Inverse Problems in Helioseismology

by

Richard K. Barrett B.Sc.

Thesis
submitted to the
University of Glasgow
for the degree of
Ph.D.

Astronomy and Astrophysics Group,
Department of Physics and Astronomy,
University of Glasgow,
Glasgow. G12 8QQ.

December 1994

© Richard K. Barrett 1995

ProQuest Number: 13818827

All rights reserved

INFORMATION TO ALL USERS
The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript
and there are missing pages, these will be noted. Also, if material had to be removed,
a note will indicate the deletion.

ProQuest 13818827

Published by ProQuest LLC (2018). Copyright of the Dissertation is held by the Author.

All rights reserved.
This work is protected against unauthorized copying under Title 17, United States Code
Microform Edition © ProQuest LLC.

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Thesis
9987
Copy

GLASGOW
UNIVERSITY
LIBRARY

With thanks to my parents.

Acknowledgements

This thesis was conducted under the supervision of Professor John C. Brown, and funded by the United Kingdom Science and Engineering Research Council. It is, I think, a tribute to my powers of endurance that I lasted longer than they did.

My research would not have been possible without the advice, support, infinite patience and good-will of John Brown. My research visits to the USA and Germany would not have been possible without his support. His encouragement helped me through times when I was all but ready to give up.

Throughout my Ph.D. research I have constantly sought the advice of all members of the Astrophysics group, and I would like to thank them all for their kindness, in particular, Dr Shashi Kanbur for running the computer system, Dr Norman Gray for knowing everything, especially about LaTeX (and words – whenever I was stuck for a word, he always had), as well as Andrew Newsam for his guidance in matters pictorial: without him, figures 3.1 and 3.2 would not have been possible.

In the 'early years' of my thesis, life was made memorable (although much more difficult to remember) by the friendship and drunken debauchery of Geoff, Moray, Fraser, Jaber, Martin and Colin. During the 'post-wilderness' years, I forged many new friends in the Astrophysics group. Thanks must, therefore, also go to Lorna (more drunken debauchery), Dr Keith (concern, encouragement and lending me his versions of the LaTeX style files) and Aidan (concern and encouragement above and beyond the call of duty) for all their concern, encouragement and, of course, drunken debauchery. Special thanks go to Lorna for providing me with a good dinner whenever I needed one, and for drowning the thesis blues from time to time with a good night out.

On a more scientific note, I would like to thank Ken Libbrecht for providing me with the data used in the inversions performed in this thesis, and Jørgen Christensen-Dalsgaard for supplying the solar model from which the kernel functions for the rotation inversions were calculated.

Finally, the most special thanks of all must be reserved for Ute Dapprich, whose assistance, materially, spiritually, linguistically and typographically, have made possibly the biggest contribution to this thesis, and to my well-being (although not typographically, of course).

Summary

The work of this thesis is concerned with investigating and improving regularized least squares (RLS) inversion methods (see §1.9), with particular regard to the application of these techniques to the inverse problems that arise in helioseismology. Many different aspects of RLS methods will be addressed, including

- the development of an improved and generalized algorithm for *discretizing* the inverse problem, that is, reducing the continuous (integral) problem to a discrete (matrix) problem suitable for numerical solution (see chapter 2),
- the assessment of the quality of the recovered solution, particularly as far as the resolution achieved is concerned (chapter 4),
- the investigation of the opportunities for optimizing the method of solution, by finding an algorithm for choosing the regularizing parameter in an optimal way (chapter 3).

Throughout this thesis the problem of recovering the solar internal rotation from helioseismic data will be used to exemplify the ideas and techniques considered.

Chapter 1 of this thesis reviews the basic principles and characteristics of inverse problems, and describes the ideas underlying the most commonly used methods for obtaining solutions of inverse problems. In particular, the various manifestations of RLS methods used to date will be described, as a prelude to chapter 2, where they will be unified into a single RLS procedure. The form of the inverse problems in helioseismology is also described.

Chapter 2 motivates and expounds a general algorithm for performing RLS inversions, which unifies the RLS methods that existed previously and are reviewed in §1.9. The reasons for developing such an algorithm are presented in detail in §2.1, but the important point is that being able to 'access' any RLS method from within a single inversion algorithm

has considerable advantages, both as far as the simplicity of using the procedure to perform inversions is concerned, and for comparing the effectiveness of different discretizations. The generality of the discretization procedure makes the introduction of constraints on the recovered solution desirable, and requires the development of a much more robust numerical method for calculating the solution of the resulting matrix problem. These issues are dealt with in sections 2.3 and 2.4, respectively.

Chapter 3 looks at optimal methods for choosing the smoothing parameter in RLS inversions. It is well known that one of the vital aspects of solving ill-posed problems is choosing some acceptable *trade-off* between the effects of the random data errors on the solution and the bias introduced by 'smoothing' the recovered solution to reduce the effect of these errors. RLS inversion methods contain a free parameter, the *smoothing parameter* (λ), for controlling exactly this trade-off. When λ is small there is very little smoothing of the solution, so bias is not a problem, but the instability of the problem will give rise to huge errors in the value of the solution at any point, which is unacceptable. When λ is very large, on the other hand, the effect is to make smoothing the dominant effect on the solution. The effect of data errors on the solution is eliminated, but the smoothing gives rise to a huge bias which renders the solution meaningless. Somewhere between these two extremes there must lie a compromise value of λ that reduces the effect of the errors to an acceptable level while keeping the bias resulting from smoothing as small as possible. There are a number of ways to make this compromise, and chapter 3 examines the advantages of two *automatic* methods for choosing the smoothing parameter. Automatic here means that they use the data for the problem to select the smoothing parameter, rather than making some selection at the outset without regard to the data.

Another vital part of inverse theory is the quantification of 'resolution'. This essentially corresponds to determining the length of the solution that must be averaged over to reduce the effects of data errors to an acceptable level, as discussed above. There are two sides to this. Firstly, there is the best resolution that could be obtained in any inversion, and this can be assessed without regard to the details of particular inversion procedures. Secondly, there is the resolution that actually has been achieved in a particular inversion, and these need not necessarily be the same at all. Chapter 4 involves an examination of the extent to which features in the solution (the rotation rate, say), such as steps or delta-functions, can be distinguished and quantified for different noise levels. This provides information about the best possible resolution obtainable in any inversion, but it also has physical significance when considering the solar rotation inversion. For example, the structure of, and, in particular, the radial gradient in, the rotation profile near the base of the solar convection zone is important in dynamo theory. The ability to differentiate between different steps in the rotation rate (that is, if a step is parametrized by its position in radius r(step) and its 'height', h(step), the ability to resolve different values of r(step) and h(step)) would provide information about the nature of the solar dynamo. Knowing the resolution of the data tells us how much we can constrain any dynamo theory. Chapter 4 also presents other 'inversion' dependent methods for determining the resolution achieved in a particular RLS inversion, one of which (correlation length – see §4.2.5) seems to have been largely overlooked before.

Chapter 5 presents the results of applying some of the techniques considered in this thesis to real data. In particular, the results of inversions using the algorithm of chapter 2 with different discretizations are given to indicate the generality of the discretization procedure. Finally, the splitting data of Libbrecht (1989) is inverted to obtain the angular velocity throughout the solar interior.

Chapter 6 discusses possible improvements and extensions to the work and ideas contained in this thesis.

Contents

Chapter 1

Inverse Problem Review

1.1 Introduction

The discovery in 1962 of an oscillatory component of the solar surface velocity field (Leighton *et al.* 1962) and its subsequent identification as the surface manifestation of global, non-radial, acoustic eigenmodes of oscillation (Ulrich 1970; Leibacher and Stein 1971) constituted a significant advance in the study of the solar internal structure, and, consequently, in the study of stellar structure in general. This is because the frequencies of the observed eigenmodes contain a great deal of information about conditions in the solar interior, and so can be used to infer the pressure and density, for example, throughout the sun. The exercise of making such inferences is a typical example of a type of problem common in astronomy, known as an inverse problem (see Craig and Brown 1986).

It will be useful throughout this chapter to have concrete examples of the various types of inverse problem to illuminate certain points. The following section provides these examples, and uses the presentation to introduce the most important features of general inverse problems. Section 1.3 discusses the significance of the term 'inverse problem', and section 1.4 describes the difficulties associated with the practical solution of inverse problems, and how these difficulties may be alleviated. After that, it is demonstrated heuristically in §1.5 how the inverse problems of helioseismology arise, how the oscillation mode frequencies contain information about the solar structure, and a simple demonstration of how this information can be extracted from the mode frequencies is given. Some aspects of the problem of obtaining a numerical solution are then considered, before sections 1.8 and 1.9 present the commonly used techniques for solving linear inverse problems, as a precursor to chapter 2, where an algorithm unifying several of these methods is formulated.

1.2 Examples of Inverse Problems

The three inverse problems described below illustrate the three forms that commonly occur in astronomical (and other) problems. Which form applies to a particular situation depends on whether the data or the unknowns are discrete, or are functions of a continuous variable. The three cases also provide an illustration of the method for solving inverse problems of type I or II: reduce type I problems to type II by *discretizing* the problem in the 'data space', reduce type II problems to type III by discretizing in the 'solution space', then solve the type III problems by matrix methods. This is described in more detail in §1.9. Note that all three problems are *linear* inverse problems (the data depend linearly on the unknowns). This makes it easier to see the principles involved.

Case I. ***Integral Transforms – Image Processing (Theory):*** Images play an important role in many areas of science and technology, and particularly in astronomy. The process of forming an image typically involves using lenses or mirrors to collect and focus the light coming from an object onto a flat (usually square or rectangular) detector, which records the amount of light arriving at any point within it. The goal of the focussing is to ensure that light emitted from a single point on the object arrives at a single point on the detector (the image is sharply focussed), and that the positional relationship between parts of the object and their corresponding points on the detector is preserved (the images are not distorted). The goal of the detector is to record every photon striking it perfectly. Here the detector will be assumed to be perfect and continuous, in the sense that it can respond to arbitrarily fine detail in the image falling on it. (Cases II and III show the treatment of the problem in practical situations, where the detector is composed of finite-sized *pixels.*) If such a perfect imaging system could be created, the amount of light recorded striking any point in the detector (the brightness at that point) would truly indicate the amount of light leaving the corresponding point on the object, and so any detail in the structure of the object would be recorded perfectly – measuring the image would be as good as measuring the light from the object directly. However, there are physical reasons why such a perfect imaging system cannot exist (diffraction – see case II), and lenses and mirrors are never perfect, so some of the light from a particular part of the object will not fall on the 'correct' part of the detector (the image may be out of focus, or light may be scattered away from its direct path by the material in the lenses or by air along the ray-path of the light). This means that a point-like object (such as a star), whose light should fall on a single point, will actually illuminate a larger area (perhaps all) of the detector. If the imaging system is good, the light should mostly end up very close to the correct point, giving only slight blur, but the worse the focussing or scattered light is, the more the light will hit parts of the detector away from the correct point, and the more blurred the image will become. The well known problem with the Hubble Space Telescope (HST), was due to an incorrectly shaped mirror which resulted in the images being blurred.

The distortions introduced by an imperfect imaging system can be completely described by considering their effect on the light from point sources. Assume that the detector is rectangular of size a cm $\times$ b cm. An image is characterized by the brightness, $B(x,y)$, at all points (x,y) of the detector. Imagine that a point source is being imaged, and that if the focussing were perfect all light would arrive at the point (x_0, y_0), so that the image would be $B_0(x,y) = B_*\delta(x - x_0, y - y_0)$, where B_* is the intrinsic brightness of the point source and $\delta(x,y)$ is the two-dimensional dirac delta-function. (It is convenient to identify the point (x_0, y_0) *on the detector* with the corresponding point on the object, so a reference to 'light emitted from (x_0, y_0)' really means 'light emitted from the point on the object corresponding to the point (x_0, y_0) on the detector'. This will simplify some of the explanations below.) The imperfections of the real imaging system will result in light falling on parts of the detector near (x_0, y_0), giving a blurred image instead of a single point – the point of light has been spread out over the detector. The resulting brightness, $B_{(x_0,y_0)}(x,y)$, at all points of the detector, gives rise to a function $K_{(x_0,y_0)}(x,y)$, defined by $K_{(x_0,y_0)}(x - x_0, y - y_0) = B_{(x_0,y_0)}(x,y)/B_*$, called, naturally enough, the *point spread function*, or PSF. The PSF says exactly how the light from a unit source at (x_0, y_0) is spread out over the detector. Figure 1.1 shows typical PSFs for two different amounts of blur. Note that the wider PSF has a much lower peak value, reflecting the fact that light is simply redistributed by the blurring (the 'volume' under the PSF is always unity). Figure 1.1 illustrates very nicely the effect of the corrector that was used to reduce the HST's blur. The wider PSF corresponds to the HST without the corrector (quite blurred images and a very much reduced maximum brightness), and the narrow PSF indicates the image quality that is now achieved. The effect of this decrease in the blur is a dramatic improvement in the performance of the telescope.

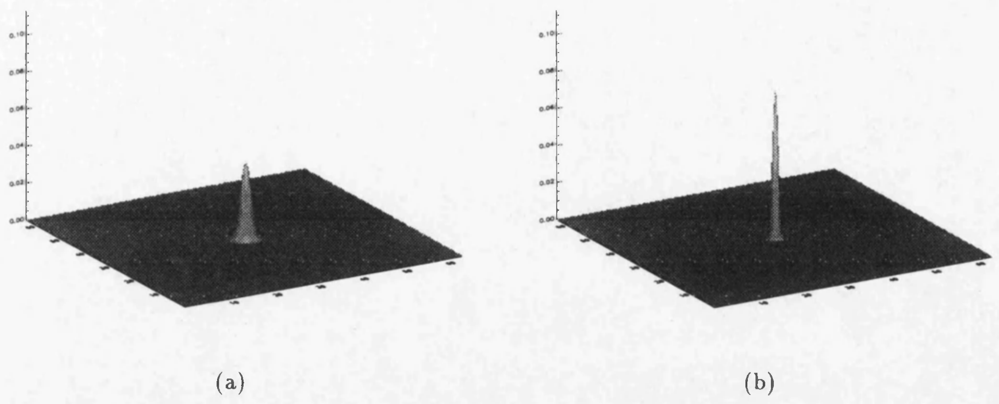


Figure 1.1: Two (gaussian) point spread functions of different widths showing how light from a point source is spread out in an out-of-focus image. The horizontal axes are detector co-ordinates, (x, y), and the vertical axis displays the brightness, scaled so that the perfect image would be a spike of unit brightness. Image (a) has high blur, whereas image (b) has much lower blur, and the image is quite sharp.

In general, the PSF depends on (x_0, y_0), the position of the 'perfect' point image, i.e. the distortion of the image varies across the detector. However, it is often adequate to assume that the distortion is constant across the image. This assumption will be made here for simplicity, and the subscript (x_0, y_0) on the PSF will be dropped. The PSF is often approximated by a two-dimensional gaussian, which has the nice property of falling away smoothly to zero from its peak value. The width of the gaussian is related to the amount of blur in the image. Note that the blurring need not be isotropic (corresponding to a rotationally symmetric PSF), although it often is.

Since a real (extended) object is made up of 'lots of points' the final distorted image will consist of the distorted images of all the single points in the object added together, giving $B(x, y) = \iint_{\text{detector}} B_{(x_0,y_0)}(x, y) dx_0 dy_0$, or, using the definition of the PSF and calling the intrinsic brightness of the 'point of light' at (x_0, y_0) $B_*(x_0, y_0)$,

$$B(x, y) = \iint\limits_{\text{detector}} K(x - x_0, y - y_0) B_*(x_0, y_0) dx_0 dy_0. \tag{1.1}$$

The function $B(x, y)$ gives the brightness of the blurred, imperfect image at the position (x, y) on the detector, whereas $B_*(x, y)$ is the brightness the sharply focussed, *perfect* image would have at (x, y). Obviously, only $B(x, y)$ can be measured, but what we really want to know is the 'true' image, $B_*(x, y)$, since this is effectively equivalent to the object.

Is it possible to get B_* from B? The answer to this question depends on the nature of the PSF, but, in theory, it may be possible, depending on the form of the PSF. In practice it can be very difficult, for the reasons described in §1.4. The process of finding B_* given B (i.e. solving (1.1) for $B_*(x, y)$) is a well known inverse problem. It is the principal problem studied in *image processing*, the goal of which is to extract as much information as possible from a given image. In a perfect world (the situation considered here), the blurred image contains exactly the same information as the perfect image, except that instead of being given in terms of 'points of light' it is made up of blurred patches of light. It is possible to envisage taking each blurred patch (which will be 'PSF-shaped'), gathering all the light it contains together, and placing it at the correct point (the centre of the PSF). This is, in essence, what solving (1.1) involves, although the mathematical formalism often clouds the simplicity of the principle. Images from the (uncorrected) Hubble Space Telescope provide a nice example of image processing, because although the mirror's curvature was hopelessly wrong, it was hopelessly wrong to a very high degree of accuracy, so it was possible to predict where the misdirected light would end up, which meant that the PSF could be calculated theoretically.

The idea of information (light, in this case) being distorted, spread around and generally 'sent to the wrong place' is central to the concept of inverse problems. In general, it is not a trivial matter to understand and visualize this distortion, but images provide a simple paradigm. The fact that the distorted image is the sum of infinitely many 'blurred points' allows the errant light to be rounded up and put back where it should have been in the first place. In image processing problems it is easy to understand and develop a mental picture of this process. This is not so easy in helioseismology, where the kernel functions do not have the simple form of those in fig. 1.1, although see §1.5.2.

Equation (1.1) is an example of a (2-D) integral transform, where one function (here, $B_*(x_0, y_0)$) is mapped to another ($B(x, y)$) by an integral operator. In fact, (1.1) is a *convolution* (*cf.* Craig and Brown 1986, equation (2.7)). If the distortion had been allowed to vary across the image a more general type of integral transform would have arisen, and the point spread function would be $K(x, y; x_0, y_0) \equiv K_{(x_0,y_0)}(x - x_0, y - y_0)$. A great many inverse problems have these forms, and such problems provide examples of most of the difficulties generally associated with the solution of inverse problems, such as instability to perturbations in the observed quantities (again, see §1.4, and Craig and Brown 1986).

This thesis deals almost exclusively with methods for solving one-dimensional inverse problems (that is, problems where the solution function depends only on one variable, unlike in (1.1), where $B_*(x_0, y_0)$ depends on two – x_0 and y_0). Equation (1.1) is therefore not a very convenient example to use. It is quite easy, though, to imagine a 1-D problem analogous to (1.1). If the detector is now assumed to be a straight line instead of a rectangle, and the focussing system and object are similarly idealized, then exactly the same principles apply as before, but now equation (1.1) is

$$B(x) = \int K(x - x_0) B_*(x_0) dx_0. \tag{1.2}$$

As this thesis concentrates on 1-D inverse problems, this idealized image processing problem will be used as an example of inverse problems involving an integral transform.

Case II. ***Functional Constraints – Image Processing (Realistic Approach) and Helioseismology:*** In the problems that occur in the 'real world' (that is, involving real data, taken with real instruments) there can only ever be a finite amount of data available, not least because just to store an infinite amount of information would require an infinite amount of 'disc space', which is obviously unobtainable. Even actually acquiring an infinite amount of data (recording, using, then discarding data as it comes in, without storing it all for later use) is impossible, though, because it would require an infinite rate of data transfer to acquire an infinite amount of data in a finite time. This means that equations like those presented in case I do not apply literally to real problems. (But they are very useful for considering the mathematical idealizations of real problems. Such idealizations avoid many of the trivial complications involved in formulating and solving problems in realistic situations, such as the specific details of the observing instrument with its inevitable biases and other idiosyncrasies, and therefore allow the fundamental principles involved to be seen more clearly.) For example, the image processing inverse problem described in case I requires an infinite amount of data (the values of $B(x, y)$ for all points (x, y) of the detector). Detectors capable of making such measurements do not exist. Generally, detectors consist of a (square or rectangular) array of finite-sized pixels – a measurement then consists of finding and recording the total amount of light falling on each pixel. Figure 1.2 shows the images of a point source for different amounts of instrumental blur (different PSF widths) taken with such a detector. If we assume that the pixels are indexed by the pair of integers (i, j) for $1 \leq i \leq n_x$ and $1 \leq j \leq n_y$ (n_x

and n_y are the numbers of pixels in each row and column, respectively, of pixels in the detector), then the data actually acquired by this detector is

$$\{B_{ij} \equiv \iint\limits_{\text{pixel }(i,j)} B(x,y)dxdy; 1 \leq i \leq n_x, 1 \leq j \leq n_y\}.$$

Equation (1.1) then becomes

$$B_{ij} = \iint\limits_{\text{detector}} K_{ij}(x_0,y_0)B_*(x_0,y_0)dx_0dy_0, \quad \text{for all } i,j, \tag{1.3}$$

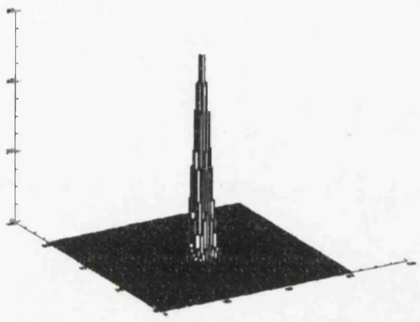
where the point spread function has become a collection of $n_x \times n_y$ functions

$$K_{ij}(x_0,y_0) \equiv \iint\limits_{\text{pixel }(i,j)} K(x-x_0,y-y_0)dxdy,$$

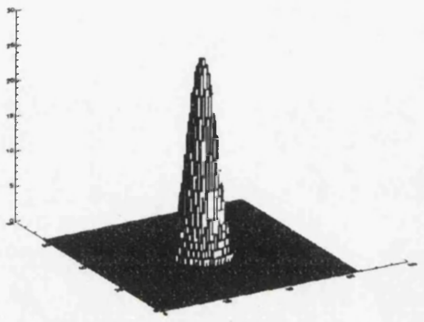
which determine the proportion of the light emanating from the point of light at (x_0, y_0) that arrives at pixel (i, j). The functions K_{ij} are known as *kernel functions.* In actual fact, the form of the image processing problem given in (1.3) is not the form that is usually used in practice. That form is described in case III.

Note that the full (2-D) form of the helioseismic rotation problem (see Ritzwoller and Lavely 1991) is of exactly the same form as (1.3). In this case, however, the data for the problem (the splitting between the frequencies of modes with different azimuthal order, m, but the same radial order, n, and spherical harmonic degree, l) are intrinsically discrete: the solar eigenmodes are 'countable' things, so there is no continuous generalization of the helioseismic problem. The theoretical formulation of this helioseismic forward problem automatically has discrete data.

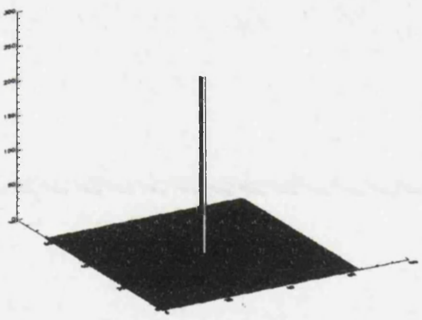
Equation (1.3) comprises a collection of linear *functionals* of the true image, B_*. For realistic data sets there will only be a finite number of data values (number of pixels in the detector), but the true solution is a function of continuous variables, and so has an infinite number of degrees of freedom. Common sense suggests, therefore, that 'solving' (1.3) is impossible. This is correct, but is not the end of the matter. Essentially, the problem is (infinitely) underdetermined, so a unique solution does not exist. It is reasonable to hope, though, that this lack of uniqueness, being entirely the result of the finite resolution in the observed image (finite pixel-size), is reflected only in an inability to determine the corresponding small-scale features of the true image. In other words, information about the features in the true image that are larger than the pixel-size is contained in



(a)



(b)



(c)

Figure 1.2: The effect of using a detector consisting of (64 × 64, in this case) finite-sized pixels to observe the images shown in figure 1.1 is illustrated. Images (a) and (b) correspond to those in figure 1.1. The images are effectively 'binned' by the detector, the height of the block corresponding to any bin gives the amount of light arriving at that bin. The 64^2 numbers corresponding to the brightness at each pixel is the only information obtainable by this detector. Image (c) shows the (hypothetical) perfectly focussed image of the point source taken with this detector. Note that all that can be said is that all the light falls on a single pixel, but it is not possible to see the image as a true 'point'.

the observed image, and it is possible to choose one of the infinity of possible solutions as 'the solution', safe in the knowledge that the large scale features in this solution do indeed reflect properties of the true image. Any short-scale components of the true solution that cannot be contained in the image might as well be assumed to be zero (as good a value as any) as they are completely undetermined. This enables the (distorted) information about the true image that is contained in the data to be extracted – the nearest it is possible to get to 'solving' (1.3). These considerations apply to the helioseismic problem too, but it is more difficult to see whether it is really the case that only information about small scale features is missing from the data. In general, this will not be the case, but the nature of the observations usually allows this to be assumed. The veracity of this assumption in general is examined in Backus and Gilbert (1968), and in §1.4. The important point to note is that the finite number of 'pieces of information' in the data inevitably imposes a limit on the extent to which small features in the true image can be seen: a limited *resolution* is unavoidable. For the solar rotation problem this means that, for example, it may not be possible to recover the details of the solar rotation profile near the base of the convection zone (a region of particular significance as far as solar dynamo theory and theories of the solar cycle are concerned) to sufficient accuracy with present data, but it may be possible with the data to be obtained by the GONG project (Harvey *et al.* 1993), which will measure more mode frequencies (and will measure them more accurately).

Again, the two-dimensional form of the inverse problem (1.3) renders it unsuitable to be used as an example of the type of problems to be studied in this thesis. It is a very simple matter, though, to apply the same reasoning to the 1-D problem (1.2) (n_x pixels, each of which is a short segment of the original one-dimensional detector). The result is

$$B_i = \int_{\text{detector}} K_i(x_0)B_*(x_0)dx_0, \quad \text{for all } i. \tag{1.4}$$

Case III. ***Matrix-type Inverse Problems – Image Processing (Practice):*** The discussions of image processing problems presented in cases I and II dealt mainly with the nature of the detector. In realistic situations it is essential to consider the details of the focussing system. For example, even a focussing system constructed with perfect precision has an absolute limit placed on its resolution (i.e. a lower limit on the width of the PSF) because the finite size of the lens or mirror results in *diffraction* of the incoming light, which means that light from a single point on the object can never arrive at a single point on the detector. This is a fundamental physical constraint: the only way to reduce the effect of diffraction is to make the focussing system larger (bigger lenses and mirrors).

At this point it is necessary to introduce the concept of *data errors* or *noise.* These are *random* errors on any item of data, about which nothing is known except their probability distribution (and sometimes not even that). Clearly, any real measurement or observation must involve such errors, not least because nothing can be measured with infinite precision. Note that these errors are of a completely different nature from the distortions introduced by, say, poor focussing in the image processing problem. The description of focussing is entirely deterministic and, given an object, calculable, whereas noise is always stochastic and will be different for two different images of the same object. Errors play an absolutely vital role in the solution of inverse problems. Section 1.4 describes their importance in more detail, but here it is enough to make a few general remarks. It turns out that often the solutions of inverse problems vary very sharply as the data changes, so that small data errors can give rise to very large errors in the solution. It is usually the case that these errors are manifested as large variations in the solution over very short length scales (so that, in image processing problems, nearby parts of the solution image, B_*, will have very different brightnesses, due to these errors), and that the errors tend to average to zero over longer scales (providing the true image varies more slowly than the error component, taking local averages of the noisy solution over lengths rather longer than the length scale of the noise variations, should give a good estimate of the true solution). Of course, such an averaging process inevitably decreases the resolution achieved: features in the *true* solution whose typical scale of variation is less than the averaging length will tend to average to zero, so that such small scale features will not appear in the solution. (Alternatively, any sharp features in the image will be smeared out by the averaging process over a length equal to the resolving length.) The larger the errors are, the longer the averaging length must be to ensure that the errors average to a sufficiently small value (the errors from more points must be averaged to reduce the resulting error average to a small enough value). The practical implementation of this averaging process takes several forms, but the result is that variations of the true image over length scales shorter than some limit set by the errors cannot be resolved. This is a very important point. For problems of the form given by equations (1.1) or (1.2) it is often the case that there is no limit on resolution imposed by the width of the PSF for *perfect, error-free* data: the solution is (in principle, at least) unique and exact. It is the data errors that reduce the resolution achievable, because their stochastic nature removes the possibility of making absolute statements about the solution. Instead, all such statements must be qualified by phrases such as 'to such-and-such a level of statistical significance'. But the ability to resolve features with a characteristic length, L, say, means being able to say 'the observed variations of the solution over lengths L are significant at the 95% level' (or whatever significance level is appropriate). For L less than some value (the resolution length) it is no longer possible to make such statements to the required level of significance because even after averaging the solution over regions of size L the averaged errors are still large enough to dominate the variation of the solution. It should be noted that the width of the PSF, while not imposing a resolution limit in itself, does very much determine the extent to which any level of noise limits the obtainable resolution: a wide PSF gives poorer resolution than a narrow PSF *for the same level of data noise.*

The image processing problem described in cases I and II has been shown to have a finite-sized PSF, even with a perfect focussing (because of diffraction). It is also inevitable that there are data errors (that is, errors on the value of 'brightness' recorded at any pixel). This is because of the quantum nature of light. The individual photons in the incoming light have random arrival times, and the brightness at any pixel merely reflects the likely number of photons arriving per unit time. The *actual* number of photons arriving may vary randomly about this likely value, giving an error in the measurement of the brightness of that pixel. In the helioseismological problem there are errors in the data (measured mode frequencies) which are, in part, due to similar statistical considerations. The random excitation of any acoustic mode by turbulent convection gives rise to a power spectrum (of the variation of mode amplitude with time) which also has a stochastic nature. Measuring the mode frequency corresponds to finding the position of the peak in this power spectrum (Anderson *et al.* 1990), so this measurement will obviously be affected by the stochastic nature of the power spectrum.

The combination of a finite-sized PSF *and* data errors results, inevitably, in a limited resolution in the image processing problem: there will be a length scale below which variations in the true image cannot be resolved. There is one advantage to this, though, because the inevitable occurence of limited resolution means that it is not necessary to attempt to recover arbitrarily small features in the image. Provided the pixels in the detector are smaller than the limiting resolution determined by the noise level, it is reasonable to adopt the attitude that some average value of the solution over each pixel is adequate to represent the available information about the true solution. In practice this means that instead of solving the integral equation (1.1), or the collection of functionals (1.3), the assumption is made that the true solution is constant over each pixel (or, at least, that the 'recoverable' part of the solution is effectively constant over each pixel), and that finding the value of the brightness, B_{*ij} at every pixel amounts to solving the problem. (This approach is known as *piecewise constant discretization* (PCD), and is described in more detail in section 1.9). Applying this to (1.3) gives

$$B_{ij} = \sum_{k=1}^{n_x} \sum_{l=1}^{n_y} \left(\iint\limits_{\text{pixel } (k,l)} K_{ij}(x_0, y_0) dx_0 dy_0 \right) B_{*kl} \tag{1.5}$$

This method of solution will be adequate as long as the pixels are smaller than the limiting resolution. Note that (1.5) is a linear, homogeneous, algebraic relationship between the data and the unknown brightnesses of the true image – a matrix equation. With a simple relabelling of the pixels by a single index, $(i,j) \to p$ and $(k,l) \to q$, (1.5) has the form

$$B_p = H_{pq} B_{*q} \tag{1.6}$$

where $H_{pq} \equiv \iint_{\text{pixel } (k,l)} K_{ij}(x_0, y_0) dx_0 dy_0$. The importance of this form of the image processing problem is that the well studied methods of linear algebra can be applied to (1.6), enabling the solution to be calculated numerically with considerable efficiency.

During the presentation of cases I, II and III most vital aspects of inverse problems were introduced. In case I the idea of information being distorted and redistributed was described, in case II the inevitability of finite data sets and the effect of this on the solution of inverse problems were considered, and in case III the very important and delicate subject of data errors was broached. However, the presentation above did not deal with the difficulties involved in solving these problems in practical situations. This will be discussed in subsequent sections, but first let us contemplate the significance and relevance of the word 'inverse' in the term 'inverse problems'.

1.3 Why Inverse?

The principal characteristic of inverse problems is that the 'known' (observed or experimentally determined) quantities (which are often of secondary interest) are related to the unknown quantities (of primary importance) in a *non-trivial* way, in the following sense:

> The relationship between two sets of quantities (observables and unknowns, in this case) will be deemed *non-trivial* if the value of any observable depends on the values of all (many, more than one) of the unknowns (and vice-versa), so that the process of calculating the solution of the inverse problem (finding the unknowns) is rather more subtle than solving an equation of the form $y = f(x)$ for the single variable x (or even a series of such problems). In other words, the equations relating the observables and the unknowns are *coupled.*

While hardly mathematically precise, this definition is adequate for the presentation here. It is essentially this coupling that ultimately gives rise to the well known difficulties with solving inverse problems, such as non-uniqueness of solution, instability to small perturbations in the observed quantities, etc., which are discussed in §1.4 and in Craig and Brown (1986), §4.3.

The preceding 'definition' of an inverse problem is philosophical rather than fundamental. In a sense, solving the equation $y = f(x)$ for a given y and a known function f is a very simple type of 'inverse' problem, but it is known that such a problem is not subject to the difficulties associated with problems involving several *coupled* equations in several variables. (Consider the solution of a square matrix equation $\mathbf{y} = A\mathbf{x}$, which is obviously $\mathbf{x} = A^{-1}\mathbf{y}$. The matrix A may be singular, so that its inverse does not exist, but this will not be obvious from looking at the elements of A, whereas the analogous problem in the one-dimensional case – $A = 0$ – is easy to diagnose, and would never occur in a realistic problem, anyway.) It is largely for this reason that it is excluded. The designation 'inverse' arises from a view of the world in which certain quantities are manifestly amenable to observation and measurement, while others are obscured from view, either by virtue of their physical location (such as the interior of the sun), or their intrinsic non-measurability (the spectrum of particle energies in a hot fusion plasma cannot be measured directly because any measuring probe would be vaporized by the extreme temperature). This division is clearly not absolute, but depends on circumstances and the nature of the problem at hand.

For example, a surgeon may consider the organs of the body to be observables in a way that the operator of a CAT scanner would not. Nevertheless, in many situations this division is quite natural. This is particularly true in astronomy, where the only observables are quantities obtainable from observation of the electro-magnetic radiation emitted by objects (and usually with no spatial resolution, so only the integrated light from the object can be measured). Division of the world into 'observables' and 'non-observables' is not the whole story, though, for if the unobservable properties of the universe were unrelated to observable properties they could never be calculated. So, the formulation of an inverse problem requires that science provides a theoretical (mathematical) relationship between some set of observables and another set of non-observables. Without such a relationship it would obviously be impossible to learn anything at all about the unobservable quantities. Let us present this relationship symbolically as follows. Denote the set of observable quantities for a particular problem by $\mathbf{y}$, and the set of non-observables by $\mathbf{x}$. In general, $\mathbf{y}$ and $\mathbf{x}$ are symbols that stand for a number of 'pieces of information'. In the event that both the observables and the number of unknowns are finite, we could write: $\mathbf{y} \equiv \{y_i; i = 1, \ldots, m\}$ and $\mathbf{x} \equiv \{x_j; j = 1, \ldots, n\}$ for some positive integers m and n. However, there is nothing to prevent the observables or the unknowns (or both) from being the values of functions of a continuous (real) variable so that there is an infinite number of pieces of information. Then it would make sense to label each piece of information with that continuous variable, $y(s)$, say, for some real number s, so that $\mathbf{y} \equiv \{y(s); u \leq s \leq v\}$ for example, for some real numbers u and v. As an illustration of this, when the unknowns in an inverse problem are the pressure and density stratifications, $p(r)$ and $\rho(r)$, in the solar structure problem, or the rotation rate as a function of depth, $\Omega(r)$, in the rotation problem, they correspond to an infinite number of unknowns (the value of $p(r)$, say, for any value of radius, r, through the sun is a single unknown). Note that functions of a continuous variable may be thought of as 'vectors' in an infinite-dimensional vector space, so that the continuous variable case is notionally equivalent to the 'discrete' case (y_i, x_j etc.), with m or n infinite. It should be borne in mind that, although the observables ultimately correspond to the data for the problem (and in practical situations there can only be a finite number of pieces of data), the symbol $\mathbf{y}$ may represent an infinite number of pieces of information: the designation 'observables' indicates, in this philosophical context, that they are potentially observable, not that they have actually been observed. The relationship between $\mathbf{y}$ and $\mathbf{x}$ may be written (with sufficient generality) symbolically as

$$\mathbf{G}(\mathbf{y}) = \mathbf{F}(\mathbf{x}), \tag{1.7}$$

where $\mathbf{G}$ and $\mathbf{F}$ represent some *known* functions of the observables and non-observables, respectively. Here the term 'function' is used in its broader mathematical sense: $\mathbf{G}$ and $\mathbf{F}$ are mappings from whatever (vector or function) spaces contain the observables and unknowns, respectively, to some other vector or function space. Equation (1.7) is intended to symbolize quite generally any mathematical relationship derived from a mathematical model of a physical process. Its apparent simplicity therefore disguises its content. The extent to which (1.7) can represent a general theoretical relationship will now be exemplified, and the discussion should clarify the meaning of (1.7).

Again, $\mathbf{G}(\mathbf{y})$ or $\mathbf{F}(\mathbf{x})$ may themselves stand for vectors or functions of a continuous variable, independently of the meaning of $\mathbf{y}$ or $\mathbf{x}$, the only restriction being that the equality in (1.7) forces $\mathbf{G}(\mathbf{y})$ and $\mathbf{F}(\mathbf{x})$ to have the *same* number of degrees of freedom, so that (1.7) is a system of equations. The relevance of this description to inverse problems will be explained below, but, in the meantime, it may be helpful to clarify the meaning of (1.7) by considering some specific examples :

1. Suppose $\mathbf{y}$ and $\mathbf{x}$ are vectors, of lengths m and n, respectively (m data values and n unknown parameters to find). Then there are two classes of problem :

   (a) $\mathbf{G}(\mathbf{y})$, and therefore $\mathbf{F}(\mathbf{x})$, is a vector, of length p, say. Then (1.7) becomes, in terms of vector components,

   $$G_i(\mathbf{y}) = F_i(\mathbf{x}), \text{ for } i = 1, \ldots, p. \tag{1.8}$$

   If $\mathbf{G}$ and $\mathbf{F}$ are both linear functions, then (1.7) may be written using matrix notation, as $G\mathbf{y} = F\mathbf{x}$, where now G and F are $p \times m$ and $p \times n$ matrices, respectively. The component form of this equation is obvious. Putting $p = m$ and $G = I_m$, the identity matrix in m dimensions, this is clearly of the same form as equation (1.6) in case III of §1.2, which is a matrix-type inverse problem, ($\mathbf{y} = F\mathbf{x}$).

   (b) $\mathbf{G}(\mathbf{y})$ is, for any given $\mathbf{y}$, a function of a real variable, t, which may vary over some range, $a \leq t \leq b$, say. The realization of (1.7) in this case is then

   $$G(\mathbf{y}; t) = F(\mathbf{x}; t), \text{ for } a \leq t \leq b. \tag{1.9}$$

This time, assuming linearity gives $\mathbf{g}^T(t)\mathbf{y} = \mathbf{f}^T(t)\mathbf{x}$, where, for any t, $\mathbf{g}(t)$ and $\mathbf{f}(t)$ are vectors of length m and n, and ^{T} denotes 'transpose', so that $\mathbf{g}^T(t)\mathbf{y}$ is the usual scalar product of $\mathbf{g}$ and $\mathbf{y}$.

2. Now, if we suppose that $\mathbf{y}$ is as before, but $\mathbf{x}$ is now a function, $x(s)$, of a real variable s, with $u \leq s \leq v$, and for simplicity only consider the case where $\mathbf{F}$ is a linear function of $\mathbf{x}$, then

   (a) with $\mathbf{G}(\mathbf{y})$ a p-vector, (1.7) becomes, using component notation for $\mathbf{G}$,

   $$G_i(\mathbf{y}) = \int_u^v f_i(s)x(s)ds, \text{ for } i = 1, \ldots, p. \tag{1.10}$$

   This has exactly the same form as (1.4) in case II of §1.2. Note, too, that, if $u = 0$, $v = R_\odot$, $p = m$ and $\mathbf{G}$ is the identity function (so $\mathbf{G}(\mathbf{y}) = \mathbf{y}$), (1.10) is exactly the generic form of the rotation and (linearized) structure problems in helioseismology, equation (1.32).

   (b) if $\mathbf{G}(\mathbf{y})$ is a function of t (again a real variable), with $a \leq t \leq b$, then (1.7) is

   $$G(\mathbf{y}; t) = \int_u^v f(t; s)x(s)ds, \text{ for } a \leq t \leq b. \tag{1.11}$$

   This is the general form of a type of linear integral equation known as a *Fredholm equation* (see Craig and Brown 1986, §2.1), and has the same form as (1.2) in case I of §1.2.

From these four cases it is easy to see the form (1.7) will take in other cases, for example when both $\mathbf{G}(\mathbf{y})$ and $\mathbf{F}(\mathbf{x})$ are functions of a real variable. A simple guide to the interpretation of (1.7) is that when $\mathbf{x}$ represents the values of a function of a continuous variable, s, $\mathbf{F}$ involves an integral over s (and an equivalent statement can be made for $\mathbf{y}$ and $\mathbf{G}$).

Having exemplified the mathematical content of the symbolic equation (1.7) it is now possible to get to the essential point of all these shenanigans. Whether the problem described by (1.7) (namely to find $\mathbf{x}$ given the values of the observables $\mathbf{y}$ and the relationship (1.7)) is an inverse problem is determined entirely by the form of the function $\mathbf{F}$: solving (1.7) is an inverse problem if, and only if, $\mathbf{F}$ is a non-trivial function of the non-observable $\mathbf{x}$ (by non-trivial it is meant that the system of equations is coupled, in the sense described in the definition on page 13). The function $\mathbf{G}$ is unimportant in this context. However, $\mathbf{G}$ is important for maintaining the symmetry and generality of (1.7). After all, the fact that the division of the world into observables and unobservables is not intrinsic would make it very surprising if it turned out that they were always related by a formula like $\mathbf{y} = \mathbf{F}(\mathbf{x})$. There are many real problems where the theoretical relationship between the observables, $\mathbf{y}$, and the unknowns, $\mathbf{x}$, has the (linear) form $\mathbf{x} = G\mathbf{y}$. For example, the expansion of the observed velocity field on the surface of the sun in terms of spherical harmonics is achieved essentially by 'multiplying' the observed velocity image by the spherical harmonic transform matrix: the unobservable spherical harmonic coefficients are then given by a formula like $\mathbf{x} = G\mathbf{y}$.

Given, then, that the form of the function $\mathbf{G}$ is largely irrelevant for the classification of a problem as an inverse problem, we might as well ignore it. Better still, we can pretend that the data for the problem is $\mathbf{G}(\mathbf{y})$, which is as good as using $\mathbf{y}$ since it is easily calculable from measurements of the observables. The data $\mathbf{G}(\mathbf{y})$ can now be represented, for simplicity, by the symbol $\mathbf{g}$, and so the problems we are interested in take the form

$$\mathbf{g} = \mathbf{F}(\mathbf{x}). \tag{1.12}$$

It is now in order to make further simplifying assumptions about the form and meaning of equation (1.12). Although many real problems (for example the structure problem in helioseismology) are actually non-linear inverse problems (i.e. the function $\mathbf{F}$, which relates the unknowns to the data, is a non-linear function of those unknowns), many formalisms for solving such problems resort to linearizing $\mathbf{F}$ about some known solution (Backus and Gilbert 1967), solving a linear inverse problem for the difference between this solution and the true one, and then linearizing the inverse problem again about the new solution, repeating this iterative procedure to convergence. It is, therefore, vital to have reliable techniques for solving *linear* inverse problems before any attempt can be made to tackle non-linear problems. Since this thesis will deal exclusively with methods for solving linear inverse problems, it is reasonable at this point to specialize to this case, and assume, in (1.12), that $\mathbf{F}$ is a linear function of the unknowns, and, furthermore, that it is homogeneous, so that there is no constant term (any such term could be absorbed by a redefinition of $\mathbf{g}$). The result of this will be that $\mathbf{F}$ becomes a linear operator on the space containing the unknowns. The various different manifestations of the relationship between the data and the unknowns that are contained within the general expression (1.12), are then:

- When the data are a function of a continuous variable, s, so $\mathbf{g} = \mathbf{g}(s)$, (1.12) becomes a Fredholm equation, (1.11), if the unknowns are also the values of a function of a continuous variable, whereas if $\mathbf{x}$ represents only a finite number of degrees of freedom (it is a vector, in other words), then (1.12) essentially corresponds to the expansion of the data function in terms of some set of functions (the x_j being the coefficients in the expansion). A good example of this is the interpolation of functions by polynomials. Finding the coefficients of the interpolating polynomial (the x_j) is a problem known to have all the instabilities characteristic of inverse problems (see §3.5 and §2.8 of Press *et al.* 1992, *Numerical Recipes*, which will be referred to hereafter just as *Numerical Recipes*). In real experimental or observational situations an infinite amount of information is never available, and in helioseismology in particular, we always have only a finite number of measured data values (mode frequencies or frequency splittings). The situation where $\mathbf{g}$ is a function will, therefore, be considered no further here. (Although see case I of §1.2. The solution of such problems proceeds by reduction to problems with a finite number of degrees of freedom by *discretization* – see §1.9 and §2.2.3.)

- The case of greatest importance and interest in most astronomical inverse problems occurs when some number (m, say) of data values have been measured, so $\mathbf{g}$ represents an m-vector. Then, if there is a finite number, n say, of unknowns, (1.12) becomes a matrix equation (non-square, $n \neq m$, in general). If the unknowns are objects with continuous degrees of freedom, (1.12) has the form (1.10), which is the generic form of the inverse problems of helioseismology.

It has been amply demonstrated in this section how the apparently innocuous equation (1.7) can be used to represent problems in experimental or observational science, and how the inverse problems common in astronomy correspond exactly to cases where the function $\mathbf{F}$ (of the unknowns) is non-trivial. It was then shown how, after assuming (largely for pragmatic reasons) that $\mathbf{F}$ is a linear function, all the important types of linear inverse problem exemplified in §1.2 can be derived from (1.7). This presentation highlights the importance of distinguishing between observable and non-observable quantities, and of the theoretical relationship between them.

1.4 Ill-posed Problems

In presenting the examples of inverse problems in cases I to III of §1.2 the concepts of non-uniqueness and instability to data perturbations in the solution of such problems were encountered. Any problem possessing either or both of these characteristics is referred to as an *ill-posed* problem. Such features are anathema to good 'traditional' scientific problems, in which a solution is expected to exist, and is also expected to be affected only slightly by data errors (for otherwise, surely, the solution obtained from such erroneous data must be completely inaccurate). Indeed, it is tempting to think that such problems just cannot be solved, and that any attempt to do so must prove futile. The fact that this is not true can be gleaned from the wealth of literature devoted to this very topic (Craig and Brown 1986, and references therein). The trick is to recognize that the non-uniqueness and instability usually involve 'small-scale' features in the solution (for example, in the image processing problems described in §1.2 the lack of uniqueness manifests itself as an inability to recover features in the image on very small scales – a lack of resolution). The goal then becomes not the recovery of 'the solution', but the extraction of as much of the information in the data about larger scale variations in the solution as possible. Implicit in this is the idea of smoothing or averaging, and methods of solution using this idea will be reviewed in §1.8 and §1.9. In this section the aim is merely to discuss the mathematical reasons for, and ramifications of, this ill-posedness.

The work in this thesis concentrates on linear inverse problems, in which changes in the solution are related linearly to changes in the data. For the preceding statement to have any meaning both the solution and the data must lie in *linear* (i.e. vector) spaces. The effects of ill-posedness to be described here apply equally to non-linear problems, where the data and solution may lie in more complicated spaces, but it is more difficult to grasp the meaning of ill-posedness. For convenience, call the vector space containing the solution $\mathcal{S}$, and that containing the data $\mathcal{D}$. Imagine that norms have been defined on both $\mathcal{S}$ and $\mathcal{D}$, so that we have some measure of the 'size' of any vector in these two spaces (see Craig and Brown 1986, §5.3.2, or Parlett 1980, §1.6, for a description of norms on vector spaces). Naturally, the norms on $\mathcal{S}$ and $\mathcal{D}$ will be denoted by $||.||_{\mathcal{S}}$ and $||.||_{\mathcal{D}}$, respectively. The most important property of norms as far as we are concerned here is that they give a definition of 'size' for the vectors in a vector space, so that concepts such as error magnification can be considered. It will be useful in what follows, though, to state the defining properties of norms on vector spaces:

> A norm $||.||$ on a vector space $\mathcal{V}$ is any function (functional, if you like) from $\mathcal{V}$ to $\mathbb{R}$ (the space of real numbers) satisfying the following three properties
>
> 1. $||\mathbf{x}|| \geq 0$, and $||\mathbf{x}|| = 0$ if and only if $\mathbf{x} \equiv \mathbf{0}$
> 2. $||\alpha\mathbf{x}|| = |\alpha| \, ||\mathbf{x}||$ for any real number α
> 3. $||\mathbf{x}+\mathbf{y}|| \leq ||\mathbf{x}|| + ||\mathbf{y}||$
>
> for any vectors $\mathbf{x}$ and $\mathbf{y}$ in the space $\mathcal{V}$.

Now, imagine any non-zero vector $\mathbf{x}$ in $\mathcal{S}$ (so that $||\mathbf{x}||_{\mathcal{S}} > 0$ by property 1 above). The forward problem (1.12) maps $\mathbf{x}$ to a vector $\mathbf{g}(\mathbf{x}) \equiv F(\mathbf{x})$ in $\mathcal{D}$. It is the relationship between $||\mathbf{g}(\mathbf{x})||_{\mathcal{D}}$ and $||\mathbf{x}||_{\mathcal{S}}$ that is crucial in understanding the significance of ill-posedness. First, consider the situation where there exist non-zero vectors $\mathbf{x}$ in $\mathcal{S}$ for which $||\mathbf{g}(\mathbf{x})||_{\mathcal{D}} = 0$ (i.e. $\mathbf{g}(\mathbf{x}) \equiv \mathbf{0}$). Assume that we have obtained by experiment or observation some set of data $\mathbf{g}_{\mathrm{obs}}$, and have found a solution to (1.12) for this data, i.e. we have found some $\mathbf{x}_0$ satisfying

$$\mathbf{g}_{\mathrm{obs}} = F(\mathbf{x}_0).$$

Now consider the vector $\mathbf{x}_0 + \mathbf{x}$ in the solution space $\mathcal{S}$, where $\mathbf{x}$ is a *non-zero* vector in $\mathcal{S}$ such that $||\mathbf{g}(\mathbf{x})||_{\mathcal{D}} = 0$. Then the linearity of the problem tells us that

$$F(\mathbf{x}_0 + \mathbf{x}) = F(\mathbf{x}_0) + F(\mathbf{x}) = \mathbf{g}_{\mathrm{obs}} + \mathbf{0} = \mathbf{g}_{\mathrm{obs}},$$

so that $\mathbf{x}_0 + \mathbf{x}$ is *also* a solution of (1.12), and $\mathbf{x}_0 + \mathbf{x} \neq \mathbf{x}_0$: the solution is *non-unique.* In linear problems this non-uniqueness occurs precisely when the mapping F from $\mathcal{S}$ to $\mathcal{D}$ is such that some non-zero vectors in $\mathcal{S}$ are mapped to the zero vector in $\mathcal{D}$.

Non-uniqueness is, in a sense, the limiting case of instability to perturbations in the data. With non-uniqueness, *no* change in the data can permit large changes in the solution, whereas instability occurs when very small changes in the data give rise to large changes in the solution. It may seem from this discussion that instability occurs when $||\mathbf{g}||_{\mathcal{D}}$ is very small ($||\mathbf{g}||_{\mathcal{D}} \ll ||\mathbf{x}||_{\mathcal{S}}$), but this is not strictly the case. For example, when the data space is the same as the solution space, and the mapping F is given by

$$F(\mathbf{x}) = \varepsilon\mathbf{x}, \quad \text{for all } \mathbf{x} \text{ in } \mathcal{S}, \tag{1.13}$$

where ε is some very small number, there is no instability to data errors. To see this consider a data vector $\mathbf{g}_{\text{obs}}$, and any perturbation $\delta\mathbf{g}$ such that

$$\frac{||\delta\mathbf{g}||_{\mathcal{D}}}{||\mathbf{g}_{\text{obs}}||_{\mathcal{D}}} = \delta \ll 1. \tag{1.14}$$

If $\mathbf{x}_0$ is again the 'correct solution', and $\delta\mathbf{x}$ is the error in the solution satisfying the perturbed data, so that

$$\mathbf{g}_{\text{obs}} + \delta\mathbf{g} = F(\mathbf{x}_0 + \delta\mathbf{x}) = F(\mathbf{x}_0) + F(\delta\mathbf{x}),$$

then it is easy to see that $\delta\mathbf{g} = F(\delta\mathbf{x}) = \varepsilon\delta\mathbf{x}$, (using (1.13)), or

$$\delta\mathbf{x} = \frac{\delta\mathbf{g}}{\varepsilon},$$

which is, apparently, a massive error amplification (ε is very small). However, the *relative* error is given by

$$\frac{||\delta\mathbf{x}||_{\mathcal{S}}}{||\mathbf{x}_0||_{\mathcal{S}}} = \frac{||\delta\mathbf{g}||_{\mathcal{D}}}{|\varepsilon|} \times \frac{|\varepsilon|}{||\mathbf{g}_{\text{obs}}||_{\mathcal{D}}} = \frac{||\delta\mathbf{g}||_{\mathcal{D}}}{||\mathbf{g}_{\text{obs}}||_{\mathcal{D}}} = \delta \ll 1$$

from (1.14) and using property 2 of norms. So the relative errors are no bigger for the solution than for the data. There is effectively no error magnification.

It turns out, in fact, that what gives rise to instability is the occurrence of differential magnification by F of vectors in the solution space. Consider a general mapping F, and define two important quantities (both positive, by property 1 of norms)

$$\lambda_{\min} = \min_{\mathbf{x}\neq\mathbf{0}} \left\{ \frac{||\mathbf{g}(\mathbf{x})||_{\mathcal{D}}}{||\mathbf{x}||_{\mathcal{S}}} \right\} \tag{1.15}$$

$$\lambda_{\max} = \max_{\mathbf{x}\neq\mathbf{0}} \left\{ \frac{||\mathbf{g}(\mathbf{x})||_{\mathcal{D}}}{||\mathbf{x}||_{\mathcal{S}}} \right\} \tag{1.16}$$

and, similarly, let the *unit* vectors in the solution space at which these extrema are obtained be denoted by $\mathbf{x}_{\min}$ and $\mathbf{x}_{\max}$ (the linearity of the problem and property 2 of norms ensure that it is always adequate to consider unit vectors in the solution space). Most mappings, F, used in physical situations are *bounded*, in the sense that $\lambda_{\max} < \infty$. Here we will also assume, since we are considering instability, that $\lambda_{\min} > 0$, so that non-uniqueness does not occur. Consider any (non-zero) solution vector $\mathbf{x}$, with corresponding data $\mathbf{g}$, and assume that the data errors, $\delta\mathbf{g}$, are again small, satisfying (1.14). Then (1.15) and (1.16) tell us that

$$\frac{||\mathbf{g}||_{\mathcal{D}}}{||\mathbf{x}||_{\mathcal{S}}} \leq \lambda_{\max}$$

$$\frac{||\delta\mathbf{g}||_{\mathcal{D}}}{||\delta\mathbf{x}||_{\mathcal{S}}} \geq \lambda_{\min}$$

From these two equations it is easy to derive the following limit on the relative error in the solution resulting from the (generally unknown) data errors

$$\frac{||\delta \mathbf{x}||_{\mathcal{S}}}{||\mathbf{x}||_{\mathcal{S}}} \leq \frac{\lambda_{\max}}{\lambda_{\min}} \frac{||\delta \mathbf{g}||_{\mathcal{D}}}{||\mathbf{g}||_{\mathcal{D}}} = \frac{\lambda_{\max}}{\lambda_{\min}} \delta. \tag{1.17}$$

For $\lambda_{\min} \sim \delta$ and $\lambda_{\max} \gtrsim 1$ this gives

$$\frac{||\delta \mathbf{x}||_{\mathcal{S}}}{||\mathbf{x}||_{\mathcal{S}}} \gtrsim 1,$$

in other words, a relative error of over 100%, when δ could have been 0.01, say, corresponding to data errors of only 1%: a huge error magnification. And don't be fooled by the fact that the error is only bounded above in (1.17) into thinking that these bounds would never be attained. If by chance the 'true' solution was $\mathbf{x}_{\max}$ and the data errors happened to correspond to a vector in the solution space parallel to $\mathbf{x}_{\min}$, then the relative error in the solution would be exactly equal to the upper bound given in (1.17).

It is clear from the above discussion that the inverse problem (1.12) will be unstable to errors in the data whenever the quantity

$$C_F \overset{\text{def}}{\equiv} \frac{\lambda_{\max}}{\lambda_{\min}} \tag{1.18}$$

is large ($\gtrsim 10$, say). C_F is called the *condition number* of the operator F (another way of saying that the solution of (1.12) is unstable to perturbations in the data is to say that F is *ill-conditioned*), and is a useful measure of the instability of the problem. Note that if the problem does not have a unique solution, then $\lambda_{\min} = 0$, and C_F is infinite – very ill-conditioned indeed.

1.5 The Inverse Problems of Helioseismology

This thesis deals with methods for optimizing the solution of inverse problems in general, but with particular emphasis on those that appear in helioseismology. In this section, the physics of the solar oscillations is briefly recapitulated, and the way in which the oscillation frequencies (which constitute the data for the helioseismological inverse problems) are affected by the internal structure of the sun (this is the *forward problem*) is explained. Solving an inverse problem amounts to extracting the information about the unknown quantities from the data. A simple example showing how it is possible to recover this information (without recourse to detailed numerical techniques) is given. This should motivate the presentation of the numerical methods of solution in sections 1.8 and 1.9.

1.5.1 The Solar Oscillations

The observed solar oscillations, whose restoring force is predominantly due to pressure gradients, hence the designation p modes, are trapped in a resonant cavity or 'trapping' region, whose upper boundary is near the top of the convection zone (just below the surface of the sun). The position of the lower boundary of this trapping zone depends on the mode under consideration, but, for most of the modes observed is somewhere within the solar convection zone. This means that most acoustic waves propagate within the convection zone, being 'reflected' at the boundaries of the trapping region. The concept of a trapping zone comes from asymptotic analysis (see Unno *et al.* 1979 §15, Gough 1986, Gough 1984), where oscillations with wavelengths much shorter than the characteristic length-scale of variation in the solar structure are treated locally as plane parallel sound waves propagating in a homogeneous medium. As the wave moves between regions with different fluid properties (density, pressure etc.) its wavelength and amplitude change to conserve energy flux and so on. (This is the WKB approximation. It is quantitatively valid only for eigenmodes with large spherical harmonic degree, l, or large radial order, n, i.e. short horizontal or vertical wavelength, respectively.) The waves are then described by their (curved) ray-paths through the star, which are determined by changes in sound speed, just as light is refracted by changes in the refractive index of the medium through which it is passing. Figure 1.3 shows just such ray-paths for two different modes. A ray travelling down through the convection zone at some angle to the vertical (so that it is not a radial mode) will be refracted further away from the vertical by the increase in sound speed with depth until it is travelling horizontally. It will then be refracted upwards until it reaches the surface where it is *reflected* by the sharp change in density near the top of the convection zone. The point of reflection marks the top of the resonant cavity, and the radius at which the ray becomes horizontal defines its lower boundary. The rays can never escape from this region: they are trapped. Although this analysis is only approximate, it is very useful for describing and classifying the solar eigenmodes, because the inaccuracies involved are really quite small for the vast majority of modes. Nevertheless, it is important to realize that the use of terms such as 'trapping region', 'reflection', 'refraction', etc. is somewhat figurative. The full solutions of the oscillation equations have properties that closely reflect aspects of the ray-path analysis, but the intrinsically global nature of the full problem renders the two approaches quite distinct. In what follows the global treatment

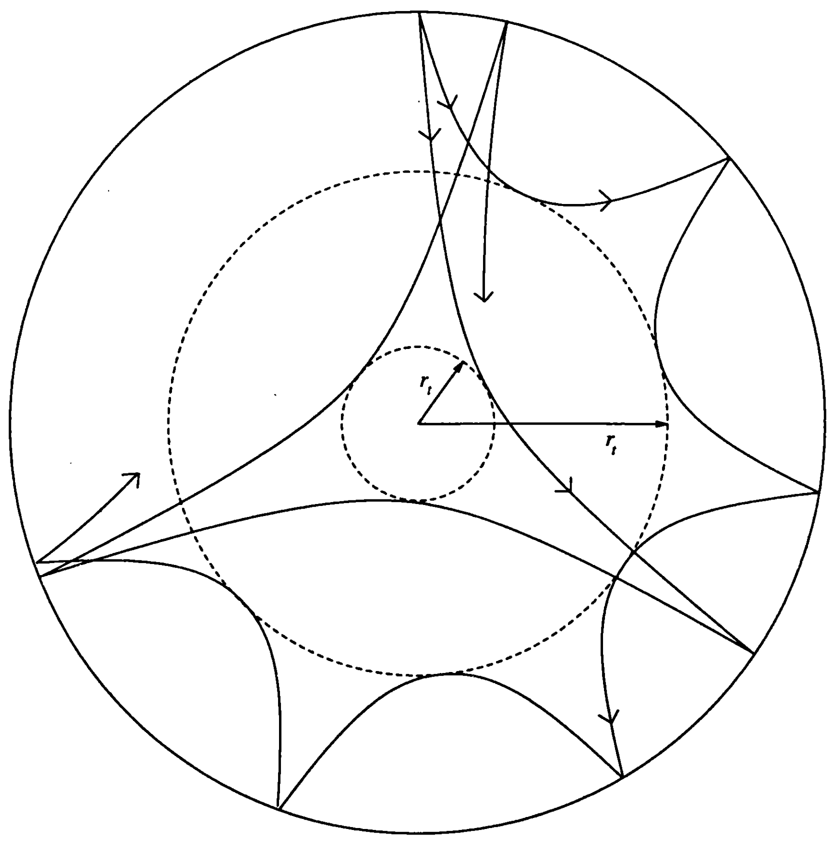


Figure 1.3: A schematic diagram of the ray-path of two different sound waves travelling through the sun. The outer circle represents the surface of the sun, and the view is a cross-section. The dotted circles marked by r_t indicate the lower turning points of the modes, which are reflected down from the surface and then refracted back up.

will be employed as far as possible, but the local description will be used frequently to clarify ideas.

The trapping zone for a given eigenmode of oscillation is the region of the solar interior where it is propagating (that is, where its wave-function oscillates in radius, so that upward and downward propagating waves interfere to produce a standing wave-pattern). Outside this region the waves no longer propagate, and the wave-function decays exponentially and monotonically with distance from the cavity boundary (the mode is said to be evanescent). This means that the amplitude of the oscillation will be largest in the trapping region, falling away rapidly in the evanescent zone. It seems quite reasonable to assume that the properties of any eigenmode (in particular, its frequency) are most sensitive to the solar structure in regions where its amplitude is large, and this is indeed the case. To see this more clearly, consider the global description of an eigenmode, which is summarized briefly here, and is described in detail in Unno *et al.* (1979). The solar oscillations are nothing more than small deviations in the hydrodynamical and thermodynamical properties of the solar material away from some stable equilibrium value (given by the equilibrium solar model). A fundamental (and perfectly reasonable) assumption made in helioseismology is that, to an adequate level of accuracy, the properties and dynamical behaviour of the solar material are well described by the equations of fluid dynamics, supplemented by equations describing the flow of radiation, the gravitational field and so on. This means that both the equilibrium state of the sun and any deviations from this state are determined by solving these equations. Solving the equations pertaining to the solar material in their most general form is outstandingly difficult, but in helioseismology there are obvious approximations that can be made which greatly simplify things. For example the solar equilibrium state is very nearly spherically symmetric and time-independent (although see Unno *et al.* 1979, §12, for a discussion of the treatment of convection), so the fluid equations to describe this are much simplified. Similarly, the fact that the observed solar oscillations have a very small amplitude allows all terms in the fluid equations that are not linear in the oscillating quantities to be dropped, leaving a set of linear equations, which are much easier to handle than the general non-linear case. The oscillation equations obey a superposition principle, i.e. they have the property that any solution (any solar oscillation) may be expanded as a linear combination of simple solutions (the eigenfunctions of the oscillation equations – the solar eigenmodes) whose development in time is harmonic (i.e. sinusoidal) with a specific frequency for each eigenmode (which is its *eigenfrequency* – this is the frequency that is measured in helioseismic observations), and, furthermore, each eigenmode evolves independently of the others (thanks to the linearity of the problem) and so it is possible to study the properties and behaviour of the eigenmodes separately. The eigenmodes and their corresponding eigenfrequencies contain all the information necessary to describe any solution of the oscillation equations.

