The Use of Descent Methods in The Solution of Seismic Inversion Problems

Herman Jaramillo
Universidad de Medellín

May 22, 2018
# Contents

1 Introduction ............................................................................. 7

2 Notes about Eigenvalues/Eigenvectors .................................. 11
   2.1 Why are they important ................................................. 11
       2.1.1 In Mathematics ....................................................... 11
       2.1.2 In Physics .............................................................. 11
       2.1.3 Computer Graphics ................................................. 12
       2.1.4 In Numerical Analysis ........................................... 14
   2.2 Eigenvectors and Eigenvalues in Numerical Analysis .......... 14
   2.3 The regularized linear least squares problem .................... 16

3 Steepest Descent Method ....................................................... 19
   3.1 The Quadratic Minimization equivalence to a linear problem $Ax = b$ 19
   3.2 The line search and SD algorithm .................................... 20
   3.3 Error Analysis .............................................................. 23
       3.3.1 Error analysis $E_k$ between consecutive iterations ........ 27
       3.3.2 Error analysis $\epsilon_k$ relative to the exact solution ........ 30

4 The Conjugate Gradient (CG) Method .................................... 33
   4.1 Origin ............................................................................. 33
   4.2 Developing of The Method .............................................. 33
   4.3 The Algorithm ............................................................. 41
       4.3.1 Orthogonality Relations between the residual vectors $r_j$ and
             the direction vectors $p_j$ ........................................... 42
       4.3.2 Finishing Details and The Algorithm ......................... 44
   4.4 Sensitivity Analysis and rates of Convergence ................... 46
   4.5 Preconditioning ........................................................... 59

5 Non–linear CG ................................................................. 63
   5.1 A general formula for $\beta_i$ ............................................ 65
   5.2 The Fletcher-Reeves formula ......................................... 66
   5.3 The Polak-Ribière formula ............................................. 67
   5.4 The Hestenes-Stiefel formula ......................................... 67
5.5 The Dai-Yuah formula ................................................. 68

6 Newton Optimization Methods 69
6.1 Several Dimensions .................................................. 75
6.2 Quasi-Newton Methods ........................................... 78
6.3 Line Search Methods ............................................... 79
6.4 Quasi-Newton methods revisited ................................ 93

7 Seismic Inversion in the Generalized Least Square (GLS) context 113
7.1 General Formulation .................................................. 113
  7.1.1 The Forward Modeling ....................................... 113
  7.1.2 The Adjoint .................................................... 114
  7.1.3 The Hessian ................................................. 116
7.2 The Point Spread Function PSF .................................. 117
  7.2.1 For Imaging .................................................. 117
  7.2.2 For Inversion ................................................. 117
7.3 The Data Covariance Matrix ...................................... 118
7.4 The Generalized Tikhonov Regularization ....................... 119
7.5 The Seismic Problem ................................................ 119
  7.5.1 Introduction .................................................. 119
  7.5.2 Born Inversion .............................................. 120
  7.5.3 Kirchhoff Inversion ......................................... 135

8 Non-linear Inversion : Full Waveform Inversion (FWI) 149
8.1 introduction .......................................................... 149
8.2 The Matrix Version ................................................ 150
  8.2.1 The Lagrange Multipliers Approach ....................... 155
8.3 The Differential Equation Version ............................... 156
  8.3.1 The ASM by Example ...................................... 156
  8.3.2 The Problem in Abstract ................................... 160
  8.3.3 The modeling equations .................................... 160
  8.3.4 The Objective Function .................................... 160
  8.3.5 The Gradient of the Objective Function ................... 161
8.4 Examples ................................................................... 163
  8.4.1 The homogeneous–isotropic wave equation .............. 163
  8.4.2 Tarantola’s 1984 inversion for $K$, $\rho$, and source $s$ 164
  8.4.3 Elastic Wave Equation ....................................... 167
  8.4.4 Transversely Isotropic Media with a Vertical axis of Symmetry: VTI 170
  8.4.5 Transversely Isotropic Media with a Horizontal Axis of Symmetry: HTI 172
8.5 Numerical Test ......................................................... 173
8.6 Conclusions ............................................................. 179
References 179

Appendices 181

A Convexity and inequalities used on this document 183
A.1 Convex Functions .............................................. 183
A.2 Jensen’s Inequality ................................................. 185
A.3 Weighted geometric/arithmetic mean inequality ............... 187
A.4 Hölder’s inequality ................................................. 187
A.5 Minkowski’s inequality ............................................ 189
A.6 Kantarovich Inequality ............................................ 190
A.7 A rational versus exponential inequality ......................... 192
A.8 Error bound for steepest descent ................................ 192

B Auxiliary Programs: 195
B.0.1 Matlab code for the steepest descent method ................. 195

C Generalized Tikhonov Regularization 199
C.1 Norm $\ell_2$, model centered at 0 ................................ 199
C.2 Model centered at $x_0$ and the Mahalanobis norms .......... 200
C.2.1 Model centered at $x_0$ ......................................... 200
C.2.2 The Mahalanobis norms ....................................... 200

D The Rayleigh–Sommerfeld Integral derived from the Angular Spectrum 203
D.1 The Angular Spectrum ............................................. 203
D.2 The Derivation: The method of stationary phase .......... 205
D.2.1 Taylor series on the phase ................................... 205
D.2.2 The stationary phase .......................................... 206
D.2.3 The Hessian ...................................................... 208
D.A The Fresnel Integral .............................................. 209
D.B The Fourier Transform of the Fresnel integral kernel ........ 210
D.C The Method of Stationary Phase for 2 fold integrals ......... 212

E The Huyghens-Fresnel Principle 215
E.1 The Principle ....................................................... 215
E.2 The Huygens Principle ............................................ 221
E.2.1 The principle in words ....................................... 222
E.2.2 The Integral Formulation ..................................... 222

F Comparison of Born, Rytov, and De Wolf Approximations 231
F.1 Introduction ......................................................... 231
F.2 General Theory ...................................................... 231
F.3 The Rytov Approximation ......................................... 233
F.3.1 The Rytov Approximation ..................................... 235
F.4 Comparison between Rytov and Born ............................................. 237
F.5 The de Wolf Approximation ......................................................... 238
F.6 In Between Born and Rytov ......................................................... 240
F.A The Laplacian of a Product of Functions ..................................... 240

G The Fréchet and Gâteaux Derivatives .......................................... 241
   G.0.1 Review of derivatives in calculus ......................................... 241
   G.0.2 What is differentiation on general operators? ......................... 244

H Adjoins of partial differential operators ........................................ 249

Index ............................................................................................... 261
Chapter 1

Introduction

*Life is an optimization problem.*

*Happiness is the objective function*

These notes are a practical guide to optimization techniques which use descent methods such as steepest descent, linear and non–linear conjugate gradient methods, and Newton and quasi-Newton methods. While we focus in descent methods, all methods shown here work equivalently for ascent direction methods by reversing the direction of the search vector. Optimization is the problem of minimizing or maximizing some objective function.

Descent methods are applied when at least there is a local minimum on a smooth scalar function \( f(x) \) where \( x \) is a point in \( \mathbb{R}^n \).

The main objective of these notes is to describe the algorithms to advance downhill toward the minimum point as fast as possible. Therefore all methods can be seen as the implementation of the formula

\[
x_{k+1} = x_k + \alpha_n p_n
\]

where \( x_k \) is the current point, \( x_{k+1} \) is the next point along the downside direction, and \( \alpha_n \) is a scalar to be found along a line search direction with direction \( p_n \). The two main parameters here are \( \alpha_k \) and \( p_k \).

Two families of algorithms are currently being used to descent toward the minimum. One family is called as trust region \(^1\) and the other is known as the line search \(^2\). The trust region family updates first the step \( \alpha_k \) based on some model (linear or quadratic for example) then with \( \alpha_k \) fixed, it tries several directions \( p_k \) to locate the minimum on a radius from the point \( x_k \). Then the \( \alpha_k \) is stretched to allow faster convergence or squeeze to avoid stepping too far in the wrong direction. The line search method finds first a direction \( p_k \) and then along that direction finds the \( \alpha_k \) which

\(^1\)http://en.wikipedia.org/wiki/Trust_region

\(^2\)http://en.wikipedia.org/wiki/Line_search
provides the minimum value of the objective function. In this way the trust region and the line search are dual methods. Some analogies, only with the purpose of illustration are, from the seismic/seismology field: one is like a moving through rays, while the other is like moving through wavefronts. From the computational geometry field: One is like computing Delaunay triangles and the other is like computing Voronoi polygons. The other family, the line search, is the topic of these notes and will be discussed in detail.

- According to the search direction \( p_n \) the method receives its name. All methods studied here, which we call “gradient based” are implemented according to the formula

\[
p_n = -B_n \nabla f(x_k),
\]

with \( B \) a matrix in \( \mathbb{R}^{n \times n} \). We name the few included in this document:

(i) **steepest descent**: \( B_n = I \), where \( I \) is the identity matrix in \( \mathbb{R}^{n \times n} \).

(ii) **Newton**: Here \( B = \nabla^2 f(x_k)^{-1} \), where \( \nabla^2 \) means Hessian or the \( \mathbb{R}^{n \times n} \) matrix of second partial derivatives.

(iii) **Quasi-Newton**: It is an approximation of the Newton method where the Hessian matrix is computed in iterations from an initial guess, together with the step. Here \( B_n = g(B_{n-1}) \). We will explain the meaning of \( g \) later.

(iv) **Conjugate Gradient**: This is steepest descent with multi-axis scaling.

- According to the step size \( \alpha_k \). The general name of the problem is **line search**. Here we want to minimize the function \( f(x + \alpha p) \) by finding a scaling \( \alpha \) along the direction vector \( p \) which is no-necessarily normalized. Assuming that \( \alpha \in (0, b] \) we can march downhill on the hyper-surface \( f \) using several methods such as:

(i) Bisection

(ii) Golden Rule

(iii) Armijo (also known as Backtracking)

(iv) Parabolic Interpolation.

We elaborate on these methods within these notes.

An analysis of convergence speed is always interesting. Finding the best direction is more expensive in terms of computing resources so there is a trade-off between speed and accuracy. Large steps take us faster to the minimum or to the wrong valley, but short steps are more accurate. Knowing the right size of the step is the key of the line searching method. The book by Nocedal and Wright offers good theoretical (and practical) background as well as additional references that complement

\(^3\)http://www.bioinfo.org.cn/wangchao/maa/Numerical_Optimization.pdf
the statements shown there. Another good reference is the book *Practical Method of Optimization* \(^4\) by Fletcher.

Together with descent methods we add a few notes about eigenvectors and eigenvalues, since they play an important role in the process of iterative inversions. While the mathematical and physical aspects of eigenvectors and eigenvalues are of great importance, we focus more on the numerical aspects of these tools in the solution of inversion problems. We assume that the reader has familiarity with basic linear algebra such as the concepts of symmetric and positive definite matrices as well as eigenvalues and eigenvectors of a matrix, among other topics.

We use Internet references often providing hyperlinks with a footnote with the explicit text for the website link. Wikipedia is the most cited website. At the moment we do not include bibliography. While paper is nice, we believe that the time is coming when any reference should be at the click of a mouse button and we try to move in that direction.

While optimization problems are useful for all science and technology fields we are interested on the use of the methods for seismic Full Waveform Inversion (FWI) in the search for geological models of pieces of the earth’s interior.

A final note about notation. It is a common practice to use small letters for vectors, capital letters for matrices and Greek letters for scalars. This is a very practical notation and saves a lot of typing in systems such as LaTeX. However in these notes we use *boldmath* symbols for vectors, CAPITAL letters for matrices and small letters as well as Greek letters for scalars. When Greek letters are in boldface, such as for example \(\alpha\) they indicate a vector. Using boldface brings one more dimension (color) into reading into the reading and this could be convenient for some readers.

Chapter 2

Notes about Eigenvalues/Eigenvectors

It is assume that the reader is familiar already with the basic knowledge of eigenvalues and eigenvectors in the context of linear algebra.

2.1 Why are they important

Eigenvectors and Eigenvalues allow us to transform a problem into its natural coordinate system where highly complex operators are greatly simplified. Here are some examples.

2.1.1 In Mathematics

This is a big topic in linear algebra courses. A few situations in which they where they are useful:

- To solve linear equations.
- To diagonalize or achieve an almost (Jordan Canonical form) diagonal matrix form.
- To simplify quadric equations (by finding the main “principal” axes of symmetry).
- To find the pseudoinverse of a matrix through singular value decomposition and then solve least square problems.

2.1.2 In Physics

- Linear partial (and ordinary) differential equations can be solved with the help of eigenfunctions (which is a generalization of the eigenvector for the space of
functions). In particular wave equations eigenvalues are associated with spatial or temporal frequencies. These identify the natural vibration modes of a system.

- In elasticity theory. The eigenvectors indicate the principal directions of deformation and stress on a body. The eigenvalues provide a measure of the relative size between those deformations.

### 2.1.3 Computer Graphics

We illustrate this section with an example of where the eigenvalues and eigenvectors could play an important role for computational graphics.

**Example 2.1.1**

- Assume you have a given image and want to deform the image in a two-dimensional plane, by stretching it along one direction (the eigenvector direction). This can be achieved by the multiplication of the pixels coordinates by a $2 \times 2$ matrix. Let us build the matrix for illustration. We want to think about diagonal matrices that are easy to comprehend and manipulate.

Starting with the pixel at $(x, y)^T$. We want to stretch this pixel along the 45 degrees direction by a factor of 1.2. We first rotate the pixel 45 degrees, then stretch it with a diagonal matrix

$$D = \begin{pmatrix} 1.2 & 0 \\ 0 & 1 \end{pmatrix}.$$  

and finally we rotate it back -45 degrees to its initial orientation state.

The rotation matrix is

$$\begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix}$$

and since $\cos 45 = \sin 45 = \sqrt{2}/2$ and $\lambda = 1.2$ then

$$A = \begin{pmatrix} \sqrt{2}/2 & -\sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{pmatrix} \begin{pmatrix} 1.2 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \sqrt{2}/2 & \sqrt{2}/2 \\ -\sqrt{2}/2 & \sqrt{2}/2 \end{pmatrix} = \begin{pmatrix} 1.1 & 0.1 \\ 0.1 & 1.1 \end{pmatrix}$$

and the new pixel locations are related by the equation

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1.1 & 0.1 \\ 0.1 & 1.1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.$$
2.1 Why are they important

Figure 2.1: Deformation (stretching by 20\%) of an image along the 45 degrees axis.

Figure 2.1 illustrates the problem. On the left panel we think about a pure isotropic pattern as the black arrows starting at 5 degrees and separated 10 degrees from each other around the unit circle. We apply the operator $A$ and the output is shown in the red arrows. We can see the stretch along the 45 degrees direction. In this direction the two (black before, red after) arrows coincide in direction, but the red arrow is 1.2 times larger as expected. Another direction that remains fixed is the orthogonal to the first. This is the -45 degrees. Here there is only one arrow seen (because they overlay exactly) since there is no change in phase, nor in magnitude. Note that the negative directions also remain invariant. If $\mathbf{x}$ is an eigenvector $-\mathbf{x}$ is an eigenvector as well.

A quick check (Wolfram Alpha)\(^1\) confirms that the eigenvectors/eigenvalues of the matrix $A$ are

$$
\mathbf{x}_1 = \left(\begin{array}{c} \sqrt{2}/2 \\ \sqrt{2}/2 \end{array}\right) \quad \lambda_1 = 1.2
$$

$$
\mathbf{x}_2 = \left(\begin{array}{c} -\sqrt{2}/2 \\ \sqrt{2}/2 \end{array}\right) \quad \lambda_2 = 1.0
$$

the eigenvalues are $\lambda_1 = 1.2, \lambda_2 = 1$.

Now think of a window in your computer screen with a picture. Get the mouse and start stretching the window from a corner. Why does not the picture deform accordingly to the mathematics explained in this example? How about stretching the window from the sides? What would be the algorithm for this stretching of the window from the corner? The answer is that we (because of aesthetics?) like to live in a world of rectangular windows. On the other hand, computer drawing software acknowledges true deformation of shapes according to the rules explained here.

---

\(^1\)http://www.wolframalpha.com/input/?i={{1.1%2C0.1},{0.1%2C1.1}}
2.1.4 In Numerical Analysis

This is along the main lines on this document and a next section will be dedicated to it.

2.2 Eigenvectors and Eigenvalues in Numerical Analysis

When a linear transformation (call it a matrix if you wish) is applied to an eigenvector, its direction does not change. The transformation squeezes it or stretches it but does not rotate it. Why is this important?

Consider iterative methods that are frequently used in inversion. What if we are doing iterations along the eigenvector direction? Assume we have a matrix $A$ for the $n$-dimensional space $\mathbb{R}^n$, with an eigenvector $x$ and an eigenvalue $\lambda$. Let us then assume that the error $dy_n$ at some iteration $n$ is related to a perturbation $dx$, in the form

$$dy_n = A^n dx = A^{n-1} A dx = \lambda A^{n-1} dx = \lambda^n dx.$$

So, no matter how many iterations you apply, the error direction is not changing. The same is not true for the magnitude of the error, which is being scaled continuously. If $\lambda < 1$ (the eigenvalue), the size of the errors diminishes with a 0 limit (as $n \to \infty$). If $\lambda = 1$, the system is frozen, nothing is moving, we are “going in circles”. If $\lambda > 1$, the system will diverge.

Think about how many practical problems use systems like this? Even non-linear problems are linearized and then we have a similar form than the one above.

The less “interesting” matrix is the identical matrix. Every vector in space (do not count zero) is an eigenvector with unit eigenvalue. Nothing will happen if you try to invert for new directions. Of course, 0 is another less-interesting matrix, even though is in the other “extreme” case.

We are interested in problems when the matrix $A$ is not well “behaved”. If the matrix $A$ is not even square, then there are no eigenvalues, all we can do is to use a square version of it: $A^T A$, and find the eigenvalues of this. Those are the singular values $\sigma_i$. Even if $A$ is square, this does not guarantee anything. If it is full rank symmetric then we can find $n$ independent directions and the matrix can be fully diagonalized. If the reader is not familiar with diagonalization of matrices, think about the example 1. From a diagonal we build a triple product with the diagonal in the middle. Diagonalization is the reverse problem. Given a square matrix, find a triple product where the center matrix is diagonal and the extreme matrices are the inverse of each other. Singular value decomposition SVD is a generalization of matrix diagonalization for non-square matrices.

\[ \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \]
can not be diagonalized.

If the error vector $dx$ is not in the direction of an eigenvector, it could happen that it is a linear combination of them. For example, if in 2D, $dx = \mu_1 du_1 + \mu_2 du_2$, where $du_1$ and $du_2$ are linearly independent eigenvectors then

$$dy = A^n dx = A^{n-1} A(\mu_1 du_1 + \mu_2 du_2) = A^{n-1}[(\lambda_1 \mu_1 du_1) + (\lambda_2 \mu_2 du_2)] = \lambda_1^n \mu_1 du_1 + \lambda_2^n \mu_2 du_2.$$ 

If either $\lambda_1$ or $\lambda_2$ is greater than 1, we can expect a divergent solution. If one of the eigenvalues is 1, the corresponding direction will not provide a solution which reduces the error. If both eigenvalues are smaller than one, then the problem converges. This is the reason why the spectral value (the largest of the eigenvalues in absolute value) $\rho(A)$ is such an important norm for a matrix. Finite Difference (FD) implementations that can be condensed into a matrix-vector multiplication benefit from this method for evaluating convergence.

Iterative methods can be shown to originate from finite differences, from matrix splitting techniques (see example in the next few paragraphs), or from descent methods (the main objective of this document), among others.

As an example of how matrix splitting methods generate iterative methods, let us consider the Jacobi method to solve $Ax = b$. We write $A = D + E$, where $D$ is a diagonal matrix with all the diagonal elements of $A$, and $E = A - D$. Then we have

$$Ax = b$$
$$Dx = -Ex + b,$$

from which

$$x = -D^{-1}Ex + D^{-1}b$$

(assuming that the diagonal entries are different from zero).

Now renaming $B = -D^{-1}E$ and $b_0 = D^{-1}b$ we write

$$x = Bx + b_0,$$

which provides the Jacobi iteration

$$x_{i+1} = Bx_i + b_0.$$ 

Let us define the error as $dx = x^* - x_i$, where $x^*$ is the exact solution, then we can write

$$x_{i+1} = Bx^* + B(x_i - x^*) + b_0 = x^* + Bd x_i$$

2.2 Eigenvectors and Eigenvalues in Numerical Analysis
from which

\[ dx_{i+1} = B dx_i \]

Then we have an iterative method where the residuals depend exclusively on the matrix \( B \) and it is necessary that the eigenvalues of that matrix to be smaller in size that 1 for the system to converge.

This is not the only way to split the matrix \( A \) in two matrices. Other methods are Gauss Seidel \(^3\), where \( A \) is split in \( L \) and \( U \) where \( L \) is the lower triangular (including the diagonal) version of \( A \), and \( U \) is the upper triangular, with zeroes in the diagonal; and the Successive over relaxation \(^4\) method, where \( A \) is split in three matrices \( L, D, U \), lower, diagonal, and upper triangular respectively. Then a system is set

\[ (D + \omega L)x = \omega b - [\omega U + (\omega - 1)D]x, \]

for \( \omega > 1 \) known as the relaxation factor. The idea is to reduce the size of the eigenvalues so that the maximum eigenvalue is smaller than 1 to guarantee convergence. In these notes, whenever we refer to the eigenvalues \( \lambda_i \) we think of them as sorted from the largest to the smallest. That is, \( \lambda_1 \) is the largest and \( \lambda_n \) is the smallest (assuming the working space to be \( \mathbb{R}^n \)).

Other methods where the eigenvectors and eigenvalues proof useful are the steepest descent and conjugate gradient methods to solve the problem \( Ax = b \). We discuss those next.

### 2.3 The regularized linear least squares problem

If \( A \) is a non–singular matrix, then the solution to the equation \( Ax = b \) is

\[ x = A^{-1}b. \]

Whether \( A \) is easy or difficult to invert is another story.

In practical problems the equation \( Ax = b \) is such that \( A \) in most cases is not a square matrix. Two cases are observed. The matrix \( A \) is over–constrained, the null space is 0, but the range does not cover the whole space. If it is under–constrained, the null space is non–zero, and the range covers the whole space. In the first case the solution does not exist and we can only get to an approximation. In the second case, there is an infinite number of solutions. The least square problem is posed in such a way that there is always a unique solution. That is, a convex objective function is defined such that the difference between the data \( b \) and \( Ax \) is minimum in the \( \ell_2 \) norm.

\(^3\)http://en.wikipedia.org/wiki/Gauss-Seidel_method

\(^4\)http://en.wikipedia.org/wiki/Successive_over-relaxation
2.3 The regularized linear least squares problem

Tikhonov regularization⁵ suggest to minimize the following objective function

\[ f(x) = \|Ax - b\|^2 + \|\Gamma x\|^2 \]

where the matrix \(\Gamma\) is known as the Tikhonov matrix. The solution by using calculus to this problem is

\[ x = (A^T A + \Gamma^T \Gamma)^{-1} A^T b. \]

It is common to see that \(\Gamma = \lambda I\) where \(\lambda > 0\) and \(I\) is the identity matrix. This will yield the solution

\[ x = (A^T A + \lambda I)^{-1} A^T b. \] (2.3.1)

We could have arrived to this equation in an informal heuristic approach as follows. Start with

\[ Ax = b, \]

if \(A\) is not even square, multiply both sides by its transpose

\[ A^T Ax = A^T b. \]

This is the “squaring” of the “rectangle”. If \(A^T A\) has non–zero eigenvalues, it is non–singular and we are done posing a regular problem. The matrix \(A^T A\) is symmetric and positive definitely. However if the matrix \(A^T A\) has a null space, then we can add \(\lambda I\) with \(\lambda > 0\), but small so that the eigenvalues will be shifted to the positive real line and the matrix would be invertible. This yields

\[ (A^T A + \lambda I)x = A^T b \] (2.3.2)

from which the inversion should provide the solution 2.3.1 above. Appendix C provides derivations of the Generalized Tikhonov Regularization method, which are consistent with the Bayesian approach.

From now on we will, for simplification purposes, refer to the \(Ax = b\), where \(A\) is already positive definite and symmetric. This means that the least square regularization part of the problem was already done in advance.

Let us assume that \(\lambda\) was absorbed into \(A\) already and write instead

\[ A^T Ax = A^T b \]

We now build the quadratic form

\[ f(x) = \frac{1}{2} x^T A^T A x - (A^T b)^T x + c \]

⁵http://en.wikipedia.org/wiki/Tikhonov_regularization
This can be seen as equivalent to the equation

\[
g(x) = \frac{1}{2} (Ax - b)^T (Ax - b) + d
\]  

(2.3.3)

where \(c\) and \(d\) are real numbers. Note that this second expression is the \(\ell_2\) norm of the error (up to a multiplicative and additive constant) \(e = Ax - b\). This is what, in one dimension, we call completion of squares. Let us figure out what is \(d\) in the second equation. Defining \(g\) by expanding it so that \(f = g\).

\[
g(x) = \frac{1}{2} x^T A^T A x - \frac{1}{2} x^T A^T b - \frac{1}{2} b^T A x + \frac{1}{2} b^T b + d
\]

Now \(x^T A^T b = b^T A x\) \(^6\), so

\[
g(x) = \frac{1}{2} x^T A^T A x - (A^T b)^T x + \frac{1}{2} b^T b + d
\]

where we can choose

\[
\frac{1}{2} b^T b + d = c
\]

and so, it is clear that the minimum of \(g(x)\) from equation 2.3.3 is given by

\[x\text{ such that } Ax = b,\]

since \(d\) is not defined in terms of \(x\).

\(^6\)Note that \(x^T A^T b = \langle Ax, b \rangle = \langle b, Ax \rangle = b^T A x\), where the angle brackets denote inner product.
Chapter 3

Steepest Descent Method

The Steepest Descent (SD) method considers a search direction along the negative gradient direction \( p = -\nabla f(x) \). For this reason is sometimes known as Gradient Descent \(^1\). We want to warn that there is another Method of Steepest Descent \(^2\), which is used to evaluate integrals using stationary phase conditions, which has not relation which what we call here the method of SD, therefore the gradient descent method seems more appropriate. However history prevails, and after all, the negative gradient is the direction of SD, so we will stick to the name steepest descent.

3.1 The Quadratic Minimization equivalence to a linear problem \( Ax = b \)

Finding the minimum of a quadratic function is equivalent to solve a linear system. That is, let us define the function

\[
f(x) = \frac{1}{2} x^T A x - b^T x + c. \quad (3.1.1)
\]

Then from calculus we know that if \( A \) is symmetric and positive definite \(^4\), \( f(x) \) has a minimum and it satisfies,

\[
\nabla f = 0.
\]

at that minimum. But

\[
\nabla f(x) = Ax - b. \quad (3.1.2)
\]

\(^1\)http://en.wikipedia.org/wiki/Gradient_descent
\(^2\)http://en.wikipedia.org/wiki/Method_of_steepest_descent
\(^3\)Note that this function is in the form of an objective function for the regularized least squares problem in section 2.3
\(^4\)that is, for any \( x \) we need to have \( x^T A x > 0 \)
This means

\[ Ax = b. \]

So, finding the minimum of \( f(x) \) is equivalent to solve the linear system \( Ax = b \). Why do we assume that the matrix \( A \) is symmetric and positive definite? The answer lies in the search for a least square minimization of \( f(x) = \frac{1}{2} \| Ax - b \| \). The gradient of \( f \) is given by

\[ \nabla f(x) = A^T(Ax - b), \]

so that the extreme (local minimum or maximum), if it exists requires that \( \nabla f(x) = 0 \) so that we need \( Ax = b \). Now, we could have arrived to this expression by premultiplying \( Ax = b \) by \( A^T \) and then write

\[ A^T A x = A^T b \]

so that here \( A^T A \) is symmetric. We would like to have it positive definite so that we can invert it. A different motivation for the symmetry of the matrix \( A \), is by thinking that we want to express a function (scalar field) \( f \) of \( x \) as a truncated Taylor series of order 2. In this case the matrix \( A \) would represent the Hessian which (assuming continuity on its derivatives of first and second order) is symmetric.

We then are interested in the problem \( Ax = b \) with \( A \) being symmetric and positive definite.

Let us in general assume a function \( f(x) \) which is locally convex, that is, for which its Hessian (matrix of second-order partial derivatives) is positive definite. Let us, in addition, assume that we start at some initial point \( x_0 \) and want to find the direction of SD. We know that the gradient is orthogonal to the surface (or contour) levels of \( f \), and this is the direction of steepest ascent. From calculus we know that the gradient is the direction of maximum change, since the directional derivative along a direction \( u \) is given by

\[ Du f = \nabla f \cdot u = \|\nabla f\| \cos \theta \]

where \( \theta \) is the angle made by the gradient and the direction of ascent or descent. The extreme descent is at \( \theta = \pi \), that is when

\[ u = - \frac{\nabla f}{\|\nabla f\|}. \]

### 3.2 The line search and SD algorithm

Let us assume that we start an iterative method at some initial point \( x_0 \), and want to move along the SD. However we do not know how far we should move along that
direction. That is, we want to find a scalar \( t \) such that \( f[x_0 - t\nabla f(x_0)] \) is minimum. Such a minimum is guaranteed by the local convexity of the function. Let us define

\[
\varphi(t) = f[x_0 - t\nabla f(x_0)].
\]  

(3.2.3)

We want to find \( t \) which minimizes \( \varphi \), by taking the derivative, That is, we want to find \( t \) such that

\[
\varphi'(t) = \nabla f[x_0 - t\nabla f(x_0)] \cdot \nabla f(x_0) = 0.
\]  

(3.2.4)

Let us define the next point in the iterative process as

\[
x_1 = x_0 - t\nabla f(x_0),
\]  

(3.2.5)

Equations 3.2.4 and 3.2.5 tell us that gradient at the new starting point \( \nabla f(x_1) \) should be orthogonal to \( \nabla f(x_0) \), and the sequence proceeds in the same way. The path created this way is a zig–zag path with 90 degrees turns at each new point \( x_k \). There is no a unique way to determine \( t \). This problem of finding \( t \) is called line search. If we assume that locally the surface (or hyper–surface) is quadratic, we can find an explicit formula for \( t \). That is, if

\[
f(x) = \frac{1}{2}x^TAx - b^Tx + c
\]  

(3.2.6)

where \( A \), the Hessian of \( f \), and \( b \), the linear term are known. Then we have

\[
\nabla f(x_0) = Ax_0 - b = r_0,
\]  

(3.2.7)

where \( r_0 \) is the residual between the \( Ax_0 \) and \( b \).

In equation 3.2.5 we multiply both sides by \( A \) and subtract \( b \), to get

\[
Ax_1 - b = Ax_0 - b - tA\nabla f(x_0)
\]  

which is

\[
r_1 = r_0 - tAr_0 = \nabla f(x_1).
\]

We now multiply (inner product) both sides with \( r_0 \). Since \( r_1 \) is orthogonal to \( r_0 \)\(^5\)

\[
0 = \|r_0\|^2 - tr_0^TAr_0,
\]

from which

\[
t = \frac{\|r_0\|^2}{r_0^TAr_0}.
\]  

(3.2.8)

\(^5\)Note that \( r_0 \) is the gradient of \( f(x) \) at \( x = x_0 \) and \( r_1 \) is the gradient of \( f(x) \) at \( x = x_1 \). Then look at equation 3.2.4.
Here, the residual $r_0$ is the error in the range (data space) for the first iteration. The error in the model domain for any iteration $k$ is defined as $E_k = x_k - x_{k-1}$.

