terms of the number of floating point operations than current state-of-the-art algorithms based on the original PRO algorithm or the implicitly restarted Lanczos algorithms (cf. [33]). Equally important is the fact that this efficiency is obtained without sacrificing the accuracy of the computed singular values. I believe that, if implemented properly, this could turn out to be a competitive alternative to the ARPACK software, which is widely used and has recently been adopted as part of the ScaLAPACK library. These algorithms certainly deserve further investigation.

Part of Appendix A deals with Lanczos algorithms for sparse least squares problems. In particular, I describe algorithms for problems with multiple right-hand sides, which have received relatively little attention so far in the literature. The MCGLS algorithm, which is a generalization of the Ritz-Galerkin projection methods discussed in [36], could be a way of completely avoiding the reorthogonalization when solving linear-least problems with multiple right-hand sides. Whether it will be useful for solving discrete ill-posed problems is uncertain: Although the algorithm does seem to have a regularizing effect for the non-seed systems, their convergence behavior is somewhat erratic, and this algorithm certainly needs further investigation.

The algorithms described in this thesis have been used in a number of helioseismic investigations. Many of the inversion results obtained with the SOLA method are discussed in Appendix D. The implementations of the 2D SOLA method described elsewhere in this thesis was used by Christensen-Dalsgaard and myself for analyzing data for the joint MDI Internal Rotation Team paper (Schou et al. 1998 [175]), where the first inversions of data from the first 144 days of MDI observations was published. The results were obtained by a number of independent methods, and to test the reliability – strengths and weaknesses – of the various methods a so-called *Hare and Hounds* exercise was performed, where an artificial datasets corresponding to a rotation law unknown to the inverters was generated by A. G. Kosovichev (the "Hare") and the corresponding splitting coefficient, with added Gaussian errors, were

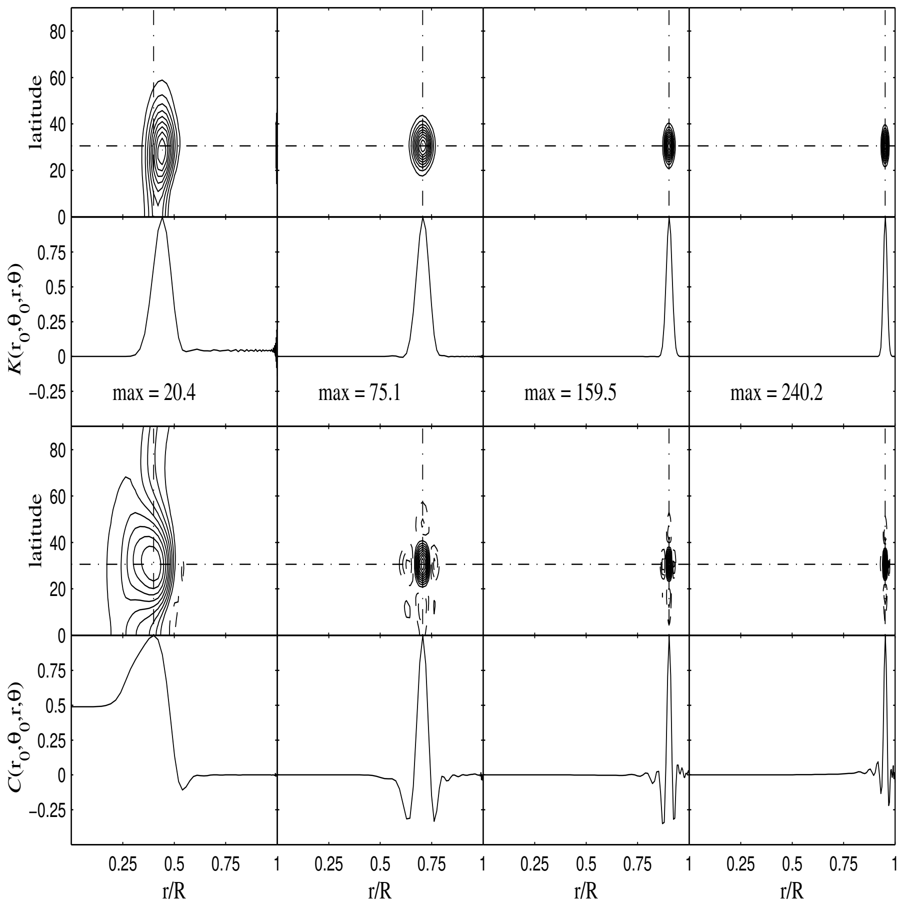


Figure 5.2: *Averaging kernels and error-correlation function for the SOLA inversion, at target latitude* 30.6°*; columns 1 through 4 correspond to target radii* r_0 *of* $0.40R$, $0.707R$, $0.903R$ *and* $0.952R$. *In the contour plots, positive contours are solid and negative contours dashed; the dash-dotted lines indicate the target location* (r_0, θ_0). *First row: averaging kernels, with contour spacings one tenth of the maximum value given in the second row. Second row: radial cuts through averaging kernels, at target latitude. Third row: Error correlation function* $C(r_0, \theta_0, r, \theta)$, *with contour spacing 0.1. Fourth row: radial cuts through error-correlation functions, at target latitude.*

distributed to a number of groups of helioseismologists (the "Hounds") for them to invert. For 2D-SOLA this test showed that the method produced well-localized averaging kernels, as illustrated in Figure 5.2 at latitudes below $\approx 75°$ and radii greater than $\approx 0.5R$, but it was also found that the averaging kernels with target location close to the rotation axis were shifted towards lower latitudes causing a systematic error seen as a spurious upturn towards the pole in the inferred rotation rate. As discussed in Appendix D, we have tried to tackle this problems in two ways: 1) by modifying the target function used in SOLA to match the latitudinal behavior of the (splitting coefficient) mode kernels, which go to zero towards the pole like $\sin^{3/2}\theta$ and 2) by discarding the solution points that still drift too far away from the intended target position. The drift of the averaging kernels and the criterion used for discarding unreliable solutions are illustrated in Figure 5.3.

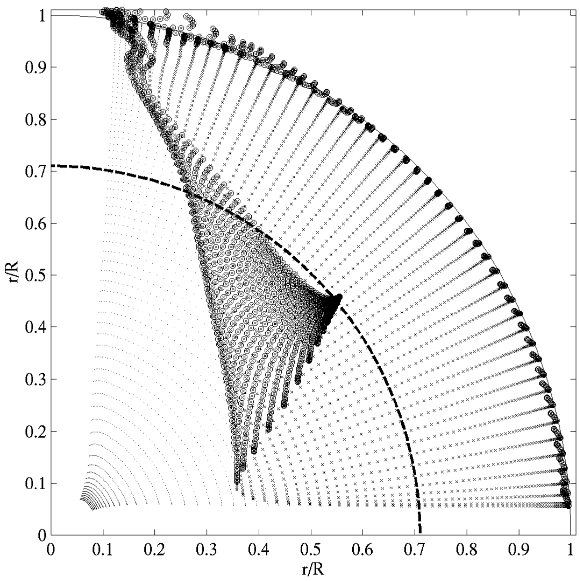


Figure 5.3: Drift of the averaging kernels away the target position. Dots indicate the center of gravity $(r_{\rm cg,0}, \theta_{\rm cg,0})$ of the targets while crosses show the venter of gravity $(r_{\rm cg}, \theta_{\rm cg})$ of the averaging kernels. The circled crosses are those solutions discarded according to the criterion $|r_{\rm cg} - r_{\rm cg,0}| < 0.2\Delta_r$ or $|\theta_{\rm cg} - \theta_{\rm cg,0}| < 0.2\Delta_\theta$.

The modifications to the SOLA technique have clearly resulted in a significant improvement in the results, as shown by the tests on artificial data shown in Appendix D. In the reliable region the agreement with the true artificial rotation rate is very good. It is likely that additional improvements can be made through further modifications of the target function, based on analysis of the asymptotic approximation to the latitudinal variation of the kernels. The two panels in Figure 5.4 show the result of applying the improved SOLA to the 144 and 360 days MDI datasets. The general picture of the differential rotation in and just below the convection zone, the slowdown of the rotation at the pole and the structure of the tachocline agrees with the findings of Schou et al. [175] from the same dataset.

Other sources of spurious features in the solutions are error correlations in the data and the solution. We have tried to include estimates of the error correlation between splitting coefficients within the same multiplet (Schou et al. [174], Gough [89]) in the inversion. Usually the correlations are neglected and simple error estimates σ_{nls} are provided for the data. As ca be seen in Figure 5.5 the general effect of including the correlations is very small, with the largest difference between the solution with and without the error-correlation included being at the 1σ level. Also, the averaging kernels and error-correlation function for the solution remain almost unchanged.