It may be helpful to consider some of the basic properties of the eigensolutions of the oscillation equations. Any eigenmode of the sun involves small periodic perturbations in the fluid properties throughout the sun. For example, perturbations in the density are $\rho'(r,\theta,\phi,t) \equiv \rho(r,\theta,\phi,t) - \rho_0(r)$ ($\rho_0(r)$ is the density of the spherically symmetric, time-independent, unperturbed equilibrium state at radius r), whereas the (vector) change in the position of a fluid element (relative to its position in the equilibrium state) could be written $\boldsymbol{\xi}(r,\theta,\phi,t) = \xi_r(r,\theta,\phi,t)\hat{\mathbf{r}} + \xi_\theta(r,\theta,\phi,t)\hat{\boldsymbol{\theta}} + \xi_\phi(r,\theta,\phi,t)\hat{\boldsymbol{\phi}}$. Obviously, the fluid equations relate these quantities, and others such as the pressure and gravitational perturbations. The eigenfunctions are very special and simple oscillations because their dependence on time and on the spherical co-ordinates θ and ϕ is simple, and the perturbed quantities are related by *ordinary* differential equations, by virtue of the fact that the linearized oscillations are *separable.* Just as the time independence of the equilibrium model allows eigenfunctions with a harmonic ($e^{-i\omega_{nlm}t}$, or $\sin\omega_{nlm}t$, where ω_{nlm} is the eigenfrequency of the (n,l,m) mode) time dependence to be chosen, the spherical symmetry of the equilibrium state permits eigenfunctions to be chosen whose dependence on the 'horizontal' co-ordinates, θ and ϕ is a *spherical harmonic*, $Y_{lm}(\theta,\phi) \propto P_{lm}(\theta)e^{im\phi}$, ($P_{lm}(\theta)$ is an associated legendre function – again see Unno *et al.* 1979). The result is that the eigenfunctions for the perturbations in a scalar quantity, say density, have the form

$$\rho'_{nlm}(r,\theta,\phi,t) = \rho'_{nlm}(r)Y_{lm}(\theta,\phi)e^{-i\omega_{nlm}t} \tag{1.19}$$

(where, with an obvious abuse of notation, the same symbol is used for the complete perturbation and for the radial dependence). The eigenfunction for a vector quantity (the displacement of fluid elements, for example) is given in (1.20), and relies on the use of *vector* spherical harmonics to describe the horizontal variation, but this is not important here. Clearly, therefore, choosing the horizontal dependence of the modes (i.e. choosing l and m) leaves the radial dependence and the eigenfrequency, ω_{nlm}, to be calculated. The equations that do this form, along with the appropriate boundary conditions, a *boundary value problem* (Unno *et al.* 1979). It is characteristic of such problems that they contain a free parameter (in this case, the oscillation frequency) such that a solution of the equations can only be found for certain values of this parameter. These values are the eigenvalues (eigenfrequencies), and, typically, there is an infinite *spectrum* of them, labelled by integers, n. So, solar eigenmodes are found by choosing the integers l and m (determining the horizontal variation) and using the system of equations that determine the radial behaviour of the eigenmodes with these values of and l and m to find eigenfrequencies and their corresponding eigenfunctions (labelling each with a different value of n).

It is clear from this that the horizontal variation of the eigenmodes, and the form of their time dependence contain no information about the solar equilibrium state (except that it is spherically symmetric and time independent). The radial variation of the solar internal structure only enters the boundary value problem to determine eigenfrequencies and the radial dependence of the eigenfunctions: this is the interesting part of the problem. Changing the solar structure changes the coefficients in the differential equations that determine the radial eigenfunctions and therefore changes the eigenfunctions *and* eigenvalues (oscillation frequencies). This is how, from a mathematical point of view, the dependence of the solar oscillation frequencies on the equilibrium solar model arises. Unno *et al.* (1979) present a derivation of the forward problems of helioseismology, for the solar structure, magnetic field and rotation problems, (see particularly §18 of Unno *et al.* 1979, and also Ritzwoller and Lavely 1991). This provides, for a given solar equilibrium model, a mathematical representation of the (quantitative) dependence of the oscillation frequencies on the stratification, angular velocity of rotation and magnetic field strength, respectively, throughout the solar interior. In other words, given a particular rotation profile, for example, the forward problem gives a simple procedure for calculating the change in the oscillation frequency of any solar eigenmode in response to this profile. (The inverse problem, of course, consists of attempting to find the rotation profile given the observed mode frequencies.)

To recap, linearization of the equations of fluid dynamics about a spherically symmetric and time independent equilibrium state (i.e. the solar model under consideration, which is assumed to be in hydrostatic equilibrium) results in a set of (obviously linear) homogeneous differential equations for the quantities that characterize the small disturbance from static equilibrium in which the coefficients depend only on radius, r. This means that the time and horizontal (θ and ϕ) dependence of the solution can be separated out: solutions of the equations which have the form $Q(r,\theta,\phi,t) = Q(r)H(\theta,\phi)T(t)$ can be found. The linearity of the problem then ensures that a general linear combination of such disturbances will also be a solution of the equations, so that each component will evolve independently of the others, and it is only necessary to consider the components separately. (Mathematically, the equation is said to be separable, and the collection of all such separable solutions forms a *complete set* so that any solution may be expanded in terms of them.)

1.5.2 Inverting Helioseismic Data by Eye

The trapping zone for a given eigenmode is the region where its energy density is largest, and is, therefore, the region of the sun to which the frequency of the mode is most sensitive. To be more specific, when solving the equations for the linear, nonradial, adiabatic eigenmodes of oscillation of some given (spherically symmetric) solar model (see Unno *et al.* 1979, chapter III) the displacement eigenfunctions $\xi_r(r)$ and $\xi_h(r)$ appear (these quantities are proportional to the radial and horizontal distances, respectively, between the position of the fluid element at radius r in the original, unperturbed state and its position in the oscillatory flow). The actual displacement of the fluid element at (r, θ, ϕ) at time t is given in terms of $\xi_r(r)$ and $\xi_h(r)$ by

$$\boldsymbol{\xi}_{nlm}(r,\theta,\phi,t) = \left(\xi_{rnl}(r),\, \xi_{hnl}(r)\frac{\partial}{\partial\theta},\, \frac{\xi_{hnl}(r)}{\sin\theta}\frac{\partial}{\partial\phi}\right) Y_{lm}(\theta,\phi)\, e^{-i\sigma_{nl}t} \tag{1.20}$$

where l denotes the degree of the mode, m the azimuthal order, n the radial order, σ_{nl} the angular frequency of the mode and $Y_{lm}(\theta,\phi)$ is the spherical harmonic which describes the horizontal (i.e. nonradial) structure of the oscillation. Clearly the velocity of the oscillation is proportional to $\sigma_{nl}\boldsymbol{\xi}_{nl}$. Thus the kinetic energy density of the oscillation is large where

$$K_{nl}(r) = \frac{1}{2}\rho_0(r)\sigma_{nl}^2||\boldsymbol{\xi}_{nl}||^2 \tag{1.21}$$

is large ($\rho_0(r)$ is the density of the solar model at radius r).

It is known from the local treatment of the oscillation equations (using the WKB approximation, see Unno *et al.* 1979, pp.89 – this approximation is quantitatively valid only for modes with large n or l, but, qualitatively, its conclusions apply generally) that any eigenmode has a trapping region, where its amplitude is oscillatory (as a function of r) and is, on average, much larger than elsewhere. Outside the trapping region are evanescent zones where the amplitude decays exponentially and monotonically. Thus any mode has large energy density at points in the trapping zone where the amplitude is large (antinodes) and small energy density in evanescent regions and near nodes in the trapping zone. The structure of the solar model, and, in particular, for p modes, the sound speed profile, determines the trapping region, and, consequently, the eigenfrequency of the oscillation mode, and the position of the nodes within the trapping region. It turns out, as might be expected, that the sensitivity of the eigenfrequency σ_{nl} of any mode to (small) perturbations in the solar structure is roughly proportional to the energy density. That is, at a point where the oscillation is 'stronger', a small, very localized perturbation in the solar structure will give a larger perturbation in the eigenfrequency than the same perturbation at a point where the oscillation is 'weaker' (evanescent zone or near a node). As the position of the lower edge of the trapping zone and the positions of the nodes vary from mode to mode (and vary in quite a regular way), the measurement of many oscillation frequencies and their deviation from the theoretically calculated eigenfrequencies of some known (and, hopefully, reasonably accurate) solar model would permit aspects of the solar structure as a function of depth to be inferred. To make this clearer consider a perturbation of the solar structure in a very narrow region (that is, a perturbation in the solar structure that is essentially localized in radius) compared to the typical wavelengths of eigenmodes under consideration, so we can express it as a delta function

$$Q(r) = Q_0(r) + Q'_{r_0}(r) \equiv Q_0(r) + q(r_0)\,\delta(r - r_0) \tag{1.22}$$

where Q is the perturbed quantity, Q_0 is its original value, and $Q'_{r_0}(r) = q(r_0)\delta(r-r_0)$ is the perturbation, which is situated at r_0 and has 'size' $q(r_0)$. Assume r_0 is somewhere in the convection zone. If the response of the eigenfrequency of the mode (n, l, m) is proportional to $K_{nlm}(r) \equiv$ the kinetic energy density, then the eigenfrequency perturbation, $\delta\sigma_{nl}$, will be proportional to

$$q(r_0)\,K_{nl}(r_0) \sim \rho_0(r_0)\,||\xi_{nl}(r_0)||^2 \tag{1.23}$$

Now, arrange the modes in order so that going from one mode to the next the position of the lower boundary of the trapping zone moves further out, and label them by i, with $i = 1, 2, \ldots$ (so that as i increases, so does $r_t(i)$ the radius of the lower boundary of the trapping zone – *cf.* figure 1.3). Analysing the observed frequency perturbations $\delta\sigma(i)$ it will be found that as i increases from 1, $\delta\sigma(i)$ oscillates, because for these modes $r_t(i)$ is small ($r_t(i) < r_0$, since we assumed r_0 is somewhere in the convection zone and we have modes whose trapping zone includes the whole convection zone: remember, the upper boundary of the trapping zone is the top of the convection zone for all modes) and so the displacement eigenfunction is oscillatory in the region of r_0 for these modes. Thus for some modes r_0 will be near the position of a node, and $\delta\sigma(i) \approx 0$, while for others the perturbation will be at an antinode and $\delta\sigma(i)$ will be large. For some i_0 we will have $r_t(i) \approx r_0$, and at this i_0 a change will occur in the variation of $\delta\sigma$ with i. It will no longer be oscillatory, but will decay monotonically with i. This is because as i increases above

i_0 the modes become evanescent at r_0, so the perturbation is small. Furthermore, as i increases $r_t(i)$ increases, so r_0 is further into the evanescent region, and so $K(i)$ and hence $\delta\sigma(i)$ get smaller. This means that we can, from the frequency perturbations $\delta\sigma$, infer the position of the perturbation Q'_{r_0}. The expression for the frequency perturbation in terms of $||\boldsymbol{\xi}(r_0)||^2$ for any modes allow $q(r_0)$ to be found, therefore determining the perturbation exactly.

The inverse problem we are considering is linear, so if we assume that the perturbation now consists of two delta functions at different positions the procedure just applied to locate the single delta-function would show the same kind of oscillatory behaviour, except that now there ought to be some kind of 'beating' phenomena, where the two perturbations sometimes add 'coherently' (for modes where both delta-functions nearly coincide with antinodes), while for others only one delta-function, or perhaps neither of them, will be near an antinode, and so the perturbation will not be so great. As the kernels are positive definite this beating phenomenon would be most easily visualized for two perturbations with opposite signs (so that a kind of constructive and destructive interference could occur). Nevertheless, arrangement of the kernels in order of increasing r_t ought to reveal a point (an $r_t(i)$) where the beating phenomenon ends (when the innermost perturbation occurs near $r_t(i)$) and the variation of the perturbation with i looks as it did in the single delta-function case. This indicates the location of the innermost perturbation. The i for which the other delta-function is near the lower turning point, $r_t(i)$, should be visible as before. So, it is, in principle, possible to locate two delta-function perturbations in the solar structure.

Analogously, we could, by virtue of the linearity of the problem, extend this argument to three, four or more delta-function perturbations. It is clear, though, that it would very soon become practically impossible to visually interpret the oscillation data in the manner described above to locate the positions of the perturbations. However, the effect of each delta-function on the eigenfrequencies would be the same, and it simply requires detailed numerical calculation to locate them. Since we could write an arbitrary perturbation as a 'sum' of δ-function perturbations

$$Q'(r) = \int_0^{R_\odot} q(r_0)\,\delta(r - r_0)\,dr \tag{1.24}$$

it is reasonably clear that accurate measurement of the frequency perturbations ought to permit the determination of the general perturbation $Q'(r)$ by some method which is, in principle at least, equivalent to that given above. A moment's thought shows that this is an example of an inverse problem, due to the highly non-trivial relationship between the observables and the unknowns.

In the following section solving the inverse problems of helioseismology will be discussed from a more realistic viewpoint.

1.5.3 The Inverse Problems of Helioseismology

By performing spherical harmonic and Fourier transforms on series of 'velocity images' of the sun, diagnostic diagrams (Unno *et al.* 1979) displaying oscillatory power as function of horizontal wavelength (or l, the degree of the spherical harmonic) and frequency can be plotted. These show that the power is concentrated along ridges each of which corresponds to modes with the same value of radial order, n. As the sophistication of observations has increased (*cf.* Libbrecht *et al.* 1990) the measurement of the frequencies of modes with many different values of n, l and m (m is the azimuthal order of the mode, in a truly spherically symmetric sun the frequencies of modes with different m, for the same (n,l) would be the same, i.e. the modes would be degenerate) with an accuracy sufficient to permit aspects of the solar structure to be inferred has become possible. This has been attempted for rotation (Duvall Jr. *et al.* 1984; Korzennik *et al.* 1988; Christensen-Dalsgaard and Schou 1988; Dziembowski *et al.* 1989; Brown *et al.* 1989; Thompson 1990; Goode *et al.* 1991; Schou *et al.* 1992), for the sound speed (Gough and Kosovichev 1988; Dziembowski *et al.* 1990; Vorontsov and Shibahashi 1991), and some attempts have been made to infer the depth of the solar convection zone (Christensen-Dalsgaard *et al.* 1991) and the helium fraction in the solar convection zone as part of the structure problem (Vorontsov *et al.* 1991; Kosovichev *et al.* 1992), as well as to use the recovered sound-speed profile to constrain solar neutrino flux predictions (Dziembowski *et al.* 1990).

All of these problems are specific examples of inverse problems. They are characterized by the fact that the quantity to be found (sound speed, rotation curve etc.) is related to the observable quantities in a non-trivial way, typically involving some kind of integral transform or functional. The solar structure problem is non-linear, in the sense that the oscillation frequencies are related to the sound speed by a non-linear functional:

$$\sigma_{nl}^2 = \int_0^{R_\odot} K_{nl}(r; c^2)\, dr, \text{ for } l = 0, 1, \ldots, \text{ and } n = 0, \pm 1, \ldots \tag{1.25}$$

where K_{nl} is a non-linear function of c^2 that depends on the mode (n,l) under consideration. The non-linearity here derives from the fact that the perturbation depends on the eigenfunction of the mode, and the eigenfunction depends on the structure, and so, to calculate the perturbation, $\delta\sigma_{nl}$, in the oscillation frequencies we must know the solar structure, which is the object of the exercise.

The solution of (1.25) is rather difficult and requires an iterative approach (Backus and Gilbert 1967, 1968, 1970). Make some (informed) guess to the real solar structure (calculate a solar model), evaluate the eigenfunctions and eigenfrequencies of the model (see Unno *et al.* 1979). These quantities can then be used in the variational principle which the eigenmodes of a spherical star are known to satisfy (Unno *et al.* 1979, §13.2; Gough 1985, §6; Dziembowski *et al.* 1990). This gives an expression for the difference in frequency between the model and the real sun in terms of the deviation of the sound speed (squared) between the model and the sun that is linear in the deviation, i.e. a linear functional for each mode. This procedure amounts to linearizing (1.25) to give

$$\delta\sigma_{nl}^2 = \int_0^{R_\odot} \frac{\partial K_{nl}}{\partial c^2}\,\delta c^2\,dr, \text{ for } l = 0,1,\ldots, \text{ and } n = 0,\pm1,\ldots \tag{1.26}$$

Make the definition

$$k_{nl}(r) = \frac{\partial K_{nl}}{\partial c^2}.$$

The $k_{nl}(r)$ are called the *kernels* of the linear integral equations (1.26). Solution of (1.26) (by one of the methods described in (1.8) or (1.9), below) gives a new solar model (new sound speed profile $c^2 + \delta c^2$) which can then be used to calculate new kernels $k_{nl}^{(2)}$ and new frequency differences $\delta\sigma_{nl}^{2(2)}$ and thus a new linear inverse problem. It is to be hoped that the initial guess to the solar structure will be sufficiently accurate to assure convergence to the correct solution. Thus, the solar structure problem is a non-linear problem which must be solved by some iterative procedure in which each step is a linear inverse problem (Backus and Gilbert 1967, 1968, 1970). In many ways this method is like Newton-Raphson iteration. There are many potential difficulties here, not least of which is the practical problem of calculating the composition of the new model (Gough 1985).

The other, very important inverse problem presented by helioseismology is the solar rotation problem. Inferring the solar internal angular velocity from frequency splitting data (the differences between the frequencies of modes with different m in the same (n,l) multiplet) is in many ways much simpler than the structure problem. This is basically because the effect of slow rotation (i.e. slow enough to be treated in perturbation theory – see Unno *et al.* 1979, §18 – as is the case for the sun) on the basic structure of the model (due to centrifugal force and the reduction in effective gravity, etc.) is of second order in the angular velocity, whereas the splittings of the frequencies within a multiplet depend linearly on the angular velocity. Thus the rotation curve can be found without needing to consider the perturbation of the original model (at least not at the present level of observational accuracy).

Ritzwoller and Lavely (1991) show that the full, two-dimensional solar rotation problem can, with judicious choices of basis functions in terms of which to expand the rotational velocity field and the variation of the eigenfrequencies of the modes in each multiplet (that is, with the same l and n), be reduced to an independent (and, in principle, infinite) sequence of 1-D inversions. Ritzwoller and Lavely expand the velocity field in terms of vector spherical harmonics as folows:

$$\mathbf{v}_{\mathrm{rot}}(r,\theta) = -\sum_{s=0}^{\infty} w^{(2s+1)}(r)\frac{\partial Y_{2s+1,0}}{\partial\theta}\hat{\phi}, \qquad (1.27)$$

where the $w^{(s)}$ (note that only $w^{(s)}$ with odd values of the index occur in the expansion (1.27)) are the functions to be determined in the inversion, Y_{s0} are *zonal* spherical harmonics ($m = 0$, and so they have no azimuthal dependence), and $\hat{\phi}$ is the unit vector in the azimuthal direction, as usual. The eigenfrequencies, ω_{nlm}, of the modes in each multiplet are expanded in terms of the *Clebsch-Gordon coefficients*, β^m_{jl} (see Ritzwoller and Lavely 1991 and references therein), to give

$$\omega_{nlm} = \omega_{nl0} + \sum_{j=1}^{n_{max}} b^j_{nl}\beta^m_{jl}, \qquad (1.28)$$

where the b^j_{nl} are the new *splitting coefficients*, and n_{max} is some upper limit to the number of coefficients to be used in the expansion for the frequencies which is determined by the quality of the data. For the data available presently, $n_{max} = 6$, but this will increase as more accurate data becomes available, from the GONG project (Harvey *et al.* 1993), for example. It is these coefficients, rather than the frequencies themselves, that are used as the data for the inversions to find the velocity field components $w^{(s)}$ ($s = 1, 3, 5, \ldots$). Using these expansions the 2-D rotation problem reduces to the sequence of 1-D inverse problems

$$q^{(s)}_i = \int_0^{R_\odot} k^{(s)}_i(r)\, w^{(s)}(r)\, dr, \text{ for } s = 1, 3, 5, \ldots, n_{max}, \text{ and } i = 1, \ldots, m, \qquad (1.29)$$

where, for convenience, the two integers n and l labelling the modes have been replaced by a single label, i. The data $q_i^{(s)}$ for these inverse problems are related to the splitting coefficients b_{nl}^j as in equation (56) of Ritzwoller and Lavely (1991). The $k_i^{(s)}$ in (1.29) are the rotational kernels, which depend on the displacement eigenfunctions ξ_{rnl} and ξ_{hnl} $(i \leftrightarrow nl)$ in the following way

$$k_i^{(s)}(r) = -\frac{r\rho_0(r)}{J}\left[\xi_{rnl}^2 + L^2\xi_{hnl}^2 - \left(2\xi_{rnl}\xi_{hnl} + \frac{1}{2}s(s+1)\xi_{hnl}^2\right)\right], \tag{1.30}$$

where the 'normalizing' constant J is

$$J = \int_0^{R_\odot} \left(\xi_{rnl}^2 + L^2\xi_{hnl}^2\right)\rho_0 r^2\,dr. \tag{1.31}$$

The expressions for the rotational kernels $k_i^{(s)}$ is also given in Ritzwoller and Lavely (1991).

Compare the equations in (1.29) with equation (1.32), below. It is clear, then, that to have any success at all in our attempts to infer aspects of the solar structure from oscillation data we must have an accurate, efficient and reliable method for solving linear integral equations of the form

$$g_i = \int_0^{R_\odot} k_i(r)\,f(r)\,dr \quad \text{for } i = 1,\ldots,m, \tag{1.32}$$

where m is the number of modes in the data set, $k_i(r)$ is the kernel corresponding to the ith mode and $f(r)$ is the quantity we wish to recover. It is interesting to note that while it is formally rather easy to solve (1.32) (approximately), it is in practice very difficult to solve accurately and efficiently. Clearly, there is no hope of obtaining an analytic solution to the set of m integral equations (1.32), so some numerical method must be found. This means that questions of truncation error, rounding error and stability of the numerical method must be addressed.

In practice the data $\{g_i; i = 1,\ldots,m\}$ is contaminated with a not inconsiderable level of noise (data errors). It is well known that any straightforward (straightbackward?) inversion of (1.32) is very unstable to small perturbations in the data. That is, a 'small' error in the data can, and usually does, result in a large error in the recovered solution (see §1.4 and Craig and Brown 1986, §1.3). This is intrinsic to the problem, and is not the result of any numerical inaccuracies. This has two implications:

- It is essential to introduce some method of stabilizing the inversion of (1.32). In all the commonly used techniques this is achieved by regularization (Craig and Brown 1986, §6.2). This is described in more detail below.

- Once the inversion has been stabilized the presence of a small amount of truncation error (or discretization error, as it will be referred to below) is acceptable, since its presence will only affect the solution by a similarly small amount. The presence of errors in the data means that the solution is already in error. Providing the discretization errors are much smaller than the data errors they are acceptable. Henceforth it will be assumed that any sensible approximator to (1.32) will be sufficiently accurate that discretization errors are unimportant.

The numerical methods for solving (1.32) that are commonly encountered (see Gough 1985) fall into two basic categories: *optimal averaging methods*, which are useful because they are designed to provide more information about the solution than just its values (information such as resolution and error magnification – see below), and *regularized least squares methods*, which are very simple to use, and very flexible (in the sense that it is possible to tailor them to the requirements of particular problems – see the discussion in §2.1), but lack some of the finesse (and complication) of the optimal averaging methods. These two methods will be described shortly, but first it is in order to consider the meaning of the term 'a solution' for problems of the form (1.32), like the solar rotation or the (linearized) structure problems. Once that is done, the requirements a successful inversion algorithm should satisfy will be discussed, before the various inversion procedures are reviewed.

1.6 What is a solution?

Although the terms 'solution' and 'true solution' are easily understood intuitively, it will be useful later to be a little pedantic here and to state explicitly their significance. This significance depends partly on the situation under consideration, so the meanings of 'solution' and 'true solution' are discussed from three different viewpoints.

1) Physically: The inverse problems of helioseismology attempt to use the observed properties (frequencies, splittings etc.) of solar oscillations to infer the value of quantities characterizing elements of the solar fluid (such as velocity, sound speed or magnetic field strength) throughout the volume of the sun. The 'true solution' would then be the actual value of the quantity under consideration, call it f, at each point within the sun: in spherical polar co-ordinates, the 'true solution' is a function assigning a value $f(r, \theta, \phi)$ to each point (r, θ, ϕ). A solution to the inverse problem for f should mimic the properties of the true solution, and so elucidation of the nature of the true solution in any inversion will motivate the choice of allowable solutions.

The inverse problems dealt with in this thesis are 1-D inversions in which the quantity of interest is dependent on only one independent variable, i.e. $f = f(r)$, whereas, in reality, the presence of a solar convection zone and latitudinal variations in the solar angular velocity mean that f depends on r, θ, ϕ and t. However, it is sufficient for the specification of the structure and rotation to consider averages of f over time and azimuth. In other words, although a direct measurement of f at any point would yield a value that depends on r, θ, ϕ and t (in the convection zone, at least), the average structure, to which the oscillation frequencies are sensitive, is axially symmetric: $f = f(r, \theta)$. In fact, most of the quantities specifying the solar structure (e.g. sound speed) have a spherically symmetric distribution to a very good approximation. In these cases the true solution is a function of radius, r, alone.

For quantities, such as angular velocity, that are strongly latitudinally dependent the reduction to 1-D inversions can be achieved by expanding the latitudinal variation in terms of a set of orthogonal basis functions, for each radius value (Ritzwoller and Lavely 1991). This reduces the 2-D inversion for $f(r, \theta)$ to an (in principle) infinite sequence of 1-D inversions of the type dealt with in this thesis, one for each component of the latitudinal expansion.

It now only remains to discuss the continuity of the 'true solutions'. Certainly any quantity $f(r)$ is defined for every value of r. In fact, it is reasonable to assume that the quantities are actually continuous functions of radius, based on the following argument. Consider sound speed (or temperature) and angular velocity. If there were a discontinuity in either of these quantities heat or momentum transport (respectively) would immediately remove the infinite gradient in these quantities: clearly, viscosity prevents infinite velocity gradients, and similar considerations (concerning the conductivity) apply to sound speed (which is very closely related to temperature). Having said this, the actual gradients may be very large (for example, the angular velocity is believed to vary sharply across the base of the convection zone), so large, in fact, that they may be well approximated, in practice, by discontinuities. The gradient will, though, be limited by the physical considerations mentioned. This will be discussed below.

2) Mathematically: In order to place the solution of (1.32) on a firm theoretical footing and to be as mathematically rigorous as possible it is necessary to specify the set of objects (functions) that will be allowed as solutions. This set must be large enough to include any conceivable 'true solution', as described above.

The nature of the problem as a set of integrals involving $f(r)$ suggests that the broadest possible class of solution functions is $L^2(0, R_\odot)$ – the set of all (Lebesgue) square-integrable functions (Weir 1973). In fact, this set is larger than necessary for practical purposes, but it is useful to have L^2 in mind when greater specificity is not required. No use will be made, in this thesis, of any properties of L^2 other than integrability and linearity ($L^2(0, R_\odot)$ is a vector space). The fact that the true solution is continuous means that we could use a space of all continuous functions on $[0, R_\odot]$, but, as the practical solution of (1.32) often employs approximation by functions with discontinuities (PCD – see §1.9), continuity will only be assumed when necessary to clarify an argument.

In general, no specific reference will be made to the space of functions to be allowed as solutions of (1.32); it is the purpose of this section to show that such a space does exist for any practical application, and, where definiteness is required, it may be assumed to be L^2 (in the absence of more information). The only property of the function space of solutions that will be used here is that of linearity. As the problem is linear it makes admirable sense to adopt such an assumption. (This is not practically necessary, and, in fact, it may not always be desirable – see the discussion of this point in §2.2.3.)

Finally, it is helpful to comment on the nature of the kernel functions. Again, these must certainly be in $L^2(0, R_\odot)$, and often they will be continuous (in heliosesimology, the kernel functions are closely related to the solutions of the differential equations describing the oscillations, and so must be differentiable). For the practical solution of (1.32) it is useful to assume that kernels and solution lie in the same space (this is essential if spectral expansion is to have any validity as a method of solution, for example, because it is assumed that the kernel functions can be used to span the solution space), and, where necessary, this assumption will be implicitly adopted.

3) Practically: Having an intuitive grasp of the physical nature of the true solution of (1.32) and a knowledge of the space of objects that are acceptable as solutions is not quite enough to complete the mental picture of the meaning of 'a solution' to (1.32). 'A solution' will be a function and the true solution will be continuous (although it may be approximated by a function with discontinuities), but it does not describe the quantities that result from the actual practical solution of (1.32). Solving (1.32) must be done numerically (the kernels are only known numerically, as they are related to the numerical solutions of the differential equations describing the oscillations, and anyway, the problem just could not be solved analytically, even if the kernels were analytic), and so the resulting solution must be characterized by a finite number of values, which may be the values of f at a finite set of points in the range 0 to $R_\odot$, or the values of coefficients of the expansion of $f(r)$ in terms of some basis of analytic functions. Which of these two approaches is adopted is largely a matter of personal choice. Attempting to recover the values of f at several points in $[0, R_\odot]$ allows complete freedom to specify these values independently, but gives no information about points inbetween. Moreover, the assumption of continuity (or slow variation/smoothness) allows the values of $f(r)$ at intermediate points to be inferred by simple interpolation: If f is recovered at sufficiently many points, plotting those points on a graph gives an adequate meaningful representation of the solution in most cases.

If the solution is expanded in terms of analytic basis functions, then the solution is essentially given at every point once the coefficients in the expansion are known. (This ignores the fact that exact calculation of the value of the basis functions at any point may not be trivial, and may be an infinite process: it is assumed that it is a simple matter to obtain the values of the basis functions to sufficient accuracy). However, knowledge of only a finite number of coefficients results in the values of the solution at different points being dependent. Note that to provide a graphical representation of the solution, it is, in general, necessary to select a number of values of r and plot the value of the solution at these points, filling the gaps by interpolating by eye.

Thus, there is little difference in the final analysis between the actual solution obtained via the two approaches: both essentially result in a picture (graph) of the solution which includes information about a finite number of points. The choice is between finding the solution only at several points without explicit approximation and then introducing approximation at intermediate points, or explicitly approximating the solution at the outset and obtaining the approximate value of the solution everywhere.

In concluding this section, it should be noted that the physical nature of the 'true solution' as essentially continuous means that members of the function space of 'acceptable' solutions which contain discontinuities or unphysically sharp variation should be prevented from appearing as the recovered solution of (1.32). In practice, this often means penalizing against sharp variation in some way, i.e. deliberately biasing the solution in favour of functions that have the property of 'smoothness' in some sense. Some methods of solution (the PCD method, for example, see §1.9) introduce solution functions with discontinuities, but such solutions can be thought of, when necessary, as convenient approximations to physical solutions that *are* continuous and smooth.

1.7 What do we require from a method of solution?

Obviously, the basic accuracy and stability of inversion algorithms is important. In certain circumstances it may be the case that the information required from an inversion is so important that no effort is too great, that no expense should be spared in the recovery of the most accurate solution possible. For example, at the moment the data is not quite sufficient to make definite statements about the structure of the solar core, which might solve the solar neutrino problem. The more information about the solar core that can be extracted the better. However, helioseismological inversions are computationally expensive, so it certainly pays to try to reduce the computational burden as far as possible. Obviously, a faster inversion scheme will be 'better' than a slower one, all other things being equal. In general, of course, a compromise must be made between speed and accuracy. This thesis addresses problems of accuracy, with little regard for the amount of computing involved. (Once methods for optimizing the inversion have been found, attempts can be made to speed them up.) Having said this, where speed and ease of computation may easily be increased this is done. (For example, a major part of all inversion algorithms presented here is the inversion of symmetric positive semi-definite matrices of the form $A + \lambda B$ for many different values of the parameter λ. A method for performing this calculation efficiently, the *Fix-Heiberger algorithm*, is given in §2.4.)

Another, very important, aspect of any numerical inversion scheme is the amount of storage required for its operation. The smaller the memory requirements of the algorithm, the more computer systems are likely to be able to run it, that is, the more people will have access to the algorithm. It is clearly undesirable to have an algorithm which can be used only by a very limited group of researchers. Although questions of storage space are given little consideration here, it should be noted that the program used to perform the inversions in this thesis was written with storage in mind, although the storage requirements are still not small.

It goes without saying that simplicity is an advantage in any algorithm. Again, simplicity is related to accessibility, but there are other factors. For example, complicated algorithm are difficult to program, and they tend often to be rather highly strung. Robustness is a valuable asset in any numerical procedure, and simplicity can be an advantage in this regard.

Finally, any inversion produces information other than the actual solution, such as errors, correlations and bias levels. This information is often very important, or even essential, and a procedure that provides more information will be better than one that produces less. The relative value of extra information is somewhat subjective, but there are some things that are, without question, very important. Any solution that does not come with error bars, for example, is practically useless (the solution could be wildly inaccurate due to data errors or bias).

1.8 Optimal Linear Averaging (OLA)

Optimal averaging was invented by Backus and Gilbert (1967, 1968, 1970) for solving geophysical inverse problems. It was improved and extended by Pijpers and Thompson (1992), who introduced the term 'optimal linear averaging', emphasizing that the averages concerned are linear averages of the solution (*cf.* equation (1.35)). It exploits the linearity of the problem (intrinsic or deriving from the linearization about some specific model) to search for and locate linear combinations of the kernels which are strongly localized near some chosen point r_0 (i.e. some combination $K_{r_0}(r) = \sum_{i=1}^{m} \alpha_i(r_0)k_i(r)$ that is large near r_0 and small elsewhere). The same linear combination of the data gives the average of f weighted by $K_{r_0}(r)$, which is obviously going to be dominated by the contribution from $f(r)$ for $r \approx r_0$, i.e. $\bar{f}(r_0) = \sum_{i=1}^{m} \alpha_i(r_0)g_i \approx f(r_0)$ provided K_{r_0} is sufficiently well localized about r_0. The idea, then, is to find, for any 'target' point r_0, the set of constants $\alpha_i(r_0)$ such that $K_{r_0}(r) = \sum_{i=1}^{m} \alpha_i(r_0)k_i(r)$ is most strongly peaked about r_0. For a description of how to stabilize this inversion against the effects of data errors see Backus and Gilbert (1968), or Gough (1985).

This method has considerable appeal for many problems, and is very useful for determining limits on the resolution obtainable with a particular data set, and the extent to which errors corrupt the solution. It does, though, suffer from the drawback that it can be computationally quite expensive (requiring the inversion of an $m \times m$ matrix, in contrast to the $n \times n$ matrix inversion ($n < m$, and often $n \ll m$) required by the regularized least squares methods to be discussed, and in some of its manifestations (although not the SOLA method, Pijpers and Thompson 1992, which was introduced precisely to avoid this problem) it requires the inversion of such a matrix for each point, r_0, at which the solution is to be estimated.

In this section the fundamental principles of the optimal averaging method are described in a general way. The optimal averaging method (OLA) attempts to produce estimates of $f(r_0)$ at a finite number of points with $0 < r_0 \leq R_\odot$. It must do this using only knowledge of the measured data (including the general properties of the error distribution) and the kernel functions (and the basic properties of the true solution described in §1.6). Each kernel function determines, through equation (1.32), the amount that the value of f at any point, r, contributes to the corresponding data value: the ith data point g_i is a linear combination (integral, weighted by the kernel $k_i(r)$) of the values of f at every r. Solution by OLA amounts to an attempt to separate out (i.e. resolve) the value of f at r_0 from the values at other r. Such a feat, should it be achieved, would be equivalent to finding a linear combination of the values of f at all r with weighting function $\delta(r - r_0)$, i.e.

$$f(r_0) = \int_0^{R_\odot} \delta(r - r_0) f(r)\, dr. \tag{1.33}$$

So, we seek a particular linear combination of the values of f and the only information we have about f is a collection of other linear combinations. This suggests that we use the combinations we have to form another combination that at least resembles (1.33), that is, form the linear combination

$$\sum_{i=1}^{m} \alpha_i(r_0) \int_0^{R_\odot} k_i(r) f(r)\, dr = \int_0^{R_\odot} K(r_0; r) f(r)\, dr \approx \int_0^{R_\odot} \delta(r - r_0) f(r)\, dr \tag{1.34}$$

where $K(r_0; r) = \sum_{i=1}^{m} \alpha_i(r_0) k_i(r)$ is the new weighting function which is supposed to resemble a δ-function as closely as possible. Define

$$\hat{f}(r_0) = \int_0^{R_\odot} K(r_0; r) f(r)\, dr. \tag{1.35}$$

Then (1.34) and (1.33) together give

$$\hat{f}(r_0) = \sum_{i=1}^{m} \alpha_i(r_0)\, g_i \approx f(r_0) \tag{1.36}$$

for error free data. Errors introduce additional considerations which are dealt with in Backus and Gilbert (1970), and in Gough (1985).

Note the different viewpoint from many expositions of OLA methods. Whereas $K(r_0; r)$ is often introduced *ab initio*, here it is the result of the attempt (1.34) to form a linear combination of known quantities giving a δ-function-weighted average of $f(r)$. The ultimate goal remains the same, though: constants $\alpha_i, i = 1, \ldots, m$ must be found that give an averaging kernel $K(r_0; r)$ resembling a δ-function as closely as possible. The main properties of the δ-function (which result in the equality (1.33)) are

$$\begin{aligned} \int_0^{R_\odot} \delta(r - r_0)\, dr &= 1 & (1.37) \\ \int_a^b \delta(r - r_0)\, dr &= 0 \text{ for } \textit{any} \text{ interval } [a, b] \text{ not containing } r_0 \,. & (1.38) \end{aligned}$$

Ideally, therefore, the $K(r_0; r)$ we seek is the one satisfying

$$\begin{aligned} \int_0^{R_\odot} K[\boldsymbol{\alpha}(r_0); r]\, dr &= 1 & (1.39) \\ \int_a^b K[\boldsymbol{\alpha}(r_0); r]\, dr &= 0, & (1.40) \end{aligned}$$

where $K[\boldsymbol{\alpha}(r_0); r]$ is simply used as an alternative notation for $K(r_0; r)$).

Since $K(\boldsymbol{\alpha}; r) = \sum_{i=1}^{m} \alpha_i k_i(r)$ is a linear combination of a finite number (m) of functions (the kernels) that are bounded on $[0, R_\odot]$ it can never perfectly recreate a δ-function. So conditions (1.39) and (1.40) need modification. (1.39) must be maintained since it ensures that K is not identically zero, or, equivalently, it ensures that $\boldsymbol{\alpha} \neq 0$ (which is clearly a necessary condition), and provides a normalization guaranteeing that $\hat{f}(r_0) \to f(r_0)$ (for noise-free data) as $K(\boldsymbol{\alpha}; r)$ becomes more sharply peaked about r_0. Condition (1.40) therefore needs to be relaxed: $K(\boldsymbol{\alpha}; r)$ must be allowed to be non-zero for $r \neq r_0$ (which would allow the existence of intervals $[a, b]$ not containing r_0 for which (1.40) does not hold). (1.40) must be replaced by some less rigid constraint which will ensure that $K(\boldsymbol{\alpha}; r)$ bears the closest possible resemblance to $\delta(r - r_0)$. What should this condition be?

The fact that (1.40), and hence the approximate equality in (1.34) cannot be made to hold exactly is equivalent to the statement that resolution is finite: the value of f at r_0 cannot be resolved from the values at other r. This clearly follows from (1.40), since if there is an interval $[a, b]$, not containing r_0, for which $\int_a^b K(r_0; r)\, dr \neq 0$, then $\hat{f}(r_0)$ will be sensitive to values of f for $a \leq r \leq b$ (i.e. $r \neq r_0$). This means that choosing a constraint to replace (1.40) amounts to deciding which points in $[0, R_0]$ $\hat{f}(r_0)$ should be allowed to be sensitive to. Certainly it would be desirable for $K(r_0; r)$ to be large for $r = r_0$ and small elsewhere. At this point the presumed continuity of the true solution, discussed in §1.6, becomes important. Continuity means that for r close to r_0 $f(r)$ must be close to $f(r_0)$, so that if only a small region around r_0 contributes to $\hat{f}(r_0)$, then $\hat{f}(r_0)$ ought to be very close to $f(r_0)$. This suggests introducing a constraint that penalizes against averaging kernels which are large away from r_0. This constraint, combined with (1.39), forces $K(r_0; r)$ to be large near r_0. To be more specific: a good averaging kernel, $K(r_0; r)$, is one which is large near r_0 and small elsewhere, i.e. which is sharply peaked about r_0. (This approach leads naturally to the MOLA method, outlined in Gough 1985, and in Pijpers and Thompson 1992.) The constraint we require is one which maximizes some measure of 'peakedness'. One way to enforce peakyness is to use a functional

$$J_M(\alpha; r_0) \overset{\text{def}}{\equiv} \int_0^{R_\odot} \{K[\alpha(r_0); r]\}^2 (r - r_0)^2 \, dr \tag{1.41}$$

which is the original form of the criterion for choosing peaky averaging kernels (Backus and Gilbert 1967; Gough 1985). Despite its simplicity and intellectual appeal, the definition (1.41) is rather unfortunate, because its practical implementation requires the inversion of a large $(m \times m)$ matrix for *every* point, r_0, at which the solution is to be recovered. Pijpers and Thompson (1992) proposed a modification to this method, which alleviates this difficulty by using an alternative definition of peakyness. This will now be considered.

Another way to see that the constraint we require involves peakedness is to consider one formal definition of the δ-function. A δ-function is (the limit of) any sequence $\phi_1, \phi_2, \ldots$ of functions satisfying (1.39) and $\lim_{n\to\infty} \phi_n(r) = 0$ for $r \neq 0$, so (1.40) follows in the limit (see Lighthill 1958). $K(r_0; r)$ can never be a δ-function, but if close approximation to a δ-function is required, then searching for an averaging kernel close to a function far along some sequence defining a δ-function would be appropriate. As an example, consider the sequence

$$\phi_n(r) = \frac{n}{\sqrt{2\pi}} e^{-\frac{n^2 r^2}{2}} \tag{1.42}$$

of gaussians. The larger n the better the approximation to a δ-function. Finding a $K(r_0; r)$ which approximates $\phi_n(r - r_0)$ for some large n would result in a good averaging kernel. The larger n the better the kernel. Note that, for large n, the 'peakedness' of $\phi_n(r - r_0)$ about r_0 increases with n. This is true of any such sequence. So, again, we are lead to a measure of peakedness. This time, though, the obvious functional to choose is (Pijpers and Thompson 1992)

$$J_S(\boldsymbol{\alpha}; r_0) \overset{\text{def}}{\equiv} \int_0^{R_\odot} \{K[\boldsymbol{\alpha}(r_0); r] - \phi_\tau(r_0)\}^2 \, dr, \tag{1.43}$$

where $\phi_\tau(r_0)$ is some *target function* sharply peaked around r_0 (for example, $\phi_\tau(r_0)$ could be some member of the sequence (1.42) with large n). Obviously, the effect of (1.43) is to force the averaging kernel to be as much like the target function as possible. The parameter τ that has been introduced here is intended to control the width of the target function, and thus control the peakyness of the averaging kernels found. As mentioned above, the biggest advantage of this modification of the original optimal averaging procedure is that it removes the need to invert a large matrix for every recovery point, r_0, requiring such an inversion to be performed only once.

Optimal averaging will not be considered further here (but see §2.4, where a method for diagonalizing symmetric matrices like those occurring in OLA inversions is presented).

1.9 Regularized Least Squares (RLS)

Given that it is not possible to find an analytic solution to equation (1.32), it behoves us to look for a numerical procedure for obtaining an approximate solution. The OLA techniques reviewed in the previous section are examples of one such procedure, here we will consider a different approach. The OLA methods approach the solution of (1.32) in a more physical, practical way. There is an alternative formalism which treats the inversion more as a mathematical problem. The basis of all the RLS methods for solving (1.32) to be described in this section is the observation that the integral in (1.32) effects a mapping from the infinite-dimensional *vector* space of possible solution functions to the finite-dimensional space of data vectors: (1.32) is therefore a kind of infinite-dimensional matrix equation. Finite-dimensional matrix equations are ideally suited to numerical solution on computers, infinite-dimensional matrix equations are not. However the power of the many numerical techniques for dealing with matrices and other aspects of linear algebra suggests that developing a method of solution that makes use of them could be very advantageous. How can (1.32) be reduced to a simple matrix problem? Clearly, some approximation is needed. This approximation of (1.32) by a (finite-dimensional) matrix equation will be called *discretization*, as it reduces the continuously infinite degrees of freedom in the solution function to a finite number of degrees of freedom in a solution *vector*.

First, notice that since there is only a finite number (m) of data points available we can never recover all the degrees of freedom in the solution. There must be some finite-dimensional subspace of the solution space about which the data contain information, and an infinite-dimensional subspace which can never be fixed by the data. We might as well concentrate on the subspace that we can learn about. In other words, ignore the unknowable subspace and assume that what we are looking for lies wholly within the knowable space, so that the effective solution space is finite-dimensional and (1.32) really reduces to a matrix equation (once some basis for the solution space has been selected). This is the basis of the spectral expansion method, described below.

Second, in section 1.9.5, below, it will be explained how the data errors generally affect the recovered solution of inverse problems such as (1.32) by introducing very large, spurious, short-wavelength oscillations. This is a reflection of the fact that components in the solution that vary over very short scales are often mapped to very small values in the data space (this is essentially a statement of the Riemann-Lebesgue lemma, see Craig and Brown 1986), and, as a result, their contribution to the data is often swamped by the data errors. These components of the solution therefore become effectively unknowable. Just as before, we might as well concentrate on the part of the solution space that does not contain functions with such rapid variations. There are two alternative approaches to this. Piecewise-constant discretization deals with the short length-scale features by explicitly assuming that the solution is constant over finite intervals (bins) – it operates in 'real' space – whereas the function expansion methods remove short-wavelength features by assuming that there is some lower limit to the characteristic length-scale of features that can be recovered, and this is reflected in the method of solution by ignoring any basis function in the solution space with wavelength shorter than this – the function expansion methods work essentially in 'Fourier' space.

The different ways of making the necessary approximation to (1.32) in common use can be divided into four subcategories, each relying on one of the principles outlined above. These will be described in the following section. Once that is done, §1.9.5 will describe how to solve the resulting matrix equation using regularization. Section 1.9.6 then briefly reviews methods for setting the level of regularization to be applied in any particular problem.

1.9.1 Spectral expansion (SE)

This method is reviewed in Gough (1985). It is based on the observation that any component $f_\perp(r)$ of $f(r)$ that is orthogonal to all of the kernels $k_i(r)$, in the sense that $\int_0^{R_\odot} k_i(r)f_\perp(r)dr = 0$ for $i = 1,\ldots,m$, is (obviously) not determined by the data. Thus, the only part of $f(r)$ that can be determined from the data is the function $f_{||}(r)$, no component of which is orthogonal to all the kernel functions. The solution space is an infinite-dimensional function space, which contains the kernel functions. (It is invariably mathematically convenient to assume that the kernel functions lie in the solution space, and in practice the fact that (1.32) involves the integral of a product of two functions immediately forces us to make the assumption that the solution space is $L^2(0,R_\odot)$ – the space of square-integrable functions on the interval $(0,R_\odot)$ – which is the weakest assumption that makes reasonable mathematical sense. $L^2(0,R_\odot)$ certainly contains any suitably continuous kernel functions. In helioseismology the kernels are always sufficiently well behaved to lie in $L^2(0,R_\odot)$.) As a result, the kernels span a subspace of the full solution space. The kernels are linearly independent (otherwise the data would not be independent) so the kernels are a basis of this subspace. Any solution function can be broken down into a component, $f_{||}(r)$, lying wholly within this subspace (parallel to it, in other words), and a component, $f_\perp(r)$, which is orthogonal to it, and therefore orthogonal to all the kernel functions. The component that lies within the subspace must be expandable in terms of the basis of kernel functions. The 'knowable' component of the solution can therefore be expanded in terms of the kernels k_j:

$$f_{||}(r) = \sum_{j=1}^{m} f_j k_j(r), \tag{1.44}$$

where the f_j are the components in the expansion in terms of the k_j. Then (1.32) becomes

$$g_i = \sum_{j=1}^{m} \left\{ \int_0^{R_\odot} k_i(r)k_j(r)dr \right\} f_j,$$

or, defining *kernel matrix*,

$$H_{ij} = \int_0^{R_\odot} k_i(r)k_j(r)dr, \tag{1.45}$$

and introducing vector notation for the components in the expansion of $f_{||}$ and for the data values,

$$\mathbf{g} = H\mathbf{f} \tag{1.46}$$

with H a (symmetric) $m \times m$ matrix.

Solution of this matrix equation for $\mathbf{f}$ then allows $f_{\|}$ to be found easily from (1.44). We may as well assume that $f_{\|}$ is f, i.e. assume $f_{\perp}(r) \equiv 0$ since we cannot determine $f_{\perp}$ (zero is as good a value as any).

Equation (1.46) shows that spectral expansion fits very comfortably into the RLS formalism described below. The problem of regularizing the spectral expansion method does not seem to have been solved satisfactorily (see Gough 1985), but chapter 2 of this thesis sheds some light on this issue.

The difficulty with this method is the need to invert an $m \times m$ matrix (often $m \gtrsim 10^3$), which can be very computationally expensive..

1.9.2 Orthogonal function expansion (OFE)

The space of solution functions, which, as was stated above, can be taken to be the Hilbert space $L^2(0, R_\odot)$ (see Weir 1973), has many possible orthogonal bases, for example, the Fourier basis consisting of sine and cosine functions. The full (infinite) basis contains functions with short-wavelength components that could never be determined in an inversion. The principle underlying this method therefore involves essentially choosing some shortest-wavelength and assuming that all components of the solution function belonging to basis functions with wavelengths shorter than this are zero.

So, the solution f is expanded in terms of some finite set of orthogonal functions such as sines or cosines. In general, let us choose a set of n functions $\phi_j(r)$, for $j = 1, \dots, n$ (obviously $n \leq m$, since we can determine at most m independent parameters) and write

$$f(r) = \sum_{j=1}^{n} f_j \phi_j(r), \tag{1.47}$$

where the f_j are the free parameters to be determined. Then (1.32) becomes

$$\mathbf{g} = H\mathbf{f}, \tag{1.48}$$

where the definition

$$H_{ij} = \int_0^{R_\odot} k_i(r)\phi_j(r)dr \tag{1.49}$$

of the kernel matrix has been used. Note the obvious similarity between (1.47) and (1.44), and (1.48) and (1.46). In practice the differences are that the ϕ_j are orthogonal, whereas the k_i are not, and $n < m$ usually, so that for orthogonal function expansion an $n \times n$ matrix, rather than an $m \times m$ matrix must be inverted. $n \sim 100$ usually gives a sufficiently accurate approximation in the cases considered here. With $m \sim 10^3$, as it typically was for the inversion performed here, this is a very considerable improvement (the speed of matrix inversion or eigenanalysis goes as N^3).

1.9.3 Non-orthogonal function expansion (NOFE)

This is very similar to OFE, except that the basis functions, ϕ_j, are not required to be orthogonal (although they must be linearly independent, of course). Equations (1.47) to (1.48) hold as before, and the kernel matrix is again given by $H_{ij} = \int_0^{R_\odot} k_i(r)\phi_j(r)dr$. Typically, the basis functions would be the polynomials $\phi_j(r) = r^{j-1}$. In general, the discretized problem is very ill-conditioned, and this method is usually only used to perform low-order parametric fitting to the solution. It is possible to use it with regularization, but there are few real advantages to this. It is included here only for completeness.

1.9.4 Piecewise constant discretization (PCD)

The trick in this case is to choose some number n ($\leq m$) of *discretization points*, X_j, for $j = 0, \ldots, n$ such that $0 = X_0 < X_1 < \ldots < X_n = 1$ are chosen. On each sub-interval $(X_{i-1}, X_i]$ we assume that $f(r)$ is approximately constant, $f(r) \approx f_j$, thus immediately preventing small scale variations over this region. We may express this mathematically as

$$f(r) = \sum_{j=1}^{n} f_j S_j(r), \tag{1.50}$$

where

$$S_j(r) = \begin{cases} 1 & \text{if } X_{j-1} < r \leq X_j, \\ 0 & \text{otherwise.} \end{cases}$$

Hence

$$\mathbf{g} = H\mathbf{f}, \tag{1.51}$$

where the kernel matrix is

$$H_{ij} = \int_0^{R_\odot} k_i(r) S_j(r) dr \equiv \int_{X_{j-1}}^{X_j} k_i(r) dr. \tag{1.52}$$

Again we usually have $n < m$, so that the inversion of an $n \times n$ matrix is all that is required for solution, which is much easier than for spectral expansion.

It is clear from equations (1.46), (1.48) and (1.51) that all of the discretizations described here are basically different realizations of some more general method. This idea is considered in more detail in chapter 2.

1.9.5 Regularization

All of the methods for discretizing (1.32) result in a matrix equation relating the data vector, $\mathbf{g}$, to the solution vector, $\mathbf{f}$, the significance of which for the approximate solution function, $f(r)$, is determined by the expansion of $f(r)$ in terms of the basis functions of the discretization (*cf.* equations (1.44), (1.47), and (1.44)). The goal now, then, is to find the vector $\mathbf{f}$ that satisfies the matrix equation

$$\mathbf{g} = H\mathbf{f} \tag{1.53}$$

for a given $\mathbf{g}$, where the kernel matrix, H, is $m \times n$.

It is in order to make a few obvious comments about the existence and uniqueness of solutions to (1.53). We always assume that $n \leq m$, because there is no point attempting to fix more parameters than there are data points available. Recalling the discussion in §1.4, introduce norms on the vector spaces containing $\mathbf{f}$ and $\mathbf{g}$, and define λ_{min} and λ_{max} as in that section. H, $\mathbf{f}$ and $\mathbf{g}$ are just collections of numbers, so λ_{max} is finite. If $\lambda_{\text{min}} = 0$ then there exist non-zero solution vectors that are mapped to the zero vector by H: the kernel matrix does not have full rank (n). If $\lambda_{\text{min}} > 0$ the kernel matrix does have full rank. The rank of H is important because it determines the uniqueness (or lack of it) of the solution to (1.53). As stated in §1.4, the solution will be unique if and only if $\lambda_{\text{min}} > 0$. For perfect (error-free) data, there always exists an exact solution to (1.53) (by an exact solution is meant a solution that satisfies the data exactly, even if the data is erroneous). In general, this solution will not be unique, as explained above. When the data is contaminated with errors (1.53) will, in general, have *no* solution. This is because the 'correct' data is constrained to lie within the subspace of the space of data vectors that can be reached by vectors $H\mathbf{f}$, for $\mathbf{f}$ in the solution space. It is well known that the image of the solution vector space under the linear mapping represented by H (that is, the set of data vectors that can be written as $\mathbf{g} = H\mathbf{f}$ for some $\mathbf{f}$ in the solution space) has dimension at most equal to the dimension of the solution space n ($\leq m$). In fact, the dimension of this image space is exactly the rank of the matrix H (see Blyth and Robertson 1986), which is at most n. This means that if $n < m$ or H has rank less than n the image of the solution space will not be the whole of the data space. The 'correct' data is constrained to lie within the image space, of course, but the contribution of the errors to the data can lie anywhere within the data space (they are not constrained by the relationship (1.53)), so the observed data can (and usually will) lie outside the image space. This means that, in general, for erroneous data $\mathbf{g}$ there will be no $\mathbf{f}$ such that $\mathbf{g} = H\mathbf{f}$: an exact solution does not exist.

Given, though, that we know the non-existence of a solution is due entirely to errors in the data, it makes sense to allow solutions that do not exactly fit that data. The errors on the data will usually be small (otherwise obtaining a solution would be essentially pointless), so it makes sense to seek the vector $\mathbf{f}$ that gets closest to fitting that data exactly (although see §1.9.6 below), and defining this to be the solution. To do this requires some measure of nearness on the space of data vectors, but we have already assumed that some norm exists on this space, so define that solution of (1.53) to be the $\mathbf{f}$ that minimizes

$$\chi^2(\mathbf{f}) \overset{\text{def}}{\equiv} \|\mathbf{g} - H\mathbf{f}\|_D^2. \tag{1.54}$$

When an exact solution exists the minimum of χ^2 will be zero (recall property 1 of norms on page 20). Unfortunately, there are many different norms that can be defined on a vector space, and each will give rise to a different solution in (1.54). Is there one choice that is better than the others? In general, there is, but the particular choice depends upon the statistical properties of the data errors. The errors on helioseismic data for the rotation and structure problems are *gaussian* errors. That is, the error on the ith data point has a normal distribution with mean zero and some variance σ_i^2. It is well known that for such errors the norm to use is the usual sum of squares norm (or euclidean norm, or 2-norm),

$$\|\mathbf{g}\|^2 = \mathbf{g}^T\mathbf{g} = \sum_{i=1}^{m} g_i^2, \tag{1.55}$$

because for gaussian errors minimizing χ^2 in (1.54) with this norm results in a *maximum likelihood* estimate of the solution (see *Numerical Recipes*, §15.1, and Anderson *et al.* 1990). With this norm, the method of solution is known as the method of least squares (for obvious reasons), and the solution obtained is called the *least squares estimate* of the true solution. When the errors are not gaussian other norms may give rise to maximum likelihood estimates of the solution, and therefore be more appropriate, but this will not concern us here.

There is one slight technicality that must still be dealt with. When the data errors are not uniform from data point to data point (that is, when the variances, σ_i^2, of the errors on different data points are different) the norm in (1.55) does not, strictly speaking, give rise to a maximum likelihood estimate. The norm must instead be replaced by

$$||\mathbf{g}||^2 = \sum_{i=1}^{m} \left(\frac{g_i}{\sigma_i}\right)^2 \tag{1.56}$$

(see *Numerical Recipes*, §15.1). With this norm, χ^2 in (1.54) becomes the standard 'chi-square', hence the notation. Henceforth it will be implicitly assumed that the norm under consideration is the weighted euclidean norm (1.56).

The reformulation of (1.53) as the minimization of χ^2 in (1.54) ensures that a solution can be found, but it does nothing to alleviate the problem of ill-posedness. Observe that $\mathbf{f}$ appears in χ^2 only in the term $H\mathbf{f}$. This means that if non-uniqueness occurs ($\lambda_{\text{min}} = 0$, and H not of full rank) adding any vector mapped to zero by H to a solution obtained by minimizing χ^2 will not change the value of χ^2, and so will be an equally valid solution. Similarly, if $\lambda_{\text{min}} > 0$ but very small, the problem will be unstable. The easiest way to see this is to define vectors $\mathbf{x}_{\text{min}}$ and $\mathbf{x}_{\text{max}}$ 'corresponding to' λ_{min} and λ_{max}, just as in §1.4, and to imagine that the true solution is $\mathbf{x}_{\text{max}}$ and the errors on the data correspond to a solution error proportional to $\mathbf{x}_{\text{min}}$ (certainly, for any solution vector there is an associated data vector, so this is not an unreasonable assumption). Then exactly the same argument as that used in §1.4 shows that when λ_{min} is very small the data errors can be magnified enormously: the problem is still ill-posed.

It turns out in practice, and can be shown with the help of the Riemann-Lebesgue lemma (Craig and Brown 1986, §4.2) that, under fairly general circumstances, components of the solution function in (1.32) that oscillate rapidly correspond to very small data values, so that even if these components are quite large they may actually make a contribution to the data that is smaller than that made by the errors, and it may therefore be impossible to determine the contribution made by such components to the solution. Equation (1.53) approximates (1.32) and tends to show similar difficulties. If the space of solution functions obtained from the solution vector via the expansions (1.44), (1.47), or (1.50), contains functions that oscillate rapidly, then the kernel matrix H will have a small value of λ_{min}, with $\mathbf{x}_{\text{min}}$ being the solution vector corresponding to one of these oscillatory components. (If a solution function $f(r) = \sum_j f_j\phi_j(r)$, with $||\mathbf{f}||_S = 1$ for simplicity, is mapped to a data vector $\mathbf{g}$ with $||\mathbf{g}|| = \varepsilon$ very small, then, of course, $\lambda_{\text{min}} \leq \varepsilon$, so that H is ill-conditioned, and any component of the solution vector proportional to $\mathbf{x}_{\text{min}}$ will be very difficult to determine from erroneous data.) In practice, the result of minimizing (1.56) to obtain a solution to (1.32) when H is ill-conditioned will be to introduce large, spurious variations in the solution: a quite small component in the data errors that is proportional to $H\mathbf{x}_{\min}$ will give rise to a very large $\mathbf{x}_{\min}$ component in the solution vector, and therefore a very large oscillatory component in the solution function.

The way around this problem is to recognize that, since these components cannot be determined we might as well pretend they are zero, and look for solutions that do not have such components. For the solution function this means looking for solution functions that are not highly oscillatory or rapidly varying. Functions that do not vary rapidly can be described as being 'smooth', so what we require is some way to give a preference to smooth functions. This way of thinking about removing the instability of (1.32) is studied in more detail in chapter 2. Here we will concentrate on the discretized form of the problem, (1.53).

Assume that some (positive) functional, $\Phi(\mathbf{f})$, of the solution vectors has been chosen that assigns a low value to those solution vectors that correspond to smooth solution functions, and a large value to those solution functions that are 'rough'. Then, looking for the minimum over all solution vectors of

$$\chi^2(\mathbf{f}) + \lambda\Phi(\mathbf{f}) \tag{1.57}$$

(where λ is a free parameter, called the *smoothing parameter*, which controls the extent to which the solution is forced to be smooth, as opposed to fitting the data as well as possible) will clearly tend to pick out smooth solutions in preference to rough ones, for a given fit to the data. It is well known that, if $\Phi(\mathbf{f})$ is reasonable and λ is set to a reasonable value (see the following section), the solution obtained from the minimization of (1.57) is stable, and can be quite accurate (Craig and Brown 1986, §6.2). The functional $\Phi(\mathbf{f})$ is called a *smoothing* or *regularizing functional*, and the use of (1.57) to obtain the solution to (1.32) is known as *regularization*.

It only remains to say what is meant by a 'reasonable' $\Phi(\mathbf{f})$. The goal of $\Phi(\mathbf{f})$ is to penalize against solution vectors corresponding to non-smooth solution functions. One choice for $\Phi(\mathbf{f})$ just says look for solution for which the solution vector is small (if $\mathbf{f}$ is small, $f(r)$ will also be small – since they are linearly related – and a function that has small values everywhere cannot have large amplitude variations). This approach is commonly used with all of the discretization methods outlined above. It is called *zeroth order smoothing* (because it seeks to minimize the 'zeroth' derivative of the solution function –

this will be clearer in a moment). Another way to think of choosing $\Phi(\mathbf{f})$ begins with the observation that if a function has large and rapid variations, so do its derivatives, in general. Looking for solution functions with small derivatives is therefore a way to impose smoothness. For the spectral expansion and function expansion discretizations it is quite difficult to see how to relate a constraint on the derivative of the solution function to a constraint on the solution vector (although see chapter 2, where this problem is considered in some depth), and for this reason this type of smoothing is not generally used with these discretizations. However, in PCD the values of the solution vector are exactly the values of the solution function on the corresponding discretization bins. This makes it possible to use some kind of difference scheme to define approximate derivatives in terms of the solution vector. Without going into too much detail (because there's plenty of that in chapter 2) a derivative of order p, say, can be defined by the operation of some difference matrix $\Delta^{(p)}$ on the solution vector, so that $\Delta^{(p)}\mathbf{f}$ somehow represents the pth derivative of $\mathbf{f}$ (the exact form of the matrix $\Delta^{(p)}$ is not needed here, but note that $\Delta^{(0)} = I$). Putting

$$\Phi(\mathbf{f}) = ||\Delta^{(p)}\mathbf{f}||^2 = \mathbf{f}^T\left(\Delta^{(p)^T}\Delta^{(p)}\right)\mathbf{f} \tag{1.58}$$

results in a smoothing functional that gives preference to solutions with small pth derivative, just as we required. The matrix $C^{(p)} \overset{\text{def}}{\equiv} \Delta^{(p)^T}\Delta^{(p)}$ is called the pth order smoothing matrix, and notice that for $p = 0$, $C^{(0)} = I$, the identity matrix. If we assume, just for simplicity, that the errors are uniform, $\sigma_i = \sigma$, for all i, the functional in (1.57) becomes

$$\frac{1}{\sigma^2}(\mathbf{g} - H\mathbf{f})^T(\mathbf{g} - H\mathbf{f}) + \lambda\mathbf{f}^T C^{(p)}\mathbf{f}$$

(it is a simple matter to renormalize the data and the kernels to reduce the problem with non-uniform errors to the same form), which is clearly quadratic in the solution vector. At a minimum of this functional the derivative of the functional with respect to any component of $\mathbf{f}$ must be zero. Taking the derivatives of the above functional with respect to all components of $\mathbf{f}$ and setting the results to zero results in the following expression

$$(H^TH + \lambda C^{(p)})\mathbf{f} = H^T\mathbf{g},$$

which can be solved to obtain an explicit expression for $\mathbf{f}$ by inverting the symmetric matrix on the left hand side, to get

$$\mathbf{f} = (H^TH + \lambda C^{(p)})^{-1}H^T\mathbf{g}. \tag{1.59}$$

There are other ways to define regularizing functionals to be used in (1.57), such as *maximum entropy* (Narayan and Nityananda 1986), or those described in Titterington (1985), but these will not be studied in any detail in this thesis.

Finally, what effect does the introduction of the smoothing functional have on the solution of (1.32)? The short answer is that it dramatically reduces the effect of the data errors on the values of the solution function, but at the expense of introducing *bias* into the problem. The effect of the data errors is reduced because the large errors in the solution function resulting from the large oscillatory components that the data errors inevitably introduce are suppressed by the introduction of a preference for smooth solution: the problem has been *stabilized* by the introduction of $\Phi(\mathbf{f})$. The recovered solution will vary quite slowly as the data is varied.

The cause of the bias is obvious. The true solution will not be perfectly smooth (flat), so among all the possible solutions that have the same χ^2 value the smoothing functional will tend to give a preference for those that are smoother than the true solution: for most possible values of the errors the recovered solution will be smoother than the true solution. The expected value of the solution over all possible realizations of the errors will then tend to be smoother than the true solution. This mismatch between the true solution and the expected value of the recovered solution is statistical bias.