One question is important. When to stop the iterative process? One answer could be given by the maximum number of iterations to try. Another could be by a threshold on the error either in the model space or in the data space. Absolute errors are not good for stopping criteria since they have dimensions and it is hard to say how big is big or how small is small. We illustrate the algorithm below with the criteria of relative residual (in norm $\ell_2$) error smaller than some threshold value $\epsilon$ and maximum number of iterations. Let us call

$$
\varepsilon_i = \frac{\|r_k\|}{\|r_{k-1}\|}
$$

and the maximum number of iterations $M$, so assuming that $A$ symmetric positive definite, we can find an explicit algorithm to compute the local minimum of the function $f$. This is, the SD method:

Initialization:

- $i = 0$
- pick $x_i$
- $r_i = Ax_i - b$
- $t = \frac{\|r_i\|^2}{r_i^T A r_i}$
- $\varepsilon_i = 2\epsilon$

Loop:

while ($\varepsilon_i > \epsilon$ and $i < M$) do

- $i = i + 1$
- $x_i = x_{i-1} - t r_{i-1}$
- $r_i = Ax_i - b$
- $t = \frac{\|r_i\|^2}{r_i^T A r_i}$
- $\varepsilon_i = \frac{\|r_i\|}{\|r_{i-1}\|}$
end do

Finalization:

return $x_i$

Figure 3.1: SD to solve for $x$ in $Ax = b$
3.3 Error Analysis

Note the evaluation of the error $E_k$ as the difference between the approximate $x_k$ and exact solution $x^\ast$. That is

$$\epsilon_k = x_k - x^\ast.$$  

We have no way to check this error, unless we know the exact solution. That is why measuring the error in the range of $A$, (in the data domain between two consecutive $r_k$ vectors) or two consecutive model-domain $x_k$ vectors is usually required. However we will use the error formula above to study convergence speed.

Also, since $Ax^\ast = b$

$$r_0 = Ax_0 - b = A(x_0 - x^\ast) = A\epsilon_0. \tag{3.3.9}$$

We use Greek $\epsilon_k$ for the error with respect to the exact solution and capital $E_k$ for the error from the current to the previous iteration (this last error we could surely estimate on a computer program).

Let us define a new norm

$$\|r\|_A^2 = r^T A r$$

which is a weighted norm according to $A$. If $A$ is positive definite and symmetric, this is formally a norm as defined in linear algebra. Then we simplify notation by saying that

$$t = \frac{\|r\|_I^2}{\|r\|_A^2}$$

is the ratio of two weighted norms where the numerator uses the identity matrix, since $r^T I r = r^T r$. In this sense $t$ is a scaling factor of the relative (to $A$) norms of the error vector $r$ in the range of the matrix $A$. If $A$ is an orthogonal matrix (such a rotation or reflection, for example) then it is easy to show that $\|r\| = \|r\|_A$ so $t = 1$ regardless of the initial guess and iteration number.

We now illustrate what we know of eigenvectors and eigenvalues to analyze the rate of convergence of the SD method.

Let us start with the simplest (uninteresting) case where $A$ is the identity matrix (or a scaled version of it). This is uninteresting because the problem

$$Ax = b, \tag{3.3.10}$$

Rotations and reflections do not change the size of the objects they are acting upon. This makes sense. The weighted norm is a quadratic form. The quadratic form of the identity is a sphere (or hypersphere). A rotation matrix is a rotation of the identity matrix, which geometrically is the rotation of a sphere, which is the same sphere.
Figure 3.2: Paraboloid of revolution corresponding to the equation $f(x, y) = x^2 + y^2 + 10$. The contours are concentric circles with the minimum $x_1 = 0$ as the common center. To get from an initial guess $x_0 = (\sqrt{35}, \sqrt{35})$ to the global minimum $0$, only one step is needed.

has obviously the solution $x = b$ and there is no need to use any sophisticated technique to solve equation 3.3.10 ; however this helps us in understanding. Let us follow the steps suggested by the SD method 3.

Pick any non-zero $x_0$, so $r_0 = x_0 - b$, and $t = 1$, $x_1 = b$. So, if $x_1 = b$ we will get $r_1 = 0$, and $x_1 = x$. So at this point the solution is stationary (fixed point) and the error is zero after just one iteration. The problem does not change if instead we use a scalar multiple of the identity matrix, since then if instead of $A$, we use $\alpha A$, for $\alpha \neq 0$, we can write the problem as

$$Ax = \frac{1}{\alpha} b,$$

and rename $b' = (1/\alpha)b$, the solution gets scaled down, but the problem also converges in one iteration.

This makes sense. Think about $x \in \mathbb{R}^2$, then the equation $f(x) = \frac{1}{2} x^T x - x^T b + c$, is a paraboloid with its minimum located at $x = b$, and its contours are circles. Pick any point in the surface, the gradient points along the radial direction of the contours (circles), the negative gradient point toward the center of the circles (the global minimum). So it is obvious that this is the direction of SD. Now, The step size is $t = 1$, meaning that we should advance exactly $r = \|x_0\|$ which is the radius of the contour where the point $x_0$ is located.

For example, the equation $f(x) = x^T x + 10$ is shwon in Figure 3.2. Obviously
the minimum is at \( x = 0 \). Let us start our initial guess at the point \((x, y)\) on the contour 80, that makes a 45 degree angle with the horizontal. That is \( x = y \) and so \( 2x^2 = 80 - 10 = 70 \). So the coordinates of this point are
\[
x_0^T = (\sqrt{35}, \sqrt{35})
\]
and here \( b = 0 \) and \( r_0 = x_0 \), and since \( t = 1 \), \( x_1 = x_0 - r_0 = 0 \). So the solution \( x_1 = 0 \) is obtained in one step.

A more interesting example is that of a matrix where the eigenvalues are different. To make the example easy, let us think about the matrix
\[
A = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}
\]  
(3.3.11)
where \( \lambda_1 \neq \lambda_2 \).

**The Lucky Numbers:**

Let us make a stop and think about what could be good guesses in any situation for the matrix \( A \) given above.

- Pick a guess which is the minimum. That is, here \( r_0 = 0 \), and there is nothing to do. This might never happen in practical situations.
- Pick an initial guess \( x_0 \), such that \( r_0 \) is an eigenvector.

We apply the SD method 3.1 to this matrix, assuming that the \( r_0 \) is an eigenvector with eigenvalue \( \lambda \), then
\[
r_0 = Ax_0 - b
\]
is such that
\[
Ar_0 = \lambda r_0,
\]
and from equation 3.2.8 we see that
\[
t = \frac{r_0^2}{\lambda ||r_0||^2} = \frac{1}{\lambda},
\]
where if \( r_0 = 0 \), we are done, otherwise we are safe dividing by its norm. Also, \( \lambda \) is either \( \lambda_1 \) or \( \lambda_2 \). The next point in Algorithm 3.1 is \( x_1 \) is
\[
x_1 = x_0 - tr_0
\]
and so
\[
Ax_1 = Ax_0 - tAr_0 = Ax_0 - t\lambda r_0 = v_0 + b - v_0 = b.
\]
So $x_1$ is the solution that produces $Ax_1 = b$ with zero error. That is, it only took us one iteration to get the minimum.

For example, let us define $f(x) = \frac{1}{2}x^T Ax + 10$, with

$$A = \begin{pmatrix} 2 & 0 \\ 0 & 10 \end{pmatrix}$$

There is nothing special about this being two-dimensional or diagonal. The same idea works the same for $n$ dimensions (but harder to plot) and non-diagonal matrices.

Explicitly this is

$$f(x, y) = x^2 + 5y^2 + 10.$$ 

The eigenvectors are along the main directions $e_1 = (1, 0)^T$ and $e_2 = (0, 1)^T$. Let us for example start the initial guess $x_0$ at a contour $f(x, y) = 800$, with $(x, y) = s (1, 0)$, That is

$$f(x, y) = s^2 + 10 = 800.$$ 

Pick $s = \sqrt{790}$, and $(x, y) = (s, 0)$. Figure 3.3 illustrates this example.

So far we found two ways to get to the answer in only one iteration. That the matrix $A$ is a multiple of the identity matrix, or that the initial residual (the SD direction $r_0$) is along any of the axes of the ellipse (an eigenvector of the matrix $A$). These possibilities are achieved in theoretical problems but when we get to numerics, they are unlikely. So, we should think of more realistic situations. We now show an error analysis between consecutive iterations.

---

7 which could be diagonalized assuming they are symmetric and positive definite. This assumption is rooted in the concept of least squares where the Hessian matrix $A$ is actually of the type $A^T A + \lambda I$ for some $\lambda \geq 0$. See section 2.3.

8 You should note that in the examples that we show $b = 0$, and also $x$, instead of $x - x_0$, and $y$ instead of $y - y_0$ are used. Here $(x_0, y_0)$ will move the whole picture to a new origin in the $(x, y)$ plane. A $b$ will also translate the whole picture to a new origin in the $(x, y, z)$ 3D volume. In the same way a non diagonal matrix will rotate the whole object without changing its shape. The ideas of descent methods are local ideas and they do not change due to a global translation or rotation of the object. Mathematically a rotation about the vertical axis does not change the minimum. Global translations change the location of the coordinates but not the concepts explained here. Putting the object in its simplest form simplifies work, while keeping the essence of the method intact. The same idea applies for higher dimensions.
3.3 Error Analysis

We study the error $E_1$ in the next few lines. There is nothing particular about this error. The same analysis applies when changing $E_1$ by $E_k$, except that 1 is at the front of the iteration system.

Let us write the error $E_1$ in terms of the eigenvectors $v_i$ of the matrix $A$ and the eigenvalues $\lambda_i$. That is,

$$E_0 = \epsilon_0 = \sum_i \alpha_i v_i,$$

From equation 3.3.9,

$$r_0 = AE_0 = \sum_i \alpha_i \lambda_i v_i,$$

We compute $t$

$$t = \frac{\|r_0\|^2}{r_0^T Ar_0}.$$ 

First

$$\|r_0\|^2_A = r_0^T Ar_0 = \sum_i \alpha^2_i \lambda_i^3.$$
where each \( \lambda_i \) is the eigenvalue of \( \mathbf{v}_i \), and from the orthonormal condition \( \mathbf{v}_i^T \mathbf{v}_j = \delta_{ij} \), with the Kronecker delta defined as

\[
\delta_{ij} = \begin{cases} 
1 & \text{if } i = j \\
0 & \text{if } i \neq j
\end{cases}
\]

and similarly

\[
\| \mathbf{r}_0 \|^2 = \mathbf{r}_0^T \mathbf{I} \mathbf{r}_0 = \sum_i \alpha_i^2 \lambda_i^2
\]

where we use the result above with the eigenvalues all 1, corresponding to the identity matrix.

Then, the initial \( t \) is

\[
t = \frac{\sum_i \alpha_i^2 \lambda_i^2}{\sum_i \alpha_i^2 \lambda_i^3}.
\]

We have, after one iteration

\[
E_1 = E_0 - t \mathbf{r}_0
\]

\[
= E_0 - \frac{\sum_i \alpha_i^2 \lambda_i^2}{\sum_i \alpha_i^2 \lambda_i^3} \mathbf{r}_0
\]

\[
= E_0 - \frac{\sum_i \alpha_i^2 \lambda_i^2}{\sum_i \alpha_i^2 \lambda_i^3} \mathbf{A} E_0
\]

\[
= E_0 - \frac{\sum_i \alpha_i^2 \lambda_i^2}{\sum_i \alpha_i^2 \lambda_i^3} \sum_j \alpha_j \lambda_j \mathbf{v}_j
\]

\[
= \sum_j \alpha_j \mathbf{v}_j - \frac{\sum_i \alpha_i^2 \lambda_i^2}{\sum_i \alpha_i^2 \lambda_i^3} \sum_j \alpha_j \lambda_j \mathbf{v}_j
\]

\[
= \sum_j \left( 1 - \frac{\sum_i \alpha_i^2 \lambda_i^2}{\sum_i \alpha_i^2 \lambda_i^3} \right) \alpha_j \mathbf{v}_j
\]

(3.3.12)

If in particular all eigenvalues are equal (the case of a spherical quadratic) we see that

\[
\sum_j \alpha_j (1 - 1) \mathbf{v}_j = 0
\]

from which is clear that the convergence is achieved in one step. This makes sense since the quadratic is spherical (level) surface and the radius is orthogonal to the surface so the solution goes directly from the spherical (level) surface to its center.

Let us, for the moment, assume that we are in \( \mathbb{R}^2 \), the error is represented by

\[
E_1 = \alpha_1 \left( 1 - \frac{\alpha_1^2 \lambda_1^3 + \alpha_2^2 \lambda_2^3}{\alpha_1^2 \lambda_1^3 + \alpha_2^2 \lambda_2^3} \right) \mathbf{v}_1 + \alpha_2 \left( 1 - \frac{\alpha_1^2 \lambda_1^2 \lambda_2 + \alpha_2^2 \lambda_2^3}{\alpha_1^2 \lambda_1^3 + \alpha_2^2 \lambda_2^3} \right) \mathbf{v}_2
\]
Then, let us divide the fractions by $\lambda_2^3$ (both numerator and denominator), and call $\kappa = \lambda_1/\lambda_2$ the conditioning number, so

$$E_1 = \alpha_1 \left(1 - \frac{\alpha_1^2 \kappa + \alpha_2^2 \kappa}{\alpha_1^3 \kappa^3 + \alpha_2^3}ight) v_1 + \alpha_2 \left(1 - \frac{\alpha_1^2 \kappa^2 + \alpha_2^2}{\alpha_1^3 \kappa^3 + \alpha_2^3}ight) v_2$$

Assume $\alpha_1 \alpha_2 \neq 0$ (otherwise $E_0 = 0$ so the convergence is achieved in one step). The minimum value of $\kappa$ is 1, for which $E_1 = 0$, then $\kappa$ grows, the first fraction starts converging to 1, while the second starts decreasing (since $\kappa^3$ in the denominator dominates over $\kappa^2$ in the numerator). In the limit when $\kappa$ goes to $\infty$, the first fraction goes to 1 while the second fraction goes to zero and $E_1$ will get to the value

$$\lim_{\kappa \to \infty} E_1 = \alpha_2 v_2.$$

That is why the condition number $\kappa$ is important. Having it close to unity is ideal for speeding convergence.

If either $\alpha_1$ or $\alpha_2$ is zero, then $E_1 = 0$ as expected, since this means that we picked the initial point so that $E_1$ is in the main axis of the ellipse.

We now see that two sets of parameters control the convergence. The coefficients $\alpha$ which indicates how far we are from a given eigenvector direction, and the eigenvalues condition number $\kappa$ which is the ratio between the largest and smallest eigenvalue.

In general, let us find bounds for $E_1$ in equation 3.3.12. We assume $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n > 0$.

Let us focus in the fraction

$$\beta_j = \sum_i \frac{\alpha_i^2 \lambda^2_i \lambda_j}{\sum_i \alpha_i^2 \lambda_i^3}$$

and from the decreasing order of the $\lambda_i$s we can assure that

$$\sum_i \frac{\alpha_i^2 \lambda^2_i \lambda_j}{\sum_i \alpha_i^2 \lambda_i^3} \geq \frac{\sum_i \alpha_i^2 \lambda_i^3}{\sum_i \alpha_i^2 \lambda_1^3} = \frac{\lambda_j^3}{\lambda_1^3} = \frac{1}{\kappa^3}.$$ 

Similarly

$$\sum_i \frac{\alpha_i^2 \lambda^2_i \lambda_j}{\sum_i \alpha_i^2 \lambda_i^3} \leq \frac{\sum_i \alpha_i^2 \lambda_i^3}{\sum_i \alpha_i^2 \lambda_n^3} = \frac{\lambda_j^3}{\lambda_n^3} = \kappa^3.$$ 

That is

$$\frac{1}{\kappa^3} \leq \beta_j \leq \kappa^3.$$ 

For $\kappa = 1$, $\beta_j = 1$, the algorithm converges in one iteration, but as $\kappa$ grows beyond 1, the possible values of $\beta$ can shrink close to 0 (when looking at the left side of the expression) or grow without bounds (when looking at the right side of the expression) leaving a large room for errors, which can slow the convergence of the process.
To illustrate the case when convergence can be slow, let us consider the diagonal matrix

$$A = \begin{pmatrix} 10 & 0 \\ 0 & 1 \end{pmatrix}$$

and the vector $b = (1, -1)^T$.

The corresponding quadratic equation is

$$f(x, y) = \frac{1}{2} (x, y) \begin{pmatrix} 10 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} - (1, -1) \begin{pmatrix} x \\ y \end{pmatrix} = 5x^2 + y^2/2 - x + y \ (3.3.13)$$

The function has an easy-to-find analytical solution:

$$x = A^{-1}b = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 0.1 \\ -1 \end{pmatrix}$$

$$z = f(0.1, -1) = -0.55$$

The Matlab code in Appendix B computes a few SD iterations toward the minimum using the algorithm 3.1. Here we choose a tolerance of 0.01 and a maximum of 100 iterations.

Figures 3.4 and 3.3.1 shows the plots computed by the Matlab script.

From the Figure we observe that the convergence starts fast and becomes asymptotically slow. Every path leg is tangential to a contour and the following leg, orthogonal to the same contour, until hitting the next contour in the downhill direction. This long zig-zag pattern is typical when condition numbers are big (and 10 is big). In fact, $r_k^T r_{k+1} = 0$, and since $r_k = (1/t)(x_k - x_{k-1})$, each leg (the segment connecting the points $x_k$ and $x_{k+1}$) of the path is orthogonal to the previous leg.

This explains the orthogonality shape of the zig-zag path, traced by the segments joining the points $x_k$ as indicated above. The 3D version of this example is also shown in Figure 3.3.1.

### 3.3.2 Error analysis $\epsilon_k$ relative to the exact solution

To end this section, we would like to reiterate about the importance of the condition number to understand convergence of iterative methods to solve linear systems. In this link\(^9\) on Tom Lyche’s\(^10\) website The following inequality is proven.

For the $A$-norms\(^11\) of the errors in the SD method, the following upper bounds hold

$$\frac{\|x^* - x_k\|_A}{\|x^* - x_0\|_A} \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \leq e^{-\frac{2}{\kappa}k}, \quad k \geq 0. \quad (3.3.14)$$

\(^9\)http://folk.uio.no/georgmu/cg.pdf
\(^10\)https://sites.google.com/site/georgmuntingh/
\(^11\) $A$-norm of $x$ means $x^T A x$, which has all the properties of a norm.
This figure shows a sequence of points picked through the SD journey (from top to bottom). The initial point is at \((x_0, y_0) = (0, 0)\), and from there a normal departs from each ellipse (contour level) until it touches the next ellipse. This creates a zigzag pattern (sometimes known as ping-pong effect) that converges slow in an infinite number of iterations to the solution. The orthogonality between successive steps is shown in equations 3.2.4 and 3.2.5, by construction of the first and the following descent paths.

Figure 3.5: A quadratic surface \(f(x, y) = 5x^2 + (1/2)y^2 - x + y\) and the sequence (path) of descent toward the minimum. The figure shows the 3D path with the projected version on the 2D plane.

Figure 3.6: Here the \(z\)-values are represented and we see that the convergence is fast at the beginning but becomes asymptotically slow approaching the minimum \(z = -0.55\) after 10 iterations.
We reproduce the proof of this in Appendix A, in section A.8. The first consequence of inequality 3.3.14 is that if the matrix $A$ is positive definite, then the SD method converges, since

$$\frac{\kappa - 1}{\kappa + 1} < 1,$$

and its power decreases to 0.

The slow convergence of the SD method is the main motivation for the method of conjugate gradient which I explain next.
Chapter 4

The Conjugate Gradient (CG) Method

4.1 Origin

The previous section served as a motivation for the CG method. The method of CG was first published in December of 1952 by Magnus Hestenes and Eduard Stiefel under the name Method of Conjugate Gradients for Solving Linear Systems. Hestenes and Stiefel derived the method independently and decided to publish it together.

4.2 Developing of The Method

Once we know that we are sitting in a level surface (or contour) of the function $f(x, y)$, (i.e., in a tangent direction to it), we found, from the SD method that the gradient direction (normal to this tangent vector) do not provide an optimal descent. Let us further illustrate this with help of Figure 4.1.

Imagine that we start the descent toward the minimum at some point $x_0$. A ray tangentially shot to the black circle kisses it at point $x_1$, and from there leads directly to the center of the circle by shooting a perpendicular in the radial direction (the black part of the figure). This is the easy problem that we know how to solve. Let us assume that we deform the space by stretching the $x$ axes by 2. We would obtain the figure in blue. This figure is an ellipse, and the two ray paths are no longer orthogonal. If we deform the figure and preserve the SD method we should travel through the orthogonal direction marked by the red dashed arrow, but this will not take us where we want. We see that the path is deformed accordingly and we should hit the minimum by moving through the new deformed (tilted) arrow from $x_1$ to $x$. So, the idea is that in this second (deformed) problem we should acknowledge the deformation factor (stretching of the $x$–axis) and consider a new definition of orthogonality that do not confine with the red dashed arrow, but with the new blue arrow ending in the minimum $x$.

Figure 4.1: Assume an initial point \( x_0 \) is chosen and a tangential ray along some descent path (a negative gradient) is traced kissing the black circle at point \( x_1 \). From there a perpendicular is traced to the center of the circle where the minimum is sitting. This is the black picture. The blue picture was constructed exactly with the same algorithm but the horizontal scale was stretched by 2. Now the vector \( x_1 - x_0 \) is no longer perpendicular to the new radial chord \( x_1 - x \). The circles in the figure (even the dots) changed into ellipses. The dotted red line indicates the SD direction, which is no longer appropriate for the new deformed problem.

In fact, here is the piece of TikZ\textsuperscript{2} code that draws Figure 4.1.

\begin{verbatim}
\begin{tikzpicture}[scale=2.0]
\draw[line width=2] (0.,0.) circle (1);
\draw[line width=1] (0.707,0.707) circle (1pt);
\draw[line width=2, color=black] (0.,0.) circle (1pt);
\draw[line width=1, dashed] (-1,0) -- (1,0);
\draw[-latex, line width=2] (0,1.4142) -- (0.707,0.707);
\draw[-latex, line width=2] (0.707,0.707) -- (0,0);
\draw (0,1.4142) node[above] \$x_0$;
\draw (0.707,0.707) node[right] \$x_1$;
\draw (0,0) node[below] \$x$;
\begin{scope}[xscale=2.0, yscale=1, color=blue]
\draw[line width=2] (0.,0.) circle (1);
\draw[line width=1] (0.707,0.707) circle (1pt);
\draw[line width=1] (0.,0.) circle (1pt);
\draw[line width=1, dashed] (-1,0) -- (1,0);
\draw[-latex, line width=2] (0,1.4142) -- (0.707,0.707);
\draw[-latex, line width=2] (0.707,0.707) -- (0,0);
\draw[-latex, line width=2, red, dashed] (0.707,0.707) -- (0.5,0);
\draw (0.707,0.707) node[above] \$x_1$;
\end{scope}
\end{tikzpicture}
\end{verbatim}

\textsuperscript{2}http://en.wikipedia.org/wiki/PGF/TikZ
The reader do not have to know programming to judge that the two pieces of code (before \texttt{\begin{scope}}” and starting after the line with “\texttt{\begin{scope}}’’) are identical (except for the labels and color names). The only difference is the stretching of scale on the line

\begin{scope}[xscale=2.0, yscale=1, color=blue]

which clearly shows that the scaling along the $x$–axis is 2 keeping the $y$ axis with scale of 1.

We can see that the code is the same in both cases but the $x$ axis was distorted (stretched by 2) in the second case.

The mathematical representation of the deformation is that of $x' = Ax$ with

$$A = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}.$$ 

The problem can be extended to more than simply two dimensions. In all iterative descent methods we know the current and previous directions. These two directions (call them if you wish $r_{k-1}$ and $r_k$) expand a plane, and the intersection of that plane with the global quadratic (hyper–ellipsoid) surface is an ellipse just as the one considered here. We will investigate what is the correct bending to hit the minimum in few steps.

With this observation we set up the problem mathematically. The key point is that a symmetric positive definite matrix will deform a circle into an ellipse and the center of both (the minimum location) is the same. So the ray path, which in the sphere bends in a 90 degrees direction, it does not bend the same in the ellipse to get to the same point.

Let us go back to the 2D case. If $A$ is symmetric positive definite, the graph of

$$f(x) = x^T A x = 1$$

is an ellipse, with semi–axis $\lambda_1$, $\lambda_2$, being the eigenvalues of $A$, and the eigenvectors corresponding with these eigenvalues are the principal directions of the ellipse.

The concept of conjugate diameters is well explained in Wikipedia\footnote{http://en.wikipedia.org/wiki/Conjugate_diameters} and particularly that of conjugate diameters of an ellipse: “for an ellipse, two diameters are conjugate if and only if the tangent line to an ellipse at an end point of one diameter is parallel to the other diameter.

Figure 4.2 shows the same ellipse shown in Figure 4.1. We see two diameters. The green diameter is parallel to the two tangents at the end points of the brown diameter and vice–versa. The two conjugate diameters intersect at right angles if the ellipse is a circle. However it is interesting to see that the intersection looks as a right angle intersection if we think of the figure as a 3D figure seen from above (or below). As a true 2D figure, the angles where the two diagonals intersect are not orthogonal. The
Figure 4.2: There are two diameters here. The green diameter is parallel to the tangents at the ends of the brown diameter. Equally, the brown diameter is parallel to the tangents sitting at the ends of the blue diameter. The green and brown diameters are conjugate diameters. It is interesting that if you think of this as a 3D picture the angles look orthogonal, while if as a 2D picture the top and down angles are obtuse and the side angles are acute.

only case where they are perpendicular is if the diameters are along the principal directions (the eigenvector directions), which provide a fast track convergence of the SD method as shown above.

Figure 4.3 shows an sketch of a 3D ball after being deformed by a linear positive definite symmetric transformation $A$ with three different eigenvalues. We start on the left side. If our initial point is $x$ and the matrix $A$ would be the identity, then the quadratic form $f(x) = x^T A x$ would correspond to a unit sphere and we can go directly from the initial guess to the center of the sphere, which holds the minimum location. The interesting case is that of $A$ being different from the identity. We see here (on the right) the transformation from a unit ball to an ellipsoid through the matrix $A$. The origin is the point $O$. The initial point is $x_0$. We start tracing a vector along the gradient direction $r_0 = b - Ax_0$, which is the direction of $x_1 - x_0$, in a way the ray touches the ellipsoid tangentially. The plane having the vectors $x_0$ and the gradient $r_0$ is now the local plane of search. That plane intersects the ellipsoid in the ellipse which in the figure is centered at $x_2$ and passes through $x_1$. The conjugate direction will go straight from the point $x_1$ tangential to the ellipse to its center at $x_2$. Now, this point $x_2$ is not the center of the ellipsoid (unless the second and third eigenvalues are equal) so we landed in a new ellipse on the plane formed by the two previous search directions. This ellipse is an ellipse in another space and it is not part of the original ellipsoid. It is like a projection of the intersection of the plane of the two previous search directions (which is an ellipse) and that plane into a lower dimensional space represented by the plane. It can be shown that this new ellipse has the center of the ellipsoid as its center. So the next conjugate direction hits the
4.2 Developing of The Method

Figure 4.3: Deformation of a ball according to equation $x' = Ax$ for a positive definite matrix $A$.

center of the ellipsoid which is the location of the minimum we are searching for. We wish, we could go in just one iteration through the red dotted line. This is equivalent to know where the center of the ellipsoid is and go directly there. That is, this implies that we know $A^{-1}$. The method would not be iterative.

Let us assume that the ellipse in 4.2.1 is parameterized with some parameter $t$, that is $x = x(t)$. If we take the derivative with respect to $t$ we see that

$$\frac{df(x)}{dt} = \left(\frac{dx}{dt}\right)^T A x + x^T A \frac{dx}{dt} = 2x^T A \frac{dx}{dt} = 0.$$  

Now, let the vector $x$ be a vector in the ellipsode surface. The vector $t = dx/dt$ is tangent at the base of tip of the arrow of $x$. That is, the conjugated directions for $x$ and $t$ satisfy the relation

$$x^T A t = 0.$$  

This is the relation that we are searching for. It is call $A$–orthogonality. If $A$ is a multiple of the identity, that is the same as the traditional orthogonality. Since $A$ is symmetric positive definite, the relation

$$\langle x , y \rangle_A = x^T A y$$

is an inner product,\footnote{http://en.wikipedia.org/wiki/Inner_product_space} and

$$\langle x , x \rangle_A = x^T A x = \|x\|_A^2.$$
defines a norm $\|x\|_A$, known as the Mahalanobis norm\(^5\).

Now we know which direction to shoot (a conjugate direction) but we need to know the step size. However, the conjugate direction is not explicit. All we know is that we should shoot in a new direction $r_{k+1}$ such that

$$r_{k+1}A r_k = 0.$$  

The algorithm is a generalization of the Gram–Schmidt orthogonalization where this last algorithm uses the identity $I$ instead of $A$.

First, we want to realize that the residual vectors $r_k$ are along the gradient direction and so they cannot be used (except at the first step, to get started) for the shooting directions. We then introduce new symbols $p_k$ for the shooting directions. We want to set up a basis of the vector space in terms of the CG directions $p_k$.

Let us next show that these vectors are linearly independent and so (assuming we can find $n$ of them in $\mathbb{R}^n$) they expand the whole space.

Assume linear combination

$$\sum_{i=1}^n \alpha_i p_k = 0.$$

then

$$\sum_{i=1}^n \alpha_i A p_k = 0.$$

and multiplying by $p_j^T$ we see that

$$\sum_{i=1}^n \alpha_i p_j^T A p_k = 0.$$

now, since

$$p_j^T A p_k = 0, \text{ for } i \neq j,$$

we find

$$\alpha_j p_j A p_j = 0,$$

and since $A$ is positive definite, $p_j A p_j > 0$ then $\alpha_j = 0$. So the $p_k$ vectors are linearly independent.

Let us first find the step size. Assume the solution of $Ax = b$ can be written as a linear combination of the $p_k$ conjugate direction vectors. That is

$$x = \sum_{i=1}^n t_i p_i.$$  

\(^5\)http://en.wikipedia.org/wiki/Mahalanobis_distance
We mimic the method used above to show linearly independence. That is, let us multiply first by $A$ to obtain

$$Ax = \sum_{i=1}^{n} t_i A p_i,$$

or

$$b = \sum_{i=1}^{n} t_i A p_i.$$

Now, we multiply by $p_j$, so

$$p_j^T b = \sum_{i=1}^{n} t_i p_j A p_i.$$

Again, since $p_j A p_i = 0$, for $i \neq j$, then

$$p_j^T b = t_j p_j A p_j,$$

and since $A$ is positive definite, and $p_j \neq 0$, then

$$t_j = \frac{p_j^T b}{p_j A p_j}.$$

We find an iterative process that walk us along the different vectors $p_i$. This method resembles the Gram–Schmidt orthogonalization.

As with the SD method we assume a starting point $x_0$, and an initial $p_0 = r_0 = A x_0 - b$ which is in the direction of the SD (or negative gradient). Here is the key step. What is the next ($p_1$) direction? We should find a $p_1$ such that

$$p_1^T A p_0 = 0.$$

Let us review the traditional Gram-Schmidt for regular orthogonalization. For simplicity we only think in $\mathbb{R}^2$ and that we have an initial vector $p_0$ and an residual vector $r_1$ which is not in the direction of $p_0$. Then, by subtracting the projection of $r_1$ into $p$, we find the orthogonal component $p_1$ of $r_1$ to $p_0$.

That is, we want to say that $r_1$ is such that

$$r_1 = t p_0 + p_1$$

where $p_0$ and $p_1$ are orthogonal and $t_0$ is a scalar to be found together with $p_1$. We can multiply $4.2.2$ by $p_0^T$ and obtain

$$p_0^T r_1 = t p_0^T p_0 + p_0^T p_1.$$
from which
\[ t = \frac{p_0^T r_1}{p_0^T p_0} \]
and
\[ p_1 = r_1 - t p_0. \]

Note that this traditional method to solve equation 4.2.2 accomplishes two tasks at the same time. Finding \( t \) and finding the new direction \( p_1 \). This trick works thanks to the orthogonality condition.

Let us try the same trick to compute the \( A \)-orthogonal direction of \( p_0 \). That is, assume that there is a \( p_1 \) vector which is \( A \)-orthogonal to \( p_0 \), and that we can write \( r_1 \) as
\[ r_1 = t p_0 + p_1. \]

We find \( t \) by a proper annihilator and use it to find \( p_1 \). The annihilator is a product by \( A \) first and then by \( p_0^T \). This yields
\[ p_0^T A r_1 = t p_0^T A p_0 + p_0^T A p_1. \]

We see how the annihilator knocked out the \( p_1 \) participation on expression 4.2.3, and paved the way to find
\[ t = \frac{p_0^T A r_1}{p_0^T A p_0} , \quad p_1 = r_1 - t p_0. \]

By direct substitution, the reader can check that \( p_0^T A p_1 = 0 \). Note that when \( A = I \) this is exactly the Gram–Schmidt orthogonalization and what we call a projection coefficient \( t \) is now an \( A \)-projection coefficient.