The analysis of the error covariance function and our experiments with including error correlations for the data in the inversions give no reason for major concern in the SOLA inversions of MDI data.

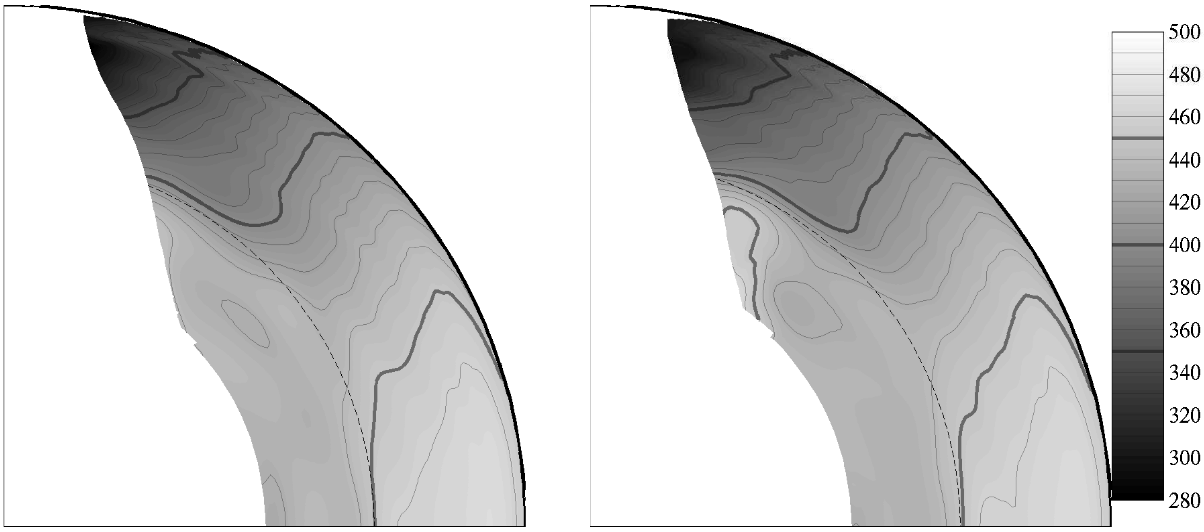


Figure 5.4: *Color-contour plot of the inferred solution rate, using SOLA with modified targets. The panels show the rotation in one quadrant of the Sun, the full rotation is found by mirroring the images around the equator (the horizontal line) and the rotation axis (the vertical line). The left panel corresponds to 144 days of MDI data, while the right panel shows results for the 360 day MDI data set. The solution has been restricted to the reliable region, and is plotted against the center of gravity of the averaging kernels.*

5.2 Conclusion

Regarded as a case study in scientific computing, the investigation in this thesis of algorithms for two-dimensional helioseismic inversion demonstrates that in many applications it is simply not enough to use standard algorithms from numerical linear algebra, say, even though they may be state-of-the-art methods for the general problem they solve. Instead, it is necessary to include special knowledge about the problem at hand – in this case the structure of the kernels – in order to obtain an efficient algorithm. In two-dimensional helioseismic inversion this approach represents a substantial breakthrough for the treatment of large datasets, and hopefully the reduction in complexity that the new methods developed here represent, will allow us to treat even the largest future datasets expected from the combination of MDI and GONG data with relative ease.

There are still some open questions with regards to the methods. In particular we have only touched upon the issue of choosing the regularization parameters. For the OLA methods this has turned out to be a harder problem than we first thought, and certainly requires more attention.

It is also interesting how the inspiration from working with large problems from a real-world application lead to the investigation of fundamental problems in connection with implementing Lanczos methods for sparse SVD calculation, and for studying methods for solving sparse linear least-squares problems with many right-hand sides. We hope that this will turn out to be useful contributions as well.

Gute Nacht, gute Nacht!
Bis alles wacht,
Schlaf aus deine Freude, schlaf aus dein Leid!
Der Vollmond steigt,
Der Nebel weicht,
Und der Himmel da oben, wie ist er so weit!

— W. Müller, from "Die Schöne Müllerin"

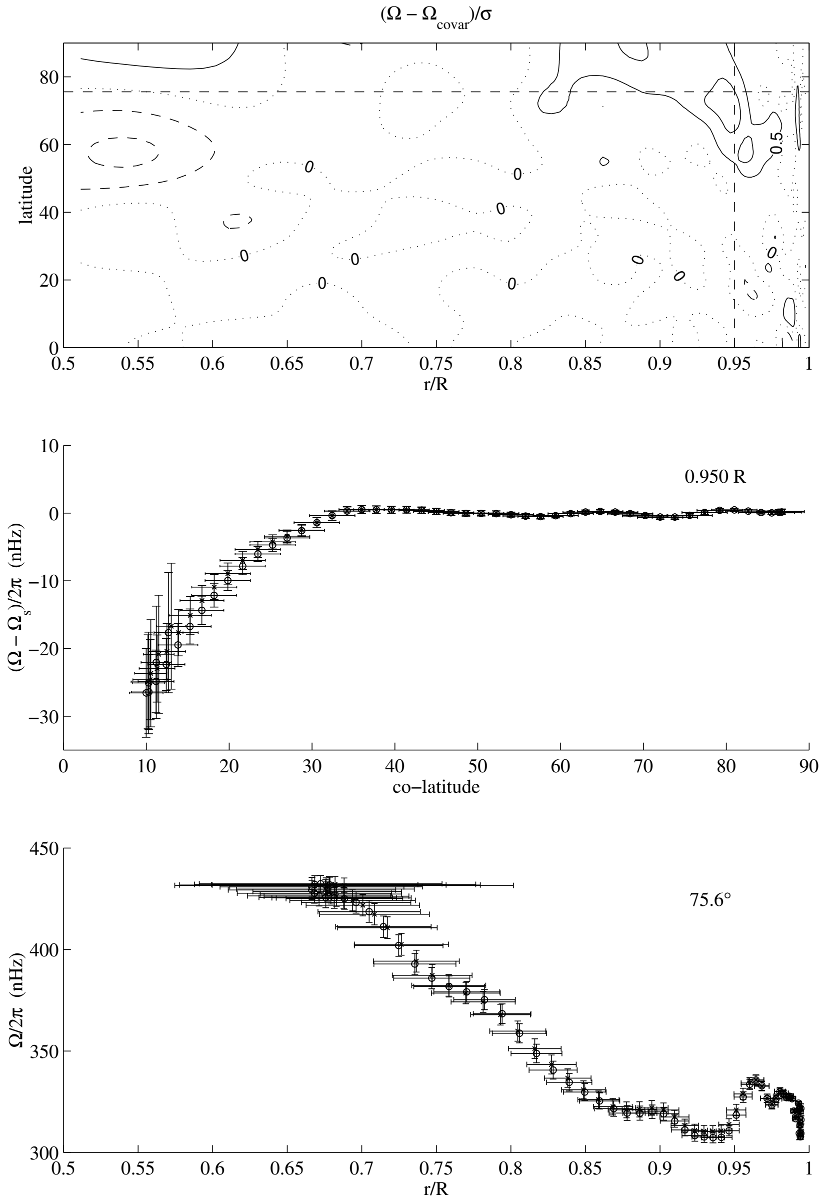


Figure 5.5: *Top panel shows normalized difference $(\Omega_{\rm covar}(r,\theta) - \Omega(r,\theta))/\sigma(r,\theta)$ between the solutions for the MDI 144 day set with and without information about the data error-correlations included; the dashed lines indicate the positions of the cuts through the two solutions shown in the middle and lower panels. Solid lines show positive contours, dashed lines show negative contours and dotted lines show zero contours. Middle panel shows a cut at constant radius $r = 0.95R$ through the residual obtained by subtracting a three term fit in $\sin^2\theta$ from solution. Bottom panel shows a cut at constant latitude $75.6°$. In the middle and bottom each rotation estimate is plottet against the position of the center of gravity of the corresponding averaging kernel. Vertical and horizontal error-bars show formal errors in the solation and inter-quartile width of the averaging kernels respectively.*

Bibliography

De higer og søger i gamle bøger.

— A. Oehlenschläger

(OK, der skal jo være lidt folkeligt)

[1] M. Abramowits and I. A. Stegun, *Handbook of Mathematical Functions*, NBS Applied Mathematics Series No. 55, NBC, Washington, 1964.