In relation to this bias it is worth remarking on the close connection between discretization and regularization. In the absence of any more specific information the true solution to (1.32) may lie *anywhere* in the infinite-dimensional space of possible solution functions (which we will take to be $L^2(0, R_\odot)$, for the sake of argument). Discretization singles out a finite-dimensional subspace of the full solution space and demands that the recovered solution absolutely *must* lie within that discretization subspace, although it could lie anywhere within it. Regularization introduces a bias that drives the recovered solution to lie in a corner of the discretization space which is forever smooth, although if the data really demands it the recovered solution might be allowed to be rather non-smooth. It can be seen from this that discretization is, in a sense, a limiting, and intransigent, form of regularization. Consider the PCD discretization space, $\mathcal{P}$, which is defined to be the set of functions that can be written in the form (1.50), where the $S_j(r)$ are as given in §1.9.4, and is therefore n-dimensional. Define a functional $\Phi_c[f]$, *acting on the full solution space*, as follows:

$$\Phi_c[f] = \begin{cases} 0 & \text{if } f \text{ lies in } \mathcal{P} \\ c & \text{otherwise} \end{cases}$$

where c is a (large) positive constant. Using this functional in (1.57) will obviously give a very strong preference for solution functions that lie in the discretization space, $\mathcal{P}$. Letting c tend to infinity will obviously force the recovered solution to lie within $\mathcal{P}$, and so the effect of the regularizing functional $\Phi_c[f]$ is, for very large c, just the same as discretization.

1.9.6 Choosing the Smoothing Parameter

Having introduced the smoothing functional $\Phi(\mathbf{f})$ in the preceding section, in order to stabilize the solution of (1.53), it is necessary to give some guidance in the matter of setting the level of regularization to be imposed. The smoothing parameter can be varied between zero and infinity to give solutions with different degrees of smoothness and different qualities of fit to the data. It is easy to see qualitatively that when λ is very small the smoothing functional is largely irrelevant and the solution obtained will be the old, unstable unregularized solution, whereas when λ is very large the value of $\Phi(\mathbf{f})$ will dominate the functional in (1.57), and so the minimization will seek to find a smooth solution regardless of the fit to the data. Neither of these situations is desirable, and somewhere inbetween these extremes there ought to lie a value of λ that effects a suitable trade-off between the need to fit the data and the desire for a smooth solution. In this section several methods for choosing the value of λ that gives the best trade-off will be reviewed.

It is possible to adopt the attitude that, provided the effects of the smoothing applied are known and completely quantified, any value of the smoothing parameter can be chosen: if only very gross features in the solution are required, then the solution can be heavily over-smoothed, providing it is made clear that this is the case, and that the absence of small scale features in the recovered solution does not mean that they are absent in the true solution, whatever that may be. In chapter 4 various ways of quantifying the effects of regularization on the recovered solutions to (1.32) are studied, and these can be used to permit informed judgements about the appropriate choice of a smoothing parameter, and the effect of that choice on the solution. This approach to choosing λ is similar in spirit to the more practical and physical philosophy underlying the optimal averaging methods reviewed in §1.8, and is invariably used with these methods to set the regularization levels. However, the more mathematical spirit of the RLS inversion methods lends itself to a more mathematical and abstract approach. Throughout this section the discretized problem (1.53), and the regularized formulation (1.57) will be the principle objects of study, although many of the conclusions drawn can also be applied to the continuous problem (1.32).

Imagine that a solution, $\mathbf{f}_\lambda$, to (1.53) has been obtained from the minimization of (1.57) (with some appropriate smoothing functional), and that the variance, σ_i, of the errors, ϵ_i, on each data point, g_i, is known. It is hoped, of course, that $\mathbf{f}_\lambda$ is a good approximation to the true solution, $\mathbf{f}_0$ (so that $\mathbf{g} = H\mathbf{f}_0 + \epsilon$), for which the following obviously holds:

$$\chi^2(\mathbf{f}_0) = \sum_{i=1}^{m} \left(\frac{g_i - (H\mathbf{f}_0)_i}{\sigma_i} \right)^2 = \sum_{i=1}^{m} \left(\frac{\epsilon_i}{\sigma_i} \right)^2 \approx m \tag{1.60}$$

(where the last step is an obvious result of the statistical properties of the data errors). Since we are looking for $\mathbf{f}_\lambda \approx \mathbf{f}_0$ we expect (1.60) to hold with $\mathbf{f}_\lambda$ in place of $\mathbf{f}_0$ when $\mathbf{f}_\lambda$ is a good solution, i.e. when we have chosen a good value for λ. In short, we expect a good estimate of the solution to leave residuals $\mathbf{g} - H\mathbf{f}_\lambda$ that look, statistically, like the expected data errors. Thus, we could choose λ to be the solution of

$$\chi^2(\mathbf{f}_\lambda) = m. \tag{1.61}$$

This method was suggested by Phillips (1962), but it has been shown to give a rather poor choice for λ, one that results in considerable oversmoothing (λ too large), as was pointed out by Turchin *et al.* (1971). The reasons for this are fairly easy to see. For any value of λ the estimated solution will try to some extent to fit the data. When λ is very small it will fit the data as well as it can, and when λ is larger it will not fit the data so well. This means, in particular, that different realizations of the data errors will give rise to different solutions, each 'following' the data errors to some extent. In other words, for finite λ the recovered solution will tend to fit the data errors: the recovered solution will lie away from the true solution $\mathbf{f}_0$ in the direction that it is pulled by the particular realization of the data errors. If we imagine that we know the value of λ that makes the recovered solution closest to the true solution, then the fact that this solution fits the data errors means that we expect the residuals to be slightly smaller than they would be otherwise. When λ is very small (zero) the solution will try to get as close to the data as possible. This means that all of the n parameters in the solution vector will be used up in fitting the data errors, leaving only $m - n$ degrees of freedom in the residuals. Essentially this is saying that a component of the error vector lies in the image space of the solution space under the mapping represented by H, and the recovered solution fits this part perfectly, which leaves only the component orthogonal to the image space left in the residuals. This component will obviously have a smaller norm than the whole error vector. In fact, since n of the degrees of freedom in the errors are swallowed up by the solution,we expect the residuals to satisfy

$$\chi^2(\mathbf{f}_\lambda) = m - n$$

for λ very small. If we assume that some quadratic smoothing functional, such as the pth order smoothing functionals in (1.58), has been chosen to regularize the problem, this argument can be extended to non-zero λ, to show that, in general, we expect the residuals to satisfy

$$\chi^2(\mathbf{f}_\lambda) = m - \mathrm{Tr}\left\{H(H^TH + \lambda C^{(p)})^{-1}H^T\right\}. \tag{1.62}$$

$\mathrm{Tr}\{A\}$ denotes the trace of the matrix A (i.e. the sum of its diagonal elements). Compare the matrix appearing in the trace above with the right hand side of (1.59). The quantity

$$T(\lambda) \overset{\mathrm{def}}{\equiv} \mathrm{Tr}\left\{I_m - H(H^TH + \lambda C^{(p)})^{-1}H^T\right\} \tag{1.63}$$

appearing above is called the *equivalent degrees of freedom for error* (see Wahba 1983 and Hall and Titterington 1987).

The preceding argument suggests that seeking the λ that satisfies (1.62) ought to give a considerable improvement over the Phillips method in (1.61), and this is indeed the case. This method for choosing λ is known as the EDF method, for obvious reasons.

There is one further technique for choosing the smoothing parameter in RLS inversions, which has many advantages over EDF, not least of which is the fact that it does not require an estimate of the data errors (recall that the χ^2 functional in (1.62) contains the values of the data error variances). The idea is to use the solution obtained from *part* of the data set to predict the remaining data values. Comparison of the actual values with the predicted values for different λ provides a method for choosing the smoothing parameter. In more detail, let $\mathbf{f}_\lambda^{(k)}$ be the solution obtained from all the data *except* the kth data point (so that the kth row of the kernel matrix is also deleted), for any λ. Use this to make a prediction, $g_\lambda^{(k)}$ of the value of the kth data point. Repeat this for $k = 1, \ldots, m$ and form the quantity

$$G(\lambda) = \sum_{k=1}^{m} (g_k - g_\lambda^{(k)})^2. \tag{1.64}$$

Then choose the smoothing parameter simply by finding the value of λ that minimizes G. This is very simple in principle, but actually performing m separate inversions for each value of λ used in the minimization of G would be horrendously time-consuming. It is fortunate, then, that a minor modification of the expression for G (a modification that can be justified for reasons other than computational efficiency) results in a function that is much easier to calculate for any value of λ. The details of this derivation are given in Golub *et al.* (1979). The result is a function

$$GCV(\lambda) = \frac{R(\lambda)}{T(\lambda)^2}, \tag{1.65}$$

where $T(\lambda)$ is given in equation (1.63), and $R(\lambda)$ is the 'residual sum of squares' function

$$R(\lambda) = ||\mathbf{g} - H\mathbf{f}_\lambda||^2. \tag{1.66}$$

Essentially, $R(\lambda)$ is the χ^2 functional without the dependence on the data error variances. If the relative sizes of the error variances are known then these should be used in $R(\lambda)$, thus making it identical (up to a constant, at least) with χ^2. Using the expression (1.59) for the solution with pth order smoothing, (1.66) may be written

$$R(\lambda) = ||\mathbf{g} - H(H^T H + \lambda C^{(p)})^{-1} H^T \mathbf{g}||^2$$

or

$$R(\lambda) = \left\| \left[I_m - H(H^T H + \lambda C^{(p)})^{-1} H^T \right] \mathbf{g} \right\|^2. \tag{1.67}$$

Compare the matrix occurring on the right hand side of this equation with that appearing in the definition (1.63) of $T(\lambda)$.

In section 2.4 the importance of being able to calculate $GCV(\lambda)$ and the functions needed for choosing the smoothing parameter by the EDF method quickly is stressed, and appendix A presents expressions for these quantities that make their evaluation very rapid.

This concludes the review of inverse methods and their occurrence in helioseismology.

Chapter 2

An Algorithm for Solving Linear Inverse Problems

2.1 Introduction

In this chapter, a complete algorithm for solving linear inverse problems like those occurring in helioseismology is presented. This algorithm unifies the various regularized least squares (RLS) methods described in §1.9, and generalizes them, at the same time including other methods such as *maximum entropy* (see Narayan and Nityananda 1986, and references therein). It takes the ideas upon which the function expansion and piecewise constant discretization methods are based and combines them, providing greater freedom for discretizing the integral constraints occurring in equation (2.5), and, equally importantly, enabling smoothing to be applied consistently when different discretizations are used. The resulting algorithm permits discretizations which have some of the characteristics of piecewise constant discretization (PCD), described in §1.9, in that they consist of a set of discretization bins, but which also have a set of basis functions in terms of which the solution is expanded, just like the function expansion methods also reviewed in the previous chapter. The discretization is fixed by first specifying a set of *discretization points*, $0 = X_0 < X_1 < \ldots < X_N = R_\odot$ (for some N), and then choosing, for each *discretization bin*, $(X_{i-1}, X_i] \equiv \{r; X_{i-1} < r \leq X_i\}$ (the reason for the use of intervals open at the left end and closed at the right is given in §2.2 – it is essentially a useful convention), a set of n_i *basis functions*, $\phi^i_1(r), \ldots, \phi^i_{n_i}(r)$. Obviously, PCD is obtained by choosing some set of discretization points and requiring that for every bin, $n_i = 1$ and $\phi^i_1(r) = 1$ for $X_{i-1} < r \leq X_i$. Orthogonal function expansion using a cosine series (for example) is obtained by fixing $N = 1$, so that $X_0 = 0$, and $X_1 = R_\odot$, and then putting $\phi_j^1(r) = \cos[(j-1)\pi r/R_\odot]$. This will become clearer in §2.2, but it is useful for the subsequent discussion in this section to have these definitions.

Several ideas motivate the derivation of this algorithm. Perhaps the simplest is the observation that, while it is common practice to discretize the the integral constraints (1.32) and then to regularize the solution of the resulting matrix equation (1.46), (1.48), or (1.51), (often in a rather *ad hoc* manner), it is more natural and satisfying to regularize the integral problem itself, using a regularizing *functional* (assigning a measure of smoothness to every function in the space of possible solutions) and then to discretize both the integral constraints and the regularizing functional according to the same discretization scheme. This would mean that the solution obtained with *any* discretization method would (for a given smoothing functional) be the approximate solution of the same (functional) equation. The results should, therefore, be largely independent of the discretization method chosen (although see §1.9.5), depending only on the smoothing functional chosen. This is important because it is always vital to know the effect of regularization on the features in the recovered solution, and to be able to compare recoveries made using different discretizations. When the effects of regularization depend explicitly on the choice of discretization (by virtue of the *ad hoc* choice of smoothing constraint in the regularization of the discretized problem) this is very difficult and requires detailed investigation in general.

Furthermore, the four different RLS methods in §1.9 derive from very different philosophies. As a result, each has its advantages and disadvantages, both from a purely mathematical or technical point of view, and with regard to the peculiarities of individual problems (for example, some methods may be more suited to bringing out the physically significant features in the solar rotation profile than others – this is an idea that will be considered in more detail later). On the technical side, the different approaches to the problem give rise to numerical procedures which, while they all result in minimizing a function of a finite number of variables (which represent the solution) involving the data and some smoothing constraint, are actually quite distinct. Oddly enough, this distinction is brought out most clearly by emphasizing that piecewise constant discretization can be thought of as a particular kind of orthogonal function expansion (OFE), described in §1.9 – the basis functions for the expansion are just the set of 'top-hat' functions that take the value one on a bin, and zero on the other bins:

$$\phi_j(r) = \begin{cases} 1 & \text{if } r \text{ lies in the } j\text{th discretization bin} \\ 0 & \text{otherwise,} \end{cases}$$

for $j = 1, \ldots, n$. As no two of these functions are non-zero in the same region they are obviously orthogonal. The possible solutions are then

$$f(r) = \sum_{j=1}^{n} f_j \phi_j(r), \tag{2.1}$$

for any (real) values of the parameters f_j (*cf.* (1.44), (1.47), and (1.50)). The discretization of the kernels proceeds, just as for OFE, by evaluating, for each kernel and each basis function, the quantity

$$H_{ij} \overset{\text{def}}{\equiv} \int_0^{R_\odot} k_i(r)\phi_j(r)\, dr \tag{2.2}$$

(*cf.* equations (1.45), (1.49), and (1.52)). There is no problem with this. The difficulty comes when regularization is applied to the discretized problems $\mathbf{g} = H\mathbf{f}$ (equations (1.46), (1.48), and (1.51)). It is very common to use a form of regularization that requires one of the derivatives (the pth derivative, say) of the estimated solution to be small (again, see §1.9). This is known as pth order smoothing. If the basis functions used in the discretization are suitably differentiable, then the calculation of the required derivative of the estimated solution ought to be a simple matter (and is obtained by differentiating each of the basis functions appearing in (2.1) individually). However, the basis functions in PCD are not even continuous, let alone differentiable, and the estimated solution will be similarly discontinuous (at each discretization point). It is, then, not possible to apply the naive definition of differentiation in pth order smoothing with PCD. But, in fact, it is precisely with PCD that derivative smoothing is most commonly used (other discretization methods, such as OFE and spectral expansion, tend to use zeroth order smoothing – where the values of the function itself are encouraged to be small – because this is usually easier to implement in those cases). The resolution to this apparent conundrum is quite simple. PCD uses the parameters f_j, which are the (constant) values that the solution takes on each bin, in a finite difference scheme. Typically, it is assumed that f_j is the value of some underlying function at the mid-point of the jth bin. The derivatives are then given, effectively, by operating on these parameters with some pth order difference matrix, $B^{(p)}$:

$$f_i^{(p)} = B_{ij}^{(p)} f_j.$$

$f_j^{(p)}$ is then the estimate of the pth derivative of the estimated solution at the mid-point of the jth bin. This is a totally different approach – one which is actually very simple and very useful in many circumstances – and it is not easy to see a way to think of the two derivatives as different aspects of some underlying procedure. But it is possible to create a discretization algorithm that takes advantage of both 'definitions' of differentiation, and, as a result, allows PCD and function expansion methods to be derived as special cases of this general procedure. This is the essence of the work in §2.2. The benefits of such an algorithm are really twofold. Firstly, the ability to employ such a wide variety of different discretizations from the same algorithm, and *with the same regularization functional*, permits easy comparison of their relative merits and demerits for particular problems. Clearly, when the discretizations are implemented with separate algorithms, and particularly when the regularization applied is not directly comparable because its meaning for the final solution depends on the form of the discretization chosen, it is very difficult to say whether the perceived advantages of some discretization are really due to that discretization, or whether they are a result of different regularizations or, perhaps, indifferent implementations of the other methods. Secondly, having a single algorithm requiring a well-defined and consistent set of parameters to be specified by the user potentially makes a much wider spectrum of inversion methods available to the non-expert inverter.

Having considered the more technical aspects of the different discretization methods, it is now appropriate to consider the relevance of the specific details of particular problems to the choice of discretization method. It has already been stated that one of the advantages of the algorithm presented in this chapter is that, with the regularization being applied first and the discretization occurring later, the quality of the inversion ought to be largely independent of the discretization used (provided, of course, that the discretization is adequate to recover the information that is actually contained in the data). This means that, from a purely mathematical point of view, as far as the effect of data errors on the recovered solution function is concerned, the choice of discretization is not vital. However, real inversions are usually performed to obtain information about some object or physical process. Each such object or process will have its own characteristics, and there will be some features of the recovered solution that are perhaps more important than others from a physical point of view. To understand this, it is perhaps easiest to consider a purely hypothetical, although not at all implausible, example from helioseismology. In the solar rotation problem (see §1.5.3) the variation of the solar internal angular velocity with radius and latitude within the sun is found from the frequency differences between certain oscillation modes of the sun. The accepted reason for this variation in the rotation rate is that the turbulent convective motions within the solar convection zone carry angular momentum with them, and therefore succeed in redistributing the solar angular momentum in latitude and radius (see Durney 1991, and references therein). Without this transport process, viscous dissipation ought to have made the internal angular velocity almost uniform, which is not what is observed, even at the surface, where the surface latitudinal differential rotation indicates an equatorward transport of angular momentum. The importance of convection in this process suggests that there should be significant features in the rotation profile near the boundaries of the convection zone. In fact, the upper boundary of the convection zone essentially coincides with the surface of the sun, and so only the lower boundary is relevant here. Such features are indeed observed (as the inversion in chapter 5 show), but the width of this region is very narrow and is right on the limit of resolution of the data available at present. Now, for simplicity, concentrate on the variation of the rotation rate with depth, ignoring the latitudinal variation, so that the inverse problem is one dimensional and the solution function, $f(r)$, is just the equatorial value of the angular velocity at radius r within the sun. Imagine that some theory of turbulent convection (possibly along the lines of that described in Durney, 1991) predicts that the variation of the equatorial rotation rate with depth near the lower boundary of the convection zone can be modelled in some fairly simple way, perhaps being described by an expression involving a few free parameters. Then, discretizations that can quite accurately reproduce the expected forms the rotation rate may take are likely to give a better recovery than those which cannot. In particular, with an appropriate discretization it may be possible to obtain estimates of the important parameters in the model for the rotation profile, and thus to learn more about convection itself. Even if the model is not actually very good, it is likely that the features it predicts will broadly reflect reality, and so using a discretization that is more 'tuned' to the model ought still to be useful. In general, the specific physical details of individual problems should influence the selection of the discretization. Having an algorithm that provides a much greater breadth of choice of discretization would make such a 'tuning' of the inversion method much simpler. While this is not, in itself, a final clinching proof of the validity and usefulness of the algorithm presented in this chapter, it does suggest that the development of the algorithm may be worth pursuing.

A further aspect of the procedure that is worth considering is related to the discussion in the previous paragraph regarding the physical relevance of recovered solutions. It will be seen as the exposition of the algorithm unfolds in the following section that the combination of the ideas behind PCD and OFE permits the choice of discretizations that can be thought of as consisting of several different discretization schemes (PCD, OFE or polynomial expansion) in different regions 'glued together'. That is, in one region of the solar interior, the solution for the solar rotation rate could be discretized according to a PCD scheme, say, whereas in the rest of the interior it could be expanded in terms of sines and cosines or some other orthogonal functions, or perhaps in terms of polynomials. There are two reasons why this is potentially helpful. Firstly, the solar interior is quite a big place, and there are a number of different physical processes that are important for determining conditions there. These processes effect different parts of the sun to different extents. For example, deep within the sun the important processes are the nuclear generation of energy, and the radiative transport of this energy out towards the surface, whereas in the outer third of the solar interior turbulent convection dominates the heat transport and determines the properties of these layers (see Kippenhahn and Weigert 1990, and Christensen-Dalsgaard 1992). These are very different phenomena, and it is more than reasonable to assume that they affect the solar rotation rate in very different ways. By the arguments of the preceding paragraph, then, we might expect that the best recovery of the solution would be given by using very different discretizations in the radiative interior and the convective envelope. Secondly, and in helioseismology this is potentially a very important point, the observed solar oscillation modes are *much* more sensitive to features in the solar rotation rate (and the sound speed) near the surface than near the centre of the sun. This gives rise to what might be termed an 'information gradient' throughout the solar interior, with plenty of information about conditions in the convective envelope being contained in the oscillation data, but very little information about the deep interior. There is no point in attempting to recover the solar rotation rate at hundreds of points within the radiative interior, because the information (resolution, if you like) is just not contained in the data currently available. If the inversion to find the rotation profile in the inner half of the solar interior could, in some sense, be completely separated from the rest of the inversion, and performed independently, it might be deemed appropriate to perform only a very simple low order polynomial fit to the rotation rate, perhaps recovering as few as three polynomial coefficients, without using regularization, which introduces bias into the determination of the polynomial coefficients (but see the discussion in §1.9.5, where the relationship between discretization and explicit regularization is considered). The idea is that when the data is very poor, as it often is in astronomical problems, it is impossible to make reliable, absolute inferences from the data without additional knowledge. It is often possible, though, to rule out some models of the process under consideration, or to obtain crude estimates of important parameters in those models, by best-fitting the models to the real data. Of course, this goes completely against the spirit of inverse theory, where the goal is to make model independent statements about reality, but in some circumstances it may be the best that can be hoped for. It may happen, too, that only knowledge of very simple aspects of the solution (such as its first few moments) is necessary to provide useful bounds on some related physical phenomenon. In the solar case, it would be interesting to see whether the available data can be used to give some indication of the rate at which the angular velocity increases towards the centre of the sun (if, indeed, this is the case), or even whether the rate at which the angular velocity increases is itself changing (such knowledge would provide information about the likely rotation rate in the core, which is related to the perihelion shift of mercury, and the associated tests of general relativity – see Moffat (1983), Campbell *et al.* (1983). These would correspond to the first and second moments of the solution – the linear and quadratic coefficients in a polynomial expansion. Such a limited solution might not be very satisfactory, but it might also be the best that can be achieved. Of course, in the outer regions of the sun, where the information content of the data is high, we would still like to perform a full inversion (with PCD or whatever). The algorithm presented in this chapter makes such a hybrid inversion possible. To effect such a solution, though, requires the specification of a regularizing functional that does not impose any restriction on the solution function in the region near the centre of the sun where polynomial fitting is to be applied. This is the only point at which consideration of the discretization to be used might come before, and influence, the choice of regularization: to perform the low order polynomial fit successfully without bias it is essential to avoid applying regularization to that region. This is a minor point, but worth noting.

There is one final reason for believing that the algorithm presented in this chapter may be worthwhile. Much of this thesis deals with ways to optimize the procedure for obtaining the solution of helioseismic (or any other) inverse problems. Chapter 3, for example, looks at optimal choices of smoothing parameter (see also §1.9.6), and is is possible to envisage explicitly performing an optimization over some of the possible discretizations that could be used in the solution of (2.5) (Barrett 1994), in order to determine the best option for any inversion. In order to examine the optimization of the solutions of inverse problems like (2.5) it is necessary to say exactly what the degrees of freedom for optimizing the inversion algorithm are. Any inversion procedure contains parameters (*meta*-parameters, if you like) which, while often thought of as fixed for the purposes of performing the inversion, may be varied to give inversions with different resolution and bias. For example, in PCD the discretization points are chosen first and the inversion is then performed, but different discretization points will give rise to solutions of different quality. Similarly, the optimal averaging methods reviewed in §1.8 rely on having some definition of 'peakyness' (*cf.* the functionals (1.41) and (1.43)), and different definitions will again give inversions with different properties. Optimization of the inversion then means finding the 'best' set of values of these meta-parameters. That is, the optimization begins with the specification of what is meant by a 'good' inversion method, and this is then turned into a function of the meta-parameters such that good methods (presumably, those with the best resolution or error-magnification) are assigned small values, while bad methods are given large values. This function is then minimized over the set of meta-parameters to find the best inversion method. It ought to be clear from this that it is only really possible to create such a function for inversion schemes that can be parametrized continuously by some set of quantities (such as the discretization points in PCD), and therefore that the different RLS methods reviewed in §1.9 cannot all be included in the optimization (there is no parametrization that gives PCD for some parameters and OFE for others). However, he algorithm presented in this chapter contains all of the RLS methods described in §1.9 as special cases, and could, in principle at least, be used as a basis for performing such an optimization over the set of possible discretization schemes, for example. The point is really that having all the RLS methods accessible from within a single inversion algorithm allows their effectiveness to be compared directly. A similar argument could be made with regard to the general form of the optimal averaging algorithm presented in §1.8, but this will not be considered here because this thesis will deal almost exclusively with RLS methods.

In the following section, the algorithm for reducing the archetypal helioseismology inverse problem, (2.5), to a form easily soluble numerically is motivated and expounded. At the end of §2.2 the felicity of quadratic smoothing constraints is pointed out, the general form of the pth order smoothing matrix is derived, and the solution to the minimization of the resulting regularized, discretized functional is found. The rest of this chapter then deals primarily with aspects of the numerical calculation of this quadratic regularized solution. The generality of this approach encourages the development of a procedure for constraining the solution found, either by fixing its value or the value of its derivatives at specific points, or, more particularly, by imposing continuity constraints across the boundaries between neighbouring discretization bins. Section 2.3 describes such a procedure, with particular regard to its implementation with quadratic regularization. Two alternative forms for the solution are given and their relative merits are discussed. In §2.4 the need for a very fast and very robust method for computing the matrix inverse $(H^T H + \lambda C)^{-1}$ (occurring in the solution, (2.82), to the quadratic regularized problem) is emphasized, and an algorithm capable of achieving this – the *Fix-Heiberger algorithm* – is described. Finally, §2.5 summarizes the important results in this chapter.

2.2 The Unified Algorithm

In order that the explanation of this algorithm be as complete as possible, it is advantageous to begin with the standard form (2.5) for linear inverse problems like those occurring in helioseismology and work from there. The difference between this procedure and the description of the RLS methods given in §1.9 is that, whereas there discretization was the first step, here discretization is deferred for as long as possible, and the problem is formulated in terms of the solution *functions.* It is only once the problem has been appropriately reformulated, and then regularized, that it is discretized in order to effect numerical solution. The method is applied to quadratic regularization, and the calculation of the solution is then discussed. The algorithm is broken down into five parts corresponding to these steps: reformulation, regularization, discretization, specialization to quadratic RLS, and solution. Some of the points made and ideas used in this section are closely related to aspects of the RLS methods of §1.9, but the slant here is rather different.

2.2.1 Reformulation

As was discussed in the previous chapter, the inverse problems common in helioseismology involve solving a set of equations of the form

$$g_i = \mathcal{K}_i[f] \equiv \int k_i(x) f(x)\, dx, \text{ for } i = 1, \ldots, m. \tag{2.3}$$

where m is the number of data points, and $\int dx$ may represent an integral over the volume of the sun (e.g. $\int_0^{2\pi} \int_0^{\pi} \int_0^{R_\odot} r^2 \sin\theta\, drd\theta d\phi$) or just an integral over the radius of the sun ($\int_0^{R_\odot} dr$). In fact, as Ritzwoller and Lavely (1991) show, the full solar rotation problem (which is the problem that will be dealt with principally with in this thesis) can be broken down into a sequence of 1-D inversions (involving integrals over radius) for different components of the variation of the rotation profile with latitude. In this chapter, therefore, only the 1-D case will be addressed directly, but the method can be generalized easily. Note that, in this case, the *functional* $\mathcal{K}_i$ becomes

$$\mathcal{K}_i[f] = \int_0^{R_\odot} k_i(r) f(r)\, dr, \tag{2.4}$$

which will be taken as the standard form for $\mathcal{K}_i$ in what follows. Throughout this chapter this (standard) square bracket notation for functionals (functions of functions) will be used. It is worth reiterating (see §1.6) that the functional $\mathcal{K}_i$ can be defined on the whole of the function space $L^2(0, R_\odot)$ of (Lebesgue) square-integrable functions (see chapter 5, and in particular §7.4, of the book by Weir, 1973). As a result, it is natural to adopt $L^2(0, R_\odot)$ as the space of possible solution functions. This choice is far from vital (although it would be difficult to think of a *larger* space to use that still allows $\mathcal{K}_i$ to be defined on all its members – the requirement of integrability of the product $k_i(r)f(r)$ inhibits this), and subspaces of $L^2(0, R_\odot)$ (such as the space of continuous functions) could be used, and will in many cases be more appropriate. In fact, as the discussion following (2.11) shows, it will be convenient later to choose the space of solutions to be such a subspace of $L^2(0, R_\odot)$. All these spaces contain far more freedom than can ever be constrained by the data, and so this is really not at all restrictive. The solution space is set out explicitly so that it is clear what kind of objects the functionals occurring in this section act on – the mathematical niceties involved in rigorously defining such functionals will not be considered, because this algorithm is designed for practical implementation, where strict mathematical precision is of secondary importance to convenience and pragmatism.

With the generic form (2.4) for the set of functionals $\mathcal{K}_i$, the system of equations (2.3) obviously becomes

$$g_i = \mathcal{K}_i[f] \equiv \int_0^{R_\odot} k_i(r) f(r)\, dr, \text{ for } i = 1, \ldots, m. \tag{2.5}$$

Strictly speaking, (2.5) is not a completely honest expression of the problem, because in real situations the data will be contaminated by errors, so that the data acquired from observations will actually be given by

$$g_i = \mathcal{K}_i[f] + \epsilon_i \equiv \int_0^{R_\odot} k_i(r) f(r)\, dr + \epsilon_i, \text{ for } i = 1, \ldots, m. \tag{2.6}$$

where ϵ_i is the error on the ith data point (which will be assumed to be gaussian here, because this is the situation that usually arises in helioseismology). Of course, the actual values of the ϵ_i are completely unknown, so that we are forced to pretend that the problem we have to solve is (2.5).

There is no possibility of solving (2.5) analytically (see §1.9), so it is useful to seek to put (2.5) in a form amenable to numerical solution. Numerical methods for solving systems of equations fall, roughly speaking, into two categories. There are the direct methods, which effect solution via a well defined procedure that takes the 'knowns' (data) and operates on them in a specific way to give the unknowns (the solution). Then there are the *iterative* methods that begin, invariably, with some guess to the solution and repeatedly correct and improve it, eventually converging to the solution. A truly direct method for solving (2.5) does not seem to exist, so we will seek to take advantage of the second approach, and to recast the problem in a form that enables to solution to be obtained by an iterative method: a minimization procedure, in fact. It will transpire, though, that in many important cases (when quadratic regularization is being used) the minimization has an analytic solution (that is, an expression, (2.82), can be derived in which the solution is given as an explicit function of the data), and it is only necessary to evaluate this solution for the given data – a direct method has resulted.

For the moment, the presence of errors on the data will be overlooked, and a formulation of the problem equivalent to (2.5) will be sought, in which it is assumed that the desired solution is one that reproduces the data precisely (that is, a function, f, such that $\mathcal{K}_i[f] = g_i$ for all i).

Equation (2.5) is not a particularly convenient formulation of the problem from the point of view of developing an iterative procedure. Iterative methods are generally based on some criterion for distinguishing between good solutions and bad ones, or, equivalently, for determining which of the possible corrections that could be made at each iteration are actually improvements on the previous solution. Obviously, 'good' solutions are characterized by the fact that they provide a good fit to the actual data: by giving rise to values of $\mathcal{K}_i[f]$ that are very close to the actual data, g_i. The better the solution is, the nearer $\mathcal{K}_i[f]$ will be to g_i. What we need, then, is some measure of 'distance' in the space $\mathbb{R}^m$ of possible data vectors $\mathbf{g} \equiv (g_1, \ldots, g_m)$. A very simple definition of nearness for the vector space $\mathbb{R}^m$ uses the usual euclidean norm, which is the sum of the squares of the components of a vector (see equation (5.17) of Craig and Brown 1986). It turns out that this is actually the best definition of nearness to use when the data errors are gaussian, because the minimizer of the sum of the squared differences between the data, g_i, and the 'predicted' data, $\mathcal{K}_i[f]$, is a maximum likelihood estimate of the solution (see §15.1 of *Numerical Recipes*, and §1.9.5). There is one slight subtlety here, though. When the data errors are not uniform (that is, when the error variances on different data points are different) the maximum likelihood estimator is not given by the naive sum of squares, but by a weighted sum of squares, in which each term is weighted by (the reciprocal of) the error variance on that data point (note the $\hat{\sigma}_i$ denominator in equation (2.7) below). Intuitively, it is easy to see why this weighted norm must be better. If the variance of the errors on a particular data point, g_k, is very small, this means that we know its value very accurately, so for a solution f to be acceptable it must have a value of $\mathcal{K}_k[f]$ correspondingly close to g_k. In contrast, if the variance of the errors on some other data point, g_l, is large we won't mind so much if the solution function does not give such a good fit to g_l. In other words, the relevant measure of distance between $\mathcal{K}_i[f]$ and g_i, for any i, is not the ***absolute*** distance but the distance ***relative*** to the expected size of the errors on that data point. (All this assumes that the errors on each of the data points are uncorrelated. If this is not the case then the appropriate definition of the χ^2-functional involves the covariance matrix of the errors. Here it will always be assumed that the errors are independent.)

With this new definition of distance on the space of possible data vectors, we can replace the problem of 'solving' (2.5) by that of minimizing the single functional

$$\chi^2[f] = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{g_i - \mathcal{K}_i[f]}{\hat{\sigma}_i}\right)^2 \tag{2.7}$$

over the space of allowable solutions, $L^2(0, R_\odot)$. Here $\hat{\sigma}_i$ is an estimate of the standard deviation of the errors on g_i. If no such estimate is available it is common practice to assume that the errors are uniform (i.e. $\hat{\sigma}_i = \hat{\sigma}$, for some $\hat{\sigma}$ and all i) and then remove the common factor from (2.7). When necessary, $\hat{\sigma}$ can be given some judiciously chosen value. The validity of replacing (2.5) by (2.7) was discussed in §1.9.5 with regard to the discretized problems considered there, but the points made are equally valid in the continuous case. The $1/m$ factor is included for convenience: essentially, it ensures that the value of χ^2 does not increase as the number of data points used increases (i.e. it makes χ^2 largely independent of m).

Clearly, the only requirement for $\chi^2[f]$ to be defined is that $\mathcal{K}_i[f]$ is defined. The discussion following equation (2.4) is therefore equally applicable to χ^2, and we can still take the solution space to be (some subspace of) $L^2(0, R_\odot)$.

2.2.2 Regularization

The discussion in §1.9.5 described the intrinsic ill-posedness (instability to data errors and lack of uniqueness in the solution) common to problems like (2.5). The original form of the inverse problem, (2.5), and its reformulation as the minimization of (2.7), are completely equivalent, in the sense that any solution of one will also be a solution of the other. As a result, solutions derived by minimizing (2.7) share all of the instability and non-uniqueness inherent in (2.5). The inevitable presence of instability and data errors in helioseismic inversions invalidates the minimization of (2.7) as a method for solving (2.5).

It was not necessary to consider the problem of non-uniqueness in §1.9 because the solution of the matrix problem that resulted from discretization was essentially unique – the discretization was used to impose uniqueness. Here, non-uniqueness occurs, but it is an easy problem to remove. Any two exact solutions of (2.5) are equivalent, and neither will be more valid as far as the available data is concerned. We might as well just pick one of the possible solutions, according to some prescription. (2.5) has been reformulated as a minimization problem, so the easiest way to implement this selection procedure is to look for the solution that satisfies the data *and* has the least 'somethingness'. That is, introduce some definition, $\Phi[f]$, of somethingness for every function, f, in the space of possible solutions (an easy one to choose is largeness: $\Phi[f] = \int_0^{R_\odot} [f(r)]^2 \, dr$, for example), such that solutions indistinguishable in terms of their χ^2-values will have different values of Φ. Then, minimizing Φ subject to the constraint $\chi^2[f] = 0$ (so that the data is satisfied) will select the desired solution, and thus remove the non-uniqueness. Mathematically, this can be achieved using the method of lagrange multipliers: find the minimum of the functional

$$\mathcal{F}_\tau[f] \overset{\text{def}}{\equiv} \Phi[f] + \tau\,\chi^2[f] \tag{2.8}$$

over all f and real numbers τ (obviously, τ is the lagrange multiplier).

The introduction of Φ and $\mathcal{F}_\tau$ removes the non-uniqueness in the solution of (2.5), but has little effect on the instability of the problem: the solution obtained by minimizing $\mathcal{F}_\tau$ will still be highly sensitive to small changes in the data. (Recall that non-uniqueness can be thought of as the limiting case of instability. When the solution is unstable to perturbations in that data, this means that very different solution functions will give rise to *almost* identical data in (2.5). Non-uniqueness occurs when different solution functions give rise to exactly identical data. Implicit in the formulation (2.8) are the constraints $g_i = \mathcal{K}_i[f]$, so that $\mathcal{F}_\tau$ is minimized over the solutions that exactly satisfy the data. This says nothing about the relationship between solutions with slightly different data, and, in general, these solutions will be very different.) Formulating the problem as the minimization of $\mathcal{F}_\tau$ in (2.8) is an excellent move, though, and with minor modification can stabilize the inverse problem, as well as removing non-uniqueness. Moreover, attempting to alleviate the instability using a functional like $\mathcal{F}_\tau$ will suggest a general form for the functional Φ, which, at the moment, is largely arbitrary.

The first point to recognize is that the inevitable presence of instability and (unknown) data errors (the ϵ_i in equation (2.6)) makes it unnecessary (and undesirable) to find a function f satisfying (2.5) exactly, as this is almost guaranteed to produce a solution radically different from the real solution. In other words, it is inadvisable to enforce the constraint $\chi^2[f] = 0$ in (2.8). Ideally, we would like instead to find the 'true' solution, $\hat{f}$, which, by definition, satisfies $g_i - \mathcal{K}_i[\hat{f}] = \epsilon_i$ for each i, and which therefore also satisfies $\chi^2[\hat{f}] = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{\epsilon_i}{\hat{\sigma}_i}\right)^2 \sim 1$, since the ϵ_i have gaussian probability distributions with mean zero and variance $\hat{\sigma}_i^2$ (assuming that the $\hat{\sigma}_i$ are good estimates of the noise levels). If we are not imposing the strict condition $\chi^2[f] = 0$, there is no need to find the value of τ as part of the minimization procedure (i.e. there is no need to apply the condition $\frac{d\mathcal{F}_\tau}{d\tau} = 0$). Instead, we can treat τ as fixed, as far as the minimization is concerned (so that it becomes a kind of hyperparameter). A solution, f_τ, can then be found for any *fixed* τ by minimizing $\mathcal{F}_\tau$ over the space of solution functions. As the parameter τ has dropped out of the minimization process it is possible remove a factor τ from $\mathcal{F}_\tau$ and so to rewrite (2.8) in the convenient and more familiar form

$$\mathcal{R}[f] = \chi^2[f] + \lambda\Phi[f] \tag{2.9}$$

(*cf.* equation (1.57)), where $\lambda \equiv 1/\tau$ is the *smoothing parameter*, which is varied to adjust the extent to which the solution is forced to have a small Φ-value, rather than fitting the data accurately (see §1.9). Correspondingly, the solutions to the minimization of $\mathcal{R}$ for each value of λ will now be called f_λ.

It is easy to see from (2.9) the effect that changing λ will have on f_λ. For λ very small, the dominant term in $\mathcal{R}$ will be $\chi^2[f]$, the new functional Φ is irrelevant, and minimization will search for f's that make $\chi^2[f]$ as small as possible – the solution f_λ will, for small λ, satisfy $\chi^2[f_\lambda] = 0$, and so it will also satisfy (2.5). Such a solution is obviously undesirable, because it was to avoid exactly this unstable solution that Φ was introduced. Note, though, that for small but non-zero λ minimization will still be slightly affected by Φ, so the solution will attempt to find the f with $\chi^2[f] = 0$ that minimizes $\Phi[f]$, resulting in a unique solution as before. For very large λ, on the other hand, only the value of $\Phi[f]$ is important, the data become irrelevant, and f_λ is just the solution with the smallest Φ, regardless of its appropriateness for fitting the data. Again, this solution will be unacceptable. Somewhere between these two extremes lie values of λ that give acceptable solutions, f_λ, and the choice of an appropriate value of λ is a vital part of the inversion procedure (see chapter 3 and §1.9.6).

For reasonable choices of Φ, and of the smooothing parameter λ, the solution (that is, the minimizer) of (2.9) will be unique and will vary quite slowly as the data is perturbed: the introduction of Φ has *stabilized* (2.7). Of course, the introduction of the function Φ has also introduced a preference for certain solutions over other possible solutions, and it is highly unlikely that the most desirable solution according to Φ (which is selected by the inverter with little or no consideration of the data) will correspond precisely with the true solution to the problem. Φ will therefore tend to pull the solution to the problem away from the true solution towards the (almost arbitrary) preferred solution: *bias* has been introduced into the problem. This is the price to be paid for removing the instability of the problem to errors (see §1.9.5 and the second paragraph of Golub *et al.* 1979).

The introduction of the stabilizing functional Φ has, so far, overlooked one important point. Permitting solutions with $\chi^2[f] \neq 0$ is necessary, but is, in itself, insufficient to ensure that the solution obtained by minimizing (2.9) is even remotely adequate. For the procedure described here to be effective it is necessary to be more specific about the form and content of Φ. As we only have knowledge of the distribution of the errors, ϵ_i, and not their actual values, we are looking for solutions that have $\chi^2[f] \sim 1$. The problem is that for some inappropriate choices of the (as yet, largely arbitrary) functional Φ the solution obtained from (2.9) could have $\chi^2 \sim 1$, but be the same as an exact solution to (2.5) belonging to data $g_i + \epsilon_i \equiv \mathcal{K}_i[\hat{f}] + 2\epsilon_i$, whereas what is required is a solution for data $g_i - \epsilon_i \equiv \mathcal{K}_i[\hat{f}]$ ($\hat{f}$ being the true solution). In other words, if Φ is not chosen carefully we could end up finding solutions that correspond to data with *twice* as much noise, instead of no noise at all. There is *no way* to distinguish between these possibilities purely on the basis of the available data. What is needed is a way to distinguish between solutions that, in some sense come from 'error-free' data, and solutions that come from very 'noisy' data. This can then be used be to construct a functional Φ that assigns large values to the noisy solutions and small values to the error-free solutions, so that the minimization of (2.9) results in a strong preference for the error-free solutions.

Experience, and the discussion in §1.9, shows that the solutions we would like to see emerging at the end of the inversion procedure are characterized by being 'smooth', i.e. not varying very quickly. We can either look at this as an assumption about the nature of physical processes in the real world (we do not expect the real solar rotatation rate to vary in a very rapid or highly oscillatory manner), or we can think of it as an acknowledgement of the fact that limited resolution means that the most that can be achieved is to recover a solution that represents local averages (*cf.* Backus-Gilbert optimal averages, §1.8) of the real solution, and is consequently a smoothed version of the real solution. This latter explanation is perhaps more appealing. It says that giving a preference to smooth solutions just means demanding that the value of the recovered solution at any point is an average of the values of the true solution at nearby points, which means that the values of the *recovered* solution at nearby points are not independent. As a result, the infinite degrees of freedom in the solution function are no longer all independent – the solution function actually only contains the finite number of 'pieces' of information represented by the data.

There is one further alternative way to look at Φ, which is closely related to the ideas motivating the introduction of regularization in §1.9. The errors (the ϵ_i) and the 'true' data (the $g_i - \epsilon_i$) have very different statistical properties. It could be expected from this that the solutions that would be obtained if the data were actually just the ϵ_i or just the $g_i - \epsilon_i$ also have very different statistical properties, so it should be possible to introduce some kind of filter that will filter out those solution functions that appear, from their statistical properties to contain a large component 'corresponding to' data errors, just as was done in §1.9 for the discretized (matrix) problems. For various reasons (see §1.9) the data errors give highly oscillatory solutions, in general (that is, solutions containing large contributions from components with short wavelength oscillations). We would therefore like to avoid such rough, oscillatory solutions, which means that the filter we seek must permit 'low-frequency' (long-wavelength) information to pass through with little distortion, but must strongly damp out high-frequency components. Again the result is a preference for smooth solutions.

How do we characterize smooth solutions in a quantitative, mathematical way? Well, smooth solutions vary slowly, so, over a small region (from r to $r + \delta r$, say) $f(r)$ hardly differs from $f(r + \delta r)$ at all: the first derivative of f must be small (this corresponds to first order smoothing, see (2.10) below). Of course, if the solution varies slowly its value at any point will never be very different from its average value, $\bar{f}$, and so we expect $f(r) - \bar{f}$ to be small (for $\bar{f} = 0$ this corresponds exactly to zeroth order smoothing in (2.10), but the principle applies for any $\bar{f}$). There also exist functions which, while not varying much about their mean value and always having a small gradient, contain 'kinks' (places at which the gradient changes rapidly or discontinuously) giving rise to large or infinite values of the second derivative. Such functions would not really be considered to be smooth, and so looking for solutions with small second derivative everywhere is another way to impose smoothness (obviously, this corresponds to second order smoothing in (2.10)). This idea can, of course, be extended to arbitrary derivatives, and can be generalized by allowing combinations of these forms of regularization, and a host of other possibilities.

There are other approaches, such as maximum entropy, which looks more at the data errors than specifically at the solution. Although we know the expected typical values of the errors, and thus that we expect $\chi^2 = 1$, we do not know what their actual values are at all. There are any number of combinations of data errors that would give $\chi^2 = 1$, but we would be very surprised to find that all the errors have exactly the same value, for example, (we would generally expect some positive errors and some negative errors, and some spread in their magnitudes), just as we would be surprised to find that all the atoms moving randomly in an ideal gas contained in a box were all in one corner of the box. This is where the concept of entropy comes in. This will not be discussed in any greater detail here as this thesis will not deal with maximum entropy regularization, but see the review by Narayan and Nityananda (1986).

There are many ways of using these ideas as the basis for a definition of the functional Φ in (2.9). To see how to use them to synthesize a general regularizing functional that includes all reasonable possibilities it will be helpful to look at the forms of regularization used with the discretized problems in §1.9, and to see how these can be extended to apply to the whole space of solution *functions*. In those methods discretization was performed first, and a smoothing constraint, $\Phi(\mathbf{f})$, was then defined on the set of solution *vectors*. The definition of an appropriate smoothing functional on the whole space of solution functions begins by noting the (often implicit) relationship, $f(\mathbf{f})$, between solution vectors and their corresponding solution functions (*cf.* equation (2.1), which gives this relationship for PCD). The regularizing functional is then found by extending the functional

$$\Phi[f(\mathbf{f})] = \Phi(\mathbf{f}), \text{ for every } \mathbf{f}$$

to the space of solution functions in some way. For example, maximum entropy regularization is often used with PCD, so the relationship between the solution function and solution vector is given in (2.1). The maximum entropy regularizing function, in its simplest form, is (see Narayan and Nityananda 1986)

$$\Phi_{ME}(\mathbf{f}) = \sum_{i=1}^{n} f_i \ln\left(\frac{f_i}{f_{max}}\right), \text{ where } f_{max} = \max_{1\le i\le n}\{f_i\}.$$

Actually, since the formalism here is based on minimization rather than maximization, this is the negative of the usual entropy function. (Recall that use of maximum entropy is restricted to problems where the solution is known *a priori* to be positive. The logarithm is then well defined, the normalization factor in the logarithm assures that the log is negative, and so the functional is negative definite.) The most natural way to extend this to functions is to use

$$\Phi_{ME}[f] = \int_0^{R_\odot} f(r) \ln\left(\frac{f(r)}{f_{max}}\right) dr, \text{ with } f_{max} = \max_{0\le r\le R_\odot}\{f(r)\}.$$

On the other hand, if pth order smoothing is the regularization to be used

$$\Phi_p(\mathbf{f}) = \mathbf{f}^T C^{(p)} \mathbf{f},$$

where $C^{(p)}$ is the pth order smoothing matrix (see §1.9). This smoothing in the discretized problem is intended to mimic the effect of constraining the pth derivative of the solution function, so it is obvious that the required smoothing functional is

$$\Phi_p[f] = \int_0^{R_\odot} \left(\frac{d^p f}{dr^p}\right)^2 dr. \tag{2.10}$$

It is often the case that regularization of an inverse problem is performed using just one of the basic forms of smoothing just mentioned. However, this is not the most general case, and many other possible 'definitions' of smoothness could be found that might have advantages in particular problems. For example, weighting the smoothing constraint across the range of the inversion (0 to $R_\odot$, in this case) to force greater smoothness in some regions than in others might often be a good idea (see chapter 4), as might using combinations of the basic methods (a mixture of zeroth order and first order smoothing, say). Such constraints can easily be defined for the discretized solutions in §1.9, and the above examples indicate the features typical of many forms of regularization, so we can use them as a basis for translating any such (discrete) smoothing constraint in to a regularizing functional Φ. To permit maximum freedom in the choice of regularization constraint a general form for the functional should be found that includes all reasonable possibilities. Probably the most general functional that is likely to be useful in practical situations can be written

$$\Phi[f] = \int_0^{R_\odot} \varphi\left(f, \frac{df}{dr}, \ldots, \frac{d^p f}{dr^p}; r\right) \rho(r) dr, \tag{2.11}$$

where $\rho(r)$ is just a (non-negative) weighting factor included for later convenience, and φ is a suitably well-behaved (analytic, say) function of f and its first p derivatives. φ must also be bounded below to ensure that the smoothing functional does not effectively give 'infinite preference' to some solution (which would guarantee that minimization of the functional $\mathcal{R}$ in (2.9) always gave that solution irrespective of that data). It is often convenient to shift the function by adding a constant, so that this lower bound is zero, but we do not always do this. *Henceforth, the general regularizing functional* (2.11) *will be the only forms of regularization that will be considered.* The functional (2.11) includes all the well known regularizations such as maximum entropy and quadratic smoothing constraints, and many more besides, so it is unlikely to be necessary to consider more general forms of regularization.

There is one important point concerning the definition (2.11) that should be mentioned.

The general regularizing functional given in (2.11) contains a dependence on the derivatives of the solution function. This means that the functional cannot formally be defined on the whole of the space $L^2(0, R_\odot)$, which contains functions with all sorts of discontinuities and singularities. When any such function is operated on by Φ, the derivatives would not be defined. Putting this less formally, the derivatives would give rise to infinities, which would, ultimately, make $\Phi[f]$ infinite. In a certain sense this is alright, because we can think of these functions as much 'rougher' than functions that *can* be differentiated p times, so assigning them infinite roughness is fairly natural. Nevertheless, from a mathematical point of view, the fact that Φ is not defined on the whole of the space $L^2(0, R_\odot)$ is a problem that requires further consideration. We will not worry too much about the formal aspects of this problem, but will instead adopt a pragmatic approach.

There is a simple, 'common-sense' resolution to this difficulty. In the analytic, non-discretized formulation it is easily avoided, because the finite resolution in the data invariably prevents discontinuities in the real solution from being resolved, so that the estimated solution can be assumed to be as smooth (i.e. as differentiable) as necessary for the acceptable definition of the regularizing functional (see the discussion in §1.6), without obstructing a satisfactory fit to the data. Mathematically, the solution can be chosen to lie in the subspace $C^p(0, R_\odot)$ of $L^2(0, R_\odot)$ comprising the p-times continuously differentiable functions on $[0, R_\odot]$. In fact, we could go further than this and assume that the estimated solution lies in the space $C^\omega(0, R_\odot)$ of *analytic* functions – which is contained in $C^p(0, R_\odot)$ for every p – because $C^\omega(0, R_\odot)$ is *dense* in $L^2(0, R_\odot)$. See §7.5 of Weir (1973) for definitions and explanations of these terms. This means that any function in $L^2(0, R_\odot)$ can be approximated arbitrarily closely by some analytic function – certainly within the limits imposed by the finite resolution and data errors. Choosing the solution space to be $C^p(0, R_\odot)$, say, will ensure that the functional Φ, and hence $\mathcal{R}$, is defined on the whole of this solution space, and the formal difficulty disappears. In discretizing the functional (see the next step in the algorithm), a finite-dimensional subset of the (infinite-dimensional) solution space is chosen (according to some prescription), and the minimization of (2.9) is performed over this subset. Thus, the estimated solutions obtained after discretization will also be in $C^p(0, R_\odot)$, and so will be p-times continuously differentiable. This is one solution to the problem of interpreting and solving (2.11) and (2.9).

However, this interpretation would rule out many very simple and commonly used discretizations such as PCD, in which the estimates of the solution are discontinuous at each discretization point (see §1.9 or the beginning of the introduction to this chapter). Despite the fact that the recovered solution can be assumed to be arbitrarily differentiable, and even that the true solution will, in real situations, be smooth (the real solar rotation curve may contain some apparently sharp features, but these will not really be infinitely sharp – see §1.6), it is often convenient to introduce discretizations containing discontinuous or non-smooth basis functions, because of the simplicity of their implementations (and PCD is a prime example of this). This does not appear to be possible when the solution space is taken to be $C^p(0, R_\odot)$. This practical inconvenience will be addressed in the next part of the algorithm.

The question now is: how can the functional (2.11) be used in practice to find a meaningful solution to (2.5)?

2.2.3 Discretization

The space of allowable solutions, whether it be $L^2(0, R_\odot)$ or any of the spaces $C^p(0, R_\odot)$ or $C^\omega(0, R_\odot)$, is an infinite-dimensional vector space. This means that the specification of a general $f(r)$ requires an infinite number of parameters (the coefficients in the expansion with respect to some basis of the solution space): there are an infinite number of degrees of freedom in the problem. Clearly, no numerical method for minimizing $\mathcal{R}$ can cope with this and some approximation must be found. What is needed is an approximation to (2.9) that contains only a finite number of degrees of freedom and results in a minimization over these degrees of freedom (the numerical solution of optimization problems being rather well understood). We want to choose some subset, $\mathcal{S}$, of the full solution space that can be parametrized continuously by (say) n real free parameters, $f_1, \ldots, f_n$, so that any values of these parameters correspond to some solution function in $\mathcal{S}$. It is convenient to write these components in vector notation as $\mathbf{f}$, so that $\mathbf{f}$ is in $\mathbb{R}^n$. From this parametrization the discretization space, $\mathcal{S}$, can be defined as the set of all functions $f(\mathbf{f})$ such that $\mathbf{f}$ is in $\mathbb{R}^n$:

$$\mathcal{S} \overset{\text{def}}{\equiv} \{f(\mathbf{f}); \mathbf{f} \in \mathbb{R}^n\}.$$

Using this in (2.9) we can write, with a slight abuse of notation,

$$\mathcal{R}(\mathbf{f}) \equiv \mathcal{R}[f(\mathbf{f})] \tag{2.12}$$

(with similar definitions of $\chi^2(\mathbf{f})$ and $\Phi(\mathbf{f})$). Solving (2.9) then amounts to finding the parameter values, $\mathbf{f}$, that minimize $\mathcal{R}$ and using them to reconstruct the solution function $f(\mathbf{f})$. The process of defining the finite-dimensional function space, $\mathcal{S}$, and deriving explicit expressions for $\chi^2(\mathbf{f})$ and $\Phi(\mathbf{f})$ in terms of the solution vector $f(\mathbf{f})$ is called *discretization.*

The fact that only m pieces of information (data) are available means that at most m degrees of freedom in the solution can be determined, and the presence of data noise effectively reduces the number of pieces of information available still further. (This fact, combined with the infinite-dimensionality of the solution space, is the reason for the lack of uniqueness in, and hence ill-posedness of, this inverse problem.) This means that restricting the solution to lie in a finite-dimensional subset of the full solution space is simply a natural way to reflect the limited information content of the data, and is not just a crass approximation – provided the subset chosen is 'reasonable' and allows the information that *is* contained in the data to be extracted.

It is now time to be more particular about the types of discretization that are to be considered here, since it is really the specification of the acceptable forms of discretization that lies at the heart of this algorithm. Many factors will influence the choice of discretization space, $\mathcal{S}$, including formal mathematical considerations, the need for computational efficiency and the possible physical significance of the recovered solutions (see the introduction to this chapter). Some $\mathcal{S}$ will clearly be more appropriate for use in the practical solution of (2.5) than others, and ultimately, of course, the goal is to find the discretization space that is best suited to any particular problem. As yet, very little restriction has been placed on the discretization space. In general, $\mathcal{S}$ need not be a vector subspace of the solution space, and the relationship, $f(\mathbf{f})$, between solution functions and solution vectors, can be a largely arbitrary non-linear function of the parameters $f_1, \ldots, f_n$. However, an argument will now be presented to show that the best discretization space to choose in general will be a *linear* subspace of the solution space (so that $\mathbf{f}$ is linear in the parameters $f_1, \ldots, f_n$), obtained via the discretization procedure shortly to be defined. It will be helpful for the discussion that follows to observe that a linear subspace of the full solution space is also a vector space, and therefore has a basis of functions $\{\phi_1(r), \ldots, \phi_n(r)\}$, so that any function in the discretization space can be written

$$f(\mathbf{f};r) = \sum_{j=1}^{n} f_j \phi_j(r) \tag{2.13}$$

(*cf.* equation (2.1)).

To perform the minimization of $\mathcal{R}$ it is necessary to be able to evaluate $\mathcal{R}(\mathbf{f})$ for any value of $\mathbf{f}$, which means evaluating $\chi^2(\mathbf{f})$ and $\Phi(\mathbf{f})$. The simpler these calculations are, and the faster they can be performed numerically, the easier and quicker finding the solution will be. Restricting the discretization subset $\mathcal{S}$ to be a linear subspace of the solution vector space will almost always have significant advantages in this respect. The important parts of the calculations of $\chi^2(\mathbf{f})$ and $\Phi(\mathbf{f})$ are the evaluations of the integrals appearing in these functionals (see equations (2.7) and (2.11)), and, usually, employing a linear discretization subspace avoids the need to explicitly evaluate the integrals numerically for each $\mathbf{f}$. It should be fairly clear from the fact that discretizing χ^2 is essentially equivalent to discretizing the *linear* functionals $\mathcal{K}_i$ (see the definition (2.7) of χ^2) that a linear discretization scheme will make the calculation of $\chi^2(\mathbf{f})$ very simple (see the equations (2.30), (2.26), and (2.27), below). It is less obvious, though, that linear discretization schemes will very often be preferable for the discretization of the regularizing functional Φ. To see that this is indeed the case, consider an example in which the function φ occurring in the definition (2.11) of $\Phi[f]$ is a sixth order polynomial in f,

$$\varphi = \varphi(f;r) = \sum_{s=0}^{6} \alpha_s(r) f^s.$$

Expanding f in terms of the n basis functions ϕ_j (equation (2.13)), and using this in (2.11), results in an expression for $\Phi[f]$ which is a sixth order *multinomial* in the components f_j (that is, for each f_j, $\Phi[f]$ is a sixth order polynomial in f_j), in which the multinomial coefficients are integrals of products of up to six of the basis functions ϕ_j. For instance, the coefficient of the term in $f_{j_1} \dots f_{j_s}$ is (proportional to)

$$\int_0^{R_\odot} \alpha_s(r) \phi_{j_1} \dots \phi_{j_s} \rho(r)\, dr.$$

These integrals need only to be evaluated (analytically or numerically) once for all of the terms in the multinomial expression for $\Phi[f]$, and the computation of $\Phi(\mathbf{f})$ for any $\mathbf{f}$ will reduce to the evaluation of a multinomial with fixed (pre-calculated) coefficients. This will obviously be much faster than having to perform all the integrations numerically for each solution vector, $\mathbf{f}$. Obviously, there will be some regularizing functionals (when $\varphi(f;r)$ involves logarithms of f, for example) where the general expression for $\Phi(\mathbf{f})$ cannot be written so that the f_j only appear *outside* the integrals, thus requiring the integrals to be evaluated for each $\mathbf{f}$ even *with* a linear discretization space. There will also be functionals for which analytic expressions for the integrals involved can be obtained even with some *non*-linear discretizations. Usually, though, using a linear discretization space will result in an increase in the speed with which $\Phi(\mathbf{f})$ can be calculated.

Increasing the speed with which the solution of the inverse problem can be calculated numerically is important, but there are certain (commonly used and simple) forms of regularization for which the advantages of using a linear discretization space are even more significant. For these smoothing constraints the minimization can be performed 'analytically', giving an explicit expression for the solution in terms of the data and the smoothing constraint (*cf.* equation (2.82)). Solution of the inverse problem (2.5) then proceeds by direct calculation, and there is no need to invoke the arcane rituals of numerical optimization. Apart from the increase in speed that results from this, there are the advantages that solution is guaranteed (a solution *will* be found, whereas general optimization procedures can fail to converge to an acceptable solution when the initial guess to the solution that such methods usually require is poor), and that this solution is unique (i.e. the solution of the numerical minimization procedure under consideration is unique – this is a completely separate issue from uniqueness or non-uniqueness in the underlying inverse problem (2.5)), in contrast to the situation that prevails when general optimization is used and the function $\mathcal{R}(\mathbf{f})$ has multiple minima. Not only this, but the availability of an explicit expression for the solution in terms of the data allows the details of the solution, such as its physical relevance and the efficiency of the method to obtain it, to be studied in much greater detail, without recourse to large numerical simulations.

These magical regularizing functionals are characterized by the property that they depend quadratically on the solution function: the function φ in (2.11) is a quadratic polynomial in f or its derivatives (the pth order smoothing functional in (2.10) is the prime example of this). They are therefore called *quadratic* regularizing functionals, and it has already been mentioned in the introduction to this chapter that they will be the principle object of study here and throughout the rest of this thesis. Mathematically it is easy to see why quadratic regularization is so efficacious. The χ^2 functional, being essentially the square of the linear functionals $\mathcal{K}_i$ in (2.5), depends quadratically on the solution function, f. When the regularizing functional, Φ, in (2.9) is quadratic, the functional $\mathcal{R}$ will also be quadratic. Minimization of this functional will then give rise to a solution in which the data and the solution are linearly related – finding the minimum of $\mathcal{R}$ amounts to taking the (variational) derivative of $\mathcal{R}$ with respect to f and setting the result to zero, which (since $\mathcal{R}$ is quadratic in f) will result in a functional in which f occurs only linearly. Linearity is such a pleasing property, and can give rise to such huge simplifications, that it would be advantageous to look for discretizations that preserve this linearity, so that the n solution parameters are also linearly related to the (m) data values. A linear relationship between the finite number of data points and the finite number of solution parameters is, inevitably, a matrix equation, and computers are ideally suited to solving these. The obvious, if not the only reasonable, way to satisfy the requirement of linearity is to restrict $\mathcal{S}$ to be some finite-dimensional linear subspace of the solution space.

Having decided that linear discretization is the way to go, it is necessary to consider *which* linear subspaces of the infinite-dimensional solution space should be used. Discretizing the functional $\mathcal{R}$ can be broken down into two parts: discretizing χ^2 and discretizing Φ. Inspection of the definition of χ^2 in (2.7) shows that discretizing this functional amounts precisely to discretizing the linear functionals $\mathcal{K}_i$. With linear discretization this is almost trivial. Using (2.13) and the linearity of $\mathcal{K}_i$ gives

$$\mathcal{K}_i[f] = \sum_{j=1}^{n} f_j \mathcal{K}_i[\phi_j] \equiv \sum_{j=1}^{n} f_j \int_0^{R_\odot} k_i(r)\phi_j(r)\,dr. \tag{2.14}$$

It is usual then to define the *kernel matrix* just as in equation (2.2) (*cf.* equations (1.45), (1.49) and (1.52) of §1.9, and equation (2.27) below), so that (2.5) reduces to the simple matrix equation $\mathbf{g} = H\mathbf{f}$. $\mathcal{K}_i$ is defined on the whole of $L^2(0, R_\odot)$, so *any* set of basis functions in $L^2(0, R_\odot)$ could be used in the discretization of χ^2.

However, the discussion following the definition of the general regularizing functional in (2.11) showed that, as a result of the appearance of derivatives of the solution function in (2.11), Φ is only formally defined on the subspace $C^p(0, R_\odot)$ of $L^2(0, R_\odot)$ containing the p-times continuously differentiable functions on $[0, R_\odot]$. Strictly speaking, then, the discretization space, $\mathcal{S}$, should be a subspace of this space, so that all the functions in $\mathcal{S}$ are also p-times continuously differentiable. But this rules out the use of PCD as a discretization, for example, because there the solution functions are almost all discontinuous at each discretization point (see the introduction to this chapter). Many perfectly acceptable, and quite appealing, discretizations will, therefore, be incompatible with the use of the regularizing functional (2.11), unless some appropriate alternative definition of the 'derivative' of discontinuous functions, such as occur in PCD, is given. The solution to this problem for PCD was quite simple: define a notion of differentiation based on finite differences (see §2.1). With the usual definition of differentiation, we now have two viable definitions of 'the derivative' of functions, and the goal is to find a way to make use of both of them to create an algorithm for which PCD, OFE, spectral expansion, polynomial expansion and maximum entropy are merely special cases. Using finite differences is perhaps less satisfying from a formal point of view, but it is practically much more powerful, having the considerable advantage that it permits much more general forms of discretization.

The functional Φ in (2.11) *cannot* be defined on functions that are not sufficiently differentiable, so we need to find a new functional which will constrain the solution function in the RLS minimization of (2.9) in a similar manner to (2.11), but which can be defined on discontinuous or non-differentiable functions, just as difference formulae are used in pth order smoothing to approximate derivatives when PCD is the chosen discretization. The idea here is to think of the operator $\frac{d^q}{dr^q}$ appearing in (2.11), not strictly as a real derivative, but as some more general operator whose effect is like differentiation, but which is defined on a much wider class of functions than the true derivative operator. Pre-empting the definition (2.43), introduce some linear operator, $\mathcal{D}$, and replace every occurrence of $\frac{d}{dr}$ in (2.11) with it, so that the regularizing functional becomes

$$\Phi[f] = \int_0^{R_\odot} \varphi(f, \mathcal{D}f, \ldots, \mathcal{D}^p f; r)\rho(r)dr. \tag{2.15}$$

$\mathcal{D}$ must satisfy certain, fairly obvious, requirements:

- it must be 'like' differentiation, so that the definition (2.15) is almost equivalent to (2.11),
- it must be more general than $\frac{d}{dr}$, that is, it must operate on a larger class of functions, including those used in discretizations such as PCD (otherwise there's not much point introducing it),
- it is essential that the numerical calculation of $\mathcal{D}^q f$ is feasible for any functions, f, that lie in the chosen discretization space (the discretizations that will be considered 'acceptable' will be outlined shortly).