Figure 4.4 illustrates the case of regular Gram-Schmidt orthogonalization (with \( A = I \), yellow) and extended \( A \)-Gram-Schmidt orthogonalization (blue), for the matrix \( A \) defined in equation 2.1.1. The background (yellow for traditional Gram-Schmidt and blue for extended \( A \)-Gram-Schmidt) shows the deformation taking place due to the matrix \( A \). Think of a computer drawing package and you use the mouse over the upper-right corner of the yellow box to stretch it to obtain the blue box. We see that the \( A \)-orthogonality is achieved by removing the parallel (drop a parallel to the direction \( p_1 \) to the axis along \( p_0 \)) projection along the vector \( p_1 \) onto the vector \( p_0 \). Since the angle between the vectors \( p_1 \) and \( p_0 \) is not 90 degrees, the projection does not drop a perpendicular in the blue frame as it does in the yellow frame.

The blue figure was drawn with the same TikZ \(^6\) algorithm (instructions) as the yellow figure but with the only fundamental difference

\(^6\)http://en.wikipedia.org/wiki/PGF/TikZ
4.3 The Algorithm

Figure 4.4: Gram-Schmid (yellow) and $A$–Gram-Schmid (blue) orthogonalization. The matrix $A$ is that in equation 2.1.1. Here $t = p_0^T A r_1 / p_0^T A p_0$. In the yellow figure instead of $A$, the identity matrix $I$ is used.

\[
p_1 = r_1 - t p_0
\]

\[
p_1 = r_1 - t p_0
\]

Where the identity matrix was used in the first scope and the matrix

\[
A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}
\]

in the second, with a shift of 6 units to avoid image overlapping. This illustrates one more application of matrix transformations in the computer graphics field.

4.3 The Algorithm

Given an initial point $x_0$, we start the iterations by assuming that the search direction is initially $r_0 = p_0 = A x_0 - b$ and

\[
x_1 = x_0 + \alpha_0 p_0,
\]

(4.3.5)

where $\alpha_0$ is a step size that minimize the objective function $f$, in equation 3.2.6. Explicitly, we want to minimize the quadratic form

\[
f(\alpha_0) = \frac{1}{2} (x_0 + \alpha_0 p_0)^T A (x_0 + \alpha_0 p_0) - b^T (x_0 + \alpha_0 p_0) + c
\]

for $\alpha_0$. We write $f'(\alpha_0) = 0$ as

\[
f'(\alpha_0) = p_0^T A (x_0 + \alpha_0 p_0) - b^T p_0 = 0
\]

That is,

\[
p_0^T A x_0 + \alpha p_0^T A p_0 = p_0^T b = p_0^T (A x_0 - r_0) = p_0^T A x_0 - p_0^T r_0.
\]
or

\[ \alpha_0 = -\frac{p_0^T r_0}{p_0^T A p_0}. \]

7The minus sign “-” comes from the fact that we defined \( r_0 = Ax_0 - b \) instead of \( r_0 = b - Ax_0 \). In the literature is common to find this second definition which eliminates the minus sign in the above equation, but introduces other minus signs in the recursions that follow.

In general we can write the recursion formula

\[ x_{i+1} = x_i + \alpha_i p_i \quad \alpha_i = -\frac{p_i^T r_i}{p_i^T A p_i}. \]  
(4.3.6)

Now, \( r_1 \) is found as

\[ r_1 = Ax_1 - b = A(x_0 + \alpha_0 p_0) - b = r_0 + \alpha_0 A p_0. \]

and in general we can write the recursion formula

\[ r_{i+1} = Ax_i - b = r_i + \alpha_i A p_i. \]  
(4.3.7)

Let us change notation in equation 4.2.4, by changing \( t \) for \( \beta_{i+1} \) and writing

\[ \beta_{i+1} = \frac{p_i^T A r_{i+1}}{p_i^T A p_i}, \quad p_{i+1} = r_{i+1} - \beta_{i+1} p_i. \]  
(4.3.8)

(equation 4.2.4 follows by choosing \( i = 0 \), and \( \beta_1 = t \)).

We highlight some orthogonality relations that will be used for the CG linear and non-linear algorithms.

### 4.3.1 Orthogonality Relations between the residual vectors \( r_j \) and the direction vectors \( p_j \)

We isolate two important orthogonality relations between the residual and direction vectors for the CG method because they are useful not only for the linear CG method but also for the non–linear CG method.

Different from SD, where the algorithm could zig-zag between orthogonal and parallel directions along too many iterations, we show that in the CG the \( r_k \) vectors navigate through other dimensions without repeating, that is staying a maximum of two iterations in the same plane without ever returning to it (until convergence where \( r_n = 0 \)), and this navigation is through orthogonal paths. The relevant equations are listed next.

\[ r_i^T p_j = \delta_{ij} \| r_i \|^2 \quad , \quad j < i \]  
(4.3.9)

\[ r_i^T r_j = \delta_{ij} \| r_i \|^2 \]  
(4.3.10)
Let us prove the first equation using induction over \(i\). Set \(j = 0\) and so the smallest \(i\) that we can use is \(i = 1\), and evaluate

\[
\mathbf{r}_1 \mathbf{p}_0 = \mathbf{r}_1 \mathbf{r}_0 = [\nabla f(\mathbf{x}_0 + t \mathbf{p}_0)]^T \nabla f(\mathbf{x}_0)
\]

Now, recall that to find \(t\) we do a line search along the direction \(\mathbf{p}_0\) to minimize the function

\[
g(t) = f(\mathbf{x}_0 + t \mathbf{p}_0),
\]

and that

\[
g'(t) = \nabla f(\mathbf{x}_0 + t \mathbf{p}_0) \cdot \mathbf{p}_0 = 0,
\]

but \(\mathbf{p}_0 = \mathbf{r}_0 = \nabla f(\mathbf{x}_0)\), so

\[
\mathbf{r}_1^T \mathbf{p}_0 = 0.
\]

Let us assume that for some \(i > j\), we have that

\[
\mathbf{r}_i^T \mathbf{p}_j = 0.
\]

and want to show that \(\mathbf{r}_{i+1}^T \mathbf{p}_j = 0\). We write, using 4.3.7

\[
\mathbf{r}_{i+1} = \mathbf{r}_i + \alpha_i A \mathbf{p}_i
\]

and so

\[
\mathbf{r}_{i+1}^T \mathbf{p}_j = (\mathbf{r}_i + \alpha_i A \mathbf{p}_i)^T \mathbf{p}_j = \mathbf{r}_i^T \mathbf{p}_j + \alpha_i A \mathbf{p}_i^T \mathbf{p}_j = 0.
\]

The first 0 is due to the induction hypothesis, the second is because the vectors \(\mathbf{p}_j\) are \(A\)-conjugate and \(j < i\). So every new residual \(\mathbf{r}_i\) is orthogonal to the space spanned by all previous direction search vectors. Now if \(i = j\), we have

\[
\mathbf{r}_i^T \mathbf{p}_i = \mathbf{r}_i^T (\mathbf{r}_i - \beta_{i-1} \mathbf{p}_{i-1}) = \mathbf{r}_i^T \mathbf{r}_i - \beta_{i-1} \mathbf{r}_i^T \mathbf{p}_{i-1} = \mathbf{r}_i^T \mathbf{r}_i - \beta_{i-1} \mathbf{r}_{i-1}^T \mathbf{p}_{i-1}^T \mathbf{p}_{i-1} = \mathbf{r}_i^T \mathbf{r}_i - \beta_{i-1} \mathbf{r}_{i-1}^2,
\]

from 4.3.8

So we showed that for \(j < i\)

\[
\mathbf{r}_i^T \mathbf{p}_j = \delta_{ij} \| \mathbf{r}_i \|^2.
\]

We now prove the second orthogonality condition 4.3.10. Since \(\mathbf{r}_j^T \mathbf{r}_i = \mathbf{r}_i^T \mathbf{r}_j\), then it does not matter if \(j < i\) or \(j > i\) because the dot product is commutative. Then, from equation 4.3.8

\[
\mathbf{p}_i = \mathbf{r}_i - \beta_{i-1} \mathbf{p}_{i-1}.
\]

The vector \(\mathbf{r}_i\) belongs to the hyperplane \(\text{span}\{\mathbf{p}_i, \mathbf{p}_{i-1}\}\) (see equation 4.3.8 ). Similarly, \(\mathbf{r}_{i-1}\) belongs to the hyperplane \(\text{span}\{\mathbf{p}_{i-1}, \mathbf{p}_{i-2}\}\), but \(\mathbf{r}_i^T \mathbf{p}_j = 0\), for \(j < i\), so \(\mathbf{r}_i^T\) is orthogonal to all hyperplane generated by vectors \(\mathbf{p}_k\) and \(\mathbf{p}_{k-1}\) with \(k = 1, \cdots, i\). That is, \(\mathbf{r}_k\) is orthogonal to all vectors \(\mathbf{r}_j, j = 0, \cdots, k - 1\). We have then that

\[
\mathbf{r}_i^T \mathbf{r}_j = \delta_{ij} \| \mathbf{r}_i \|^2.
\]
4.3.2 Finishing Details and The Algorithm

We have all the elements to follow and iterative solution. However equation 4.3.8 can be simplified by avoiding the use of the matrix $A$, and we do that next. In equation 4.3.7 we multiply both sides by $r_i^T$ and find

$$r_i^T r_{i+1} = r_i^T r_i + \alpha_i r_i^T A p_i.$$

Since $p_i^T A r_{i+1}$ is a scalar, it is equal to $r_{i+1}^T A p_i$, and so (provided that $\alpha_i \neq 0$. If $\alpha_i = 0$ then $r_{i+1} = 0$ and we are done since the residual is zero. )

$$p_i^T A r_{i+1} = \frac{r_i^T r_{i+1}}{\alpha_i} = \frac{r_i^T r_{i+1}}{p_i^T A p_i} = -\frac{(p_i^T A p_i)(r_i^T r_{i+1})}{p_i^T r}$$

So

$$\beta_{i+1} = -\frac{r_{i+1}^T r_{i+1}}{p_i^T r_i}.$$ 

The advantage of this representation is that it is independent of the matrix $A$ and then it requires less amount of computations.

Now, with this, and equation 4.3.9, we simplify $\alpha_i$ and $\beta_i$ as follows:

$\alpha_i$ and $\beta_i$ are:

$$\alpha_i = -\frac{r_i^T r_i}{p_i^T A p_i}. \quad (4.3.11)$$

$$\beta_{i+1} = -\frac{r_{i+1}^T r_{i+1}}{r_i^T r_i}. \quad (4.3.12)$$

As indicated while working on the SD method the important question about when to stop the iterative process is posted here again and we follow the same procedure done in SD. That is, we could assume a maximum number of iterations and in addition estimate the threshold on the error in the data space. We take the ratio between two consecutive errors because this will make the error dimensionless and easy to evaluate and understand. We illustrate the algorithm below with the criteria of relative residual (in norm $\ell_2$) error smaller than some threshold value $\epsilon$ and maximum number of iterations. Let us call

$$\varepsilon_i = \frac{\|r_i\|}{\|r_{i-1}\|}$$

and the maximum number of iterations $M$, so assuming that $A$ is symmetric positive definite we find the following
Algorithm 4.3.0

- Pick an initial guess $x_0$.
- Compute the first residual $r_0 = Ax_0 - b$. This is also named $p_0$. With this residual and $\alpha_i$ from equation 4.3.11, find the new $x_1$ and the new residual, that is:

  While the relative residual error $\varepsilon_i$ is larger than some threshold value

  -

  $$x_{i+1} = x_i + \alpha_i p_i$$

  $$r_{i+1} = r_i + \alpha_i Ap_i.$$  

  Use $p_i$ and $r_{i+1}$ to find the new conjugate direction $p_{i+1}$.

  - Using $\beta_i$ from equation 4.3.12:

  $$p_{i+1} = r_{i+1} - \beta p_i,$$

  Or in a more compact form:

  If we create a matrix

  $$S = [p_0 \ p_1 \cdots p_{n-1}]$$

  and since

  $$p_i^T Ap_j = d_{ij}\delta_{ij},$$

  where $d_{ij}$ is a diagonal matrix, we have that

  $$S^T AS = D$$

  is a diagonal matrix. This is, the CG method diagonalizes the matrix $A$ into $D$. In different words. We can define a coordinate transformation where the vector $x$ can be written as $Sx'$ and so the quadratic form 3.1.1, in the new coordinate system $x'$ turns out to be

  $$f(x) = \frac{1}{2}x'^T S^T AS x' - b'^T S x' + c = \frac{1}{2}x'Dx' - b'^T S x' + c$$

  where the quadratic form is diagonalized and so the inversion in this new system can be done along the new coordinate axis in at most $n$ iterations. So, the conjugate gradient method finds the principal axis of the hyper-ellipsoid corresponding to the positive definite (Hessian) matrix $A$.

  Finally, we finish by saying that the name “CG” is a misnomer since what is conjugate are the search directions $p_k$ and not the gradients. Again, history prevails.
Initialization:
\[ i = 0 \]
\[ \text{pick } x_i \]
\[ r_i = Ax_i - b \]
\[ p_i = r_i \]
\[ t = \frac{\|r_i\|^2}{p_i^T A p_i} \]
\[ \epsilon_i = 2\epsilon \]

Loop:
\[ \text{while } (\epsilon_i > \epsilon \text{ and } i < M) \text{ do} \]
\[ i = i + 1 \]
\[ x_i = x_{i-1} - tp_{i-1} \]
\[ r_i = r_{i-1} + tAp_{i-1} \]
\[ \beta = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}} \]
\[ \epsilon_i = \frac{\|r_i\|}{\|r_{i-1}\|} \]
\[ p_i = r_i - \beta p_{i-1} \]
\[ t = \frac{\|r_{i-1}\|^2}{p_{i-1}^T A p_{i-1}} \]
\[ \text{end do} \]

Finalization:
\[ \text{return } x_i \]

Figure 4.5: CG to solve for \( x \) in \( Ax = b \)

4.4 Sensitivity Analysis and rates of Convergence

According to Nocedal and Wright\(^8\) the CG method is more sensitive to round off errors than Gaussian elimination or matrix factorization methods, such for example, Cholesky, LU, or Singular Value Decomposition (SVD). However for large problems the CG algorithm is more efficient. The major cost of the algorithm is in the matrix vector multiplication. However for sparse matrices the cost of matrix vector multiplication can be reduced.

Convergence in \( n \) iterations:

It is well known that the (linear) CG method in exact arithmetic converges in at most \( n \) iterations where \( n \) is the dimension of the working space (that is, the rank of matrix

\(^8\)http://www.bioinfo.org.cn/wangchao/maa/Numerical_Optimization.pdf
4.4 Sensitivity Analysis and rates of Convergence

The key idea here is that any point of $\mathbb{R}^n$ can be written as a linear combination of base vectors, and since the vectors $p_k$ span $\mathbb{R}^n$, then think that the solution $x^*$ with respect to the point $x_0$ as a new coordinate origin, can be spanned in terms of the vectors $p_i$. That is,

$$x^* - x_0 = \sum_{i=0}^{n-1} \gamma_i p_i.$$

So, we need to find the coefficients $\gamma_i$ to know the total displacement from the starting point to the solution (or final point $x_n = x^*$).

Without even looking at the CG algorithm, we can find the coefficients by projecting the vector component into its A-orthogonal basis $\{p_i\}$. This is done be pre-multiplying by the matrix $A$ and then by a base vector $p_j$ in the previous equation and transposing. That is,

$$(x^* - x_0)^T A p_j = \sum_{i=0}^{n-1} \gamma_i p_i^T A p_j = \gamma_j p_j^T A p_j$$

where we used $A$–orthogonality. Now from (recall that $Ax_0 - b = r_0$)

$$x_0^T A p_j = (r_0^T + b^T) p_j = r_0^T p_j + b^T p_j$$

and, since $Ax^* = b$,

$$\gamma_j = \frac{(x^* - x_0)^T A p_j}{p_j^T A p_j} = \frac{b^T p_j - r_0^T p_j - b^T p_j}{p_j^T A p_j} = \frac{-r_0^T p_j}{p_j^T A p_j}.$$

We show that the last displacement $x_n - x_0$ in the CG iteration has exactly the same coefficients $\gamma_i$ and hence $x_n = x^*$ is obtained after $n$ iterations. Let us unroll the partial solutions of the CG method. That is, up to some iteration $k$, we have that

$$x_k = x_{k-1} - \alpha_{k-1} p_{k-1} = x_{k-2} - \alpha_{k-2} p_{k-2} - \alpha_{k-1} p_{k-1} = \cdots = x_0 - \sum_{i=0}^{k-1} \alpha_i p_i$$ (4.4.13)

where $\alpha_{k-1}$ is computed from equation 4.3.11, and the last computed $x_n$ is

$$x_n = x_0 - \sum_{i=0}^{n-1} \alpha_i p_i$$ (4.4.14)

That is,

$$x_n - x_0 = - \sum_{i=0}^{n-1} \alpha_i p_i$$
with
\[ \alpha_i = -\frac{p_i^T r_i}{p_i^T A p_i}. \]

from 4.3.6. Now
\[
\begin{align*}
p_i^T r_i &= r_i^T p_i \\
&= (r_{i-1} + \alpha_i A p_{i-1})^T p_i = r_{i-1}^T p_i + \alpha_i p_{i-1}^T A p_i \\
&= r_{i-2}^T p_i \\
&\vdots \\
&= r_0^T p_i
\end{align*}
\]
so
\[ \alpha_i = -\frac{r_0^T p_i}{p_i^T A p_i} = \gamma_i. \] (4.4.15)

This shows that the displacement vector \( x_n - x_0 \) is the same as the exact displacement vector \( x^* - x_0 \) (because they have the same coordinates on the basis \( \{ p_i \} \) ) and \( x^* = x_n \) is obtained after \( n \) iterations.

Having a number of iterations as \( n \) is not a guarantee of efficiency, although this is much better that a SD algorithm where the number of iterations could zig-zag many times the dimension of the space. In geophysical problems such as (FWI) the vector spaces have dimensions of the order of millions and each iteration is quite expensive, so in practice that number is reduced by choosing an initial solution as closed as possible to the true solution, by conditioning both, the model and the data, and by increasing the error tolerance. We discuss next some cases that reduce the number of iterations regardless the starting guess \( x_0 \), and any type of data/model conditioning.

**Convergence in less than \( n \) iterations:**

It could happen that some of the \( \gamma_i \) coefficients are zero and the process could finish before \( n \) iterations. For example, as in the steepest descent method, if \( x_0 \) is an eigenvector of \( A \) only one step is required and all coefficients , except \( \gamma_0 \), are zero. Another case is that of all eigenvalues equal. The quadratic form is a hypersphere and the method goes from any initial guess to the center of the sphere in the first step. The SD proof works here as well since the first direction is the negative of the gradient. Having repeated eigenvalues means faster convergence. For example, assume that three eigenvalues are equal. If we constrain the space to those three directions the piece of surface corresponding to the quadratic form is a sphere and the method shoots in one step a ray from any point of the sphere to the center, reducing by 2 the number of iterations. In general, it is expected that the number of iterations is equal to the number of distinct eigenvalues. This seems natural geometrically but we will build a formal proof. We will show this in steps.
(i) We first show that the point $x_k$ minimizes the function over the space spanned by the direction vectors $p_i, (i = 0 \cdots k - 1)$. We use the method of Lagrange multipliers.

(ii) Second, we show that the point $x_k$ is the closest point to the solution $x^*$ from all possible points in the space generated by the vectors $p_i, (i = 0 \cdots k - 1)$. Here we just use algebra.

(iii) We then use the previous two conclusions to proof that a number $m$ of different eigenvalues of the matrix $A$ implies an upper bound on the number of iterations of the CG method. An important new basis in terms of $A^t r$ terms is introduced which let us write the expansion of $x_k$ on the space of directions as a polynomial on $A$. This will further simplify the process.

First, let us recall from equation 4.4.13 that any point $x_k$ is equal to the initial point $x_0$ plus a linear combination of the conjugate directions $p_i$ up to the $k - 1$. We prove that, in fact, it is the best approximation to the quadratic function $f(x) = x^T A x/2 - b^T x + c$, defined in equation 3.1.1, constrained to the set

$$S = \{ x | x = x_0 + \text{span}\{p_0, p_1, \cdots, p_{k-1}\} \}.$$ 

That is, let us pick a point $x \in S$. Then

$$x = x_0 + \sum_{i=0}^{k-1} \gamma_i p_i \quad (4.4.16)$$

for some $\gamma_i$. We use Lagrange multipliers to solve the constraint minimization problem. The extended minimization problem is

$$\Lambda(x, \Gamma, \lambda) = f(x) + \lambda^T \left( x - x_0 - \sum_{i=0}^{k-1} \gamma_i p_i \right).$$

with $\Gamma = (\gamma_0, \gamma_1, \cdots, \gamma_{k-1})^T$. This extended system has $2n + k$ variables, $n$ for $x$ and $n$ for $\lambda$, and $k$ for $\Gamma$.

Lagrange multipliers method indicates that we should solve the equation

$$\nabla \Lambda(x, \Gamma, \lambda) = 0,$$

where the gradient is now computed with respect to $x$, $\Gamma$, and $\lambda$. The gradient with respect to $\lambda$ provides the constraints 4.4.16. That is, this is no new information. The gradient with respect to $x$ provides

$$\nabla f(x) + \lambda = 0$$

http://en.wikipedia.org/wiki/Lagrange_multiplier
but \( \nabla f(x) = Ax - b \). The gradient with respect to \( \gamma_j \) \((j < k)\) yields

\[
\lambda^T p_j = 0. \tag{4.4.17}
\]

So, we want to solve for \( x \) and \( \lambda \) such that

\[
\begin{align*}
Ax & = b - \lambda \\
x & = x_0 + \sum_{i=0}^{k-1} \gamma_i p_i \\
\lambda^T p_j & = 0.
\end{align*}
\]

We eliminate \( x \) and find

\[
Ax_0 + \sum_{i=0}^{k-1} \gamma_i A p_i = b - \lambda. \tag{4.4.18}
\]

From the first of these equations \( \lambda = b - Ax \). Plugging this into the third equation we find the new system

\[
\begin{align*}
Ax_0 + \sum_{i=0}^{k-1} \gamma_i A p_i & = b - \lambda \\
(Ax - b)^T p_i & = 0.
\end{align*}
\]

Now take equation 4.4.18 together with equation 4.4.17

\[
\left( Ax_0 + \sum_{i=0}^{k-1} \gamma_i A p_i - b \right)^T p_j = 0,
\]

and expanding

\[
x_0^T A p_j + \sum_{i=0}^{k-1} \gamma_i p_j^T A p_i - b^T p_j = 0.
\]

The term in the middle, due to the \( A \)-orthogonality of the \( p_j \) vectors reduces to \( \gamma_j p_j^T A p_j \) so we have

\[
x_0^T A p_j + \gamma_j p_j^T A p_j - b^T p_j = 0,
\]

from which

\[
\gamma_j = \frac{b^T p_j - x_0^T A p_j}{p_j^T A p_j} = \frac{(b^T - x_0^T A)p_j}{p_j^T A p_j}
\]
and since $r_0 = Ax_0 - b$, we find that $\gamma_j$ in this problem

$$\gamma_j = -\frac{r_0^T p_j}{p_j^T Ap_j}$$

is the $\alpha_j$ obtained from the CG method (see equation 4.4.15). So, the CG method could be seen as a consequence of a constrained Lagrange multipliers problem, provided the conjugate base vectors $p_j$. An we proved that while the SD method falls faster, it does not gets to the minimum faster, due to the location of the minimum away from the gradient direction. Faster (fall out) speed does not mean shortest time unless the path is the same.

Now we find a new basis which spans the space of directions which is simpler than the basis corresponding to the direction vectors $p_i$. Let us now consider again equation 4.4.13, that we write again

$$x_k = x_0 - \sum_{i=0}^{k-1} \gamma_i p_i.$$ 

This equation shows the evolution up to $x_k$ as a function of the initial point $x_0$ and the base conjugate directions $p_i$. We consider a different approach to the span of the partial basis $\{p_i\}_{i=0}^k$. From the CG algorithm 4.3.0 we find that

$$p_0 = r_0$$
$$p_1 = r_1 - \beta_0 p_0 = r_0 + \alpha_0 Ap_0 - \beta_0 r_0 = (1 - \beta_0)r_0 + \alpha_0 Ar_0$$
$$p_2 = r_2 - \beta_1 p_1 = r_1 + \alpha_1 Ap_1 - \beta_1 p_1 = r_0 + (\alpha_0 + \alpha_1(1 - \beta_0))Ar_0 + \alpha_1 \alpha_0 A^2 r_0.$$ 

Inductively we see that each vector $p_k$ can be written as a linear combination of the vectors $\{r_0, Ar_0, A^2 r_0, \ldots A^k r_0\}$. That is

$$\mathcal{K}(r_0; k) \equiv \text{span}\{r_0, Ar_0, \ldots A^k r_0\} = \text{span}\{p_0, p_1, \ldots p_k\}.$$ 

This space is called the Krylov subspace\(^{10}\) of degree $k$ for $r_0$, and the basis representation on the center expression has the advantage that it is a function of only two parameters $r_0$ and $A$, while the basis on the right is function of $k + 1$ parameters. These simplified basis will be useful to understand more how the conjugate gradient algorithm behaves on its descent efficiency.

Let us rewrite again equation 4.4.13 but this time in terms of the basis $\{A^i r_0\}$ for $i = 0 \cdots k$,

$$x_k = x_0 + \sum_{i=0}^{k-1} \gamma_i A^i r_0 = x_0 + P_{k-1}(A)r_0,$$ 

\(^{10}\)https://en.wikipedia.org/wiki/Krylov_subspace
where the polynomial $P_{k-1}$ is defined as
\[ P_{k-1}(t) = \sum_{i=0}^{k-1} \gamma_i t^i. \]

Since the span of the base vectors $\{p_i\}$ is the same as the span of the vectors $\{A^T r_0\}$ then the vector $x_k$ minimizes the quadratic function 3.1.1, but the coefficients $\gamma_i$ are different due to the change of basis. We show next that the point $x_k$ along the CG method is the closest point to the solution $x^*$ from all possible vectors of the form 4.4.19 with arbitrary real coefficient polynomials $P_{k-1}(t)$. Please note that we already know that $x_k$ provides the minimum of the function $f(x)$ from all points of the form 4.4.19. But it is not obvious from here that the point $x_k$ is the closest to $x^*$, although a picture of this fact looks obvious (look through any picture above. Even the SD pictures provide this evidence, however a picture is far from a proof.)

In other words, let us define the set
\[ S = \{ y | y = x_0 + P_{k-1}(A) r_0 \}, \]
we know that the CG iteration point $x_k$ (from using Lagrange Multipliers and constraint minimization) minimizes the range of the function $f(y)$ over all possible values of $y$ subject to the be in $S$. That is over all polynomials $P_{k-1}(t)$, of degree $k-1$, with arbitrary real coefficients $\gamma_i$. We show next that this point also minimizes the domain of the function $f(x)$.

That is, we want to show that if $x_k$ is the $k+1$ point along the CG iteration
\[ \| x_k - x^* \|_A^2 \leq \| y - x^* \|_A^2 \quad \forall y \in S, \tag{4.4.20} \]
where $\| x \|_A^2 = x^T A x$. This is good (and happens because the convexity of the function close to the minimum. The convexity is assured by the positive definiteness of $A$.) In other words, we are saying that we are not only minimizing the function $f(x)$ but we are getting closer and closer to the minimum point $x^*$. In this case the iteration is monotonic.

We use the fact that $x_k$ minimizes the function $f(x)$ in 3.1.1, in $S$. That is,
\[ f(x_k) \leq f(y) \quad \forall y \in S. \]

Expanding $f$ from 3.1.1 we find
\[
\begin{align*}
\frac{1}{2} x_k^T A x_k - b^T x_k + \epsilon & \leq \frac{1}{2} y^T A y - b^T y + \epsilon \\
\frac{1}{2} x_k^T A x_k - b^T x_k & \leq \frac{1}{2} y^T A y - b^T y \\
x_k^T A x_k - 2b^T x_k & \leq y^T A y - 2b^T y \quad \forall y \in S. \tag{4.4.21}
\end{align*}
\]
In addition we will need the following derivation (from $A^*x = b$),

$$\begin{align*}
    b^T(y - x_k) + x^*^T A x_k &= (A^*x)^T(y - x_k) + x^*^T A x_k \\
    &= x^*^T A y + x^*^T Ax_k - x^*^T Ax_k \\
    &= x^*^T A y,
\end{align*}$$

and so, multiplying this equation by $-2$,

$$-2b^T(y - x_k) - 2x^*^T Ax_k = -2x^*^T Ay \quad (4.4.22)$$

We have now the elements to show the minimization over the domain. Let us start with the left side of 4.4.20. That is

$$\|x_k - x^*\|^2_A = (x_k - x^*)^T A (x_k - x^*)$$

by definition

$$= x_k^T A x_k - x^*^T A x_k - x_k^T A x^* + x^*^T A x^*$$

expanding

$$= x_k^T A x_k - 2x^*^T A x_k + x^*^T A x^*$$

collecting equal terms

$$\leq y^T A y - 2b^T y + 2b^T x_k - 2x^*^T A x_k + x^*^T A x^*$$

factoring $b$

$$\leq y^T A y - 2x^*^T A y + x^*^T A x^*$$

from 4.4.22

$$\leq \|y - x^*\|^2_A$$

factoring.

Now, since

$$r_0 = Ax_0 - b = Ax_0 - A x^* = A(x_0 - x^*)$$

we have that

$$x_k - x^* = x_0 + P_{k-1}(A)r_0 - x^* = [I + P_{k-1}(A)] (x_0 - x^*). \quad (4.4.23)$$

At this point we have not used the eigenvalues/eigenvectors to study the conjugate gradient algorithm. Let us assume that $0 < \lambda_0 \leq \lambda_2 \cdots \leq \lambda_{n-1}$. The eigenvectors corresponding to these eigenvalues are $v_0, v_1, \ldots, v_{n-1}$. Since $A$ is symmetric and positive definite those eigenvectors are orthogonal. We can assume that they are normalized and so they are orthonormal. The total displacement vector can be written as

$$x_0 - x^* = \sum_{i=0}^{n-1} \xi_i v_i. \quad (4.4.24)$$

Now, if $\lambda_i$ is an eigenvalue of $A$, then $\lambda_i^k$ is an eigenvalue of $A^k$ and so we can say that

$$P_{k-1}(A)v_i = P_{k-1}(\lambda_i) v_i, \quad i = 0 \cdots n - 1$$
and from 4.4.23

\[ x_k - x^* = \sum_{i=0}^{n-1} [I + \lambda_i P_{k-1}^*(\lambda_i)]\xi_i v_i, \]

where now \( P_{k-1}^* \) is the exact polynomial that minimizes the distance between the point \( x_k \) and the exact solution \( x^* \). Then

\[
\| x_k - x^* \|^2_A = (x_k - x^*)^T A (x_k - x^*)
= (x_k - x^*)^T A \sum_{i=0}^{n-1} [I + \lambda_i P_{k-1}^*(\lambda_i)] \xi_i v_i
= (x_k - x^*)^T \sum_{i=0}^{n-1} [I + \lambda_i P_{k-1}^*(\lambda_i)] \xi_i \lambda_i v_i
= \left( \sum_{i=0}^{n-1} [I + \lambda_i P_{k-1}^*(\lambda_i)] \xi_i v_i \right)^T \sum_{i=0}^{n-1} [I + \lambda_i P_{k-1}^*(\lambda_i)] \xi_i \lambda_i v_i,
\]

but since \( v_i^T v_j = \delta_{ij} \) then we simplify this to

\[
\| x_k - x^* \|^2_A = \sum_{i=0}^{n-1} [1 + \lambda_i P_{k-1}^*(\lambda_i)]^2 \lambda_i \xi_i^2 = \min_{P_{k-1}} \sum_{i=0}^{n-1} [1 + \lambda_i P_{k-1}^*(\lambda_i)]^2 \lambda_i \xi_i^2 \tag{4.4.25}
\]

where now we let \( P_{k-1} \) be any arbitrary polynomial over the real numbers of degree \( k - 1 \).

This expression estimates the \( A \)-square-distance between the point \( x_k \) and the exact solution \( x^* \), as a minimization over all possible polynomials \( P_{k-1} \) evaluated in the eigenvalues of the matrix \( A \), the eigenvalues themselves and the coefficients of the total displacement vector \( x_0 - x^* \) (the \( \xi_i \) values). Equation 4.4.25 will be useful to find more concrete bounds on the error for the CG algorithm.