[2] J. C. Allen, *Matrix expansion by orthogonal Kronecker products*, Amer. Math. Monthly, **102** (1995), no. 6, 538–540.

[3] G. M. Amdahl, *Validity of the single processor approach to achieving large-scale computing capabilities*, in AFIPS Conference Proceedings **30**, AFIPS Press, Montvale, NJ, 1967, 483–485.

[4] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, O. Ostrouchow and D. Sorensen, *LAPACK Users' Guide*, 2nd ed., SIAM, Philadelphia, 1995, `http://www.netlib.org/lapack/`.

[5] E. Anderson, J. J. Dongarra, *Evaluating Block Algorithm Variants in LAPACK*, LAPACK Working Note # 19, 1990.

[6] E. R. Anderson, T. L. Duvall, Jr., S. M. Jefferies, *Modeling of Solar Oscillation Power Spectra*, Astrophys. J., **364** (1990), 699–705.

[7] T. Appourchaux, L. Gizon and M.-C. Rabello-Soares, *The art of fitting p-mode spectra: Part I. Maximum Likelihood Estimation*, Astron. Astrophys., 1997.

[8] T. Appourchaux, L. Gizon and M.-C. Rabello-Soares, *The art of fitting p-mode spectra: Part II. Leakage and noise covariance matrices*, Astron. Astrophys., 1997.

[9] E. F. D'Azevedo, V.L. Eijkhout and C.H. Romine, *Reducing Communication Costs in the Conjugate Gradient Algorithm on Distributed Memory Multiprocessors*, University of Tennesee, CS-93-187, May 1993.

[10] G. E. Backus and J. F. Gilbert, *The resolving power of gross earth data*, Geophys. J. R. Astron. Soc. **16** (1968), 169–205.

[11] D. H. Bailey, *Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers*, RNR Technical Report RNR-91-020, 1991.

[12] S. Balay, W. Gropp, L. C. McInnes and B. Smith. *PETSc 2.0 users manual*, Technical Report ANL-95/11, Mathematics and Computer Science Division, Argonne National Laboratory, 1995. Latest source code available from `http://www.mcs.anl.goc/petsc`.

[13] S. M. Balle and P. C. Hansen, *Block Algorithms for the Connection Machine*, Technical Report UNIC-93-10, UNI•C, 1993.

[14] S. M. Balle, P. C. Hansen and N. J. Higham, *A Strassen-Type Matrix Inversion Algorithm for the Connection Machine*, Technical Report UNIC-93-11, UNI•C, 1993.

[15] V. A. Barker, *Iterative methods for sparse systems of linear equations*, Hæfte 62, Institute for Numerical Analysis, Technical University of Denmark, 1992.

[16] R. Barrett, M. Berry, T. 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, 1993, `http://www.netlib.org/templates/`.

[17] A. Basermann, M. Bücker, P. Weidner, P. C. Hansen and R. M. Larsen, *Parallel Iterative Methods for Nonsymmetric Large-Scale Problems*, APPARC PaA5a Deliverable, Esprit BRA III Contract # 6634, 1995.

[18] A. Basermann and P. Weidner, *A parallel algorithm for determining all eigenvalues oflarge real symmetric tridiagonal matrices*, Parallel Computing, **18** (1992), 1129–1141.

[19] F. Baskett and J. L. Hennesy, *Microprocessors: From Desktops to Supercomputers*, Science, **261** (1993), 864–870.

[20] S. Basu, W. J. Chaplin, J. Christensen-Dalsgaard, Y. Elsworth, G. R. Isaak, R. New, J. Schou, M. J. Thompson & S. Tomczyk, *Solar internal sound speed as inferred from combined BiSON and LOWL oscillation frequencies.* Mon. Not. R. astr. Soc., **291** (1997), 243–251.

[21] W. Bath, R. S. Martin & J. H. Wilkinson, *Calculation of the Eigenvalues of a Symmetric Tridiagonal Matrix by the Method of Bisection*, Numerische Mathematik **9**, 1967.

[22] M. W. Berry, *Large-scale sparse singular value computations*, Int. J. Supercomputer Appl. **6** (1992), no. 1, 1-21.

[23] M. W. Berry, *Computing the sparse singular value decomposition via SVDPACK.* Recent advances in iterative methods, 13–29, IMA Vol. Math. Appl., 60, Springer, New York, 1994.

[24] Å. Björck, *A bidiagonalization algorithm for solving large and sparse ill-posed systems of linear equations*, BIT **28** (1988), 659–670.

[25] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.

[26] Å. Björck, T. Elfving and Z. Strakoš. *Stability of conjugate gradient and Lanczos methods for linear least squares problems*, SIAM J. Matrix Anal. Appl. **19** (1998), no. 3, 720–736.

[27] Å. Björck, E. Grimme and P. Van Dooren, *An implicit shift bidiagonalization algorithm for ill-posed systems*, BIT **34** (1994), no. 4, 510–534.

[28] T. M. Brown *Techniques for Oberving solar Oscillations*, In [165], (1996)

[29] T. M. Brown & R. L. Gilliland *Asteroseismology.* Ann. Rev. Astron. Astrophys., **32** (1994), 37–82.

[30] N. H. Brummell, F. Cattaneo, and J. Toomre, *Turbulent dynamics in the solar convection zone*, Science, **269** (1995), 1370–1379.

[31] B. Caccin, C. Roberti, P. Russo & L. A. Smaldone, *The Backus-Gilbert inversion method and the processing of sampled data*, IEEE Trans. Signal Processing, **40** (1992), 2823–2825.

[32] D. Calvetti, G. Golub, L. Reichel, *Estimation of the L-Curve via Lanczos Bidiagonalization*, Tech. Rep. SCCM-97-12, Stanford University, 1997.

[33] D. Calvetti, L. Reichel and D. C. Sorensen, *An Implicitly Restarted Lanczos Method for Large Symmetric Eigenvalue Problems*, Elec. Trans. on Num. Anal. **2** (1994), 1–21.

[34] T. F. Chan, *An improved Algorithm for Computing the Singular Value Decomposition*, ACM Trans. Math. Software **8**, 72–83.

[35] T. F. Chan, *Algorithm 581: An improved Algorithm for Computing the Singular Value Decomposition*, ACM Trans. Math. Software **8**, 84–88

[36] T. F. Chan and W. L. Wan, *Analysis of Projection Methods for solving Linear Systems with Multiple Right-hand Sides*, SIAM J. Sci. Comput., **18** (1997), no. 6, 1698-1721.

[37] W. J. Chaplin, J. Christensen-Dalsgaard, Y. Elsworth, R. Howe, G. R. Isaak, R. M. Larsen, R. New, J. Schou, M. J. Thompson, & S. Tomczyk, *Rotation of the solar core from BiSON and LOWL frequency observations*, MNRAS, 1998, in preparation.

[38] P. Charbonneau, *Genetic algorithms in astronomy and astrophysics*, Astrophys. J. Suppl., **101** (1995), 309–334.

[39] P. Charbonneau, J. Christensen-Dalsgaard, R. Henning, R. M. Larsen, J. Schou, M. J.Thompson and S. Tomczyk, *Helioseismic constraints on the structure of the solar tachocline*, Astrophys. J., 1998., submitted.

[40] J. Choi et al., *The design and implementation of the ScaLAPACK LU, QR, and Cholesky factorization routines*, ORNL/TM-12470, Oak Ridge National Laboraroty, Oak Ridge, Tennesee, 1994.

[41] J. Choi et al., *ScaLAPACK: A Portable Linear Algebra Library for Distributed Memory Computers - Design issues and Performance*, In proceedings of Fourth Symposium on the Frontiers of massively Parallel Computation (McLean, Virginia), IEEE Computing Society Press, Los Alamitos, California, 1992, `http://www.netlib.org/scalapack/`.

[42] J. Choi et al., *A Proposal for a Set of Parallel Basic LinearAlgebra Subprograms*, LAPACK Working Note 100, 1995

[43] J. Christensen-Dalsgaard et al., *The current state of solar modelling*, Science, **272** (1996), 1286–1290.

[44] J. Christensen-Dalsgaard, *Lecture Notes on Stellar Oscillations*, Institute of Physics and Astronomy, University of Aarhus, 1998.

[45] J. Christensen-Dalsgaard and G. Berthomieu, *Theory of solar oscillations*, In “Solar interior and atmosphere”, eds A. N. Cox, W. C. Livingston, & M. Matthews, Space Science Series, University of Arizona Press, 1991, 401–478.