We *could* think of $\mathcal{D}$ as a true generalization of the derivative operator to some (as yet unspecified) larger function space. Strictly, the definition of $\mathcal{D}$ should then be accompanied by a specification of the space of functions on which it operates. When $\mathcal{D}$ is defined for functions in particular discretization spaces below this question will be partly answered, but rather than giving a mathematically complete definition, the following intuitive approach is adequate here: $\mathcal{D}f$ is defined for a function, f, if an 'acceptable' discretization (i.e. of the form described below) can be found such that f is contained in the corresponding discretization space.

Alternatively, and preferably, $\mathcal{D}$ can be thought of just as an *approximation* to the real derivative operator. That is, we still consider the real solutions to lie in $C^p(0, R_\odot)$, but now the discretization space, $\mathcal{S}$, is not restricted to be a subspace of this solution space, and so can contain non-smooth functions. View a discontinuous or non-differentiable function, f, in $\mathcal{S}$, as a *numerically convenient* way of approximating a function, $\tilde{f}$ that *is* in $C^p(0, R_\odot)$. The operator $\mathcal{D}$ then operates on f to give a similarly convenient approximation to the derivatives of $\tilde{f}$. That is,

$$f(r) \approx \tilde{f}(r) \text{ and } \mathcal{D}^q f|_r \approx \left.\frac{d^q \tilde{f}}{dr^q}\right|_r \quad (q \leq p).$$

The same argument (lack of resolution *etc.*) that said we can take the solutions to be arbitrarily differentiable can now be used in reverse to say that, since f and $\tilde{f}$ are indistinguishable in terms of their validity as solutions of (2.5), there is no need to look for a function in $C^p(0, R_\odot)$ at all. The required solution might as well be a function f lying in $\mathcal{S}$.

Although the discretization spaces, $\mathcal{S}$, will be rather more general than just subspaces of $C^p(0, R_\odot)$, there will be some restrictions on them. They will have to be subspaces of $L^2(0, R_\odot)$ (in order for the discretization of χ^2 still to be valid), but this is hardly a restriction at all: it is certainly not a practical consideration. The limitation on the possible discretization spaces comes largely from the extent of the validity of the definition of the operator $\mathcal{D}$. It should be obvious from previous comments that the definition of this new operator involves the use of finite differences, as well as the usual derivative operator – the precise details will not be given here, because they will be studied fully very shortly, and because they are not vital to the understanding of (2.15). This combination of two 'derivatives' leads naturally to the definition of the useful discretization spaces to be given, and it is only with such spaces that the operator $\mathcal{D}$ can be defined. If other reasonable definitions of differentiation were available, these could be included in the definition of $\mathcal{D}$ to give an algorithm with even greater generality, allowing other discretization spaces to be used, but the two derivatives we have seem to be quite general enough. In the practical implementation of this algorithm the only functions, f, for which $\Phi[f]$ ever needs to be evaluated lie in the finite-dimensional discretization space. Thus, once it has been shown how to calculate $\mathcal{D}f$, and then $\Phi[f]$ for f in this discretization space, there is nothing to hinder the successful application of this procedure.

Whatever interpretation of the operator $\mathcal{D}$ is adopted, the most important point is that $\mathcal{D}f|_{r_0}$ reflects the rate at which f varies in the vicinity of r_0: when $\mathcal{D}f|_{r_0}$ is large we expect the value of $f(r)$ for r slightly less than r_0 to be very different from its value for r slightly greater than r_0. This is very similar to the meaning of $\frac{d}{dr}\big|_{r_0}$ for f in $C^p(0, R_\odot)$, except that $\frac{d}{dr}\big|_{r_0}$ is a purely local statement about the variation of f in an infinitesimal region about r_0. As a result of this, (2.15) symbolizes a constraint on the solution function f whose effect on the estimated solution, f (obtained from the minimization of (2.9)), is basically what would be expected if f were in $C^p(0, R_\odot)$ and the real derivative operator could be used. That is, if,when used in (2.9), Φ in (2.11) constrains the qth derivative (for example) of a function in $C^p(0, R_\odot)$ to be small at some point, then (2.15) would also tend to encourage the value of $\mathcal{D}^q f$ to be small at the same point.

The time has come to describe the details of the generalized discretization procedure. Henceforth, only linear discretizations will be considered. The preceding discussion has tended to give the impression that discretization is achieved by first choosing some subspace, $\mathcal{S}$, and then finding a basis for $\mathcal{S}$ in terms of which the solution functions can be expanded, as in (2.13). In fact, in practice what happens is that some set of (linearly independent) basis functions, $\{\phi_1, \ldots, \phi_n\}$, is selected, and the discretization space is defined to be the vector space spanned by these functions, that is, the space containing all functions that can be obtained from (2.13) for $\mathbf{f}$ in $\mathbb{R}^n$. The natural bases to choose are those which consist of very simple functions such as sines, cosines, exponentials, 'top hat' functions, etc., or have a particular relevance to the problem at hand (such as the set of kernel functions). The various RLS inversion methods described in §1.9 each take advantage of one particular type of basis: orthogonal function expansion uses some set of orthogonal functions (such as sines and cosines) defined over all of $[0, R_\odot]$, PCD uses a set of top hat functions, whereas spectral expansion takes the kernel functions as the basis for $\mathcal{S}$. The discretization to be described here unites these apparently distinct RLS procedures and generalizes them, resulting in a single algorithm that can implement any of the commonly used RLS inversion methods simply by (roughly speaking) specifying the particular basis functions for that method.

The easiest way to begin is with the definition of the acceptable discretizations. It will then be possible to examine how to define the concept of differentiation (i.e. the operator $\mathcal{D}$ in (2.15)) for all the functions in the discretization space corresponding to any such discretization. In the presentation of the procedure there will be one or two irritating technical diversions (such as the need to introduce half-open intervals in (2.17)), which are necessary for completeness and clarity, but which rather distract from the main line of the presentation. Such points should not be allowed to cloud the understanding of other more important aspects of the procedure.

The preceding discussion referred to the fact that this algorithm unifies the PCD and function expansion methods by using a form of discretization that combines the principal features of these methods: namely, the set of discretization points used in PCD, and the set of basis function that the function expansion methods rely on. The first step in the definition of such a discretization scheme is, therefore, the selection of a set of discretization points,

$$0 = X_0 < X_1 < \ldots < X_N = R_\odot, \tag{2.16}$$

spanning the range of the inversion. This set of points naturally partitions the interval from zero to $R_\odot$ into a set of N disjoint intervals or *bins*. A vital part of this algorithm is the definition of a set of basis functions on each such interval, and for this definition to be unambiguous and to have some formal validity it is necessary to specify the interval on which the function is defined. There are many possible choices of intervals based on the points X_i that partition the interval from 0 to $R_\odot$, and as these intervals will be referred to many times in what follows it will be advantageous to be quite specific about their definition:

$$I_i \overset{\text{def}}{\equiv} (X_{i-1}, X_i] \equiv \{x; X_{i-1} < x \leq X_i\}, \text{ for } i = 1, \ldots, N. \tag{2.17}$$

The chosen intervals I_i are open at the left end and closed at the right end (see Burkill 1962, p.13, for definitions of the terms open and closed) for two reasons. Firstly, the I_i

must partition the range of the inversion (that is, they must cover all the points from 0 to $R_\odot$ and they must not overlap) so neighbouring bins I_{i-1} and I_i cannot both contain the discretization point X_i. This obviously does not determine which ends of the interval should be open or closed, but it is convenient to choose intervals open at the left end for the following reason. *None* of the solar oscillation modes are sensitive to the solar structure at the very centre of the sun (at $r = 0$ the kernels, k_i, are all zero). It is possible, then, to ignore the point $r = 0$ in the inversion so that the range of the inversion is the (half-open) interval $(0, R_\odot]$. It is obvious that the easiest way to partition this interval is with intervals that are also open at the left end. This is purely a matter of convention and in other problems other choices may be more natural.

In any event, given such a set of partitioning intervals, I_i, based on the discretization points in (2.16) it is possible to choose a set of functions on each bin in terms of which the approximate solution can be expanded. So, for each interval, I_i, introduce some set of (n_i) linearly independent functions (such as sines, cosines, polynomials, etc.):

$$\phi^i_j \ \text{for } j = 1, \ldots, n_i, \tag{2.18}$$

so that if r is in I_i (i.e. if $X_{i-1} < r \leq X_i$), then $\phi^i_j(r)$ is defined for $j = 1, \ldots, n_i$. Later, the definition of the new 'derivative' operator, $\mathcal{D}$, will rely on the ability to differentiate (in the usual sense) each of the functions ϕ^i_j as many times as required by the regularizing functional Φ (p times, in other words – see (2.11)) within its corresponding discretization bin I_i. This should not be a problem because the ϕ^i_j will usually be simple analytic functions (sines, cosines,...), which can obviously be differentiated arbitrarily many times, but even when the ϕ^i_j are not such simple functions – such as in spectral expansion (see §1.9) where they are the (numerically calculated) kernel functions – we will assume that the required derivatives are suitably well-behaved, and that adequate estimates of them can be obtained. (If there is some reason to allow functions that are discontinuous or non-differentiable in some bin then just add another discretization point there.) In future, then, it will always be assumed that the ϕ^i_j can be differentiated as many times as necessary *within* the discretization bin on which they are defined.

Of course, the reason for introducing these functions is to use them in an expansion like (2.13) to give the set of possible solution functions (the space $\mathcal{S}$). It is clear, therefore, that (since we are dealing exclusively with linear discretizations) the value of any solution function is given, for r in I_i, by the expansion

$$f(r) = \sum_{j=1}^{n_i} f_j^i \, \phi_j^i(r), \text{ for } X_{i-1} < r \leq X_i. \tag{2.19}$$

The f_j^i are just numbers: they are the free parameters in the solution. The total number of free parameters in the solution (which is, of course, the dimension of the discretization space $\mathcal{S}$) is

$$n \overset{\text{def}}{\equiv} \sum_{i=1}^{N} n_i. \tag{2.20}$$

Note that the differentiability of the basis functions ϕ_j^i in the bin on which they are defined ensures that the solution functions are similarly differentiable within each bin (although not, of course, at the end-points of the bins).

There are another couple of minor points that need consideration here. Firstly, for simplicity it would be distinctly advantageous to have an expression for the solution function that was valid everywhere, rather than having a separate expression, (2.19), for each bin. At the moment this is not strictly possible because the ϕ_j^i are only defined on a single bin. It is a very simple matter, though, to extend the definition of these functions to the whole of $(0, R_\odot]$ by simply assigning them the value zero outside the interval on which they were initially defined. We then arrive at the desired expression for the solution functions in terms of the free parameters:

$$f(r) = \sum_{i=1}^{N} \sum_{j=1}^{n_i} f_j^i \, \phi_j^i(r), \text{ for } 0 < r \leq R_\odot. \tag{2.21}$$

Secondly, it would also be useful for notational convenience, and for consistency with the earlier discussions, to write the free parameters as the components of a vector, $\mathbf{f}$ (which, by virtue of (2.20), must be n-dimensional), so that we can refer to the solution function corresponding to a set of parameters as $f(\mathbf{f})$ or as $f(\mathbf{f}; r)$. The f_j^i are labelled by two integers, so it is not obvious how they should be ordered as the components of $\mathbf{f}$. Actually, any permutation of the parameters f_j^i will do, but it will be essential in what follows to have a specific definition of $\mathbf{f}$. The simplest and most obvious choice will be made here: namely that the parameters f_j^1 for the first bin will be taken first, then for the second bin, and so on. This sets up a correspondence between the components of $\mathbf{f}$ and the free parameters as follows:

$$\begin{aligned} \mathbf{f} &\equiv (f_1, \ldots, f_{n_1}, f_{n_1+1}, \ldots, f_{n_1+n_2}, \ldots, f_l, \ldots, f_n) \\ &= (f_1^1, \ldots, f_{n_1}^1, f_1^2, \ldots, f_{n_2}^2, \ldots, f_j^i, \ldots, f_{n_N}^N). \end{aligned} \tag{2.22}$$

It will often be convenient to use this correspondence to refer to both the free parameters *and* the basis functions ϕ_j^i by a single integer, f_l and ϕ_l, not least because use of this convention allows equation (2.21) to be written in the equivalent, and simpler, form

$$f(\mathbf{f};r) = \sum_{l=1}^{n} f_l\,\phi_l(r), \text{ for } 0 < r \leq R_\odot, \tag{2.23}$$

which exactly matches the form of equations (2.13) and (2.1), for example. It is possible to give the explicit relationship between the pair (i,j) and the single integer l, but this will not be needed, although the fact that such a relationship exists will sometimes be used to write

$$l = l(i,j) \text{ and } i = i(l),\ j = j(l). \tag{2.24}$$

The definitions (2.22), (2.23) and (2.24), while necessary, amount to little more than book-keeping – they are not the most important parts of the discretization procedure.

The discretization is defined, then, by specifying the discretization points, (2.16), and the set of basis functions, (2.18), on each bin, and then using (2.21), or, equivalently, (2.23), to obtain all the functions in the n-dimensional discretization space, $\mathcal{S}$ (see (2.20)). Having outlined the acceptable discretizations it is necessary to describe and define the procedure for actually evaluating the functional $\mathcal{R}$ in (2.9) for any function in the space $\mathcal{S}$, which, in turn, requires the discretization of both of the functionals χ^2 and Φ appearing in $\mathcal{R}$.

It is a simple matter to present an expression for χ^2 in terms of the solution vector $\mathbf{f}$. Discretizing χ^2 just means discretizing the linear functionals $\mathcal{K}_i$ (as can be seen from an inspection of (2.7)), and the derivation of the $\mathcal{K}_i(\mathbf{f})$ proceeds basically as in the examples given earlier (*cf.* equations (2.2) and (2.14) and the associated discussions). In fact, using (2.23) to expand f in (2.14) gives the answer we want. However, this expression does not make explicit use of the properties of the discretizations to be considered here, so we revert to the expansion (2.21). The basis functions, ϕ_j^k, are each non-zero only on their corresponding interval I_k, so the functionals $\mathcal{K}_i$ can be written

$$\begin{aligned}\mathcal{K}_i(\mathbf{f}) &\equiv \mathcal{K}_i[f(\mathbf{f})] = \sum_{k=1}^{N} \int_{X_{k-1}}^{X_k} k_i(r) \left\{ \sum_{j=1}^{n_k} f_j^k \phi_j^k(r) \right\} dr \\ &= \sum_{k=1}^{N} \sum_{j=1}^{n_k} f_j^k \int_{X_{k-1}}^{X_k} k_i(r)\phi_j^k(r)\,dr. \end{aligned} \tag{2.25}$$

The kernels are known, as are the ϕ_j^k, so the integrals on the right hand side of (2.25) can easily be calculated. The advantage of this expression for $\mathcal{K}_i(\mathbf{f})$ is that the interval where each basis function is non-zero is made explicit, so the integrals can be evaluated as written. Normally at this stage, the kernel matrix would be defined to be the matrix whose elements are the integrals in (2.25). Here, though, we modify this step slightly for notational convenience. The appearance of a double sum requires some interpretation. The relationship (2.24) between the integers pairs (k, j) and the single integer l can be used to reduce the double sum to a single sum:

$$\mathcal{K}_i(\mathbf{f}) = \sum_{l=1}^{n} f_l \int_{X_{k(l)-1}}^{X_{k(l)}} k_i(r)\phi_l(r)\,dr \tag{2.26}$$

(which is actually identical to the sum appearing in (2.14) except for the limits of integration – the fact that the ϕ_j^k are non-zero only on I_k ensures that both expressions give the same value). Rather than defining the elements of the kernel matrix, H_{il}, just to be the integrals appearing in this expression, it will simplify notation if the denominators $\hat{\sigma}_i$ in the definition of χ^2 are also included in the kernel matrix. We therefore make two definitions. First, we define the elments of the kernel matrix to be

$$H_{il} \overset{\text{def}}{\equiv} \frac{1}{\hat{\sigma}_i} \int_{X_{k(l)-1}}^{X_k(l)} k_i(r)\phi_l(r)\,dr, \tag{2.27}$$

which, when used in (2.26), obviously reduces the $\mathcal{K}_i/\hat{\sigma}_i$ term in χ^2 to

$$\frac{\mathcal{K}_i(\mathbf{f})}{\hat{\sigma}_i} = \sum_{l=1}^{n} H_{il} f_l. \tag{2.28}$$

Then, in keeping with (2.27), redefine the data vector by including the $\hat{\sigma}_i$ factors in that as well, so that

$$\mathbf{g} \overset{\text{def}}{\equiv} \left(\frac{g_1}{\hat{\sigma}_1}, \frac{g_2}{\hat{\sigma}_2}, \ldots, \frac{g_m}{\hat{\sigma}_m}\right). \tag{2.29}$$

The point of these definitions is to allow vector notation to be used in the expression for χ^2, reducing $\chi^2(\mathbf{f})$ to the concise form

$$\chi^2(\mathbf{f}) = ||\mathbf{g} - H\mathbf{f}||^2 \tag{2.30}$$

(the norm $||.||^2$ here is the usual euclidean sum of squares). The dependence of χ^2 on the standard deviations, $\hat{\sigma}_i$, of the data errors has been completely absorbed into the 'new' data vector $\mathbf{g}$ and kernel matrix H. This simplifies the notation considerably. In future these definitions will be implicitly adopted, so that when g_i occurs in an expression this will really mean $g_i/\hat{\sigma}_i$.

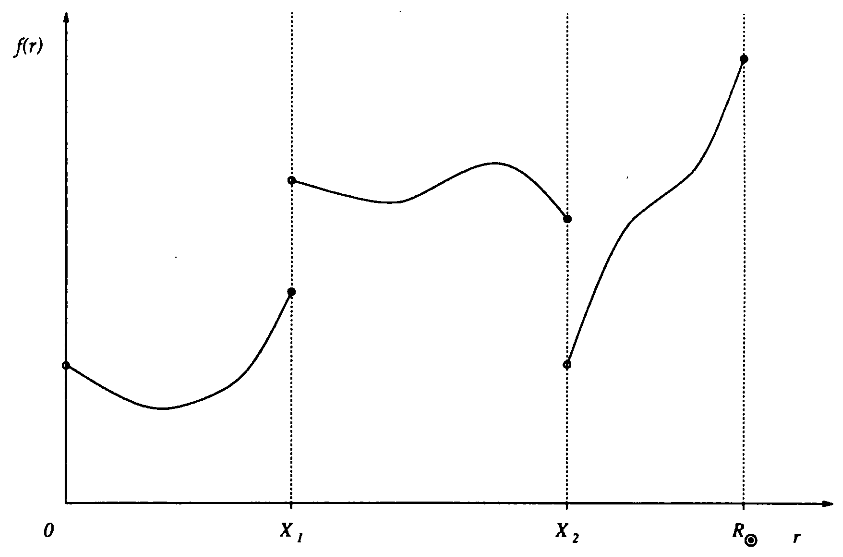


Figure 2.1: A schematic illustration of the type of solution functions occurring with the discretizations allowed in the algorithm presented here. The discretization has only three bins ($N = 3$). Filled circles indicate points which actually are part of the graph, and open circles points which are not (this allows the value of the function at discretization points to be seen more clearly).

For any given $\mathbf{f}$, all the quantities appearing in equation (2.30) are known (the data – and the error variances – in $\mathbf{g}$ are assumed to have been measured, and the kernel matrix has been calculated from the kernel functions using the discretization chosen), so evaluating $\chi^2(\mathbf{f})$ reduces to a simple arithmetic calculation. This now leaves only the problem of evaluating the regularizing functional Φ for any solution vector $\mathbf{f}$.

To guide the derivation of the expression for $\Phi(\mathbf{f})$ it is helpful to think more deeply about the form of the functions in the discretization space, $\mathcal{S}$, and to say what these functions look like. From an inspection of (2.21), and with the knowledge that the basis functions are non-zero only on a single bin, it is easy to see that the values of any solution function in $\mathcal{S}$ on two different bins are completely independent and each can be varied without changing the other. Figure 2.1 shows a typical solution function, based on a discretization using three bins. Note, in particular, that the value of f at any discretization point, X_i (which is given by the position of the filled circle at that X_i in fig. 2.1), is unconnected to the value of $f(r)$ for any $r > X_i$ – the solution function is (in general) discontinuous at each X_i. Note, too, that there is, of course, no relationship between the gradients – or the derivatives of any order – of the solution on either side of a discretization point. The general form for $\Phi[f]$ given in (2.11) involves derivatives of the solution function, but, since the discretization spaces will contain functions that have discontinuities at every discretization point, X_i, the usual definition of differentiation will not work in general, and cannot be used in the evaluation of $\Phi(\mathbf{f})$ without modification. It has already been said that the resolution to this problem is to steal the ideas used in PCD to calculate the derivatives of discontinuous functions. Those ideas by themselves are not enough, though, because it is possible to choose discretizations with just one bin, for example (when $N = 1$ the discretization would correspond to one of the function expansion methods reviewed in §1.9), in which the required derivative is just the usual derivative. We therefore need some formalism for combining these two forms of differentiation into a single procedure. Such a formalism essentially amounts to a definition of the operator $\mathcal{D}$ in (2.15).

Whatever alternative definition of $\mathcal{D}$ is finally chosen, there are three properties that we would definitely like it to have:

- It must operate on any function that, like the function in fig. 2.1, can be obtained from some acceptable discretization, to give a sensible measure of the rate of change of that function.
- It should be *recursive*. That is, it should be possible (in principle, at least) to apply the derivative operator $\mathcal{D}$ to a solution function any number of times, to obtain derivatives, $\mathcal{D}f, \mathcal{D}^2 f \equiv \mathcal{D}(\mathcal{D}f), \ldots$, of any order.
- As already mentioned, we seek a definition of $\mathcal{D}$ that uses both real differentiation and finite differences.

The first two points, taken together, force us to look for an operator $\mathcal{D}$ that sends solution functions to new functions that are like solution functions in that they are piecewise continuous and can have jump discontinuities at each discretization point. To see how to satisfy the third point, observe that it is possible to decompose any such function, f, into a continuous part, $c(r)$, and a 'step-function' part, $s(r)$ (that is, a piecewise constant function just like the solutions functions used in PCD), in the following manner.

1. Define the *jump* in the function f at the discretization point X_i to be

$$J_i = \lim_{r\to X_i+} f(r) - f(X_i), \text{ for } i = 1,\ldots,N-1 \tag{2.31}$$

(the limit in this expression arises from our convention about half-open discretization intervals – the notation $\lim_{r\to X_i+}$ means 'r tends to X_i from above', so that the limit is taken for values of r greater than X_i and therefore lying in the $(i+1)$th bin, I_{i+1}). Augment (2.31) with the convenient definition $J_0 = \lim_{r\to 0+} f(r)$.

2. Define the N-dimensional vector $\mathbf{s} = (s_1,\ldots,s_N)$ to be

$$s_i = \sum_{j=0}^{i-1} J_j, \text{ for } i = 1,\ldots,N. \tag{2.32}$$

This gives $s_1 = J_0 \equiv f(0)$, and makes s_i the cumulative jump in the function f.

3. Define the step function $s(r)$ to be the piecewise-constant function that has the constant value s_i throughout the ith bin, i.e.

$$s(r) = s_i \text{ for } X_{i-1} < r \le X_i, \text{ and for each } i. \tag{2.33}$$

Note that s has exactly the same jumps as f: $\lim_{r\to X_i+} s(r) - s(X_i) = s_{i+1} - s_i = J_i$, using (2.32).

4. Define the function $c(r)$ simply by

$$c(r) = f(r) - s(r) \text{ for } 0 < r \le R_\odot. \tag{2.34}$$

It should be quite clear from the fact that f and s have the same discontinuities that c is continuous everywhere, but here's the proof anyway: Certainly, c is continuous throughout each discretization bin, since f and s are. At the discretization point X_i we have

$$\begin{aligned}
\lim_{r\to X_i+} c(r) &= \lim_{r\to X_i+} f(r) - s_{i+1} = J_i + f(X_i) - \sum_{j=0}^{i} J_j \\
&= f(X_i) - \sum_{j=0}^{i-1} J_j = f(X_i) - s(X_i) \\
&= c(X_i),
\end{aligned}$$

using, at various points, (2.33), (2.31), (2.32), and (2.34).

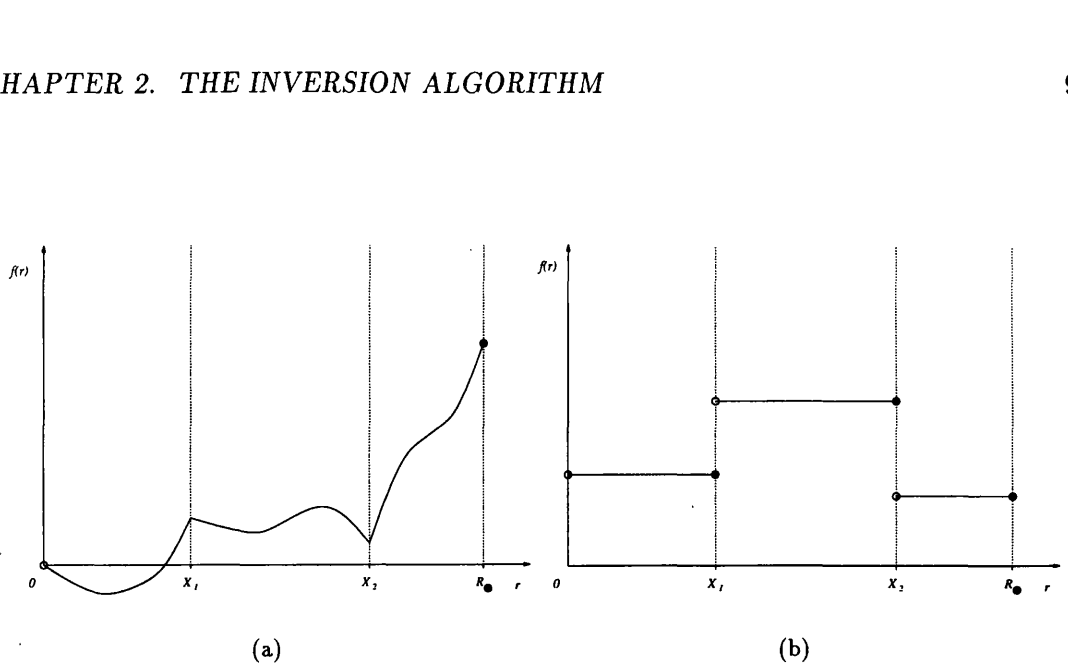


Figure 2.2: An example of the decomposition of a solution function into a continuous part and a step-function part. (a) shows the continuous function obtained from the solution in fig. 2.1 when the step function in (b) (which is constructed from the jumps in the solution function – see the text) is subtracted from it. In other words, adding (a) and (b) together will give the function in fig. 2.1. Filled and open circles have the same meaning as in fig. 2.1.

5. Obviously, equation (2.34) gives

$$f(r) = c(r) + s(r), \tag{2.35}$$

where c is continuous everywhere and s is a step function with jumps at each discretization point, as required.

Figure 2.2 shows just such a decomposition for the solution function in fig. 2.1. There are a few points worth noting about the function c. Firstly, since $s(0) = s_1 = f(0)$, $c(0)$ is always zero. Secondly, a nice way to visualize the construction of c from f is to think of sliding the part of f in the first bin down (or up) until it has the value zero at $r = 0$, then moving the part of f in the second bin up or down until it joins up with the solution on the first bin (so that the open and filled circles at X_i in figure 2.1(a) lie on top of one another), and then repeating this for every discretization bin. Finally, c is suitably differentiable (in the usual sense) *within* each bin, because both f and s are.

There is one very important aspect of the decomposition (2.35) that will be required for the definition of the operator $\mathcal{D}$: it is unique. This should be fairly clear from the 'graphical' construction of c just described, but to be certain, consider two decompositions $f(r) = c_1(r) + s_1(r)$ and $f(r) = c_2(r) + s_2(r)$. Then $c_1(r) - c_2(r) = s_1(r) - s_2(r)$, which means that $s_1 - s_2$ must be a continuous function (because $c_1 - c_2$ is). Obviously,

$s_1 - s_2$ is a step function, so it is constant except for simple jumps. Continuity ensures that $s_1 - s_2$ has no simple jumps, and so must be constant: $s_1(r) - s_2(r) = C$ everywhere. $c_1(0) = 0$ and $c_2(0) = 0$, so $c_1(0) - c_2(0) = C = 0$. Thus, $s_1(r) = s_2(r)$ and $c_1(r) = c_2(r)$ everywhere, i.e. $c_1 = c_2$ and $s_1 = s_2$.

The decomposition (2.35) is just what we need to be able to define an extended derivative operator. The continuity of c, along with its assured differentiability within each bin, means that it is possible (with some care at the discretization points) to take the derivative of c in the usual way, whereas experience with the difference methods used in PCD to calculate 'derivatives' of step functions means that we can also 'differentiate' s. We can then define the derivative of f to be just the sum of these two derivatives.

A little more precisely, we define an operator D acting on continuous and piecewise-differentiable functions like c that is just the *left-hand* derivative with respect to r. In other words

$$Dc|_r \overset{\text{def}}{\equiv} \lim_{h \to 0} \left\{ \frac{c(r) - c(r-h)}{h} \right\}. \tag{2.36}$$

This use of the left-hand derivative is necessary, given the convention about intervals closed at the right end, to allow the derivative to be properly defined at the discretization points. The definition (2.36) ensures that the value of Dc at X_i is 'attached to' the values of Dc on the ith bin. As a result, the new function Dc is, like f, piecewise-continuous with only simple jumps at the discretization points, which means that the operator $\mathcal{D}$ that will finally be defined in (2.43) can also be applied to Dc. This is the basis of the recursive definition of higher derivatives.

There are many ways to use finite difference schemes to define an operator $\mathbb{D}$ that acts on step functions like s to give some measure of the rate of change of s. Here we will think of the values, s_i, that the step function takes as sampled values of some underlying smooth function whose derivatives we wish to approximate. The difference scheme will then be used to operate on the s_i to give a set of numbers s_i' (representing the derivatives of the underlying function at the sampled points), which will be used to constuct another step function, $s'(r)$, just as we did in (2.33). To do this it is necessary to specify the values of r at which the samples were notionally taken. Of course, these points – denote them by r_i for $i = 1, \ldots, N$ – should lie within their corresponding bins, I_i, but apart from that they can be chosen freely. Usually the mid-points of the bins will be chosen by default, but other choices might sometimes be better, especially when a wide bin, I_i, abuts a narrow bin, I_{i+1} say, for then the estimated values of the derivative may be more appropriate when r_i is chosen to lie rather nearer to X_i (and hence to r_{i+1}) than if it were at the centre of its bin. In the implementation of this algorithm that was used for the inversions performed in this thesis a middle course was steered between these two extremes. A single parameter α, lying between 0 and 1, was introduced such that $\alpha = 0$ gives $r_i = X_{i-1}$, $\alpha = 1$ gives $r_i = X_i$, and, in general, $r_i \overset{\text{def}}{\equiv} X_{i-1} + \alpha(X_i - X_{i-1})$, for every bin. In particular, $\alpha = \frac{1}{2}$ puts the r_i at the mid-points, $\frac{1}{2}(X_{i-1} + X_i)$, of their bins. Henceforth, assume that some choice of the r_i has been made (mid-points, say).

Having fixed the sample points, r_i, the difference scheme still has to be chosen. The choice is between *forward*, *backward*, or *centred* differences, which in terms of the s_i and the r_i are given by

$$s'_i = \frac{s_{i+1} - s_i}{r_{i+1} - r_i} \tag{2.37}$$

$$s'_i = \frac{s_i - s_{i-1}}{r_i - r_{i-1}} \tag{2.38}$$

$$s'_i = \frac{1}{r_{i+1} - r_{i-1}} \left\{ (r_i - r_{i-1}) \frac{s_{i+1} - s_i}{r_{i+1} - r_i} + (r_{i+1} - r_i) \frac{s_i - s_{i-1}}{r_i - r_{i-1}} \right\} \tag{2.39}$$

(for $i = 1, \ldots, N$), respectively. Strictly speaking, these definitions should include separate definitions of the s'_i at one or both of the end-points, $i = 1$ and $i = N$, where the above definitions do not apply (because s_{N+1} does not exist in (2.37), for example). There are many options here too, but we will skate over this issue, because it is not really pivotal.

The important point is that whatever difference scheme is chosen, the relationship between the step function values, s_i, and the values of its derivative, s'_i, are related by a matrix equation:

$$s'_i = \sum_{j=1}^{N} \Delta_{ij} s_j, \quad \text{for } i = 1, \ldots, N. \tag{2.40}$$

As an example of this, consider forward differences, (2.37). For this scheme the difference matrix, Δ, is given by

$$\Delta_{ij} = \begin{cases} -1/(r_{i+1} - r_i) & \text{for } j = i \\ 1/(r_{i+1} - r_i) & \text{for } j = i + 1 \\ 0 & \text{otherwise} \end{cases}$$

(This definition is not valid for the last row, Δ_{Nj}, of Δ, for the reasons alluded to above. For this row we must make some alternative definition, such as setting $\Delta_{Nj} = \Delta_{N-1,j}$, which amounts to using *backward* differences to define the derivative, s'_N, at the last point.) Similar definitions can obviously be made for the other difference schemes.

Another point to note is that in all of the difference schemes (2.37), (2.38) and (2.39) the s_i only enter in the form of differences between neighbouring values, $s_{i+1} - s_i$, (as we would expect from an operation intended to mimic differentiation, where only the change in value between nearby points is important, not the absolute values at those points). The definition (2.32) tells us that $s_{i+1} - s_i = J_i$, which means that the difference formulae can be simplified if they are written in terms of the J_i rather than the s_i. This results in the alternative difference formula

$$s'_i = \sum_{j=1}^{N-1} \Theta_{ij} J_j, \quad \text{for } i = 1, \ldots, N, \tag{2.41}$$

which will also be used below. An inspection of (2.37) shows that the difference matrix, Θ, for forward differences is

$$\Theta_{ij} = \frac{1}{(r_{i+1} - r_i)} \delta_{ij} = \begin{cases} \frac{1}{(r_{i+1} - r_i)} & \text{for } j = i \\ 0 & \text{otherwise} \end{cases}$$

Whichever formula for calculating the s'_i is chosen, the final step in defining the operation of $\mathbb{D}$ on s is to use (2.33) to obtain

$$\mathbb{D}s \overset{\text{def}}{\equiv} s'(r) = s'_i \ \text{ for } X_{i-1} < r \leq X_i, \text{ and for each } i. \tag{2.42}$$

This is, of course, another step function, a fact that is important in the repeated application of $\mathcal{D}$ to obtain higher derivatives.

Having obtained the definition of the effect of D on continuous, piecewise-differentiable functions, c, in (2.36), and of $\mathbb{D}$ on step functions, s, in (2.42) (via (2.40) or (2.41)), it is possible to define the operation of $\mathcal{D}$ on any function f of the type considered here using the decomposition (2.35) to give

$$\mathcal{D}f \overset{\text{def}}{\equiv} Dc + \mathbb{D}s, \tag{2.43}$$

the consistency of which is guaranteed by the uniqueness (see page 95) of the decomposition (2.35) of f into c and s. Note that D, $\mathbb{D}$, and therefore $\mathcal{D}$ are *linear* operators.

The discussion of the new derivative operator $\mathcal{D}$ so far has been relevant to any piecewise-continuous function with jumps only at discretization points (and which is differentiable everywhere within each bin I_i – so that $\mathbb{D}$ can be used). What we are most interested in, though, are functions in the chosen discretization space, $\mathcal{S}$. In particular, we want to know how to express the derivative $\mathcal{D}f$ of a function in $\mathcal{S}$ in terms of the free parameters, **f**, in the solution, so that the functional $\Phi[f]$ in (2.11) can be rewritten explicitly as a function of these parameters.

Given some function, $f(r)$, in $\mathcal{S}$, and thus expandable in terms of the basis functions, ϕ_j^i, as in (2.21), finding the expression for the function Dc is easy. Within each of the N bins, I_i, c is just given by $f(r) - s_i$, and it has already been stated that f can be differentiated in the usual sense within each bin (because the basis functions themselves can be), so

$$\begin{aligned} Dc\big|_r &= \frac{d}{dr}\left\{\sum_{j=1}^{n_i} f_j^i \phi_j^i(r) - s_i\right\} \\ &= \sum_{j=1}^{n_i} f_j^i \left.\frac{d\phi_j^i}{dr}\right|_r \quad \text{for } X_{i-1} < r \le X_i. \end{aligned} \tag{2.44}$$

In other words, c is just given by the usual derivative of the expansion (2.21) on each bin. Of course, we can make use of the relabelling (2.24) to get

$$Dc\big|_r = \sum_{l=L_{i-1}+1}^{L_i} f_l \left.\frac{d\phi_l}{dr}\right|_r \quad \text{for } X_{i-1} < r \le X_i, \tag{2.45}$$

where the notation

$$L_i = \sum_{k=1}^{i} n_k, \quad \text{for } i = 0, \ldots, N \tag{2.46}$$

has been used, so that the summation limits in (2.44) can be expressed in the new labelling.

Obtaining an expression for $\mathbb{D}s$ in terms of the f_j^i is a little more difficult. The jumps, J_i, can easily be obtained from (2.31):

$$J_i = \sum_{j=1}^{n_{i+1}} f_j^{i+1} \lim_{r \to X_i +} \phi_j^{i+1}(r) - \sum_{j=1}^{n_i} f_j^i \phi_j^i(X_i). \tag{2.47}$$

In this context the notation $\hat{\phi}_j^i(X_{i-1}) \equiv \lim_{r \to X_{i-1}+} \phi_j^i(r)$ will be introduced. Within the I_i the ϕ_j^i are often functions like sines, cosines, polynomials, etc., which can be evaluated for *any* values of their argument and are continuous, so this notation is quite handy (if $\phi_j^i = \sin a_{ij} r$, for example, then $\hat{\phi}_j^i(X_{i-1}) = \sin a_{ij} X_{i-1}$). With the goal of writing (2.47) as a matrix equation, use the relabelling (2.24) and this new notation in (2.47) to obtain

$$J_i = \sum_{l=L_i+1}^{L_{i+1}} f_l \hat{\phi}_l(X_i) - \sum_{l=L_{i-1}+1}^{L_i} f_l \phi_l(X_i). \tag{2.48}$$

With the benefit of (2.48) an $N \times n$ matrix, Ψ, can be defined by

$$\Psi_{il} = \begin{cases} -\phi_l(X_i) & \text{if } L_{i-1} + 1 \le l \le L_i \\ \hat{\phi}_l(X_i) & \text{if } L_i + 1 \le l \le L_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.49}$$

which obviously allows (2.48) to be rewritten as the matrix equation

$$J_i = \sum_{l=1}^{n} \Psi_{il} f_l \text{ for } i = 1, \ldots, N. \tag{2.50}$$

Using the formula (2.41) and writing the J_i and the s_i' in vector notation for conciseness, gives the desired expression for the s_i':

$$\mathbf{s}' = \Theta \mathbf{J} = \Theta \Psi \mathbf{f}. \tag{2.51}$$

Application of (2.42) then converts this vector to the required function of r.

From these results we can use (2.43) to say that the value of the function $\mathcal{D}f$ at a point r lying in bin I_i is given by

$$\mathcal{D}f\big|_r = \sum_{l=L_{i-1}+1}^{L_i} f_l \left.\frac{d\phi_l}{dr}\right|_r + \sum_{l=1}^{n} (\Theta\Psi)_{il} f_l. \tag{2.52}$$

It still remains to determine how to use this definition of $\mathcal{D}$ to obtain higher derivatives $\mathcal{D}^p f$. The fact that both Dc and $\mathbb{D}s$ in (2.43) are piecewise-continuous functions with jumps only at the discretization points means, obviously, that $\mathcal{D}f$ is also such a function. It can then be decomposed, just as f was in (2.35), into a continuous function c_1 and a step function s_1. The definition (2.43) can then be used to calculate $\mathcal{D}(\mathcal{D}f) = \mathcal{D}^2 f$. This, too, will be a piecewise-continuous function with jumps only at the discretization points, so this procedure can be repeated to obtain, recursively, derivatives of any order. In the practical implementation of this algorithm it is easier to keep the step functions resulting from previous decompositions separate, and apply the operator $\mathcal{D}$ only to the functions Dc derived at each stage, with the knowledge that the uniqueness of the decomposition (2.35) and the linearity of $\mathbb{D}$ make the two approaches identical. In the repeated application of the formalism for calculating $\mathcal{D}^q f$ the same objects, c, s, J_i, etc., will crop up at each step, so to distinguish them subscripts or superscripts will be used. An inductive argument will be used to prove the expression (2.63) for $\mathcal{D}^q f$, but first the second derivative will be calculated explicitly, so that (2.63) does not come as a complete surprise.

Starting with f in $\mathcal{S}$, use (2.35) to write $f = c_0 + s_0$ just as before (renaming the J_i and Ψ_{il} in (2.48) and (2.49) J_i^0 and Ψ_{il}^0 – so that (2.51) becomes $\mathbf{s}'_0 = \Theta \mathbf{J}^0 = \Theta \Psi^0 \mathbf{f}$, for example), and use the definitions of the operators D, $\mathbb{D}$ and $\mathcal{D}$ to obtain $\mathcal{D}f = Dc_0 + \mathbb{D}s_0$ as in (2.43). Dc_0 and $\mathbb{D}s_0$ are given in terms of the free parameters by (2.45) and (2.51), respectively. As was stated above, Dc_0 is piecewise continuous, with simple jumps at the X_i, so the (full) derivative operator $\mathcal{D}$ can be used on it. Now decompose Dc_0 according to (2.35) into $Dc_0 = c_1 + s_1$ (c_1 is continuous and s_1 is a step function as usual). Obviously, this gives $\mathcal{D}f = c_1 + (s_1 + s_0')$, where $s_1 + s_0'$ is a step function. From (2.45) it is easy to see that the jumps, J_i^1, in Dc_0 (from which s_1 is constructed via (2.32) and (2.33)) are given by

$$J_i^1 = \sum_{l=L_i+1}^{L_{i+1}} f_l \left.\widehat{\frac{d\phi_l}{dr}}\right|_{X_i} - \sum_{l=L_{i-1}+1}^{L_i} f_l \left.\frac{d\phi_l}{dr}\right|_{X_i} \tag{2.53}$$

using (2.48) (recall the paragraph following (2.47) where the hat notation, $\hat{}$, was defined). This can be reduced to matrix form (2.50), i.e. $\mathbf{J}^1 = \Psi^1\mathbf{f}$, with the definition

$$\Psi_{il}^1 = \begin{cases} -\left.\frac{d\phi_l}{dr}\right|_{X_i} & \text{if } L_{i-1}+1 \le l \le L_i \\ \left.\widehat{\frac{d\phi_l}{dr}}\right|_{X_i} & \text{if } L_i + 1 \le l \le L_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.54}$$

Applying the definition of $\mathcal{D}$ in (2.43) to $\mathcal{D}f = c_1 + (s_1 + s_0')$ gives

$$\mathcal{D}^2 f = Dc_1 + \mathbb{D}(s_1 + s_0')$$

The first term is easy to evaluate (recall that D is basically just the usual derivative operator except at discretization points where care must be taken to ensure that the *left-hand* derivative is taken – see (2.36)):

$$Dc_1\big|_r = \sum_{l=L_{i-1}+1}^{L_i} f_l \left.\frac{d^2\phi_l}{dr^2}\right|_r, \quad \text{for } X_{i-1} < r \le X_i.$$

To derive the expression for $\mathbb{D}(s_1 + s_0')$ first note that the linearity of $\mathbb{D}$ allows this to be written $\mathbb{D}s_1 + \mathbb{D}s_0'$. $\mathbb{D}$ operates on step functions through the application of the difference matrix Δ on the N values taken by the step function (equation (2.40)) to give the N values taken by its derivative. So

$$\mathbf{s}''_0 = \Delta\mathbf{s}'_0 = \Delta\Theta\mathbf{J}^0 = \Delta\Theta\Psi^0\mathbf{f},$$

and $\mathbb{D}s_0'$ can be obtained from this by using (2.42). The alternative form (2.41) for the operation of the difference matrix was not appropriate for use on s_0', because it assumes that we know the jumps in the step function, rather than its actual values, and the expression, (2.51), we have for s_0' only gives the values. However, the new step function s_1, obtained from decomposing Dc_0, is defined using the jumps $\mathbf{J}^1 = \Psi^1\mathbf{f}$ (see (2.53) and (2.54)), so the second difference formula can be used (and is actually simpler). This gives

$$\mathbf{s'}_1 = \Theta \mathbf{J}^1 = \Theta \Psi^1 \mathbf{f}.$$

Putting these results together we get

$$\mathcal{D}^2 f\Big|_r = Dc_1\big|_r + [s_1'(r) + s_0''(r)] = \sum_{l=L_{i-1}+1}^{L_i} f_l \left.\frac{d^2\phi_l}{dr^2}\right|_r + \left[\Theta\Psi^1\mathbf{f} + \Delta\Theta\Psi^0\mathbf{f}\right]_i \qquad (2.55)$$

(for $X_{i-1} < r \le X_i$, and for $i = 1, \ldots, N$). Comparing this with equation (2.52) for $\mathcal{D}f$, a pattern can be seen to be emerging. The operator $\mathcal{D}$ could be applied again to $\mathcal{D}^2 f$ in (2.55), and the pattern would be obvious, but the following inductive argument will be more than sufficient.

The set of equations

$$\begin{aligned} \mathcal{D}^0 f &= f \\ \mathcal{D}^1 f &= Dc_0 + \mathbb{D}s_0 = Dc_0 + s_0' \\ \mathcal{D}^2 f &= Dc_1 + \mathbb{D}(s_1 + s_0') = Dc_1 + (s_1' + s_0'') \\ &\vdots \\ \mathcal{D}^p f &= Dc_{p-1} + \sum_{q=1}^{p} s_{p-q}^{(q)} \end{aligned} \qquad (2.56)$$

illustrate the sequence of derivatives (on each line the decomposition (2.35) has been used to write $Dc_{q-1} = c_q + s_q$, so that the definition (2.43) of $\mathcal{D}$ is to be applied). A glance at equations (2.23), (2.45) and (2.55) indicates that the general expression for Dc_{q-1} is

$$Dc_{q-1}\big|_r = \sum_{l=L_{i-1}+1}^{L_i} f_l \left.\frac{d^q\phi_l}{dr^q}\right|_r \quad \text{for } X_i < r \le X_{i-1}, \qquad (2.57)$$

and, as can be seen from the fact that this is defined on each bin, this is piecewise-continuous function. This can easily be proved inductively. Obviously, it holds for $q = 1$ by virtue of (2.45). If (2.57) holds for some q, then piecewise-continuity allows (2.35) to be used to define a step function, $s_q(r)$, from the jumps, J_i^q, in Dc_{q-1}, through the procedure outlined in equations (2.31), (2.32) and (2.33) (with f replaced by Dc_{q-1}). In fact, applying (2.31) to the expansion (2.57) (which holds by assumption) gives

$$J_i^q = \sum_{l=1}^{n} \Psi_{il}^q f_l, \qquad (2.58)$$

(i.e. $\mathbf{J}^q = \Psi^q\mathbf{f}$, in vector notation) where

$$\Psi^q_{il} = \begin{cases} -\left.\frac{d^q\phi_l}{dr^q}\right|_{X_i} & \text{if } L_{i-1}+1 \le l \le L_i \\ \left.\widehat{\frac{d^q\phi_l}{dr^q}}\right|_{X_i} & \text{if } L_i+1 \le l \le L_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.59}$$

(*cf.* equations (2.49) and (2.54)). This expression for Ψ will be useful later in the evaluation of the step functions occurring in $\mathcal{D}^p f$. The value, s_i^q, of $s_q(r)$ for r in the ith bin is given from (2.32). A continuous function $c_q(r)$ can then be introduced, according to (2.34), such that $c_q(r) = Dc_{q-1}|_r - s_i^q$ for r in the ith bin. The defintion (2.36) of the operator D then says that, on the ith bin, Dc_q is given by

$$Dc_q = \sum_{l=L_{i-1}+1}^{L_i} f_l \frac{d}{dr}\left.\frac{d^q\phi_l}{dr^q}\right|_r - \frac{d}{dr}s_i^q = \sum_{l=L_{i-1}+1}^{L_i} f_l \left.\frac{d^{q+1}\phi_l}{dr^{q+1}}\right|_r$$

(taking care at the end-points of the bins as usual) which satisfies (2.57) and thus proves the hypothesis.

It only remains now to consider the step function part of $\mathcal{D}^p f$. The proof of (2.57) constructed the step function $s_q(r)$, which can be used to show that (2.56) is indeed true. (It is true for $p = 1$ by (2.43) and (2.42). If it holds for some p, then $Dc_{p-1} = c_p + s_p$ gives

$$\mathcal{D}^{p+1} f = \mathcal{D}\left(c_p + s_p + \sum_{q=1}^{p} s^{(q)}_{p-q}\right) = Dc_p + \mathbb{D}\left(s_p + \sum_{q=1}^{p} s^{(q)}_{p-q}\right) = Dc_p + \sum_{q=1}^{p+1} s^{(q)}_{(p+1)-q},$$

which is (2.56) again). Furthermore, from the proof of (2.57) we have an expression, (2.58), for the jumps in Dc_{q-1}, from which s_q is constructed. We also know that these jumps alone are sufficient to allow the derivative of s_q to be calculated from (2.41):

$$s_q'(r) \equiv s_i^{q\prime} = \sum_{j=1}^{N-1} \Theta_{ij} J_i^q = (\Theta\Psi^q\mathbf{f})_i \;\; \text{for } X_{i-1} < r \le X_i \tag{2.60}$$

(using (2.58)). For the repeated derivatives of the s_q' the form (2.41) for the derivative is no longer available, because only the values on each bin of the step functions to be differentiated are known, not their jumps. The alternative expression (2.40) must therefore be used, so that

$$s^{(q)}_{p-q}(r) = {s'_{(p-q)}}^{(q-1)}(r) = (\Delta^{q-1}\mathbf{s}^{p-q\prime})_i = (\Delta^{q-1}\Theta\Psi^{p-q}\mathbf{f})_i \;\; \text{for } X_{i-1} < r \le X_i.$$

Introducing the notation

$$B^{(p)} = \sum_{q=1}^{p} \Delta^{q-1}\Theta\Psi^{p-q} \tag{2.61}$$

(so that $B^{(p)}$ is an $N \times n$ matrix) obviously then allows (2.56) to be written

$$\mathcal{D}^p f\big|_r = Dc_{p-1}\big|_r + \sum_{l=1}^{n} B_{il}^{(p)} f_l \quad \text{for } X_{i-1} < r \leq X_i.$$

At last, making use of (2.57) for Dc_{p-1} gives the expression we need for $\mathcal{D}^p f$:

$$\mathcal{D}^q f\big|_r = \sum_{l=L_{i-1}+1}^{L_i} f_l \left.\frac{d^q\phi_l}{dr^q}\right|_r + \sum_{l=1}^{n} B_{il}^{(q)} f_l, \quad \text{for } X_{i-1} < r \leq X_i, \tag{2.62}$$

or, alternatively, using the relabelling (2.24) to rewrite this in terms of the f_j^i, which makes it easier to see the significance of the discretization bins:

$$\mathcal{D}^q f\big|_r = \sum_{j=1}^{n_i} f_j^i \left.\frac{d^q\phi_j^i}{dr^q}\right|_r + (B^{(q)}\mathbf{f})_i, \quad \text{for } X_{i-1} < r \leq X_i. \tag{2.63}$$

Calculating the generalized difference matrix $B^{(p)}$ requires knowledge only of the matrices Θ and Δ (which can be determined entirely from the discretization points in (2.16)), and of the values of the basis functions and their derivatives at the end-points of each bin. Evaluation of the Dc_{p-1} part of $\mathcal{D}^p f$ needs only the values of the pth derivatives of the basis functions within each bin (see (2.57)). In the implementation of this algorithm it is a simple matter to provide numerical routines to supply these quantities as required. This will be considered in more detail in §2.2.4.

Making use of the final expression, (2.63), for $\mathcal{D}^q f$ in terms of the free parameters f_j^i, as well as equation (2.19), in the general regularizing functional (2.15) gives $\Phi(\mathbf{f})$ as an explicit function of the f_j^i:

$$\Phi(\mathbf{f}) = \sum_{i=1}^{N} \int_{X_{i-1}}^{X_i} \varphi\left(\sum_{j=1}^{n_i} f_j^i \phi_j^i(r)\,, \ldots, \sum_{j=1}^{n_i} f_j^i \left.\frac{d^p\phi_j^i}{dr^p}\right|_r + (B^{(p)}\mathbf{f})_i\,; r\right) \rho(r)\, dr, \tag{2.64}$$

where the integral has already been broken up into integrals over the individual discretization bins, allowing equation (2.19) to be used to give the value of $f(r)$, and equation (2.61) to be used for the value of the step function part, $(B^{(p)}\mathbf{f})_i$, of the derivatives on each bin. This is the clearest expression for $\Phi(\mathbf{f})$, but in some circumstances (see the following section, §2.2.4, for example) it may be better to write f and its derivatives using the alternative expression (2.62), so that both summations occurring in (2.64) become $\sum_{L_{i-1}+1}^{L_i}$:

$$\Phi(\mathbf{f}) = \sum_{i=1}^{N} \int_{X_{i-1}}^{X_i} \varphi\left(\sum_{L_{i-1}+1}^{L_i} f_l \phi_l(r)\,, \ldots, \sum_{L_{i-1}+1}^{L_i} f_l \left.\frac{d^p\phi_l}{dr^p}\right|_r + (B^{(p)}\mathbf{f})_i\,; r\right) \rho(r)\, dr. \tag{2.65}$$

The following section will make use of this expression to derive a more explicit form for the smoothing constraint with a particular, and very important, type of regularization: *quadratic* regularization.

2.2.4 Specialization to quadratic RLS

Quadratic regularizing constraints are those functionals (2.11) in which the function φ is just a quadratic polynomial in f and its derivatives. In the general case there will be cross-terms of the form $\frac{d^s f}{dr^s}\frac{d^t f}{dr^t}$, but these will not be evaluated here because we will concentrate on the most common form of quadratic RLS, simple pth-order smoothing (2.10), and also because once the derivation of the expression for $\Phi(\mathbf{f})$ with pth-order smoothing has been given it will be obvious how to generalize this to *any* quadratic smoothing constraint.

As the equation (2.10) shows, for pth-order smoothing the function φ in (2.11) is given by $\varphi = \left(\frac{d^p f}{dr^p}\right)^2$. Replacing the occurrence of derivatives in this φ with the operator $\mathcal{D}$ gives $\varphi = (\mathcal{D}^p f)^2$, making the regularizing functional

$$\Phi^{(p)}[f] = \int_0^{R_\odot} (\mathcal{D}^p f|_r)^2 \rho(r)\, dr. \tag{2.66}$$

The positive weighting function has been retained here (in contrast to (2.10)) because it will be used in later chapters to allow the amount of smoothing applied to vary between different parts of the solution. Using the expression (2.65) (with $\varphi = \left(\frac{d^p f}{dr^p}\right)^2$) to evaluate this functional for f in the discretization space results in

$$\Phi^{(p)}(\mathbf{f}) = \sum_{i=1}^{N} \int_{X_{i-1}}^{X_i} \left[\sum_{L_{i-1}+1}^{L_i} f_l \left.\frac{d^p \phi_l}{dr^p}\right|_r + (B^{(p)}\mathbf{f})_i \right]^2 \rho(r)\, dr. \tag{2.67}$$

The goal in this section is to reduce $\Phi^{(p)}(\mathbf{f})$ to the simple form (2.74), in which a double sum over the n components of $\mathbf{f}$ is implied. To this end, it is a good idea to attempt to rewrite the restricted sum over l in (2.67) as a sum over all l, and it is a good idea to do this now before things become unbearably messy. The trick is to recognize that the definition (2.46) and the relabelling (2.24) give

$$\sum_{L_{i-1}+1}^{L_i} \equiv \sum_{l=1}^{n} \delta_{i,i(l)} \tag{2.68}$$

($\delta_{i,i(l)}$ is the kronecker delta. Its indices are the free index i and the 'dummy' index $i(l)$ obtained from (2.24). Consequently, $\delta_{i,i(l)}$ is zero unless l is the label of a basis function $\phi_{j(l)}^{i(l)}$

defined on the ith bin, i.e. with $L_{i-1}+1 \leq l \leq L_i$.) Note, too, that $(B^{(p)}\mathbf{f})_i = \sum_{l=1}^{n} B_{il}^{(p)} f_l$. So, equation (2.67) becomes

$$\Phi^{(p)}(\mathbf{f}) = \sum_{i=1}^{N} \int_{X_{i-1}}^{X_i} \left\{ \sum_{l+1}^{n} f_l \left[\delta_{i,i(l)} \left. \frac{d^p \phi_l}{dr^p} \right|_r + B_{il}^{(p)} \right] \right\}^2 \rho(r)\, dr, \tag{2.69}$$

the integrand of which can be expanded to give four terms so that (2.69) becomes

$$\begin{aligned} \Phi^{(p)}(\mathbf{f}) &= \sum_{i=1}^{N} \sum_{k,l=1}^{n} f_k f_l \left\{ \delta_{i,i(l)} \delta_{i,i(k)} \int_{X_{i-1}}^{X_i} \left. \frac{d^p \phi_k}{dr^p} \right|_r \left. \frac{d^p \phi_l}{dr^p} \right|_r \rho(r)\, dr \right. \\ &+ \left(B_{il}^{(p)} \delta_{i,i(k)} \int_{X_{i-1}}^{X_i} \left. \frac{d^p \phi_k}{dr^p} \right|_r \rho(r)\, dr + B_{ik}^{(p)} \delta_{i,i(l)} \int_{X_{i-1}}^{X_i} \left. \frac{d^p \phi_l}{dr^p} \right|_r \rho(r)\, dr \right) \\ &+ \left. B_{ik}^{(p)} B_{il}^{(p)} \int_{X_{i-1}}^{X_i} \rho(r)\, dr \right\}. \end{aligned} \tag{2.70}$$

Now interchange the order of the sums over i and over k and l, observe in the first term that the sum over i can be evaluated explicitly, giving $\sum_{i=1}^{N} \delta_{i,i(k)} \delta_{i,i(l)} = \delta_{i(k)i(l)}$, and define three new matrices as follows:

$$C_{0\,kl}^{(p)} = \delta_{i(k)i(l)} \int_{X_{i(k)-1}}^{X_{i(k)}} \left. \frac{d^p \phi_k}{dr^p} \right|_r \left. \frac{d^p \phi_l}{dr^p} \right|_r \rho(r)\, dr \tag{2.71}$$

$$P_{il}^{(p)} = \delta_{i,i(l)} \int_{X_{i-1}}^{X_i} \left. \frac{d^p \phi_l}{dr^p} \right|_r \rho(r)\, dr \tag{2.72}$$

$$C_{S\,kl}^{(p)} = \sum_{i=1}^{N} B_{ik}^{(p)} B_{il}^{(p)} \int_{X_{i-1}}^{X_i} \rho(r)\, dr. \tag{2.73}$$

Obviously, $C_0{}^{(p)}$ and $C_S{}^{(p)}$ are $n \times n$ matrices, whereas $P^{(p)}$ is $N \times n$ (the same size as $B^{(p)}$ – see equation (2.61)). Together, these definitions allow (2.70) to be reduced to the much simpler form

$$\Phi^{(p)}(\mathbf{f}) = \sum_{k,l=1}^{n} f_k f_l \left[C_{0\,kl}^{(p)} + \sum_{i=1}^{N} \left(B_{ik}^{(p)} P_{il}^{(p)} + B_{il}^{(p)} P_{ik}^{(p)} \right) + C_{S\,kl}^{(p)} \right],$$

or, in vector notation, even more concisely

$$\Phi^{(p)}(\mathbf{f}) = \mathbf{f}^T C^{(p)} \mathbf{f} \tag{2.74}$$

(^{T} denotes matrix transpose) where the pth-order smoothing matrix $C^{(p)}$ can be broken down into

$$C^{(p)} = C_0{}^{(p)} + ({B^{(p)}}^T P^{(p)} + {P^{(p)}}^T B^{(p)}) + C_S{}^{(p)}. \tag{2.75}$$

$C_0{}^{(p)}$, $B^{(p)}$, $P^{(p)}$ and $C_S{}^{(p)}$ are defined in equations (2.71), (2.61), (2.72) and (2.73), respectively.

The important point in all this is that, although the process of deriving the preceding expressions has been messy and awkward, the calculations that must be performed to effect the numerical implementation of this algorithm are actually quite simple. First, note that the kronecker deltas appearing in equations (2.71) and (2.72) for ${C_0}^{(p)}$ and $P^{(p)}$ just mean 'only evaluate this quantity for basis functions in the same bin (for ${C_0}^{(p)}$) or in the ith bin (for $P^{(p)}$), all other matrix elements are zero'. This considerably reduces the number of quantities that must be evaluated. Note also, that it is only necessary to provide a subroutine returning the values of each of the basis functions ϕ^i_j and their derivatives at the boundaries of their discretization bin X_{i-1} and X_i, for use in the procedure to calculate the matrix $B^{(p)}$, and to provide subroutines to return the values of the integrals occurring, in equations (2.71), (2.72) and (2.73). For the most common discretization procedures this is a lot easier than it sounds, particularly when the weighting function is taken to be a constant on each bin (ρ_i, on the ith bin, say), for then $\int_{X_{i-1}}^{X_i} \rho(r)\,dr = \rho_i(X_i - X_{i-1})$ in (2.73), the integrals in (2.72) are just the integrals of the pth derivatives of the basis functions, which means they are the difference between the $(p-1)$th derivatives evaluated at the ends of each bin (so the routine used to return derivatives in the calculation of $B^{(p)}$ can be used here too), and the integral in (2.71) is just the 'inner product' of two functions, which means that when those functions are orthogonal (as often occurs in practical situations) this integral is zero. When zeroth order smoothing is being employed ($p = 0$), there are no derivatives, and the elements of C_0 are just the inner products of the basis functions, which will be zero unlesss $k = l$ for *any* orthogonal basis (such as a Fourier series, legendre polynomials, etc.). When the basis functions are sines and cosines, these integrals are zero unless $k = l$ even for their derivatives (because the derivatives of sines and cosines are cosines and sines with the same argument). In other words, for the commonly used discretizations the integrals that must be evaluated in the calculation of the smoothing matrix are often very simple and analytically defined, so that the subroutines required can be written with some generality. This is probably the most important aspect of this algorithm: it is possible to create subroutines to evaluate the integrals and derivatives required by this algorithm in a general way, so that the user need only specify the discretization points (2.16) and the type of discretization to be used on each bin (e.g. polynomial expansion, orthogonal function expansion using cosines, orthogonal function expansion using legendre polynomials, piecewise-constant approximation etc.) and the numerical routines will do the rest.

To make the creation of such general subroutines possible it is helpful to think of a 'generic' discretization bin with a coordinate y going between zero and one, to write any set of basis functions in terms of this new coordinate, and to perform the evaluations of the derivatives and integrals in this new coordinate system. Suppose, for example, that some set of basis functions $\beta_1, \beta_2, \ldots$ is to be included in the implementation of this algorithm. Any such set of functions has a natural range of definition. The natural range to use for polynomial expansion (basis functions $1, y, y^2, \ldots$) is (0,1]. If the basis comprises *orthogonal* functions these will only be orthogonal over a specific interval (for a sine and cosine expansion the interval must contain exactly a whole number of half-wavelengths, for example). If necessary, rescale the coordinate in the definition of these functions so that this interval is $(0, 1]$, and write subroutines that return the values of:

1. $\left.\frac{d^q\beta_k}{dy^q}\right|_y$ for any k and q, and for any y in $[0, 1]$. Actually, it is only strictly necessary to calculate $\left.\frac{d^q\beta_k}{dy^q}\right|_y$ for $y = 0, 1$, but it is often useful to obtain the values of the derivatives of the solution at any point which means calculating $\left.\frac{d^q\beta_k}{dy^q}\right|_y$ for any y.

2. $\int_0^y \left.\frac{d^q\beta_k}{dy^q}\right|_{y'} dy'$ for any q and k and any y in $[0, 1]$. Again, it is only strictly necessary to evaluate this for $y = 1$. Note that for $q \geq 1$ this integral is $\left.\frac{d^{q-1}\beta_k}{dy^{q-1}}\right|_y - \left.\frac{d^{q-1}\beta_k}{dy^{q-1}}\right|_0$, which can be obtained from the subroutine in 1., so that all that is needed here is the evaluation of $\int_0^y \left.\frac{d^q\beta_k}{dy^q}\right|_{y'} dy'$ for $q = 0$, i.e. $\int_0^y \beta_k(y')\,dy'$. Indeed, for $y = 1$, which is all that is really required, evaluation of this integral ought to be very simple in most cases.

3. $\int_0^1 \frac{d^q\beta_k}{dy^q}\frac{d^q\beta_l}{dy^q}\,dy$ for any q, k and l. In certain cases (such as when the β_k are sines and cosines in a Fourier series) these integrals will only be non-zero for $k = l$, but this will not be true in general.

All of these quantities can be calculated with a knowledge only of the β_k; they are completely independent of any discretization. It will now be shown how these quantities arise in the discretization procedure, and that they are all that is required for the general application of any such set of basis functions β_k to the RLS inversion of (2.5).