We are interested in finding bounds for the distance from the current point to the solution. Let us extract the polynomial expression from the sum by considering the largest value. That is we write

\[
\| x_k - x^* \|^2_A \leq \min_{P_{k-1}} \max_{0 \leq i \leq n-1} \sum_{j=0}^{n-1} \lambda_j \xi_j^2.
\]

Now, from equation 4.4.24

\[
\| x_0 - x^* \|^2 = (x_0 - x^*)^T A (x_0 - x^*) = \sum_{i=0}^{n-1} \xi_i v_i^T A \sum_{i=0}^{n-1} \xi_i v_i = \sum_{i=0}^{n-1} \xi_i v_i^T \sum_{i=0}^{n-1} \xi_i \lambda_i v_i
\]

and from the orthonormality of the \( v_i \) vectors, we see that

\[
\| x_0 - x^* \|^2_A = \sum_{i=0}^{n-1} \lambda_i \xi_i^2,
\]
and we find

\[ \| x_k - x^* \|_A^2 \leq \| x_0 - x^* \|_A^2 \min_{P_{k-1}} \max_{0 \leq i \leq n-1} [1 + \lambda_i P_{k-1}(\lambda_i)]^2 \] (4.4.26)

that is, relative to the total displacement \( \| x_0 - x^* \|_A \), the convergence rate is determined by the factor

\[ \min_{P_{k-1}} \max_{0 \leq i \leq n-1} [1 + \lambda_i P_{k-1}(\lambda_i)]^2. \] (4.4.27)

We completed the tools to prove the main result of this section. This is Theorem 5.4 in the book by Nocedal and Wright \(^\text{11}\) that we reproduce below.

If \( A \) has only \( r \) distinct eigenvalues, then the CG iteration will find the solution in at most \( r \) iterations. For this, the idea is to construct a polynomial \( Q_r(\lambda) \) as

\[ Q_{r-1}(\lambda) = \frac{(-1)^r}{\mu_0 \mu_2 \cdots \mu_{r-1}} (\lambda - \mu_0)(\lambda - \mu_1) \cdots (\lambda - \mu_{r-1}), \]

where \( \mu_0 < \mu_2 < \cdots < \mu_{r-1} \) are the distinct eigenvalues. Then each eigenvalue \( \lambda_i \) is a root of \( Q_{r-1} \) and \( Q_{r-1}(0) = 1 \). So \( Q_{r-1}(\lambda) - 1 \) is a polynomial of degree \( r \) with a root at \( \lambda = 0 \), so

\[ R_{r-1}(\lambda) = \frac{Q_{r-1}(\lambda) - 1}{\lambda} \]

is a polynomial of degree \( r - 1 \). So, from 4.4.27 with \( r - 1 = k - 1 \) we find

\[ 0 \leq \min_{P_{r-1}} \max_{0 \leq i \leq n-1} [1 + \lambda_i P_{r-1}(\lambda_i)]^2 \leq \max_{0 \leq i \leq n-1} [1 + \lambda_i R_{r-1}(\lambda_i)]^2 = \max_{0 \leq i \leq n-1} Q_{r-1}(\lambda_i) = 0, \]

so from equation 4.4.26 we find that \( x_k \) arrives at \( x^* \) in \( r \) steps (from 0 to \( r - 1 \)), corresponding to the \( r \) distinct eigenvalues of \( A \).

**Error Analysis for CG:**

As we did in the SD method, we define the error as

\[ \epsilon_k = x^* - x_k, \]

where \( x^* \) is the exact solution. We want to find bounds for this error.

We start by using equation 4.4.25 to prove an useful inequality. That is, we will prove that if \( I = [a, b] \), \( 0 < a < b \) is the interval containing all the eigenvalues of \( A \), then for all \( Q_k \), polynomials of order \( k \), such that \( Q_k(0) = 1 \) we have

\[ \frac{\| x^* - x_k \|_A}{\| x^* - x^0 \|_A} \leq \max_{0 \leq t \leq b} |Q_k(t)|. \]

Define $Q_k(t) = 1 + tP_{k-1}(t)$. We see that $Q_k$ is a $k$ order polynomial. Since $P_{k-1}$ is arbitrary of order $k$, then $Q_k - 1$ should have a root at $t = 0$, then the definition of $Q_k$ satisfies the requirements, and using equation 4.4.25 we find

$$
\|x^* - x_k\|_A^2 = \min_{Q_k(0)=1} \sum_{i=0}^{n-1} [Q_k(\lambda_i)]^2 \lambda_i \xi_i^2.
$$

(4.4.28)

From total displacement equation 4.4.24, we find

$$
\|x^* - x_0\|_A = (x^* - x_0)^T A (x^* - x_0) = \left( \sum_{i=0}^{n-1} \xi_i v_i^T \right) A \left( \sum_{i=0}^{n-1} \xi_i v_i \right) = \sum_{i=0}^{n-1} \xi_i^2 \lambda_i.
$$

From this, and equation 4.4.28 we find

$$
\|x^* - x_k\|_A^2 = \min_{Q_k(0)=1} \sum_{i=0}^{n-1} [Q_k(\lambda_i)]^2 \lambda_i \xi_i^2 \leq \max_{a \leq t \leq b} |Q_k(t)|^2 \sum_{i=0}^{n-1} \lambda_i \xi_i^2,
$$

or

$$
\|x^* - x_k\|_A^2 \leq \max_{a \leq t \leq b} |Q_k(t)|^2 \|x^* - x_0\|_A.
$$

which means

$$
\frac{\|x^* - x_k\|_A}{\|x^* - x_0\|_A} \leq \max_{a \leq t \leq b} |Q_k(t)|.
$$

(4.4.29)

As shown with the polynomials $P_{k-1}$, we can use special polynomials $Q_k(t)$ to find bounds on the CG errors. This is the case when $Q_k(t)$ is chosen as a special Chebyshev Polynomial 12. Next we present the minimum information required to find the bounds of the CG relative error.

Some little notes on Chebyshev Polynomials: A Chebyshev polynomial is defined by the recursive formula

$$
T_{n+1}(t) = 2t T_n(t) - T_{n-1}(t), \quad n \geq 1.
$$

(4.4.30)

starting with $T_0(t) = 1$ and $T_1(t) = t$. We see that then $T_2(t) = 2t T_1(t) - T_0(t) = 2t^2 - 1$, and if $T_n(t)$ and $T_{n-1}(t)$ are polynomials of order $n$ and $n-1$ respectively, then $T_{n+1}(t)$ defined in this way is a polynomial of order $n + 1$.

There are closed form expressions for Chebyshev polynomials. For example

(i) 

$$
T_n(t) = \cos(n \arccos t) \quad \text{for} \quad t \in [-1, 1]
$$

(4.4.31)

and

12http://en.wikipedia.org/wiki/Chebyshev_polynomials
(ii) \( T_n(t) = \frac{1}{2} [(t + \sqrt{t^2 - 1})^n + (t + \sqrt{t^2 - 1})^{-n}] \) for \(|t| \geq 1\).

The first of this properties is easy to check by inspection. That is, we assume that a polynomial satisfies the definition \( P_n(t) = \cos(n \arccos t) \) and show that the recursion 4.4.30 is satisfied. Call \( \phi = \arccos t \), so \( P_n(t) = \cos n\phi \) with \( t = \cos \phi \). So

\[
P_{n+1}(t) + P_{n-1}(t) = \cos(n+1)\phi + \cos(n-1)\phi = 2 \cos \phi \cos n\phi = 2tP_n(t),
\]

so the recursion is satisfied. In addition, \( P_0 = \cos 0 = 1 \) and \( P_1 = \cos \phi = t \).

The second property is shown by converting the recursive equation into a polynomial equation. That is, we write the recursive equation 4.4.30 as

\[
x_{n+1} - 2tx_n + x_{n-1} = 0 \quad \text{for} \quad n \geq 1, \quad \text{with} \quad x_0 = 1, x_1 = t \quad (4.4.32)
\]

where \( x_n(t) = T_n(t) \) and \(|t| \geq 1\).

The trick shown next is based on the Z-transform. \(^{13}\) We replace \( x_n \) by \( z^n \) and write

\[
z^{n+1} - 2tz^n + z^{n-1} = 0,
\]

from which, simplifying

\[
z^2 - 2tz + 1 = 0.
\]

We find the solutions for \( z \) as

\[
z_{1,2} = t \pm \sqrt{t^2 - 1},
\]

Any general solution of the homogeneous system is a linear combination of these two solutions. That is, the general solution of the recursive equation 4.4.32 is given by \( x_n(t) = c_1z_1^n(t) + c_2z_2^n(t) \). To find the values of \( c_1 \) and \( c_2 \) we use the knowledge of

\[
x_0(t) = c_1(t + \sqrt{t^2 - 1})^0 + c_2(t - \sqrt{t^2 - 1})^0 = 1
\]

\[
x_1(t) = c_1(t + \sqrt{t^2 - 1}) + c_2(t - \sqrt{t^2 - 1}) = t.
\]

From the first equation we find \( c_1 + c_2 = 1 \), from the second equation we require that the coefficient of \( \sqrt{t^2 - 1} \) disappears, that is, \( c_1 - c_2 = 0 \). So \( c_1 = c_2 = 1/2 \) and then

\[
x_n(t) = \frac{1}{2}(t + \sqrt{t^2 - 1})^n + (t - \sqrt{t^2 - 1})^n
\]

Now since

\[
(t + \sqrt{t^2 - 1})(t - \sqrt{t^2 - 1}) = t^2 - t^2 + 1 = 1,
\]

\(^{13}\)http://en.wikipedia.org/wiki/Z-transform. In the same way that differential equations can be converted to algebraic equations using the Fourier or Laplace transforms, difference equations can be converted to algebraic equations using the Z-transform.
then

\[
(t + \sqrt{t^2 - 1})^{-1} = t - \sqrt{t^2 - 1}
\]

and so

\[
T_n(t) = \frac{1}{2} \left[ (t + \sqrt{t^2 - 1})^n + (t + \sqrt{t^2 - 1})^{-n} \right]
\]

for \(|t| \geq 1\).

as required.

We are now ready to construct the polynomial \(Q_k(t)\) that will provide bounds for the CG algorithm error.

Let \(m = \min \lambda_i\) and \(M = \max \lambda_i\), \(i = 0, \ldots, n - 1\). We define

\[
Q_k(t) = \frac{1}{T_k\left(\frac{M + m}{M - m}\right)} T_k\left(\frac{M + m - 2t}{M - m}\right).
\]

We verify that \(Q_k\) satisfies the conditions imposed by inequality 4.4.29.

First, \(Q_k\) is a scaled Chebyshev polynomial of order \(k\). Now, \(Q_k(0) = 1\). Now, since the domain of \(Q_k(t)\) is such that \(m \leq t \leq M\), we have that the linear function of \(t\), satisfies

\[
-1 \leq \frac{M + m - 2t}{M - m} \leq 1
\]

and from equation 4.4.31

\[
\max_{-1 \leq t \leq 1} |T_k(t)| = 1,
\]

and so, since by re-defining \(t\) as

\[
t = \frac{M + m}{M - m} = \frac{1 + \kappa}{1 - \kappa}
\]

we find that

\[
t + \sqrt{t^2 - 1} = \frac{1 + \kappa}{1 - \kappa} + \sqrt{\left(\frac{1 + \kappa}{1 - \kappa}\right)^2 - 1} = \frac{1 + \kappa}{1 - \kappa} + \frac{2\sqrt{\kappa}}{1 - \kappa} = \frac{1 + 2\sqrt{\kappa} + \kappa}{1 - \kappa},
\]

which we simplify as

\[
t + \sqrt{t^2 - 1} = \frac{(1 + \sqrt{\kappa})^2}{(1 - \sqrt{\kappa})(1 + \sqrt{\kappa})},
\]

so

\[
\left| T_k\left(\frac{M + m}{M - m}\right) \right| \geq \frac{1}{2} \left( \frac{1 + \sqrt{\kappa}}{1 - \sqrt{\kappa}} \right)^k + \left( \frac{1 + \sqrt{\kappa}}{(1 - \sqrt{\kappa})} \right)^{-k} \geq \frac{1}{2} \left( \frac{1 + \sqrt{\kappa}}{1 - \sqrt{\kappa}} \right)^k
\]
and from equation 4.4.29 we find that the bounds of the relative error of the CG method, are given by

\[
\frac{\| x^* - x_k \|_A}{\| x^* - x^0 \|_A} \leq 2 \left( \frac{1 - \sqrt{\kappa}}{1 + \sqrt{\kappa}} \right)^k
\]

We now use the inequality A.7 to find an upper bound for the right hand side. That is, we know that

\[
\frac{x - 1}{x + 1} < e^{-2/x},
\]

then we can write

\[
\frac{\| \epsilon_k \|_A}{\| \epsilon_0 \|_A} = \frac{\| x^* - x_k \|_A}{\| x^* - x^0 \|_A} \leq 2 \left( \frac{1 - \sqrt{\kappa}}{1 + \sqrt{\kappa}} \right)^k < 2e^{-2k/\sqrt{\kappa}}.
\]

This equation is a quantitative estimation of the error bounds in the CG method. Compare equation 4.4.33 with its corresponding SD equation 3.3.14. It is clear that the error goes smaller with CG than with SD. In both cases we see that if the condition number \( \kappa \) is closed to 1 the methods should converge quickly. This motivates the next section which has as objective bring the eigenvalues together to decrease the condition number \( \kappa \) close to 1.

### 4.5 Preconditioning

Motivated by equations 4.4.33 and 3.3.14 for the error bounds of the CG method and the SD method respectively, we claim that getting \( \kappa \) close to 1 is a way to speed convergence. Getting \( \kappa \) close to 1 is the same as squeeze the spectrum of eigenvalues to lie in a narrow circular band. One way to achieve this is by a change of coordinates. For example, a diagonal matrix has all the eigenvalues in the diagonal. They can be spread along a large interval. A pre-multiplication of \( A \) with another diagonal matrix that squeezes the directions of large eigenvalues while stretching or preserving those of small eigenvalues has the net effect of leaving the product \( PA \) as a matrix with a smaller condition number. Here is an example of the new coordinates \( \hat{x} \) in terms of the old coordinates \( x \).

\[
\hat{x} = Px,
\]

where \( P \) is the pre-conditioning matrix. We now want to solve the system

\[
PAx = Pb.
\]

Even if \( P \) is symmetric (and we assume that \( A \) is symmetric) the product \( PA \) is necessarily symmetric. Let us assume that \( P \) is symmetric positive definite. The
Cholesky Decomposition theorem\(^\text{14}\) guarantees that we can take a “square root” of the matrix \(B\). That is, there is matrix \(C\) such that
\[
P = C^T C,
\]
so
\[
PAx = Pb \iff C^T (CAC^T)C^{-T} x = C^T Cb \iff (CAC^T) \hat{x} = Cb \quad \& \quad x = C^T \hat{x}
\]
We found that
\[
CAC^T \hat{x} = Cb \quad \& \quad x = C^T \hat{x}.
\]
(4.5.34)
So even if \(PA\) is not positive definite symmetric, \(CAC^T\) is. In addition, \(CAC^T\) is similar to \(PA\), since
\[
C^T (CAC^T)C^{-T} = PA
\]
thus both \(CAC^T\) and \(PA\) share the same eigenvalues, and the CG method applied to this equation is such that the convergence will be determined by the eigenvalues of \(PA\). We apply the CG method to equation 4.5.34. Denoting the search direction by \(p_k\), and the residual by \(r_k = CAC^T \hat{x}_k - Cb\) we find from equations 4.3.6, 4.3.11, 4.3.8, 4.3.12, and 4.3.7, the CG equations in the new coordinate system
\[
\begin{align*}
\alpha_k &= -\frac{r_k^T r_k}{p_k^T CAC^T p_k} \\
\hat{x}_{k+1} &= \hat{x}_k + \alpha_k p_k \\
r_{k+1} &= r_k + \alpha_k CAC^T p_k \\
\beta_k &= -\frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} \\
p_{k+1} &= r_{k+1} - \beta_k p_k.
\end{align*}
\]
We want to return to the original coordinates \(x\). We do the following substitutions \(x_k = C^T \hat{x}\), \(q_k = C^T p_k\), \(s_k = C^T r_k\), and \(R^k = C^{-1} r_k\), into the previous equation to find:
\[
\begin{align*}
x_{k+1} &= x_k - \alpha_k q_k \quad \alpha_k = -\frac{s_k^T R_k}{q_k^T A q_k} \\
R_{k+1} &= R_k - \alpha_k A q_k \\
s_{k+1} &= s_k - \alpha_k PA q_k \\
q_{k+1} &= s_k - \beta_k q_k \quad \beta_k = -\frac{s_{k+1}^T R_{k+1}}{s_k^T R_k}
\end{align*}
\]
The interesting thing here is that we got rid of \(C\). We only need the matrix \(P\). The preconditioned CG algorithm is listed next.

Next we introduce the non–linear CG.

\(^{14}\)http://en.wikipedia.org/wiki/Cholesky_decomposition
4.5 Preconditioning

Initialization:

\[ i = 0 \]
\[ \text{pick } x_k \]
\[ r_k = Ax_k - b \]
\[ p_k = Pr_k \]
\[ s_k = p_k \]
\[ t = -\frac{s_k^T r_k}{p_k^T A p_k} \]
\[ \varepsilon_k = 2\epsilon \]

Loop:

\[ \text{while } (\varepsilon_k > \epsilon \text{ and } i < M) \text{ do} \]
\[ k = k + 1 \]
\[ x_k = x_{k-1} - t p_{k-1} \]
\[ r_k = r_{k-1} - t A p_{k-1} \]
\[ s_k = s_{k-1} - t P A p_{k-1} \]
\[ \beta = -\frac{s_k^T r_k}{s_{k-1}^T r_{k-1}} \]
\[ \varepsilon_k = \frac{\|r_k\|}{\|r_{k-1}\|} \]
\[ p_k = s_k - \beta p_{k-1} \]
\[ t = -\frac{s_k^T r_k}{p_k^T A p_k} \]
\[ \text{end do} \]

Finalization:

\[ \text{return } x_k \]

Figure 4.6: Algorithm to solve for the preconditioned CG problem \( x \) in \( PAx = P b \)
Chapter 5

Non–linear CG

Recall in the SD method (see 3), that we assumed that the function $f$ (equation 3.1.1), is quadratic. If the function is not quadratic, we still can think about descent methods. The idea that the gradient pushes directions toward an ascent method is local and based on multi–dimensional analysis (or calculus). The negative of the gradient move down toward a minimum value. Such minimum exists as long as the Hessian (the second derivative matrix) exists and is non–singular. The gradient of the function $f$ in equation 3.1.1 is given by equation 3.1.2 that we rewrite here.

$$\nabla f(x) = Ax - b = r = g$$

In this way it is natural to think that we can rewrite the CG algorithm 4.3.0 by replacing any occurrence of $Ax - b$ by $\nabla f(x)$. The non–linear CG algorithm 5.0.0 follows:

Algorithm 5.0.0

- Pick an initial guess $x_0$.
- Compute the first residual $r_0 = \nabla f(x_0)$. This is also named $p_0$. With this residual and $\alpha_1$ (to be defined later), find the new $x_1$ and the new residual, that is: While the relative residual $\|r_i\|/\|r_{i-1}\|$ is larger than some threshold value

$$x_{i+1} = x_i - \alpha_i p_i$$
$$r_{i+1} = \nabla f(x_1)$$

Use $p_i$ and $r_{i+1}$ to find the new conjugate direction $p_{i+1}$.

$$p_{i+1} = r_{i+1} - \beta_i p_i.$$  (5.0.1)
The big question here is what are $\alpha_i$ and $\beta_i$. The value of $\alpha_i$ comes from the SD idea, and it is a value that minimizes the objective function along a line search. That is, $\alpha_i$ is the value of $t$ such that

$$\varphi(t) = f[x_i - tp],$$

with $p$ as the search direction, is minimum. Since we do not know the expression for the surface $f$, which is a quadratic surface for the linear CG method, we have no way to know an analytic formula for $t$ (which is called usually in the literature $\alpha_k$, along the $k$ iteration). The $x_{i+1} = x_i - tp$ should occur at a stationary point. That is, we require

$$\varphi'(t) = \nabla f(x_{i+1})^T p = 0.$$ \hspace{1cm} (5.0.2)

In this chapter we present the most known optimization techniques for non–linear CG. Historically, the Fletcher–Reeves formula (which is a natural extension of formula 4.3.12) was the first of a series of formulas for $\beta_i$. We review some concepts which should be common to all methods explained below, before focusing on the best known formulas for $\beta_i$.

We are trying to find a minimum of a scalar function $f(x)$ around some local valley. We observe the following steps which should be followed:

(i) **Initial guess** $x_0$, initial search direction $p_0 = \nabla f(x_0)$. We want to provide a solution as closed as possible to the exact solution to speed convergence. If we totally ignore how to pick a close solution we could start at $x_0 = 0$. In the case of FWI a good tomography job through the seismic data could provide an acceptable initial guess.

(ii) **Line search along the current direction $p_i$**. The line search algorithm should acknowledge the Wolfe’s conditions 6.3. Section 6.3 contains a detailed discussion of line search algorithms. We know that a line search which locally minimizes the path along the search direction $p$ should satisfy equation 5.0.2. That is, the line search should be such that the new gradient is orthogonal to the search direction $p$.

(iii) **Formula for $\beta_i$**, which is needed to update the search direction. The Fletcher–Reeves formula is a natural continuation from the linear CG formula for $\beta_i$ 4.3.12. The main objective of this chapter is to discuss this and other formulae commonly used in the numeric analysis community.

(iv) Formula to update search direction and update of the next point. Once the line search scalar as well as the $\beta_i$ scalar are found the update to the new point $x_{i+1}$ and new direction $p_{i+1}$ is a trivial sum with the proper weighted values ($\alpha_i$ and $\beta_i$).
5.1 A general formula for $\beta_i$

(v) Evaluation of rate of convergence. Error analysis. We need to know stopping criteria. As previously done with SD and linear CG we use the ratio of the norms of the current and previous residuals as well as the maximum number of iterations.

(vi) If the cycle finishes due to the low rate convergence, it can get re-started by picking the gradient (SD) direction on a new sequence of iterations.

In all cases, the formulas for $\beta_i$ should provide the same solution for an exact quadratic surface. Also, CG methods have the advantage over Newton and quasi-Newton method that they have low memory requirements (although the L-BFGS seems to be the “best of the two wolds”.) In what follows we will focus only on the finding of $\beta_i$. All other steps are covered under the linear CG algorithm above.

We first derive a general formula for $\beta_i$ which fits each of the models explained below.

5.1 A general formula for $\beta_i$

When referring to linear quadratic models we use $A$ for the Hessian matrix. We use $H$ when the model is non-quadratic. As done in the Newton optimization methods let us define

$$g_k = \nabla f(x_k).$$

The initial direction (steepest descent) is given by $p_0 = g_0$. The first computed point is $x_1 = x_0 - \alpha_0 p_0$ where $\alpha_0$ is the scalar that minimizes the line search function $f(x_0 - \alpha p_0)$. Let us consider equation 4.2.4 where we find $t$ (which is our $\beta_i$) such that

$$p_1 = r_1 - tp_0 = g_1 - \frac{p_0^T H g_1}{p_0^T H p_0} p_0. \quad (5.1.3)$$

Here $H$ is the Hessian matrix of $f(x)$ and $r_1$ is our gradient $f(x_1)$. That is, here

$$h_{ij} = \frac{\partial f(x)}{\partial x_i \partial x_j}, \quad g_i = r_i = \nabla f(x_i).$$

Note that this equation is exact if the model is quadratic and there $H = A$. What we are saying here is that the conjugate directions $p_i$ are conjugate with respect to a local quadratic model with Hessian matrix $H$.

We know that a CG iteration estimates

$$x_k = x_{k-1} - \alpha_{k-1} p_{k-1} \Rightarrow \Delta x_k = -\alpha_{k-1} p_{k-1}$$

If the surface is quadratic then by taking the gradient in two different points we find

$$\nabla f(x_{k-1}) = H x_{k-1} - b$$

$$\nabla f(x_k) = H x_k - b$$
and subtract, to find

\[ \nabla f(x_k) - \nabla f(x_{k-1}) = H(\Delta x_k) = -\alpha_{k-1} H p_{k-1} \]

that is,

\[ g_k - g_{k-1} = -\alpha_{k-1} H p_{k-1} \]

and so we can write equation 5.1.3, for \( k \), as

\[
p_k = g_k - \frac{p_{k-1}^T H g_k}{p_{k-1}^T H p_{k-1}} p_{k-1} = g_k - \frac{g_k^T H p_{k-1}}{p_{k-1}^T H p_{k-1}} p_{k-1} = g_k - \frac{g_k^T (g_k - g_{k-1})}{p_{k-1}^T (g_k - g_{k-1})} p_{k-1}. \]

In other words, we are saying that

\[
p_k = g_k - \beta_k p_{k-1}, \quad \beta_k = \frac{g_k^T (g_k - g_{k-1})}{p_{k-1}^T (g_k - g_{k-1})}, \quad (5.1.4)\]

This formula generates the main equations known for non–linear conjugate gradient as we show below.

### 5.2 The Fletcher–Reeves formula

The first published work in non–linear CG is by Fletcher and Reeves \(^1\). The Fletcher–Reeves formula for \( \beta_k \) is the same formula for linear CG. This is given by equation 4.3.12 which we derive again from formula 5.1.4.

The numerator of \( \beta_k \) is

\[ g_k^T (g_k - g_{k-1}). \]

In the regular CG method (quadratic surface) the gradient of the previous iteration is orthogonal to the gradient of the current iteration (see equation 4.3.10). That is,

\[ g_k^T g_{k-1}^T = 0, \]

so the numerator turns out to be

\[ g_k^T g_k. \]

Now, for the denominator. This is

\[
p_{k-1}^T (g_k - g_{k-1}) = p_{k-1}^T g_k - p_{k-1}^T g_{k-1} \]

---

\(^1\)http://www.yaroslavvb.com/papers/fletcher-function.pdf
5.3 The Polak–Ribière formula

From equation 4.3.9 we observe that \( p_{k-1}^T g_k = g_k^T p_{k-1} = 0 \), and \( p_{k-1}^T g_{k-1} = \|g_{k-1}\|^2 \). Then

\[
\beta_k = -\frac{\|g_k\|^2}{\|g_{k-1}\|^2}.
\]

This is the Fletcher–Reeves equation for \( \beta_k \) in the non-linear CG method, sometimes written as

\[
\beta_k = -\frac{\nabla f(x_k)^T \nabla f(x_k)}{\nabla f(x_{k-1})^T \nabla f(x_{k-1})}.
\]

Again, the minus “−” sign here is due to the fact that we consider the difference on the update of \( p \) instead of the addition. Compare this equation with equation 4.3.12.

5.3 The Polak–Ribière formula

The Polak–Ribière formula considers the case where the surface is no necessarily quadratic and the gradient of the previous iteration does not have to be orthogonal to the gradient of the previous iteration. That is

\[
g_k^T g_{k-1} \neq 0.
\]

However still, the search direction \( p_{k-1} \) is orthogonal to the gradient \( g_k \). That is \( g_k^T p_{k-1} = 0 \). Then equation 5.1.4 for \( \beta_k \) becomes

\[
\beta_k = -\frac{g_k^T (g_k - g_{k-1})}{p_{k-1}^T (g_k - g_{k-1})} = -\frac{g_k^T (g_k - g_{k-1})}{\|p_{k-1}^T g_{k-1}\|^2},
\]

which in gradient notation is

\[
\beta_k = -\frac{\nabla f(x_k)^T (\nabla f(x_k) - \nabla f(x_{k-1}))}{\nabla f(x_{k-1})^T \nabla f(x_{k-1})}.
\]

This is the Polak–Ribière formula.

5.4 The Hestenes-Stiefel formula

This formula is precisely equation 5.1.4 where usually is written with a minus “−” sign but here is plus “+” because we subtract instead of add the direction vector.

It is interesting to observe that the Hestenes-Stiefel formula could be seen as a quasi-Newton algorithm and more specifically as a L-BFGS algorithm with \( m = 1 \). We prove this just after equation 6.4.44 which belongs to the quasi-Newton methods section.
5.5 The Dai-Yuah formula

The Dai-Yuah formula is obtained from formula 5.1.4 by assuming that $g_k$ is orthogonal to $g_{k-1}$ and so the 5.1.4 formula becomes

$$
\beta_k = \frac{g_k^T g_k}{p_k^T (g_k - g_{k-1})}
$$
Chapter 6

Newton Optimization Methods

The Newton optimization methods are based on approximations built from the Taylor series expansions. We start with the simplest one dimensional case. By doing Taylor series expansions we know that

\[
  f(x_k + \Delta x) = f(x) \approx f(x_k) + f'(x_k)\Delta x + \frac{1}{2}f''(x_k)\Delta x^2
\]

(6.0.1)

where \( \Delta x = x - x_k \). If \( f \) is linear then \( f'' = 0 \) and

\[
  f(x_k + \Delta x) = f(x_k) + f'(x_k)\Delta x,
\]

This is the equation of the line going through the points \( x_k \) and \( x = x_k + \Delta x \) with slope

\[
  f'(x_k) = \frac{f(x_k + \Delta x) - f(x_k)}{\Delta x}.
\]

The interesting case is when \( f \) is non-linear. Let us assume, for the moment, that \( f''(x_k) \neq 0 \) (either \( f \) is non-linear or \( x_k \) is not an inflection point).

Knowing that we are standing on a given point \( x_k \), we would like to find an increment \( \Delta x \) such that the error

\[
  f(x) - f(x_k)
\]

with \( \Delta x \neq 0 \), and \( x = x_k + \Delta x \), is minimum.

Then considered the quadratic error \( E \) function (in \( \Delta x \) ) equation

\[
  E(\Delta x) = f'(x_k)\Delta x + \frac{1}{2}f''(x_k)\Delta x^2
\]

obtained from the truncation of the Taylor approximation up to the second order. Then

\[
  E'(\Delta x) = 0,
\]
implies
\[ f'(x_k) + f''(x_k)\Delta x = 0, \]
from which we obtain the increment
\[ \Delta x = -\frac{f'(x_k)}{f''(x_k)}. \]
Provided that \( f''(x_k) \neq 0 \).

It is common to call \( \Delta x = x_{k+1} - x_k \) as if \( x_{k+1} \) would be the target point \( x \). We then can write the recursion formula
\[ x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}. \] (6.0.2)

Assume the quadratic function \( f(x) = ax^2 + bx + c \), with \( a > 0 \). The minimum of this function is reached at \( x_m \) such that \( f'(x_m) = 0 \). That is
\[ 2ax_m + b = 0, \]
\[ x_m = -\frac{b}{2a}. \]
Now, pick any initial guess \( x_k \). Then, since \( f''(x_k) = 2a \) and \( f'(x_k) = 2ax_k + b \),
\[ b = f'(x_k) - 2ax_k, \]
and so
\[ x_m = -\frac{b}{2a} = -\frac{f'(x_k) - 2ax_k}{f''(x_k)} = x_k - \frac{f'(x_k)}{f''(x_k)}. \]
That is, the minimization problem is solved with only one iteration. This is expected since the method is based on a second order Taylor series approximation.

If the function is higher order it will likely take more than just one iteration to get to the minimum.

Let us now compare formula 6.0.2 with the Newton-Raphson iteration method \(^1\)
\[ x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \] (6.0.3)
Newton-Raphson iterative method is used to find roots of functions, that is, zeroes. But all local minima/maxima points are laying on places where the derivative is zero.

\(^1\)http://en.wikipedia.org/wiki/Newton%27s_method
So, we can say that the Newton optimization method is the same Newton-Raphson method applied to the function $f'(x)$ instead of to the function $f(x)$.

Let us illustrate this with a simple example.