[46] J. Christensen-Dalsgaard, P. C. Hansen and M. J. Thompson, *GSVD analysis of helioseismic inversions*, Mon. Not. R. Astr. Soc. **264** (1993), 541–465.

[47] J. Christensen-Dalsgaard, R. M. Larsen, J. Schou and M. J. Thompson, *Optimally localized kernels for 2D helioseismis inversion*, In Proc. GONG'94: Helio- and Astro-seismology from Earth and Space, eds R. K. Ulrich, E. J. Rhodes Jr, W. Däppen, Astron. Soc. of the Pacific Conference Series, San Francisco, **76** (1994), 70–73.

[48] J. Christensen-Dalsgaard, J. Schou and M. J. Thompson, *A comparison of methods for inverting helioseismic data*, Mon. Not. R. Astr. Soc. **242** (1990), 353–369.

[49] J. Christensen-Dalsgaard and M. J. Thompson, *A preprocessing strategy for helioseismic inversions*, Astron. Astrophys., **272**, (1993), L1–L4.

[50] Convex Computing Corporation, *Convex Exemplar Programming Guide*, March 1994, DSW-067.

[51] Convex Computing Corporation, *cps - parallel compiler support library*, Manual pages, March 1994.

[52] Convex Computing Corporation, *Exemplar Architecture*, November 1993, DHW-014.

[53] A. N. Cox, W. C. Livingston & M. Mathews (eds), *Solar interior and atmosphere*, Space Science Series, University of Arizano Press.

[54] I. J. D. Craig and J. C. Brown, *Inverse Problems in Astronomy*, Adam Hilger, Bristol, 1986.

[55] J. K. Cullum, *Peaks and plateaus in Lanczos methods for solving nonsymmetric systems of equations* $Ax = b$, Report RC 18084, IBM Research Division, T. J. Watson Research Center, June 1992.

[56] J. K. Cullum and R. A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations. Vol. I Theory*, Birkhäuser, Boston, 1985.

[57] J. Cullum, R. Willoughby and M. Lake, *A Lanczos algorithm for computing singular values and vectors of large matrices*, SIAM J. Sci. Statist. Comput. 4 (1983), no. 2, 197–215.

[58] J. W. Daniel, W. B. Gragg, L. Kaufman and G. W. Stewart, *Reorthogonalization and Stable Algorithms Updating the Gram-Schmidt QR Factorization*, Math. Comp., **30** (1976), no. 136, 772 – 795.

[59] J. W. Demmel, M. T. Heath, H. A. van der Vorst, *Parallel Numerical Linear Algebra*, Acta Numerica 1993, 111–197.

[60] J. W. Demmel, N. J. Higham and R. S. Schreiber, *Stability of Block LU Factorization*, Num. Lin. Alg. Appl. Vol. 2(2) (1995), 173–190

[61] J. W. Demmel and W. Kahan, *Accurate singular values of bidiagonal matrices*, SIAM J. Sci. Statist. Comput. **11** (1990), 712–719.

[62] J. J. Dongarra, *Improving the accuracy of computed singular values*, SIAM J. Sci. Statist. Comput. **4** (1983), 712–719.

[63] J. J. Dongarra, J. R. Bunch, C. B. Moler and G. W. Stewart, *Linpack Users' Guide*, SIAM, Philadelphia, 1979.

[64] J. J. Dongarra, J. Du Croz, S. Hammerling and R. Hanson, *A Proposal of Level 3 Linear Algebra Subprograms*, Argonne National Laboratory, ANL-MCS-TM-41, 1986.

[65] J. J. Dongarra, J. Du Croz, S. Hammerling and R. Hanson, *A Proposal of Level 3 Linear Algebra Subprograms*, Argonne National Laboratory, ANL-MCS-TM-88, 1987.

[66] J. J. Dongarra, I. S. Duff, D. C. Sorensen, H. A. van der Vorst, *Solving Linear Systems on Vector and Shared Memory Computers*, SIAM, Philadelphia, 1991.

[67] J. J. Dongarra, F. G. Gustavson and A. Karp, *Implementing Algorithms for Dense Matrices on a Vector Pipeline Machine*, SIAM Review 26 (1984), 91–112.

[68] J. J. Dongarra and R. C. Whaley, *A Users's Guide to the BLACS v1.0*, LAPACK Working Note 94, 1995.

[69] J. J. Dongarra, *Freely available software for linear algebra on the web*, `http://www.netlib.org/utk/people/JackDongarra/la-sw.html`.

[70] L. Eldén *Algorithms for the regularization of ill-conditioned least-squares problems*, BIT **17** (1977), 134–145.

[71] L. Elden and E. Sjöström, *Fast computation of the principal singular values of Toeplitz matrices arising in exponential data fitting*, Signal Processing **50** (1996), 151–164.

[72] Y. Elsworth, R. Howe, G. R. Isaak, C. P. McLeod, B. A. Miller, S. J. Wheeler and R. New, *Fine spacing of the $l = 0$, $l = 2$ acoustic eigenmodes and the solar neutrino problem: particle physics and cosmological implications*, In *Proc. GONG'94: Helio- and Astero-seismology from Earth and Space*, eds R. K. Ulrich E. J. Rhodes Jr and W. Däppen, Astronomical Society of the Pacific Conference Series, vol. 76, San Francisco, **76** (1995), 51–54.

[73] Fausett, D. W., Fulton, C. T., *Large least saures problems involving Kronecker products*, SIAM J. Matrix Anal. Appl. **15** (1994), 219-227.

[74] R. D. Fierro, G. H. Golub, P. C. Hansen, D. P. O'Leary, *Regularization by truncated total least squares*, submitted to SIAM J. Sci. Comput., **18** (1997), 1223–1241.

[75] G. C. Fox, M. A. Johnson, G. A. Lyzenga, S. Otto, J. Salmon, D. Walker. *Solving Problems on Concurrent Procccesors*, Vol I: *General techniques and Regular Problems*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[76] R.W. Freund and N.M. Nachtigal, *QMR: A quasi-minimal residual method for non-Hermitian linear systems*, Numerische Mathematik, **60** (1991), 315–339.

[77] C. Fröhlich, D. A. Crommelynck, C. Wehrli, M. Anklin, S. Dewitte, A. Fichot, W. Finsterle, A. Jiménez, A. Chevalier & H. Roth, *In-flight performance of the VIRGO solar irradiance instrument on SOHO*, Solar Phys., **175** (1997), 267–286.

[78] A. H. Gabriel, J. Charra, G. Grec, J.-M. Robillot, T. Roca Cortés, S. Turck-Chièze, R. Ulrich, S. Basu, F. Baudin, L. Bertello, P. Boumier, M. Charra, J. Christensen-Dalsgaard M. Decaudin, H. Dzitko, T. Foglizzo, E. Fossat, R. A. García, J. M. Herreros, M. Lazrek, P. L. Pallé, N. Pétrou, C. Renaud & C. Régulo, *Performance and early results from the GOLF instrument flown on the SOHO mission*, Solar Phys., **175** (1997), 207–226.

[79] K. A. Gallivan et al., *Parallel Algorithms for Matrix Computations*, SIAM, Philadelphia, 1990.

[80] G. H. Golub, P. C. Hansen, D. P. O'Leary, *Tikhonov regularization and total least squares*, submitted to SIAM J. Matrix Anal. Appl., 1998.

[81] G. H. Golub and W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, J. Soc. Indust. Appl. Math. Ser. B Numer. Anal. **2** 1965 205–224.

[82] G. H. Golub, F. T. Luk and M. L. Overton, *A block Lanczos method for computing the singular values and corresponding vectors of a matrix*, ACM Trans. Math. Software, **7** (1981), 149–169.

[83] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2. Ed., Johns Hopkins University Press, Baltimore, 1989.

[84] G. H. Golub, U. von Matt, Generalized Cross-Validation for Large Scale Problems, *Journal of Computational and Graphical Statistics*, 6, 1–34,(1997).

[85] G. H. Golub, U. von Matt, *Tikhonov Regularization for Large Scale Problems*, in "Workshop on Scientific Computing", eds. G. H. Golub, S. H. Lui, F. Luk, and R. Plemmons Springer, New York, 1997.

[86] R. C. Gonzales and P. Wintz, *Digital Image Processing*, Addison-Wesley, Reading, Massachusetts, 1977.

[87] D. O. Gough, *Inverting helioseismic data*, Solar Phys., **100** (1985), 65–99.

[88] D. O. Gough, *Course 7. Linear adiabatic stellar pulsation*, In "Astrophysical fluid dynamics, Les Houches Session XLVII", eds J.-P. Zahn & J. Zinn-Justin, Elsevier, Amsterdam, 399–560, 1993.