Whenever the basis functions β_k are used in a discretization, the bin on which they are defined will not be $(0,1]$, but will instead be $I_i = (X_{i-1}, X_i]$ for some i. The simple linear coordinate transformation

$$y(r) = mr + c, \quad \text{with } m = \frac{1}{(X_i - X_{i-1})} \text{ and } c = -\frac{X_{i-1}}{(X_i - X_{i-1})} \tag{2.76}$$

can be used to define the basis functions on the ith bin to be

$$\phi_j^i(r) = \beta_j[y(r)] \quad \text{for } X_{i-1} < r \leq X_i, \text{ and for } j = 1, \ldots, n_i. \tag{2.77}$$

From the coordinate transformation (2.76) it is easy to see that $\frac{d}{dr} = \frac{dy}{dr}\frac{d}{dy} = m\frac{d}{dy}$, which can be used repeatedly to give

$$\frac{d^q}{dr^q} = m^q \frac{d^q}{dy^q}. \tag{2.78}$$

Also, $\int_{X_{i-1}}^{X_i} dr = \int_0^1 \frac{dr}{dy} dy = \frac{1}{m}\int_0^1 dy$. These two results can be used in the evaluation of the three quantities needed in the calculation of the smoothing matrix:

$$\begin{aligned}
\left.\frac{d^p\phi_j^i}{dr^p}\right|_r &= m^p \left.\frac{d^p\beta_j}{dy^p}\right|_y \\
\int_{X_{i-1}}^{X_i} \left.\frac{d^p\phi_j^i}{dr^p}\right|_r dr &= \frac{1}{m}\int_0^1 m^p \left.\frac{d^p\beta_j}{dy^p}\right|_y dy = m^{p-1}\int_0^1 \left.\frac{d^p\beta_j}{dy^p}\right|_y dy \\
\int_{X_{i-1}}^{X_i} \frac{d^p\phi_{j_1}^i}{dr^p}\frac{d^p\phi_{j_2}^i}{dr^p} dr &= \frac{1}{m}\int_0^1 m^{2p}\frac{d^p\beta_{j_1}}{dy^p}\frac{d^p\beta_{j_2}}{dy^p} dy = m^{2p-1}\int_0^1 \frac{d^p\beta_{j_1}}{dy^p}\frac{d^p\beta_{j_2}}{dy^p} dy.
\end{aligned}$$

Quite clearly, these calculations only require knowledge of the discretization points (so that m can be calculated from (2.76)) and the subroutines referred to above for calculating derivatives and integrals of the generic basis functions β_k. This means that once the three subroutines have been written for each set of basis functions (polynomial expansion, sines, cosines, legendre polynomials, etc.) to be used in the implementation of the algorithm, the basis functions can be used on any bin without difficulty, and *without the need to write code specifically for that discretization.*

It turns out that in the discretization of the kernel functions in (2.25) it is advantageous to supplement the three subroutines described above with another to return the value of repeated integrals of the basis functions β_k:

$$\mathcal{I}_k^q(y) \stackrel{\text{def}}{\equiv} \int_0^y \int_0^{y_q} \cdots \int_0^{y_2} \beta_k(y_1)\, dy_1 \ldots dy_q. \tag{2.79}$$

The kernel functions in helioseismology are generally found by solving the differential equations of oscillation numerically, which procedure typically results in finding the values of the kernel functions at a grid of points $\zeta_1, \zeta_2, \ldots$ throughout the model sun. The kernels are then defined at intervening points by interpolation. Splines (Cox 1975) of some kind are the best bet – either piecewise-linear or cubic splines – which means that between any two grid-points the kernel is approximated by some polynomial, of degree ν, say (for linear interpolation $\nu = 1$, for cubic spline interpolation $\nu = 3$): $k_i(r) \approx \sum_{\kappa=0}^{\nu} \alpha_\kappa^{(\gamma)} r^\kappa$ for $\zeta_{\gamma-1} \leq r \leq \zeta_\gamma$. When the basis functions ϕ_j^i are just related to the β_k as in (2.77), the integrals appearing in the discretization of the $\mathcal{K}_i$ in (2.25) then just break up into a sequence of integrals between adjacent grid-points of (sums of) moments of the β_k:

$$\begin{aligned}\sum_{\kappa=0}^{\nu} \alpha_\kappa^{(\gamma)} \int_{\zeta_{\gamma-1}}^{\zeta_\gamma} r^\kappa \beta_k[y(r)]\, dr &= \sum_{\kappa=0}^{\nu} \alpha_\kappa^{(\gamma)} \frac{1}{m} \int_{y_{\gamma-1}}^{y_\gamma} \frac{(y-c)^\kappa}{m^\kappa} \beta_k(y)\, dy \\ &= \sum_{\kappa=0}^{\nu} \tilde{\alpha}_\kappa^{(\gamma)} \frac{1}{m^{\kappa+1}} \int_{y_{\gamma-1}}^{y_\gamma} y^\kappa \beta_k(y)\, dy,\end{aligned}$$

where the inverse of the transformation in (2.76) has been used to write r as a (linear) function, $r = (y-c)/m$, of y and the integral variable has been changed to y in the first step, $(y-c)^\kappa$ has then been expanded, and the terms collected up (giving new coefficients $\tilde{\alpha}_\kappa$ in the expansion). Integrating this by parts repeatedly, it is easy to see that each such term will become a sum of terms containing $\mathcal{I}_k^q(y_\gamma) - \mathcal{I}_k^q(y_{\gamma-1})$ for different q. It may seem that providing a subroutine to evaluate the $\mathcal{I}_k^q$ for any set of functions β_k involves considerable work, and in some cases it will, but often (such as when the β_k are simple analytic functions such as powers of y or sines and cosines) an explicit analytic expression for the integrals can be derived.

The procedure just described enables any of the matrices and other quantities needed for performing the inversion of (2.5) using this algorithm to be calculated. In particular, the preceding discussion shows exactly how to calculate the smoothing matrix in (2.75) and the associated equations (2.61), (2.71), (2.72) and (2.73), and the kernel matrix in (2.25). It can be seen from equation (2.82) for the explicit form of the solution with quadratic regularization that these two matrices are all that is needed to calculate the solution, apart from the data vector, g, given in (2.29).

It is still necessary, though, to give the final discretized form of the functional $\mathcal{R}$ that must be minimized to find the solution when quadratic regularization is used. The expression (2.74) for the discretized regularizing functional corresponding to pth-order smoothing can be combined with equation (2.30) for the discretized χ^2 functional to give the discretized form of the functional $\mathcal{R}$ (equation (2.9)) with pth-order smoothing, which will be denoted by $\mathcal{R}^{(p)}$:

$$\mathcal{R}^{(p)}(\mathbf{f}) = \|\mathbf{g} - H\mathbf{f}\|^2 + \lambda \mathbf{f}^T C^{(p)} \mathbf{f} \tag{2.80}$$

(remember the definitions (2.27) and (2.29)).

The final step in the procedure will take (2.80) and use it to derive an explicit expression (2.82) for the solution vector $\mathbf{f}$ in terms of the data $\mathbf{g}$ (and the kernel matrix and smoothing matrix).

2.2.5 Solution

For general (non-quadratic) regularizing functionals, the minimization of $\mathcal{R}(\mathbf{f})$ will usually require the use of general optimization algorithms (see *Numerical Recipes*, chapter 10). Here, though, we concentrate on the minimization of $\mathcal{R}(\mathbf{f})$ when quadratic RLS is being used.

The first step in the (well-known and standard) derivation of the explicit solution to the minimization of $\mathcal{R}^{(p)}$ in (2.80) over all possible solution vectors, $\mathbf{f}$, is to expand $\mathcal{R}^{(p)}$ and collect terms of the same order in $\mathbf{f}$ together, to give

$$\mathcal{R}^{(p)}(\mathbf{f}) = g^2 + \mathbf{f}^T(H^T H + \lambda C^{(p)})\mathbf{f} - 2\mathbf{g}^T H \mathbf{f} \tag{2.81}$$

(where $g^2 = \|\mathbf{g}\|^2 = \mathbf{g}^T\mathbf{g}$). Obviously, at a minimum of $\mathcal{R}^{(p)}$ the derivatives of $\mathcal{R}^{(p)}$ with respect to the components of $\mathbf{f}$ must be zero, so (2.81) gives, in component notation,

$$\frac{\partial \mathcal{R}^{(p)}}{\partial f_l} = 2\sum_{k=1}^{n}(H^T H + \lambda C^{(p)})_{lk} f_k - 2(H^T\mathbf{g})_l = 0 \quad \text{for } l = 1, \ldots, n,$$

which immediately gives the desired solution:

$$\mathbf{f} = (H^T H + \lambda C^{(p)})^{-1} H^T \mathbf{g}. \tag{2.82}$$

In this last step it has been assumed that the matrix $H^T H + \lambda C^{(p)}$ is non-singular, but this is reasonable since it was to ensure this that regularization was introduced in the first place (although see §2.4). The zero eigenvalues of $H^T H + \lambda C^{(p)}$ are a reflection of the non-uniqueness in the original problem (2.5), because any two solutions differing only by a component that is an eigenvector of $H^T H + \lambda C^{(p)}$ belonging to a zero eigenvalue will give rise to the same data. Moreover, very small eigenvalues of $H^T H + \lambda C^{(p)}$ (which would make this matrix ill-conditioned) would correspond to instability for much the same reason. If the regularization was at all successful both of these problems should have been removed, so $H^T H + \lambda C^{(p)}$ should be well-conditioned for inversion.

This concludes the explanation of the inversion algorithm that will be used in numerical calculations throughout the rest of this thesis. The remaining sections of this chapter concentrate primarily on important details of the practical implementation of quadratic RLS.

2.3 Applying constraints to the solution

It is often useful in solving inverse problems such as (2.5) to apply constraints to the solution (Jeffrey and Rosner 1988). In helioseismology the most obvious example of this is constraining the surface rotation rate inferred from splitting data to match the observed values. Such constraints can be enforced in any method to solve such inverse problems, but implementing the constraints when using RLS methods is particularly simple. Here we adopt the formalism of the previous section, where the solution was expanded in terms of a (finite) set of basis functions, ϕ_j^i, to examine the effect on the recovered solution of such constraints. It should be noted, though, that the ideas and procedures presented can be applied to all the RLS methods for solving (2.5), not just to the algorithm described in this chapter.

Constraints (such as the surface constraint on the solar rotation rate) that fix the value of the solution, $f(r)$, or more generally, $f^{(p)}(r)$ (for $p > 0$), at some point $r = \zeta$ have the form

$$f^{(p)}(\zeta) \equiv \sum_{j=1}^{n_i} f_j^i \left. \frac{d^p \phi_j^i}{dr^p} \right|_{\zeta} = c \tag{2.83}$$

for ζ in $(X_{i-1}, X_i]$. Since the $\left. \frac{d^p \phi_j^i}{dr^p} \right|_{\zeta}$ are, for a given ζ, just numbers, (2.83) is just a linear relationship involving the free parameters in the inversion (the components of the solution, f, in the basis $\{\phi_j^i\}$ of $\mathcal{S}$). With the algorithm described in this chapter applying constraints takes on even greater significance. Apart from fixing the solution and its derivatives at some point, it may be desirable to require continuity in the solution or its derivatives across the boundary between two discretization bins (i.e. continuity at a discretization point, X_i). This is because, although the smoothing restricts jumps in the solution across discretization points to some extent (through the $B^{(p)}$ term in the expression (2.63) for the derivative of the solution function, f, in terms of the free parameters, **f**), the contribution to the derivative from jumps at discretization points is weighted so that a jump contributes an amount to the derivative which decreases as the width of the neighbouring bins increases. So, for large bins the jumps at the ends of the bins may not be very well controlled by the smooothing and can be quite large (see fig. 5.3, for example, where the jumps in the solution for discretization method 3 are very noticeable). Often this will be acceptable, but in cases where the solution is known to be continuous, for example, such large jumps must be avoided. Obviously, enforcing such continuity constraints gives rise to a condition of the form

$$f^{(p)}(X_i) = \lim_{r \to X_i+} f^{(p)}(r)$$

(where the limit arises from the usual conventions about half-open intervals, described in the discussion following equation (2.17)), which may easily be written

$$\sum_{j=1}^{n_i} f_j^i \left. \frac{d^p \phi_j^i}{dr^p} \right|_{X_i} - \sum_{j=1}^{n_{i+1}} f_j^{i+1} \left. \frac{d^p \phi_j^{i+1}}{dr^p} \right|_{X_i} = 0. \tag{2.84}$$

This is again a linear relationship between the free parameters in the inversion. In general, all such constraints may be written

$$\sum_{l=1}^{n} \alpha_l f_l = \beta.$$

If several (ν, say) such constraints are being enforced, then we can write

$$\sum_{l=1}^{n} \alpha_l^{(r)} f_l = \beta^{(r)}, \text{ for } r = 1, \ldots, \nu, \tag{2.85}$$

or,

$$A\mathbf{f} = \mathbf{b} \tag{2.86}$$

in matrix notation, with A a $\nu \times n$ matrix and $\mathbf{b}$ a ν-vector. Obviously, $A_{rl} \equiv \alpha_l^{(r)}$ and $b_r \equiv \beta^{(r)}$.

Mathematically, each (independent) constraint removes a degree of freedom from the solution function, and so all the constraints together force the solution to lie in some proper affine subspace of the original discretization space, $\mathcal{S}$. It is necessary, though, to discuss the practical implementation of the constraints. How is this system of constraints used in combination with the data and smoothing constraint contained in the discretized functional $\mathcal{R}(\mathbf{f})$ in (2.80), which previously was to be minimized over all of the parameters in **f**? In what follows the regularization will generally be assumed to be quadratic, but not necessarily pth-order smoothing (2.10). The superscript on $\mathcal{R}^{(p)}$ will therefore be dropped.

It is perfectly possible to introduce the restrictions imposed by (2.85) into the functional $\mathcal{R}$ using lagrange multipliers, μ_r, in the minimization over the f_l, and to derive an analytic expression for the solution vector in the usual way from this new functional. That is, we replace $\mathcal{R}(\mathbf{f})$ by

$$\mathcal{R}(\mathbf{f}) + \sum_{r=1}^{\nu} \mu_r(\boldsymbol{\alpha}^{(r)T}\mathbf{f} - \beta^{(r)})$$

and take derivatives of this functional with respect to the f_l for $l = 1, \ldots, n$. With some effort the resulting formulae can be written in such a way that they give an explicit expression for the **f** that minimizes this functional.

However, it is much simpler (and equivalent) to consider (2.86) right at the outset, and to effectively change basis in the discretization subspace, $\mathcal{S}$, to a new basis in which the constraints become

$$\begin{array}{ll} \tilde{f}_l & = \tilde{\beta}_l \text{ for } i = 1, \ldots, \tilde{\nu} \\ \tilde{f}_l & \text{is unconstrained otherwise.} \end{array}$$

Here $\tilde{\nu}$ is the number of *independent* constraints, which ought to be the same as the actual number of constraints, ν (there is no point applying the same constraint twice), but if ν is large constraints may be effectively linearly dependent (to the level of accuracy that can be achieved numerically), so it can happen that $\tilde{\nu} < \nu$.

This change of basis is purely a mathematical procedure and does not require a consideration of the basis functions ϕ_j^i. It is achieved by performing *singular value decomposition*, or SVD (see *Numerical Recipes*, §2.6), on the matrix A. Note that $\tilde{\nu}$ must be less than n, the dimension of the space $\mathcal{S}$ and hence the length of the vector **f**, otherwise the solution is completely determined by the constraints and the data become irrelevant. In any sensible case $\nu < n$ (usually, $\nu \ll n$), and in what follows we will assume that this holds. Usually, too, $\tilde{\nu} = \nu$, but we will not insist on this here. Using SVD it is possible to write

$$A = PDQ^T \tag{2.87}$$

where P is a $\nu \times \nu$ orthogonal matrix, Q is an $n \times n$ orthogonal matrix and D is a $\nu \times n$ matrix which is 'diagonal' in the sense that $D_{ij} = 0$ if $i \neq j$. The elements $d_i \equiv D_{ii}$ for $i = 1, \ldots, \nu\}$ are the *singular values* of A, and the number of non-zero singular values is the rank of A, which equals the number of linearly independent constraints, $\tilde{\nu}$. So $d_i \neq 0$

for $i = 1, \ldots, \tilde{\nu} < n$ and $d_i = 0$ for $i > \tilde{\nu}$. Then

$$A\mathbf{f} = PDQ^T\mathbf{f} = \mathbf{b},$$

so, using the orthonormailty of P,

$$D(Q^T\mathbf{f}) = P^T\mathbf{b} \overset{\text{def}}{\equiv} \tilde{\mathbf{b}}.$$

Defining $\tilde{\mathbf{f}} = Q^T\mathbf{f}$ then gives

$$D\tilde{\mathbf{f}} = \tilde{\mathbf{b}}.$$

Since $D_{ij} \neq 0$ if and only if $i = j$ and $i \leq \tilde{\nu}$, write D as a block matrix $(D_1 | \mathbf{0})$, where D_1 is $\nu \times \tilde{\nu}$, and partition $\tilde{\mathbf{f}}$ correspondingly as

$$\tilde{\mathbf{f}} = \begin{pmatrix} \tilde{\mathbf{f}}_1 \\ \hline \tilde{\mathbf{f}}_2 \end{pmatrix} \begin{matrix} \}\tilde{\nu} \\ \}n - \tilde{\nu} \end{matrix} . \tag{2.88}$$

Then

$$D\tilde{\mathbf{f}} = (D_1 | \mathbf{0}) \begin{pmatrix} \tilde{\mathbf{f}}_1 \\ \hline \tilde{\mathbf{f}}_2 \end{pmatrix} = D_1\tilde{\mathbf{f}}_1 = \tilde{\mathbf{b}}. \tag{2.89}$$

This is almost the desired result. The only subtlety arises from the fact that $\nu = \tilde{\nu}$ was not insisted upon. If $\nu = \tilde{\nu}$ then D_1 is $\tilde{\nu} \times \tilde{\nu}$, diagonal and of full rank, so

$$\tilde{\mathbf{f}}_1 = D_1^{-1}\tilde{\mathbf{b}}, \text{ i.e. } \tilde{f}_{1i} = \frac{1}{d_i}\tilde{b}_i, \text{ for } i = 1, \ldots, \tilde{\nu}, \tag{2.90}$$

which fixes $\tilde{\mathbf{f}}_1$ exactly, and leaves $\tilde{\mathbf{f}}_2$ (the remaining elements of $\tilde{\mathbf{f}}$) completely free.

When $\tilde{\nu} < \nu$ the matrix D_1, which is $\nu \times \tilde{\nu}$ and only has non-zero elements on the diagonal, must have zeros in the last $\nu - \tilde{\nu}$ rows. It can therefore be written

$$D_1 = \begin{pmatrix} \tilde{D}_1 \\ \hline \mathbf{0} \end{pmatrix} \begin{matrix} \}\tilde{\nu} \\ \}\nu - \tilde{\nu} \end{matrix} ,$$

where $\tilde{D}_1$ is $\tilde{\nu} \times \tilde{\nu}$ (*square*), diagonal and of full rank. Then, writing

$$\tilde{\mathbf{b}} = \begin{pmatrix} \tilde{\mathbf{b}}_1 \\ \hline \tilde{\mathbf{b}}_2 \end{pmatrix} \begin{matrix} \}\tilde{\nu} \\ \}\nu - \tilde{\nu} \end{matrix}$$

in equation 2.89, gives

$$D_1\tilde{\mathbf{f}}_1 \equiv \begin{pmatrix} \tilde{D}_1 \\ \hline \mathbf{0} \end{pmatrix} \tilde{\mathbf{f}}_1 = \begin{pmatrix} \tilde{D}_1\tilde{\mathbf{f}}_1 \\ \hline \mathbf{0} \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{b}}_1 \\ \hline \tilde{\mathbf{b}}_2 \end{pmatrix}.$$

It follows from this that the constraints are only consistent if $\tilde{\mathbf{b}}_2 = \mathbf{0}$ (in numerical calculation it is adequate for the elements of $\tilde{\mathbf{b}}_2$ to be no larger than round-off error might cause them to be). If the constraints are consistent (as they should be in practical situations), then

$$\tilde{\mathbf{f}}_1 = \tilde{D}_1^{-1}\tilde{\mathbf{b}}_1, \text{ i.e. } \tilde{f}_{1i} = \frac{1}{d_i}\tilde{b}_{1i}, \text{ for } i = 1, \ldots, \tilde{\nu}, \tag{2.91}$$

so equations very similar to (2.90) hold, and again the first $\tilde{\nu}$ components of $\tilde{\mathbf{f}}$ are fixed by the constraints.

So far the use of SVD has provided a new basis in terms of which the first $\tilde{\nu}$ components of $\tilde{\mathbf{f}}$ are fixed, while the remaining $n - \tilde{\nu}$ components are left free to be determined by the data (and the smoothing constraint). Application of this to the practical solution of the regularized matrix problem will now be described, the essential point being that minimization is performed only over that subspace of $\mathcal{S}$ that has $\tilde{\mathbf{f}}_1 = \tilde{D}_1^{-1}\tilde{\mathbf{b}}_1$ in the new basis. Throughout the succeeding discussion it will be tacitly assumed that (2.90) or (2.91) holds, so that $\tilde{\mathbf{f}}_1$ is fixed by the constraints. In effect, $\tilde{\mathbf{f}}_1$ will be used as a shorthand for $\tilde{D}_1^{-1}\tilde{\mathbf{b}}_1$.

Since the matrix Q in (2.87) is $n \times n$ and orthonormal it is trivially invertible. It is therefore possible to use the definition $\tilde{\mathbf{f}} = Q^T\mathbf{f}$ to write

$$\mathbf{f} = Q\tilde{\mathbf{f}} \equiv Q\left(\frac{\tilde{\mathbf{f}}_1}{\tilde{\mathbf{f}}_2}\right) \tag{2.92}$$

Using this in the discretized functional $\mathcal{R}$ defined in equation (2.80), with the knowledge that only the last $n - \tilde{\nu}$ components of $\tilde{\mathbf{f}}$ ($\tilde{\mathbf{f}}_2$ in other words) are free, allows $\mathcal{R}(\mathbf{f})$ to be rewritten as a functional, $\tilde{\mathcal{R}}$, of the new free parameters in $\tilde{\mathbf{f}}_2$:

$$\tilde{\mathcal{R}}(\tilde{\mathbf{f}}_2) \overset{\text{def}}{\equiv} \mathcal{R}(Q\tilde{\mathbf{f}}) = \mathcal{R}\left[Q\left(\frac{\tilde{\mathbf{f}}_1}{\tilde{\mathbf{f}}_2}\right)\right]. \tag{2.93}$$

This new functional is quadratic in $\tilde{\mathbf{f}}_2$ for quadratic smoothing constraints (which is the case we are principally interested in here) and so can be differentiated with respect to the $\tilde{f}_{2l}$ to give an analytic expression for $\tilde{\mathbf{f}}_2$ in terms of the data, kernel matrix and smoothing matrix, just as described in §2.2.5 for the solution without constraints.

The first stage in the derivation of the constrained solution to the minimization of $\mathcal{R}$ is, therefore, to derive the functional $\tilde{\mathcal{R}}(\tilde{\mathbf{f}}_2)$. This derivation is actually quite simple, but the need to use block matrices corresponding to the division of $\tilde{\mathbf{f}}$ into fixed and free components can give it a rather intimidating appearance. It is helpful first to expand the expression (2.80) and collect terms of the same order in $\mathbf{f}$ together, just as was done in (2.81), and then to use the relation $\mathbf{f} = Q\tilde{\mathbf{f}}$ to get

$$\mathcal{R}(\mathbf{f}) \equiv \mathcal{R}(Q\tilde{\mathbf{f}}) = g^2 - 2\mathbf{g}^T HQ\tilde{\mathbf{f}} + \tilde{\mathbf{f}}^T Q^T(H^T H + \lambda C)Q\tilde{\mathbf{f}}. \tag{2.94}$$

It is natural now to make the definitions

$$\tilde{H} = HQ \text{ and } \tilde{C} = Q^T CQ \tag{2.95}$$

so that $Q^T(H^T H + \lambda C)Q = \tilde{H}^T\tilde{H} + \lambda\tilde{C}$. This amounts to nothing more than expressing the linear mapping represented by H and the bilinear form represented by C in terms of a new basis (for the solution space) in which the basis vectors are the right singular vectors of the constraint matrix, A, in (2.86) (i.e. they are the columns of the matrix Q). The next step is to partition $\tilde{H}$ and $\tilde{C}$ in a way that corresponds to the partitioning of $\tilde{\mathbf{f}}$ into fixed and free components ($\tilde{\mathbf{f}}_1$ and $\tilde{\mathbf{f}}_2$, respectively). That is

$$\tilde{H} = (H_1|H_2) \text{ and } \tilde{C} = \left(\begin{array}{c|c} C_1 & C_3 \\ \hline C_3^T & C_2 \end{array}\right) \tag{2.96}$$

(where the tildes have been omitted from the submatrices for ease of notation). Here, H_1 is $m \times \tilde{\nu}$, H_2 is $m \times n - \tilde{\nu}$, C_1 is $\tilde{\nu} \times \tilde{\nu}$, C_2 is $n - \tilde{\nu} \times n - \tilde{\nu}$ and C_3 is $\tilde{\nu} \times n - \tilde{\nu}$. This ensures that the products of the submatrices that occur in $\tilde{H}\tilde{\mathbf{f}}$ and $\tilde{\mathbf{f}}^T(\tilde{H}^T\tilde{H} + \lambda\tilde{C})\tilde{\mathbf{f}}$ are compatible. In fact, it is easily seen that

$$\tilde{H}\tilde{\mathbf{f}} = (H_1|H_2)\left(\begin{array}{c} \tilde{\mathbf{f}}_1 \\ \hline \tilde{\mathbf{f}}_2 \end{array}\right) = H_1\tilde{\mathbf{f}}_1 + H_2\tilde{\mathbf{f}}_2, \tag{2.97}$$

and that

$$\tilde{H}^T\tilde{H} = \left(\begin{array}{c|c} H_1^T H_1 & H_1^T H_2 \\ \hline H_2^T H_1 & H_2^T H_2 \end{array}\right) \tag{2.98}$$

is partitioned into submatrices exactly corresponding to the partitioning of $\tilde{C}$. This means that $\tilde{\mathbf{f}}^T(\tilde{H}^T\tilde{H} + \lambda\tilde{C})\tilde{\mathbf{f}}$ can be expressed in block-matrix form as

$$\left(\tilde{\mathbf{f}}_1^T \,|\, \tilde{\mathbf{f}}_2^T\right)\left(\begin{array}{c|c} H_1^T H_1 + \lambda C_1 & H_1^T H_2 + \lambda C_3 \\ \hline H_2^T H_1 + \lambda C_3^T & H_2^T H_2 + \lambda C_2 \end{array}\right)\left(\begin{array}{c} \tilde{\mathbf{f}}_1 \\ \hline \tilde{\mathbf{f}}_2 \end{array}\right),$$

which, on expansion, results in the following expression in terms of the vectors $\tilde{\mathbf{f}}_1$ and $\tilde{\mathbf{f}}_2$

$$\tilde{\mathbf{f}}_1^T(H_1^T H_1 + \lambda C_1)\tilde{\mathbf{f}}_1 + 2\tilde{\mathbf{f}}_1^T(H_1^T H_2 + \lambda C_3)\tilde{\mathbf{f}}_2 + \tilde{\mathbf{f}}_2^T(H_2^T H_2 + \lambda C_2)\tilde{\mathbf{f}}_2. \tag{2.99}$$

Using this and (2.97) in equation (2.94) and again collecting terms of the same order in $\tilde{\mathbf{f}}_2$ gives the sought for definition of $\tilde{\mathcal{R}}(\tilde{\mathbf{f}}_2)$:

$$\begin{aligned}\tilde{\mathcal{R}}(\tilde{\mathbf{f}}_2) &= \tilde{\mathbf{f}}_2^T(H_2^T H_2 + \lambda C_2)\tilde{\mathbf{f}}_2 - 2[\mathbf{g}^T H_2 - \tilde{\mathbf{f}}_1^T(H_1^T H_2 + \lambda C_3)]\tilde{\mathbf{f}}_2 \\ &\quad + g^2 - 2\mathbf{g}^T H_1\tilde{\mathbf{f}}_1 + \tilde{\mathbf{f}}_1^T(H_1^T H_1 + \lambda C_1)\tilde{\mathbf{f}}_1. \end{aligned} \tag{2.100}$$

(Note that the second line of this equation contains only terms that are independent of $\tilde{\mathbf{f}}_2$, and so do not affect the result of the minimization over $\tilde{\mathbf{f}}_2$.)

Taking the derivative of $\tilde{\mathcal{R}}$ with respect to the components of $\tilde{\mathbf{f}}_2$ and setting the result to zero in the usual way (see §2.2.5) gives

$$(H_2^T H_2 + \lambda C_2)\tilde{\mathbf{f}}_2 = H_2^T \mathbf{g} - (H_2^T H_1 + \lambda C_3^T)\tilde{\mathbf{f}}_1. \tag{2.101}$$

Although solving (2.101) to obtain the expression for the solution that appears in (2.102) below seems the inevitable next step, there is another approach which may have advantages in some circumstances. In examining these alternatives it is neccessary to consider some of the practical aspects of obtaining such solutions by numerical procedures.

There is some freedom in the choice of the analytic expression for the solution of (2.101). It is possible to follow the simplest route, which is identical to the derivation of the solution without constraints given in §2.2.5: calculate the inverse of the matrix $H_2^T H_2 + \lambda C_2$ and define the solution by

$$\tilde{\mathbf{f}}_2 = (H_2^T H_2 + \lambda C_2)^{-1}[H_2^T \mathbf{g} - (H_2^T H_1 + \lambda C_3^T)\tilde{\mathbf{f}}_1]. \tag{2.102}$$

(Equation (2.102) determines $\tilde{\mathbf{f}}_2$ and the constraints fix $\tilde{\mathbf{f}}_1$, so $\tilde{\mathbf{f}}$ is known. The desired solution then follows from $\mathbf{f} = Q\tilde{\mathbf{f}}$.) This has the drawback, though, that the matrix $(H_2^T H_2 + \lambda C_2)^{-1}$ depends on the matrix Q in (2.87) – through the definitions (2.95) – and hence on the set of constraints being used. Whenever new constraints on the solution are applied, a complete recomputation of the matrix inverse will be required, which entails considerable computational expense. If only one set of constraints is to be applied then this would be the method of choice, but if several sets of constraints (including, perhaps, no constraints at all) are to be considered, it would be more satisfactory to find an expression for the solution for which changing the constraints involves a smaller computational burden. It is possible to derive such an expression, but its implementation for a single set of constraints introduces greater complexity and some cost in terms of efficiency. Which form is appropriate for any inversion depends on the individual circumstances, and in particular on the number of independent constraints, $\tilde{\nu}$, and the number of different sets of constraints being applied.

Derivation of the second expression for the solution begins with the simple observation that (2.101) is actually a consequence of the equation

$$\left(\begin{array}{c|c} H_1^T H_1 + \lambda C_1 & H_1^T H_2 + \lambda C_3 \\ \hline H_2^T H_1 + \lambda C_3^T & H_2^T H_2 + \lambda C_2 \end{array}\right) \left(\begin{array}{c} \tilde{\mathbf{f}}_1 \\ \hline \tilde{\mathbf{f}}_2 \end{array}\right) = \left(\begin{array}{c} H_1^T \mathbf{g} + \mathbf{v} \\ \hline H_2^T \mathbf{g} \end{array}\right), \tag{2.103}$$

where $H_1^T\mathbf{g} + \mathbf{v}$ is some (as yet unknown) $\tilde{\nu}$-vector: it turns out to be convenient later to include the $H_1^T\mathbf{g}$ term at this stage. In fact, expansion of (2.103) gives

$$(H_1^T H_1 + \lambda C_1)\tilde{\mathbf{f}}_1 + (H_1^T H_2 + \lambda C_3)\tilde{\mathbf{f}}_2 = H_1^T\mathbf{g} + \mathbf{v}, \tag{2.104}$$

$$(H_2^T H_1 + \lambda C_3^T)\tilde{\mathbf{f}}_1 + (H_2^T H_2 + \lambda C_2)\tilde{\mathbf{f}}_2 = H_2^T\mathbf{g}. \tag{2.105}$$

The second of these equations is easily seen to be equivalent to (2.101). There is still equation (2.104) to consider, though. Obviously, since $\tilde{\mathbf{f}}_1$ is fixed by the constraints and $\tilde{\mathbf{f}}_2$ is determined by (2.105) the left hand side of (2.104) is given. We therefore regard (2.104) as a *definition* of $\mathbf{v}$. How does this help? Well, the left hand side of (2.103) is just $(\tilde{H}^T\tilde{H} + \lambda\tilde{C})\tilde{\mathbf{f}}$, so the solution is

$$\tilde{\mathbf{f}} = (\tilde{H}^T\tilde{H} + \lambda\tilde{C})^{-1}\left[\tilde{H}^T\mathbf{g} + \left(\begin{array}{c} \mathbf{v} \\ \hline \mathbf{0} \end{array}\right)\right]. \tag{2.106}$$

The advantage of this is that, since $\tilde{H}^T\tilde{H} + \lambda\tilde{C}$ is just $Q^T(H^TH + \lambda C)Q$ (by virtue of the definitions (2.95)) and Q is an orthonormal matrix (containing the right singular vectors of the constraint matrix A), the inverse matrix can be written

$$(\tilde{H}^T\tilde{H} + \lambda\tilde{C})^{-1} = Q^T(H^TH + \lambda C)^{-1}Q. \tag{2.107}$$

The matrix inverse on the right hand side of (2.107) is the inverse that must be calculated to solve the unconstrained problem: it is entirely independent of the constraints. This means that, once $(H^TH + \lambda C)^{-1}$ has been found, the calculation of $(\tilde{H}^T\tilde{H} + \lambda\tilde{C})^{-1}$ for *any* set of constraints is achieved by multiplying $(H^TH + \lambda C)^{-1}$ by the matrix Q for those constraints. In fact, it is not actually necessary in practice to perform this matrix multiplication. The quantity that we really want to find is $\mathbf{f} = Q\tilde{\mathbf{f}}$. Using this and (2.107) in equation (2.106) gives

$$\mathbf{f} = (H^TH + \lambda C)^{-1}H^T\mathbf{g} + (H^TH + \lambda C)^{-1}Q\left(\begin{array}{c} \mathbf{v} \\ \hline \mathbf{0} \end{array}\right). \tag{2.108}$$

(The definition of $\tilde{H}$ in (2.95) tells us that $Q\tilde{H}^T = QQ^TH^T = H^T$, since Q is orthonormal.) The first term on the right hand side of (2.108) is just the solution to the problem *without* constraints. The other term is therefore a 'correction' which ensures that the constraints are obeyed.

While the preceding discussion appears to provide a very satisfactory solution to the problem, the calculation of the vector $\mathbf{v}$ has not been addressed. This is the main drawback in the practical implementation of this procedure. The fact that $\mathbf{v}$ is a $\tilde{\nu}$-vector indicates that it is the constraints imposed on the solution determine $\mathbf{v}$, so it is easiest to derive the expression for $\mathbf{v}$ in the basis in which the solution vector is split into fixed and free components, $\tilde{\mathbf{f}}_1$ and $\tilde{\mathbf{f}}_2$, respectively. Ultimately, we should expect to obtain an expression relating $\mathbf{v}$ to the $\tilde{\nu}$-vector $\tilde{\mathbf{f}}_1$ (which has itself been fixed by the constraints – see (2.90) and (2.91)).

It is necessary at this point to partition the matrix Q into $(Q_1|Q_2)$, corresponding to the partitioning of $\tilde{\mathbf{f}}$ (this amounts to dividing the right singular vectors of the constraint matrix – which form the columns of Q – into those whose contribution to the solution is fixed by the constraints, and those which span the 'free' part of the solution space). The advantage of introducing this partition is that, since $\tilde{\mathbf{f}} = Q^T\mathbf{f}$ implies

$$\tilde{\mathbf{f}}_1 = Q_1^T\mathbf{f}, \tag{2.109}$$

it is possible to use definition the expression for $\mathbf{f}$ given in (2.108) to obtain the desired relationship between $\mathbf{v}$ and $\tilde{\mathbf{f}}_1$. In fact, observing that the last term in (2.108) is actually $(H^TH + \lambda C)^{-1}Q_1\mathbf{v}$, we get from(2.108) and (2.109)

$$\tilde{\mathbf{f}}_1 = Q_1^T(H^TH + \lambda C)^{-1}H^T\mathbf{g} + Q_1^T(H^TH + \lambda C)^{-1}Q_1\mathbf{v}. \tag{2.110}$$

The matrix $Q_1^T(H^TH + \lambda C)^{-1}Q_1$ appearing on the right hand side of (2.110) is a $\tilde{\nu} \times \tilde{\nu}$, square, symmetric positive definite matrix. It can be inverted to give

$$\mathbf{v} = [Q_1^T(H^TH + \lambda C)^{-1}Q_1]^{-1}[\tilde{\mathbf{f}}_1 - Q_1^T(H^TH + \lambda C)^{-1}H^T\mathbf{g}]. \tag{2.111}$$

It may be possible that, for a given set of constraints, there is a way to reduce the matrix $Q_1^T(H^TH + \lambda C)^{-1}Q_1$ to a form that can be easily inverted for many different values of λ. For example, by using the decomposition $(H^TH + \lambda C)^{-1} = P^{-T}M^{-1}(\lambda)P^{-1}$ described in §2.4 we may be able to derive a formula something like

$$[Q_1^T(H^TH + \lambda C)^{-1}Q_1]^{-1} = \tilde{Q}^T\tilde{M}^{-1}(\lambda)\tilde{Q}$$

where $\tilde{Q}$ is independent of λ, and $\tilde{M}$ is diagonal (or at least trivially invertible). This would make the evaluation of $\mathbf{v}$, and subsequently the solution, for different λ, very fast. If this were the case, then the second method for calculating the constrained solution would be preferable in almost all cases (only when a single *large* set of constraints is to be applied would the advantages of inverting the smaller matrix $H_2^T H_2 + \lambda C_2$ be significant). However, at the moment the need to explicitly evaluate and then invert $Q_1^T (H^T H + \lambda C)^{-1} Q_1$ makes this method unappealing if a large number of constraints are being used (the time taken for the calculation of the inverse scales roughly as $\tilde{\nu}^3$). In any event, once (2.111) has been used to evaluate $\mathbf{v}$, the result can be used in (2.108) to obtain the solution vector $\mathbf{f}$.

To conclude this section it may be helpful to summarize the results obtained. Solving the problem of minimizing the discretized functional $\mathcal{R}(\mathbf{f})$ in (2.80) subject to the set of constraints (2.85) or (2.86) proceeds by changing to the basis for the solution space in which the constraint matrix becomes diagonal (so that the transformed solution vector breaks up into the fixed and free parts $\tilde{\mathbf{f}}_1$ and $\tilde{\mathbf{f}}_2$), and then using the simplification gained to derive the functional $\tilde{\mathcal{R}}(\tilde{\mathbf{f}}_2)$, given in (2.100). The solution can be written in either of two forms. The simplest expression

$$\tilde{\mathbf{f}}_2 = (H_2^T H_2 + \lambda C_2)^{-1} [H_2^T \mathbf{g} - (H_2^T H_1 + \lambda C_3^T)\tilde{\mathbf{f}}_1] \qquad (2.102)$$

has the advantages that it is easy to understand and that $H_2^T H_2 + \lambda C_2$ is a $n - \tilde{\nu} \times n - \tilde{\nu}$ matrix, so calculating its inverse will (for a single set of constraints) be faster than the inversion of $H^T H + \lambda C$ (which is $n \times n$) required by the second method. The need to completely recalculate $(H_2^T H_2 + \lambda C_2)^{-1}$ for each new set of constraints penalizes against the use of this form of the solution when several sets of constraints are to be used. It should be borne in mind that when, as is often the case, the smoothing matrix, C, is diagonal, the change of basis effected by the matrix Q will generally result in a matrix $\tilde{C} = Q^T C Q$ that is full. This will make the calculation of $(H_2^T H_2 + \lambda C_2)^{-1}$ slower and more prone to round-off error, because the first stage in the calculation of $(H_2^T H_2 + \lambda C_2)^{-1}$ is the diagonalization of C_2 (see §2.4).

The second form of the solution is obtained from,

$$\mathbf{f} = (H^T H + \lambda C)^{-1} [H^T \mathbf{g} + Q_1 \mathbf{v}] \qquad (2.108')$$

(where (2.108) has been simplified slightly by using the partitioning of Q into $(Q_1|Q_2)$ in the second term and factoring out $(H^TH + \lambda C)^{-1}$), with $\mathbf{v}$ given by

$$\mathbf{v} = [Q_1^T(H^TH + \lambda C)^{-1}Q_1]^{-1}[\tilde{\mathbf{f}}_1 - Q_1^T(H^TH + \lambda C)^{-1}H^T\mathbf{g}]. \qquad (2.111)$$

(Remember that $\tilde{\mathbf{f}}_1$ has been fixed by the constraints in (2.90) or (2.91)). This form is more complicated, and requires inversion of the (constraint and smoothing parameter dependent) $\tilde{\nu} \times \tilde{\nu}$ matrix $Q_1^T(H^TH + \lambda C)^{-1}Q_1$ for each value of the smoothing parameter, λ. For a relatively large number of constraints, $\tilde{\nu}$, the time taken to perform this calculation would be prohibitive (see the discussion in §2.4 about the need to perform the calculations for many different values of λ when automatic methods for choosing the smoothing parameter are being used). However, the effect of changing the set of constraints applied enters only in the determination of $\mathbf{v}$ (through the submatrix Q_1 in (2.111)), which only requires the inversion of a $\tilde{\nu} \times \tilde{\nu}$ matrix.

Broadly speaking, then, the first method is preferable when $\tilde{\nu}$ is large and there is only one set of constraints to be used, whereas the second is better when there are several sets of constraints to be applied, especially when the number of constraints in each set is small.

2.4 Fix-Heiberger Reduction of $H^TH + \lambda C$

An absolutely essential part of any method for solving linear inverse problems of the type discussed in this thesis (including the optimal averaging methods) is the numerical calculation of the inverses of matrices of the form $A + \lambda B$, where A and B are symmetric, positive *semi*-definite matrices, and λ is a parameter which can be varied to change the level of regularization applied to the solution (see, for example, equations (2.82), (2.102) and (2.108′), and §1.8). If it was just a case of choosing a single value of λ for any problem and finding the solution for that value alone it would be a simple matter to calculate the inverse: add λB to A and calculate the inverse of this by the numerical method of your choice. However, this is a very time consuming process for matrices of the size commonly occurring in helioseismology ($n \times n$ matrices, where n can be anywhere from around a hundred to several thousand), and would take just too long if it were necessary to determine the solution for several different values of λ, as is the case when automatic methods for choosing the smoothing parameter (see §1.9.6) are to be used. For example, with the GCV and EDF methods discussed in §1.9.6 and studied in chapter 3 it is necessary to calculate $(H^TH + \lambda C)^{-1}$ for many values of λ, because choosing the value of the smoothing parameter involves minimizing a function of λ involving $(H^TH+\lambda C)^{-1}$ (for GCV) or finding the root of a similar function (for EDF), and this typically requires several tens of function evaluations for different values of λ. For this to be achieved in a reasonable time it is really essential to find a method that makes the calculation of the matrix inverse in these function evaluations almost trivial. Of course, it is not possible to find an algorithm for inverting $(H^TH+\lambda C)^{-1}$ that is actually faster overall than the standard methods for a single value of λ; the goal is to perform, in some sense, as much of the computation of the inverse as possible in a λ-independent way, leaving a very simple calculation (perhaps equivalent to a matrix-vector multiplication, or even a scalar product) to be performed for each value of λ to complete the inversion. The bulk of the work is then done just once, and the simplicity of the remaining calculation, even though it must be performed repetitively for different λ, means that choosing the smoothing parameter is a fast and relatively painless exercise. In fact, it is possible to reduce the computation of the GCV and EDF functions to something approaching the speed of a scalar product, which is very fast compared to the speed of explicit inversion. As this chapter deals with RLS methods, where the matrix $(H^TH+\lambda C)^{-1}$ frequently appears, the following discussion will describe the reduction of this matrix, but the formalism can be applied to any pair of positive semi-definite matrices.

Experience shows that the way to proceed must be to use a sequence of transformations of $H^TH+\lambda C$ (changes of basis, in linear algebra terminology) that eventually reduce it to a form that can be inverted with very little effort. The symmetry of $H^TH+\lambda C$ strongly suggests that the transformations we should seek are congruence transformations, which preserve symmetry, rather than similarity transformations, which destroy it. (A congruence transformation of some symmetric matrix A is performed by finding some non-singular square matrix, Q say, and evaluating $\tilde{A} \overset{\text{def}}{\equiv} Q^TAQ$. In fact, it is more convenient for what follows to write this as $A = P\tilde{A}P^T$, where $P = Q^{-T}$.) Furthermore, the simplest matrices to have to invert are diagonal matrices, because their inverses are just given by taking the reciprocals of all the diagonal elements, and matrix multiplication with them reduces to a simple row-by-row (or column-by-column) rescaling, which is much quicker than multiplying two full matrices. Our mission, then, is to find a sequence of λ-independent congruence transformations of $H^TH+\lambda C$ that reduce it to a diagonal matrix (which will depend on λ). Anyone familiar with linear algebra will recognize this as the *simultaneous diagonalization of two quadratic forms*, a process that involves finding a basis for the vector space on which the symmetric matrices act (the solution space, $\mathcal{S}$, in this case) in which *both* matrices H^TH and C are diagonal.

In cases where the smoothing matrix, C, is non-singular, the procedure for diagonalizing $H^TH + \lambda C$ is well known (Christensen-Dalsgaard *et al.* 1993). First, find the 'square root' of C, which is any square matrix P_1 (non-singular, because C is) such that $C = P_1P_1^T$. P_1 is not unique, and there are two alternative methods for finding an appropriate matrix. *Cholesky decomposition* (See *Numerical Recipes*, §2.9) determines a P_1 that is a *lower triangular* matrix. Alternatively, we can diagonalize C, that is, find its eigenvalues and orthonormal eigenvectors (which exist by virtue of the symmetry of C). It is well known that we can then create an orthonormal matrix, U, whose columns are the eigenvectors of C, and a diagonal matrix D, whose ith diagonal element is the eigenvalue corresponding to the eigenvector stored in the ith column of U, such that

$$C = UDU^T = (UD^{1/2})(UD^{1/2})^T. \tag{2.112}$$

(As D is a diagonal matrix, $D^{1/2}$ just denotes the diagonal matrix whose diagonal elements are the square roots of the diagonal elements of D. This, of course, gives $D^{1/2}D^{1/2} = D$ in (2.112).) The diagonal elements of D, being the eigenvalues of the real, symmetric positive-*definite* matrix C (C being non-singular) are real and strictly positive, so $D^{1/2}$ exists, and is non-singular. The last equality in (2.112) shows that it is natural to make the definition $P_1 = UD^{1/2}$. Having calculated some acceptable P_1, the next step is to factor out P_1 from $H^TH + \lambda C$ as follows. As P_1 is invertible, we can use the trivial and obvious identity $H^TH \equiv P_1(P_1^{-1}H^THP_1^{-T})P_1^T$ to get

$$H^TH + \lambda C = P_1(P_1^{-1}H^THP_1^{-T} + \lambda I)P_1^T. \tag{2.113}$$

Finally, the symmetric matrix $P_1^{-1}H^THP_1^{-T}$ can be diagonalized just as C was, to give $P_1^{-1}H^THP_1^{-T} = V\Delta V^T$, where V is orthonormal and Δ is diagonal. Then

$$H^TH + \lambda C = P_1V(\Delta + \lambda I)V^TP_1^T. \tag{2.114}$$

The matrix P_1V is non-singular and independent of λ, so the desired transformation has been achieved, and if we define $P = (P_1V)^{-T}$ we eventually arrive at

$$(H^TH + \lambda C)^{-1} = P(\Delta + \lambda I)^{-1}P^T. \tag{2.115}$$

If we chose $P_1 = UD^{1/2}$ then, by virtue of the orthonormality of U and V, $P = P_1^{-T}V = (UD^{-1/2})V$, which is obviously very easy to evaluate. Note, too, that it is precisely the matrix $P_1^{-T} = UD^{-1/2}$ that appears in (2.113), so it is only this matrix that needs to be computed and stored in the calculation of $(H^TH + \lambda C)^{-1}$.

The only requirement for the algorithm just described to work is that one of the matrices H^TH or C be positive definite. (There is no reason, in principle, why we could not have performed the reduction with the roles of H^TH and C interchanged, first diagonalizing H^TH. In practice, however H^TH is generally highly singular, so there is little reason to consider this.) In many situations the smoothing matrix is non-singular and well-conditioned for inversion, and the reduction given is perfectly adequate. Unfortunately, in general both H^TH and C are singular matrices, and the algorithm will fail. This will be the case, for example, when pth derivative (pth order) smoothing is used, for $p \geq 1$. Then the smoothing matrix should have p zero eigenvalues, because there ought to be p independent vectors that can be added to the solution vector, $\mathbf{f}$ (see §2.2) without changing the value of the smoothing functional $\mathbf{f}^TC\mathbf{f}$ (these correspond to the polynomials $1, x, \ldots, x^{p-1}$ which are annihilated by the pth derivative operator). When both matrices are singular, therefore, another approach is needed. Three possible solutions to this problem will now be presented, each of which has its merits, but the last is without doubt the most robust and effective.

I. The simplest way to remedy the singularity of C is to cheat. The eigenvalues, $\{d_1, \ldots, d_n\}$, of C are positive or zero (C is positive semi-definite). The eigenvalues of $C + \alpha I$, for any α are then all $\geq \alpha$. Choosing some positive α large enough to ensure that $C + \alpha I$ is non-singular and well-conditioned (i.e. choosing α much larger than the typical numerical error on any eigenvalues computed), but small enough to leave the original smoothing matrix dominating the smoothing will allow the reduction for non-singular smoothing matrices to be used on the amended matrix $H^TH + \lambda(C + \alpha I)$, while affecting the results of the inversion (hopefully) only slightly. This trick is often useful for checking implementations of the next two methods to be described, to see whether they really deal with the singular eigenvalues of C correctly (results should be very similar, differing only through the effect of the small contribution from αI, which will tend to make the entries in the solution vector, $\mathbf{f}$, slightly smaller – the new effective smoothing constraint in the discretized problem is $\mathbf{f}^T(C + \alpha I)\mathbf{f} = \mathbf{f}^T C\mathbf{f} + \alpha||\mathbf{f}||^2$, which is the old smoothing functional plus a term that penalizes against solution vectors with larger norms). However, such underhand techniques go completely against the philosophy of this chapter, where the algorithm described in §2.2 was devised largely to ensure that the effects of regularization are controllable, predictable and independent of the discretization. The small effect of αI will have a predictable effect on the solution *vector* (tending to make its components slightly smaller), but the translation of this effect to the solution function via (2.21) (or, equivalently, (2.23)) will be harder to understand, and will depend on the discretization (through the ϕ_j^i in (2.21), which vary with the discretization chosen).

II. Once it has been recognized that the smoothing matrix is singular, and that method I is unsatisfactory, it is natural to look for a method that, like I, modifies C in such a way as to remove the zero eigenvalues, but improves upon I by using a modification that does not change the matrix $H^TH + \lambda C$ overall: in other words, if you add the matrix αA to C to make it non-singular, you must subtract $\lambda\alpha A$ from H^TH. It is convenient for several reasons to retain the (redundant) factor α: discussing the choice of an appropriate value for α will be deferred for the moment. The matrix A must be chosen judiciously, for it is absolutely essential, when diagonalizing $H^TH + \lambda C$, to use transformations that are independent of λ (otherwise the transformations must be performed separately for each λ, losing all of the benefits in terms of efficiency of performing the transformations in the first place). To see where an unfortunate choice for A would spoil the procedure given above for diagonalizing $H^TH + \lambda C$ when C is non-singular, repeat that procedure on the pair of matrices $H^TH - \alpha\lambda A$ and $\tilde{C} = C + \alpha A$. The first step, writing $\tilde{C} = P_1P_1^T$ and removing the P_1 factors, proceeds just as before, but now equation (2.113) reads

$$(H^TH - \alpha\lambda A) + \lambda\tilde{C} = P_1[P_1^{-1}(H^TH - \alpha\lambda A)P_1^{-T} + \lambda I]P_1^T.$$

Before, we continued by diagonalizing $P_1^{-1}(H^TH - \alpha\lambda A)P_1^{-T}$, but now this matrix depends on λ, and, in general, the orthonormal matrix, V that diagonalizes it will too. However, if it is possible to find a form for A that allows the occurrence of λ to be taken out as a common factor to all elements of the matrix, then V will again be independent of the smoothing parameter and we can proceed basically as before. It ought to be obvious that putting $A = H^T H$ achieves the desired result, giving

$$P_1^{-1}(H^T H - \alpha\lambda A)P_1^{-T} = (1-\alpha\lambda)P_1^{-1}H^T H P_1^{-T} = (1-\alpha\lambda)V\Delta V^T$$

(using the diagonalization of $P_1^{-1}H^T H P_1^{-T}$ performed before). Completing the procedure and as before defining $P = P_1^{-T}V$ gives the solution

$$(H^T H + \lambda C)^{-1} = P[(1-\alpha\lambda)\Delta + \lambda I]^{-1}P^T, \tag{2.115$'$}$$

which differs from (2.115) only in the appearance of the $(1-\alpha\lambda)$ factor. The result is again an expression for $(H^T H + \lambda C)^{-1}$ in which it is only necessary to invert a diagonal matrix. There are a couple of important points that have been skated over, though. Firstly, it is not obvious that the amended matrix $\tilde{C} = C + \alpha H^T H$ *will* be non-singular. In fact, $\tilde{C}$ will be non-singular for all $\alpha > 0$ if and only if $H^T H + \lambda C$ is invertible for *any* $\lambda > 0$ (this result follows from the positive semi-definiteness of $H^T H$ and C). So if $H^T H + \lambda C$ is invertible, this method can be used to invert it. Even if $H^T H + \lambda C$ is singular it will be possible to 'invert' it using singular value decomposition (see *Numerical Recipes*, §2.6), and this is addressed in method III. Secondly, what is the best value of α to use? It is difficult to give hard and fast rules about this, but the aim is to find a value that makes $\tilde{C}$ as well-conditioned as possible. This means that we must ensure that the smallest eigenvalue of $\tilde{C} = C + \alpha H^T H$ is considerably larger than the maximum round-off error in the calculation of the eigenvalues. Actually, the range of α that will be satisfactory is generally very wide, and a little experimenting should allow a suitable value to be found.

III. Fix-Heiberger Reduction. Method II is often sufficient for diagonalizing the matrices that occur in the RLS inversion of (2.5). However, it does have (at least) two serious shortcomings. These problems will be discussed and exemplified, to indicate the need for an algorithm that can diagonalize and 'invert' $H^T H + \lambda C$ even when it is singular (or nearly singular). Following this discussion, the Fix-Heiberger algorithm will be explained.

As was mentioned above, if $H^T H + \lambda C$ is singular, then there will be *no* value of α for which the matrix $\tilde{C} = C + \alpha H^T H$ in method II is non-singular. Method II will not work for such cases. This is a serious obstacle to the effective implementation of the the algorithm presented in §2.2, because some commonly used discretizations can give rise to $H^T H + \lambda C$ matrices that are singular. For example, if the solution function is expanded as a polynomial on some discretization bin, with the basis functions ϕ_j^i on that bin taken to be the polynomials $1, y, \ldots, y^{n_i}$ (remember that, from (2.76) and (2.77), y is the 'standard' coordinate on the discretization bin), then the smoothing matrix will contain a block whose properties resemble those of Vandermonde matrices (see *Numerical Recipes*, §2.8). Vandermonde matrices are notoriously ill-conditioned, and the smoothing matrix will tend to share this poor conditioning. When such a discretization was used with zeroth order smoothing in the inversions studied in chapter 5, it was found that the smoothing matrix (which ought to be non-singular for zeroth order smoothing) had a spectrum of eigenvalues which were roughly equally spaced in logarithm, and soon dipped into the region where they were zero to within the accuracy of the eigenvalue routine. This very poor conditioning means that there can be many components of the solution that are not constrained at all by the regularization, and this will often include components that are also in the null space of H^TH (i.e. eigenvectors corresponding to zero eigenvalues of H^TH): the null spaces of H^TH and C overlap. $H^TH + \lambda C$ will then be singular and method II will fail. While this seems reasonable (we do not expect to be able to invert singular matrices), it is often possible to gain useful information in such cases by finding a *pseudo-inverse* of the matrix. If the formalism of §2.2 is to be applied successfully, then, we need to find a method for solving the inverse problem that can cope with such cases. Truncated SVD is such a method, and the reduction of $H^TH + \lambda C$ given below allows truncated SVD to be used very easily.

To understand what is meant by truncated SVD, consider a singular, symmetric, $n \times n$ matrix, M, which can be diagonalized as usual to get $M = UDU^T$, with U orthonormal and D diagonal. The diagonal elements, $\{d_1, \ldots, d_n\}$, of D are the eigenvalues of M, of which some are zero, because M is singular. Normally, if M were non-singular, its inverse would be given by

$$M^{-1} = UD^{-1}U^T$$

where D^{-1} is the diagonal matrix whose diagonal elements are just the reciprocals of the corresponding elements of D. This is not possible for M because some of those elements are zero. In numerical calculation there will be a number of eigenvalues that, while not actually zero, are small enough to be effectively zero to the level of accuracy obtained. Some of these may correspond to 'true' zero eigenvalues, whereas others may just be indistinguishable from zero due to round-off. For example, with pth order smoothing for $p \gtrsim 4$ the smoothing matrix tends to have more than p very small eigenvalues, only p of which can correspond to true zero eigenvalues. In fact, there may be eigenvalues which, although they are clearly not zero, are sufficiently small to make the matrix very ill-conditioned (see §1.4). To use the inverse matrix in the solution to an inverse problem would then be very difficult and give rise to unstable inversions. It is advisable, therefore, to treat these eigenvalues as though they were zero, and accord them the same respect (not to say trepidation) that would be given to truly zero eigenvalues. An important part of the following method is the specification of a criterion for determining when eigenvalues are too small for comfort and should be treated as zero eigenvalues. For the moment we will just assume that some ε has been chosen such that eigenvalues are effectively zero if they are smaller than ε. This ε can be as large as is deemed necessary to ameliorate the adverse effects of the small eigenvalues that give rise to ill-conditioning. Returning to the matrix M, and its eigenvalues $\{d_1, \ldots, d_n\}$, assume that the eigenvalues have been arranged in descending order. M is singular, so certainly some (the last s, say) of its eigenvalues will be less than ε. Set these eigenvalues to zero, if they are not already zero: that is, *truncate* the eigenvalue spectrum at ε. Surely this has made things worse: we now have *more* zero eigenvalues. Well, the outstanding deviousness to come turns this apparent disaster into a positive boon. Define the pseudo-inverse of D to be the diagonal matrix $\tilde{D}$ whose diagonal elements are $1/d_1, 1/d_2, \ldots, 1/d_{n-s}, 0, \ldots, 0$. In other words, instead of taking the reciprocal of the zero eigenvalues (giving infinity, and the corresponding bad behaviour) we *replace this infinity with zero*, thus removing all the problems in one fell swoop. Of course, the pseudo-inverse of M is then defined to be

$$\tilde{M} = U\tilde{D}U^T.$$

This is then perfectly well defined and (providing ε has been chosen properly) can be used to give stable and meaningful solutions to matrix problems of the form $M\mathbf{x} = \mathbf{y}$, which is essentially why we want to use it. Note that $\tilde{M}$ is not a true inverse because both M and $\tilde{M}$ are singular, so $M\tilde{M}$ cannot be the identity (which is not singular).

The second flaw in method II is more technical in nature. Those familiar with linear algebra and matrix methods will recognize the very close relationship between the diagonalization of $H^TH + \lambda C$ and solving the *generalized symmetric eigenvalue problem* (see Parlett 1980, chapter 15 for a clear description of this topic) for the pair (H^TH, C), which means finding the set of eigenvalues μ that satisfy

$$\det(H^T H - \mu C) = 0 \tag{2.116}$$

and their corresponding eigenvectors. Here we will concentrate on the eigenvalues, because that is where most of the problems lie. (When $C = I$, (2.116) reduces to the standard eigenvalue problem.) Such problems are known to be quite difficult to solve numerically, and can provide some rather unpleasant surprises. For example, the pair (actually, the usual term is *pencil*) $(H^T H, C)$ can, in general, have infinite eigenvalues, eigenvalues that are simply undetermined (in the way that 0/0 is meaningless), and if we relaxed the constraint of positive semi-definiteness we could even find complex eigenvalues. Here, though, the last difficulty will not apply. When both $H^T H$ and C are strictly positive definite and well-conditioned, these problems do not occur. The explicit reduction of $H^T H + \lambda C$ given earlier in this section for non-singular $H^T H$ and C reduces (2.116) to

$$\det(\Delta - \mu I) = 0$$

(using (2.114) and the fact that $P_1 V$ is non-singular, so $\det(P_1 V) \neq 0$). Obviously the eigenvalues of $H^T H + \lambda C$ must be just the diagonal elements of Δ. When both matrices are well-conditioned there is no problem with this and the eigenvalues and eigenvectors are well determined. Unfortunately, it is precisely when $H^T H$ and C are singular or ill-conditioned that we want to study this problem.

When C is singular, infinite eigenvalues can arise, but if $H^T H$ and C have null spaces that overlap (so that $H^T H + \lambda C$ is singular) undetermined eigenvalues occur. In numerical work, as usual, these statements are blurred by the effects of round-off error. For example, if $H^T H$ and C nearly share a null eigenvector, $H^T H + \lambda C$ will be almost singular. It will then have an eigenvalue which, while not actually undetermined, is *highly* sensitive to perturbations in the matrices or errors in the calculations performed to find the eigenvalues (round-off errors). Such awkward eigenvalues are called *ill-disposed*, and their presence can, if they are not properly treated, affect the accuracy with which the other eigenvalues can be found.

The explicit reduction of $H^T H$ and C to diagonal form (either directly or using method II) cannot cope with ill-disposed eigenvalues, and this can have important consequences. To exemplify this, consider minimizing GCV to choose the smoothing parameter (see §1.9.6). As was discussed at the beginning of this section, the goal of simultaneously diagonalizing $H^T H$ and C is to leave a very simple and fast diagonal matrix inversion to speed up the calculation of $(H^T H + \lambda C)^{-1}$, which occurs (twice) in the GCV function. The diagonal matrix (found from the explicit reduction given, assuming $H^T H$ and C are non-singular) that must be inverted is $\Delta + \lambda I$, where Δ is contains the generalized eigenvalues, $\{\delta_1, \ldots, \delta_n\}$, of the pencil $(H^T H, C)$. If one of these eigenvalues (δ_n, say) is ill-disposed it can have quite a large negative value purely as a result of the numerical error in the calculation (and despite the fact that $H^T H + \lambda C$ is positive semi-definite). Quite large here means much larger than the typical effects of round-off error on finding the eigenvalues of a single symmetric matrix (which may be around, say, 100 times the machine precision), and certainly large enough to take $|\delta_n|$ into the range of values of λ that might be searched to find the minimum of GCV (the search region will generally exclude very small λ because of the effect of numerical error on the smallest eigenvalues). To quote a number from personal experience, it can happen that $\delta_n \approx -10^{-9}$ when the largest error that might have been expected was around 10^{-14}. The effect of this negative eigenvalue on the minimization of the GCV function is catastrophic. Evaluating GCV for $\lambda \approx |\delta_n|$ will involve inverting $\Delta + \lambda I$, which is now virtually singular. The calculation will be highly unreliable and will tend to return anomalous values. Experience shows that the graph of the GCV function can acquire a downward-pointing spike around $\lambda = |\delta_n|$. This spike will be a minimum of GCV, and usually it will be *the* minimum. The value of the smoothing parameter chosen by GCV will then be $|\delta_n|$, when in fact the best value is likely to be that corresponding to the position of another true minimum at larger λ. The ill-disposed eigenvalue has resulted in a very poor choice of smoothing parameter, one which drastically undersmooths. The same behaviour can afflict the EDF method for choosing the smoothing parameter.

This chapter presents an algorithm that allows quite general discretizations of the integrals occurring in the inverse problems of helioseismology. Another part of the work in this thesis deals with automatic methods for setting the regularization levels used in RLS inversions (chapter 3). The ethos behind both of these ideas is to make the process of inversion as 'hands-off' as possible. It is recognized, of course, that inverse problems of the type dealt with in this thesis are so widespread, so varied and so difficult that some intervention is almost invariably necessary in any serious inversion. However, by creating greater scope for using a wider range of discretizations (for example) from within a single algorithm, the need to tinker with various different methods, and the extent of that tinkering, should be reduced. Apart from the obvious benefit of making inverse methods more available to, and usable by, non-experts, perhaps the biggest advantage of such automation is that it increases the ease with which different inversion methods can be compared for their effectiveness. If two different inversion methods run using the same algorithm produce results of markedly different quality, this must certainly be a reflection of their relative merits for the problem at hand, rather than, perhaps, because the implementation of one of those methods was somewhat indifferent.

The use of automatic methods for choosing the smoothing parameter demands that we have a procedure for simultaneously diagonalizing two symmetric matrices. To fully achieve the kind of freedom to choose different discretizations that is made possible by the algorithm presented in §2.2 requires a numerical method for performing the diagonalization that is very robust and can deal with the matrices resulting from the whole range of discretizations. From the preceding discussion it should be clear, therefore, that there is a need to find a procedure that can recognize and effectively treat any infinite or ill-disposed eigenvalues that appear during the course of the diagonalization, and can allow SVD to be used in cases where $H^TH + \lambda C$ is singular or ill-conditioned. The formalism for diagonalizing $H^TH + \lambda C$ when C is non-singular, combined with method II when C is singular just cannot do this. Fortunately, the Fix-Heiberger algorithm (Fix and Heiberger 1972, Parlett 1980, §15.5) was designed for just those circumstances. It works precisely because it finds and isolates any infinite or ill-disposed eigenvalues at the earliest possible stage. These are then treated with the contempt they deserve, being set to zero when necessary to remove the effects they may have on the computation of the other eigenvalues.