Assume

$$ f(x) = x + \frac{e^x}{500} - 2. $$

Since $f(0) = -1.998 < 0$ and $f(10) = 52.0529315896134 > 0$, there is a root in the interval $[0, 10]$. $f'(x) = 1 + e^x/500$. Let us naively start at $x_0 = 10$. So we have

First iteration:

$$ f(x_0) = 52.529315896134 \quad , \quad f'(x_0) = 45.0529315896134 $$

$$ x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 10 - \frac{52.529315896134}{45.0529315896134} = 8.84462720286966 $$

Second iteration:

$$ f(x_1) = 20.7186621582397 \quad , \quad f'(x_1) = 14.87403495537 $$

$$ x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 8.84462720286966 - \frac{20.7186621582397}{14.87403495537} = 7.45168559553805 $$

Third iteration:

$$ f(x_2) = 8.89721476285518 \quad , \quad f'(x_2) = 4.44552916731713 $$

$$ x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} = 7.45168559553805 - \frac{8.89721476285518}{4.44552916731713} = 5.45030073718054 $$

Fourth iteration:

$$ f(x_3) = 3.91595708811841 \quad , \quad f'(x_3) = 1.46565635093787 $$

$$ x_4 = x_3 - \frac{f(x_3)}{f'(x_3)} = 5.45030073718054 - \frac{3.91595708811841}{1.46565635093787} = 2.77848951375658 $$

Fifth iteration:

$$ f(x_4) = 0.81067889729198 \quad , \quad f'(x_4) = 1.0321893835354 $$

$$ x_5 = x_4 - \frac{f(x_4)}{f'(x_4)} = \frac{3.91595708811841}{1.46565635093787} = 1.993092075822 $$

We now that $f(x_5) = 0.00776845373223534$ which is very close to 0. So we can stop at $x_5 = 1.993092075822$ as the root of the function $f(x)$ in the interval $[0, 10]$. 

Figure 6.1: The Newton-Raphson method on $y = x + e^x/500.0 - 2.0$. Here a root close to $x_5$ is found. The Newton descent method is sketched in the upper curve, where the abscissas are exactly the same. While the lower curve finds a zero, the upper curve (which is an integral of the lower curve) finds a local minimum, hence the name Newton Optimization.
The lower curve (blue) in Figure 6.1 shows all the graphic steps corresponding to the Newton-Raphson method for the example developed here. The tangent lines are solid red lines, and the abcisas $x_0$ through $x_5$ are the brown circles. In contrast to this methods that use tangent lines to find the roots, there is another method called the **Secant Method**\(^2\) which use secants instead of tangents. The idea is that while the derivative of a smooth curve in a given point represents the tangent of the curve at that point, the central difference approximation

$$\frac{f(x_{k+1}) - f(x_{n-1})}{x_{k+1} - x_{n-1}}$$

represents a secant between the points $[x_{n-1}, f(x_{n-1})]$ and $[x_{k+1}, f(x_{k+1})]$. The secant method algorithm is hence given by the recursion

$$x_{k+1} = x_k - f(x_k)/(f(x_{k+1} - f_{x_{n-1}})/(x_{k+1} - x_{n-1}) = x_k - \frac{f(x_k)(x_{k+1} - x_{n-1})}{f(x_{k+1}) - f(x_{n-1})} \quad (6.0.4)$$

Figure 6.2 illustrates one step of the secant method. The secant line between the abscissas $x_1$ and $x_2$ cuts through the curve and finds a zero at $x_3$ which actually is closer to the root. We might even think that the secant algorithm is faster because it cuts through, however it might jump to the other side of the local minimum. We do not want to quantify the speed versus accuracy of the secant algorithm. Papakonstantinou's Ph. D. thesis\(^3\) explores the history of the secant method which dates to more than 3000 years on Babylonian clay tables and Egyptian Rhind Papyrus.

The Newton descent method is obtained on the integral of this curve. That is

$$g(x) = \frac{x^2}{2} + \frac{e^x}{500} - 2x + 10,$$

where we set the integration constant to 10 for convenient display in Figure 6.1. Note that starting at $x_0 = 10$ and following the same steps, the abscissas are the same and while finding a zero in the Newton-Raphson algorithm we are simultaneously finding the local minimum in the integrated curve. We should warn the reader that for higher dimensions, the problem of finding roots and the problem optimization problems of finding local minima/maxima differ. The reason for this, is that in optimization the matrix which represents the first derivative function is in general symmetric (since it is of the type $A^T A$), while the Jacobian matrix to find a zero of an arbitrary multi-dimensional function do not have to be symmetric.

Let us, for comparison, apply the SD algorithm 6.0.0, with replacement $r = \nabla f(x) = Ax - b$, $A = f''(x)$. Then the SD for functions of one variable becomes

\(^2\)http://en.wikipedia.org/wiki/Secant_method

\(^3\)http://www.caam.rice.edu/ttech_reports/2009/TR09-43.pdf
$f(x) = x + e^x/500.0 - 2.0$

$g(x) = x^2/2 + e^x/500 - 2x + 10$

Figure 6.2: One step of the secant method corresponding to the experiment in Figure 6.1. The abssisa $x_3$ is computed from the previous found $x_2$ and $x_1$. 
Algorithm 6.0.0

- Pick an initial $x_0$, and compute $r_0 = f'(x_0)$.
- If $r_0 \neq 0$ and $f''(x_0) \neq 0$, find $t = 1/f''(x_0)$.
- Update $x_1 = x_0 - tr_0$.
- Go back to the first step by replacing $x_0$ with $x_1$ and follow the recursion inductively, until some threshold has been crossed. For example until $|r_0| < \epsilon$.

If you observe this algorithm, it is saying that the update $x_{k+1}$ is done from the recursion formula

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}.$$

But, this is precisely the formula represented in equation 6.0.2. So, for functions of one variable the SD method and the Newton optimization method are equivalent.

What about several dimensions?

### 6.1 Several Dimensions

The problem of finding the root of system of equations

$$g(x) = \nabla f(x) = 0,$$

is equivalent to that of finding the local minimum of a local convex function which is twice differentiable. The tangent hyper–plane is given by the Jacobian matrix

$$(J_{ij}) = \left( \frac{\partial g_i(x)}{\partial x_j} \right).$$

This Jacobian matrix is the same the Hessian of $f(x)$. A finite difference approximation of this equations leads to the secant method.

The idea from one dimension can be word–to-word copied to several dimensions. That is, we can start with a Taylor series expansion in several dimensions. We use boldface fonts for vectors translate the second order approximation 6.0.1 into its vector from

$$f(x_k + \Delta x) = f(x) \approx f(x_k) + (\Delta x)^T \nabla f(x_k) + \frac{1}{2} \Delta x^T H(x_k) \Delta x. \quad (6.1.5)$$

where $H$ is the Hessian matrix of second partial derivatives defined as

$$(h_{ij}) = \frac{\partial f}{\partial x_i \partial x_j}.$$
We say that the vector $x$ is obtained by updating the vector $x_k$ with the residual $\Delta x$. We want to see 6.1.5 as a function of $\Delta x$, and want to minimize the error

$$E(\Delta x) = f(x) - f(x_k) = (\Delta x)^T \nabla f(x_k) + \frac{1}{2} \Delta x^T H(x_k) \Delta x.$$ 

The error gets minimized (if it does) at a critical point. That is we should satisfy the equation

$$E'(\Delta x) = 0 \quad \Rightarrow \quad \nabla f(x_k) = -H(x_k) \Delta x,$$

from which

$$\Delta x = -H^{-1} \nabla f(x_k). \quad (6.1.6)$$

Differentiation was taken with respect to $\Delta x$.

The Newton optimization recursion is then

$$x_{k+1} = x_k - H^{-1} \nabla f(x_k), \quad (6.1.7)$$

assuming, again, that the Hessian matrix $H$ is non–singular. Note that for one dimension this reduces to formula 6.0.2. Formula 6.1.6 is quite powerful. All quasi-Newton and gradient methods could be seen as derived from this formula. For example let us assume of the linear CG for a quadratic form

$$f(x) = \frac{1}{2} x^T H x - bx + c. \quad (6.1.8)$$

The CG iteration computes

$$x_{k+1} = x_k - \alpha_k p_k,$$

That is

$$\alpha p_k = -\Delta x_k.$$ 

On the other hand, if we take the gradient of the quadratic form 6.1.8 in two different points, that is

$$\nabla f(x_k) = H x_k - b$$
$$\nabla f(x_{k+1}) = H x_{k+1} - b$$

and subtract, we find

$$\nabla f(x_{k+1}) - \nabla f(x_k) = H(\Delta x_{k+1}) = -H \alpha p_k \quad (6.1.9)$$
from which
\[ \alpha p_k = -H^{-1}(\nabla f(x_{k+1}) - \nabla f(x_k)), \]

Note that \( \alpha = 1 \) can work here. That is, what matters is the product of the inverse of the Hessian times the difference of the gradient. The \( \alpha \) comes embedded on that product. That is why we see in the literature commonly the expression that \( \alpha = 1 \) is the “natural” step length associated with the Newton direction above. Some quasi-Newton programs encourage users to start with \( \alpha = 1 \).

In terms of \( \Delta x \) we have
\[ \Delta x_k = H^{-1}(\nabla f(x_{k+1}) - \nabla f(x_k)), \tag{6.1.10} \]

In summary we find

<table>
<thead>
<tr>
<th>Method</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steepest Descent</td>
<td>( x_{k+1} = x_k - t\nabla f(x_k) )</td>
</tr>
<tr>
<td>Linear Conjugate Gradient</td>
<td>( x_{k+1} = x_k + \alpha p_k )</td>
</tr>
<tr>
<td>quasi-Newton Optimization</td>
<td>( x_{k+1} = x_k - H^{-1}\nabla f(x_k) )</td>
</tr>
</tbody>
</table>

Even non–linear CG could be seen as particular cases of the Newton method. We will address this issue later. In functions of one independent variable \( H^{-1} = t \). Gradient methods in one dimension should all coincide since there is only one direction to go (left or right of a point) and that direction should be always in the direction of the decreasing side of the function. For functions of more than one independent variable \( H \) is an \( n \times n \) matrix (where \( n \) is the number of independent variables). The SD method is pointing always along the negative gradient direction, while the Newton method has a combination of the curvature and the gradient in a way that for a quadratic function it will converge in one step. The SD method would require more than one step (and perhaps many, as shown in the example for Figure 3.4.). Newton methods should require less iterations. Upon observing the Newton optimization equation 6.1.10, we see that
\[ H\Delta x = \nabla f(x). \]

So that if the gradient \( \nabla f(x) \), and the step direction \( \Delta x \) (which is the same conjugate direction \( p \)) coincide then we say that the conjugate direction \( p \) is an eigenvector of the Hessian matrix \( H \). In this case we are on a principal axis, and so the CG iteration coincides with the SD. The computation of the Hessian matrix in practice can be a challenge. Gradient methods require more iterations but the gradient could be quickly computed using the Adjoint State method (a reference here).

Given that the computation of the Hessian is the bottle-neck of the Newton methods, new theories were developed to make the Newton optimization methods more efficient. We sketch a few ideas below.
6.2 Quasi-Newton Methods

Quasi-Newton methods are algorithms provided to overcome the high cost of estimating the Hessian matrix. In these methods the Hessian matrix is computed indirectly or not computed at all.

For example, if we want to find a local minimum using the Newton optimization method, we can differentiate the function that we want to minimize. Then we find a root of the resulting function. A root of the differentiated function is a critical point (minimum, maximum, or inflection point). When finding the root of that function we can use the secant method (see the algorithm 6.0.4), which does not require differentiation. The generalization of the secant method to several variables is known as the Broyden's Method in the name of his inventor, who introduced it in 1965.

Let us re-write the Taylor approximation 6.1.5

\[ f(x_k + \Delta x) \approx f(x_k) + (\Delta x)^T \nabla f(x_k) + \frac{1}{2} \Delta x^T H(x_k) \Delta x. \quad (6.2.11) \]

If we compute the gradient of \( f \), this time with respect to the multi-dimensional variable \( \Delta x \), we find

\[ \nabla f(x_k + \Delta x) \approx \nabla f(x_k) + H(x_k) \Delta x. \]

Note that this is the Taylor series of \( \nabla f \), at \( x_k \). We find the Hessian \( H \) by solving the equation

\[ \nabla f(x_k + \Delta x) = \nabla f(x_k) + H(x_k) \Delta x. \quad (6.2.12) \]

Seen in this way, the Hessian is computed as a forward difference operator on the gradient. The problem is that while we would love to say that

\[ H(x) = \frac{\nabla f(x_k + \Delta x) + \nabla f(x_k)}{\Delta x} \quad \text{(wrong!)} \]

this is incorrect, since we can not just divide by a vector. What is correct is to write

\[ \nabla f(x_k + \Delta x) + \nabla f(x_k) = H(x) \Delta x. \quad (6.2.14) \]

This is secant condition derived in 6.1.10.

Hence we can say that the Quasi-Newton methods consist of algorithms that implement the step increment, according to equation 6.1.7, but where the Hessian \( H \) is updated at each iteration from an initial guess \( H_0 \). We then rewrite the Newton equation 6.1.7 as the quasi-Newton approximation

\[ x_{k+1} = x_k + H_n^{-1} \nabla f(x_k), \quad (6.2.15) \]

where \( H_n \) is assumed known from a previous iteration. We say that in the Quasi-Newton methods both the initial solution and the initial Hessian are guessed, and after that they are updated at each step. Next is the general algorithm for the Quasi-Newton methods.

Algorithm 6.2.0

- Pick and initial $x_0$ in the domain.
- Compute the quasi-Newton direction
  \[ \Delta x = H_0^{-1}\nabla f(x_0), \]  
  \[ (6.2.16) \]
  from equation 6.2.15
- Determine the step size $t$ (line search parameter) from
  \[ x_1 = x_0 + t\Delta x \]
- Compute $H_1$.
- Increment index and go back to the second step on this algorithm in a loop.

Up to this point we have not said much new. This algorithm is almost identical to the Newton algorithm but two things deserve special attention. First, is how to update the Hessian and second, how about the line search?

We had already discussed the line search when doing SD in the derivation between equations 3.2.3 and 3.2.8. However for the Newton and Quasi-Newton methods we need to know more about line search methods. Hence, we make a stop here to study line search methods.

### 6.3 Line Search Methods

Let us, for the moment, assume that given a function $f(x_k)$ with $x_k \in \mathbb{R}^n$, we already know a vector $p \in \mathbb{R}^n$ along which the function will locally decrease. That is, we want to search $\alpha \in (0, b)$, such that

\[ g(\alpha) = f(x + \alpha p) \]

has the largest descent. Note that $\alpha = 0$ is not an option since we want to move to a different point. A very large $\alpha$ might take us out of the local convexity to another valley where we might not want to be, so the size of $\alpha$ should be limited by some value $b > 0$. How to find $b$? Figure 6.3 shows an illustration of the problem. If very small values are used, the algorithm becomes slow due to the increase on the number of iterations. For example, in the figure we have a starting point $x_0 = (0.148, 0.7)$. In the black path the search is moving at a turtle jumps and so, although we are moving to the right local minimum of $(0.183, 0.5)$, which we arbitrarily picked, it will move slowly. On the other hand along the green path the search is moving at hare steps but the steps were too long and the first step took us out to the wrong valley where the local minimum
Figure 6.3: This figure illustrates a function with peaks and valleys, and how a step size $\alpha$ could slow down the process or take us to the wrong minimum.
6.3 Line Search Methods

Figure 6.4: Two possible functions \( g(\alpha) \), and their optimal bracket values \( b_1 \) and \( b_2 \).

We could start with a very small (how small?) value \( \epsilon \), and then double it each time asking the question at each \( k \)-th step.

\[
\text{is still } f(x + 2^k \epsilon p) \leq f(x) \?
\]

The exit \( k \) tells us that \( b = 2^{k-1} \epsilon \). So \( \epsilon \) is a parameter to start the line search on each case. Controlling the size of the step at each iteration is important. A large step produces an inaccurate search, with less iterations, a small step requires better precision at a higher cost. This is a multi-scaling problem and a strategy is to start with the low frequencies, that is a large step, then refine the steps for resolution \((0.43, 0.5)\) would be found instead.

One good thing is that no matter how many dimensions are represented by the point \( x \), the function \( g \) is a function of only one variable \( \alpha \).

Let us, for the moment, assume that we want to use only the condition \( g(b) < g(0) \) so that we guarantee that we are moving down. If \( g(0) \) is sitting at the global maximum, such a \( b \) would not exist (or it has to be infinity). Let first see an illustration about the initial bracketing to locate \( b \). If \( g(0) \) is at a local maximum, the optimum \( b \) (least upper bound) will still skip over the next valley. This illustrated in Figure 6.4. The blue figure shows that \( b_1 \) bracketed the search passed the second valley, but perhaps if our initial point (red) is in the first valley, we wanted to point to the local minimum in that valley. If instead, \( g(0) \) would start lower (brown curve) then we see that the optimum \( b_2 \) brackets the search interval fine.

Knowing the right size of \( \epsilon \) is of great importance to accelerate convergence. It is important to understand that \( \epsilon \) should be smaller than half the shortest wavelength
of the objective function to avoid skipping to the wrong valley. However an analytic or even numerical estimation of the cost function as a function of the parameter $\alpha$ might not be at hand or could be difficult to estimate. In other words, it might be easier to find a local minimum of $g(\alpha)$ that to know its Fourier spectrum and so the minimum wavelength.

The Wolfe Conditions:

Some ideas to estimate how large is large and how small is small have been explored. These ideas are synthesized under the name Wolfe Conditions \(^5\) that we study next.

We want to focus in size steps on the $\alpha$ axis. Of course at each $n$ step, we should have that $g(\alpha_n) < g(\alpha_{n-1})$, however this will not be of great help, since if the function is very flat, small steps would take much longer to get to the minimum.

Recall that the first order Taylor series approximation corresponding to $f$ is

$$g(\alpha) = f(x_k + \alpha p_n) \approx f(x_k) + \alpha \nabla^T f(x_k) p_n.$$  

The function

$$l(\alpha) = f(x_k) + \alpha \nabla^T f(x_k) p_n,$$

is a tangent line to $g(\alpha)$, at $\alpha = 0$. If we assume global convexity (a single global minimum), The function $g(\alpha)$ is then always on top of its tangent $l(\alpha)$. For local convexity the could be sevaral local minima. Figure 6.5 shows a sketch the case of local convexity with two local minima (on the green valleys). The tangent is below the graphic. However if the second green valley would drop much for to the negative side, that tangent could still intersect the curve and some points of the curve would bo below the tangent line.

We would like to bracket the first green valley. A horizontal line (obtained by breaking the condition $g(\alpha) < g(0)$ ) can estimate an $\alpha$ too far. To get a better estimate Armijo formulated the following idea.

It makes sense to raise the line from a tangent to a secant, reducing the size of the slope with a factor $c_1 \in (0, 1)$. That is, we can choose a (secant) line

$$\ell(\alpha) = f(x_k) + c_1 \alpha \nabla^T f(x_k) p_n,$$

The extreme case of $c_1 = 0$ whould draw a horizontal line through the point $(0, g(0))$. This is the case of violating the inequality $g(\alpha) < g(0)$. This horizontal line, will never intesect the $\alpha$ axis and so it is not of interest. The case of $c_1 = 1$ is the tangent line which is another extreme. All secant lines of interest are in the angle between the tangent and horizontal lines. This corresponds to values of $c_1 \in (0, 1)$. The range of $\alpha$ values which are appropriate

The Armijo extreme value tells us to pick $b$ as the smallest number $\alpha$ where the condition

\(^5\)http://en.wikipedia.org/wiki/Wolfe_conditions
6.3 Line Search Methods

Figure 6.5: A tangent line $l$, corresponding to $c_1 = 1$, secant line $ℓ$, for $c_1 \in (0, 1)$ and a horizontal line $h$ for $c_1 = 0$. The point marked as $b$ is the least number such that $g(\alpha)$ is still smaller than the linear function $c_1 g'(0)\alpha$.

$$g(\alpha) = f(x_k + \alpha p_n) > f(x_k) + c_1 \alpha \nabla^T f(x_k)p_n$$

for $c_1 \in (0, 1)$, is achieved.

The range of $\alpha$ values in Figure 6.5, that satisfy the Armijo condition are those corresponding to the green regions. The $\alpha$ searched in the Armijo condition is labeled in the figure at the end of the first (green) valley as $b$.

Before we study the second Wolfe condition let us illustrate the Armijo condition with an specific example in one variable.

**Example 6.3.1**

- Let us assume the simple function

$$f(x) = x^2 - 1, \quad -2 \leq x \leq 2 \quad (6.3.17)$$

The left panel on Figure 6.6 illustrates the function. It is obvious that the global minimum is located at $(0, -1)$, but the purpose is to exercise the descent method.

$$f'(x) = 2x \quad (6.3.18)$$

We use as the direction vector the scalar $p_k = -df/dx|_{x=x_k}$. At any step point $x_k$. The function $g$ is defined as

$$g(\alpha) = f(x_k + \alpha p_k) = f(x_k - \alpha f'(x_k))$$
and

\[ g'(\alpha) = f'(x_k + \alpha p_k) p_k \]

from which

\[ g'(0) = -[f'(x_k)]^2 < 0 \]

which guarantees that \( g \) is decreasing at start. Let us pick \( x_0 = -2 \) as the initial point. We find \( f(-2) = g(0) = 3, f'(-2) = -4, \) and \( g'(0) = -16. \) We find an \( \alpha \) which satisfies the Armijo condition, by choosing \( c_1 = 0.1 \) (recall \( c_1 \in (0, 1) \)). The secant line at for \( x_0, \) is given by

\[ l(\alpha) = g(0) + c_1 g'(0) \alpha = 3 - (16)(0.1)\alpha = 3 - 1.6\alpha \tag{6.3.19} \]

In addition, \( g(\alpha), \) given this initial point is given by

\[ g(\alpha) = f(-2 - \alpha f'(-2)) = f(-2 + 4\alpha) = (4\alpha - 2)^2 - 1. \tag{6.3.20} \]

Figure 6.6 (right panel) illustrates the function \( g(\alpha) \) in blue color. The secant function is also drawn in blue color.

The maximum \( \alpha \) that we can choose is in the intersection of the function \( g(\alpha) \) defined in equation 6.3.20 and the secant line defined in equation 6.3.19. This \( \alpha \approx 0.904. \) We choose the next \( \alpha \) to be \( \alpha = 0.9 \) and so

\[ x_1 = x_0 - \alpha f'(x_0) = -2 - (0.9)(-4) = 1.6 \]
We redefine \( g(\alpha) \) as \( g(\alpha) = f(x_1 + \alpha p) = f(1.6 - f'(1.6)\alpha) \), and since \( f'(1.6) = 3.2 \) then

\[
g(\alpha) = f(1.6 - 3.2\alpha).
\]

A shift to the left of 1.6 and a squeeze/flip of 3.2 along the \( x \) axis on the function \( f \). The evaluation of the function \( f(x) \) with \( x = 1.6 - 3.2\alpha \) is given by

\[
g(\alpha) = (1.6 - 3.2\alpha)^2 - 1
\] (6.3.21)

From \( g(0) = f(1.6) = 1.56, f'(x_1) = 3.2, g'(0) = -f'(x_1)^2 = -10.4 \), \( c_1 * g'(0) = -1.4 \). The secant line \( l(\alpha) \) corresponding to this new point is

\[
l(\alpha) = g(0) + c_1 g'(0)\alpha = 1.56 - 1.4\alpha.
\]

Figure 6.6 shows in red (right panel) the curve \( g(\alpha) \) together with the secant that limits the Armijo condition.

The maximum \( \alpha \) value, which is the intersection between the graphics of \( l(\alpha) \) and \( g(\alpha) \) is approximatelly 0.903, so we can, again, pick \( \alpha = 0.9 \). The next point is

\[
x_2 = x_1 - \alpha f'(x_1) = 1.6 - (3.2) * 0.9 = -1.28.
\]

We redefine \( g(\alpha) = f(x_2 - \alpha f'(x_2)) \), and since \( f'(x_2) = -2.56 \), we have

\[
g(\alpha) = (-1.28 + 2.56\alpha)^2 - 1
\] (6.3.22)

From \( g(0) = f(x_2) = 0.6384, f'(x_2) = -2.56, g'(0) = -f'(x_2)^2 = -6.5536 \), \( c_1 * g'(0) = -0.5536 \). The secant line \( l(\alpha) \) corresponding to this new point is

\[
l(\alpha) = g(0) + c_1 g'(0)\alpha = 0.6384 - 0.65536\alpha.
\]

Functions \( g(\alpha) \) and \( l(\alpha) \) are plotted in Figure 6.6 in green color.

The maximum \( \alpha \) if about 0.9002. With \( \alpha = 0.9 \) we compute

\[
x_3 = x_2 - \alpha f'(x_2) = -1.28 - (0.9)(-2.56) = 1.024
\]

We evaluate \( f(x_3) = 0.048576 \)

We redefine \( g(\alpha) = f(x_3 - \alpha f'(x_3)) \), and since \( f'(x_3) = 2.048 \), with \( x = 1.024 - 2.048\alpha \),

\[
g(\alpha) = (1.024 - 2.048\alpha)^2 - 1.
\] (6.3.23)
From $g(0) = f(x_3) = 0.048576$, $f'(x_3) = 2.048$, $g'(0) = -f'(x_3)^2 = -4.194304$, $c_1 * g'(0) = -0.4194304$. The secant line $l(\alpha)$ corresponding to this new point is $l(\alpha) = g(0) + c_1 g'(0) \alpha = 0.048576 - 0.4194304 \alpha$.

Functions $g(\alpha)$ and $l(\alpha)$ are plotted in Figure 6.6 in brown color.

The maximum $\alpha$ is about 0.9. With $\alpha = 0.9$ we compute

$$x_4 = x_3 - \alpha f'(x_3) = 1.024 - (0.9)(2.048) = -0.8192$$

We evaluate $f(x_4) = -0.33$

We stop at this point and list the points found so far

$$(x_0, f(x_0)) = (-2, 3)$$
$$(x_1, f(x_1)) = (1.6, 1.56)$$
$$(x_2, f(x_2)) = (-1.28, 0.6384)$$
$$(x_3, f(x_3)) = (1.024, 0.048576)$$
$$(x_4, f(x_4)) = (-0.8192, -0.32891136)$$

Figure 6.7 shows on the left panel the path between the descent points found so far. If the Armijo, as well as the second Wolfe curvature (described below) are satisfied, together with a Lipchitz condition

$$\|g(x) - g(x')\| \leq L\|x - x'\|$$

for form $L > 0$, then we can guarantee convergence. See the notes on the Zoutendijk’s condition below.
The example shown here is powerful not only in the sense that it details the step follows in the Armijo algorithm (also described in the Armijo Rule search 6.3 but because it can serve as a counter–example to show that if the Armijo condition is not acknowledged there might be no convergence. Let us question the convergence of not of the sequence

\[ x_{k+1} = x_k - \alpha f'(x_k) \]

as a function of \( \alpha \). Since \( f'(x) = 2x \), we write the analytical expression

\[ x_{k+1} = x_k - 2\alpha x_k \]

(6.3.24)

Let us assume, without loosing generality, that \( k = 0 \) and that \( x_0 \) is not -2, but the first value in the \([-1, 1]\) interval where the original sequence of \( x_k \) numbers fell. Then we have by calling \( \beta = (1 - 2\alpha) \),

\[
\begin{align*}
x_1 &= \beta x_0 \\
x_2 &= \beta x_1 = \beta^2 x_0 \\
\vdots \\
x_{k+1} &= \beta^{k+1} x_0.
\end{align*}
\]

For the simple function we are studying we know the solution (minimum abscissa) is \( x = 0 \). We have the following observation

- if \( |\beta| < 1 \) then \( x_k \to 0 \).
- if \( |\beta| = 1 \) then \( x_k = \pm x_0 \).
- if \( |\beta| > 1 \) then \( x_k \to \infty \).

So, for convergence we want

\[ |1 - 2\alpha| < 1. \]

That is

\[-1 < 2\alpha - 1 < 1 \quad \Rightarrow \quad \alpha < 0 < 1 \]

The right hand side panel on Figure 6.7 shows the situation when \( \alpha = 1 \). The sequence of points oscillates between the points \((-0.8192, -0.32891136)\) and \((0.8192, -0.32891136)\) for ever.

It is clear in this simple example how the Armijo condition helps on the convergence of the line search algorithm. In functions much more complicated, the Armijo condition helps in the same way, and this is the power of the constrain.

The second Wolfe condition is the curvature condition. This condition prevent us from choosing a too small step.
The curvature condition is a condition of the derivative of $g$, this is we want to pick $\alpha$ such that

$$g'(\alpha) \leq c_2g'(0),$$

with $c_2 \in (c_1, 1)$. In terms of the function $f$ this is

$$\nabla f(x_k + \alpha p_n)^T p \leq c_2\alpha \nabla f^T_n p_n$$

The ideal $\alpha$ would be the largest $\alpha$ such that

$$\nabla f(x_k + \alpha p_n)^T p \leq c_2\alpha \nabla f^T_n p_n$$

Figure 6.8 illustrates the curvature condition. The brown line $C$ represents the linear function $C(\alpha) = c_2g'(0)$, where, since $c_2 > c_1$ its slope is larger (in absolute value, recall $g'(0) < 0$) than that of the secant $l = c_1g'(0)\alpha$, but smaller (in absolute value) that the tangent line $\ell(\alpha) = g'(0)\alpha$. Since the curvature of the function is changing we find that at some point the line $C(\alpha)$ will have a tangent to the curve $g(\alpha)$. The parallel line tangent to $g(\alpha)$ with the slope $c_2g'(0)$ is noted as $C\parallel$. It is at this point where the second Wolfe condition turns around. Let us call this point $a$. That is, at this point the slope was smaller (negatively, but larger in absolute value) and turn larger (negatively, but smaller in absolute value). So the turning point $a$ is the largest $\alpha$ such that $g'(\alpha) < c_2g'(0)$. In Figure 6.8 $a$ is labeled above the dashed line from the tangent point. The thick dashed line spanning the interval $[a, b]$ is the ideal bracket for the Wolfe conditions on the figure.

However in spite of the important bracketing of the initial search interval this is not the most important consequence of the Wolfe conditions. The Wolfe conditions ensure two things.

(i) If the gradient $\nabla f(x_k)$ is not orthogonal to $p_n$, (which is the case for CG, SD and the Newton methods) then it can be shown that any iterative methods (as the ones described below) satisfying the Wolfe conditions, will converge in a way that $\nabla f(x_k) \to 0$. The following inequality

$$\sum_{k \leq 0} \cos^2 \theta_k \| \nabla f_k \|^2 \leq \infty$$

where, $\theta$ is the angle between the descent direction and the gradient of $f$

$$\cos \theta_k = \frac{\nabla f_k^T p_k}{\| \nabla f_k \| \| p_k \|}$$

is known as the Zoutendijk’s Condition after his inventor. From this inequality, and providing that $\cos \theta_k$ is bounded away from zero, we have that $\nabla f_k \to 0$. The requirements to guarantee the Zoutendijk’s condition as well as the proof of it are shown as Theorem 3.1 in the link above. It is interesting that as $\theta \to 0$
the method approaches the SD. We want to avoid \( \theta \) close to \( \pi/2 \). Some quasi-Newton methods can get the angle close to \( \pi/2 \) and so there should be a constraint to limit the increase of \( \theta \). There is a descent method called *coordinate descent* which loops through coordinate axes as the vectors \( p_k \). In this method it could happen that the angle between the gradient and the direction \( p_k \) could be \( \pi/2 \). This method (no classified here) is not recommended, even though it does not requires the gradient and neither the Hessian. If it converges it could converge very slow. As a final note we want to point out that once satisfied the Wolfe conditions the line search can stop and a new direction can be searched for. The Wolfe conditions provided an inexact search but they are necessary to allow convergence along the way. However this does not imply that we can not further zoom up into the minimum before searching for a new direction \( p_k \).

(ii) The other important property of the Wolfe conditions is that if they are satisfied and we have an estimation of the Hessian \( B_k^{-1} \) during an iteration on quasi-Newton method, the next \( B_{k+1}^{-1} \) is assured to be positive definite provided that the first \( (B_k^{-1}) \) is positive definite.

Next step, after bracketing the initial search interval, is to to start the search for the minimum in such an interval. The following line search algorithms have the power that they do not require any derivatives and work with any set of data smooth or not, however they require additional iterations.
**Bisection search:**

This is perhaps the simplest, but expensive, rule. Knowing that the interval for searching is \((0, b]\), we can assign to \(\alpha\), the value on the middle of the searched interval, which in this case is \(\alpha = b/2\) and choose the better half interval for the next iteration. That is, if \(f(x + (b/2)p) < f(x + b p)\), the next iteration will search the interval \((0, b/2]\), otherwise the search proceeds to the interval \([b/2, b]\). Each steps narrows the search by half. Still this algorithm could be expensive. The stopping condition could be the length of the current interval, which would be \(2^{-k}b\) at the \(k\)-th iteration.