[89] D. O. Gough, *Testing solar models: The inverse problems*, In "The structure of the Sun", T. Roca Cortes (ed.), Cambridge University Press, 1996.

[90] D. O. Gough, J. W. Leibacher, P. H. Scherrer and J. Toomre, *Perspectives in Helioseismology*, Science, **272** (1996), 1281–1283.

[91] Graham, A., *Kronecker Products and Matrix Calculus: with Applications*, Halsted Press, John Wiley and Sons, Inc., New York, 1981.

[92] J. L. Gustavson, *Reevaluating Amdahl's law*, Comm. of the ACM, **31** (1988), 532–533.

[93] A. Greenbaum and J. J. Dongarra, *Experiments with QL/QR methods for the symmetric tridiagonal eigenproblem*, Computer Science Dept. Technical Report CS-89-92, University of Tennessee, Knoxville, 1989. (LAPACK Working Note 17).

[94] A. Greenbaum and Z. Strakos, *Predicting the behavior of finite precision Lanczos and conjugate gradient computations*, SIAM J. Matrix Anal. Appl., Vol. 13(1), 121–137, 1992.

[95] M. Hanke, *Regularization with differential operators: an iterative approach*, Numer. Funct. Anal. Optim. **13** (1992), no. 5-6, 523–540.

[96] M. Hanke, *Limitations of the L-curve method in ill-posed problems*, BIT **36** (1996), no. 2, 287–301.

[97] M. Hanke and P. C. Hansen, *Regularization methods for large-scale problems*, Surv. Math. Ind. **3** (1993), 253–315.

[98] M. Hanke and J. G. Nagy, *Restoration of Atmospherically Blurred Images by Symmetric Indefinite Conjugate Gradient Techniques*, Inverse Problems **12** (1996), 157–173.

[99] M. Hanke, J. Nagy and R. Plemmons, *Preconditioned iterative regularization for ill-posed problems*, Numerical linear algebra (Kent, OH, 1992), 141–163, de Gruyter, Berlin, 1993.

[100] P. C. Hansen, *Regularization, GSVD and truncated GSVD*, BIT **29** (1989), 491–504.

[101] P. C. Hansen, *The discrete Picard criterion for discrete ill-posed problems*, BIT **30** (1990), 658–672.

[102] P. C. Hansen, *Regularization Tools: A Matlab package for analysis and solution of discrete ill-posed problems*, Numerical Algorithms **6** (1994), 1–35.

[103] P. C. Hansen, *The Backus-Gilbert method: SVD analysis and fast implementation*, Inverse Problems **10** (1994), 895–904.

[104] P. C. Hansen, *Experience with Regularizing CG Iterations*, Report UNIC-94-05, UNI•C, 1994.

[105] P. C. Hansen, *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*, SIAM, Philadelphia, 1998.

[106] P. C. Hansen and D. P. O'Leary, *The use of the L-curve in the regularization of discrete ill-posed problems*, SIAM J. Sci. Comput. **14** (1993), 1487–1503.

[107] H. Haario and E. Somersalo, *The Backus-Gilbert method revisited: background, implementation and examples*, Numer. Funct. Anal. Optim., **9** (1987), no. 9–10, 917–943.

[108] J. W. Harvey, F. Hill, R. P. Hubbard, J. R. Kennedy, J. W. Leibacher, J. A. Pintar, P. A. Gilman, R. W. Noyes, A. M. Title, J. Toomre, R. K. Ulrich, A. Bhatnagar, J. A. Kennewell, W. Marquette, J. Partrón, O. Saá & E. Yasukawa, *The Global Oscillation Network Group (GONG) project*, Science, **272** (1996), 1284–1286.

[109] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 2. Ed., Morgan Kaufmann Publishers, Inc., San Mateo, California, 1996.

[110] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, *B49* (1952), pp. 409–436.

[111] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.

[112] F. Hill et al., *GONG Estimates of Solar Eigenspectral Parameters*, Science **272** (1996).

[113] W. D. Hillis and G. L. Steele, Jr., *Data Parallel Algorithms*, Comm. of the ACM, **29** (1986), 1170–1183.

[114] R. Howe and M. J. Thompson, *On the use of the error correlation function in helioseismic inversion*, Mon. Not. R. Astron. Soc. **281** (1996), 1385–1392.

[115] S. A. Hutchinson, J. N. Shadid and R. S. Tuminaro, *AZTEC user's guide*, Technical Report SAND95-1559, Massively parallel computing research laboratory, Sandia National Laboratories, Albuquerque, NM, 1995,
`http://www.cs.sandia.gov/CRF/aztec1.html`.

[116] International Business Machines Corporation, *PowerPC and Power2: Technical Aspects of the New IBM RISC System/6000*, Austin, Texas, 1994.

[117] *IMSL Math/Library*, Version 2.0, 1991.

[118] D. D. Jackson, *The use of a priori data to resolve non-uniqueness in linear inversion*, Geophys. J. Royal Astron. Soc., **75**, 137–157.

[119] M. T. Jones and P. E. Plassmann, *Blocksolve95 user's manual: scalable library software for parallel soution of sparse linear systems*, Technical Report ANL-95/48, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 1995,
`http://www.mcs.anl.gov/sumaa3d/BlockSolve/`.

[120] S. Kaniel, *Estimates for Some Computational Techniques in Linear Algebra*, Math. Comp., **20** (1966), 369–78.

[121] A. G. Kosovichev, J. Schou, *Detection of zonal shear flows beneath the Sun's surface from f-mode frequency splitting*, Astrophys. J., **482** (1997), L207 – L210.

[122] A. G. Kosovichev, J. Schou, P. H. Scherrer, R. S. Bogart, R. I. Bush, J. T. Hoeksema, J. Aloise, L. Bacon, A. Burnette, C. de Forest, P. M. Giles, K. Leibrand, R. Nigam, M. Rubin, K. Scott, S. D. Williams, S. Basu, J. Christensen-Dalsgaard, W. Däppen, E. J. Rhodes Jr, T. L. Duvall Jr, R. Howe, M. J. Thompson, D. O. Gough, T. Sekii, J. Toomre, T. D. Tarbell, A. M. Title, D. Mathur, M. Morrison, J. L. R. Saba, C. J. Wolfson, I. Zayer and P. N. Milford, *Structure and Rotation of the Solar Interior: Initial Results from the MDI Medium-l Program*, Solar Phys. **170** (1997), 43 – 61.

[123] B. Kågström and C. Van Loan, *GEMM-Based Level-3 BLAS*, Tech. rep. CTC91TR47, Department of Computer Science, Cornell University, Dec. 1989.

[124] C. Lanczos, *Solutions of systems of linear equations by minimized iterations*, Journal of Research of the National Bureau of Standards, **49**, (1952), 33–53.

[125] R. M. Larsen, *Iterative algorithms for two-dimensional helioseismic inversion*, in Proc. of the Interdisciplinary Inversion Workshop 5, ed. Jacobsen, B. H (Århus: Dept. Earth Sciences, Aarhus University), 123 – 138, 1997.

[126] R. M. Larsen, J. Christensen-Dalsgaard, A. G. Kosovichev and J. Schou, *Improved SOLA Inversions of MDI Data*, In Structure and Dynamics of the Interior of the Sun and Sun-like Stars (Eds. S. G. Korzennik & A. Wilson), ESA SP-418, ESA Publications Division, Noordwijk, The Netherlands, 1998, in the press (6 pages).

[127] R. M. Larsen, *Lanczos bidiagonalization with partial reorthogonalization*, Aarhus University, DAIMI PB-357, September 1998 (93 pages).

[128] R. M. Larsen, J. Christensen-Dalsgaard, A. G. Kosovichev and J. Schou, *Improved SOLA Inversions of MDI Data*, In Structure and Dynamics of the Interior of the Sun and Sun-like Stars (Eds. S.G. Korzennik & A. Wilson), ESA SP-418, ESA Publications Division, Noordwijk, The Netherlands, 1998, in the press.

[129] R. M. Larsen and P. C. Hansen, *Efficient Implementations of the SOLA Mollifier Method*, Astron. Astrophys. Suppl., **121** (1997), 587.