The Fix-Heiberger reduction is just an extension of the formalism given earlier in this section to perform diagonalization when C is non-singular, but the way the ill-disposed eigenvalues are isolated introduces block matrices (again). In the most general case, the final reduced matrix contains five rows and columns of blocks, but in most situations this will not be required, and much of the algorithm can be omitted. For example, if C is well-conditioned most of the procedure is omitted, the algorithm becomes exactly equivalent to the standard method for C non-singular, and no block matrices are needed. As before, the reduction is achieved by a sequence of λ-independent congruence transformations, but now six, $P_1, \ldots, P_6$, are required in general. The end result is a relationship

$$H^T H + \lambda C = [P_1 \ldots P_6] M(\lambda) [P_6^T \ldots P_1^T],$$

where M is not, in the most general case, diagonal, but is equally trivially invertible, and contains only a very small number of non-zero off-diagonal elements (and they are all 1), so that multiplying with M^{-1} is no slower than for a diagonal matrix.

Broadly, the plan of attack is to diagonalize submatrices that lie on the diagonal, and to find congruence transformations to remove off-diagonal blocks, where this is possible. There are three procedures that are central to the algorithm. The first is diagonalization of matrices and submatrices (and in one case the use of SVD to 'diagonalize' a non-square submatrix). This is achieved as usual by finding eigenvalues (singular values) and vectors. The next is the truncation of the eigenvalue spectrum according to some rule for determining when eigenvalues (or singular values) are small enough to cause problems in the reduction. This requires some such rule (an appropriate value of ε, in the notation used during the explanation of SVD given earlier) to be specified in each case. Finally, once the truncated spectrum of eigenvalues has been computed, it is necessary to find an invertible matrix related to the diagonal matrix of eigenvalues (which will in general be singular, of course) that can be used in a congruence transformation (that is, takes on a similar role to $D^{1/2}$ in the old diagonalization procedure for non-singular C – see equations (2.112) and (2.113)). As the very first step in the Fix-Heiberger algorithm involves all three of these procedures, more detailed discussion of them will be made there, where it will be easier to see their significance.

Finally, then, here is the Fix-Heiberger algorithm for simultaneously diagonalizing (well, almost) the two real, symmetric, $n \times n$ matrices $H^T H$ and C:

STEP 1) Diagonalize the smoothing matrix C, just as before, to obtain $C = UDU^T$, with U orthonormal and D the diagonal matrix of eigenvalues of C, which we assume (as we will at every stage of the algorithm) are arranged into descending order – this is important for determining the block structure of the matrices that occur. We now need to examine the eigenvalues to check for any trouble-makers. That is, we need to decide which eigenvalues to accept, and which eigenvalues to treat as though they were zero.

The criterion to do this must certainly exclude any eigenvalues that are small enough to be indistinguishable from round-off, but it should not reject too many of the larger eigenvalues, as this will result in significant truncation error in the diagonalization. In fact, ignoring small but non-zero eigenvalues is what makes this algorithm reliable, and Fix and Heiberger (1972) discuss the errors caused by neglecting these small eigenvalues. This is such an important part of the algorithm that it will pay to be consistent in our determination of when an eigenvalue is negligible. As the errors on the eigenvalues of a matrix are relative to the size of the elements of the matrix, the criterion should not be absolute, but rather, should depend on the *norms* of the matrices involved (see Craig and Brown 1986, §5.3.2, or Parlett 1980, §1.6). Now that we have the eigenvalues of C the easiest matrix norm to work with is the *spectral norm* which just equals the largest eigenvalue of the matrix. A sensible criterion for rejecting an eigenvalue d_i of C is: Given some (small) number ε, neglect the eigenvalue d_i if

$$|d_i| < \varepsilon \|C\| = \varepsilon d_1. \tag{2.117}$$

(Remember, d_1 is the largest eigenvalue of C.) There is no 'correct' value of ε – it just must be not too large and not too small. Experimentation should suggest an appropriate value. Parlett recommends trying the two values $\varepsilon = \surd(\text{machine precision})$ and $\varepsilon = n\times(\text{machine precision})$, and comparing the results (Parlett 1980, §15.5). Henceforth it will be assumed that some such ε has been set, and the criterion (2.117) will be used to determine zero eigenvalues.

Using (2.117) we can say that the first r_1 diagonal elements of D are non-zero, and can treat the remaining $n - r_1$ entries as zero. (So that r_1 is the effective rank of C.) The goal of this step in the reduction is to use a congruence transformation to reduce the smoothing matrix to the identity, as was achieved in the old diagonalization procedure. In the general case, C will be singular and so cannot be congruent to I (because congruence transfomations preserve the rank of matrices) and we must find another tranformation that simplifies C as much as possible. The simplest matrix that has the same rank as C is the diagonal matrix with r_1 1's along the diagonal and zeros everywhere else. In general, C will not have full rank (i.e. $r_1 < n$), which means that it is not possible to use $D^{1/2}$ in a congruence transformation, so we must find an alternative. We will need to perform similar tricks later, so we will introduce some notation now. Define $\tilde{D}$ to be the diagonal matrix obtained from D by replacing all the zero eigenvalues by 1, so that

$$\tilde{D} = \mathrm{diag}\{d_1, \ldots, d_{r_1}, 1, \ldots, 1\}, \tag{2.118}$$

and define the matrix $\tilde{I}_{r_1}$ to be

$$\tilde{I}_{r_1} = \mathrm{diag}\{\underbrace{1, \ldots, 1}_{r_1\ 1\text{'s}}, \underbrace{0, \ldots, 0}_{n-r_1\ 0\text{'s}}\}. \tag{2.119}$$

Then $\tilde{D}$ is non-singular, and it is quite obvious that we can write $D = \tilde{D}^{1/2}\tilde{I}_{r_1}\tilde{D}^{1/2}$, so that $C = (U\tilde{D}^{1/2})\tilde{I}_{r_1}(U\tilde{D}^{1/2})^T$. It is clear now that we should make the definition

$$P_1 = U\tilde{D}^{1/2}. \tag{2.120}$$

$H^TH + \lambda C$ can now be written

$$H^TH + \lambda C = P_1(A + \lambda\tilde{I}_{r_1})P_1^T, \tag{2.121}$$

where we have already defined $A = \tilde{D}^{-1/2}U^TH^THU\tilde{D}^{-1/2}$, so that A is another symmetric matrix. This completes step 1, except to note for later reference (when it comes to inverting $H^TH + \lambda C$) that $P_1^{-T} = U\tilde{D}^{-1/2}$.

If $r_1 = n$, so that C is well-conditioned, then the next four steps do not apply, and step 6 completes the diagonalization as usual (only the second diagonal block, $\Delta + \lambda I$, in equation (2.135) actually appears in this case, and the final transformation matrix is similarly abbreviated).

STEP 2) First, note that $\tilde{I}_{r_1}$ can be written as a 2×2 block matrix, with the upper left block being the identity matrix, and the other blocks being zero. Partitioning A in a corresponding manner, it is easy to see that

$$A + \lambda\tilde{I}_{r_1} \equiv \left(\begin{array}{c|c} A_1 + \lambda I & A_3 \\ \hline A_3^T & A_2 \end{array}\right)\begin{array}{l}\}r_1 \\ \}n - r_1\end{array}$$

Now diagonalize the $n-r_1 \times n-r_1$ matrix A_2, to get $A_2 = VEV^T$, with V orthonormal and E diagonal. Applying the condition (2.117) to A_2 (instead of C) we can say that the first r_2 diagonal elements of E are non-zero, and define $\tilde{E}$ and $\tilde{I}_{r_2}$ just as was done for D. Then $A_2 = (V\tilde{E}^{1/2})\tilde{I}_{r_2}(V\tilde{E}^{1/2})^T$, so if we define the second transformation matrix by

$$P_2 = \left(\begin{array}{c|c} I & 0 \\ \hline 0 & V\tilde{E}^{1/2} \end{array}\right)\begin{array}{l}\}r_1 \\ \}n - r_1\end{array} \tag{2.122}$$

it is a simple matter to check that

$$A + \lambda \tilde{I}_{r_1} = P_2 \left(\begin{array}{c|c} A_1 + \lambda I & \bar{A}_3 \\ \hline \bar{A}_3^T & \tilde{I}_{r_2} \end{array} \right) P_2^T, \tag{2.123}$$

where $\bar{A}_3 = A_3 V \tilde{E}^{-1/2}$. P_2^{-T} can be found from $(V\tilde{E}^{1/2})^{-T} = V\tilde{E}^{-1/2}$.

STEP 3) In order to extend the partitioning of $\tilde{I}_{r_2}$ into block matrices ($\tilde{I}_{r_2}$ is partitioned in a similar way to $\tilde{I}_{r_1}$ in step 1, *cf.* equation (2.119)), split the submatrix $\bar{A}_3$ into $(B_1|B_2)$, where B_1 is $r_1 \times r_2$ and B_2 is $r_1 \times n - (r_1 + r_2)$. We then arrive at

$$\left(\begin{array}{c|c} A_1 + \lambda I & \bar{A}_3 \\ \hline \bar{A}_3^T & \tilde{I}_{r_2} \end{array} \right) \equiv \left(\begin{array}{c|c|c} A_1 + \lambda I & B_1 & B_2 \\ \hline B_1^T & I & 0 \\ \hline B_2^T & 0 & 0 \end{array} \right) \begin{array}{l} \}r_1 \\ \}r_2 \\ \}n - (r_1 + r_2) \end{array}$$

Defining the third transformation matrix to be

$$P_3 = \left(\begin{array}{c|c|c} I & B_1 & 0 \\ \hline 0 & I & 0 \\ \hline 0 & 0 & I \end{array} \right) \begin{array}{l} \}r_1 \\ \}r_2 \\ \}n - (r_1 + r_2) \end{array} \tag{2.124}$$

will facilitate the zeroing of the B_1 submatrices. The result of this transformation is then

$$\left(\begin{array}{c|c} A_1 + \lambda I & \bar{A}_3 \\ \hline \bar{A}_3^T & \tilde{I}_{r_2} \end{array} \right) = P_3 \left(\begin{array}{c|c|c} (A_1 - B_1 B_1^T) + \lambda I & 0 & B_2 \\ \hline 0 & I & 0 \\ \hline B_2^T & 0 & 0 \end{array} \right) P_3^T \tag{2.125}$$

It can be verified that P_3^{-1} has exactly the same block structure as P_3, the only difference being that the B_1 block is replaced with $-B_1$. P_3^{-T} can easily be found from this.

If, in step 2, the matrix A_2 had full rank (so that $r_2 = n - r_1$) the third row and column of blocks will be absent, and steps 4 and 5 will be unnecessary. In that case, go straight from here to step 6, remembering that that last two rows and columns of the matrices there do not appear.

STEP 4) Now try to simplify the (non-square) B_2 submatrices. It is (it seems) impossible to zero these blocks with λ-independent congruence transformations. The next best thing is to reduce them to the simplest form possible, which is the matrix $\hat{I}$ given in equation (2.128). This is more than adequate for our purposes. The trick here is to use SVD to write B_2 as

$$B_2 = XWY^T, \tag{2.126}$$

where X is an $r_1 \times r_1$ orthonormal matrix, Y is an $n-(r_1+r_2) \times n-(r_1+r_2)$ orthonormal matrix and W is a non-square matrix (having the same shape as B_2) which is zero off the 'diagonal' and contains the singular values, $w_1, \ldots, w_{n-(r_1+r_2)}$, of B_2 along the diagonal. Any reasonable smoothing matrix will only have a relatively few zero eigenvalues, meaning that we definitely expect $n - r_1$ to be less than r_1. This assures that $n - (r_1 + r_2) < r_1$, and hence that B_2 has $n-(r_1+r_2)$ singular values. W is not square, but the fact that it has more rows than columns, and that it is diagonal (i.e. only elements W_{ii} are non-zero) allows it to be written

$$W = \left(\begin{array}{c} W_1 \\ \hline 0 \end{array}\right)\begin{array}{l} \}n-(r_1+r_2) \\ \}r_1 - [n-(r_1+r_2)] \end{array} \tag{2.127}$$

Here, W_1 is a square matrix with $n-(r_1+r_2)$ rows whose diagonal elements are the singular values of B_2. We can now apply (2.117) to W_1 to determine the singular values that are effectively zero, and again we can define a new diagonal matrix $\tilde{W}_1$ differing from W_1 only in that the zero singular values are replaced with 1. Let the number of non-zero singular values, which is the rank of the matrix B_2, be r_3. Then $\tilde{W}_1 = \mathrm{diag}\{w_1, \ldots, w_{r_3}, 1, \ldots, 1\}$, and we can introduce the matrix $\tilde{I}_{r_3}$ having the same dimensions as W_1 and containing 1's in the first r_3 places along the diagonal, and zeros everywhere else. It is obvious that $W_1 = \tilde{I}_{r_3}\tilde{W}_1$. Note that we do not exactly follow the procedure outlined in step 1 (where we wrote $D = \tilde{D}^{1/2}\tilde{I}_{r_1}\tilde{D}^{1/2}$). This is because W is a non-square off-diagonal block, so the need to preserve symmetry does not arise. To make use of this expression for W_1 it is helpful to define a new block-matrix $\hat{I}$ having exactly the same dimensions and partitioning as W in (2.127) but with $\tilde{I}_{r_3}$ in the top block. That is,

$$\hat{I} = \left(\begin{array}{c} \tilde{I}_{r_3} \\ \hline 0 \end{array}\right). \tag{2.128}$$

Using (2.126), (2.127) and the preceding definitions it is a simple matter to verify that

$$B_2 = X\left(\begin{array}{c} \tilde{I}_{r_3}\tilde{W}_1 \\ \hline 0 \end{array}\right)Y^T = X\hat{I}\tilde{W}_1Y^T.$$

Given that it has already been stated that we are attempting to reduce B_2 to $\hat{I}$, a moment's thought shows that the required transformation matrix is

$$P_4 = \left(\begin{array}{c|c|c} X & 0 & 0 \\ \hline 0 & I & 0 \\ \hline 0 & 0 & Y\tilde{W}_1 \end{array}\right), \tag{2.129}$$

which results in the following transformation

$$\left(\begin{array}{c|c|c} (A_1 - B_1 B_1^T) + \lambda I & 0 & B_2 \\ \hline 0 & I & 0 \\ \hline B_2^T & 0 & 0 \end{array}\right) = P_4 \left(\begin{array}{c|c|c} \hat{A} + \lambda I & 0 & \hat{I} \\ \hline 0 & I & 0 \\ \hline \hat{I}^T & 0 & 0 \end{array}\right) P_4^T, \qquad (2.130)$$

where the definition $\hat{A} = X^T(A_1 - B_1 B_1^T)X$ has been used.

Again it is helpful later to notice that the expression for P_4^{-T} follows easily once it is observed that $(Y\tilde{W}_1)^{-T} = Y\tilde{W}_1^{-1}$ and $X^{-T} = X$.

STEP 5) The result of the reduction in step 4 is a block matrix containing only $\hat{I}$ in its last row and column. $\hat{I}$ is itself a block matrix, and it will be advantageous to extend the partitioning if $\hat{I}$ to the entire matrix. In fact, the submatrix $\tilde{I}_{r_3}$ appearing in the definition of $\hat{I}$ in (2.128) is *also* a block matrix, containing the single non-zero block I (the $r_3 \times r_3$ identity matrix) in the upper left corner. Adjoining the last row of zero blocks in $\tilde{I}_{r_3}$ to the last row of zero blocks in the definition (2.128) gives the following block structure to $\hat{I}$:

$$\hat{I} = \left(\begin{array}{c|c} I & 0 \\ \hline 0 & 0 \end{array}\right)\begin{array}{l} \}r_3 \\ \}r_1 - r_3 \end{array}$$

(This is the general case. If the matrix B_2 in step 4 has full rank, then $r_3 = n - (r_1 + r_2)$, and the last column of $\hat{I}$ is absent.) Extending this partitioning to the whole of the matrix requires the division of the submatrix $\hat{A}$ in a corresponding manner. It should be obvious from the following tableau how this partitioning is done.

$$\left(\begin{array}{c|c|c} \hat{A} + \lambda I & 0 & \hat{I} \\ \hline 0 & I & 0 \\ \hline \hat{I}^T & 0 & 0 \end{array}\right) = \left(\begin{array}{c|c|c|c|c} \hat{A}_1 + \lambda I & \hat{A}_3 & 0 & I & 0 \\ \hline \hat{A}_3^T & \hat{A}_2 + \lambda I & 0 & 0 & 0 \\ \hline 0 & 0 & I & 0 & 0 \\ \hline I & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right)\begin{array}{l} \}r_3 \\ \}r_1 - r_3 \\ \}r_2 \\ \}r_3 \\ \}n - R \end{array} \qquad (2.131)$$

In (2.131) the shorthand $R = r_1 + r_2 + r_3$ has been used in declaring the size of the last row (and column) of blocks. Quite obviously, the presence of the last row and column of zero blocks makes the matrix singular, in general. There are $n - R$ rows of zeros, so the rank of the matrix (which must also be the rank of $H^T H + \lambda C$, since they are related by a sequence of rank-preserving congruence transformations) can be at most R. In fact, the rank of $H^T H + \lambda C$ will turn out to be exactly $R = r_1 + r_2 + r_3$. When $R < n$, $H^T H + \lambda C$ will not be invertible in the usual sense. However, the earlier discussion suggests that

SVD will give a useful inverse. The discussion following step 6 considers the inversion of $H^TH + \lambda C$.

Having arrived at the final partitioning of the matrix, we want to try to remove off-diagonal blocks. Defining

$$P_5 = \left(\begin{array}{c|c|c|c|c} I & 0 & 0 & 0 & 0 \\ \hline 0 & I & 0 & \hat{A}_3^T & 0 \\ \hline 0 & 0 & I & 0 & 0 \\ \hline 0 & 0 & 0 & I & 0 \\ \hline 0 & 0 & 0 & 0 & I \end{array}\right) \qquad (2.132)$$

can be seen to remove the $\hat{A}_3$ blocks, resulting in the matrix in (2.131) being given by

$$\left(\begin{array}{c|c|c} \hat{A} + \lambda I & 0 & \hat{I} \\ \hline 0 & I & 0 \\ \hline \hat{I}^T & 0 & 0 \end{array}\right) = P_5 \left(\begin{array}{c|c|c|c|c} \hat{A}_1 + \lambda I & 0 & 0 & I & 0 \\ \hline 0 & \hat{A}_2 + \lambda I & 0 & 0 & 0 \\ \hline 0 & 0 & I & 0 & 0 \\ \hline I & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right) P_5^T \qquad (2.133)$$

To derive an expression for P_5^{-T}, observe that P_5^{-1} has precisely the same block structure as P_5, but with $\hat{A}_3^T$ replaced by $-\hat{A}_3^T$. The form of P_5^{-T} follows immediately from this.

STEP 6) The only thing left to do now is to diagonalize any remaining blocks on the diagonal ($\hat{A}_1$ and $\hat{A}_2$, in other words). As usual, find the eigenvectors and eigenvalues of $\hat{A}_1$ and $\hat{A}_2$ and write them as

$$\hat{A}_1 = LFL^T \text{ and } \hat{A}_2 = Q\Delta Q^T,$$

where L and Q are orthonormal (of different sizes) and F and Δ are diagonal matrices of eigenvalues. Then, defining the final transformation matrix by

$$P_6 = \left(\begin{array}{c|c|c|c|c} L & 0 & 0 & 0 & 0 \\ \hline 0 & Q & 0 & 0 & 0 \\ \hline 0 & 0 & I & 0 & 0 \\ \hline 0 & 0 & 0 & L & 0 \\ \hline 0 & 0 & 0 & 0 & I \end{array}\right) \qquad (2.134)$$

results in the final form for the reduced $H^TH + \lambda C$ matrix. It is convenient typographically to introduce the notation

$$M(\lambda) \overset{\text{def}}{\equiv} \left(\begin{array}{c|c|c|c|c} F+\lambda I & 0 & 0 & I & 0 \\ \hline 0 & \Delta + \lambda I & 0 & 0 & 0 \\ \hline 0 & 0 & I & 0 & 0 \\ \hline I & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right) \tag{2.135}$$

for the reduced form of $H^TH + \lambda C$. Application of the transformation matrix P_6 to the block matrix on the right hand side of (2.133) then gives

$$\left(\begin{array}{c|c|c|c|c} \hat{A}_1+\lambda I & 0 & 0 & I & 0 \\ \hline 0 & \hat{A}_2 + \lambda I & 0 & 0 & 0 \\ \hline 0 & 0 & I & 0 & 0 \\ \hline I & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right) = P_6 M(\lambda) P_6^T. \tag{2.136}$$

Note that there is no need to apply the criterion (2.117) to the eigenvalues of $\hat{A}_1$ and $\hat{A}_2$, because they are not used in the congruence transformation (so there is no need to define matrices $\tilde{F}$ or $\tilde{\Delta}$ as there was when diagonalizing C in step 1), and because the eigenvalues always appear with the regularization parameter, so that there should never be a problem with having to take the reciprocal of a zero eigenvalue.

P_6 is a block diagonal matrix whose diagonal blocks are all orthonormal matrices. This means that P_6 is itself orthonormal, and so $P_6^{-T} = P_6$.

That completes the description of the Fix-Heiberger algorithm. The separate steps in the algorithm, when combined, give the following expression for $H^TH + \lambda C$:

$$H^TH + \lambda C = [P_1P_2P_3P_4P_5P_6]\, M(\lambda)\, [P_1P_2P_3P_4P_5P_6]^T. \tag{2.137}$$

The combination of the six congruence transformations in equation (2.137) has succeeded in reducing $H^TH + \lambda C$ to a form, $M(\lambda)$, which is almost diagonal. The only off-diagonal blocks are two $r_3 \times r_3$ identity matrices, and, as will be shown below, these do not in any way prevent the fast and efficient inversion and use of the transformed matrix in the GCV and EDF function evaluations (see also appendix A).

It may be helpful to review the way in which some of the steps of the algorithm are omitted depending on the values of r_1, r_2 and r_3.

1. If $r_1 = n$ only steps 1 and 6 are required, and in step 6 the matrix to be transformed has only one block, which corresponds to $\hat{A}_2$ – all the other blocks do not appear. Similarly, only the diagonal block containing Q appears in the transformation matrix P_6. The procedure is then exactly the same as the old method for diagonalizing $H^T H + \lambda C$ when C is non-singular, and the transformed matrix is just given by $M(\lambda) = \Delta + \lambda I$.

2. If $r_1 + r_2 = n$ the final matrix is only a 2×2 block matrix. Steps 4 and 5 are omitted and the final transformed matrix is
$$M(\lambda) = \left(\begin{array}{c|c} \Delta + \lambda I & 0 \\ \hline 0 & I \end{array} \right).$$
The transformation matrix P_3 does not have the final row and column appearing in (2.125) in this case.

3. If $R \equiv r_1 + r_2 + r_3 = n$, then all the steps in the reduction are required, but the final matrix, $M(\lambda)$, does not contain the last row and column of zero blocks. The transformation matrices in steps 5 and 6 are similarly abbreviated.

To conclude this section, it is necessary to examine inverting $H^T H + \lambda C$ once the congruences of the Fix-Heiberger reduction have been implemented, since it was the calculation of this inverse that was the prime motivation for introducing the algorithm. Inverting (2.137) directly results in

$$(H^T H + \lambda C)^{-1} = P M^{-1}(\lambda) P^T, \tag{2.138}$$

where we have set

$$P \overset{\text{def}}{\equiv} [P_1 P_2 P_3 P_4 P_5 P_6]^{-T} = P_1^{-T} P_2^{-T} P_3^{-T} P_4^{-T} P_5^{-T} P_6^{-T}. \tag{2.139}$$

The inverse transposes of the congruence matrices were calculated in each step of the Fix-Heiberger algorithm, so the inverse transpose of P is known. As we only require the inverse of $H^T H + \lambda C$, introducing P makes the notation much simpler. It is necessary to find the inverse of the block matrix $M(\lambda)$ appearing in (2.138) and defined in (2.135). To begin with, consider the case $R = n$, so that $H^T H + \lambda C$ is non-singular, and the last row and column of zeros in the matrix on the right hand side of (2.138) do not arise.

Essentially, the final transformed matrix is block diagonal with the three blocks $\Delta + \lambda I$, I and

$$\left(\begin{array}{c|c} F + \lambda I & I \\ \hline I & 0 \end{array}\right), \tag{2.140}$$

except that the last matrix has been split into quarters and put in the first and last rows/columns. The inverse of a block-diagonal matrix is well known to be a similar matrix whose diagonal blocks are just the inverses of their corresponding blocks. The resulting matrix, then, has the blocks $\Delta + \lambda I$ and I on the diagonal, and these can be inverted separately, because there are no off-diagonal blocks appearing in the same row or column. This leaves the first and fourth rows and columns to be dealt with. It is easy to verify by direct calculation that the inverse of the matrix in (2.140) is just

$$\left(\begin{array}{c|c} 0 & I \\ \hline I & -(F + \lambda I) \end{array}\right).$$

Using these results allows $M^{-1}(\lambda)$ to be written

$$M^{-1}(\lambda) = \left(\begin{array}{c|c|c|c} 0 & 0 & 0 & I \\ \hline 0 & (\Delta + \lambda I)^{-1} & 0 & 0 \\ \hline 0 & 0 & I & 0 \\ \hline I & 0 & 0 & -(F + \lambda I) \end{array}\right). \tag{2.141}$$

(Remember that $R = n$ has been assumed, so a row and a column have been lost.)

What happens in the most general case when $R < n$, so that $H^T H + \lambda C$ is singular? Imagine that the zero block in the lower right corner of $M(\lambda)$ in (2.135) is instead some diagonal matrix $T = \text{diag}\{t_1, \ldots, t_{n-R}\}$, so that $M(\lambda)$ becomes

$$\left(\begin{array}{c|c|c|c|c} F + \lambda I & 0 & 0 & I & 0 \\ \hline 0 & \Delta + \lambda I & 0 & 0 & 0 \\ \hline 0 & 0 & I & 0 & 0 \\ \hline I & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & T \end{array}\right)$$

This is effectively a block-diagonal matrix comprising the large matrix (containing the first four rows and columns of blocks in $M(\lambda)$) whose inverse was given in (2.141) and the corner block T. The inverse of this modified $M(\lambda)$ is found by inverting these two blocks separately. We know the inverse of the first block (it is given in (2.141)), but what is the inverse of T? That's easy. T is diagonal, so its inverse is trivial to calculate using SVD. T^{-1} is the diagonal matrix containing $1/t_i$ in the ith place along the diagonal if $t_i \neq 0$, and 0 there if $t_i = 0$. But the zero matrix that really appears in the corner of $M(\lambda)$ is a diagonal matrix with *all* its diagonal entries zero. Its inverse is then also a diagonal matrix with all its diagonal elements zero. In other words, when using SVD, the inverse of the zero matrix is itself. This solves the problem, and we have, in general

$$M^{-1}(\lambda) = \left(\begin{array}{c|c|c|c|c} 0 & 0 & 0 & I & 0 \\ \hline 0 & (\Delta + \lambda I)^{-1} & 0 & 0 & 0 \\ \hline 0 & 0 & I & 0 & 0 \\ \hline I & 0 & 0 & -(F + \lambda I) & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right) . \tag{2.142}$$

Effectively what this means is that the Fix-Heiberger reduction has isolated the zero eigenvalues of $H^T H + \lambda C$ and placed them in the lower right corner of $M(\lambda)$ out of the way.

Using this expression for $M^{-1}(\lambda)$ in (2.138) gives the final answer we require:

$$(H^T H + \lambda C)^{-1} = P \left(\begin{array}{c|c|c|c|c} 0 & 0 & 0 & I & 0 \\ \hline 0 & (\Delta + \lambda I)^{-1} & 0 & 0 & 0 \\ \hline 0 & 0 & I & 0 & 0 \\ \hline I & 0 & 0 & -(F + \lambda I) & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right) P^T . \tag{2.143}$$

It is in order to ask what the significance of a solution obtained through SVD is. In fact, use of SVD amounts to little more than another regularization. SVD does not allow vectors in the null space of $H^T H$ to contribute to the solution. To see this recall that, from (2.82), the solution vector is given by

$$\mathbf{f} = (H^T H + \lambda C)^{-1} H^T \mathbf{g}.$$

So whatever the vector $H^T\mathbf{g}$ actually is, any components belonging to the null space of $H^T H + \lambda C$ are multiplied by the zeros in the SVD inverse of $H^T H + \lambda C$ and so contribute nothing to the solution. But any vector in the null space of $H^T H + \lambda C$ must also be in the null space of $H^T H$. (The argument goes like this: Suppose $\mathbf{v}$ is a unit vector in the null space of $H^T H + \lambda C$. Then

$$(H^T H + \lambda C)\mathbf{v} = \mathbf{0} \Rightarrow \mathbf{v}^T H^T H \mathbf{v} + \lambda \mathbf{v}^T C \mathbf{v} = 0.$$

But $H^T H$ and C are positive semi-definite and $\lambda > 0$, so we must have $\mathbf{v}^T H^T H \mathbf{v} = 0$ and $\mathbf{v}^T C \mathbf{v} = 0$ separately, which means that $\mathbf{v}$ must be in the null spaces of both $H^T H$

and C.) Now consider components of the (unknown) true solution, $\mathbf{f}_0$, that belong to the null space of $H^T H$. The solution and the data are related by

$$\mathbf{g} = H\mathbf{f}_0$$

(plus errors, but we won't worry about those here). Any component, $\mathbf{v}$, of the true solution lying in the null space of $H^T H$ must be mapped to zero by the kernel matrix H (otherwise, $||H\mathbf{v}||^2 \equiv \mathbf{v}^T H^T H \mathbf{v} \neq 0$). In other words, these components will contribute nothing to the data vector $\mathbf{g}$. This means that information about components of the true solution lying in the null space of $H^T H$ is not contained in the data, so setting them to zero in the recovered solution is a natural way to reflect this lack of information. This is why SVD works so well for solving linear problems of the type considered here.

2.5 Summary

In this chapter a quite detailed and general formalism for solving linear inverse problems of the type (2.5) has been put forward. The heart of this procedure is the algorithm outlined in §2.2, for regularizing and discretizing the continuous inverse problem (2.5). The idea of using a *regularizing functional* on the function space of solutions was introduced. Only once this was done was a reduction of the number of degrees of freedom in the solution performed by discretizing the regularized problem. The advantage of this approach is that the regularization applied ought to be largely independent of the form of discretization chosen, something that was not always the case in previous implementations of RLS techniques. The major part of the algorithm, though, is the discretization itself, §2.2.3. This unifies and generalizes the four types of RLS method commonly used (see §1.9), taking the ideas underlying piecewise-constant discretization and function-expansion methods and combining them to provide even greater freedom for choosing a discretization suited to the problem at hand. Although the discretizations can be used with any form of regularization constraint, a large part of the discussion in §2.2 dealt with quadratic smoothing constraints and general pth order (pth derivative) smoothing. A quite general formalism for calculating the smoothing matrix in such cases was given in §2.2.4.

The greater freedom for discretization, and the possible existence of large jumps in the solution at the boundaries between discretization bins (when two neighbouring bins are large, derivative smoothing constraints may not actually constrain the jump in the solution across their boundary very much), suggest that some procedure for imposing constraints on the solution is desirable. In §2.3 just such a procedure was derived. The idea was to change basis in the vector space containing possible solutions so that the constraints completely fix the contribution to the solution from some of the new basis vectors (the components contained in the vector $\tilde{\mathbf{f}}_1$, using the notation of §2.3, are fixed), and leave the other components ($\tilde{\mathbf{f}}_2$) completely free. The expression for the analytic solution in the presence of these constraints was then derived using the knowledge that only the components in $\tilde{\mathbf{f}}_2$ can be varied in the minimization of the regularized functional $\mathcal{R}(\mathbf{f})$ (equation (2.80)). Two alternative forms for the solution were given, each with its own advantages and disadvantages. The pros and cons of each were discussed, to provide some indication of the circumstances in which the use of one in preference to the other would be advisable.

As was described in §2.4, the use of automatic methods for choosing the smoothing parameter in quadratic RLS inversions forces us to find a way to compute $(H^TH + \lambda C)^{-1}$ very quickly. The solution to this problem is to diagonalize $H^TH + \lambda C$. Standard methods for performing this diagonalization when C is non-singular or when C is singular but $H^TH + \lambda C$ is well-conditioned for inversion were shown to be unsuitable for use with the algorithm presented in §2.2, because the wide range of discretizations that can be used creates the possibility that the smoothing matrix will be non-singular and $H^TH + \lambda C$ will be ill-conditioned. The Fix-Heiberger algorithm, which was designed for solving just such problems, was then described in some detail, and an expression for $(H^TH+\lambda C)^{-1}$ in terms of the final reduced matrix was given (in equation (2.143)). The Fix-Heiberger algorithm has not been used in helioseismology inversions before, but its increased robustness make it very appealing for the solution of discretized RLS problems with the procedure given in §2.2.

Chapter 3

Choosing the Smoothing Parameter

3.1 Introduction

Helioseismology relies on the existence of efficient techniques for inferring aspects of the solar structure from (often large quantities of) data derived from observations of solar oscillations. Typically, the data are the measured frequencies of individual modes or frequency splittings between modes, and the quantities recovered will be, in the former case, the run of sound speed throughout the sun, perhaps along with information about the composition of the solar material (Dziembowski *et al.* 1990), and in the latter, components of the sun's internal angular velocity as a function of depth (see Ritzwoller and Lavely 1991). In general, the relationship between the data, $\{g_i; i = 1, \ldots, p\}$, where p is the number of mode frequencies or splittings contained in the data set, and the quantity of interest, $f(r)$, where r is the distance from the centre of the sun, is of the form

$$g_i = \int_0^{R_\odot} k_i(r) f(r) dr. \tag{3.1}$$

Here $k_i(r)$ are the *kernel functions* which essentially decide how much the value of f near r contributes to the value of the ith data point. To fix ideas, consider the solar rotation problem. If we assume that the internal angular velocity, Ω, of the sun is a function only of radius, $\Omega = \Omega(r)$, then the frequency splitting caused by this rotation for any mode (n, l, m) is (see Unno *et al.* 1979)

$$\nu_{nlm} - \nu_{nl0} = -\frac{m}{J} \int_0^{R_\odot} K_{nl}(r) \Omega(r) \rho(r) r^2 dr, \tag{3.2}$$

with

$$K_{nl}(r) = \xi_{r\,nl}^2 - [1 - l(l+1)]\xi_{h\,nl}^2 - 2\xi_{r\,nl}\xi_{h\,nl}, \tag{3.3}$$

and

$$J = \int_0^{R_\odot} [\xi_{rnl}^2 + l(l+1)\xi_{hnl}]\rho r^2 dr.$$

($\rho(r)$ is the density of the solar material at radius r.)

In these equations ξ_{rnl} and ξ_{hnl} are, respectively, the radial and horizontal components of the displacement eigenfunction of the mode with degree l and radial order n. As usual, m is the azimuthal order of the mode. With a simple relabelling of the modes so that each (n, l, m) is assigned a value of i the connection between (3.2) and (3.1) is obvious.

Note that in the case of inversions to find the solar sound speed, a linear relation of the form (3.1) is derived by a linearization of the true equations about some given theoretical solar model (see Gough 1985). The result of the inversion of real solar oscillation data provides corrections to the solar model, which must then be recomputed and the procedure iterated until the corrections become sufficiently small.

It is clear that, in both of these cases, an efficient procedure for inverting systems of equations of the form (3.1) (that is, for finding $f(r)$ given $\{g_i\}$) is required. There are basically two methods commonly used for this. The first is known as *optimal averaging* and is described in Gough (1985). It consists of taking linear combinations of the kernel functions in an attempt to produce an *averaging kernel* which is sharply peaked about some point and has a small value away from this point. The linear combination that satisfies this gives a weighted average of the solution which is localized around the region where the kernel is large (i.e. only the value of $f(r)$ in the region where the kernel is large contributes significantly to the average). This average thus gives an estimate of the value of f in the region where the optimal averaging kernel is large. The other is based on the method of least squares and consists of approximating the solution $f(r)$ (which, being a function of a continuous variable, has infinitely many degrees of freedom) by a function that lies in a finite dimensional, linear function space with a basis $\{\phi_j(r); j = 1, \ldots, q\}$ where q is the dimension of this approximation space, and then finding the particular function in the approximation space that gives g_is which are as near as possible, in a least squares sense, to the observed data. This procedure amounts to approximating the integral in (3.1) by a sum, and the function $f(r)$ by a p-dimensional vector $\mathbf{f}$, giving, as an approximation to (3.1)

$$\mathbf{g} = H\mathbf{f},$$

where H is a $p \times q$ matrix given by

$$H_{ij} = \int_0^{R_\odot} k_i(r)\phi_j(r)dr$$

and the approximation to the solution is

$$\hat{f}(r) = \sum_{j=1}^{q} f_j\phi_j(r).$$

(This defines the vector $\mathbf{f}$ with components f_j.)

In the following we always use *piecewise-constant discretization* (PCD) – also known as *product integration* – where each ϕ_j is given by

$$\phi_j(r) = \begin{cases} 1, & \text{if } (j-1)R_\odot/q < r \leq jR_\odot/q \\ 0, & \text{otherwise.} \end{cases}$$

Then H is given by

$$H_{ij} = \int_{\frac{(j-1)R_\odot}{q}}^{\frac{jR_\odot}{q}} k_i(r)dr, \tag{3.4}$$

and the solution function is defined in terms of the solution vector (the f_j) by

$$\hat{f}(r) = \hat{f}_j \text{ for } (j-1)R_\odot/q < r \leq jR_\odot/q.$$

It is well known (see Craig and Brown 1986) that, in general, inversion procedures are unstable in the sense that the solution is very sensitive to perturbations in the data, giving rise to solutions which oscillate wildly and are clearly incorrect. These large, high spatial frequency components of the recovered solution are largely a consequence of the Riemann-Lebesgue lemma (Craig and Brown 1986, §4.2), which says that, with certain mild constraints on the kernel functions, as the spatial frequency, k, of the component $\sin kx$ of the true solution increases the amount that that component contributes to g_i, for any i, tends to zero, which means that solutions that differ only by high frequency components will be mapped by (3.1) to points in 'data-space' that are very close together. This problem is ameliorated by the technique of *regularization* (see Craig and Brown 1986), which introduces into the minimization over the approximation space an $\mathbf{f}$ dependent term $\Phi(\mathbf{f})$ which penalizes against any solution that has large high frequency components, i.e. against solutions that are not smooth in some sense. If this *smoothing constraint* is chosen to be of the form

$$\Phi(\mathbf{f}) = \mathbf{f}^T C\mathbf{f}, \tag{3.5}$$

(^{T} denotes matrix transpose), for some symmetric positive semi-definite $q \times q$ matrix C, then the quantity that must be minimized to find the estimate $\hat{\mathbf{f}}$ of $\mathbf{f}$ is

$$\mathcal{R}(\mathbf{f}) = ||\mathbf{g} - H\mathbf{f}||_p^2 + \lambda \mathbf{f}^T C \mathbf{f}, \tag{3.6}$$

($||.||_p^2$ is the Euclidean norm in p-dimensions), and the solution is given by

$$\hat{\mathbf{f}}(\lambda) = (H^T H + \lambda C)^{-1} H^T \mathbf{g}, \tag{3.7}$$

where $\lambda > 0$ is the *regularization* or *smoothing parameter.* The larger the value of λ, the more the smoothing constraint contributes to (3.6) and the smoother the solution must be to minimize (3.6). The choice of smoothing constraint (3.6) is known as *quadratic* regularization.

Sometimes it is appropriate to choose $C = I$, the identity matrix, which means that the smoothing constraint forces the solution vector to have a small norm. For PCD the norm of the vector $\hat{\mathbf{f}}$ is simply proportional to the norm of the approximate solution function, that is

$$||\hat{\mathbf{f}}||_q^2 = \hat{\mathbf{f}}^T I \hat{\mathbf{f}} \propto \int_0^{R_\odot} f(r)^2 dr.$$

So, for *zeroth order smoothing* ($C = I$), the norm of the approximate solution is forced to be small.

Often, a better choice for C is $C = X^T X$, where X is the 'first derivative matrix' (which, since $\hat{\mathbf{f}}$ provides information about the values of $\hat{f}(r)$ in a series of narrow bins, is just the matrix that maps $\hat{\mathbf{f}}$ to the finite difference approximation to the derivative of $\hat{f}(r)$). This forces the gradient of the approximate solution to be small, in the sense that $(f_j - f_{j-1})$ is small. This can easily be generalized to higher derivatives. In general, nth order smoothing forces the nth derivative of the approximate solution to be small.

For any given C, the solution (3.7) can be calculated for any λ. The calculation of $\hat{\mathbf{f}}$ is greatly facilitated by the use of a theorem of linear algebra which says that there exists a basis of $\mathbb{R}^q$ in which $H^T H$ and C are simultaneously diagonalized. It is therefore possible to write

$$\hat{\mathbf{f}} = A D^{-1}(\lambda) A^T H^T \mathbf{g}, \tag{3.8}$$

where A is a $q \times q$, invertible, λ-independent matrix, and $D(\lambda)$ is a diagonal matrix in which each diagonal element may be written $D_{jj}(\lambda) = \alpha_j + \lambda \beta_j$ for some α_j and β_j. Once this decomposition has been found the calculation of $\hat{\mathbf{f}}(\lambda)$ for many different values of λ

becomes computationally very easy. All we have to do now is decide which value of λ gives the best recovery. It will be the purpose of this chapter to address the question of choosing the smoothing parameter, λ, in any regularized least squares solution of linear inverse problems of the form (3.1) in an optimal way.

The next section discusses the effects of regularization on the estimated solution and uses these considerations to motivate criteria for the assessment of the efficiency of any method for choosing λ. Section 3.3 presents three alternative ways of choosing λ and briefly discusses their derivation. Section 3.4 describes the procedure used here to examine the efficiency of these methods, while §3.5 presents the results of this analysis, including an inversion of real data. Section 3.6 discusses the conclusions that can be drawn from these results.

First, however, the question of the errors on the data must be considered. Throughout this chapter all errors are assumed to be normally distributed, with mean zero. In real problems the variance, σ_i^2 of the errors on the ith data point, g_i, often varies with i. (In helioseismology the mode parameters are measured more accurately for some modes than for others.) Then minimizing (3.6) does not give the *maximum likelihood* estimate of $\mathbf{f}$, which is actually given by the minimizer of

$$\frac{1}{p}\sum_{i=1}^{p}\left(\frac{g_i - H_{ij}f_j}{\sigma_i}\right)^2 . \tag{3.9}$$

We now write $\sigma_i = \sigma_0\delta_i$, for some σ_0, not necessarily 1, assuming for simplicity that as data improves the variances of the errors on the g_i will reduce proportionately, i.e. σ_0 will decrease, but the δ_i will not change. (3.9) can then be written

$$\frac{1}{p\sigma_0^2}\sum_{i=1}^{p}\left(\frac{g_i - H_{ij}f_j}{\delta_i^2}\right)^2 = \frac{1}{p\sigma_0^2}\|\tilde{\mathbf{g}} - \tilde{H}\mathbf{f}\|^2,$$

where

$$\tilde{g}_i = \frac{g_i}{\delta_i} \text{ and } \tilde{H}_{ij} = \frac{H_{ij}}{\delta_i} \tag{3.10}$$

(*cf.* equations (2.27) and (2.29), although note the different notation used in chapter 2). From now on we assume that the normalization (3.10) has been performed, using the known errors from some data set, and drop the tildes. Introducing regularization as before (3.6) is recovered if λ is rescaled according to

$$\lambda \to p\sigma_0^2\lambda. \tag{3.11}$$

So, with the rescalings (3.10) and (3.11) the problem has data errors that are independent, identically distributed, gaussian errors with zero mean and variance σ_0^2.

3.2 The Effect of Different Values of λ on the Recovered Solution

Once the form of the smoothing matrix has been chosen, and a perusal of the data is often sufficient to guide this choice (for example, a large mean value of the data for a rotation inversion suggests, since the kernels are positive everywhere, that the solution has a large mean value, thus making zeroth order smoothing inappropriate), the value of the smoothing parameter must be chosen. There are a few comments worth making about the effect of different values of λ on the recovered solution:

a) As $\lambda \to 0$ the recovered solution tends towards the unstable unregularized solution.

b) As $\lambda \to \infty$ the solution conforms more and more with the demands of the smoothing constraint, and the data eventually becomes irrelevant. Somewhere between these two extremes there must be a value of λ that provides the best possible recovery of the solution. If we knew what the true solution of the inverse problem was it would be a relatively simple task to adjust λ until the recovered solution matched the true solution as accurately as possible. a) and b) together are equivalent to the statement that choosing λ amounts to finding a trade-off between the variance and bias of the estimator $\hat{\mathbf{f}}$.

c) In practical situations there is always a source of error, even when inverting 'exact' artificial data, since the numerical inversion procedure will suffer from truncation and rounding error. This means that in general we must have $\lambda > 0$ to ensure a satisfactory solution.

d) Clearly, if the true solution $\mathbf{f}$ matches very poorly with the smoothing constraint (i.e. if $\mathbf{f}^T C \mathbf{f}$ is large) then a large value of the smoothing parameter is unlikely to give satisfactory results. Conversely, if $\mathbf{f}^T C \mathbf{f} = 0$ then $\lambda \to \infty$ will give an excellent (perfect) recovery. Thus the value of λ is affected by the appropriateness of the smoothing constraint chosen.

It is clear, therefore, that the optimal value of the smoothing parameter, λ_{OPT}, given the smoothing matrix, C, depends on the true solution. Of course, in real situations the true solution is unknown, so methods must be found of estimating λ_{OPT} from the data, without recourse to knowledge of the true solution. Once such methods have been found their efficiency and accuracy must be examined. As the data errors are statistical in nature the properties of any estimator of λ_{OPT} can only be determined if some thought is given to the statistical properties of the noise. What properties would it be desirable for an estimator of λ_{OPT} to have? Certainly the expectation value of $\hat{\lambda}$, the estimate of λ_{OPT}, ought to be close to the mean value of λ_{OPT}, and the variance of $\hat{\lambda}$ should be similar to that of λ_{OPT}. This ensures that any estimate will at least lie in the correct range. There is one other property that the estimator should have for it to be completely successful, and that is a rather high correlation with λ_{OPT}. This would mean that when the noise realization led to a value of λ_{OPT} that was unusually large or small the estimator would follow suit. If the variances of λ_{OPT} and $\hat{\lambda}$ are small this last requirement is less important, as the estimate ought always to be in the right range. Thus, the criteria for assessing the estimators of λ_{OPT} presented in the next section will be the means and variances of their distributions, along with the correlation they show with the value of λ_{OPT}.

3.3 Methods for Estimating λ_{OPT}

a) With the notation:

$$R(\lambda) = \|\mathbf{g} - H\hat{\mathbf{f}}\|_p^2 \qquad (3.12)$$

$$T(\lambda) = \mathrm{Tr}\left\{I_p - H(H^T H + \lambda C)^{-1} H^T\right\} \qquad (3.13)$$

where Tr $\{A\}$ denotes the trace of the matrix A, the first estimator is given by the solution of

$$EDF(\lambda) = \frac{R(\lambda)}{T(\lambda)} - \hat{\sigma}^2 = 0, \qquad (3.14)$$

where $\hat{\sigma}^2$ is an estimate is the variance of the errors on the data. The value of the trace is a quantity known as the *equivalent degrees of freedom for error.* It basically measures the degrees of freedom in the *actual* errors ($p = \mathrm{Tr}\{I_p\}$) minus the number of degrees of freedom taken up by the fact that the actual inversion 'fits the noise' to a certain degree, so that the residuals have less degrees of freedom. The amount of freedom 'taken up' by the solution is given by $\mathrm{Tr}\left\{H(H^T H + \lambda C)^{-1} H^T\right\}$. This method will be referred to as EDF, and the resulting estimate as $\hat{\lambda}_{EDF}$.

Note that, usually, the exact variance of the data errors is unknown and so it must be estimated in some way. The accuracy of the estimator $\hat{\sigma}$ then becomes a crucial question. As is discussed below, the estimator of λ_{OPT} just given is extremely sensitive to $\hat{\sigma}$. This has important consequences which will be considered in section 3.5. For this method it is assumed that we know that $\hat{\sigma} = \sigma_0$ (but see §3.4), so that the dependence of $\hat{\lambda}_{EDF}$ on $\hat{\sigma}$ is not being examined.

The earliest method for estimating λ_{OPT} resulted from assuming that the residuals ought to be very close to the true errors for the optimal value of λ and thus that they ought to be distributed as $N(\mathbf{0}, \hat{\sigma}^2 I_p)$. The χ^2 estimate of λ_{OPT} is then given by the solution of

$$CHI(\lambda) = \frac{R(\lambda)}{p} - \hat{\sigma}^2 = 0. \tag{3.15}$$

Unfortunately, as mentioned above, the residuals do not have the simple distribution just given, but have a reduced 'effective variance' owing to the fact that the data will tend to fit the true errors. The shape of the function $EDF(\lambda)$ is such that its gradient is very small near the solution $\hat{\lambda}_{EDF}$ (hence, an incorrect value for $\hat{\sigma}^2$ will give an estimate of λ_{OPT} which is far away from the correct value, which explains the sensitivity of EDF to the noise estimate), this means that using p instead of $T(\lambda)$ will cause a large shift in the value of the root of (3.15) relative to the root of (3.14). Also, EDF is generally increasing near its root, so that using p ($> T(\lambda)$ for all λ) instead of $T(\lambda)$ gives $\hat{\lambda}_{\chi^2} \gg \hat{\lambda}_{EDF}$ and leads to severe oversmoothing. Once the reason for this was realized (3.15) was replaced by (3.14), which is a much better estimator of λ_{OPT}.

Equation (3.14) can also be derived using *bayesian maximum likelihood* (see Fitzpatrick 1991, and references therein) assuming that the solution $\mathbf{f}$ is drawn from a normal distribution with zero mean and covariance matrix $\tau^2 C$ with $\lambda = \frac{\sigma^2}{\tau^2}$. (It can easily be shown that quadratic regularization is exactly equivalent – for fixed λ – to bayesian ML with the prior just given. Maximizing the likelihood of the observed data over λ gives (3.14).)

b) The second method for choosing λ is known as *generalized cross validation* and is derived in Golub *et al.* (1979). The derivation goes as follows: For each $k = 1, \ldots, p$ remove g_k from the data set and infer

$$\hat{\mathbf{f}}^{(k)}(\lambda) = (H^{(k)T} H^{(k)} + \lambda C)^{-1} H^{(k)T} \mathbf{g}^{(k)},$$

where $H^{(k)}$ is the same as H but with the kth row deleted. Then predict ${g_k}^{(k)} = H_{kj}\hat{f}_j^{(k)}$ and minimize the *total mean square error of prediction* given by

$$GCV(\lambda) = \frac{1}{p}\sum_{k=1}^{p}({g_k}^{(k)} - g_k)^2 \tag{3.16}$$

over λ. The theory behind GCV says that the minimizer of $GCV(\lambda)$ should be a very good estimator of λ_{OPT}. In fact, as $p \rightarrow \infty$, $\hat{\lambda}_{GCV} \rightarrow \lambda_{OPT}$.

As Golub *et al.* (1979) show, (3.16) may be approximated by

$$GCV(\lambda) = \frac{R(\lambda)}{T(\lambda)^2}. \tag{3.17}$$

This form has certain advantages over (3.16) (Golub *et al.* 1979), not least the fact that it is much easier to work with, and will be used from now on.

Note that a knowledge of the variance of the data errors is not required for GCV to work. Moreover, the fact that EDF is very sensitive to the value of $\hat{\sigma}^2$ means that if a good estimate of λ_{OPT} is available EDF may be used to provide an excellent estimate of the error variance. $\hat{\lambda}_{GCV}$ is very well suited to this. Experience with fake data shows that this procedure gives a very good estimate of σ_0. This point will not be addressed further here.

c) Ideally, the optimal value of λ would be given by the minimizer of

$$TE_0(\lambda) = ||\hat{\mathbf{f}}(\lambda) - \mathbf{f}_0||_q^2. \tag{3.18}$$

It would be nice to make the identification $\hat{\lambda}_{TE_0} = \lambda_{OPT}$, but there is some subtlety here. Consider the solar rotation problem. The fact that the p modes sample the centre of the sun very poorly means that there is very little information contained in the data about the rotation of the sun in that region. This results in inversions which conform almost exactly to the smoothing constraint near the centre of the sun. For example, figure 3.3 shows the solar equatorial angular velocity inferred from real data using first order (first derivative) smoothing. The recovered solution is actually flat for $r < 0.2R_\odot$, with a value of about 493nHz. As λ is varied the point at which the solution is forced to become flat changes slightly, but, more importantly, the constant value of the angular momentum near the centre also varies. This means that the variation of the solution near the centre of the sun contributes to TE_0. However, we know that this part of the solution is nonsense, and so including it in the definition of TE_0 is unhelpful, to say the least. In fact, using the definition (3.18) often gave rather poor inversion for this very reason. We therefore modify (3.18) to

$$TE(\lambda) = \|\hat{\mathbf{f}}(\lambda) - \mathbf{f}_0\|^2_{q,w}, \tag{3.19}$$

where the subscript w represents the fact that the norm is now weighted so that the contribution to TE from the solution in the region $0 \leq r \leq 0.2R_\odot$ is zero. This modified definition does give a very good value of λ. Henceforth, the estimate $\hat{\lambda}_{TE}$ is implicitly identified with λ_{OPT}.

The TE method for choosing λ is practically useless (because it requires knowledge of $\mathbf{f}$, the true solution). However, the identification of $\hat{\lambda}_{TE}$ with λ_{OPT} permits the first two methods to be assessed by comparing their distributions and correlations with $\hat{\lambda}_{TE}$. This is the subject of the next section.

3.4 Assessing the EDF and GCV methods

Using artificial data created by substituting a known function into (3.1) and kernel functions calculated from a solar model (*cf.* equations (3.2) and (3.3)) the efficiency of EDF and GCV is examined. The procedure used for this testing was as follows:

1. Choose a function $f(r)$ to use as the true solution. This must have sufficient structure to provide a reasonable test of the inversion algorithm (i.e. it should not fit too well with any smoothing constraint that may be applied. The solution chosen was given by

$$f(r) = 450 + 50\ e^{-3r/R_\odot} + 30\cos\frac{3\pi r}{R_\odot} + 20\cos\frac{8\pi r}{R_\odot} + 10\cos\frac{20\pi r}{R_\odot} \tag{3.20}$$

(in units of nHz). This particular form was chosen because it had a vague resemblance to the inversion of the real data and had structure on quite small scales, providing a test of any inversion algorithm.

2. Take $f(r)$ and, using a set of kernels, calculate the data corresponding to it. The kernels used here were the GONG kernels provided by Gough for use in the GONG 'Hound and Hare' exercise (GONG Newsletter No. 9). The data set thus constructed consisted of 810 modes (i.e. $p = 810$). with $5 \leq l \leq 60$ and was determined by the overlap between the Libbrecht data set and the set of GONG kernels. The noise

levels (i.e. $\{\delta_i; i = 1, \ldots, 810\}$) used for the renormalizations (3.10) and (3.11) were determined from the Libbrecht data set by calculating the standard deviation of the errors on $a_1 + a_3 + a_5$. (See Durney 1990 for a brief explanation of these coefficients. $a_1 + a_3 + a_5$ give the data appropriate to an equatorial inversion.)

3. The (unrenormalized) data was used to create 1000 noisy data sets with the errors on the ith data point taken from a gaussian distribution with mean zero and variance $\sigma_i^2 = (\sigma_0 \delta_i)^2$. The random number generator GASDEV, given in *Numerical Recipes* was used to generate the errors. Three values of σ_0 were used: 0.01, 0.1 and 1, the assumption being that the data will improve with time and there will be a need to examine the performance of EDF and GCV for helioseismological inversion with much lower noise levels than is currently possible.

4. Each of the 1000 data sets (for each value of σ_0), was inverted using regularized least squares with PCD and first order smoothing. The discretization was performed with 150 bins of equal width (i.e. $q = 150$). This was chosen because it is quite simple to implement and because the true solution was clearly not compatible with zeroth order smoothing (having a mean value of about 460nHz, and a maximum deviation from this of around 50nHz). The three estimators $\hat{\lambda}_{EDF}$, $\hat{\lambda}_{GCV}$ and $\hat{\lambda}_{TE}$ were calculated for each data using a grid search with 30 equal width bins in $\ln \lambda$ for λ between the largest and smallest non-zero elements of the matrix $D(0)$ (equation (3.8)). (The motivation for these limits is that smaller values must be unacceptable because of the data noise, and larger values would result in a solution in which the smoothing parameter dominated every spectral component of the solution, so that the bias level would be unacceptable.) The grid search looks for intervals which bracket a root or a minimum. The *Numerical Recipes* routines ZBRENT and BRENT were then used to perform rootfinding or minimization over the interval which bracketed the best solution. (Sometimes $GCV(\lambda)$ has multiple minima. The global minimum was chosen in this case.)

5. The resulting values of the three estimators were used to find the mean and variance of their distributions and the correlation coefficients between the first two estimators and the last.

	$\sigma_0 = 0.01$	$\sigma_0 = 0.1$	$\sigma_0 = 1.0$
λ_{EDF}	−3.928	−2.746	−0.651
λ_{GCV}	−5.543	−3.589	−2.263
λ_{TE}	−5.818	−4.012	−3.017

Table 3.1: Means of $\log_{10} \hat{\lambda}$ for the three estimators considered here, for each data noise level, σ_0.

	$\sigma_0 = 0.01$	$\sigma_0 = 0.1$	$\sigma_0 = 1.0$
λ_{EDF}	0.268	0.706	0.904
λ_{GCV}	0.695	1.070	1.622
λ_{TE}	1.257	1.548	2.027

Table 3.2: Standard deviations of $\log_{10} \hat{\lambda}$ for the three estimators, for each data noise level.

6. GCV and EDF were used in an inversion of the $a_1 + a_3 + a_5$ splitting coefficients from the Libbrecht (1989) data. The inversion procedure was as described above (with $\sigma_0 = 1.0$).

The results are presented in §3.5.

3.5 Results

Table 3.1 and table 3.2 show the means and variances, respectively, of the distributions of the three estimators for each value of σ_0. The results for TE and GCV may be taken at face value. There was a problem with EDF which makes the results for this method slightly less credible, and this was that EDF seemed to be so sensitive to the estimate of the data error variance that the truncation error contributed sufficiently to the noise to make $EDF(\lambda)$ positive definite (i.e. the noise estimate, σ_0, which was the actual variance if the errors added to the data, was too small – see equation (3.14)), so no solution to (3.14) could be found. From past experience with the kernels and method used here the truncation error was determined to contribute an effective increase to the added error variance of approximately 7.2×10^{-7}. Adding this (*tiny*) correction to σ_0 ensured that $EDF(\lambda)$ usually had a solution, although the results suggest that the noise estimate is perhaps a little too large, because $\hat{\lambda}_{EDF}$ is generally too large compared to $\hat{\lambda}_{TE}$. In fact, part of the problem with EDF is that its calculation involves a subtraction of two quantities that, for λ in quite a large range around $\hat{\lambda}_{EDF}$, are almost equal. Its calculation is therefore prone

	$\sigma_0 = 0.01$	$\sigma_0 = 0.1$	$\sigma_0 = 1.0$
λ_{EDF}	988	549	577
λ_{GCV}	693	769	761
λ_{TE}	945	966	955

Table 3.3: Number of times each method for choosing the smoothing parameter actually succeeded in returning a value (out of a thousand trials), for each data noise level.

	$\sigma_0 = 0.01$	$\sigma_0 = 0.1$	$\sigma_0 = 1.0$
λ_{EDF}	-0.696×10^{-2}	-0.192×10^{-1}	-0.607×10^{-1}
λ_{GCV}	-0.551×10^{-1}	-0.547×10^{-1}	-0.128

Table 3.4: Spearman's correlation coefficient of the two estimators $\hat{\lambda}_{EDF}$ and $\hat{\lambda}_{GCV}$ with $\hat{\lambda}_{TE}$ for the three different values of σ_0.

to cancellation error, which may be quite considerable owing to the number of floating point operations necessary to calculate $R(\lambda)$ and $T(\lambda)$. For these reasons the EDF method was not always successful in providing an estimate of λ_{OPT} (the success rate of EDF was 99%, 55% and 58% for $\sigma_0 = 0.01$, 0.1 and 1.0, respectively – see table 3.3), so the means and variances calculated for the EDF method use fewer points than for the GCV and TE methods. Note, though, that these methods, too, were fallible. Table 3.3 shows the number of successes in selecting the smoothing parameter with each of the methods (recall that there were 1000 trials). The TE method was 95% successful for every value of the data noise level (although the value returned was not always terribly impressive – see below). The failures here are a result of a minimization procedure that is not quite robust enough to deal with some of the pathologies arising from numerical error, particularly in relation to the effect of bias on the solution nearer the centre of the sun. GCV was successful about 70% of the time for the smallest noise level ($\sigma_0 = 0.01$), but this rose to 76% for higher noise levels. This improved performance of GCV with higher noise levels, in contrast to the deterioration of EDF, is a typical feature of these methods for choosing the smoothing parameter.

Figures 3.1, a) ,b) and c) show the distributions of $\log_{10} \hat{\lambda}$ for the three estimators for $\sigma_0 = 0.1$ plotted in terms of $\ln \hat{\lambda}$. (The distributions for other values of σ_0 look similar in shape, but with different means and variances.) All three estimators have the property that they occasionally return an anomalously small value of $\hat{\lambda}$. In practice, it is easy to

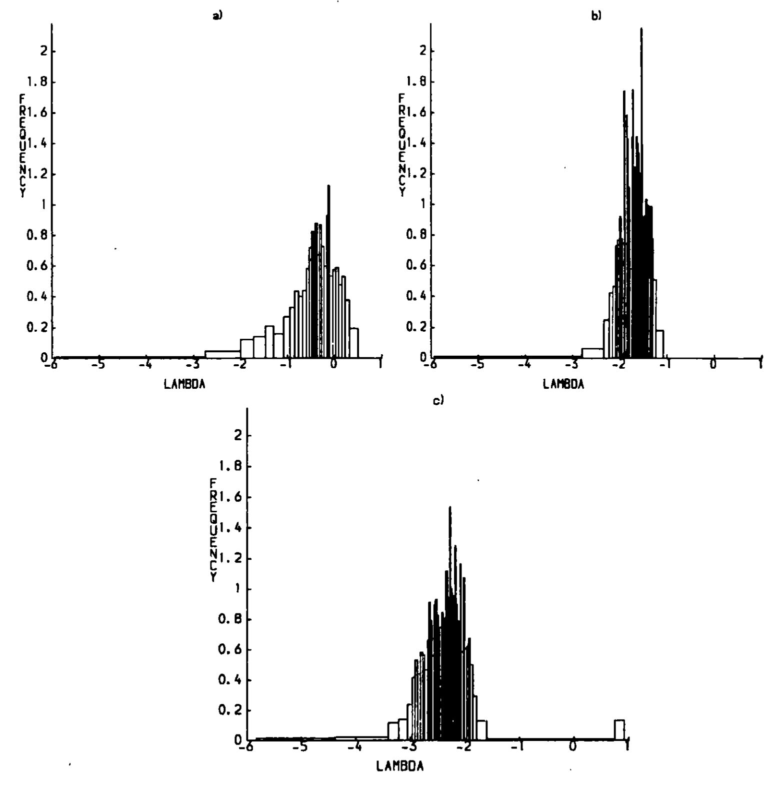


Figure 3.1: Histograms showing the distributions of the values of $\log_{10}\hat{\lambda}$ returned by the three methods for choosing the smoothing parameter, for the noise level $\sigma_0 = 0.1$. Each box contains the same number of points, i.e. the boxes have equal *areas*, but variable widths. a) $\log_{10}\hat{\lambda}_{EDF}$, b) $\log_{10}\hat{\lambda}_{GCV}$, and c) $\log_{10}\hat{\lambda}_{TE}$.

see that such values are incorrect and discard them, using some other method to choose λ, so they ought really to be counted as failures of the method. This would improve the means and variances in tables 3.1 and 3.2, at the expense of decreasing the success rates given in table 3.3. With EDF the anomalous values are quite likely to be due to the general sensitivity of the method, but with GCV experience suggests that the methods themselves are very reliable, and that the failures are due (in part, at least) to a weakness of the minimization algorithm. It is interesting that the TE method occasionally returns an unusually small value of $\hat{\lambda}_{TE}$. Again, this seems to be related to problems with the robustness of the minimization routine. There are only a few cases of this, so it probably has little effect on the statistics given.

Figures 3.2, a) and b) show the (lack of) correlations between the natural logarithms

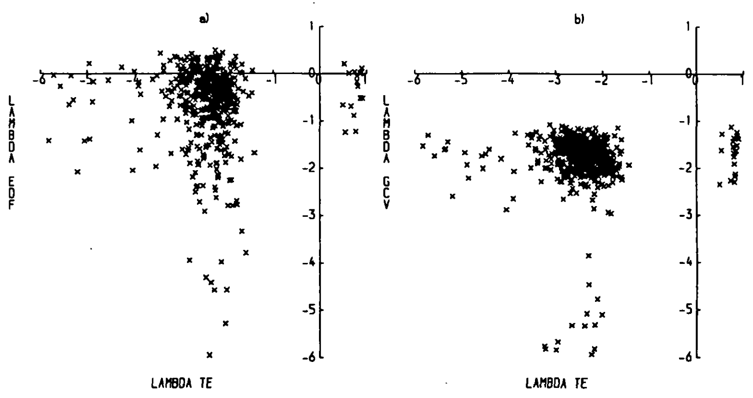


Figure 3.2: Scattergrams illustrating the poor correlations between the estimated values of $\log_{10}\hat{\lambda}$ obtained from EDF or GCV, and the value obtained from TE, for the noise level $\sigma_0 = 0.1$. a) shows the correlation between $\log_{10}\hat{\lambda}_{EDF}$ and $\log_{10}\hat{\lambda}_{TE}$, whereas b) shows the correlation between $\log_{10}\hat{\lambda}_{GCV}$ and $\log_{10}\hat{\lambda}_{TE}$

of $\hat{\lambda}_{TE}$ and the other two estimators. These are clearly very poor. Table 3.4 shows the values of Spearman's correlation coefficient (see *Numerical Recipes*, §14.6) for the two estimators with $\hat{\lambda}_{TE}$ for each value of σ_0. Pearson's and Kendall's coefficients (*Numerical Recipes*, §14.5 and §14.6) were also calculated, and they were, of course, similarly low.