**Golden Rule search:**

The [Golden section search] rule. Let us assume that our function is convex defined in the domain \((a, c)\), and the current search point is some \(x \in (a, c)\). The bisection method uses a ratio of \(r = 1/2\) to divide the original interval into two new intervals. Can we do better than this? Our answer is “no”, however the Internet is polluted with sites that claim that the golden ratio search is faster. We show next that this is not the case, but first, let us define the golden rule method.

We start our golden ratio rule with two points \(x_1, x_2\) in the interval \((a, c)\). Please refer to Figure 6.9 for the following discussion.

For both points \(f(x_1) < f(a), f(b)\). However if \(f(x_1) < f(x_2)\) we only know that the next bracketed interval is \((a, x_2)\). On the other hand, if \(f(x_1) \geq f(x_2)\), then the next bracketed interval should be \((x_1, c)\).

We do not know, ahead of time, which is the case \(f(x_1) > f(x_2)\) or the reverse. 

---

inequality. Then we want to assign equal probability to avoid the risk of getting into the low probability zone. That is, we want to assume that the length of both intervals is equal. That is,

\[ x_2 - a = c - x_1 \]  \hspace{1cm} (6.3.25)

We want additionally condition that the proportion between the initial interval and the length for the inverval determined by \( x_1 \) be the same as the length of the inverval determined by \( x_1 \) and the interval after picking \( x_2 \). That is we want the proportion

\[
\frac{c - x_1}{c - a} = \frac{c - x_2}{c - x_1} = \phi
\]

This is the reason for the name golden ratio\(^7\) search. Let us write an equation to solve for \( \phi \). Using equation 6.3.25 and the previous equation, we see that

\[
\phi = \frac{c - x_1}{c - a} = \frac{x_2 - a}{c - a} = \frac{x_2 - c + c - a}{c - a} = 1 + \frac{x_2 - c}{c - a} = 1 + \frac{x_2 - c}{x_1 - c} \cdot \frac{c - a}{x_1 - c} = 1 + \frac{r}{-1/\phi}
\]

From which we find

\[
\phi^2 - \phi - 1 = 0 \quad \Rightarrow \quad \phi = \frac{1 \pm \sqrt{5}}{2}
\]

Because \( \phi > 0 \) (the ratio of two segment lengths) we pick the positive solution

\[
\phi \approx 1.61803398874989484820
\]

This is the golden ratio\(^8\). We found that the reduction on side of the interval is

\[
r = \frac{1}{\phi} = .61803398874989484820
\]

of the original side, and since \( 1/2 < 1/\phi \), the convergence rate of the bisection method which is like \( (1/2)^k \) is faster (smaller interval length) that \( (1/\phi)^k \) which is the convergence rate of the golden ratio search.

Please note that while in the golden ratio search we started with two points, the next points are added one–by–one by shifting indices by one.

With the observation that the golden ratio method is not faster than the bisection and still the algorithm is much more complex, we do not yet understand why it is this method still part of the scientific literature. Yes, the golden ratio has a lot of art and a lot of science but it is our believe that this is not the place for it and it confuses more than helps on the searching for optimal solutions.

\(^7\)http://en.wikipedia.org/wiki/Golden_ratio
\(^8\)http://en.wikipedia.org/wiki/Golden_ratio
Armijo Rule search:

More than in any other line search method, Armijo’s algorithm put emphasis on the size of the step so that it is not too large as to lose accuracy or too small to lose speed. The idea behind the algorithm is to use a first order estimation of the descent distance to find the step that bounds the evaluation of $f$ under the linear regime. That is, by doing a first order Taylor series expansion we find

$$f(x_k + \alpha_n p_n) \approx f(x_k) + \alpha_n \nabla f(x_k)^T p_n.$$  

The way to bound the new point $x_{k+1} = x_k + \alpha_n p_n$ so that is is under the linear regime is to pick a shrink parameter $c \in (0, 1)$ such that

$$f(x_k + \alpha_n p_n) \leq f(x_k) + c \alpha_n \nabla f(x_k)^T p_n.$$  \hspace{1cm} (6.3.26)

This conditions ensures that the step $\alpha_n$ is not too large. To reduce the step the algorithm use a shrink factor $\tau \in (0, 1)$. Armijo\(^9\) used $1/2$. This shrinking is the same level of shrinking used for the bisector search method.

The Armijo, or Backtracking line search algorithm is as follows.

**Algorithm 6.3.1**

- Bracket the initial $\alpha$ to a value $b$ as indicated in the beginning of section 6.3. Give it the name $\alpha_0$.

- Loop (for each $n$) until the Armijo condition 6.3.27 is violated. That is, until

$$f(x_k + \alpha_n p_n) > f(x_k) + c \alpha_n \nabla f(x_k)^T p_n$$  \hspace{1cm} (6.3.27)

If the Armijo condition is violated, then reduce the search interval by a factor $\tau$. That is, the new interval will have the length $\tau \alpha_n$.

This reduces the interval until the Armijo condition is fulfilled.

Example 1 shows a sequence of steps in which the Armijo condition is always checked. The algorithm here is done to find the bracket of the Armijo condition, since there is not an analytical way to evaluate the equation $g(\alpha) = l(\alpha)$ to bracket this first interval.

Note that this method is not a precise method. It is more a tuning method to guide the search so that the step length makes sense for the problem at hand.

\(^9\)http://en.wikipedia.org/wiki/Backtracking_line_search
### Parabolic Interpolation search:

In functions of one variable. If the function has a local minimum, we can approximate the function, up to a second order, with a parabola. To approximate a parabola we need exactly 3 points. Assume a function of one dimension \( f(\alpha) \). Use three points \( a, b \) and \( c \) in the interval of interest. Solve the system

\[
f(a) = Sa^2 + Ta + Uf(b) = Sb^2 + Tb + Uf(c) = Sb^2 + Tb + U
\]

for the unknown variables \( S, T \) and \( U \). This will provide the theoretical fitting parabola. The minimum should be located at that point such that \( f'(\alpha) = 0 \) which is \( 2S\alpha + T = 0 \), or

\[
\alpha = -\frac{T}{2S}.
\]

An explicit evaluation of the function is given by the formula

\[
\alpha = b - \frac{1}{2} \frac{(b-a)^2[(f(b)-f(c)) - (b-c)^2[(f(b)-f(a))]}{(b-a)[(f(b)-f(c)) - (b-c)[f(b)-f(a)]}.
\]

For a derivation of this equation visit [this](http://linneus20.ethz.ch:8080/1_5_2.html) website. Some algorithms use cubic splines to obtain intermediate values along a smooth curve.

As a final note, before finishing this section, we point out that for Newton methods \( \alpha = 1 \) and no line search should be necessary. For quasi-Newton methods, the search should start with \( \alpha = 1 \). If, for \( \alpha = 1 \), the Wolfe conditions are satisfied then no further line search is required.

### 6.4 Quasi-Newton methods revisited

As indicated above the optimization methods have two main elements. A direction to search and a line search algorithm through the searched direction. It is the case that the computation of the direction is at a much higher cost that that of the line search parameter. In the case of Quasi–Newton algorithms, since the Hessian could be very expensive, several approximations have been researched.

The general algorithm 6.2.0 shows the steps to evaluate the new increment toward the optimal solution of the problem. The step size along the line search was discussed in the previous section. However, the Hessian \( H_0 \) and its updates were left unresolved. The completion of the algorithm requires a technique to find the Hessian iterations. We will study a few quasi-Newton algorithms which we consider important because of the historic context and the best current practices. The Davidon-Fletcher-Powell method \(^{11}\), known as the DFP algorithm, the Broyden-Fletcher-Goldfarb-Shannon

\(^{10}\)http://linneus20.ethz.ch:8080/1_5_2.html
\(^{11}\)http://en.wikipedia.org/wiki/Davidon%E2%80%93Fletcher%E2%80%93Powell_formula
algorithm 12 most popularly known as the BFGS algorithm, and the Limited Memory BFGS 13 algorithm (referred to as L-BFGS), are likely the most popular in the published literature for quasi-Newton techniques.

In the quasi-Newton methods we start with the secant condition formula 6.2.14

\[ \nabla f(x_k + \Delta x) - \nabla f(x_k) = H_{k+1}(x) \Delta x, \]

where, instead of using the exact Hessian \( H \) (as it is assume in the Newton equation 6.2.14) we use an approximated Hessian \( H_{k+1}(x_k) \).

To simplify notation we rename

\[ g_k = \nabla f(x_k), \quad \gamma_k = g_{k+1} - g_k, \quad \Delta x_k = \delta_k, \quad B_{k+1} = H_{k+1} \]

so the secant condition can be written as

\[ \gamma_k = B_{k+1} \delta_k \]

The notation \( B_{k+1} \) is well spread on the literature and we believe that it makes honor to Charles Broyden 14 of the BFGS method. The notation \( H_k \) will be reserved for \( B_k^{-1} \). The notation of \( \gamma \) for the difference of gradients and \( \delta \) for the difference of coordinates is taken from Fletcher’s

**Historical Introduction:**

The starting of the quasi-Newton algorithms is attributed to William Davidon. Davidon discovered in 1959 the first quasi-Newton method which today carries his first initial on the DFP method. The manuscript describing this idea was not accepted for publication, and it was written as an internal report for the Argonne National Laboratory. His manuscript 15 appeared published more than 40 years later (in 1991) in the first issue of the SIAM Journal of Optimization. The original name on Davidon’s paper was “variable metric method for minimization”. The reason for this name is that closed to a minimum, a convex function can be approximated by the equation 6.1.5, where by assuming \( \nabla f(x_k) = 0 \) at the local minimum, we estimate the change from the minimum (in Davidon’s notation) as

\[ \Delta f(x_k) \approx \frac{1}{2} h^{\mu \nu} d_{\mu} d_{\nu} \quad \text{sum over repeated indices} \]

In Differential geometry the Hessian matrix \( H = h_{ij} \) (which is assumed positive definite and symmetric, can be considered as a metric tensor. That explains the word “metric” in the title. The word “variable” is because this Hessian matrix is changing

---

12Search for BFGS in Wikipedia  
at each iteration. It is like changing the metric at each iteration. Fletcher and Powell demonstrated that this algorithm was much faster and more reliable than existing methods and it revolutionized the field of non-linear optimization as the first quasi-Newton method (hence the initial ‘F’ and ‘P’ in the name DFP). Fletcher and Powell paper modify the original Davidon algorithm. While the results of Davidon origina’s work are important his manuscript is hard to read and we want to rescue his main idea in a few lines, from which all other quasi-Newton algorithms were built upon.

Davidon posted an equation which is at the center of the Hessian updates and which was the key starting of the quasi-Newton methods. This equation was introduced as follows: “If the step had been taken on the basis of \( h^{\nu \mu} \), we modify \( h^{\nu \mu} \) so as to double the squared length of \( s^\mu \), leaving the length of all perpendicular vectors unchanged. This is accomplished by

\[
(4.4) \quad h^{\nu \mu} + \ell \frac{1}{\ell} s^\mu s^\nu \rightarrow h^{\nu \mu},
\]

where \( \ell \) is the square length of \( s^\mu \). This doubles the determinant of \( h^{\nu \mu} \).” For Davidon \( s^\mu \) is our search direction vector \( p \), and what he is saying is that the equation does not change the length of any perpendicular vector to the direction \( p \). That is, if \( r \) is a direction which is perpendicular to \( p \) then the quadratic form

\[
 r^T H r
\]
on the new matrix \( H \) is

\[
 r^T H_{old} r + \ell \frac{1}{\ell} (r^T p)^2 (r^T r)^2 = r^T H_{new} r.
\]

unchanged. Let us compute the quadratic form for \( p \). That is

\[
 p^T H_{new} p = p^T H_{old} p + \frac{(p^T p)(p^T p)}{\|p^2\|} = p^T H_{old} p + \frac{\|p^4\|}{\|p^2\|^2} = p^T H_{old} p + \|p\|^2
\]

If \( H \) is preserving the norm of \( p \), then we say that

\[
p^T H p = \|p\|^2
\]

and so

\[
p^T H_{new} p = 2\|p\|^2
\]

with the new Hessian update the square length of the vector \( p \) on the new “metric” (the new Hessian matrix) is duplicated.

The construction of the new Hessian matrix with the formula

\[
 H_{new} = H_{old} + \frac{p^T p}{\|p\|^2};
\]
is not very natural but it is powerful. Besides preserving the behavior of the Hessian on orthogonal directions to \( p \) and stretching the length of the direction \( p \), it needs only an outer product which is a one–rank matrix, it preserves symmetry (the new \( H \) is symmetric) and it is still positive definite. We will show below the preservation of positive definitness in a more general representation of the new Hessian \( H_{\text{new}} \).

In general, the change to a new matrix \( A \) by adding an outer product, that is

\[
A + uv^T \rightarrow A
\]

is a well known problem. For example Ding and Zhou \(^{16}\) show a set of properties for rank-one updated matrices. Among them the property

\[
\det(A + uv^T) = (1 + v^T A^{-1} u) \det A
\]

where if \( A \) and \( A^{-1} \) preserve the metric along the direction \( v \) and \( v = u \) are unit vectors (as in Davidon’s paper, since he normalized with \( 1/\ell \) with \( \ell \) being square size) then,

\[
\det(A + uv^T) = 2 \det A
\]

for \( A = H \), and \( u = v = p \). This explains his statement about duplicating the determinant. An important property of of rank one matrices is its easy inversion. Sherman-Morrison formula \(^{17}\) states:

\[
(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^T A^{-1}u}.
\]

(6.4.31)

This formula is used on the implementations of methods based on the one rank matrix updates. Another important rank-one update of a matrix is the construction of the Housholder matrix

\[
P = I - 2vv^T
\]

which is used to reflect a vector along some direction \( v \).

Finally, a nice property (which Davidon did not mention at the time and that took more 20 years to be shown) is that Hessian updates based on rank-one and rank-two (we will elaborate about rank-two below) matrix consistent with the quasi-Newton updates satisfy a minimum change between one update and the next when using a weighed Frobenious norm. This property was found by J. Greenstadt and published in 1970. Greenstad’s paper is titled: Variations on Variable-Metric Methods.

Today, an axiomatic derivation of the DFP method is based on three basic postulates:

\(^{16}\)http://www.sciencedirect.com/science/article/pii/S0893965907000614
\(^{17}\)http://en.wikipedia.org/wiki/Sherman%E2%80%93Morrison_formula
(i) The approximate Hessian matrix $H_{k+1}$ is symmetric and positive definite. The symmetry can be accepted if we think that the Hessian is a matrix of second partial derivatives and that those mixed derivatives commute. The positive definiteness is necessary because we want to be able to invert the matrix. In addition this guarantees that our initial point is closed to a local minimum. In a different context, many optimization problems come from a least squares approach when the Hessian matrix is of the type $A^T A$, which, if full rank, has the properties of symmetry and positive definiteness. In fact, only the $H_0$ is required to be positive definite because, as we will see, this will imply that later $H_k$ matrices will be positive definite.

(ii) The approximate Hessian $H_{k+1}$ is such that it satisfies the secant condition 6.2.14.

(iii) Given that the secant condition 6.2.14 has $n$ equations, but the Hessian (symmetric) has $n(n+1)/2$ unknowns, the solution is far from being unique. $H_{k+1}$ should be as close as possible to $H_k$. The chosen norm is the weighted Frobenius norm. This property is attributed to J. Greenstadt. That is, we want to find $H_{k+1}$ such that

$$
\|H_{k+1} - H_k\|_W = \|W^{1/2}(H_{k+1} - H_k)W^{1/2}\|
$$

is minimum. Provided the convexity of the local region this defines $H_{k+1}$ uniquely.

We hope that with this historical introduction we motivated the reader to see the development of rank-one updates and its influence on quasi-Newton methods.

**One-rank updates on quasi-Newton Methods:**

The one-rank update is defined by the formula 6.4.30. That is, if $B_+$ is the updated matrix, then it can be written as

$$
B_+ = B + uv^T.
$$

Note that we dropped the subindex $k$ in favor of the subindex “+” as proposed originally by Davidon. In what follows, and to simplify notation, we will drop subindices $k$ and only introduce a subindex “+” when we want to indicate that the object will be updated. The name “one-rank” update comes from the fact that the matrix $uv^T$ is one rank. That is, all columns have the same direction. To better see this let us assume that we are working on the three-dimensional space of vectors, so

$$
uv^T = \begin{pmatrix}
    u_1 v_1 & u_1 v_2 & u_1 v_3 \\
    u_2 v_1 & u_2 v_2 & u_2 v_3 \\
    u_3 v_1 & u_3 v_2 & u_3 v_3
\end{pmatrix}
$$

\[18\] This is not always the case but for most practical purposes it does.
where it is clear that each column is a scaled version of $u$. Then the matrix has rank one.

If we acknowledge the secant condition 6.4.29, then

$$
\gamma = B_+ \delta,
$$

and so

$$
\gamma = (B + uv^T)\delta \\
= B\delta + uv^T\delta
$$

from which if $v^T\delta \neq 0$ then

$$
u = \frac{1}{v^T\delta}(\gamma - B\delta).
$$

(6.4.32)

This equation reveals an interesting result. The rank-one update will change the Hessian matrix in some direction. The direction is that of the vector $u$. The vector $u$ up to a scalar constant is the residual between the gradient difference $\gamma$ and the estimated Hessian direction product $B\delta$. If that residual is 0 (in which case there is no need to update) we are done. Otherwise we need to perturb the Hessian and the perturbation is precisely given by this residual. Let us think in terms of high school physics. Assume a particle in a circular trajectory. At two close points the velocities of the particles are along the tangent directions (the first derivatives which we compare with the gradients here). The difference of those derivatives (a secant to the circle) should measure an approximation (a secant approximation) to the centripetal acceleration with direction to the center of the circle. The curvature of the circle is also pointing at the center of the circle. In this example the approximation is good enough if both vectors are pointing to the center of the circle. If instead they point at different directions a correction (supplied by the difference between them) should be supplied. In a more general context the trajectory could be locally elliptical and the direction between the difference of two close velocities (tangents, or gradients) and the center of the ellipse is not orthogonal, but is $B$-orthogonal if $B$ is the exact Hessian matrix corresponding to this quadratic form. That is, the directions are conjugate.

From equation 6.4.32 we find the formula

$$
B_+ = B + \frac{(\gamma - B\delta)v^T}{v^T\delta}.
$$

(6.4.33)

This formula is quite general and by choosing convenient forms of the vector $v$ a gamma of known one-rank update methods is created.

Note that if instead of $B$ we work with its inverse $B^{-1}$ which we will call $H$, equation 6.4.33 would be written (interchanging the roles of $\delta$ and $\gamma$) as

$$
H_+ = H + \frac{(\delta - H\gamma)v^T}{v^T\gamma}.
$$

(6.4.34)
Table 6.1: A few one-rank update operators based on one-rank update dual equations

<table>
<thead>
<tr>
<th>Vector $v$</th>
<th>One-rank update</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v = \delta$</td>
<td>$B_+ = B + \frac{(\gamma - B\delta)\delta^T}{\delta^T \delta}$</td>
<td>Good Broyden</td>
</tr>
<tr>
<td>$v = \gamma$</td>
<td>$H_+ = H + \frac{(\delta - H\gamma)\gamma^T}{\gamma^T \gamma}$</td>
<td>Bad Broyden</td>
</tr>
<tr>
<td>$v = \gamma - B\delta$</td>
<td>$B_+ = B + \frac{(\gamma - B\delta)(\gamma - B\delta)^T}{(\gamma - B\delta)^T \delta}$</td>
<td>Symmetric Rank-one (SR1)</td>
</tr>
</tbody>
</table>

The equation pair 6.4.33 and 6.4.34 are a dual equations to find the one-rank update. While the rank one update seems to be simple and fast enough, it has a few limitations. It does not preserve the positive definiteness of the Hessian/Invert Hessian matrices ($H$ or $B$). If the denominators $||\delta||^2$, or $||\gamma||^2$ or $(\gamma - B\delta)^T \delta$ are too small (or even worse, zero) the updates are unstable, and in most cases it does not preserve Symmetry (except for the SR1 formula).

In an obscure analysis, Davidon justifies the use of a rank-two update in his derivation of the his (7.3) equation (which is at the center of the DFP algorithm). That is, the addition of two rank-one updates in his equation (7.3) he writes

$$(7.3) \quad h^{\nu \mu} - \frac{t^\nu t^\mu}{t_0} + \frac{\lambda}{g_{ss}} s^\mu s^\nu \rightarrow h^{\nu \mu}.$$  

We will not bother explaining the different symbols here. This equation was taken by Fletcher and Powell and implemented in a successful algorithm that carries the name DFP honoring the authors initials. The DFP algorithm gave birth soon after to the BFGS algorithm that we describe later on these notes. Next, we make a brief description of the DFP algorithm.

**A Rank Two and Up Updates:**

Rank two updates are generated with the formula

$$H_+ = H + auu^T + bvv^T.$$  

One idea that could have been motivated the optimization comunity to use rank updates is that any positive definite matrix $A$ in $\mathbb{R}^{n \times n}$ matrix can be written as as

$$A = \sum_{i=1}^{n} \lambda_i x_i x_i^T.$$
where \( \lambda_i \) is an eigenvalue and \( x_i \) is a normalized eigenvector (this is a consequence of the more general Singular Value Decomposition (SVD) theorem of linear algebra.

We can see the matrix \( A \) as a rank-n update where each rank is recovered in a one-by-one sequence of rank-one updates.

The DFP algorithm. A Rank Two Update:

Let us assume that the update of the Hessian (or its inverse) is given as rank-two update of the form

\[
H_+ = H + auu^T + bvv^T.
\]

for two unknown vectors \( u \) and \( v \) and scalars \( a \) and \( b \).

Then, from the secant condition \( H_+ \gamma = \delta \) we find

\[
H_+ \gamma = H \gamma + auu^T \gamma + bvv^T \gamma = H \gamma + a(u^T \gamma)u + b(v^T \gamma)v = \delta.
\]

When using the one-rank update, we were forced (after acknowledging the secant condition) to pick \( u = \delta - H \gamma \). We now have two degrees of freedom (in terms of vectors) and what Davidon chose using a heuristic (and messy) approach was to decouple the difference above by distributing the difference among the vectors \( u \) and \( v \). That is, in the equation above he chose \( u = \delta \) and \( v = -H \gamma \), and conveniently force the scalars \( a \) and \( b \) such that \( a(u^T \gamma) = 1 \), and \( b(v^T \gamma) = 1 \) to get his DFP formula

\[
H_{DFP}^{k+1} = H + \frac{\delta \delta^T}{\delta^T \gamma} - \frac{H \gamma \gamma^T H}{\gamma^T H \gamma}.
\]

with notation \( H_+ = H_{DFP}^{k+1} \). What is, to us, very interesting is that more than 20 years after, Greenstad, showed that this chosen Hessian \( H_+ \) is the closest in a weighted Frobenius norm to the previous \( H \) Hessian and this makes of this new Hessian a smooth and unique transition.

It is obvious that, since \( H \) is symmetric, then \( H_+ \) is also symmetric. Also, if a positive definite matrix is symmetric, its square root is also symmetric. Let us verify that if the initial matrix \( H \) is positive definite, then all matrices after that, satisfying the algorithm 6.4.0 should be positive definite as well. If \( H \) is positive definite then its inverse \( H = B \) is positive definite as well. Now, by induction, let us assume that \( H \) is positive definite and pick any \( x \in \mathbb{R}^n \). Then

\[
\langle x, H_+ x \rangle = \langle x, Hx \rangle + \frac{\langle x, \delta \delta^T x \rangle}{\langle \delta, \gamma \rangle} - \frac{\langle x, H \gamma \gamma^T Hx \rangle}{\langle \gamma, H \gamma \rangle}.
\]

\[
= \langle x, Hx \rangle + \frac{\langle \delta, x \rangle^2}{\langle \delta, \gamma \rangle} - \frac{\langle x, H \gamma \gamma^T Hx \rangle}{\langle \gamma, H \gamma \rangle}.
\]

\(^{19}\)http://en.wikipedia.org/wiki/Singular_value_decomposition
To simplify, let us redefine
\[ a = H^{1/2}x, \quad b = H^{1/2} \gamma \]

Then
\[
\begin{align*}
\langle x, Hx \rangle &= x^T H^T x = x^T (H^{1/2})^T (H^{1/2}) x = \langle a, a \rangle, \\
\langle x, H \gamma \gamma^T Hx \rangle &= x^T H^T \gamma \gamma^T H x = \langle a^T b, b^T a \rangle = (a, b)^2 \\
\langle \gamma, H \gamma \rangle &= \gamma^T H^{1/2} H^{1/2} \gamma^T a = \langle b, b \rangle.
\end{align*}
\]

from which we find
\[
\langle x, H_+ x \rangle = \langle a, a \rangle - \frac{(a, b)^2}{\langle b, b \rangle} + \frac{(\delta, x)^2}{\langle \delta, \gamma \rangle} = \frac{(a, a)(b, b) - (a, b)^2}{\langle b, b \rangle} + \frac{(\delta, x)^2}{\langle \delta, \gamma \rangle}
\]

The first term is no negative due to the Cauchy-Swartz inequality
\[ |\langle a, b \rangle| \leq \|a\| \|b\|. \]

We prove that the second term no negative. It is obvious that the numerator (which is squared) is no negative. Let us check the denominator. That is
\[ \langle \delta, \gamma \rangle = \langle \delta, g_{k+1} \rangle - \langle \delta, g \rangle. \]

We first show that \[ \langle \delta, g_{k+1} \rangle = 0, \] and than the second term (including the sign) is positive.

Let us define a function \( g(\alpha) \) as \footnote{\( g \) is not bold, so it is not to be confused with the gradient \( g \).}
\[ g(\alpha) = f(x + \alpha \delta) \]

From the line search in Algorithm 6.4.0 we know that \( g'(\alpha)|_{\alpha=\alpha} = 0 \). That is
\[ \nabla f(x_{k+1})^T \delta = 0, \]

but this expression is
\[ \langle \delta, g_{k+1} \rangle = 0. \]

Finally, since from the iterative step on the Algorithm 6.4.0,
\[ \delta = -Hg, \]
then
\[-\langle \delta, g \rangle = -\langle -Hg, g \rangle > 0\]
since $H$ is positive definite.

Then we showed that $H_+$ is positive definite and $g$ is non-zero (if $g$ is zero the algorithm had already finished.)

We provide the following interpretation to the DFP formula 6.4.0. The residual vector $\delta - H\gamma$ which, if zero, provides an exact solution of the secant condition for the Hessian, is splitted in two vectors $\delta$ and $-H\gamma$. The rank of the matrix $H$ is $n$ (since it is positive definite). The one-rank update $-H\gamma$ subtract a rank from $H$ while the other one-rank update along $\delta$ restores the rank to leave $H$ as an $n$-rank matrix. Chu, Funderlic and Golub show a formal proof of the two DFP updating steps as subtracting and restoring the rank on $H$. If the final point is at the local minimum, then $g_{k+1} = 0$ (this is the gradient at the minimum point) and we have

\[
\delta = -H \nabla f(x_{k+1}),
\]

which is the exact secant condition 6.2.16 (there $H_0^{-1}$, here $H$). What this is telling us is that, if the objective function is quadratic then the Hessian iterations should converge to the true Hessian (or inverse Hessian if that is the model being used). If the gradient $g_{k+1}$ is non-zero then we are not done and probably the directions $-H\gamma$ and $\delta$ are not aligned. This two directions (which are the reason for the rank-two update) represent the updated gradient direction and the search direction $\delta$, and their difference is an estimate of the error on direction search.

The DFP algorithm is implemented as follows:

**Algorithm 6.4.0**

- **Initial step:**
  Provide an initial Hessian $B_0$ which is positive definite. This could be, for example, the identity matrix $I$, a diagonal matrix with the true Hessian values, a scaled version of the identity, etc. Define an initial point $x_0$. Set $k = 0$.

- **Iterative Step:** Find the gradient

\[g_k = \nabla f(x_k)\]

- Find $H_k = B_k^{-1}$. This can be done through Cholesky decomposition if the initial matrix is not simple. In general the starting matrix is chosen to be the identity or a scalar multiple of it.

- Set the direction search

\[p_k = -H_k g_k\]
• Do the line search. That is minimize \( f(x_k + \alpha p_k) \), with \( \alpha \geq 0 \) to obtain \( \alpha_k \). Set

\[
x_{k+1} = x_k + \alpha_k p_k \ , \quad g_{k+1} = \nabla f(x_{k+1}) \ , \quad \delta_k = x_{k+1} - x_k.
\]

At this moment we can compute the gradient difference. That is, let us set

• **Update step:**

\[
\gamma_k = g_{k+1} - g_k
\]

and finally

• The new Hessian \( H_{k+1} \) is

\[
H_{k+1} = H_k + \frac{\delta_k \delta_k^T}{\gamma_k} - \frac{H_k \gamma_k \gamma_k^T H_k^T}{\gamma_k^T H_k \gamma_k}.
\] (6.4.36)

According to Fletcher the DFP algorithm works well in practice and has been used widely.

“...It proved to be much more efficient than the SD method, and also somewhat more efficient than the CG methods introduced in 1964 onwards. Early implementations attempted to carry out fairly accurate line searches. The coming of low accuracy line searches in 1970 however showed the DFP formula in a less satisfactory light than other formulae which were being introduced, and currently the DFP method is no longer preferred. However the method has a number of important properties as follows:

(1) for **quadratic functions** (with exact line searches)

(i) terminates in at most \( n \) iterations with \( H_{k+1} = H^{-1} \)

(ii) previous quasi-Newton conditions are preserved;

(iii) generates conjugate directions, and CG when \( H_0 = I \).

(2) for **general functions**

(i) preserves positive definite \( H_k \) matrices -hence the descent property holds;

(ii) requires \( 3n^2 + \mathcal{O}(n) \) multiplications per iteration;

(iii) superlinear rate of convergence

(iv) global convergence for strictly convex function (with exact line searches).

We do not provide for all the proofs of Fletcher’s statements but some we have already been outlined above. The proofs on convergence and the maximum number of iterations can be found in Fletcher’s book or somewhere else. We are interested in examine item (1)-(iii). “generates conjugate directions, and CG when \( H_0 = I \)”.
It is by many authors listed in Papakonstantinou’s thesis Ph. D. thesis 21 that the DFP method suffer from a conditioning problem. That is, as the number of iterations increase, the eigenvalues become smaller tending to zero. This approach to singularity is known as the conditioning problem. This is the main reason why this method did not last long and the BFGS method appeared. We describe the BFGS algorithm next.

The BFGS Algorithm:

Let us recall the secant condition equation:

\[ \delta = H \gamma, \]

which can be written as

\[ \gamma = H^{-1} \delta = B \delta \]

this is secant condition 6.4.29.

Since \( H \) is positive definite symmetric its inverse \( H^{-1} \) is positive definite symmetric as well, and so, all the analysis done in the matrix \( H \) can be copied exactly by changing \( H \) for \( B \) and interchanging the role of \( \gamma \) and \( \delta \). That is, we can rewrite equation 6.4.36 as

\[
B_{k+1} = B_k + \frac{\gamma_k \gamma_k^T}{\gamma_k^T \delta_k} - \frac{B_k \delta_k \delta_k^T B_k^T}{\delta_k^T B_k \delta_k}. \tag{6.4.37}
\]

This is the BFGS 22 algorithm as found by Broyden, Fletcher, Goldfarb, and Shanno in 1970. The most successful algorithm to find minimum in unconstrained minimization problems on multi-dimensional spaces. It is interesting that while the way to derive the BFGS algorithm here was proposed by Fletcher, each of the other three authors (Broyden, Goldfarb, and Shanno) developed the same equation using different paths (and the four of them in the same year of 1970). These approaches are explained in Papakonstantinou’s Ph.D. thesis. We just want to point out that Goldfarb used the weighted Frobenius norm to minimize the transition between the previous and the current Hessian to come up with the BFGS algorithm, while Broyden and Shanno used algebraic rules and heuristic (like the rank-two update) methods to arrive to the BFGS formula.

In the sense of Fletcher we can say that the BFGS algorithm is the dual of the DFP algorithm. What makes this algorithm more stable is the fact that working directly on \( B \) instead of its inverse \( H \) will eliminate the conditioning problem of the DFP method. That is the eigenvalues will no longer decrease toward 0. Fletcher proposed the idea of bounding the eigenvalues from above (since now the eigenvalues could become very large).