[130] R. M. Larsen and S. Larsen, *Analyse af parallelle programmer p SGI Power Challenge ved hjlp af profileringsvrktjet* `prof`, Department of Computer Science, University of Aarhus, May 1998, URL: `http://www.daimi.aau.dk/~rmunk/Profile/profile_rev2.ps.gz`

[131] R. M. Larsen, S. Larsen og J. B. Pedersen, *EARTH-C On the SGI Power Challenge XL Symmetrical Multiprocessor*, Department of Computer Science, University of Aarhus, Jan. 1997, URL: `http://www.daimi.aau.dk/~rmunk/OPC\_Report/Rap.html`

[132] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Prentice-Hall, 1974.

[133] C. Lawson, R. Hanson, D. Kinkaid and F. Krogh, *Basic Linear Algebra Subprograms for Fortran Usage*, ACM Trans. on Math. Software, **5** (1979), 308–323.

[134] C. Lawson, R. Hanson, D. Kinkaid and F. Krogh, *Algorithm 539: Basic Linear Algebra Subprograms for Fortran Usage*, ACM Trans. on Math. Software, **5** (1979), 324–325.

[135] R. B. Lehoucq, D. C Sorensen and C. Yang, *ARPACK Users's Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998, URL: `http://www.caam.rice.edu/software/ARPACK`.

[136] A. K. Louis and P. Maass, *A mollifier method for linear operator equations of the first kind*, Inverse Problems **6** (1990), 427–440.

[137] A. K. Louis and P. Maass, *Smoothed projection methods for the moment problem*, Numer. Math. **59** (1991), 277–294.

[138] K. Madsen, S. Zertchaninov and Antanas Žilinskas, *Global Optimization using Branch-and-Bound*, Tech. Rep., Inst. of Math. Modelling, Technical university of Denmark, 1998.

[139] P. C . Messina, *Parallel computing in the 1980s – one person's view*, Concurrency: Practice and Experience, Vol. 3 (6), 501–524, 1991.

[140] *NAG Fortran Library Manual, Mark 15*, NAG Ltd., Oxford, 1991.

[141] H. B. Nielsen, *Direkte metoder for sparse matricer*, Hæfte 65, Institute of Mathematical Modelling, Technical University of Denmark, 1995.

[142] R. Nigam, A. G. Kosovichev, P. H. Scherrer and J. Schou *Asymmetry in velocity and intensity helioseismic spectra: a solution to a long-standing puzzle*, Astrophys. J., **495** (1998), L115–L118.

[143] D. P. O'Leary, *The block conjugate gradient algorithm and related methods*, Linear Algebra Appl. **29** (1980), 292–322.

[144] D. P. O'Leary and J. A. Simmons, *A bidiagonalization-regularization procedure for large scale discretizations of ill-posed problems*, SIAM J. Sci. Statist. Comput., **2** (1981), 474–489.

[145] C. C. Paige, *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices*, Ph. D. thesis, University of London, United Kingdom, 1971.

[146] C. C. Paige and M. A. Saunders, *LSQR: an algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software **8** (1982), 43–71.

[147] C. C. Paige and M. A. Saunders, *Algorithm 583. LSQR: sparse linear equations and least squares problems*, ACM Trans. Math. Software **8** (1982), 195–209.

[148] C. M. Pancake, *Is parallelism for you?*, IEEE J. Computational Sci. Eng., **3** (1996), no. 2, 18–37.

[149] R. L. Parker, *Understanding inverse theory*, Ann. Rev. Earth Planet. Sci. **5** (1977), 35–64.

[150] R. L. Parker, *Geophysical Inverse Theory*, Princeton University Press, 1994.

[151] B. N. Parlett, *A new look at the Lanczos algorithm for solving symmetric systems of linear equations*, Linear Algebra Appl., **29** (1980), 323–346.

[152] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[153] B. N. Parlett and B. Nour-Omid, *The use of a refined error bound when updating the eigenvalues of tridiagonals*, Lin. Alg. Appl. *68* (1985), 179–219.

[154] B. N. Parlett, J. K. Reid, *Tracking the progress of the Lanczos algorithm for large symmetric eigenproblems*, IMA J. Numer. Anal. **1** (1981), no. 2, 135–155.

[155] B. N Parlett and D. S. Scott *The Lanczos algorithm with selective orthogonalization*, Math. Comp. **33** (1979), no. 145, 217–238.

[156] G. Peters and J. H. Wilkinson, *The Calculation of Specified Eigenvectors by Inverse Iteration*, in J. H. Wilkinson and C. Reinsch (eds) "Handbook for Automatic Computation. Volume II", Springer Verlag, Berlin, Heidelberg, New York, 1971.

[157] F. P. Pijpers, *Solar rotation inversions and the relationship between a-coefficients and mode splittings*, Astron. Astrophys., **326** (1997), 1235.

[158] F. P. Pijpers and M. J. Thompson, *Faster formulation of the optimally localized averages method for helioseismic inversions*, Astr. Astrophys. **262** (1992), L33–L36.

[159] F. P. Pijpers and M. J. Thompson, *The SOLA method for helioseismic inversion*, Astr. Astrophys. **281** (1994), 231–240.

[160] F. P. Pijpers and M. J. Thompson, *A modified* $\mathbb{R}^1 \otimes \mathbb{R}^1$ *method for helioseismic rotation inversions*, Mon. Not. R. Astron. Soc. **279** (1996), 498–510.

[161] W. H. Press, S. A. Teukolsky, T. W. Vetterling, B. P. Flannery, 1992, *Numerical Recipes*, 2. Ed., Cambridge University Press.

[162] E. J. Rhodes, A. G. Kosovichev, J. Schou, P. H. Scherrer & J. Reiter, *Measurements of frequencies of solar oscillations from the MDI medium-l program*, Solar Phys., **175** (1997), 287–310.

[163] E. J. Rhodes, Jr, J. Reiter, A. G. Kosovichev, J. Schou, & P. H. Scherrer, *Initial SOI/MDI high-degree frequencies and frequency splittings*, In *Structure and dynamics of the interior of the Sun and Sun-like stars; Proc. SOHO 6/GONG 98 Workshop*, eds S.G. Korzennik & A. Wilson, ESA **SP-418**, ESA Publications Division, Noordwijk, The Netherlands, 1998, in the press.

[164] M. H. Ritzvoller and E. M. Laveley, *A Unified Approach to the Helioseismic Forward and Inverse Problems of Differential Rotation*, Astrophys. J. **369** (1991), 557–566.

[165] T. Roca Cortes (ed.), *The structure of the Sun*, Cambridge University Press, 1996.

[166] A. Ruhe, *Numerical aspects of Gram-Schmidt orthogonalization of vectors*, Linear Algebra Appl. *52/53* (1983), 591–601.

[167] Y. Saad, *On the rates of convergence of the Lanczos and the block-Lanczos Methods*, SIAM J. Numer. Anal., **17** (1980), no. 5, 687 – 706

[168] Y. Saad, *On the Lanczos Method for Solving Symmetric Linear Systems with Several Right-Hand Sides*, Math. Comp., **48** (1987), no. 178, 651 – 662.

[169] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishig company, Boston, MA, 1996.

[170] Y. Saad, G. Lo, S. Kuznettsov, *PSPARLIB Users Manual: A Portable Library of parallel Sparse Iterative Solvers*, January 1998, `http://www.cs.umn.edu/Research/arpa/p_sparslib`.

[171] P. H. Scherrer, R. S. Bogart, R. I. Bush, J. T. Hoeksema, A. G. Kosovichev, J. Schou, W. Rosenberg, L. Springer, T. D. Tarbell, A. Title, C. J. Wolfson, I. Zayer, and the MDI engineering team, *The Solar Oscillation Investigation – Michelson Doppler Imager* Solar Phys., **162** (1995), 129–188.

[172] B. Schomburg and G. Berendt, *On the convergence of the Backus-Gilbert algorithm*, Inverse Problems **3**, 341–346.

[173] J. Schou, *On the analysis of helioseismic data*, PhD thesis, Aarhus University, Århus, 1992.

[174] J. Schou, J. Christensen-Dalsgaard, M. J. Thompson, *Some aspects of helioseismic time-series analysis*, In Proc. "GONG'94: Helio- and Astro-seismology from Earth and Space", eds Ulrich, R. K., Rhodes Jr, E. J., Däppen, W., Astron. Soc. of the Pacific Conference Series, San Francisco, **76** (1995) , 528.

[175] J. Schou, H. M. Antia, S. Basu, R. S. Bogart, R. I. Bush, S. M. Chitre, J. Christensen-Dalsgaard, M. P. Di Mauro, W. A. Dziembowski, A. Eff-Darwich, D. O. Gough, D. A. Haber, J. T. Hoeksema, R. Howe, S. G. Korzennik, A. G. Kosovichev, R. M. Larsen, F. P. Pijpers, P. H. Scherrer, T. Sekii, T. D. Tarbell, A. M. Title, M. J. Thompson and J. Toomre, *Helioseismic Studies With SOI-MDI of Differential Rotation in the Solar Envelope*, Astrophys. J. **505** (1998), 390–417.