Finally, figure 3.3 shows an inversion of the data of Libbrecht (1989), using the estimate of λ returned by GCV. The EDF method also returned an estimate, but this was clearly rather too large and resulted in considerable oversmoothing. (In fact, $\hat{\lambda}_{EDF} = 5.866$, which compares to $\hat{\lambda}_{GCV} = 1.310 \times 10^{-2}$.)

3.6 Conclusions

The trends that are generally found in the efficiency of EDF and GCV are that EDF (when a good noise estimate is available) works reasonably well for low noise levels, while GCV gives rather superior results for higher noise levels. Moreover, the analysis presented here can be applied more generally and with greater detail to provide a more thorough assessment of any method of choosing λ, including the trade-off curve method. Also, if several good methods can be found, the possibility exists of finding some combination of them which improves on all of them individually (by maximizing their correlation with λ_{OPT} over all linear combinations of the estimators, for example).

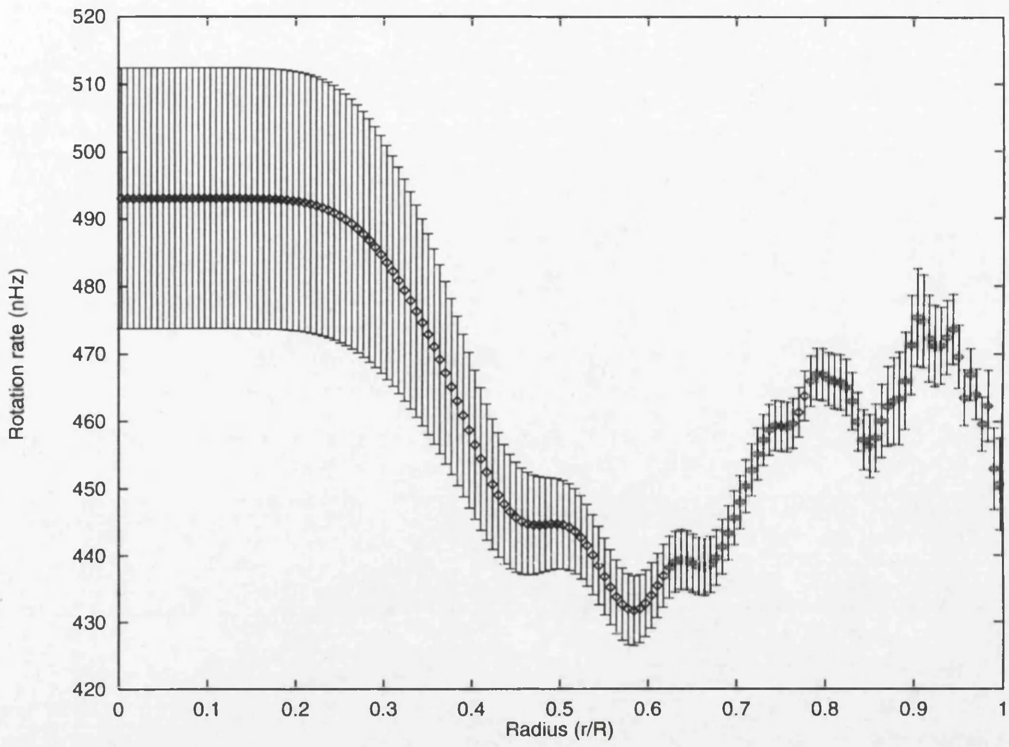


Figure 3.3: An inversion of the data of Libbrecht (1989) for the equatorial value of the solar internal rotation (i.e. an inversion of $a_1 + a_3 + a_5$), using the estimate of λ returned by GCV: $\hat{\lambda}_{GCV} = 1.310 \times 10^{-2}$. The error-bars are 1-$\sigma$ pseudo-confidence intervals.

The results presented here show that it is possible to find methods which provide a good choice for the smoothing parameters in inversions. For, while the performance of EDF left a lot to be desired, GCV gave excellent results. To illustrate this more clearly, study figure 3.4, which shows inversions of the artificial data corresponding to the solution (3.20) for a *single* noise realization (with noise variance $\sigma_0 = 0.1$, to allow comparison with figure 3.1), and for several different values of the smoothing parameter, λ. The true solution is also shown. The range of values of λ chosen allows the significance of the differences between the distributions of the three estimators of the smoothing parameter shown in fig. 3.1 to be understood more clearly. As a guide to interpreting fig. 3.4, recall that the smoother solutions correspond to the larger values of λ. The smallest λ shown is $\lambda = 1 \times 10^{-3}$, which corresponds to the smallest value of the smoothing parameter that would normally be returned by the estimators. It is clear that this value of λ undersmooths somewhat (observe the small, rapid oscillations near $r/R_\odot \approx 0.9$). This accords

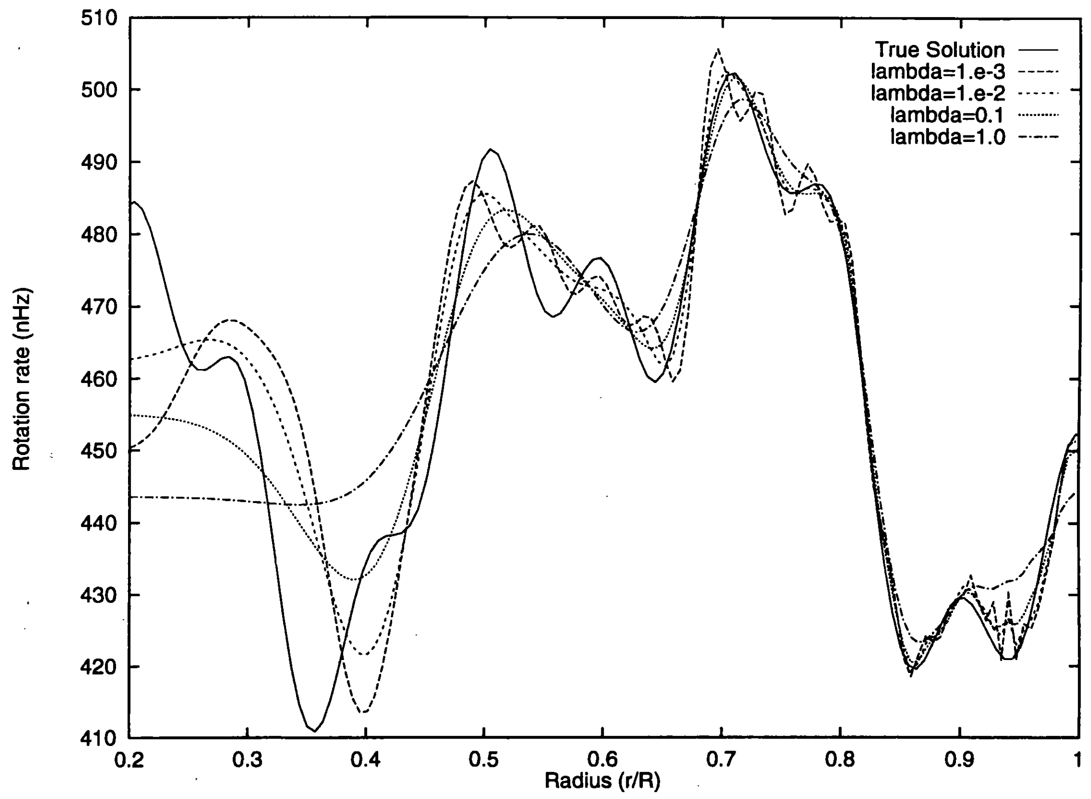


Figure 3.4: Inversions of noisy data corresponding to the artificial solution (3.20) for a single noise realization ($\sigma_0 = 0.1$) with several choices of the smoothing parameter, λ, as shown in the key. The smoother solutions in the figure correspond to larger values of λ, of course.

with our expectations based on figure 3.1, since this value of λ is smaller than the values generally returned by all the methods. The inversion with $\lambda = 1 \times 10^{-2}$ is much less oscillatory, and follows the variations of the true solution fairly well (although there are some local differences due to the data errors). This is just about the closest of the four estimates of the solution plotted in figure 3.4, and, again, this reflects the information in figure 3.1, since $\lambda = 1 \times 10^{-2}$ corresponds (approximately) to the peak of the distribution of the optimal value of the smoothing parameter, λ_{TE}. Most importantly for the work here, though, $\lambda = 1 \times 10^{-2}$ corresponds to the peak of the λ_{GCV} distribution in figure 3.1b), which means that the GCV method is likely, in general, to give good results. It is quite easy to see from fig. 3.4 that the smoothest solution, which corresponds to $\lambda = 1$, oversmooths considerably everywhere, and is a less satisfactory solution than those for smaller values of λ. Inspection of figure 3.1a) shows that this value of λ is close to the peak of the distribution of λ_{EDF}, which indicates that, in general, the EDF method will (with the problem considered here, at least) tend to oversmooth and give results that are unsatisfactory.

The only flaw with GCV is that its correlation with λ_{OPT} is poor, but the rather small variance of its distribution makes this criterion less important. The theory behind GCV says that it is asymptotically optimal (as $p \to \infty$), so that the only way to improve on GCV (without a perfect error estimate) would be to find an estimator which converges to λ_{OPT} faster than GCV. With the large data sets soon to be available via the GONG project it will be difficult to surpass GCV for the objective choosing of smoothing parameters in helioseismological inversions.

Chapter 4

Studying the Resolution of Data

4.1 Introduction

In this chapter the meaning and importance of resolution in the solution of inverse problems are discussed, and methods for quantifying the resolution achieved in a particular inversion, and for quantifying the maximum resolution that could be achieved, for any given level of data errors, are presented.

The term resolution cannot be defined in an absolutely rigorous, quantitative way, but it is obvious from an intuitive point of view that the meaning of the term is essentially:

> the extent to which the values of the solution to an inverse problem at two nearby points can be distinguished, and, more particularly, the distance two points must be apart before it is possible to make statements about significant differences in the values of the solution at those two points with any degree of confidence.

There are two senses in which gauging resolution is important. In a general sense, in the absence of any specific data and armed only with a knowledge of the form of the forward problem, and perhaps some thoughts on what data error levels are of relevance, it is possible, and often necessary, to make statements about the best resolution that could be achieved in an inversion, purely on the basis of the extent to which the kernel functions would act differently on two putative solution functions that differ only in small regions. Often, for example, it may be important to know the resolution that could be achieved with some given collection of solar oscillation modes and some particular data noise level before any data has been acquired, perhaps to enable the designers of an observing project to concentrate their energies on some set of modes in preference to any other. Or it may be desirable to determine whether some set of data actually has the power to accept or reject some model of, say, the solar rotation, before performing an inversion, the results of which may turn out to be inconclusive.

Aside from this, though, there is the question of determining the resolution that actually has been achieved in a particular inversion. For a specific inversion it is necessary to specify both the resolution achieved (which will, in general, be less than optimal) and the errors on the recovered solution values. Without such information the solution values are essentially meaningless, and they certainly cannot be interpreted and compared with other inversions.

There is no single, unequivocal definition of 'resolution', although there are accepted standards (such as the width of the averaging kernels in OLA inversions – see §1.8 – although even this is not really absolute as it depends on the choice of 'width' functional used), by which any other definition will be judged. Several alternative approaches to assessing resolution in helioseismic inverse problems are presented in §4.2. These can be broken down loosely into two classes: those that are intimately connected with inversion procedures, can be used in those procedures to estimate the resolution attained, and those which are essentially independent of any such procedure and rely on studying the data that would be obtained if the true solution were one of some set of functions depending on a small number of parameters. The former type are very useful when performing inversions is also one of the goals, but they tend to be more complicated and computationally intensive. The latter group are very simple to use, but cannot give information about the resolution achieved in specific inversions.

One of the methods for assessing resolution described in §4.2 (and which lies in the class of inverse-procedure related methods) uses the correlation profile of the errors on the solution to determine the resolution length, in much the same way as the averaging kernels at any point can be used for the same purpose. Strangely, despite its simplicity, this method seems to have been largely overlooked in the past. It turns out to have some considerable advantages over the averaging kernel definition of resolution, most noticeably in terms of speed of calculation, although it is not entirely free from problems.

Having outlined the alternative definitions of resolution in §4.2, they will be used in §4.3 to examine the resolution that can be achieved with the splitting data of Libbrecht (1989), for the error levels provided with the splitting coefficients. Section 4.4 will then discuss the significance of these results for the resolution achievable in real inversions with this data, such as those in chapter 5.

4.2 Quantifying Resolution

Several alternative ways of measuring 'resolution' in helioseismic (or indeed any similar) inverse problems are briefly described and explained, as a prelude to §4.3 where they will be applied in a real helioseismic problem.

4.2.1 Sines and Cosines

One way to think of resolution is to consider the sharpness of features that could be distinguished in an inversion. It is clear that if the inversion can recover sharp features in a solution, then the resolution is good. But sharp features in a function correspond to the presence of large, short-wavelength components in the Fourier expansion of that function. As the inverse problem is linear, the Fourier components can be treated separately. A particular mode set, with some error levels assumed, can be seen to give rise to high resolution if, when the true solution is assumed to be a sine function with short wavelength, this solution can be recovered adequately in an inversion. What this means really is that the short-wavelength sine function *does* affect the data above the level of the data noise, for if it does then its presence can certainly be determined, and if it does not there is no way we could ever know it was there. The idea of short-wavelength sine functions being detectable is related to the Riemann-Lebesgue lemma (see Craig and Brown 1986, §4.2).

This method is 'global' in the sense that the information given necessarily relates to the whole range of the inversion, rather than just the region around individual points. Obviously, this is a result of the fact that the Fourier components are homogeneous across the range of the inversion, that is, each component is sort of the same everywhere. In problems where the resolution achieved is similarly uniform across the range of the inversion this would not matter, and Fourier components could be used to assess resolution. However, helioseismic inverse problems are certainly not in this category, for it is well known that the solar p-mode frequencies contain a great deal of information about the interior of the sun nearer the surface (and thus give excellent resolution there) and very little information about the solar core (meaning poor resolution there). Using Fourier components to gauge resolution is just not adequate for helioseismic inversions.

It might be possible to rescue the situation by considering other bases that have some property of localization, such as *wavelet* bases (*Numerical Recipes*, §13.10), but this will not be pursued here.

4.2.2 Delta-functions

Another way to produce a measure of resolution that is local, and therefore appropriate to helioseismic inverse problems is to come out of Fourier space altogether and look at localized features in 'real' space. The extent to which two nearby localized features can be distinguished obviously gives a measure of resolution. The simplest and most localized 'feature' possible is a dirac delta-function, so here we consider the measure of resolution defined by comparing the data that would result from two different delta-function true solutions at nearby points.

First, it is obvious that the (error-free) data that would arise if the true solution was proportional to a delta-function at some point r_0, i.e. $f(r) = \alpha_0\delta(r - r_0)$, is

$$g_i = \int_0^{R_\odot} k_i(r)\alpha_0\delta(r - r_0)\,dr = \alpha_0 k_i(r_0). \tag{4.1}$$

The difference between this data set and that for a different delta-function solution $f(r) = \alpha_1\delta(r - r_1)$ at another point r_1, is just

$$\delta g_i = \alpha_1 k_i(r_1) - \alpha_0 k_i(r_0). \tag{4.2}$$

Obviously, if the data contains noise, this difference could only be detected errors if the δg_i are larger than the typical size of the errors. A little more precisely, if the errors on the ith data point have standard deviation σ_i, the difference between the two delta-function true solutions can only be detected at, say, the 2σ-level if, on average, the δg_i are at least twice as large as the σ_i, that is, if

$$\frac{1}{m}\sum_{i=1}^{m}\left(\frac{\delta g_i}{\sigma_i}\right)^2 > 2. \tag{4.3}$$

By varying α_0, α_1, r_0 and r_1 to find the points where the limit in (4.3) is reached, the ability of the data set used to distinguish between different such solutions can be found. Since we are interested here primarily in resolution, which is the ability to distinguish features at different positions, the variation with α_0 and α_1 will not be considered. We will, however, retain α_0 and α_1 to allow renormalization of the data, which is necessary because, as the errors we are considering are absolute, choosing a very small amplitude for the delta-functions would mean that the δg_i would be swamped by the noise wherever the delta-functions are. To overcome this problem, the normalizing constant α is chosen so that the norm of the data vector, $g^2 = \sum_{i=1}^{m} g_i^2$, at each point is the same as the norm of the data vector for the $a_1+a_3+a_5$ inversions of the Libbrecht (1989) data, $G^2 = 9.93\times 10^6$. At any point, r_0, then, α_0 is given by

$$\alpha_0 = G\left(\sum_{i=1}^{m} k_i(r_0)^2\right)^{-1/2}.$$

In §4.4, when this definition of resolution is tested on a real helioseismic problem, the preceding normalization will be assumed, and the criterion for accepting that a difference between the delta-functions is detectable will be that the difference between the two data sets is significant at the 2σ-level, i.e. that (4.3) holds.

4.2.3 Step-functions

Precisely the same principles apply to the use of step functions (in this chapter, the term step-function will be taken to mean a piecewise-constant function with a *single* jump, or step, in it) to study resolution as applied to delta-functions. The steps in step-functions are localized features, just like delta-functions, and so they can be used to probe the local resolution. Step-functions may in some ways be even more appealing than delta-functions for the study of resolution, because we often expect to see steps in the solutions of inverse problems, but a delta-function-like spike is much less realistic. In fact, the inversions for the equatorial rotation rate shown in chapter 5 show very strong evidence of a sharp feature at the base of the convection zone ($r/R_\odot \approx 0.73$). This feature is almost certainly much narrower than the resolution length at that point (dynamo theory suggests that the gradient there might be quite steep). So, using step-functions as a probe of resolution may well give better results, than delta-functions because a step-function bears a closer resemblance to the features we expect to find in real solutions of the inverse problem.

Step-functions are a little more complicated than delta functions, because they are parametrized with three parameters, the value of the step-function on either side of the step, and the position of the step. However, again we are interested primarily in resolution, so the parameter of most relevance is the position of the step. We fix the values of the function on either side of the step in the following way:

- as the data is rather insensitive to the value of the solution near the centre of the sun, fixing the value below the step to some constant value should not be too restrictive, or cause too many problems. For physical reasons, related to the possible existence of a very sharp feature in the solar equatorial rotation rate (suggested by the inversions of chapter 5), this value was taken to be 440 nHz, which is approximately the value of the solar rotation rate near the base of the convection zone. The reasons for this choice are explained below.

- just as with the delta-function method, it is necessary to fix the norm of the data to avoid having such small data values that the data itself is swamped by the noise, so that there is effectively zero reolution everywhere. This will fix the value of the step-function above the step.

The reason for 'normalizing' the step functions in this way is to ensure that when the step is at the base of the convection zone it looks as much like the solutions shown in chapter 5 as possible. That way the definition of resolution it provides near the base of the convection zone is more likely to have a direct physical relevence, both quantitatively and qualitatively.

It only remains to describe how the above normalization fixes the value of the step-function. If we denote a general step-function with step at r_0, value a below r_0, and value b above it, by $S(a,b,r_0;r)$, the value of the ith data point will be given by

$$g_i = \int_0^{R_\odot} k_i(r) S(a,b,r_0;r)\,dr = a\int_0^{r_0} k_i(r)\,dr + b\int_{r_0}^{R_\odot} k_i(r)\,dr \tag{4.4}$$

The norm (squared) of the data vector is then

$$g^2 = \gamma_1 b^2 + \gamma_2 ab + \gamma_3 a^2,$$

where

$$\begin{aligned}
\gamma_1 &= \sum_{i=1}^{m}\left(\int_{r_0}^{R_\odot} k_i(r)\,dr\right)^2, \\
\gamma_2 &= 2\sum_{i=1}^{m}\left(\int_0^{r_0} k_i(r)\,dr\right)\left(\int_{r_0}^{R_\odot} k_i(r)\,dr\right), \\
\gamma_3 &= \sum_{i=1}^{m}\left(\int_0^{r_0} k_i(r)\,dr\right)^2.
\end{aligned}$$

Demanding that $g^2 = G^2$, the norm of the Libbrecht data vector, as in §4.2.2, leaves a quadratic equation for b (remembering that a is fixed by the constraint outlined above).

This can easily be solved for b. In the event that there are two possible solutions, the positive solution should be taken, since that is more in accord with the observed solutions.

Having completely fixed the values of the step-function for any position of the step we can now say that two steps, $S(a_0, b_0, r_0; r)$ and $S(a_1, b_1, r_1; r)$, can be resolved if the differences, δg_i, between their respective data values again satisfy (4.3). The argument is exactly as before for the delta-functions. Again, finding, for any r_0, the limiting values of r_1 for which (4.3) just holds defines the resolution lengths at r_0.

4.2.4 Averaging Kernels

The use of averaging kernels in OLA methods (see §1.8) has long been an accepted way to define the resolution achieved in an inversion. Although it is most often used in OLA inversions the concept of an averaging kernel applies equally well to RLS methods (Christensen-Dalsgaard *et al.* 1990). The RLS averaging kernels are derived as follows. We know from equation (1.59) and from §2.2.5 that the solution *vector* in an RLS inversion is given in terms of the data vector, kernel matrix and smoothing matrix by

$$\mathbf{f} = (H^T H + \lambda C)^{-1} H^T \mathbf{g}. \tag{4.5}$$

The discretization procedure told us that the solution function is related to this solution vector by an equation of the form (2.23), i.e.

$$f(r) = \sum_{l=1}^{n} f_l \phi_l(r) \equiv \phi^T(r)\mathbf{f}, \tag{4.6}$$

where vector notation has been introduced to simplify the equations here. Using (4.5) in (4.6) gives the solution function in terms of the data

$$f(r) = \phi^T(r)(H^T H + \lambda C)^{-1} H^T \mathbf{g}. \tag{4.7}$$

We know exactly how the data are related to the solution (this is just the forward problem, (1.32)). Using this in(4.7) tells us that

$$f(r) = \int_0^{R_\odot} \left\{ \phi^T(r)(H^T H + \lambda C)^{-1} H^T \mathbf{k}(r') \right\} f(r')\, dr'$$

(where the kernels have also been written in vector notation). The factor in braces says exactly how the recovered solution depends on the true solution. It gives a weighted average of the true solution at every point of the recovered solution. Hence, the quantity

$$K(r, r') \overset{\text{def}}{\equiv} \phi^T(r)(H^T H + \lambda C)^{-1} H^T \mathbf{k}(r') \tag{4.8}$$

is an *RLS averaging kernel.* Generally, like the OLA averaging kernels $K(r, r')$ has a peak at or near $r' = r$ for any r, and the width of the peak determines the resolving length. There are many ways to define the width of the peak (full-width at half maximum, quartiles, etc.). In §4.3 the definition of width that was taken was a *two-sided* one. Essentially, the two points, r_l and r_u, above and below the point of interest, r, at which the averaging kernel falls to half its value at r are used to define the lower and upper resolution lengths as $l_l = r - r_l$, and $l_u = r_u - r$. This gives a better indication of the relationship between the recovered solution and the true solution. It should be noted, at this point, that the averaging kernels depend on the value of the smoothing parameter (*cf.* figure 4.1), so their width is not an absolute quantity. In order to define the resolution it is necessary first to choose the smoothing parameter. This is rather a circular argument, because it could be said that the choice of smoothing parameter should be determined by considering the stability and resolution of the inversion (the trade-off method), rather than the other way around. Here, though, the view is taken that assessing the resolution of real inversions is an important aspect of this work, so to use the value of the smoothing parameter chosen in an inversion is perfectly valid. This statement also applies to the correlation profile method to be described shortly. The averaging kernel and the correlation profile results in §4.3 were obtained in the inversion illustrated in figure 5.4(a) (discretization 1) using the value of λ chosen by the GCV method in that inversion.

4.2.5 Correlation Length

Although it rarely seems to have been given much attention, there is a great deal of information about the effect of smoothing and regularization on the independence of, and relationship between, the components of the solution vector in the *covariance matrix* of errors on the solution. This can easily be converted into information about correlations between the values of the errors in the solution function at different points, and thus used to draw conclusions about the effect of the smoothing on the solution function. The idea is that the errors at nearby points *must* be correlated by the smoothing constraint, or the solution would never be smooth. At points further apart there is no reason for the errors to be related to one another in general, so the errors at points further apart should normally be less well correlated. Picking one point, r, and plotting its correlation with the errors at other points, r', a profile should be obtained that has a high value near r (highly correlated errors) and falls to smaller values for r' further away from r. Such a profile has many of the properties of an averaging kernel, and so the fact that its formulation is explicitly related to the extent to which the smoothing constraint forces the solution to be smooth suggests that it might make a good measure of resolving length. In this section the *correlation profile*, $c(r,r')$ will be derived, and in §4.3 its similarities to the RLS averaging kernels will be described and investigated.

It is a simple matter to show that the covariance matrix of errors on the solution vector, which is defined by

$$C_{cov} \stackrel{\text{def}}{\equiv} \mathbb{E}\left\{(\mathbf{f}-\bar{\mathbf{f}})(\mathbf{f}-\bar{\mathbf{f}})^T\right\},$$

(where a bar denotes the mean value, and $\mathbb{E}$ denotes the expectation value, taken over realizations of the data errors) can be written in terms of the kernel and smoothing matrices as

$$C_{cov} = (H^T H + \lambda C)^{-1} H^T H (H^T H + \lambda C)^{-1}. \tag{4.9}$$

Since the solution function is related to the solution vector as in (4.7) it is possible to use the covariance matrix to calculate the correlations between the errors on the solution at different points. We define the correlation profile to be

$$c(r,r') \stackrel{\text{def}}{\equiv} \mathbb{E}\left\{(f(r)-\bar{f}(r))(f(r')-\bar{f}(r'))\right\}. \tag{4.10}$$

The mean, $\bar{f}(r)$, of $f(r)$ is easily seen from (4.6) to be given by

$$\bar{f}(r) = \phi^T(r)\bar{\mathbf{f}},$$

so that $c(r,r')$ can be written (after removing the basis functions of the discretization from the expectation value because they are obviously independent of the data errors)

$$c(r,r') = \phi^T(r)\mathbb{E}\left\{(\mathbf{f}-\bar{\mathbf{f}})(\mathbf{f}-\bar{\mathbf{f}})^T\right\}\phi(r') = \phi^T(r)C_{cov}\phi(r'). \tag{4.11}$$

Observe that the expression for the covariance matrix (4.9) in terms of the kernel and smoothing matrices gives the expression (4.11) a very similar appearance to the RLS averaging kernel in (4.8). This is one more reason for expecting that the width of the correlation profile might provide a useful measure of resolution. In the work of §4.3 the width of the correlation profile was defined exactly as for the RLS averaging kernels (i.e. a two-sided definition was adopted).

In §4.3 the correlation profile used was not exactly the profile derived above, but was instead the normalized profile $\tilde{c}(r, r')$, defined by

$$\tilde{c}(r, r') = \frac{c(r, r')}{\sqrt{c(r, r)c(r', r')}} \tag{4.12}$$

which has the pleasing properties of being symmetric in r and r', and having the value 1 when $r' = r$, as can easily be seen from the form of (4.12).

4.3 Results

Before comparing the measures of resolution obtained by the above methods, we should take time to compare the properties of the averaging kernels and the correlation profiles. Figure 4.1 shows the correlation profiles and the averaging kernel profiles for different values of the smoothing parameter, and for three points within the sun. It is clear that neither type of profile gives a very good measure of resolution for very small λ. They both have narrow central peaks, but the sidelobes are rather large. This means that measuring the resolving width by finding the HWHM of the profile will give completely misleading results. For large λ, too, the profiles do not provide any easy way to measure resolution. They are much less erratic than for very small λ, but they are not at all well localized about the relevant point. For λ lying in the range relevant for inversions of real solar data the profiles are much better, and give a very clear and effective measure of resolution. Note the close similarity between the correponding profiles of each type. Although they do not follow each other exactly, they do have a very similar structure, both being large in modulus in the same regions. The correlation profiles in regions where the resolution is poor (near the centre of the sun) do not perform as well, because they tend to adopt a constant value close to one in such regions, which merely indicates that the smoothing constraint is doing all the work and tying the values of the solution together very rigidly.

Broadly speaking, the relationship between the correlation profiles and the averaging kernels is that, while the averaging kernels often give a cleaner profile, they are prone to exhibiting oscillations away from the central peak, whereas the correlation profiles are generally smoother and less prone to oscillation. This smoothness does mean, though, that they often give a measure of resolution that overestimates the region over which the solution is averaged relative to the averaging kernel estimate of the resolution.

Figure 4.1 is a clear demonstration of the potential relevance of the correlation profile method for measuring resolution in helioseismic, and other, inversions. Having validated this new measure of resolution, it is time to ask what the different ways of measuring resolution outlined in §4.2 have to say about the resolving power of the modes in the Libbrecht data set, and how do their measures of resolution compare with one another. Figure 4.2 plots the value of the resolution determined by the delta-function and step-function methods. The agreement between the different methods is so good that the curves lie almost on top of each other. This is perhaps hardly surprising, as the delta-function can be thought of as the derivative of a step-function, so these two types of solution function are very closely related. There are one or two points to notice about the curves in figure 4.2. Firstly, there is a definite break in the curve at around $r/R_\odot = 0.73$, indicating the position of the base of the convection zone. This shows very vividly the way in which even relatively small details in the solar structure can affect solar rotation inversions. Secondly, note that the curve begins to dip down for $r/R_\odot < 0.6$, despite the fact that the resolution is expected to get worse nearer the centre of the sun (the resolving length should increase). This is almost certainly a result of the normalizations of the delta-functions and step-functions, which will tend to give rise to unwanted effects nearer the solar centre where the kernels are small.

Figure 4.3 displays the values of the resolution determined from the correlation length and averaging kernel methods. The agreement between the curves is rather satisfying. The correlation profile method tends to give large values of the resolving length (particularly the left-side resolving length) near the centre of the sun, but apart from that curves match up very well. Figure 4.3 provides a further indication that correlation profiles can be used to estimate resolution in inversions.

What is less satisfying is the considerable mismatch between the correlation length and averaging kernel measures of resolution and the delta-function and step-function measures. There really is very little agreement either qualitatively or quantitatively between them. It is not clear why this is.

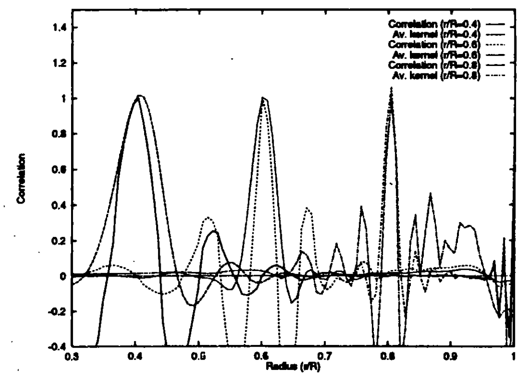
4.4 Conclusions

The correlation profile method has been shown to provide an adquate measure of resolution in inversions. Although it has some problems, its real advantage is that it is much faster and simpler to calculate than the averaging kernel width. This is partly because in

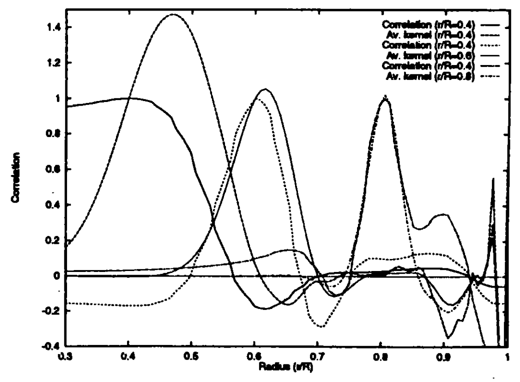
RLS inversions there are more kernel functions than basis functions in the discretization (compare equations (4.11) and (4.8)).

The poor agreement between the measures of resolution obtained from the two different classes of technique for determining resolution (the inversion related methods and the parametric methods) is rather disheartening, and cannot be explained at the present time. In spite of this, it is still possible to make some statements about the resolution achievable through the solar interior (two statements, in fact, depending on the type of method used). For example, an important quantity to ascertain from a physical point of view is the resolving power of the data near the base of the convection zone. This is because the physics near the base of the solar convection zone is believed to play a role in many of the global phenomena observed on the sun, such as the solar cycle. It is therefore very interesting to know how much detail in the solar rotation velocity can be determined from any given data set. The delta-function and step-function measures say that features in the rotation profile as small as $0.0018r/R_\odot$ can be resolved. This seems unreasonably small, especially given that the correlation length and averaging kernel methods both say that features smaller than about $0.04r/R_\odot$ would not be seen in an inversion. The latter figure must be taken as the more realistic assessment of the resolution achievable in inversions of the Libbrecht data.

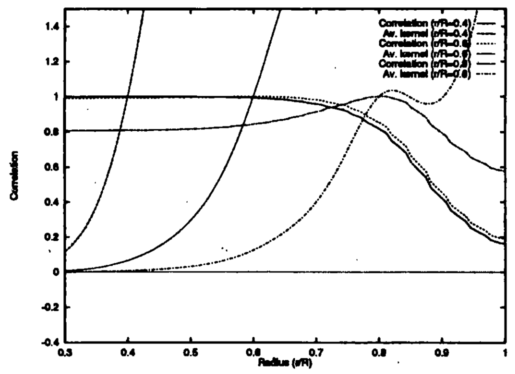
We could, of course, extend this analysis to consider other data noise levels, larger mode sets, smaller mode sets, and so on. The combinations of possible configurations that could be studied is endless. In particular, the work of this chapter could be applied to data sets such as the GONG data set (Harvey *et al.* 1993) to consider the resolution obtainable with that data.



(a)



(b)



(c)

Figure 4.1: Comparisons of the correlation profiles and the averaging kernels with three different values of the smoothing parameter, and for three different points within the sun. Each graph contains the correlation profiles and *normalized* averaging kernels for the points $r/R_\odot = 0.4$, 0.6 and 0.8. The averaging kernels are renormalized by their values at these points. The correlation profiles are the normalized version given in (4.12). Graph (a) is for $\lambda = 10^{-4}$ (much smaller than the value of the smoothing parameter used in the inversion in chapter 5). Graph (b) is for $\lambda = 7.5 \times 10^{-2}$ (which is very close to the value of the smoothing parameter used in chapter 5). Graph (c) is for $\lambda = 10^2$ (much larger than the value of the smoothing parameter used in chapter 5).

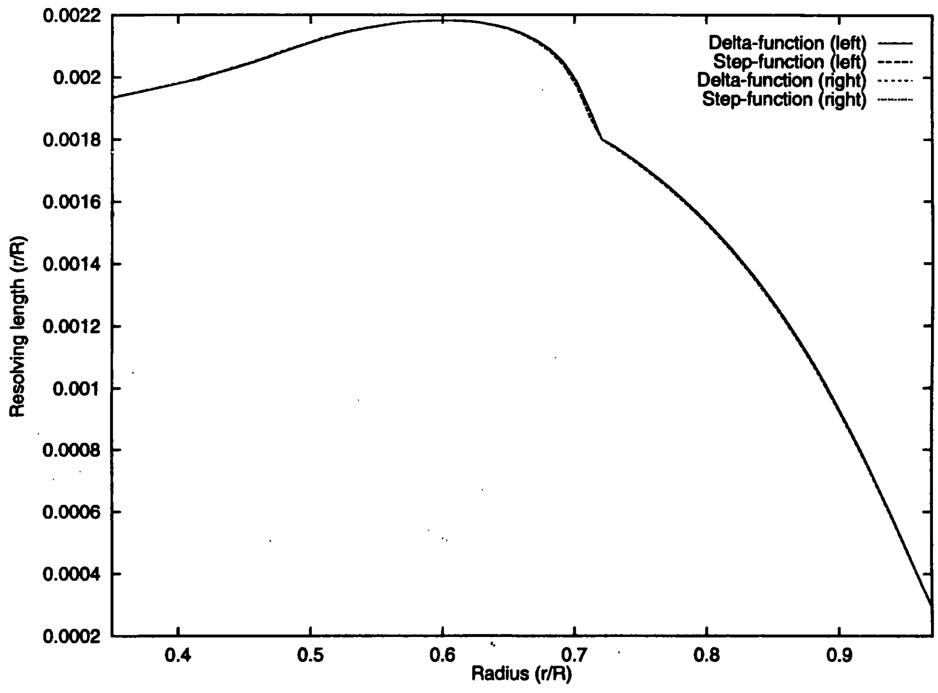


Figure 4.2: The left and right resolving lengths obtained from the delta-function and step-function methods, as functions of the solar radius. Note that the curves are so similar that they can barely be distinuished.

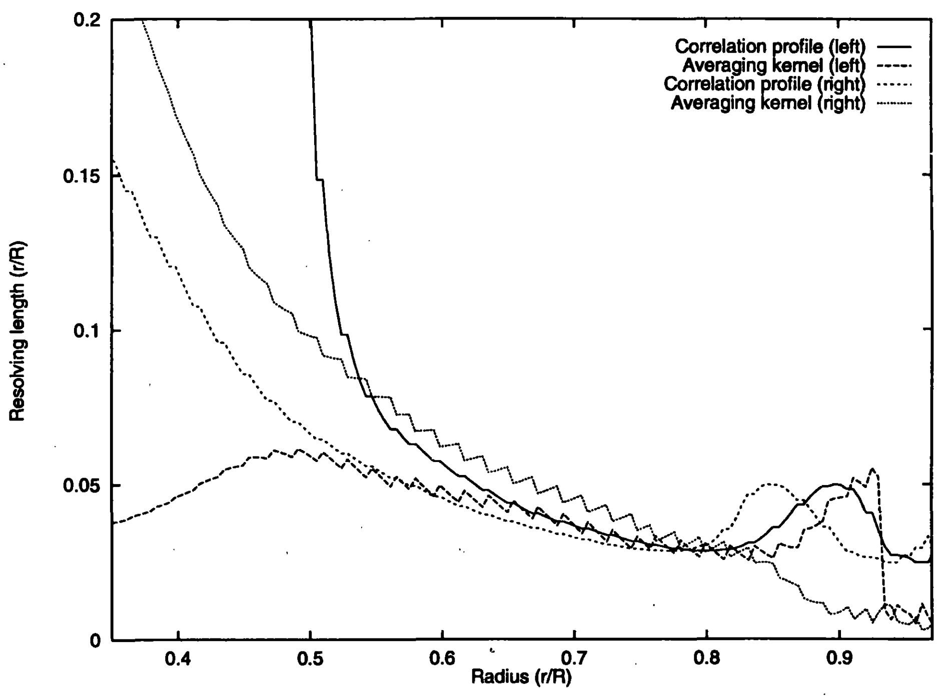


Figure 4.3: The left and right resolving lengths obtained from the correlation length and averaging kernel methods. The saw-tooth effect is merely due to the interplay between the 150 point uniform grid used in the discretization and the 120 points at which the profiles were calculated.

Chapter 5

Results and Conclusions

5.1 Introduction

In this chapter the methods described in earlier chapters are illustrated by applying them to real helioseismic data. The rotational splitting data obtained by Libbrecht (1989) in 1986 is used throughout this chapter. In section 5.2 various inversions of the Libbrecht data are performed using the algorithm of chapter 2, with different discretizations to demonstrate the flexibilty of the algorithm. The 'equatorial' inversion is used for this demonstration. An equatorial inversion is the inversion of the combination $a_1 + a_3 + a_5$ of the old splitting coefficients (obtained by expanding the mode frequencies in a multiplet in terms of legendre polynomials – *cf.* (1.28), where alternative, and more useful, splitting coefficients are obtained from an expansion in terms of Clebsch-Gordon coefficients). The inversion of these coefficients is called an equatorial inversion because it essentially makes use only of the frequencies of sectoral modes (modes corresponding to spherical harmonics with $m = \pm l$), and these modes have a latitudinal dependence that makes them strongly concentrated near the equator of the sun, for all but the smallest l at least, which means that they are much more sensitive to the rotation rate near the equator than elsewhere. The equatorial inversion was chosen to illustrate the effectiveness of the algorithm of chapter 2 partly for simplicity, and partly because the form of the solution to this inverse problem is the one that is most familiar and well-known.

Section 5.3 presents inversions of the Libbrecht data for (quantities closely related to) the Ritzwoller and Lavely (1991) components $w^{(1)}$, $w^{(3)}$ and $w^{(5)}$ (see §1.5.3), of the solar velocity field expanded in terms of vector spherical harmonics as in (1.27). The significance of these inversions for the solar internal rotation rate is discussed.

Finally, section 5.4 briefly summarizes the work in this thesis.

5.2 Using the Inversion Algorithm

To demonstrate the use of the algorithm of chapter 2 several inversions of the same data are performed using different discretizations. The data chosen are the combinations $a_1+a_3+a_5$ of the splitting coefficients in the data set of Libbrecht (1989). The details of these inversions will now be described.

5.2.1 The Data, the Errors and the Kernels

The Libbrecht data contains the $a_1, \ldots, a_6$ splitting coefficients for 872 oscillation modes with $5 \leq l \leq 60$, and $2 \leq n \leq 26$ (see Libbrecht 1989 and Duvall Jr. *et al.* 1986 for an explanation of the a_i coefficients). In the inversions presented in this chapter all these modes were used. The kernel functions for the inversion were calculated from a solar model using up-to-date physics kindly provided by J. Christensen-Dalsgaard. The numerical procedure for performing the calculation used a parallel shooting method to solve the boundary value problem that the equations of adiabatic oscillations form. The integrations were performed using simple fourth-order Runge-Kutta and the 'matching point' for the integrations was taken as the outer point in the model table, at which the outer boundary conditions were applied. See Unno *et al.* (1979), chapter III for a description of the physics, and §17 for a description of methods for solving the equations of oscillation, and see *Numerical Recipes*, chapter 17 for a description of the specific details of the numerical procedures referred to above. The code to perform these calculations was based quite heavily on the formulation of the equations in chapter 17 of Unno *et al.* (1979). The parallel shooting method chosen was perhaps not the most accurate possible method (it might have been better to use the *Henyey relaxation method* described in chapter 17 of Unno *et al.* 1979) but the accuracy is adequate for the purposes of this chapter. The kernels were calculated on the same grid as the model table, which contained 1281 points, with a high density of points near the solar surface.

The splitting coefficients are provided with error estimates, and these were used in the inversion to renormalize the kernel matrix and data vector according to (2.27) and (2.29), respectively. Of course, if the standard deviations of the errors on the coefficients a_1, a_3 and a_5 for some mode are σ_1, σ_3 and σ_5, then the standard deviation of the error on $a_1 + a_3 + a_5$ for that mode is

$$\sqrt{\sigma_1^2 + \sigma_3^2 + \sigma_5^2}.$$

With these renormalizations the errors on the (normalized) data, g_i, become uniform, with standard deviation 1.

5.2.2 Discretization

Three different discretizations are used to perform the inversion. Each had the same number of free parameters (150), to simplify the comparison of the results. The discretizations were:

1. Uniform PCD with 150 bins between $r/R_\odot = 5.905 \times 10^{-2}$ and $r/R_\odot = 1.0007$. The lower limit was determined by the point below which all the kernels are effectively zero ($< 10^{-35}$), and the upper point is the outer point in the model table from which the kernels were calculated.

2. Cosine expansion (an OFE method). A single discretization bin covered the whole range of the inversion, and the solution was assumed to be expanded in terms of the basis functions
$$\phi_j(r) = \cos\left[\frac{(j-1)\pi r}{R_\odot}\right] \quad \text{for } j = 1, \ldots, 150. \tag{5.1}$$
This is effectively equivalent to a half-range Fourier expansion.

3. A hybrid method that involves using different forms of discretization in different regions. The range of the inversion was broken up into three separate regions: $X_0 = 5.905 \times 10^{-2}$ to $X_1 = 0.6$, 0.6 to $X_2 = 0.8$, and 0.8 to 1.0007. On the first region a low (third) order polynomial expansion of the solution was used. In other words, the solution was expanded in terms of the three basis functions
$$\phi_1^1(r) = 1,\ \phi_2^1(r) = \frac{r - X_0}{X_1 - X_0},\ \text{ and } \phi_3^1(r) = \left(\frac{r - X_0}{X_1 - X_0}\right)^2.$$
On the second region (which includes the base of the convection zone) a cosine expansion with 28 terms was used. This is essentially the same as in (5.1), except that the radius co-ordinate is transformed so that the Fourier expansion is over the interval from X_1 to X_2. In the final region the solution was discretized by choosing 118 non-uniformly spaced discretization points between X_2 and $X_{121} = 1.0007$ and using PCD; that is, on each of the discretization bins $I_3, \ldots, I_{121}$ the solution is assumed to be a constant, and the basis function for the discretization on each bin is just $\phi_1^i(r) = 1$. Within this last region the discretization points were chosen from

the following considerations. The work of Barrett (1994) demonstrates that, in a certain sense, it is possible to make an optimal choice for the discretization points in an inversion with PCD (the condition for optimality was essentially that the error magnification in the *unregularized* inversion is minimized). It is also shown that, for the solar rotation problem, at least, the distribution of support points found matches very well with the distribution that would be obtained by demanding that the area under the sum of all the kernel funtions used in the inversion be divided into equal parts by the discretization points. This distribution of points was worked out assuming 150 bins. The support points in the last region were then chosen by using any of those points that lay between 0.8 and 1.0007. Note that the solution was *not* constrained to be continuous across the boundaries between the three discretization regions (i.e. at $r/R_\odot = 0.6$ and 0.8). That is, the formalism of §2.3 was not applied here.

5.2.3 Regularization

To demonstrate the ease with which the recursive scheme for evaluating the smoothing matrix can calculate smoothing matrices of high order (in contrast to other inversion procedures where the smoothing matrices have to be put in 'by hand'), and to illustrate the smoothing effect of higher orders of smoothing, the uniform PCD inversion is performed for first to sixth order smoothing. With all of these inversions the smoothing parameter was chosen using the GCV method. Zeroth order smoothing was not used because the solution to the inversion has a large positive mean value (much larger than the typical variation) so that zeroth order smoothing is inappropriate and ineffective.

The inversions with the three different discretizations were all performed using first order smoothing, for simplicity. Again the smoothing parameter was chosen by GCV.

5.2.4 Results

Figure 5.1 shows the results of the inversions of the Libbrecht data using uniform PCD and six different order of smoothing. The errorbars and resolving lengths are not plotted because these would confuse the issue, and because they are essentially the same as for the inversion in figure 5.4(a). Figure 5.1 clearly illustrates two things. Firstly, nearer the surface of the sun, where the splitting data contain a lot of information about the solar rotation rate, the recovery is only affected slightly by the different smoothing constraints,

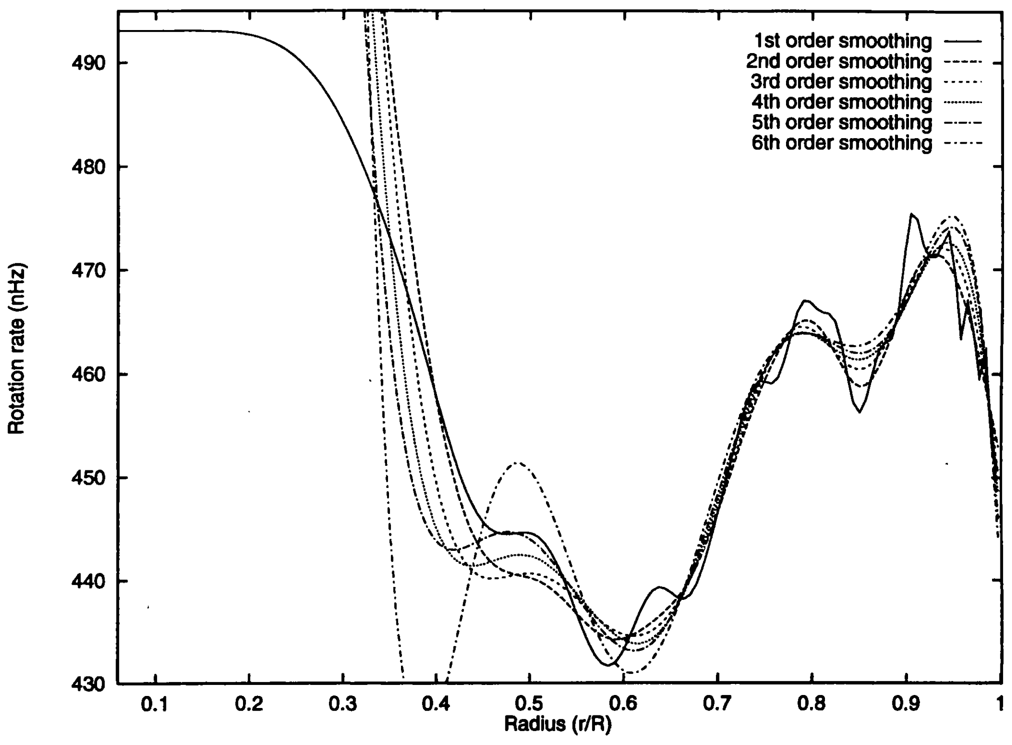


Figure 5.1: Six inversions of the data of Libbrecht (1989) for the equatorial value of the solar internal rotation for six different orders of smoothing, $p = 1, \ldots, 6$. The smoothing parameters were chosen by the GCV method.

whereas nearer the centre of the sun where the information content of the data is much lower, the solution varies wildly with the different discretizations. This property is characteristic of the effect of different smoothing constraints in any inversion. When the data is poor, or when some part of the solution is poorly constrained by the data, the solution tends to be very different when different smooothing constraints are used. This is because almost all of the 'information' in the solution in such regions comes from the smoothing constraint and how it ties the solution in that region to the solution elsewhere, and different smoothing constraints will obviously do this in different ways. Figure 5.2 emphasizes this. It shows exactly the same solutions as figure 5.1, except that now the vertical range of the graph has not been restricted to highlight the solutions near the surface. Note the scale on the vertical axis. Providing it is clear that the solution in these regions is not well constrained by the data this is not really a problem; the values of the solution there have to be written off as essentially unknown. The important thing is that the solution in

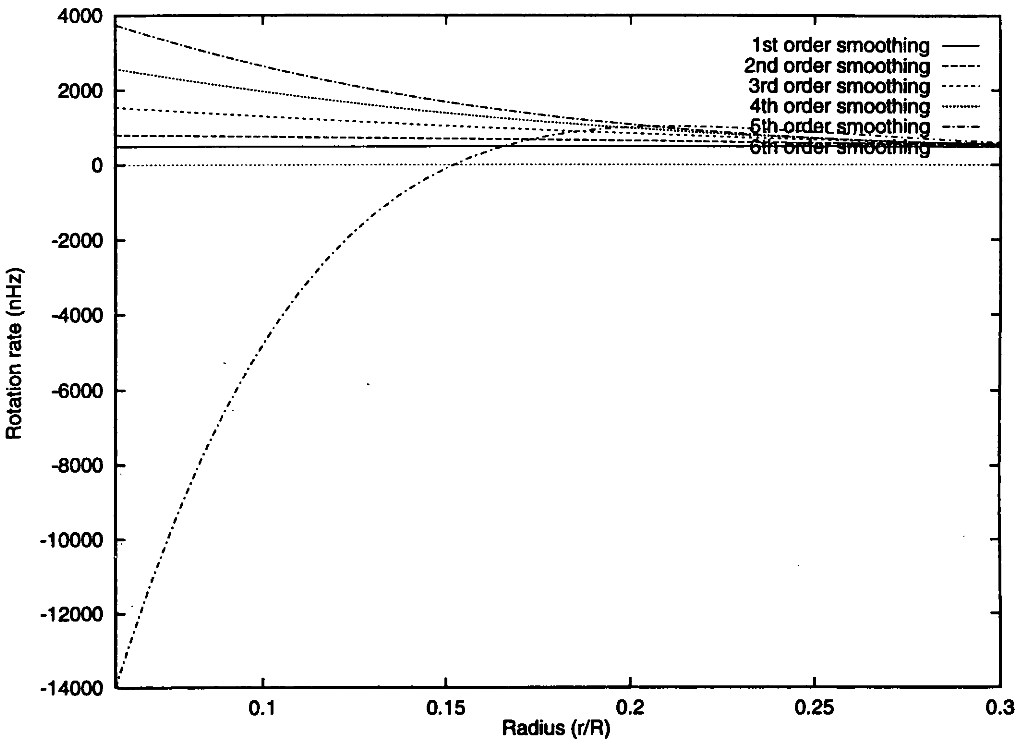


Figure 5.2: As for figure 5.1, but with the horizontal axis contracted and the vertical axis expanded to show how the solution near the centre of the sun changes as the smoothing constraint is changed.

regions where the data is good is hardly affected by the smoothing constraint at all. This means that it is not necessary to agonize over the choice of smoothing constraint, or over the effects it may have had on the solution in those regions.

The second point to note is that the small variations in the solution between different smoothing constraints nearer the surface of the sun are also characteristic of the effects of general pth-order smoothing. The higher the order of smoothing, the more the solution tends to have its corners rounded off. The reasons for this are also obvious. Any sharp 'corner' in the solution coresponds to a large change in the gradient, and therefore a discontinuity in the second derivative, and therefore an infinity in the third derivative, and so on. Each increase in the order of smoothing turns the singularity in to a worse singularity. Clearly, therefore, in order to satisfy a high order smoothing constraint the solution must have less sharp corners. Alternatively we could look at this as a result of higher orders of smoothing causing more points to be directly correlated with each other. The pth-order difference matrix will, in general, require knowledge of the value of the

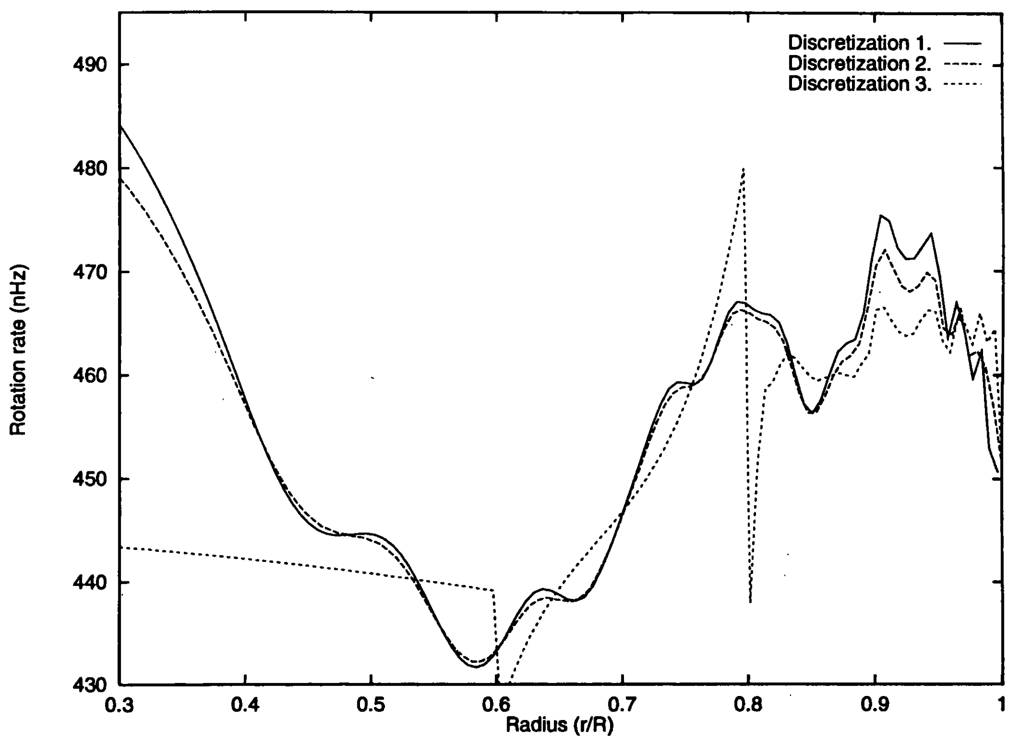


Figure 5.3: Inversions of the Libbrecht data using three different discretizations described in the text. The inversions were performed with first order smoothing and the smoothing parameters were chosen by the GCV method. Observe the sharp jumps in the solution with the third discretization method, which are due to the poor constraining of variation in the solution across the boundaries of large discretization bins (see the discussion at the top of page 113, and the text below – recall that no continuity constraints are applied here). Note that the radius range has been shortened so that the region where the solution is poorest is not shown.

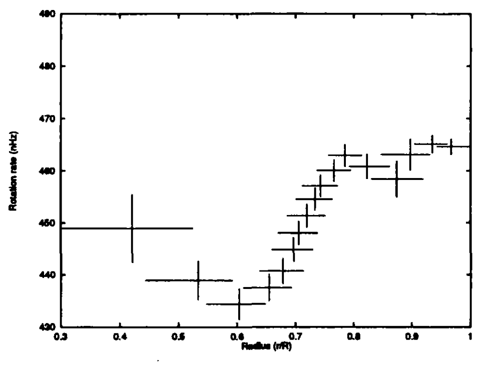
solution at p points neighbouring any given point to define the derivative at that point, and so high order smoothing will tend to correlate points that are further apart than low order smoothing, making the solution look smoother locally.

Figure 5.3 shows the results of the inversions with the three different discretizations and using first order smoothing. Again the errorbars and resolving lengths are not plotted for clarity. The solutions match up quite well, particularly for the uniform PCD and cosine expansion methods, 1 and 2. Again the mismatch is greater nearer the centre of the sun. The large jumps in the solution with the third method are due to the poor constraining of variation in the solution across the boundaries of large discretization bins by the finite-difference approximation to the derivative occurring in the first-order smoothing functional, as discussed on page 113. (No explicit constraints on the regularity of the solution were imposed in these cases, i.e. the formalism of §2.3 was not applied here.) It is difficult to say whether these jumps are a reflection of remaining instability in that part of the solution, or whether they really reflect behaviour that is normally smoothed over with other discretizations. The former may seem more likely, but look at the errorbars and resolution lengths in that region in fig. 5.4. The errorbars are rather small and the resolving lengths quite large, so that the solution is (over-)smoothed over quite a wide interval, and the (first order, i.e. first derivative) smoothing constraint must play a significant role in determining the form of the solution in that region: the smoothing must be trying very hard to reduce the gradient there. In spite of this, the solution still has a very steep gradient, which does rather suggest that there is more to the jumps than mere instability. Observe also, that the jump in the solution lies well outside the errorbars. Figure 5.4 shows the three solutions separately with errorbars and resolution lengths plotted. The errorbar, $\varepsilon(r)$, on the solution at any point r is defined to be the square root of the 'variance' of the errors on f, as in

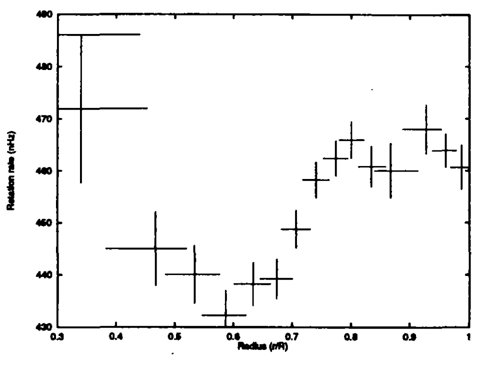
$$\varepsilon(r)^2 = \mathbb{E}\{(f(r) - \bar{f}(r))^2\} = \phi^T(r)C_{cov}\phi(r)$$

where ϕ represents the basis functions of the discretization written in vector notation – a notation that was introduced in §4.2.4 – C_{cov} is the covariance matrix of errors on the solution vector, and everything has been normalized using the given data errors, so that the effective data errors have unit variance (otherwise, there would be a factor of σ^2 on the right hand side). (Compare this with the definition (4.11) of the unrenormalized correlation profile.) The resolution lengths were calculated using the correlation profile method of chapter 4. Note that with discretization 3 the value of solution on each of the three discretization regions is hardly correlated at all with the value on the other regions. The significance of these inversions, and of the other more complete inversions presented in §5.3, is discussed in §5.3.2.

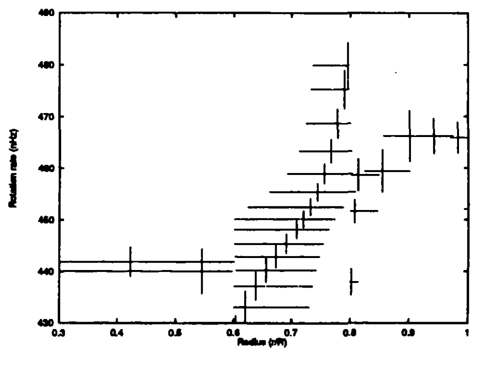
From these results we can conclude that the algorithm of chapter 2 performs quite well and consistently with a wide range of different discretizations, and can be used with almost any order of smoothing without difficulty. Note, too, that the success of the GCV method for choosing the smoothing parameter is impressive. Not only did it always return an acceptable value in these inversions, but the values it returned gave rise to very similar solutions, despite the vast difference in the discretizations and smoothing constraints with which it was dealing (although some of the differences between the various solutions could



(a)



(b)



(c)

Figure 5.4: Solutions to inversions using the three different discretizations, plotted separately for clarity, with error bars and resolving lengths. The resolving lengths were determined from the correlation profile method of chapter 4. (a) discretization 1, (b) discretization 2, (c) discretization 3. Note the absence of a correlation between the values of the solution on the different solution regions for discretization 3.

perhaps be accounted for by slightly different levels of smoothing: it may be that the solutions could be made even more similar by slightly changing the values of the smoothing parameters used). These facts together make the algorithm of chapter 2, combined with automatic methods for choosing the smoothing parameter, very useful for solving linear inverse problems like those that arise in helioseismology.

5.3 Full Inversions of the Splitting data

In this section the rotational splitting data of Libbrecht (1989) is inverted to obtain the solar angular velocity throughout the solar interior. The formalism of Ritzwoller and

Lavely (1991) is used, so that three separate inversions are performed, one for each of the three components, $w^{(1)}$, $w^{(3)}$ and $w^{(5)}$, in the expansion (1.27) of the rotational velocity field in terms of vector spherical harmonics. In fact, it turns out to be advantageous to invert for the closely related quantities

$$\tilde{w}^{(i)}(r) = -\frac{w^{(i)}(r)}{r},\ i = 1, 3, 5. \tag{5.2}$$

Since the rotational velocity, $\mathbf{v}$, and the angular velocity, Ω, at any point, (r, θ), are obviously related by

$$\mathbf{v}(r,\theta) = \Omega(r,\theta)\, r \sin\theta\, \hat{\phi}, \tag{5.3}$$

equations (1.27) and (5.2) together show that inverting for the components $\tilde{w}^{(i)}$ allows an expression for the *angular* velocity to be found:

$$\Omega(r,\theta) = \sum_{s=0}^{\infty} \tilde{w}^{(2s+1)}(r) \left\{ \frac{1}{\sin\theta} \frac{\partial Y_{2s+1,0}}{\partial\theta} \right\}. \tag{5.4}$$

There are several advantages to this. Firstly, the $\sin\theta$ factors in the expressions for the $\frac{\partial Y_{2s+1,0}}{\partial\theta}$ given in equations (13), (14) and (15) of Ritzwoller and Lavely (1991) will obviously cancel out in (5.4), making the latitudinal dependence of the angular velocity on the components $\tilde{w}^{(i)}$ slightly easier to interpret. Secondly, the rotational velocity goes to zero like r as $r \to 0$ (this is clear from the appearance of r in (5.3)), which would mean that the solutions for the $w^{(i)}$ would do the same. This would give them a significant variation, and hence a large gradient, across the whole range of the inversion, thus invalidating the use of first-order smoothing. Inverting for the $\tilde{w}^{(i)}$ will not suffer from these problems (see figures 5.5 and 5.6). Another advantage to inverting for the $\tilde{w}^{(i)}$ is that the kernels for the inversion lose a factor of $-1/r$, becoming

$$\tilde{k}_i^{(s)}(r) = \frac{r^2 \rho_0(r)}{J} \left[\xi_{rnl}^2 + L^2 \xi_{hnl}^2 - \left(2\xi_{rnl}\xi_{hnl} + \frac{1}{2} s(s+1) \xi_{hnl}^2 \right) \right] \tag{5.5}$$

(*cf.* equation (1.30), which gives the kernels for the $w^{(i)}$ inversions). This makes the kernel functions for the $\tilde{w}^{(1)}$ inversion the same as for the equatorial inversions of $a_1 + a_3 + a_5$ considered in §5.2.1, thus allowing the set of kernel functions for that case to be used here also. Similarly, the removal of the $-1/r$ factor eliminates the need for the kernels in the $s = 3, 5$ inversions to be obtained from the 'old' kernels used in the $\cos^2\theta$ expansion (see equation (2) of Ritzwoller and Lavely 1991) of the angular velocity by multiplying by such a factor.