---

In addition the BFGS algorithm retains all good properties of the DFP listed above. The algorithmic implementation is the same (using $B$ instead of $H$) under algorithm 6.4.0. Note that since the computation for the direction search is carried out by the equation

$$ p_k = -H_k g_k = -B_k^{-1} g_k $$

we need to invert the matrix $B_k$ at this step. For this inversion we use the Sherman-Morrison formula 6.4.31 acting on the BFGS equation 6.4.37. Let us do the algebra. To simplify further this expression let us redefine

$$ a = \frac{\gamma_k}{\sqrt{\gamma_k^T \delta_k}}, \quad b = i \frac{B_k \delta_k}{\sqrt{\delta_k^T B_k \delta_k}}, $$

with $i = \sqrt{-1}$. We rewrite the BFGS equation 6.4.37 as

$$ B_{k+1} = B_k + aa^T + bb^T, $$

and use the Sherman-Morrison formula 6.4.31 twice. That is, let us call

$$ A = B_k + aa^T \quad B_{k+1} = A + bb^T $$

(6.4.38)

and so

$$ A^{-1} = B_k^{-1} - \frac{B_k^{-1} aa^T B_k^{-1}}{1 + a^T B_k^{-1} a}. $$

Now,

$$ B_k^{-1} a = B_k^{-1} \frac{\gamma_k}{\sqrt{\gamma_k^T \delta_k}} $$

and

$$ a^T B_k^{-1} = \frac{1}{\sqrt{\gamma_k^T \delta_k}} \gamma_k^T B_k^{-1} $$

from which

$$ B_k^{-1} aa^T B_k^{-1} = \frac{B_k^{-1} \gamma_k \gamma_k^T B_k^{-1}}{\gamma_k^T \delta_k}. $$

Now

$$ a^T B_k^{-1} a = \frac{\gamma_k^T B_k^{-1} \gamma_k}{\gamma_k^T \delta_k} \quad 1 + a^T B_k^{-1} a = \frac{\gamma_k^T \delta_k + \gamma_k^T B_k^{-1} \gamma_k}{\gamma_k^T \delta_k} $$
and so
\[ A^{-1} = B_k^{-1} - \frac{B_k^{-1} \gamma_k \gamma_k^T B_k^{-1}}{\gamma_k^T \delta_k + \gamma_k^T B_k^{-1} \gamma_k} = B_k^{-1} - \frac{B_k^{-1} \gamma_k \gamma_k^T B_k^{-1}}{\lambda} \]  
(6.4.39)

with \( \lambda = \gamma_k^T \delta_k + \gamma_k^T B_k^{-1} \gamma_k \).

We apply now the Sherman-Morrison condition to the second equation 6.4.38. That is,
\[ B_{k+1}^{-1} = A^{-1} - \frac{A^{-1} b b^T A^{-1}}{1 + b^T A^{-1} b}. \]  
(6.4.40)

Let us do this by pieces. We simplify the denominator \( 1 + b^T A^{-1} b \).

\[
b^T A^{-1} b = -\frac{\delta_k^T B_k A^{-1} B_k \delta_k}{\delta_k^T B_k \delta_k}
= -\frac{\delta_k^T B_k B_k \delta_k}{\delta_k^T B_k \delta_k} + \frac{\delta_k^T B_k B_k \gamma_k \gamma_k^T B_k^{-1} B_k \delta_k}{\delta_k^T B_k \delta_k \lambda}
= -1 + \frac{(\gamma_k^T \delta_k)^2}{\delta_k^T B_k \delta_k \lambda}
\]

So the denominator becomes
\[
1 + b^T A^{-1} b = \frac{(\gamma_k^T \delta_k)^2}{\lambda \delta_k^T B_k \delta_k}
\]

Let us now evaluate the numerator \( A^{-1} b b^T A^{-1} \).

\[
A^{-1} b b^T A^{-1} = -\frac{1}{\delta_k^T B_k \delta_k} \left( B_k^{-1} - \frac{B_k^{-1} \gamma_k \gamma_k^T B_k^{-1}}{\lambda} \right) B_k \delta_k \delta_k^T B_k \left( B_k^{-1} - \frac{B_k^{-1} \gamma_k \gamma_k^T B_k^{-1}}{\lambda} \right)
= -\frac{1}{\delta_k^T B_k \delta_k} \left( I - \frac{B_k^{-1} \gamma_k \gamma_k^T B_k^{-1}}{\lambda} \delta_k \delta_k^T \right) \left( I - \gamma_k \gamma_k^T B_k^{-1} \right)
= -\frac{1}{\delta_k^T B_k \delta_k} \left( \delta_k \delta_k^T - \frac{(\gamma_k^T \delta_k) B_k^{-1} \gamma_k \delta_k}{\lambda} \right) \left( I - \frac{\gamma_k \gamma_k^T B_k^{-1}}{\lambda} \right)
\]

Now the fraction (numerator/denominator) turns out to be
\[
\frac{A^{-1} b b^T A^{-1}}{1 + b^T A^{-1} b} = \frac{\lambda}{(\gamma_k^T \delta_k)^2} \left( \delta_k \delta_k^T - \frac{(\gamma_k^T \delta_k) B_k^{-1} \gamma_k \delta_k}{\lambda} \right) \left( I - \frac{\gamma_k \gamma_k^T B_k^{-1}}{\lambda} \right)
= \frac{\lambda \delta_k \delta_k^T}{(\gamma_k^T \delta_k)^2} - \frac{B_k^{-1} \gamma_k \delta_k}{(\gamma_k^T \delta_k)^2} - \frac{\delta_k \delta_k^T \gamma_k \gamma_k^T B_k^{-1} \gamma_k \delta_k}{\lambda (\gamma_k^T \delta_k)^2} + \frac{(\gamma_k^T \delta_k)^2 B_k^{-1} \gamma_k \gamma_k^T B_k^{-1}}{\lambda (\gamma_k^T \delta_k)^2}
= \frac{\lambda \delta_k \delta_k^T}{(\gamma_k^T \delta_k)^2} - \frac{B_k^{-1} \gamma_k \delta_k}{(\gamma_k^T \delta_k)^2} - \frac{(\gamma_k^T \delta_k)^2 B_k^{-1} \gamma_k \gamma_k^T B_k^{-1}}{\lambda (\gamma_k^T \delta_k)^2} + \frac{B_k^{-1} \gamma_k \gamma_k^T B_k^{-1}}{\lambda}
\]
We now evaluate equation 6.4.40 with the use of equation 6.4.39 and the previous result. This is

\[
B_{k+1}^{-1} = B_k^{-1} - \frac{B_k^{-1} \gamma_k \gamma_k^T B_k^{-1}}{\lambda} + \frac{\lambda \delta_k \delta_k^T}{(\gamma_k^T \delta_k)^2} - \frac{B_k^{-1} \gamma_k \delta_k^T}{\gamma_k^T \delta_k} - \frac{\delta_k \gamma_k B_k^{-1}}{\gamma_k^T \delta_k} + \frac{B_k^{-1} \gamma_k \gamma_k^T B_k^{-1}}{\lambda}
\]

\[
= B_k^{-1} + \frac{\lambda \delta_k \delta_k^T}{(\gamma_k^T \delta_k)^2} - \frac{B_k^{-1} \gamma_k \delta_k^T}{\gamma_k^T \delta_k} - \frac{\delta_k \gamma_k B_k^{-1}}{\gamma_k^T \delta_k}
\]

This equation appears in the Wikipedia site as a more efficient way to compute the BFGS inverse. However there is another form of the equation commonly posted in the literature. Let us expand \( \lambda \) in the previous equation and find

\[
B_{k+1}^{-1} = B_k^{-1} + \frac{(\gamma_k^T \delta_k + \gamma_k^T B_k^{-1} \gamma_k) \delta_k \delta_k^T}{(\gamma_k^T \delta_k)^2} - \frac{B_k^{-1} \gamma_k \delta_k^T}{\gamma_k^T \delta_k} - \frac{\delta_k \gamma_k B_k^{-1}}{\gamma_k^T \delta_k}
\] (6.4.41)

and expanding and rearranging,

\[
B_{k+1}^{-1} = B_k^{-1} + \frac{\delta_k \delta_k^T}{\gamma_k^T \delta_k} + \frac{(\gamma_k^T B_k^{-1} \gamma_k) \delta_k \delta_k^T}{(\gamma_k^T \delta_k)^2} - \frac{B_k^{-1} \gamma_k \delta_k^T}{\gamma_k^T \delta_k} - \frac{\delta_k \gamma_k B_k^{-1}}{\gamma_k^T \delta_k}
\]

\[
= B_k^{-1} + \frac{\delta_k (\gamma_k^T B_k^{-1} \gamma_k) \delta_k^T}{(\gamma_k^T \delta_k)^2} - \frac{B_k^{-1} \gamma_k \delta_k^T}{\gamma_k^T \delta_k} - \frac{\delta_k \gamma_k B_k^{-1}}{\gamma_k^T \delta_k} + \frac{\delta_k \delta_k^T}{\gamma_k^T \delta_k}
\]

\[
= \left( I - \frac{\delta_k \gamma_k^T}{\gamma_k^T \delta_k} \right) B_k^{-1} \left( I - \frac{\gamma_k \delta_k^T}{\gamma_k^T \delta_k} \right) + \frac{\delta_k \delta_k^T}{\gamma_k^T \delta_k}
\]

Now, recall that \( B^{-1} = H \), from which, writing the previous equation in terms of \( H \) we find

\[
H_{k+1} = \left( I - \frac{\delta_k \gamma_k^T}{\gamma_k^T \delta_k} \right) H_k \left( I - \frac{\gamma_k \delta_k^T}{\gamma_k^T \delta_k} \right) + \frac{\delta_k \delta_k^T}{\gamma_k^T \delta_k}
\] (6.4.42)

\[
= H_k + \frac{(\gamma_k^T \delta_k + \gamma_k^T H_k^{-1} \gamma_k) \delta_k \delta_k^T}{(\gamma_k^T \delta_k)^2} - \frac{H_k \gamma_k \delta_k^T}{\gamma_k^T \delta_k} + \frac{\delta_k \gamma_k^T H_k}{\gamma_k^T \delta_k}
\] (6.4.43)

This is a form that appears commonly in the literature with the notation

\[ y_k = \gamma_k \quad s_k = \delta_k \]

Another common form is written from equation 6.4.41, that is

\[
H_{k+1} = H_k + \frac{(\gamma_k^T \delta_k + \gamma_k^T H_k^{-1} \gamma_k) \delta_k \delta_k^T}{(\gamma_k^T \delta_k)^2} - \frac{H_k \gamma_k \delta_k^T}{\gamma_k^T \delta_k} + \frac{\delta_k \gamma_k^T H_k}{\gamma_k^T \delta_k}
\]

We call this the BFGS update for \( H \) and removing the subindex \( k \) from the right hand side we find

\[
\frac{y_k}{s_k} = \gamma_k \quad s_k = \delta_k
\]

\[ \text{Search for BFGS in Wikipedia} \]
\[ H_{BFGS}^{k+1} = H + \frac{(\gamma^T \delta + \gamma^T H \gamma) \delta \delta^T}{(\gamma^T \delta)^2} - \frac{H \gamma \delta^T + \delta \gamma^T}{\gamma^T \delta} \]  

(6.4.44)

Note that when working with this explicit equations, there is no need to invert matrices. The inverse is computed by multiplications based on the previous inverse. If \( B_0 \) is the identity there is no work to do in its inverse. If \( B_0 \) is a diagonal matrix, its inverse is simple, otherwise the first inverse can be computed by using Cholesky factorization.

The Hestenes-Stiefel formula as a quasi-Newton method

As we indicated in the non–linear CG section 5.4 the Hestenes-Stiefel formula is a particular case of the quasi-Newton formula 6.4.44. Let us show that this. We start with equation 6.4.44 which we rewrite next, with \( H = I \).

\[ H_{BFGS}^{k+1} = I + \frac{(\gamma^T \delta + \gamma^T \gamma) \delta \delta^T}{(\gamma^T \delta)^2} - \frac{\gamma \delta^T + \delta \gamma^T}{\gamma^T \delta} \]  

(6.4.45)

where, using the notation of the non–linear CG session,

\[ \gamma = g_k - g_{k-1} \]

\[ \delta = x_k - x_{k-1} = -\alpha_{k-1} p_{k-1}, \]

so, from the line search, we know that \( p_{k-1} \) is orthogonal to \( g_k \) and so \( \delta \) is orthogonal to \( g_k \), we post-multiply equation 6.4.45 by \(-g_k\) and find

\[ -H_{BFGS}^{k+1} g_k = -g_k + \frac{\gamma^T g_k}{\gamma^T \delta} \]

\[ = -g_k + \frac{\gamma^T g_k}{\gamma^T \delta} \]

\[ = -g_k + \frac{(g_k - g_{k-1})^T g_k}{(g_k - g_{k-1})^T (-\alpha_{k-1} p_{k-1})} (-\alpha_{k-1} p_{k-1}) \]

\[ = -g_k + \frac{(g_k - g_{k-1})^T g_k}{(g_k - g_{k-1})^T p_{k-1}} p_{k-1} \]

The negative right hand side of this equation corresponds exactly with equation 5.1.4. So the Hessian times the gradient

\[ H_{BFGS}^{k+1} g_k \]

does the same update as the Hestenes-Stiefel formula, and we can conclude that the non–linear conjugate method is a particular case of the quasi–Newton BFGS method. Note that equation 5.1.4 is the most general form for \( \beta_k \) from which all other methods are derived.
6.4 Quasi-Newton methods revisited

The Broyden Family

Let us make a summary of the two main updates of the inverse Hessian matrix base on Davidon’s algorithm. These are the DFP formula 6.4.35 and the 6.4.44 formula

\[
H^{(k+1)}_{\text{DFP}} = H + \frac{\delta \delta^T}{\gamma^T \delta} - \frac{H \gamma \gamma^T H}{\gamma^T H \gamma}
\]

\[
H^{k+1}_{\text{BFGS}} = H + \frac{(\gamma^T \delta + \gamma^T H \gamma) \delta \delta^T}{(\gamma^T \delta)^2} - \frac{H \gamma \delta^T + \delta \gamma^T H}{\gamma^T \delta}
\]

The dual updates in terms of matrices Hessian matrices \(B\) and not its inverses are obtained by inter-changing \(H\) for \(B\), and \(\delta\) for \(\gamma\) in the previous couple. That is,

\[
B^{(k+1)}_{\text{DFP}} = B + \frac{\gamma \gamma^T}{\gamma^T \delta} - \frac{B \delta \delta^T B}{\delta^T B \delta}
\]

\[
B^{k+1}_{\text{BFGS}} = B + \frac{(\delta^T \gamma + \delta^T B \delta) \gamma \gamma^T}{(\gamma^T \delta)^2} - \frac{B \delta \gamma^T + \gamma \delta^T B}{\gamma^T \delta}
\]

The Broyden family is a linear interpolation (\(\phi \in [0, 1]\)) between the two representations of the inverse Hessian updates \(H^{(k+1)}_{\text{DFP}}\) and \(H^{k+1}_{\text{BFGS}}\). That is, for a real number \(\phi\),

\[
H^{k+1}_{\phi} = (1 - \phi)H^{(k+1)}_{\text{DFP}} + \phi H^{k+1}_{\text{BFGS}}.
\]

Note that for \(H^0_{k+1} = H_{DFP}^{k+1}\), and \(H_1 = H_{BFGS}^{k+1}\). The values \(\phi = 0\) and \(\phi = 1\) are the end points of the segment of updates. Some interest values other than 1 and 0 are used to generate other updates. For example, let us assume \(\phi = \gamma^T \delta / \lambda\), then with \(\lambda = \gamma^T \delta - \gamma^T H \gamma\).

\[
H_{\phi} = \left(1 - \frac{\gamma^T \delta}{\lambda}\right) \left(H + \frac{\delta \delta^T}{\gamma^T \delta} - \frac{H \gamma \gamma^T H}{\gamma^T H \gamma}\right)
\]

\[
+ \frac{\gamma^T \delta}{\lambda} \left(H + \frac{(\gamma^T \delta + \gamma^T H \gamma) \delta \delta^T}{(\gamma^T \delta)^2} - \frac{H \delta \gamma^T + \gamma \delta^T H}{\gamma^T \delta}\right)
\]

\[
= H + \frac{\delta \delta^T}{\gamma^T \delta} - \frac{H \gamma \gamma^T H}{\gamma^T H \gamma} - \frac{\delta \delta^T}{\gamma^T \delta} + \frac{(\gamma^T \delta)(H \gamma \gamma^T H)}{\lambda \gamma \gamma^T \gamma}
\]

\[
+ \frac{(\gamma^T \delta + \gamma^T H \gamma) \delta \delta^T}{\lambda} - \frac{H \delta \gamma^T + \gamma \delta^T H}{\gamma^T \delta}
\]

\[
= H + \frac{\lambda(\gamma^T H \gamma) \delta \delta^T - \lambda(\gamma^T \delta)H \gamma \gamma^T H - (\gamma^T \delta)(\gamma^T H \gamma) \delta \delta^T + (\gamma^T \delta)^2 H \gamma \gamma^T H}{\lambda(\gamma^T \delta)(\gamma^T H \gamma)}
\]

\[
+ \frac{(\gamma^T \delta + \gamma^T H \gamma)(\gamma^T H \gamma) \delta \delta^T - (\gamma^T \delta)(\gamma^T H \gamma)(H \delta \gamma^T + \gamma \delta^T H)}{\lambda(\gamma^T \delta)(\gamma^T H \gamma)}
\]
Let us simplify this expression by selecting matrices (except for the first $H$) and finding their coefficients. That is
\[
\delta \delta^T \Rightarrow (\gamma^T \delta)(\gamma^T H \gamma) - (\gamma^T H \gamma)^2 - (\gamma^T \delta)(\gamma^T H \gamma) + (\gamma^T \delta)^2 H \gamma
\]
\[
H \gamma \gamma^T H \Rightarrow - (\gamma^T \delta)^2 + (\gamma^T \delta)(\gamma^T H \gamma + (\gamma^T \delta)^2).
\]
\[
H \delta \gamma^T + \gamma \delta^T H \Rightarrow - (\gamma^T \delta) \gamma^T H \gamma
\]
from which find
\[
H_\phi = H + \delta \delta^T + H \gamma \gamma^T H - H \delta \gamma^T - \gamma \delta^T H \frac{\lambda}{\gamma^T (\delta - H \gamma)} = H + (\delta - H \gamma)(\delta - H \gamma)^T / \gamma^T (\delta - H \gamma)
\]

If we observe Table 6.1 and find the dual operator of the Symmetric Rank-one (SR1) (third row). That is, inter-changing $B$ for $H$ and $\gamma$ for $\delta$ we see that the matrix $H_\phi$ for $\phi = \gamma^T \delta / (\gamma^T \delta - \gamma^T H \gamma)$, is the SR1 version on the matrices $H$ (the Hessian inverses). So even rank-one updates are included in the Broyden family $\{H_\phi\}$.

The L-BFGS Algorithm:
Large scale problems can not been solved using regular BFGS algorithms. Let us make an example on a geophysical problem. We assume that a velocity model for (FWI) is a 3D volume with 10 Km on each side (which is no too large for current seismic exploration surveys). We choose the grid spacing to be 10 meters along the vertical and 20 meters along each horizontal direction. That will give us an estimate of $1000 \times 500 \times 500 = 250000000$ grid points. This is the dimension of the space we are working on. The Hessian should store $n^2$ floats where $n$ is the dimension of the space. That is, the memory of Hessian is more than 222 petabytes. So, for today’s technology we can not think about a whole Hessian, no matter the method. The L-BFGS (Limited Memory BFGS) offers the possibility of computing only parts of the Hessian. As far as we know, the idea of using a limited storage algorithm on top of the BFGS method was started by Jorge Nocedal. 24 In 1980. In addition the paper On the Limited Memory BFGS Method For Large Scale Optimization 25 by Dong C. Liu and Jorge Nocedal show the refined development known today and comparison the method with other known algorithms in the field. The 1999 book 26 by Nocedal and Wright is a good reference, not just for L-BFGS, but for general constrained and unconstrained optimization techniques.

Let us see how the method works. We start with equation 6.4.42, and introduce the notation $V_k$ used by Nocedal. That is,
\[
V_k = \left( I - \frac{\gamma_k \delta_k^T}{\gamma_k^T \delta_k} \right) \quad \rho_k = \frac{1}{\gamma_k^T \delta_k}
\]
\[
H_{k+1} = V_k^T H_k V_k + \rho_k \delta_k \delta_k^T
\]

24http://folk.uib.no/ssu029/Pdf_file/Nocedal80.pdf
25http://www.ece.northwestern.edu/~nocedal/PDFfiles/limited-memory.pdf
We unroll recursively the representation for the Hessian \( m \) times. That is,

\[
H_{k+1} = V_k^T H_k V_k + \rho_k \delta_k \delta_k^T
\]

\[
= V_k^T (V_{k-1}^T H_{k-1} V_{k-1} + \rho_{k-1} s_{k-1} s_{k-1}^T) V_k + \rho \delta_k \delta_k^T
\]

\[
= V_k^T V_{k-1}^T H_{k-1} V_{k-1} V_k + \rho_{k-1} s_{k-1} s_{k-1}^T V_k + \rho \delta_k \delta_k^T
\]

and if we keep iterating \( m \) times, we find

\[
H_{k+1} = (V_k^T \cdots V_{k-m}^T) H_{k-m} (V_{k-m} \cdots V_k)
\]

\[
+ \rho_{k-m} (V_k^T \cdots V_{k-m+1}^T) s_{k-m} s_{k-m}^T (V_{k-m+1} \cdots V_k)
\]

\[
+ \rho_{k-m+1} (V_k^T \cdots V_{k-m+2}^T) s_{k-m+1} s_{k-m+1}^T (V_{k-m+2} \cdots V_k)
\]

\[\vdots\]

\[
+ \rho \delta_k \delta_k^T.
\]

where for \( m = 0 \) is the full Hessian computation and for \( m = k + 1 \) the sequence unrolled down to the \( H_0 \). In practice \( m \) starts low and grows up to certain limit where it stays fixed. This rolling is much like a rolling window filter (like a median filter or an Automatic Gain Control filter) of length \( m \). Ultimately what we need to compute is the search direction \( p_{k+1} = -H_{k+1} \nabla f_{k+1} \). The idea is to perform this computation without explicit knowledge of \( H_{k+1} \). We can start with \( H_{k-m} \) to be a diagonal matrix (or the identity) which only requires the storage of a gradient, and store the \( m \) vectors \( \gamma_k \) and \( \delta_k \) required to do the computation. The memory storage will be \( O(n m) \) instead of \( O(n^2) \) where \( m \ll n \).

Although the algorithm seems to be a two nested (recursive) loops, one to compute the products and the other to compute the sums, the actual published algorithm shows that while there are two loops they can be implemented in a no nested way, by computing the right (to the \( H \)) side of the products and then in the second loop the left side of the products and together with the sum.

Here is the algorithm to compute the next search direction \( p_{k+1} \).

Let us verify why this algorithm indeed computes the search direction. Assume that we start with the gradient \( g_0 = \nabla f_k \) and then after iteration \( i \) \(^{27} \) from the first loop

\[ g_i = g_i - \rho_{k-i} \gamma_{k-i} \delta_{k-i}^T g_{i-1} = V_{k-i} g_{i-1} = \cdots = V_{k-i} V_{k-i+1} \cdots V_k g_0 \]

and then

\[ r_m = H_{k-m} g_m = H_{k-m} (V_{k-m} V_{k-m+1} \cdots V_k) g_0 \]

At this point the \( m \) \( r_i \) vectors, of the size of the gradient have all the right products including the Hessian \( H_{k-m} \).

\(^{27}\) we use indices for \( g_i \) to better explain, however the algorithm does not require to store more than \( g_i \) on memory, it needs to store \( \rho_i, \delta_i \) and \( y_i \).
\[ g_k = \nabla f_k \]

for (i = 1 \ldots m) do

\[ \alpha_i = \rho_{k-i} \delta_{k-i}^T g_k \]
\[ g_k = g_k - \alpha_i \gamma_{k-i} \]

end for

\[ H_k = \frac{\gamma_{k-1}^T \delta_{k-1}}{\gamma_{k-1}^T \gamma_{k-1}} I \]

\[ r_k = -H_{k-m} g_k \]

for (i = m \ldots 1) do

\[ \beta = \rho_{k-i} \gamma_{k-i}^T r_k \]
\[ r_k = r_k + \delta_{k-i}(\alpha_i - \beta) \]

end for

return \( r_k \)

Figure 6.10: The Algorithm for the loop to compute the next search direction vector \( r_k \) in the L-BFGS method. We assume that the differential gradient vectors \( \gamma_i = g_i - g_{i-1} \) are stored in disk, together with the model differentials \( \delta_i = x_i - x_{i-1} \), for all \( i = k - m, \ldots, k - 1 \). Similarly the scalar values \( \rho_i = 1/\gamma_i^T \delta_i \) are stored, for the same values of \( i \). Now we want to compute the next gradient \( g_k \) and the next residual direction \( r_k \). The scalars \( \alpha_i \) and \( \beta \) are computed on the fly. The vector \( \alpha \) of components \( \alpha_i \) is created in the first loop and saved to be used in the second loop. The scalar \( \beta \) is clobbered on each iteration of the second loop. Note that for the search direction a matrix vector multiplication is required in between the two loops. This matrix is the (inverse) Hessian matrix that is assumed to be diagonal, otherwise the storage would be too large, is computed from the ratio of inner product between the differential gradient and differential model, and the differential model square (in norm \( \ell - 2 \)).
Chapter 7

Seismic Inversion in the Generalized Least Square (GLS) context

We present the GLS problem in general and then apply the results to seismic inversion. Three types of inversion are examined:

- Born Inversion
- Kirchhoff Inversion
- FWI

We explain the concepts of

- The Resolution (Matrix) Operator.
- The Point Spread Function.

and produce interpretations of the Hessian matrix in the context of seismic inversion.

7.1 General Formulation

7.1.1 The Forward Modeling

While the theory is extended to general operators in infinity dimensional spaces we consider here finite dimensions and the operators are then matrices.

The idea of model resolution operator is as follows:

Let us assume a linear model that produces some data. That is

\[ d = Lm. \]  \hspace{1cm} (7.1.1)

where \( L \) is a linear operator, \( m \) is a model, and \( d \) is some data produced from the model into the linear operator \( L \).
We want to invert for the model in terms of the data. If $L^{-1}$ exists then

$$m = L^{-1}d$$

and plugging equation 7.1.1 into the previous equation we find

$$m = Im.$$  

Here $I$ is the identity matrix. This is an uninteresting problem. In general $L$ is not even an square matrix, and the best we can do is find a pseudo-inverse $L^+$ such that

$$m = L^+d,$$

is a good approximation for $m$. Plugging equation 7.1.1 into the previous equation we find

$$\overline{m} = L^+Lm = Rm$$

The matrix

$$R = L^+L$$

is known as the model resolution matrix. We believe the name comes from how well this operator resolves the model $m$. If it is the identity, it will resolve it exactly. Due to uncertainties in the modeling equations and in the data, this is never the case in practice.

### 7.1.2 The Adjoint

The adjoint operator is important in the solution of the least squares problem. In the least squares sense, let us define a cost function.

$$C(m) = \frac{1}{2}\|d - d_{obs}\|^2 = \|Lm - d_{obs}\|^2,$$  

(7.1.2)

where $d_{obs}$ stands for the observed data.

We can expand equation 7.1.2 using inner product notation as

$$2C(m) = \langle Lm - d_{obs}, Lm - d_{obs} \rangle$$

$$= \langle d_{obs}, d_{obs} \rangle - \langle Lm, d_{obs} \rangle - \langle d_{obs}, Lm \rangle + \langle Lm, Lm \rangle$$  

(7.1.3)
We want to compute the gradient $\nabla m C$. Let us first answer the following question: What is the gradient (Fréchet Derivative) $D$ of $-\langle L m , d_{\text{obs}} \rangle$? We assume that the reader does not have much knowledge on Hilbert spaces, other than the inner product is distributive (property that we used above), and that $D\langle f , g \rangle = \langle Df , g \rangle + \langle f , Dg \rangle$ as in regular products of functions. We further ask what is the gradient of $m$ with respect to $m$? This seems to be a delta function. For the discrete it is obvious that $\partial m_i / \partial m_j = \delta_{ij}$. Let us justify this for the continuum.

Here is the justification. Let us call $x$ the index that labels the elements $m(x)$.

$$m(x) = F(m) = \int dx' m(x') \delta(x - x').$$

where $F$ is the identity operator on $m$. Then

$$\frac{\delta m(x)}{\delta m(x')} = \frac{\delta F}{\delta m} = \delta(x - x')$$

We are ready to compute the gradient of $C(m)$ on equation 7.1.3

That is,

$$2\nabla m C = -\langle L \delta(x - x'), d_{\text{obs}} \rangle - \langle d_{\text{obs}}, L \delta(x - x') \rangle + \langle L \delta(x - x'), Lm \rangle + \langle Lm, L \delta(x - x') \rangle$$

$$= -\langle \delta(x - x'), L^* d_{\text{obs}} \rangle - \langle L^* d_{\text{obs}}, \delta(x - x') \rangle + \langle \delta(x - x'), L^* Lm' \rangle + \langle L^* Lm', \delta(x - x') \rangle$$

$$= -2L^* d_{\text{obs}} + 2L^* Lm'$$

after collapsing the Dirac delta distributions. Then

$$\nabla m C = -L^* d_{\text{obs}} + L^* Lm'$$

We find two conclusions here:

(i) The gradient is a linear function of the model. This is so, because the cost function is quadratic of the model.

(ii) Taking one more derivative will extract $L^* L$ which is the Hessian. The constant term is the adjoint (pointing in the opposite direction) operator acting on the observed data. The quadratic term is the curvature (departure from linear) or Hessian.

---

\[1\] In this chapter we use the symbol $L^*$ for adjoint instead of transposed $L^T$, since the term adjoint is more general and includes, for example, complex matrices for which the adjoint is the conjugate-transposed matrix.
CHAPTER 7. SEISMIC INVERSION IN THE GENERALIZED LEAST SQUARE (GLS) CONTEXT

Making the gradient equal to zero brings the equation

\[ L^* d_{obs} = L^* Lm' \]

We observe that in practice, migration assumes \( L^* L = I \) the identity, that is migration assumes that the operator \( L \) is unitary. and so \( m_I = L^* d_{obs} \) is migration. Inversion tries to invert for the Hessian \( L^* L \).

Here \( m_I \) is the migrated image from the data \( d_{obs} \). From equation 7.1.1 we then write

\[ m_I = L^* Lm = Rm \]

where now the model resolution matrix is \( R = L^* L \).

This matrix indicates how well the model \( m \) is resolved by the image \( m_I \). Still this is a poor resolved problem. The adjoint is fast to compute but it is far from achieving a good resolution. The Generalized Least Squares Problem (GLS) Menke \(^2\) is a standard way to proceed into better solutions. We will follow Menke’s work closely.

### 7.1.3 The Hessian

Let us assume further than the matrix \( L^* L \) has an inverse. Then

\[ \hat{m} = (L^* L)^{-1} L^* d_{obs} = H^{-1} m_I. \] \hspace{1cm} (7.1.4)

Here \( H = L^* L \) is the Hessian (see discussion in 7.1.2 of the cost function \( C(m) \). Now the pseudo-inverse operator is

\[ L^+ = (L^* L)^{-1} L^* \]

and the model resolution operator is

\[ R = L^+ L = (L^* L)^{-1} L^* L = H^{-1} H = I_m \] \hspace{1cm} (7.1.5)

again (assuming \( L^* L = H \) has an inverse )

So we see that the model resolution operator is the inverse Hessian times the Hessian.

We expect that \( \hat{m} \) is closer to \( m \) than \( m_I \). In this sense, the migrated image is “divided” by the Hessian to compensate for illumination and aperture limitations.

We can rewrite equation 7.1.4 as

\[ H \hat{m} = m_I \] \hspace{1cm} (7.1.6)

This equation yields the next concept.

\(^2\)http://www.ldeo.columbia.edu/users/menke/gls/menke_gls_paper.pdf
7.2 The Point Spread Function PSF

Let us recall the three variations of the model that we have encountered so far.