[176] J. Schou, J. Christensen-Dalsgaard, R. Howe, R. M. Larsen, M. J. Thompson, J. Toomre, *Slow poles and shearing flows from helioseismic observations with MDI and GONG data spanning a year*, In Structure and Dynamics of the Interior of the Sun and Sun-like Stars (Eds. S. G. Korzennik & A. Wilson), ESA SP-418, ESA Publications Division, Noordwijk, The Netherlands, 1998, in the press (6 pages).

[177] J. Schou *Observations of Medium- and High-Degree Modes: Methods and Sand-Traps*, In Structure and Dynamics of the Interior of the Sun and Sun-like Stars (Eds. S. G. Korzennik & A. Wilson), ESA SP-418, ESA Publications Division, Noordwijk, The Netherlands, 1998, in the press.

[178] J. Schou, J. Christensen-Dalsgaard, and M. J. Thompson, *On Comparing Helioseismic 2-dimensional Inversion Methods*, Astrophys. J., **433** (1994), 389.

[179] Science special issue on helioseismology, Science, **272** (1996), 1281–1309.

[180] T. Sekii, *A new strategy for 2D inversion for solar rotation*, Mon. Not. R. Astr. Soc. **264** (1993), 1018 – 1024.

[181] J. R. Shewchuk, *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*, School of Computer Science, Carnegie Mellon University, 1994.

[182] H. D. Simon, *The Lanczos algorithm with partial reorthogonalization*, Math. Comp. **42** (1984), no. 165, 115–142.

[183] H. D. Simon, *Analysis of the symmetric Lanczos algorithm with reorthogonalization methods*, Linear Algebra Appl. **61** (1984), 101–131.

[184] D. C. Sorensen, *Implicitly restarted Arnoldi/Lanczos methods for large scale eigenvalue calculations*, Dept. Comp. Appl. Math., Rice University, Houston, Texas, 1995.

[185] Silicon Graphics Computer Systems, *The advent of Powercomputing*, Silicon Graphics Technical Report, California, 1994.

[186] A. A. Stepanov, *Inverse problem of the rotional splitting*, Institute of Physics and Astronomy, University of Aarhus, 1995.

[187] A. A. Stepanov, J. Christensen-Dalsgaard *On the choice of trade-off parameter in helioseismic SOLA inversion*, Institute of Physics and Astronomy, University of Aarhus, 1995.

[188] E. de Sturler and H. A. van der Vorst, *Reducing the effect of global communication in GMRES(m) and CG on parallel distributed memory computers*, Appl. Num. Math. **18** (1995), 441–459.

[189] A. S. Tannenbaum, *Structured Computer Organization*, Third ed., Prentice-Hall, 1990.

[190] A. S. Tannenbaum, *Modern Operating Systems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1992.

[191] A. Tarantola, *Inverse Problem Theory*, Elsevier, 1987.

[192] A. Tarantola and B. Valette, *Generalized nonlinear inverse problems solved using the least squares criterion*, Reviews of Geophysics and Space Physics, *20*, 219–232.

[193] M. J. Thompson, *Seismic investigation of the Sun's internal structure and rotation*, in Proc. GONG 1992: Seismic Investigation of the Sun and stars, ed. T. M. Brown, PASPC, **42** (1993), 141.

[194] Thompson, M. J., Toomre, J., Anderson, E. R., Antia, H. M., Berthomieu, G., Burtonclay, D., Chitre, S. M., Christensen-Dalsgaard, J., Corbard, T., DeRosa, M., Genovese, C. R., Gough, D. O., Haber, D. A., Harvey, J. W., Hill, F., Howe, R., Korzennik, S. G., Kosovichev, A. G., Leibacher, J. W., Pijpers, F. P., Provost, J., Rhodes Jr., E. J., Schou, J., Sekii, T., Stark, P. B., Wilson, P. R., *Differential Rotation and Dynamics of the Solar Interior*, Science, **272** (1996), 1300–1305.

[195] A. N. Tikhonov, *Solution of incorrectly formulated problems and the regularization method*, Soviet Math. Dokl. **4** (1963), 1035–1038.

[196] S. Tomczyk, K. Streander, G. Card, D. Elmore, H. Hull, and A. Caccani, *An instrument to observe low-degree solar oscillations*, Solar Phys., **159** (1995), 1–21.

[197] H. A. van der Vorst, *An iterative solution method for solving $f(A)\,x = b$, using Krylov subspace information obtained for the symmetric positive definite matrix A*, J. Comput. Appl. Math. **18** (1987), no. 2, 249–263.

[198] A. van der Sluis and H. A. van der Vorst, *The rate of convergence of conjugate gradients*, Numer. Math. **48** (1986), no. 5, 543–560.

[199] E. F. Van de Velde, *Concurrent Scientific Computing*, Springer-Verlag, New York, 1994.

[200] B. van Rietdergen et al., *A new method to determine trabecular bone elastic properties and loading using micromechanical finite-element models*, J. Biomech., Vol. 28 1 (1995), 69–81.

[201] C. R. Vogel, *Optimal choice of a truncation level for the truncated SVD solution of linear first kind integral equations when the data are noisy*, SIAM J. Numer. Anal. **23** (1986), 109–117.

[202] C. R. Vogel, *Solving Ill-conditioned Linear Systems using the Conjugate Gradient Method*, Tech. Rep., Dept. of Mathematical Sciences, Montana State University, Bozeman, MT, 1986.

[203] C. R. Vogel, *Sparse matrix equations arising in distributed parameter identification*, SIAM Journal on Matrix Analysis and Applications, to appear, 1998.

[204] K. Vuik, A. G. J. Sevink and G. C. Herman, *A Preconditioned Krylov Subspace Method for the Solution of Least Squares Problems in inverse Scattering*, J. Comp. Phys. **123** (1996), 330.

[205] L. Wang and A. Zunger, *Large scale electronic structure calculations using the Lanczos method*, Computational Material Science **2** (1994), 326–340.

[206] G. M. Wing, *A Primer on Integral Equations of the First Kind*, SIAM, Philadelphia, 1991.

[207] K. Wu and H. D. Simon, *A Parallel Lanczos Method for Symmetric Generalized Eigenvalue Problems*, Tech. Report., NERSC, 1997,
URL: `http://www.nersc.gov/research/SIMON/planso.html`.

Part II

Papers

Overview

The articles found in the following appendices constitute a main part of this thesis. In this section I will give a list of the papers included, and describe what they cover and what my involvement in them was.

Appendices A through F contain 6 papers of where I am the main author and did all or most of the work. In Appendix G is listed 6 additional papers, where I am a co-author and have contributed to the paper by either writing parts of the text, doing some of the numerical calculations or where the algorithms and implementations described in this thesis were used by others for analyzing helioseismic data. Please notice that Appendix E is a revised version of the part I contributed to the EEC Esprit deliverable listed in Appendix G.

Appendix A: R. M. Larsen, *Lanczos bidiagonalization with partial reorthogonalization*, Aarhus University, DAIMI PB-357, September 1998.

The appendix contains an extended version of a technical report written over the summer of 1998 and in some sense distills the knowledge and experience I have acquired during this project in the use of Lanczos algorithms. It provides an overview of the theory for Lanczos algorithms in connection with sparse SVD and least squares problems, and deals especially with the question of how to handle the effects of finite precision arithmetic. A new algorithm for incorporating partial reorthogonalization in the Lanczos bidiagonalization algorithm is presented, and through a series of numerical experiments in Matlab it is demonstrated that the method is robust and often 2-3 times faster in terms of the number of floating point operations than current state-of-the-art algorithms for sparse SVD. Just as important is the fact that this efficiency is obtained without sacrificing the accuracy of the computed singular values. A number of issues that might affect the robustness of such an implementation are carefully examined, and in this connection I have discovered a bug in the widely used LANSO package, which caused the algorithm to fail when asked to compute very small singular values. This misbehavior has been reported by other authors (Eldén & Sjöström (1996) [71]). The bug has been reported to the authors of LANSO and will be corrected in a future version. One part of the report deals with Lanczos algorithms for sparse least squares problem. In particular, I describe algorithms for problems with multiple right-hand sides, which have received relatively little attention so far in the literature, and a number of new algorithmic ideas are presented, some of which are of special interest when solving discrete ill-posed problems with multiple right-hand sides. These algorithms will be an important topic in my future work. I intend to submit this work to SIAM Journal on Scientific Computing or a similar journal, possibly as two separate papers; one about the sparse SVD part and one about algorithms for large sparse discrete ill-posed problems with multiple right-hand sides. The report also describes a Matlab software package PROPACK for sparse SVD and sparse symmetric eigenvalue problems based on the developed Lanczos algorithms with partial reorthogonalization, which I have implemented and intend to make available as a public domain Matlab toolbox. As indicated by the author list, all programming, calculations and writing was done by me.