With the new definitions (5.2) and (5.5), equation (1.29) becomes

$$q_i^{(s)} = \int_0^{R_\odot} \tilde{k}_i^{(s)}(r)\, \tilde{w}^{(s)}(r)\, dr, \text{ for } s = 1, 3, 5, \ldots, n_{max}, \text{ and } i = 1, \ldots, m, \tag{5.6}$$

(so that the data for each inversion is just the same as before, in (1.29)). Equation (5.6) expresses the relationship between the components, $\tilde{w}^{(i)}$, of the angular velocity and the Ritzwoller and Lavely splitting coefficients, and (5.5) gives the form of the kernels in the inversion for each component. It only remains to describe the relationship between the data $q_{nl}^{(s)}$ in the Ritzwoller and Lavely inversion (5.6) and the a_{nl_i} coefficients that Libbrecht (1989) used to expand his splitting data. Ritzwoller and Lavely (1991) provide this relationship for s =1, 3 and 5 (equations (61) to (63) of their paper). Using these expressions for the data, and (5.5) for the kernels, the equations were performed using piecewise constant discretization with *non-uniform* discretization points. Again the support points were fixed by demanding that they divide the area under the sum of all the kernels used into equal pieces, except that extras points were added at small radii, because the nature of the solar p-mode kernels (very large near the surface of the sun and small near the centre) results in the discretization points clustering near the surface, leaving inadequate coverage nearer the centre. The region between $r/R_\odot = 5.905 \times 10^{-2}$ and $r/R_\odot = 0.4663$ was filled with 20 evenly spaced discretization points, giving a total of 170 support points for the inversion. All the inversions were performed with the same discretization, partly because this makes interpretation and comparison of the results for the different components much easier, but also because the difference in the kernels for the different inversions is quite small (as can be seen from (5.5), the kernels only differ by a term proportional to ξh_{nl}^2, which is quite small for all but the lowest degree p modes), so the difference in the distribution of discretization points using the 'equal area' method (which is, anyway, only an approximate empirical relationship) is small. The distribution of points was calculated using the kernels for the $\tilde{w}^{(1)}$ inversion.

5.3.1 Results

Figure 5.5 shows the solutions to the inversions for the components $\tilde{w}^{(1)}(r)$, $\tilde{w}^{(3)}(r)$ and $\tilde{w}^{(5)}(r)$ of the solar internal rotation, without the resolving lengths plotted, to see the variation in the recovered solution better. Figure 5.6 shows the solutions to the three inversions *with* the resolving lengths plotted. These were determined by using the correlation profile method of chapter 4. As usual, in all of these inversions the smoothing

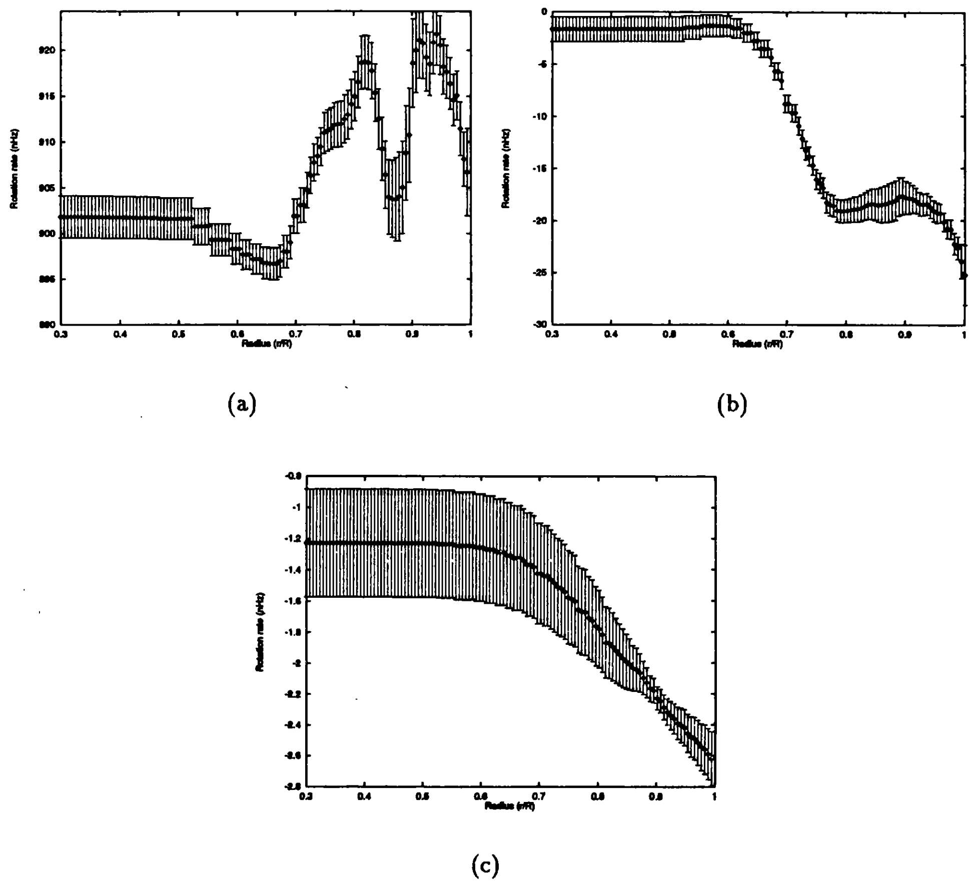


Figure 5.5: Solutions to inversions for the three different components of the solar angular velocity field ($\Omega/2\pi$), plotted with error bars, but without the resolving lengths as this makes the detail in the recovered solution easier to see. a) $\tilde{w}^{(1)}(r)$, b) $\tilde{w}^{(3)}(r)$ and c) $\tilde{w}^{(5)}(r)$.

parameter was chosen using the GCV method.

Finally, figure 5.7 shows the dependence of the solar internal angular velocity on radius at three different latitudes: $\theta = \pi/2$ (equatorial), $\theta = \pi/4$ (mid-latitude) and $\theta = 0$ (polar). These are obtained from the recovered values of the $\tilde{w}^{(i)}(r)$ through equation (5.4), with the help of equations (13) to (15) of Ritzwoller and Lavely (1991).

5.3.2 Conclusions

All three of the inversions for the angular velocity components show structure that is above the noise. The inversion for $\tilde{w}^{(5)}(r)$ is of rather poor quality, but this is not unexpected since the higher splitting coefficients are always noisier. The behaviour of $\tilde{w}^{(1)}(r)$ and $\tilde{w}^{(3)}(r)$ near the base of the convection zone is quite striking. Both show clear evidence

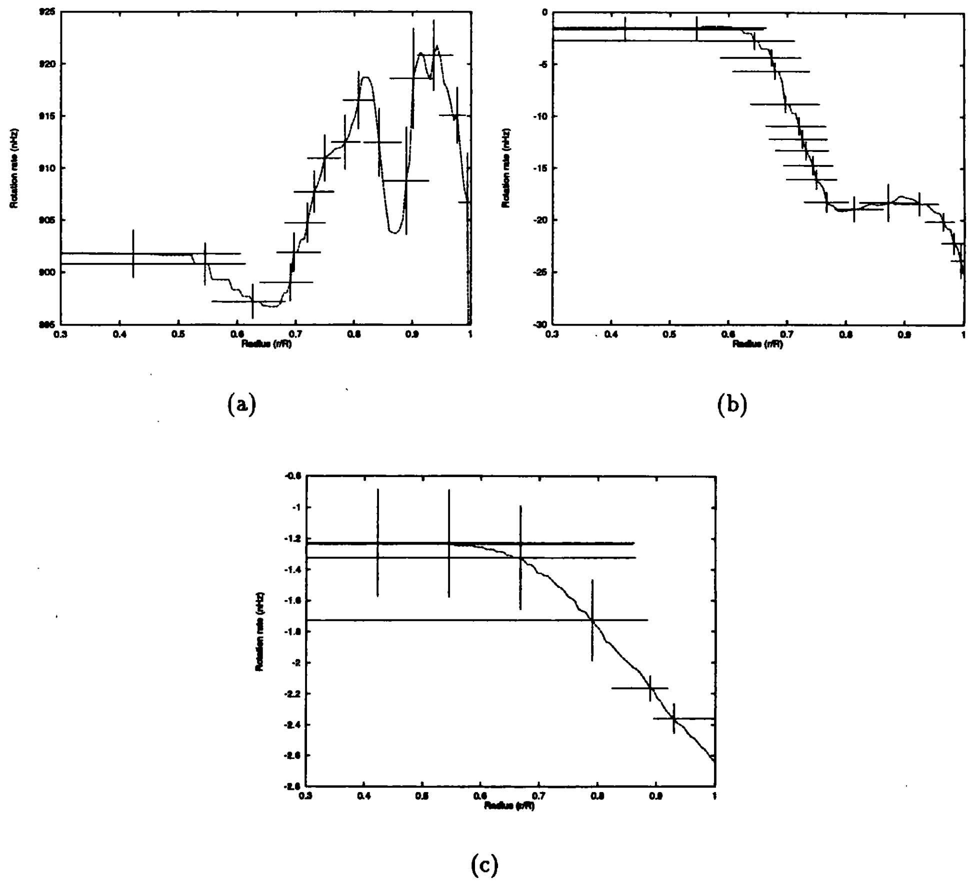


Figure 5.6: Solutions to inversions for the three different components solar angular velocity field plotted with error bars and resolving lengths. the resolving lengths were determined from the correlation profile method of chapter 4. The panels are as in figure 5.5.

for a quite sharp step at the base of the convection zone. Unfortunately, as can be seen in figure 5.6, the resolution in both inversions is similar to the characterisic width of the step, so it may actually be that the step is rather sharper than it appears. The resolution in the data is not really adequate to make firmer statements than this. The large dip in the value of the $\tilde{w}^{(1)}(r)$ component in the middle of the convection zone shows the effects of convective redistribution of angular momentum (Durney 1991).

It is possible to compare the results of the inversions obtained here with earlier results such as those of Duvall Jr. *et al.* (1984), Dziembowski (1988), Korzennik *et al.* (1988), Christensen-Dalsgaard and Schou (1988), Dziembowski *et al.* (1989), Brown *et al.* (1989), Thompson (1990), Goode *et al.* (1991) and Schou *et al.* (1992). In general, the agreement

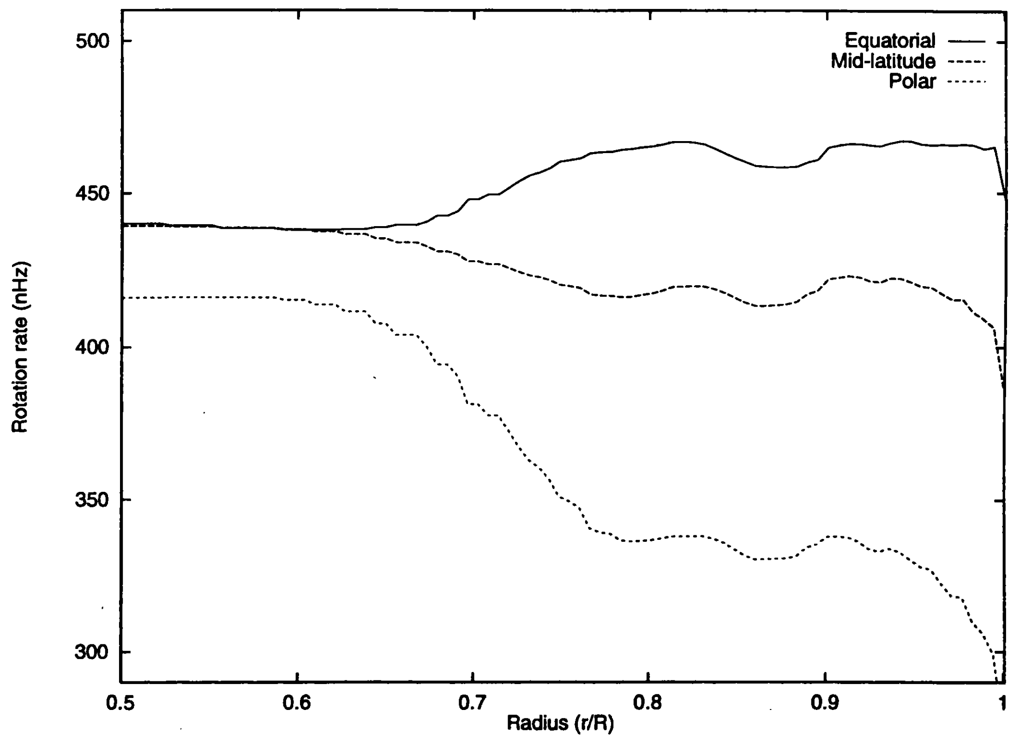


Figure 5.7: Radial dependence of the solar internal angular velocity for three different latitudes: $\theta = \pi/2$ (equatorial), $\theta = \pi/4$ (mid-latitude) and $\theta = 0$ (polar).

is good, but there are minor differences between the various inversions. The equatorial inversion of Duvall Jr. *et al.* (1984), using a different (and poorer) data set, reflects the main features of figures 5.3 and 5.4, as do the equatorial inversions of Dziembowski (1988), Dziembowski *et al.* (1989) and Goode *et al.* (1991) (figures 1a, 2a and 3a), namely, evidence for an increase in the rotation rate with depth just below the surface, of the sun, strong (if not conclusive) evidence for a dip in the rotation rate in the middle of the convection zone, a rather sharp decrease in the rotation rate with depth near the base of the convection zone ($r/R_\odot \approx 0.73$), and a subsequent increase moving down through the radiative interior. (The inversions in Goode *et al.* (1991) are more heavily smoothed than the inversions presented here, and some of the features are less easily seen – the rise in rotation rate below the surface does not appear in their inversion, for example). Some of these characteristics are of questionable significance. There seems little doubt that there is a dip in the rotation rate in the convection zone, and there is certainly a sharp change at the base of the convection zone (as mentioned above, this change is probably considerably sharper than can be seen with the limited resolution in the Libbrecht data). However, the change in the rotation rate just below the surface of the sun is questionable, because it is difficult to produce well-localized averaging kernels in this region (Thompson 1990), so the recovered rotation rate is sensitive to the rotation at greater depth. The rise in the rotation rate moving in towards the core of the sun is more pronounced in the inversions shown in figures 5.3 and 5.4 than in the earlier works. This is, in large measure, due to the different levels of smoothing used. Dziembowski (1988), Dziembowski *et al.* (1989) and Goode *et al.* (1991) also invert the data of Libbrecht (1989), so any differences are certainly a result of different methods used, rather than differences in the data. The appearance of a pronounced rise in the solution despite the use of first-order smoothing suggests that the data requires it (although there is again the problem that the averaging kernels are not well localized at smaller radii, and so the recovered values of the rotation rate could be contaminated). However, the information about the rotation rate at great depth comes from low order modes, which are subject to systematic errors in the measurements of their frequencies and splittings, and so this rise may not reflect properties of the real sun. Nevertheless, the inversions in figures 5.3 and 5.4 show that, ignoring its significance for the real sun, the recovered rotation rate at small radii is larger than previously found.

In the work of Christensen-Dalsgaard and Schou (1988) (fig. 1), Brown *et al.* (1989) (fig. 13), Thompson (1990) (fig. 1) and Goode *et al.* (1991) (fig. 8), for example, the rotation rate is plotted as a function of radius for different latitudes, just as in figure 5.7. Again, there is broad agreement between these results with some differences. The dip in rotation rate in the middle of the convection zone is generally evident, as is a rise in the rotation rate moving in from the surface, and a change at the base of the convection zone. The different inversion techniques and levels of regularization can, in general, account for the differences between the inversions. It is interesting to note the drop in the rotation rates (especially the polar rate) at smaller radii, shown clearly by Christensen-Dalsgaard and Schou (1988) (who also invert the Libbrecht data). This conflicts with the other inversions, and is somewhat contentious owing to the difficulties of obtaining meaningful averaging kernels for the polar rotation rate (particularly at small radii).

Generally speaking, it is very difficult with the limited quality of the data presently available to use the improved techniques presented in this thesis to learn anything more about the sun than has already been discovered with other, more basic, techniques – the resolution provided by the data is really insufficient to warrant the application of more complicated discretizations to pick out subtle features in the rotation profiles. However, when the data improves, as it will with the successful operation of the GONG network (Harvey *et al.* 1993), the wider choice of discretizations and smoothing constraints made possible by the algorithm of chapter 2 should permit more general discretizations (like the hybrid discretization illustrated in fig. 5.4c) to be used to recover more detail in the solution than is possible with simpler discretizations. Numerical trials using artificial data and noise levels appropriate to those expected for the GONG data could be used to investigate the effectiveness of the algorithm of chapter 2. The effectiveness of the new techniques will depend on the specific form of the real solution, but by testing the algorithm on solutions containing various different features and comparing the results with those from simpler inversion procedures the usefulness of the algorithm ought to be clearly indicated.

5.4 Summary and Conclusions

In this chapter the effectiveness of the methods and formalisms introduced, developed and studied in this thesis have been examined. The inversion algorithm of chapter 2, including the implementation of the Fix-Heiberger algorithm, has proved to be very robust and flexible. The GCV method for choosing the smoothing parameter, which was studied in chapter 3, along with the EDF method, has also shown itself to be a very reliable and effective way to set the regularization levels in RLS inversions. The correlation profile method for assessing the resolution achieved in inversions produces acceptable results, and is very much faster to calculate than the definition of resolution that results from measuring the width of the RLS averaging kernels. All these things together make the performing and interpreting of inversions in helioseismology and elsewhere simpler and more accessible to non-expert inverters. The generality of the inversion algorithm used here increases the degree of control that the inverter has over the inversion, by permitting more general discretizations and a greater choice of smoothing constraint (higher orders of smoothing).

Chapter 6

Future Work

6.1 Introduction

To conclude the work in this thesis, this chapter will suggest several ways to improve upon the ideas and techniques presented, and will outline other possible avenues of research. Sections 6.2 to 6.4 consider the work in chapters 2 to 4, and discuss improvements and extensions that could be made to the techniques described in those chapters. Section 6.5 presents some other unresolved problems less directly related to the research contained in this thesis and makes some suggestions as to how they should be attacked.

6.2 Improving the Algorithm

The inversion algorithm expounded in chapter 2 is very flexible, and, with the implementation of the Fix-Heiberger algorithm described there it is also robust and efficient. There is one commonly used discretization scheme that was not dealt with in chapter 2, though, and that is *spline* discretization. This is essentially a form of discretization in which the solution is expanded in terms of low order polynomials (cubic splines use cubic polynomials, for example – see *Numerical Recipes*, §3.3) on a number of small bins (rather like PCD, – see §1.9.4). If this were all there was to it, this method would fit very naturally into the formalism of chapter 2. However, the important, indeed defining, property of splines is that they are continuous, and have continuous derivatives of each order up to one less than the degree of the spline (the degree of a spline is the degree of the polynomial used to expand the solution on each bin), so that cubic splines have continuous second derivatives and linear splines are merely continuous. It is possible to employ the methodology of §2.3, and impose continuity on the derivatives through external constraints, but this would be rather inefficient and unappealing. In fact, there is a way of thinking about splines that makes it slightly easier to see how to incorporate them into the algorithm of chapter 2. Cox (1975) describes an algorithm for evaluating and interpolating with splines. The basic idea of the formulation described is that the splines are defined not in terms of separate polynomials on each bin, but rather in terms of a collection, $N_1(r), N_2(r), \ldots,$ of basis functions, called B-splines, defined over the whole range of interpolation: the spline used to represent the solution function is then just a linear combination of these basis functions:

$$f(r) = \sum_j f_j N_j(r),$$

which of course compares very well with the expansions of the solution function in terms of other basis functions given in §1.9. In this form interpolation with B-splines fits in perfectly with the formalism of chapter 2 if the whole range of interpolation is taken to be a single bin, rather than using the discretization points (*knots* is the technical term) on which the spline is defined.

However, the B-splines themselves have rather special properties that would render the naive implementation of spline interpolation with the algorithm of chapter 2 rather inefficient. Consider nth degree splines, say. The B-splines are defined essentially by demanding that:

- on each bin they are nth degree polynomials,
- they, and their first $(n-1)$st derivatives are continuous across the boundaries of the bins (so that there are continuous everywhere, since they are nth degree polynomials within the bins), and
- they are each non-zero only within $n + 1$ of the bins on which the spline is defined (although the bins on which different B-splines are non-zero overlap).

These conditions, along with specification of the knots, fix the B-spline basis (up to a constant). The last condition shows that the B-splines can be envisaged as a kind of discretization where the basis functions are not restricted to be non-zero within a single bin, but can be zero on several neighbouring bins. This way of thinking of spline discretization is not catered for by the algorithm of chapter 2. In a certain sense, spline discretization is intermediate between PCD, where the regions over which different basis functions are non-zero do not overlap, and the function expansion methods (see §1.9), where every basis function is non-zero on the same bin. The most important effect of the 'partially overlapping' basis functions is to complicate the calculation of the integrals and derivatives required in the definition of the smoothing matrix (see equations (2.71), (2.72) and (2.73)), but the calculation of the kernel matrix would also be affected.

It would certainly be useful and satisfying to extend the algorithm of chapter 2 to include spline discretization. This would make the resulting RLS inversion procedure almost completely general, and allow almost any kind of discretization to be used simply by setting the appropriate input parameters.

6.3 Better Choices of λ

The methods for choosing the smoothing parameter reviewed in §1.9.6 and examined in more detail in chapter 3 all rely basically on the chi-square measure of how well the residuals fit the expected distribution (see *Numerical Recipes*, §14.3). This essentially only considers the overall size of the vector of residuals. There are other way to compare the distributions of two sets of random numbers that use more information about the individual values of those numbers. For example, the *Kolmogorov-Smirnoff statistic* (*Numerical Recipes*, §14.3) compares the cumulative distributions of the two sets of random numbers, and can therefore identify residuals that have some kind of pattern to them that is clearly at odds with the expected distribution. (The cumulative distribution of the residuals in an inversion is obtained by ranking them, to get $\epsilon_1 \leq \epsilon_2 \leq \ldots \leq \epsilon_m$, and then notionally plotting and joining the set of points $(-\infty, 0)$, $(\epsilon_1, 0)$, $(\epsilon_1, 1/m)$, $(\epsilon_2, 1/m), \ldots, (\epsilon_i, (i-1)/m)$, $(\epsilon_i, i/m), \ldots, (\infty, 1)$ to get a kind of step function. Comparing this with the cumulative distribution of the expected gaussian errors is the basis of the Kolmogorov-Smirnoff test. See *Numerical Recipes*) To see more clearly how the Kolmogorov-Smirnoff test can sometimes improve upon the chi-square statistic, consider, for some value of the smoothing parameter, a set of residuals that are all the same size, ϵ, say (assume, for the sake of argument that the variances of the data errors are uniform). Then the value of χ^2 in the EDF method (equation (1.62)) will be $m\epsilon^2/\sigma^2$. ϵ could be such that (1.62) holds exactly, and so we would choose that value of the smoothing parameter in the inversion. However, we would certainly not say that the residuals in any way mimic gaussian errors, which was the principle on which the EDF method was based. The Kolmogorov-Smirnoff test would detect this difference between the two distributions immediately, because their cumulative distributions would be radically different. It should be pointed out, though, that there can be cases when the K-S test fails to distinguish between two distributions, although the chi-square statistic would easily separate them.

The K-S test is known to be robust and work well with quite small samples, and so if a method for choosing the smoothing parameter can be found that is based on the K-S test it should be very useful in many areas of inverse theory. One possible idea is simply to minimize the K-S statistic (*Numerical Recipes*, equation (14.3.5)) of the difference between the residuals and the expected gaussian distribution of data errors (just like the Phillips method, see equation (1.61), but with the K-S instead of the chi-square statistic) over λ. There are two problems with this, though. First, experience shows that the function of λ that is to be minimized is rather rough, and so numerical mimimization becomes something of a challenge. This roughness is largely a result of the definition of the K-S statistic as the maximum distance between the two distributions. It could perhaps be improved if a different statistic, such as the sum-squared distance was used. Secondly, such a method for choosing λ would, like the Phillips method, suffer from the problem of the solution 'fitting' the data errors. This could perhaps be overcome with the use of the introduction of the equivalent degrees of freedom factor, just as with EDF, but this requires more detailed investigation.

To recap: there is much reason for expecting that alternative methods for choosing the smoothing parameter in RLS inversions can be found, based on criteria for determining when the residuals 'look like' the expected data errors other than chi-square, such as the Kolmogorov-Smirnoff test. Such methods may improve upon the methods examined in chapter 3.

6.4 More on Resolution

Determining the resolution achieved in the solution of an inverse problem is an absolutely vital part of the inversion procedure, and the work in chapter 4 of this thesis extends the range of options available for making this determination. There are several question relating to this work that warrant further investigation. The close relationship between the correlation profiles and the averaging kernels was clearly demonstrated in chapter 4, but the differences between the two defintions of resolution are significant and require further investigation. For example, the fact that the two profiles seem to be most similar when the smoothing parameter is nearly optimal (i.e. chosen by GCV – see §1.9.6) is interesting. Is this merely a coincidence or is there an underlying reason? If there is a reason, could this perhaps be used as a way to make a good choice for the smoothing parameter?

It is well known that performing inversions requires a trade-off between error magnification in the solution and resolution (smoothing or bias). Is there some 'conserved quantity' that can be obtained from the error estimates and the resolving length at any point, which is (approximately, at least) independent of the value of the smoothing parameter, and could be used as an absolute measure of the amount of information in the data about the value of the solution at that particular point? For example, since the error, $e(r)$, decreases as the smoothing parameter, λ, increases, and the resolution length, $l(r)$, increases with λ, perhaps a quantity of the form $e(r)l(r)$ is largely independent of λ. If such a quantity could be found it would make interpretation of the results of inversions much easier.

6.5 Further Development

Up to the present time, apart from studying the solar internal rotation, helioseismologists have considered primarily the problem of determining the pressure and density stratification throughout the solar interior – the solar stucture problem. This requires knowledge only of the *frequencies* of the oscillation modes of the real sun, and only of the ***adiabatic*** oscillation frequencies of the solar model. At the moment, oscillation data is really inadequate for doing much more than this. However, in the not too distant future projects such as the GONG project (Harvey *et al.* 1993) will be producing excellent measurements, not only of mode frequencies, but also of amplitudes and damping rates. Obviously, the equations of adiabatic oscillation provide no information about these quantities, because the adiabatic condition ensures that the mode neither loses nor gains energy, and therefore has a constant amplitude (zero damping). This amplitude is arbitrary, amounting to nothing more than a normalization of the eigenfunction (because, again, no energy can be put into the mode to excite it to a particular amplitude). The power spectrum of the oscillation for a particular mode contains a great deal more information than just the mode frequency, and this information constrains more than just the sound speed profile or the pressure and density stratification. For example, the mode lifetime, which is reflected in the width of the (almost Lorentzian) peak in the power spectrum for any mode (see Anderson *et al.* 1990), is determined by the transfer of energy from the oscillation to the background state (principally the exchange of heat by radiative transfer in the outer layers of the sun, which tends to damp the oscillations, but turbulent convection also has an effect). Calculation of the modes of non-adiabatic oscillation of a solar model automatically includes a model of this energy transfer (Unno *et al.* 1979, chapter IV), which results in the mode frequency becoming a complex quantity. The imaginary part of the frequency is very closely related to the damping rate for the oscillations. This opens up the possibility of using this extra information to learn more about the solar structure, in particular, to constrain those quantities that are known to cause the damping. It is generally accepted that the uncertainties in the model are the result of our incomplete knowledge of the microphysical parameters that appear in the fluid equations (such as in the calculation of opacity and energy generation rates), and our inability to solve the full equations to a satisfactory level of accuracy due to their complex, non-linear nature: the presence of a turbulent convection zone in the outer layers of the solar interior is a source of considerable error, as the usual mixing-length approximation used to calculate the average stratification of the convection zone is based on little more than dimensional arguments, and this is inadequate given the present, and expected, level of accuracy obtained, or to be obtained, from helioseismic observations of the solar oscillations. Certainly, non-adiabatic effects are very much related to the presence and properties of the solar convection zone, and so the excitation and damping of the oscillation modes ought to contain information about this process. Also, the treatment of the radiative transfer of energy near the solar surface (where the diffusion approximation that is usually used – see Unno *et al.* 1979, chapter IV – breaks down) is not accurately modelled in solar structure calculations. Studying the damping of oscillation modes may help to reduce this inaccuracy.

It is now fairly well established (Goldreich and Kumar 1990; Osaki 1993) that the solar oscillations are stochastically excited by the turbulent convection, and are intrinsically damped. The amplitudes to which the modes are excited are determined by the interplay between the excitation and damping mechanisms. An analysis of the amplitudes of different oscillation modes would provide information on the rate at which energy is put into the oscillation by the turbulent convection, which would place useful constraints on the nature of the distribution of the convective velocities (at least in the region where the excitation occurs). This is an important and interesting problem, for the following reasons. Convection in astrophysical situations is very diffucult to model, is rather poorly understood, and is impossible to reproduce in the laboratory situation (Gough 1977), so any observational constraints could provide a stiff test of theories of convection. This would, in turn, assist research in stellar structure in general, as one of the major unknowns would have been reduced. Of course, ultimately, one of the main aims of helioseismology is to provide a realistic test of theories of stellar structure and evolution. Furthermore, convection is responsible for many of the observed properties of the sun: granulation, supergranulation, differential rotation and meridional circulation (Durney 1991), the solar cycle, generation of the solar magnetic field, and so on. It would be difficult to claim that we had a good model of the sun (or any star with an outer convection zone) without modelling the convection accurately. How could the observed mode lifetimes and amplitudes be utilized to constrain the convection (and heat transport in general) in the sun?

It is important to realize that, unlike in the case of adiabatic oscillations, the equations of non-adiabatic, non-radial oscillation are not self-adjoint, and therefore do not give rise to a variational principle (Unno *et al.* 1979) – it perhaps should be stated here that even the adiabatic oscillations satisfy a variational principle only for simplified boundary conditions. This means that some other method for determining the sensitivity of the mode frequencies (including the imaginary part corresponding to excitation and damping effects) must be found. The approach of Rosenwald and Rabaey (1991) seems potentially very useful here. Extending their application of the *continuous orthonormalization and adjoint methods* to the equations of non-adiabatic oscillation would immediately permit the sensitivities of the damping rates to any aspect of the solar structure to be calculated. This would permit measurements of the widths of the mode peaks in oscillation power spectra to be used in inversions.

Making use of the amplitude data would be a little more complicated. Although Goldreich and Kumar (1990) and others have modelled the stochastic excitation of p modes, their models have been rather too simple. The interaction between the convective velocities and the acoustic (and even gravity) modes must be examined and modelled as completely as possible. One possibility is to recognize that there are unstable linear eigenmodes of the sun that correspond exactly to the modes that grow when the convectively unstable sun is perturbed away from exact (unstable) hydrodynamic equilibrium: they are the 'convective' eigenmodes. In the real sun these modes have sufficient amplitude for the non-linear terms in the fluid equations to become important, and the result is that the modes interact to produce the turbulent convection that is observed. It has been shown (Narashima and Antia 1982) that the convective eigenmodes can be used to describe convection (or, at least, to reproduce the mixing length model) providing that non-adiabatic (energy transfer) effects are taken into account in the calculation of the linear convective eigenmodes. This suggests that such an expansion of the convective flows in terms of convective eigenmodes may be useful. If the velocity field (and the other fields) in the fluid equations is written in terms of this expansion, and the non-linear terms that couple the convective velocity to the acoustic modes are considered, it may be possible to calculate the interaction between the convection and the p modes. The strength of this interaction will depend on the specifics of the convective flow field and on the eigenfunction of the p mode in question: the larger the amplitudes of convective eigenmodes with eigenfunctions quite similar to the p mode considered, and the closer the timescales of the convective and acoustic modes, the more the p mode is likely to be excited. Measuring the amplitudes of many p modes therefore places a constraint on the convective flow field. Perhaps, as an aside, it could be noted that the work of Durney (1991) on the interaction of convection with rotation and meridional circulation is in a similar (though not identical) spirit to the work suggested here.

To summarize: it is suggested that it would be worthwhile extending the work in this thesis, and of other researchers, to the study of inverse problems in helioseismology that involve the non-abiabatic effects on the solar oscillations and structure, and, in particular, to the use of mode amplitudes and lifetimes to place strong constraints on theories of solar (and general astrophysical) convection, and on descriptions of radiative transfer in the optically thin outer layers of the sun. The real solar p modes are certainly not completely described by the adiabatic approach, so it makes sense to attempt to introduce these non-adiabatic effects.

There is another reason that non-adiabatic effects are potentially of some importance in helioseismic inverse problems. In the solar rotation problem the kernels for performing the inversion are calculated using oscillation displacement eigenfunctions (see §1.5.3) obtained from the solutions of the equations of *adiabatic* oscillation. The eigenmodes for non-adiabatic oscillations are slightly different from the adiabatic eigenmodes, particularly in regions (such as near the surface of the sun) where non-adiabatic effects are important.

This means that the sensitivities of the splitting coefficients to the solar rotation will, in general, be different from the sensitivities that would be found if the non-adiabatic eigenmodes were used. Furthermore, the forward problem that describes the effect on the splittings of any given rotation profile relies on the existence of a variational principle relating perturbations in the solar structure to perturbations in the mode frequencies. For non-adiabatic oscillations there is no such variational principle. The question of how to improve the formulation of the inverse problem for solar rotation in the face of non-adiabatic effects then breaks down into two parts:

- Is it necessary to reformulate the problem to derive a completely new relationship between the solar rotation profile and the splitting frequencies, including all of the new physics required to account for the non-adiabatic effects, or is the old statement of the forward problem (*cf.* equations (3.2) and (3.3)) adequate if the adiabatic eigenfunctions originally used are replaced by their non-adiabatic counterparts?
- If it is not necessary to reformulate the problem completely, is it even necessary to use the non-adiabatic eigenfunctions to calculate the kernels, or are the adiabatic ones perfectly adequate?

These questions have not been answered, as yet, but they are questions that require investigation before the results of rotation inversions with very high-quality data can be completely trusted. Of course, with the data presently available there is little justification for making what are probably relatively minor improvements in the accuracy of the formulation of the problem, but with the vastly improved quality of the GONG data (Harvey *et al.* 1993) that will soon be available small corrections such as these may be significant, and are certainly worthy of study. It seems possible that the work of Rosenwald and Rabaey (1991) may again be useful in the development of a reformulation of the problem, should this prove to be necessary, because their methods do not require the existence of a variational principle for the mode frequencies.

Some of the points just made with regard to the effects of non-adiabaticity on the solar rotation problem can equally be applied to the effect of an inaccurate solar model on the calculation of the kernel functions in the rotation inversion. So far, no solar model has been found which is completely consistent with the available oscillation data. This again means that the eigenfunctions obtained from a solar model will differ from the eigenfunctions calculated using the 'correct' solar model, and so the kernel functions themselves will, in general, be different from the 'correct' kernel functions that would be obtained from a completely accurate solar model. It is necessary to investigate whether this difference is important for the determination of the solar internal rotation.

Appendix A

Calculating $T(\lambda)$ and $R(\lambda)$

In section 2.4 an algorithm for diagonalizing the matrix $H^TH + \lambda C$ that appears in the solution (2.82) to the RLS inversion of (2.5) was described. The algorithm is needed to speed up the function evaluations in the routines to choose the smoothing parameter using automatic methods such as GCV and EDF (see §1.9.6). Here we will look at the expressions for the trace and residual sum of squares functions, $T(\lambda)$ and $R(\lambda)$, that appear in the definitions (1.65) and (1.62) of those functions. Asymptotic expressions for the values of these functions as $\lambda \to 0$ and as $\lambda \to \infty$ will be derived, and will be used to give the asymptotic values of the GCV and EDF functions. These values are often useful in the root-finding and minimization procedures to choose λ.

To begin with, recall some of the definitions and results of §2.4: The matrix $H^TH+\lambda C$ is reduced by a sequence of (in general) six congruence transformations, $P_1, \ldots, P_6$, to

$$H^TH + \lambda C = [P_1P_2P_3P_4P_5P_6]\, M(\lambda)\, [P_1P_2P_3P_4P_5P_6]^T, \tag{A.1}$$

where the reduced matrix, $M(\lambda)$ is given by

$$M(\lambda) = \left(\begin{array}{c|c|c|c|c} F+\lambda I & 0 & 0 & I & 0 \\ \hline 0 & \Delta+\lambda I & 0 & 0 & 0 \\ \hline 0 & 0 & I & 0 & 0 \\ \hline I & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right) \begin{array}{l} \}r_3 \\ \}r_1 - r_3 \\ \}r_2 \\ \}r_3 \\ \}n-R \end{array} \tag{A.2}$$

The inverse of $H^TH + \lambda C$, calculated using singular value decomposition, is

$$(H^TH + \lambda C)^{-1} = PM^{-1}(\lambda)P^T, \tag{A.3}$$

where P is the inverse transpose of the sequence of congruences,

$$P = [P_1P_2P_3P_4P_5P_6]^{-T}, \tag{A.4}$$

and $M^{-1}(\lambda)$ is given by

$$M^{-1}(\lambda) = \left(\begin{array}{c|c|c|c|c} 0 & 0 & 0 & I & 0 \\ \hline 0 & (\Delta+\lambda I)^{-1} & 0 & 0 & 0 \\ \hline 0 & 0 & I & 0 & 0 \\ \hline I & 0 & 0 & -(F+\lambda I) & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right).$$

Δ and F are diagonal matrices, and the congruence matrices are defined in equations (2.120), (2.122), (2.124), (2.129), (2.132) and (2.134) of §2.4.

There is one result that we need before we can begin the derivation of the expressions for $T(\lambda)$ and $R(\lambda)$. It can easily be verified by direct calculation that

$$P_i^{-1}\tilde{I}_{r_1}P_i^{-T} = \tilde{I}_{r_1} \text{ for } i = 2,\ldots,6, \tag{A.5}$$

where $\tilde{I}_{r_1}$ is defined in equation (2.119), and the inverses (actually the inverse transposes) of the congruence matrices are given at the end of each step in the explanation of the Fix-Heiberger reduction in §2.4.

A.1 Evaluation of $T(\lambda)$

Beginning with the definition (equation (1.63))

$$T(\lambda) = \mathrm{Tr}\left\{I_m - H(H^TH+\lambda C)^{-1}H^T\right\}$$

(where I_m is the $m \times m$ identity matrix – m being the number of data points being used in the inversion, and $\mathrm{Tr}\{A\}$ denotes the trace of the matrix A), we can make the following simplification (remembering that within a trace the matrices can be cyclically permuted without changing the value of the trace)

$$\begin{aligned} T(\lambda) &= m - \mathrm{Tr}\left\{(H^TH+\lambda C)^{-1}H^TH\right\} \\ &= m - \mathrm{Tr}\left\{(H^TH+\lambda C)^{-1}(H^TH+\lambda C-\lambda C)\right\} \\ &= m - \mathrm{Tr}\left\{(H^TH+\lambda C)^{-1}(H^TH+\lambda C)\right\} + \lambda\mathrm{Tr}\left\{(H^TH+\lambda C)^{-1}C\right\} \end{aligned} \tag{A.6}$$

Normally, the first trace in the last line of this equation would just be the trace of I_n, the $n \times n$ identity matrix, which is, of course, n. However, the inverse appearing there is not necessarily the true inverse. If $H^TH + \lambda C$ is singular it will be the pseudo-inverse found using SVD. The value of this trace must therefore be examined more closely.

Inverting the definition of P in (A.4) to get $[P_1P_2P_3P_4P_5P_6] = P^{-T}$, allows (A.1) to be written

$$H^TH + \lambda C = P^{-T}M(\lambda)P^{-1}. \tag{A.7}$$

Using this and (A.3) gives

$$\begin{aligned}(H^TH+\lambda C)^{-1}(H^TH+\lambda C) &= PM^{-1}P^TP^{-T}MP^{-1} \\ &= PM^{-1}MP^{-1}.\end{aligned} \tag{A.8}$$

The discussion following equation (2.138) in §2.4 can be used to show that

$$M^{-1}M = \left(\begin{array}{c|c} I_R & 0 \\ \hline 0 & 0 \end{array}\right). \tag{A.9}$$

(This can also be verified by direct calculation). I_R is the $R \times R$ identity matrix, with $R = r_1 + r_2 + r_3$ being the rank of $H^TH + \lambda C$ (see §2.4). It is clear then, from the obvious result $\mathrm{Tr}\{PM^{-1}MP^{-1}\} = \mathrm{Tr}\{M^{-1}M\} = R$, that (A.8) gives the result we want:

$$\mathrm{Tr}\left\{(H^TH+\lambda C)^{-1}(H^TH+\lambda C)\right\} = R. \tag{A.10}$$

Now only the $\mathrm{Tr}\left\{(H^TH+\lambda C)^{-1}C\right\}$ term in the expression (A.6) for $T(\lambda)$ remains to be evaluated. Using (A.3) and permuting gives

$$\mathrm{Tr}\left\{(H^TH+\lambda C)^{-1}C\right\} = \mathrm{Tr}\left\{M^{-1}P^TCP\right\}. \tag{A.11}$$

As $C = (UD^{1/2})\tilde{I}_{r_1}(UD^{1/2})^T = P_1\tilde{I}_{r_1}P_1^T$ (from equation (2.120) and the sentence preceding it) it is possible to write

$$\begin{aligned}P^TCP &= [P_1P_2P_3P_4P_5P_6]^{-1}P_1\tilde{I}_{r_1}P_1^T[P_1P_2P_3P_4P_5P_6]^{-T} \\ &= [P_6^{-1}P_5^{-1}P_4^{-1}P_3^{-1}](P_2^{-1}\tilde{I}_{r_1}P_2^{-T})[P_3^{-T}P_4^{-T}P_5^{-T}P_6^{-T}]\end{aligned} \tag{A.12}$$

Now we can use result (A.5) repeatedly for each of the matrices $P_2, \ldots, P_6$, giving

$$P^TCP = \tilde{I}_{r_1}. \tag{A.13}$$

(A result which will also be used later, in the derivation of the expression for $R(\lambda)$). Inserting this into (A.11) gives

$$\mathrm{Tr}\left\{(H^TH+\lambda C)^{-1}C\right\} = \mathrm{Tr}\left\{M^{-1}\tilde{I}_{r_1}\right\}. \tag{A.14}$$

The evaluation of the matrix product $M^{-1}\tilde{I}_{r_1}$ can be performed directly, and the result is

$$M^{-1}\tilde{I}_{r_1} = \left(\begin{array}{c|c|c} 0 & 0 & 0 \\ \hline 0 & (\Delta+\lambda I)^{-1} & 0 \\ \hline 0 & 0 & 0 \end{array}\right)\begin{array}{l} \}r_3 \\ \}r_1 - r_3 \\ \}n - r_1 \end{array}$$

where the block structure has been simplified by, where possible, combining zero blocks that occur in the 5 × 5 block matrix. Quite clearly,

$$\mathrm{Tr}\left\{(H^TH+\lambda C)^{-1}C\right\} = \mathrm{Tr}\left\{(\Delta+\lambda I)^{-1}\right\} = \sum_{i=1}^{r_1-r_3}\frac{1}{\delta_i+\lambda}. \tag{A.15}$$

The δ_i are, of course, the diagonal elements of Δ. In the light of (A.15) and the earlier result (A.10), the expression (A.6) for $T(\lambda)$ becomes

$$T(\lambda) = m - R + \sum_{i=1}^{r_1-r_3}\frac{\lambda}{\delta_i+\lambda}. \tag{A.16}$$

This is enough to calculate $T(\lambda)$ quickly in the GCV or EDF subroutines, but it is often useful to be able to obtain the asymptotic values of the GCV and EDF functions as $\lambda \to 0$ and $\lambda \to \infty$. To derive these quantities it is necessary to consider values of the δ_i. If δ_i is zero for some i, the ith term in the sum in (A.16) will be $\lambda/\lambda = 1$, for all λ. On the other hand, if $\delta_i \neq 0$ the ith term will not be constant, but will tend to zero as $\lambda \to 0$. This is an important distinction if we want to know the asymptotic behaviour of $T(\lambda)$. The δ_i have been calculated numerically, so they will suffer from the usual numerical errors. δ_i that should actually be zero will not be, thanks to these errors. We therefore need a criterion for determining whether the δ_i really are zero. This criterion should be based only on the likely amount of error on the δ_i, rather than any of the concerns about ill-conditioning that affected the choice of the criterion (2.117): we are not concerned with ill-conditioning here. Let us assume that we have an expression for the likely round-off error in the δ_i. Then it is sensible to say that any δ_i that is smaller than this is really zero. Let ρ ($\leq r_1 - r_3$) be the number of non-zero δ_i's according to this criterion, i.e. ρ is the rank of the matrix $\hat{A}_2$ in step 6 of the Fix-Heiberger algorithm. Then (remembering that the diagonal elements of Δ have been put into decreasing order along the diagonal)

$$\begin{aligned} T(\lambda) &= m - R + \sum_{i=1}^{\rho}\frac{\lambda}{\delta_i+\lambda} + \sum_{i=\rho+1}^{r_1-r_3}\frac{\lambda}{\lambda} \\ &= m - R + [(r_1-r_3)-\rho] + \sum_{i=1}^{\rho}\frac{\lambda}{\delta_i+\lambda}, \end{aligned} \tag{A.17}$$

and all the δ_i in the second line are positive. The asymptotic expressions we are seeking can easily be derived from this:
As $\lambda \to 0$

$$T(\lambda) \to m - R + [(r_1 - r_3) - \rho] \tag{A.18}$$

and as $\lambda \to \infty$

$$T(\lambda) \to m - R + (r_1 - r_3). \tag{A.19}$$

A.2 Evaluation of $R(\lambda)$

The derivation of the expression for $R(\lambda)$ from the Fix-Heiberger reduction of $H^T H + \lambda C$ is a little more involved than for $T(\lambda)$. The original defintion of $R(\lambda)$ given in (1.67) can be expanded to

$$\begin{aligned} R(\lambda) &= g^2 - 2\mathbf{g}^T H(H^T H + \lambda C)^{-1} H^T \mathbf{g} \\ &\quad + \mathbf{g}^T H(H^T H + \lambda C)^{-1} H^T H (H^T H + \lambda C)^{-1} H^T \mathbf{g}. \end{aligned} \tag{A.20}$$

Concentrate initially on the last term in (A.20). This can be written

$$\begin{aligned} &\mathbf{g}^T H(H^T H + \lambda C)^{-1}(H^T H + \lambda C)(H^T H + \lambda C)^{-1} H^T \mathbf{g} \\ &-\lambda \mathbf{g}^T H(H^T H + \lambda C)^{-1} C (H^T H + \lambda C)^{-1} H^T \mathbf{g}. \end{aligned} \tag{A.21}$$

Making use of equations (A.8) and (A.9) reduces $(H^T H + \lambda C)^{-1}(H^T H + \lambda C)$ to

$$P \left(\begin{array}{c|c} I_R & 0 \\ \hline 0 & 0 \end{array} \right) P^{-1},$$

giving (with the help of (A.3) again)

$$\begin{aligned} (H^T H + \lambda C)^{-1}(H^T H + \lambda C)(H^T H + \lambda C)^{-1} &= P \left(\begin{array}{c|c} I_R & 0 \\ \hline 0 & 0 \end{array} \right) P^{-1} P M^{-1} P^T \\ &= P M^{-1} P^T \\ &= (H^T H + \lambda C)^{-1}. \end{aligned} \tag{A.22}$$

The second step in the preceding sequence can easily be verified directly.

The result (A.13) can be used to show that

$$\begin{aligned} (H^T H + \lambda C)^{-1} C (H^T H + \lambda C)^{-1} &= P M^{-1} P^T C P M^{-1} P^T \\ &= P M^{-1} \tilde{I}_{r_1} M^{-1} P^T \end{aligned} \tag{A.23}$$

Using (A.22) and (A.23) together in (A.21) results in the following expression for the last term in the definition of $R(\lambda)$ (equation (A.20)):

$$\mathbf{g}^T H(H^T H + \lambda C)^{-1} H^T \mathbf{g} - \lambda \mathbf{g}^T H P M^{-1} \tilde{I}_{r_1} M^{-1} P^T H^T \mathbf{g}.$$

Replacing the last term in (A.20) with this expression, collecting like terms together and using (A.3) to replace the occurrence of $(H^T H + \lambda C)^{-1}$ leaves

$$R(\lambda) = g^2 - \mathbf{g}^T H P M^{-1} P^T H^T \mathbf{g} - \lambda \mathbf{g}^T H P M^{-1} \tilde{I}_{r_1} M^{-1} P^T H^T \mathbf{g}. \qquad \text{(A.24)}$$

The obvious definition

$$\mathbf{w} \overset{\text{def}}{\equiv} P^T H^T \mathbf{g} \qquad \text{(A.25)}$$

simplifies the appearance of (A.24) considerably, giving

$$R(\lambda) = g^2 - \mathbf{w}^T (M^{-1} + \lambda M^{-1} \tilde{I}_{r_1} M^{-1}) \mathbf{w}. \qquad \text{(A.26)}$$

The matrix $M^{-1}\tilde{I}_{r_1}M^{-1}$ can be calculated straight from the definitions (2.142) and (2.119) of §2.4, once it is noted that in the 5×5 block matrix format

$$\tilde{I}_{r_1} = \left(\begin{array}{c|c|c|c|c} I & 0 & 0 & 0 & 0 \\ \hline 0 & I & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right). \qquad \text{(A.27)}$$

The result is

$$M^{-1}\tilde{I}_{r_1}M^{-1} = \left(\begin{array}{c|c|c|c|c} 0 & 0 & 0 & 0 & 0 \\ \hline 0 & (\Delta + \lambda I)^{-2} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & I & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right), \qquad \text{(A.28)}$$

which leads to

$$M^{-1} + \lambda M^{-1}\tilde{I}_{r_1}M^{-1} = \left(\begin{array}{c|c|c|c|c} 0 & 0 & 0 & I & 0 \\ \hline 0 & (\Delta+\lambda I)^{-1} + \lambda(\Delta+\lambda I)^{-2} & 0 & 0 & 0 \\ \hline 0 & 0 & I & 0 & 0 \\ \hline I & 0 & 0 & -F & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \end{array}\right), \qquad \text{(A.29)}$$

where, in the block containing $-F$, a $-\lambda I$ term in M^{-1} cancelled with the λI coming from the I block in (A.28).

Having derived the block matrix expression (A.29) for $M^{-1} + \lambda M^{-1}\tilde{I}_{r_1}M^{-1}$ it is necessary to partition the vector $\mathbf{w}$ in an corresponding manner to effect the evaluation of $\mathbf{w}^T(M^{-1} + \lambda M^{-1}\tilde{I}_{r_1}M^{-1})\mathbf{w}$. The appropriate partitioning is

$$\mathbf{w} = \left(\begin{array}{c} \mathbf{w}_1 \\ \hline \mathbf{w}_2 \\ \hline \mathbf{w}_3 \\ \hline \mathbf{w}_4 \\ \hline \mathbf{w}_5 \end{array}\right)\begin{array}{l} \}r_3 \\ \}r_1 - r_3 \\ \}r_2 \\ \}r_3 \\ \}n - R \end{array} \tag{A.30}$$

(Compare this with the partitioning of $M(\lambda)$, which was initially introduced in (2.131).) Using this in the previous expression for $R(\lambda)$ (equation (A.26)) and expanding the term involving $\mathbf{w}$ gives

$$\begin{aligned} R(\lambda) &= g^2 - \{\mathbf{w}_1^T\mathbf{w}_4 + \mathbf{w}_2^T[(\Delta + \lambda I)^{-1} + \lambda(\Delta + \lambda I)^{-2}]\mathbf{w}_2 \\ &+ \mathbf{w}_3^T\mathbf{w}_3 + \mathbf{w}_4^T\mathbf{w}_1 - \mathbf{w}_4^T F\mathbf{w}_4\}. \end{aligned} \tag{A.31}$$

Collecting all the λ-independent terms together with the definition

$$\gamma = g^2 - [\mathbf{w}_3^T\mathbf{w}_3 + 2\mathbf{w}_1^T\mathbf{w}_4] + \mathbf{w}_4^T F\mathbf{w}_4, \tag{A.32}$$

and expanding the summation implicit in the terms containing Δ (remembering that Δ is a diagonal $r_1 - r_3 \times r_1 - r_3$ matrix) results in

$$R(\lambda) = \gamma - \sum_{i=1}^{r_1-r_3} \frac{\delta_i + 2\lambda}{(\delta_i + \lambda)^2} w_{2i}^2. \tag{A.33}$$

Again, to derive asymptotic expressions for $R(\lambda)$ it is necessary to distinguish between zero and non-zero δ_i, as was discussed in the paragraph preceding equation (A.17). With ρ non-zero δ_i's, equation (A.33) becomes

$$R(\lambda) = \gamma - \sum_{i=1}^{\rho} \frac{\delta_i + 2\lambda}{(\delta_i + \lambda)^2} w_{2i}^2 - \frac{2}{\lambda}\sum_{i=\rho+1}^{r_1-r_3} w_{2i}^2. \tag{A.34}$$

It is clear from the last term that if any of the components of $\mathbf{w}_2$ corresponding to zero eigenvalues in Δ are non-zero, $R(\lambda)$ will diverge as $\lambda \to 0$. It will now be shown that this cannot happen (in exact arithmetic, at least), and that if $\delta_i = 0$, then $w_{2i} = 0$.

First, note that, from the definition of $M(\lambda)$ in (A.2), and the 5×5 block form for $\tilde{I}_{r_1}$ given in (A.27)

$$M(\lambda) = M(0) + \lambda\tilde{I}_{r_1},$$

and so putting $\lambda = 0$ in (A.7) shows that

$$H^T H = P^{-T} M(0) P^{-1},$$

which immediately gives

$$(HP)^T (HP) = M(0) \tag{A.35}$$

Now, denote the $m \times n$ matrix HP (which also appears in the definition (A.25) of $\mathbf{w}$) by $\mathcal{H}$, and denote the ith column of $\mathcal{H}$ by $\mathbf{h}^{(i)}$, for $i = 1, \ldots, n$, so that, for each i, $\mathbf{h}^{(i)}$ is an m-vector, and

$$h_j^{(i)} \equiv \mathcal{H}_{ji}, \text{ for } i = 1, \ldots, n, \text{ and } j = 1, \ldots, m.$$

Then, using $||.||$ to denote the usual euclidean vector norm (see equation (5.17) of Craig and Brown 1986),

$$||\mathbf{h}^{(i)}||^2 \equiv \sum_{j=1}^{m} (h_j^{(i)})^2 = \sum_{j=1}^{m} \mathcal{H}_{ij}^T \mathcal{H}_{ji} = (\mathcal{H}^T \mathcal{H})_{ii} = M_{ii}(0), \tag{A.36}$$

where the last equality follows from equation (A.35). A quick inspection of (A.2) shows that if $r_3 + 1 \leq i \leq r_1$, then $M_{ii}(0) = \Delta_{i-r_3, i-r_3} = \delta_{i-r_3}$. For convenience, introduce the notation $\hat{\imath}$ for $i - r_3$. It was established in the paragraph preceding (A.17) that

$$\begin{aligned} \delta_{\hat{\imath}} &> 0 \text{ for } 1 \leq \hat{\imath} \leq \rho \\ \delta_{\hat{\imath}} &= 0 \text{ for } \rho + 1 \leq \hat{\imath} \leq r_1 - r_3 \end{aligned}$$

If $\delta_{\hat{\imath}} = 0$, then, by virtue of equation (A.36), $||\mathbf{h}^{(i)}|| = 0$. But it is a defining property of norms that $||\mathbf{h}^{(i)}|| = 0 \Leftrightarrow \mathbf{h}^{(i)} = \mathbf{0}$, which in turn means that $h_j^{(i)} \equiv \mathcal{H}_{ji} = 0$ for $j = 1, \ldots, m$. (In other words, any column of $\mathcal{H}$ corresponding to a zero eigenvalue $\delta_{\hat{\imath}}$ will have all its elements zero.) From the definition (A.25) of $\mathbf{w}$, the partitioning (A.30) and the definition of $\mathcal{H}$,

$$w_{2\hat{\imath}} \equiv w_i = \sum_{j=1}^{m} \mathcal{H}_{ji} g_j \equiv \mathbf{h}^{(i)T} \mathbf{g},$$

and it was established above that $\delta_{\hat{\imath}} = 0 \Rightarrow \mathbf{h}^{(i)} = 0$. It is clear, therefore, that if $\delta_{\hat{\imath}} = 0$, then $w_{2\hat{\imath}} = 0$: every term of the last summation in (A.34) is zero, and so that final summation disappears. This leaves the final form for $R(\lambda)$:

$$R(\lambda) = \gamma - \sum_{\hat{\imath}=1}^{\rho} \frac{\delta_{\hat{\imath}} + 2\lambda}{(\delta_{\hat{\imath}} + \lambda)^2} w_{2\hat{\imath}}^2. \tag{A.37}$$

From this it is a simple matter to obtain the asymptotic expressions for $R(\lambda)$:

As $\lambda \to 0$

$$R(\lambda) \to \gamma - \sum_{i=1}^{\rho} \frac{w_{2i}^2}{\delta_i}, \tag{A.38}$$

and as $\lambda \to \infty$

$$R(\lambda) \to \gamma. \tag{A.39}$$

To conclude this appendix the asymptotic expressions for the trace and residual sum of squares functions will be used to give the limiting values of the EDF and GCV functions using the definitions of these functions in equations (1.65) and (1.62) of §1.9.6.

As $\lambda \to 0$

$$EDF(\lambda) \to \frac{\gamma - \sum_{i=1}^{\rho} \frac{w_{2i}^2}{\delta_i}}{m - R + [(r_1 - r_3) - \rho]} - \hat{\sigma}^2 \tag{A.40}$$

and

$$GCV(\lambda) \to \frac{\gamma - \sum_{i=1}^{\rho} \frac{w_{2i}^2}{\delta_i}}{\{m - R + [(r_1 - r_3) - \rho]\}^2}. \tag{A.41}$$

As $\lambda \to \infty$

$$EDF(\lambda) \to \frac{\gamma}{m - R + (r_1 - r_3)} - \hat{\sigma}^2 \tag{A.42}$$

and

$$GCV(\lambda) \to \frac{\gamma}{\{m - R + (r_1 - r_3)\}^2}. \tag{A.43}$$

Bibliography

Anderson, E. R., Duvall Jr., T. L., and Jefferies, S. M.: 1990, *Astrophys. J.* **364**, 699

Backus, G. and Gilbert, J.: 1967, *Geophys. J. R. astr. Soc.* **13**, 247

Backus, G. and Gilbert, J.: 1968, *Geophys. J. R. astr. Soc.* **16**, 169

Backus, G. and Gilbert, J.: 1970, *Phil. Trans.* **266A**, 123

Barrett, R.: 1994, *Optimizing the Discretization in RLS Inversions.*, In Preparation

Blyth, T. S. and Robertson, E. F.: 1986, *Matrices and Vector Spaces*, Vol. 2 of *Essential Student Algebra*, Chapman and Hall

Brown, T. M., Christensen-Dalsgaard, J., Dziembowski, W. A., and Goode, P. R.: 1989, *Astrophys. J.* **343**, 526

Burkill, J. C.: 1962, *A First Course in Mathematical Analysis*, Cambridge University Press

Campbell, L., McDow, J. C., Moffat, J. W., and Vincent, D.: 1983, *Nature* **305**, 508

Christensen-Dalsgaard, J.: 1992, in J. T. Schmelz and J. C. Brown (eds.), *The Sun: A Laboratory for Astrophysics*, pp 11–28, NATO ASI Series, Kluwer Academic Publishers

Christensen-Dalsgaard, J., Gough, D. O., and Thompson, M. J.: 1991, *Astrophys. J.* **378**, 413

Christensen-Dalsgaard, J., Hansen, P. C., and Thompson, M. J.: 1993, *Mon. Not. R. astr. Soc.* **264**, 541

Christensen-Dalsgaard, J. and Schou, J.: 1988, in E. J. Rolfe (ed.), *Seismology of the Sun and Sun-Like Stars*, pp 149–53, ESA SP-286, Noordwijk, Holland

Christensen-Dalsgaard, J., Schou, J., and Thompson, M. J.: 1990, *Mon. Not. R. astr. Soc.* **242**, 353

Cox, M. G.: 1975, *J. Inst. Math. Appl.* **15**, 95

Craig, I. J. and Brown, J. C.: 1986, *Inverse Problems in Astronomy*, Adam Hilger Ltd

Durney, B. R.: 1990, *Astrophys. J.* **351**, 682

Durney, B. R.: 1991, *Astrophys. J.* **378**, 378

Duvall Jr., T., Harvey, J. W., and Pomerantz, M. A.: 1986, *Nature* **321**, 500

Duvall Jr., T. L., Dziembowski, W. A., Goode, P. R., Gough, D. O., Harvey, J. W., and Leibacher, J. W.: 1984, *Nature* **310**, 22

Dziembowski, W.: 1988, in E. J. Rolfe (ed.), *Seismology of the Sun and Sun-Like Stars*, pp 259–63, ESA SP-286, Noordwijk

Dziembowski, W. A., Goode, P. R., and Libbrecht, K. G.: 1989, *Astrophys. J. (Letters)* **337**, L53

Dziembowski, W. A., Pamyatnykh, A. A., and Sienkiewicz, R.: 1990, *Mon. Not. R. astr. Soc.* **244**, 542

Fitzpatrick, B. G.: 1991, *Inverse Problems* **7**, 675

Fix, G. and Heiberger, R.: 1972, *Soc. Ind. Appl. Math. J. Num. Anal.* **9**, 788

Goldreich, P. and Kumar, P.: 1990, *Astrophys. J.* pp 694–704

Golub, G. H., Heath, M., and Wahba, G.: 1979, *Technometrics* **21**, 215

Goode, P. R., Dziembowski, W. A., Korzennik, S. G., and Rhodes Jr., E. J.: 1991, *Astrophys. J.* **367**, 649

Gough, D. O.: 1977, in E. A. Spiegel and J.-P. Zahn (eds.), *Problems of Stellar Convection*, pp 349–63, Springer-Verlag

Gough, D. O.: 1984, *Mem. S. A. It.* **55(1-2)**, 13

Gough, D. O.: 1985, *Solar Phys.* **100**, 65

Gough, D. O.: 1986, in Y. Osaki (ed.), *Hydrodynamic and Magnetohydrodynamic Problems in the Sun and Stars*, pp 117–43, University of Tokyo Press

Gough, D. O.: 1989, *'Hound and Hare' Exercise*, GONG Newsletter No. 9

Gough, D. O. and Kosovichev, A. G.: 1988, in E. J. Rolfe (ed.), *Seismology of the Sun and Sun-Like Stars*, pp 195–201, ESA SP-286, Noordwijk, Holland

Hall, P. and Titterington, D. M.: 1987, *J. R. Statist. Soc.* B **49(2)**, 184

Harvey, J., Hill, F., Kennedy, J., and Leibacher, J.: 1993, in T. M. Brown (ed.), *GONG 1992: Seismic Investigation of the Sun and Stars*, pp 397–410, Astronomical Society of the Pacific Conference Series

Jeffrey, W. and Rosner, R.: 1988, in J. Christensen-Dalsgaard and S. Frandsen (eds.), *Advances in Helio- and Asteroseismology*, pp 129–32, IAU Symposium No. 123., D. Reidel

Kippenhahn, R. and Weigert, A.: 1990, *Stellar Structure and Evolution*, A&A Library Series, Springer-Verlag

Korzennik, S. G., Cacciani, A., Jr., E. J. R., Tomczyk, S., and Ulrich, R. K.: 1988, in E. J. Rolfe (ed.), *Seismology of the Sun and Sun-Like Stars*, pp 117–24, ESA SP-286, Noordwijk, Holland

Kosovichev, A. G., Christensen-Dalsgaard, J., Däppen, W., Dziembowski, W. A., and Gough, D. O.: 1992, *Mon. Not. R. astr. Soc.* **259**, 536

Leibacher, J. W. and Stein, R. F.: 1971, *Astrophys. Lett.* **7**, 191

Leighton, R. B., Noyes, R. W., and Simon, G. W.: 1962, *Astrophys. J.* **135**, 474

Libbrecht, K. G.: 1989, *Astrophys. J.* **336**, 1092

Libbrecht, K. G., Woodard, M. F., and Kaufman, J. M.: 1990, *Astrophys. J. Suppl.* **74**, 1129

Lighthill, J.: 1958, *Introduction to Fourier Analysis and Generalized Functions*, Cambridge University Press

Moffat, J. W.: 1983, *Phys. Rev. Lett.* **50**, 709

Narashima, D. and Antia, H. M.: 1982, *Astrophys. J.* **262**, 358

Narayan, R. and Nityananda, R.: 1986, *Ann. Rev. Astron. Astrophys.* **24**, 127

Osaki, Y.: 1993, in *Inside the Stars; Proceedings og the 137th IAU Colloquium*, pp 512–20, Astronomical Society of the Pacific

Parlett, B. N.: 1980, *The Symmetric Eigenvalue Problem*, Prentice-Hall Series in Computational Mathematics, Prentice-Hall, Inc.

Phillips, D. L.: 1962, *J. Ass. Comput. Mach.* **9**, 84

Pijpers, F. P. and Thompson, M. J.: 1992, *Astron. Astrophys.* **262**, L33

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P.: 1992, *Numerical Recipes in FORTRAN*, Cambridge University Press, 2nd edition

Ritzwoller, M. H. and Lavely, E. M.: 1991, *Astrophys. J.* **369**, 557

Rosenwald, R. D. and Rabaey, G. F.: 1991, *Astrophys. J. Suppl.* **77**, 97

Schou, J., Christensen-Dalsgaard, J., and Thompson, M. J.: 1992, *Astrophys. J. (Letters)* **385**, L59

Thompson, M. J.: 1990, *Solar Phys.* **125**, 1

Titterington, D.: 1985, *Astron. Astrophys.* **144**, 381

Turchin, V. F., Kazlov, V. P., and Malkevich, M. S.: 1971, *Sov. Phys.-Usp.* **13**, 681

Ulrich, R. K.: 1970, *Astrophys. J.* **162**, 993

Unno, W., Osaki, Y., Ando, H., and Shibahashi, H.: 1979, *Nonradial Oscillations of Stars*, University of Tokyo Press

Vorontsov, S. V., Baturin, V. A., and Pam, A. A.: 1991, *Nature* **349**, 49

Vorontsov, S. V. and Shibahashi, H.: 1991, *PASJ* **43(5)**, 739

Wahba, G.: 1983, *J. R. Statist. Soc. B* **45(1)**, 133

Weir, A. J.: 1973, *Lebesgue Integration & Measure*, Cambridge University Press

GLASGOW
UNIVERSITY
LIBRARY