Appendix B: R. M. Larsen and P. C. Hansen, *Efficient Implementations of the SOLA Mollifier Method*, Astron. Astrophys. Suppl., **121** (1997), 587–598.
This paper, which was written in collaboration with Per Christian Hansen, introduces the fundamental algorithms for SOLA inversion based on the nullspace method and bidiagonalization. The paper introduces both direct algorithms based on Householder bidiagonalization, and an iterative variant based on the LSQR algorithm. The results in this paper lay the foundation on which many of the subsequent algorithms (e.g. generalization to MOLA, incorporating the Kronecker products) and implementations have been built. A comparison with different implementations based on the normal equations and Lagrange multipliers is made, and pointers are given to subroutines from the most common software libraries, which can be used to implement the main steps in the algorithms. I did the majority of the writing and all of the calculations.

Appendix C: R. M. Larsen, *Iterative algorithms for two-dimensional helioseismic inversion*, in Proc. of the Interdisciplinary Inversion Workshop 5 (Ed. B. H Jacobsen), Dept. Earth Sciences, Aarhus University (1997), 123–138.
This paper introduces the block Kronecker product structure that arises from discretization of the kernels in the 2-dimensional helioseismic rotational inversions, and is thus central to this thesis. The performance of the three implementations outlined in Chapter 4 is compared for a number of different helioseismic datasets, and the time/space trade-off between the hybrid Lanczos algorithm and an implementation based on the LSQR algorithm is discussed.

Appendix D: R. M. Larsen, J. Christensen-Dalsgaard, A. G. Kosovichev and J. Schou, *Improved SOLA Inversions of MDI Data*, In Structure and Dynamics of the Interior of the Sun and Sun-like Stars (Eds. S. G. Korzennik & A. Wilson), ESA SP-418, ESA Publications Division, Noordwijk, The Netherlands, 1998, in the press (6 pages).
This paper describes an attempt to improve the reliability of the SOLA inversion of MDI data, especially at high latitudes. This is done by modifying the target functions to match the behavior of the kernels for rotational splitting coefficients near the rotation axis. A test of the robustness of the improved method is carried out by inverting artificial data, and this shows that the effective resolution is improved near the pole, which allows us to assess more reliably the nature of the high-latitude features seen with other inversion methods (see [175]). The effect of incorporating information about error correlations in the data was also examined, and more importantly a comparison of the 144 and 360 day MDI sets was made, showing significant differences at the position of the so-called jet discussed in [175]. The paper was written in collaboration with Christensen-Dalsgaard, based on a poster which I wrote for the SOHO 6 conference. All calculations were carried out by me; Schou provided the splittings and error-correlation data, while Kosovichev contributed to the general discussions on how to improve the method, and the interpretation of the two different datasets.

Appendix E: R. M. Larsen & P. C. Hansen, *Parallel implementation of Lanczos bidiago-*

nalization on Convex Exemplar, 1998.
This appendix contains a revised version of the EEC Esprit Basic Research Action deliverable "Parallel Iterative Methods for Nonsymmetric Large-Scale Problems" written in collaboration with A. Basermann, M. Bücker and P. Weidner and P. C. Hansen. The original paper was divided into two separate parts: 1) Quasi-Minimal Residual Methods and 2) Lanczos Bidiagonalization, of which I was responsible for the latter. The appendix contains a revised version of this part of the paper, which was primarily written by me. The revised version describes an regularization method based on Lanczos bidiagonalization without reorthogonalization, where the spurious and extra multiple copies of the singular values are removed using the techniques developed by Cullum and Willoughby [56]. The paper describes how the Lanczos algorithm was implemented on the Convex Exemplar parallel multiprocessor, and present a restructured formulation of the Lanczos bidiagonalization, which allows the number of cache misses to be halved compared to the usual formulation of the algorithm. A model of the parallel performance is established and compared with measurements. I was responsible for all programming and calculations.

Appendix F: R. M. Larsen, *Experiments with tuning the spectral bisection algorithm* This appendix summarizes the results from an experiment with optimizing the classical spectral bisection algorithm for the symmetric tridiagonal eigenvalue problem from [21]. The algorithm has been implemented on a number of different computer platforms, and we investigate the benefit of using optimizing compilers. Finally we show how to improve the performance of the implementation by making manual modifications to the code, guided by knowledge of the architecture of the CPU/FPU system.

Appendix A

Lanczos bidiagonalization with partial reorthogonalization

Appendix B

Efficient implementations of the SOLA mollifier method

Appendix C

Iterative algorithms for two-dimensional helioseismic inversion

Appendix D

Improved SOLA inversions of MDI data

Appendix E

Parallel implementation of Lanczos bidiagonalization on Convex Exemplar

Appendix F

Experiments with tuning the spectral bisection algorithm

Appendix G

Other papers

1. J. Christensen-Dalsgaard, R. M. Larsen, J. Schou & M. J. Thompson, 1995. *Optimally localized kernels for 2D helioseismic inversion*, In *Proc. GONG'94: Helio- and Astero-seismology from Earth and Space*, eds Ulrich, R. K., Rhodes Jr, E. J. & Däppen, W., Astronomical Society of the Pacific Conference Series, San Francisco, **76**, 70 – 73.

2. A. Basermann, M. Bücker, P. Weidner, P. C. Hansen and R. M. Larsen, *Parallel Iterative Methods for Nonsymmetric Large-Scale Problems*, APPARC PaA5a Deliverable, Esprit BRA III Contract # 6634, 1995.

3. J. Schou, H. M. Antia, S. Basu, R. S. Bogart, R. I. Bush, S. M. Chitre, J. Christensen-Dalsgaard, M. P. Di Mauro, W. A. Dziembowski, A. Eff-Darwich, D. O. Gough, D. A. Haber, J. T. Hoeksema, R. Howe, S. G. Korzennik, A. G. Kosovichev, R. M. Larsen, F. P. Pijpers, P. H. Scherrer, T. Sekii, T. D. Tarbell, A. M. Title, M. J. Thompson and J. Toomre, *Helioseismic Studies With SOI-MDI of Differential Rotation in the Solar Envelope*, Astrophys. J. **505** (1998), 390–417.

4. J. Schou, J. Christensen-Dalsgaard, R. Howe, R. M. Larsen, M. J. Thompson, J. Toomre, *Slow poles and shearing flows from helioseismic observations with MDI and GONG data spanning a year*, In Structure and Dynamics of the Interior of the Sun and Sun-like Stars (Eds. S.G. Korzennik & A. Wilson), ESA SP-418, ESA Publications Division, Noordwijk, The Netherlands, 1998, in press.

5. P. Charbonneau, J. Christensen-Dalsgaard, R. Henning, R. M. Larsen, J. Schou, M. J. Thompson, & S. Tomczyk, *Helioseismic constraints on the structure of the solar tachocline*, Astrophys. J., 1998., submitted.

In this paper the SOLA algorithm is used as well as a forward modelling technique based on a genetic algorithm for global optimization (see Charbonneau [38]). The forward modelling technique uses the Kronecker product formulations to compute the integrals of the modekernels and a parametrized rotation profile, and determines a non-linear least-squares fit to the measured splitting coefficients. Using the notation of Section 3.3.1 the calculation of the integrals corresponds to computing the matrix-vector product $\mathbf{Kx}$, where vector $\mathbf{x}$ simply contains samples the parametrized rotation profile on the discretization grid. Hence, the evaluation of the objective function in the global optimization essentially consist of a call to (a slightly modified version of) subroutine KRONMV.

These techniques where applied to data from the LOWL instrument to study to structure of the solar tahocline. In particular the location and thickness.

6. W. J. Chaplin, J. Christensen-Dalsgaard, Y. Elsworth, R. Howe, G. R. Isaak, R. M. Larsen, R. New, J. Schou, M. J. Thompson, & S. Tomczyk, *Rotation of the solar core from BiSON and LOWL frequency observations*, MNRAS, 1998, In preparation.