- \( m \) is the true model.
- \( m_I \) is the migrated image
- \( \hat{m} \) is the best (in least squares sense) inversion model.

There is not a unique definition of PSF. We show two definitions here. One is based on imaging (using the Hessian matrix as Valenciano does). The other is based on the Resolution matrix.

7.2.1 For Imaging

Let us rewrite equation (7.1.6).

\[
H\hat{m} = m_I \]

This equation shows that the Hessian \( H \) distorts the model \( \hat{m} \) into the image \( m_I \). In the next section 7.2.2 we show how this distortion occurs, for a different matrix (the resolution matrix \( R \)), but that explanation works in the same way here. This distortion is sometimes called the Point Spread Function (PSF).

The reason of the name “Point Spreading” is because in general \( m_I \) is a low model resolution image of the object \( \hat{m} \).

We see that the Hessian acts an operator that spreads the model \( \hat{m} \) into the image \( m_I \).

In section 7.4 we show that the Hessian \( H \) acts as the model (posterior) covariance matrix. Covariance matrix are measures of dispersion. Then the role of \( H^{-1} \) is to collapse the dispersion which the system implicitly introduced in the imaging system through equation 7.1.6.

7.2.2 For Inversion

Perhaps a more precise definition is that which measures the departure of the true model \( m \) from the estimated model \( \hat{m} \) and this is that

\[
\hat{m} = Rm. \]

In fact, if in this model (say a column vector) we single out a parameter, by saying that the \( k \)-th parameter is \( \rho \) and all others are zero. That is \( m_i = \delta_{ik}\rho \), then (assuming repeated index notation)

\[
Rm = r_{ij}m_j = r_{ij}\delta_{jk}\rho = r_{ik}\rho. \]
So the $k$ column of $R$ is “spreading” the parameter $\rho$. In the ideal case that column will only contain a one non-zero component in the row $k$, (the diagonal) which then will reproduce the input parameter $\rho$ exactly. Otherwise the input parameter $\rho$ is smeared in as many samples (labeled with the index $i$) as non-zero entries in the column $k$ of the $R$ matrix. On the other hand, we see that any row of $R$ acts as a weighted sum of the input model parameters. If $\sum_j r_{ij} = 1$, we could say that the row is computing a weighted average of the input model parameters. Again the ideal case here is $R = I$ which is hard to get in many practical cases.

The resolution operator $R$, if it can be explicitly computed (which is difficult in most of the practical cases) can be used to test the inversion algorithm. For example define $m_{ij}$ as a sets of ones and zeroes (the “checkboard” test), and see that the output should resemble the input.

### 7.3 The Data Covariance Matrix

Inversion is statistical and often the matrix (function) $L^*L$ is singular (having a null space). We need to add additional constraints to have a unique solution. One way to add constraints is by considering a covariance matrix $C$ and redefine the cost function 7.1.2 as

$$C(m) = [d - d_{\text{obs}}]^* C_d^{-1} [d - d_{\text{obs}}]$$

where $C_d$ is the covariance matrix of the data. Under a covariance matrix the pseudo-inverse operator becomes

$$L^+ = (L^*C_d^{-1}L)^{-1} L^*C_d^{-1}$$

and the model resolution matrix is

$$R = L^+ L = (L^*C_d^{-1}L)^{-1} L^*C_d^{-1}L.$$
7.4 The Generalized Tikhonov Regularization

The mathematical details of this regularization are shown in Appendix C. A general cost function is defined as

\[ C(m) = \|d - d_{\text{obs}}\|_P + \|m - m_0\|_Q. \]

where we have the Mahalanobis $P$-norm for the first term and the $Q$-norm for the second.

This objective function is set in a way that satisfies the demands of Bayesian inversion. The interpretation (see Wikipedia)\(^3\) is as follows.

- The matrix $P$ is the inverse of the data covariance matrix $C_d$.
- The matrix $Q$ is the inverse of the model covariance matrix $C_m$, and
- The vector $m_0$ is the expected value of $m$.

The simple case of the GSL in equation 7.1.2 is obtained by choosing $Q = 0$ and $P = I$. While the solution for this problem C.2.4 is set up in explicit closed form, we ignore in many cases some of the information required here. For example, the variance (mean) of the model $m_0$, which we are searching for. In a way this is the chicken-and-egg problem. However, when we are working with iterative models, we can start with a simple guess (let us say a variance of 0) and from the result of the first iteration update this variance. Similarly, we could do the same for the covariance of the model matrix. Since we know the data, we could compute the covariance matrix for the data in advance, and once computed it is a constant for all possible iterations. It is in this sense that the covariance matrix for the data $C_d$ is known as the “prior” and the covariance matrix of the model $C_{m_0}$ is called the “posterior”, in the Bayesian literature. So, it is interesting to know that the matrix $Q$ of the Generalized Tikhonov Regularization problem is the inverse Hessian $H^{-1}$ of the objective function.

Staying away from statistical arguments, we can regularize by choosing $x_0 = 0$, $P = I$ and $Q = \lambda I$ for a positive $\lambda$, then equation C.2.4 is the classical

\[ \hat{m} = (L^*L + \lambda I)^{-1}L^*d, \]

solution which always exists, for $\lambda > 0$.

7.5 The Seismic Problem

7.5.1 Introduction

The basic tools used for seismic inversion are borrowed from the field of physics, and numerical optimization theory. The Born approximation was developed in the field

\(^3\)http://en.wikipedia.org/wiki/Tikhonov_regularization
CHAPTER 7. SEISMIC INVERSION IN THE GENERALIZED LEAST SQUARE (GLS) CONTEXT

of quantum mechanics in the context of the Scattering Theory framework. The Kirchhoff diffraction formula from optics is modified to fit seismic wavefield computations.

We discuss below the problems of

- Born Inversion
- Kirchhoff Inversion

In the next chapter, we discuss FWI.

7.5.2 Born Inversion

Introduction

It is important to understand if a problem is linear and linear on what.

For example, let us write the simple equation

\[ f = ab^2c. \]

At this moment we have not defined if \( f \) is a function of \( a \), or \( b \), or \( c \). The reader can check that \( f \) is linear as a function of \( a \) and \( c \) but not as a function of \( b \). In the context of operators, the problem is a bit more complicated. We will show in the Born problem what do we mean by linear and what do we mean by linearization.

In general, a function which is linear, if it has an inverse, such an inverse is linear. The simplest example, is

\[ y = f(x) = ax, \]

where \( a \) and \( x \) are real numbers \( a \neq 0 \). The inverse here can be written as

\[ x = \frac{y}{a} \]

Then \( x \) is linear in \( y \), and the constant of proportionality is \( 1/a \). In any finite dimensional space, linear operators are matrices and if the inverse exists it is linear. That is if

\[ y = Ax \]

where \( A \) is some square non-singular matrix, then

---

4http://en.wikipedia.org/wiki/Scattering_theory
\[ y = T(x) = A^{-1}x \]

and

\[ T(\alpha x_1 + \beta x_2) = \alpha A^{-1}x_1 + \beta A^{-1}x_2 = \alpha T(x_1) + \beta T(x_2). \]

for any \( \alpha \) and \( \beta \) in the complex domain. We can extend this to infinite dimensional spaces. This is well illustrated with the Born problem presented next.

Here is the list of things that we are looking for:

- A Linear operator (the Helmholtz equation), from a point source to data.
- Decomposition of the data into incident and scattered data.
- A modified operator acting just on scattered data. This operator is non-linear in the scattered data.
- Linearization of the modified operator on scattered data to invert for.
- Inversion of the linearized modified operator into a forward modeling problem.
- At this point the forward modeling problem should be inverted. This time for the model parameters. There is analytical inversion for this operator Bleistein et al., 2000\(^5\) but we are looking for least squares solutions, which are numerical solutions.

If it sounds complicated it is. The first time any one reads this topic there is a high chance to get lost. There is a lot of history in the development of science on this few lines.

**Derivation**

We derive the Forward Modeling Born formula. Appendix F presents a similar derivation of the formula derived here, together with other approximations such as the Rytov and the de Wolf approximations, as well as the relation between them.

- The problem is formulated starting at the acoustic (Helmhotz) wave equation

\[ \left[ \nabla^2 + \frac{\omega^2}{v^2} \right] u(x, x_s, \omega) = -\delta(x_s - x)F(\omega), \quad (7.5.7) \]

Where we assumed a perfect localized source at \( x_s \) with a finite frequency (\( \omega \)) bandwidth.

\(^5\)https://books.google.com/books?isbn=1461300010
At this moment we need to define what we are inverting for. Inverting this equation for \( u(x, x_s) \) is known as forward modeling. The equation is linear in \( u(x, x_s) \), but it is not in wavespeed \( v(x) \). That is, if we write

\[
L(v, u) \equiv \left[ \nabla^2 + \frac{\omega^2}{v^2} \right] u(x, x_s)
\]

we have that

\[
L(v, \alpha u_1 + u_2) = \alpha L(v, u_1) + L(v, u_2)
\]

\[
L(\alpha v_1 + v_2, u) \neq \alpha L(v_1, u) + L(v_2, u)
\]

for arbitrary \( \alpha \) in the real (or complex) field.

We, for the moment, forget about inverting for \( v \) \(^6\) and try to invert for \( u \). While the operator \( L \) is linear in \( u \), it is a partial differential equation. One way to invert this is to perform FD, but we want to find a closed form solution using the theory of Green’s functions. Please observe that the operator \( L \) is linear in \( u \), and so the inverse, that is given a source (the right hand side) is to find the data \( u(x_s, x, \omega) \), which is forward modeling. However we would like to invert for other parameters. For example \( v \). Since \( L \) is non-linear on \( v \), we should expect that the inverse is non-linear in \( v \) as well.

Actually the Born modeling problem pretends to invert not for \( u(x, x_s, \omega) \) but for the scattering wavefield \( u_S(x, x_s, \omega) \), and we will show that this new problem (inversion for scattering) is non linear on the scattering field \( u_S(x, x_s, \omega) \) and non-linear in the velocity \( v(x) \) neither.

When dealing with differential equations we should use the tools of Green functions (impulse responses) which provide the mechanics for inversion.

The Green’s function for the acoustic wave equation satisfies:

\[
\left[ \nabla^2 + \frac{\omega^2}{v^2} \right] G(x, x_s, \omega) = -\delta(x - x_s)
\]

- We have the following important points to recognize
  - The full wavefield \( u \) can be divided into the incident \( u_I \) and the scattered \( u_S \) wavefields. That is \( u = u_I + u_S \). This is important to recognize because the incident wave field \( u_I \) is uninteresting in the way that what we record in the field is the scattering \( u_S \) wavefield, and if we can get an equation for just the scattering wave field \( u_S \), we would be eliminating a source of noise which is the \( u_I \) field. The idea of the Born approximation is to be able to come up with a formula where \( u_I \) dissapears.

\(^6\)inverting for \( v \) corresponds to the FWI problem considered in chapter 8
The incident wavefield satisfies the equation
\[
\left[ \nabla^2 + \frac{\omega^2}{v^2} \right] u_I(x, x_s, \omega) = -\delta(x - x_s) F(\omega)
\]

Please observe that the source here coincide exactly with the source in equation 7.5.7. Otherwise the derivations here (and in general in the literature) will not work.

There is a subtle issue here. We want to make the point that to get a contrast, that is to get a scattering wavefield \( u_S \) back, we need to consider a change of velocities. Then to the incident wave field we assign a background velocity \( c \), which at the is equal to \( v \) above the scattering discontinuity (reflecting surface?) but under it changes. Hence we write
\[
\left[ \nabla^2 + \frac{\omega^2}{c^2} \right] u_I(x, x_s, \omega) = -\delta(x - x_s) F(\omega) \tag{7.5.8}
\]

So it seems natural, to remove the incident wave field from subtracting the incident wavefield equation 7.5.8 from the total wavefield equation 7.5.7.

To avoid tedious algebra, we work with operators. The treatment here is valid for general operators, and the algebra is reduced to a small number of symbols.

Write equation 7.5.7 as
\[
Lu = -S
\]

with \( S(x, x_0, \omega) = F(\omega) \delta(x - x_s) \), where \( L \) being the Helmholtz operator in that equation.

The operator \( L \) can be decomposed in two operators \( L = L_0 + \Delta L \), where
\[
\begin{align*}
L_0 &= \nabla^2 + \frac{\omega^2}{c^2} \\
\Delta L &= \left( \frac{\omega}{v^2} - \frac{\omega}{c^2} \right)
\end{align*} \tag{7.5.9}
\]

So \( \Delta L \) is a perturbation to the background operator \( L_0 \). In addition, we decompose the total wavefield into incident and scattered wavefields,
\[
u = u_I + u_S. \tag{7.5.10}
\]

Then
\[ Lu = (L_0 + \Delta L)(u_I + u_S) = L_0u_I + L_0u_S + \Delta Lu \]

Now, from \( Lu = -\delta(x - x_s)F(\omega) = L_0u_I \), we find
\[ L_0u_S = -\Delta Lu. \]

At any time, on this analysis, we can replace the operators for their explicit definition. For example we can rewrite the previous equation as
\[ \left[ \nabla^2 + \frac{\omega^2}{c^2} \right] u(x, x_s, \omega) = -\left( \frac{\omega^2}{v^2} - \frac{\omega^2}{c^2} \right) u(x, x_s, \omega) \]

Apparently we can invert for \( u_S \) by saying that
\[ u_S = -L_0^{-1}\Delta Lu, \quad (7.5.11) \]

but there is a problem. \( u = u_I + u_S \), so there is a recursion here. This is a non-linear problem.

You can come up with an iterative solution as follows. To simplify call \( A = -L_0^{-1}\Delta L \). Assume, to start the iterative cycle, that the first scattering field \( u_S \) is zero \(^7\), so
\[ u_S = Au_I, \]

Then for the second iteration
\[ u = u_I + u_S = u_I + Au_I, \]

For the third iteration
\[ u_S = Au = Au_I + A^2u_I, \]

keep going (for ever) and find this
\[ u_S = \left( \sum_{i=1}^{\infty} A^i \right) u_I \quad (7.5.12) \]

\(^7\)this is the Born approximation
This is the Neumann series.\(^8\) It is sometimes called the Born series. See for example, Wu et. al, 2006\(^9\) or Innanen, 2009.\(^10\)

We prefer the “Neumann Series” naming since it has a broader scope. It should converge as long as the spectral (maximum absolute eigenvalue) norm of \(A\) is smaller than 1. No surprise here since, by writing the equation as

\[
us = A(ul + us),
\]

we are saying that

\[
(I - A)us = Au,
\]

and so we can write

\[
us = (I - A)^{-1}Au
\]

but

\[
(I - A)^{-1} = I + A + A^2 + \cdots + A^n + \cdots,
\]

this series converges provided \(|A| < 1\). We write again the Neuman series as

\[
us = \left(\sum_{i=1}^{\infty} A^i\right)ul
\]  \hspace{1cm} (7.5.13)

which is the same as equation 7.5.12.

We recognize the geometrical series from the multiple–scattering problem. Doing Neumann iterations should provide multiple–scattering. Multiple scattering with Neumann series is useful to model (or invert for) multiples or special events, by choosing particular subseries. For example, the first term is the Born approximation, but we could pick only those terms that provide all forward scattering and just one back scattering. That is, model only primaries. This is the de Wolf approximation and we will talk about this in Appendix F together with the Ritov approximation, which can be seen also as an especial case of the Neumann series. Weglein et. al, 1997.\(^11\) show applications of the Neumann

---

\(^8\)http://en.wikipedia.org/wiki/Neumann_series
\(^9\)http://es.ucsc.edu/ xjie/papers/Wu_Xie_Wu_2006_one_return.pdf
\(^10\)http://gji.oxfordjournals.org/content/177/3/1197.full.pdf+html
\(^11\)http://www.slb.com/ /media/Files/technical_papers/140/1444298.pdf
series for attenuation of multiples in seismic reflection data and Matson and Weglein\textsuperscript{12} provide a mathematical derivation of the relationship between the scattering theory and the primaries and multiples of reflection seismic data. Next, we provide a few insights about why the Neumann series represents a multi-scattering problem.

**Multiple Scattering:** To better understand how the Neumann series is related to the multiple scattering experiment, let us think about the seismic problem. We will write the Neumann series 7.5.13 with $A = - L_0^{-1} \Delta L = G_0 \alpha$, where $G_0 = - L_0^{-1}$ is the Green’s function for the background operator $L_0$, and $\Delta L = \alpha$ is the model perturbation. In addition we assume that the incident wave field $u_I$ is equal to some source signature factor $F(\omega)$ times the source Green’s function. That is,

$$u_I = F(\omega) G_s = F(\omega) G(x_s, x, \omega)$$

We then rewrite the Neumann series 7.5.13 as

$$u_S = F(\omega) [(G_0 \alpha) + (G_0 \alpha)^2 + \cdots + (G_0 \alpha)^n + \cdots] G(x_s, x, \omega) \quad (7.5.14)$$

To understand the meaning of this formula, let us think of the simplest multi-scattering model with just two scatter point perturbations at two locations $x_1$, $x_2$. That is, the background model and the actual model are identical, except at two different points. The total perturbation is given by

$$\alpha(x) = \alpha_1 \delta(x - x_1) + \alpha_2 \delta(x - x_2)$$

Let us write just the first two terms of the Neumann series, and this time we use integrals (this might help for the interpretation).

That is

1. The first term (the **Born Approximation**):
   
   This is
   
   $$u_S(x_s, x_g, \omega) = F(\omega) G_0 \alpha G = F(\omega) \int dy G(x_s, y, \omega) \alpha(y) G(y, x_g)$$
   
   $$= F(\omega) \alpha_1 G(x_s, x_1) G(x_1, x_g) + F(\omega) \alpha_1 G(x_s, x_1, \omega) G(x_1, x_g, \omega)$$

\textsuperscript{12}http://www.eos.ubc.ca/research/cdsst/members/2_96.pdf
7.5 The Seismic Problem

Figure 7.1: A model with source at $x_s$, receiver at $x_g$, and two point scatterers, $x_1$ and $x_2$.

where we use the sifting property of the Dirac $\delta$ functions to collapse the integrals. Let us, for the purpose of illustration, assume the free space Green’s function with constant velocity $c$. That is,

$$G(x, y, \omega) = \frac{1}{|x - y|} \exp \left( \frac{i\omega |x - y|}{c} \right)$$

Then the Born approximation predicts

$$u_S(x_s, x_g, \omega) = F(\omega) \frac{\alpha_1}{|x_s - x_1|} \exp \left( \frac{i\omega (|x_s - x_1| + |x_g - x_1|)}{c} \right) + F(\omega) \frac{\alpha_2}{|x_s - x_2|} \exp \left( \frac{i\omega (|x_s - x_2| + |x_g - x_2|)}{c} \right)$$

which is correct up to a source with spectrum $F(\omega)$ and a constant velocity ray tracing with wavespeed $c$, and perturbations $\alpha_1$ and $\alpha_2$. Figure 1 illustrates this case, with source at $x_s$, receiver at $x_g$. The blue rays correspond to scatter $x_1$ and the brown rays to scatter $x_2$.

2. The second term: This is

$$u_S = F(\omega)G_0\alpha G_0\alpha G$$

When we see several $G_0$ terms, this means that they are expanded like matrix multiplications\(^{13}\) which in the continuum is

$$u_S(x_s, x_g, \omega) = F(\omega) \int d\mathbf{y}_1 d\mathbf{y}_2 G_0(\mathbf{x}_s, \mathbf{y}_1, \omega) \alpha(\mathbf{y}_1) G_0(\mathbf{y}_1, \mathbf{y}_2, \omega) \alpha(\mathbf{y}_2) G_0(\mathbf{y}_2, \mathbf{x}_g, \omega)$$

\(^{13}\)While matrix multiplications do not commute (and integral operator neither), the operands under the integral commute, and in addition we do not have to think about how many rows and columns per each operand. How convenient is to use integrals instead of sums. Without calculus (and hence the continuum assumption) we would not get near where we are today.
and using the sifting property of the Dirac δ functions we collapse the double multidimensional integrals into

\[ u_S(x_s, x_g, \omega) = \alpha_1 \alpha_2 G(x_s, x_1, \omega) G(x_1, x_2, \omega) G(x_2, x_g, \omega) + \alpha_1 \alpha_2 G(x_s, x_2, \omega) G(x_2, x_1, \omega) G(x_1, x_g, \omega) \]

We can use the constant velocity Green’s function to explain that the ray tracing approximation for this term corresponds the experiment in Figure 2.

A total of 4 paths for the two points and two terms. Two paths for single scattering, and 2 for double scattering. Adding more terms will bring new ray paths. For example the third term of the Neumann series will have a path such as S121R, where the ray goes from source x_s to point x_1, then to point x_2, then back to point x_1 and then to the receiver x_g. In general the perturbation is defined in a continuum of scatterers which are discretized in a grid for computational purposes, allowing an intensive evaluation of multi-dimensional integrals.

We can look at the Neumann series as a sum of all possible terms as shown with the simple two scatterers example where only two terms were considered, but we can also look at the series in a different way. For example, we can write the Green’s function as the sum of other Green’s functions and expand the Neumann series as this sum. As an illustration, in the marine case, we use Weglein et. al, 1997.\(^ {14}\) to show how the ghost, created by a free surface boundary, can be differentiated from the primary event without air-water interface (no ghost).

\(^ {14}\)http://www.slb.com/ /media/Files/technical_papers/140/1444298.pdf
Figure 7.3: Ghost in blue. Direct wave in red. Source $S$, receiver $R$. Two Green’s functions are labeled. The ghosted Green’s function $G_g^0$, and the forward (direct) Green’s $G_f^0$.

Figure 7.5.2 shows an sketch of two ray paths, a ghost path and a direct path. The background (with water velocity) Green’s function can be thought of the sum of the two components

$$G_0 = G_g^0 + G_f^0$$

where $G_g^0$ is the ghosted free surface reflection and $G_f^0$ is the direct (forward) wavefield. Both recorded at $R$. Hence the Neumann series 7.5.14 can be decomposed in three new equations:

The complete wavefield with ghost included

$$u_S = F(\omega)[(G_0\alpha) + (G_0\alpha)^2 + \cdots + (G_0\alpha)^n + \cdots]G$$

The ghost wavefield (no direct arrival) included

$$u_{Sg} = F(\omega)[(G_g^0\alpha) + (G_g^0\alpha)^2 + \cdots + (G_g^0\alpha)^n + \cdots]G$$

The forward wavefield, no ghost included.

$$u_{Sd} = F(\omega)[(G_f^0\alpha) + (G_f^0\alpha)^2 + \cdots + (G_f^0\alpha)^n + \cdots]G$$

Weglein et. al., use these ideas to remove free surface (as well as internal) multiples from marine seismic reflection data. A mathematical framework to remove internal multiples (in 2D) is provided by Matson and Weglein. ¹⁵. In Matson

¹⁵http://www.eos.ubc.ca/research/cdsst/members/2_96.pdf
and Weglein simple 2D models (two and three layer models) are used to illustrate the Neumann series and its use in the removal of internal multiples.

Finally, before moving on to single scattering, we could also look at the model perturbation as a compound of two partials. That is as the perturbation for the forward scattering wavefield and a perturbation backscattering wavefield.

\[ \alpha = \alpha_f + \alpha_b. \]

This observation took David de Wolf into an approximation for only primaries. This problem is considered in Appendix F.

Single Scattering: The Born Approximation: The Born approximation changes equation 7.5.11 to

\[ u_S = -L_0^{-1} \Delta Lu_I, \]

where we replace the total field \( u \) in the right hand side by the incident field \( u_I \).

- So we found that the inversion for the scattering field \( u_S \) is non–linear and the linear approximation (truncating the Neumann series from powers of 2 and above) is the Born approximation.

- We want to evaluate \( A \) so we can have an explicit solution of the Born approximation problem.

Recall that \( A = -L_0^{-1} \Delta L \). First, let us call

\[ m = \frac{\Delta L}{\omega^2} = \frac{1}{v^2} - \frac{1}{c^2} \]

This is our model \( m \), and the Born forward modeling formula is linear on \( m \). This is good news for inversion for \( m \). Now for \( L_0^{-1} \). This is the Green’s function for background scattering model. That is \( L_0 \), shown in equation 7.5.8.

\[ L_0 \left[ \nabla^2 + \frac{\omega^2}{c^2} \right] G(x, x_g, \omega) = -\delta(x - x_g) \]

where now we call \( x_g \) some observation point and \( x \).

Finally, for the formula \( u_S = Au_I \), we know that \( u_I \) is the incident wave which is a convolution of the Greens function for the background field with the source wavelet \( F(\omega) \). That is from equation 7.5.8

\[ u_I = G(x_s, x, \omega)F(\omega). \]

The operator \( Au_I = -L_0^{-1} \Delta Lu_I \) is an integral operator (in the finite dimensional space it is a matrix vector multiplication). That is
\[ d = u_s(x_s, x_g, \omega) = -\omega^2 F(\omega) \int_V dx G(x_s, x, \omega) m(x) G(x, x_g, \omega). \] (7.5.15)

Please observe that this integration did not require integration over volumes with delta functions (as it is common in the literature). Also we did not use boundary conditions (radiation to and from infinity, the Sommerfeld radiation conditions), however they are required to guarantee convergence. In a discrete world, we always need to work with boundaries.

Here is the interpretation of this formula. The symbol \( \omega \) is the frequency. \( F(\omega) \) is the source signature in the frequency domain, \( \omega^2 \) comes from the (Fourier transform, or in other words the Helmholtz) second time derivative of the acoustic wave equation, \( m(x) \) is a model perturbation (the model) and \( G(x, x_g, \omega) \) and \( G(x_s, x, \omega) \) are the receiver and source Green’s function. The wave propagation starts in time domain at the source location \( x_s \) with a source with signature in the frequency domain given by \( F(\omega) \). We assume this function to be real. The Green’s function \( G(x_s, x, \omega) \) represents the forward propagating wavefield through the medium, then a velocity contrast (the model perturbation) is picked up by \( m(x) \), a backscattering wavefield will be generated sending a wave upward. This wavefield is computed with the receiver Green’s function \( G(x, x_g, \omega) \). The triple product in the integrand is added for each volume scatter. The factor \( \omega^2 \) is a double derivative in time coming from the Green’s theorems.

It is common to hear that the Born inversion formula is inverted for reflectivity. It is clear here that the inversion should find an approximation of the perturbation of the slowness square. In this sense, Born inversion can be used to find a new “better” velocity. This is a linearization of the FWI problem where the model is the acoustic wave equation\(^\text{??}\)

\[ Lu = \left( \frac{1}{v^2} \frac{\partial}{\partial t^2} - \nabla^2 \right) u = s. \]

Note that this operator is non-linear with respect to \( v \), but it is linear with respect to \( s_q = 1/v^2 \). The FWI problem in chapter 8 looks at \( L \) as an operator of \( v \) and not as an operator on \( 1/v^2 \). This makes the FWI non-linear and more complicated.

The forward modeling Born formula, for inversion is, in a way poststack. That is, the model is independent of the source/receiver configuration and we can

---

\( ^\text{??} \)We keep the minus “-” sign in front of the integral here. This sign came from the way we picked the Green’s functions. That is, from the equation is \( LG = \pm \delta \), the \( \pm \) sign is picked. This sign is arbitrary as long as all the computations related to this equation are consistent with that sign.
use redundancy to improve the model’s signal to noise ratio by stacking. Another advantage of equation 7.5.15 is that it is well suited to be computed using FD. A more complex model which depends of source/receiver configuration is reflectivity. This model is simulated by the Kirchhoff approximation and it is explained in section 7.5.3. The Kirchhoff approximation shown there is not well suited to be computed using FD.

We can write the Born modeling as

\[ d = Lm \]

with

\[ L = -\omega^2 F(\omega) \int_V d\mathbf{x} G(\mathbf{x}, \mathbf{x}_g, \omega)G(\mathbf{x}_s, \mathbf{x}, \omega)[.] \]

where [.] means that the operator \( L \), operates on the model \( m \) by assigning it to the box [ ], making of \( L \) a linear operator in the space of models \( m \).

The Adjoint: Born Migration.

The term “migration” is not well defined in the imaging community. Migration seems to be a method that provides an image of a model from data. The term “adjoint” has a clear and unique meaning for each linear operator, be this an integral equation or a differential equation, etc. Now, for the Born approximation we can find an adjoint (Born migration); for the Kirchhoff approximation we can find an adjoint (Kirchhoff migration). For the acoustic wave equation Helmholtz operator \( L = \nabla^2 + \omega^2/v^2 \) is self-adjoint, but the operator \( L \) is non-linear in \( v \).

Taking the adjoint of the operator \( L \) above, from the data space to the model space we find

\[ m_1(y) = -\omega^2 F(\omega) \int_\Sigma d\xi G^*(y, x_s, \omega)d(x_s, x_g).G^*(y, x_g, \omega) \]

where \( x_s = x_s(\xi) \) and \( x_g = x_g(\xi) \) are in a surface \( \Sigma \) parameterized by the 2D vector \( \xi \).

The interpretation of this is the following. The point \( y \) is the image output point. The data collected at the receiver \( x_g \), and due to a source \( x_s \) is back propagated by the adjoint (complex conjugate ) Greens function \( G^*(y, x_g, \omega) \). At the same time a
7.5 The Seismic Problem

synthetic source with signature\(^{17}F(\omega)\), is forward propagated by the adjoint Greens function \(G^*(y, x_s, \omega)\) from the source location \(x_s\) down to the image point \(y\). At that point the product of the two propagated Green’s functions represents the imaging condition (cross-correlation in time) which will be added to the image space while scaling over the acquisition surface \(\xi\). For example in shot profile, the source \(x_s\) is fixed and all receivers traces are migrated by identifying the proper receiver location through the vector \(\xi\).

The adjoint operator \(L^*\) is then represented by

\[
L^* = -\omega^2 F(\omega) \int_{\Sigma} d\xi G^*(y, x_s, \omega) G^*(y, y_g, \omega) [.] 
\]

The Hessian

The Hessian is given by \(L^* L\). This is,

\[
H(y) = \omega^4 F^2(\omega) \int_{\Sigma} d\xi \int_V dx G^*(x_s, y, \omega) G^*(y, x_g, \omega) G(x, x_g) G(x_s, x, \omega) [.] 
\]

We we just cascaded \(L\) with \(L^*\), from left to right. Recognizing equation 7.1.6

\[
H \hat{m} = m_I 
\]

The application of \(H\) to \(\hat{m}\) is given by

\[
H \hat{m} = \omega^4 F^2(\omega) \int_{\Sigma} d\xi \int_V dx G^*(x_s, y, \omega) G^*(y, x_g, \omega) G(x, x_g) G(x_s, x, \omega) \hat{m}(x). 
\]

We can write \(H(y) \hat{m}(x) = m_I(y)\) but it is common to see in the literature

\[
H(x, y) = \omega^4 F^2(\omega) \int_{\Sigma} d\xi G^*(y, x_s, \omega) G^*(y, x_g, \omega) G(x, x_g, \omega) G(x_s, x, \omega) \quad (7.5.16) 
\]

where the integration over \(x\) was suppressed. This notation has sense if we consider that the application to a model \(\hat{m}(x)\) is seen as an inner product. That is,

\[
H(x, y) \hat{m} = \langle H(x, y), \hat{m}(x) \rangle \\
= \int_V dx H(x, y) \hat{m}(x).
\]

\(^{17}\)It is common to use a different source signature for migration, other than \(F(\omega)\). For example a zero phase Klauder or Ricker wavelet with a limit spectrum within the range of the field data. In that case instead of \(F(\omega)\) we could write \(T(\omega)\), however for the purposes of interpretation the names \(S\) or \(T\) are irrelevant. At the end we will get some spectrum function, name it \(S\), or \(T\), or something else which will represent the extend of the output wavelet temporal aperture and, of course, it plays a very important issue on resolution.
(assuming $H$ is a real matrix), which still is a linear operator.

While knowing $H^{-1}$ is of great help to solve the inverse problem the computation is still an issue. Not only the multi-fold integral in equation 7.5.16 is expensive, but the number of elements to compute for a complete Hessian is enormous. The way to go around the cost of storing and computing the Hessian is by exploiting the following ideas

- **The locality of the Hessian.** By defining a target zone with reduced size, we can let the set of possible $(x, y)$ coordinates to be bounded to the target zone, which could be a small fraction of the whole model.

- **The symmetry of the Hessian.** Since $H(x, y) = H(y, x)$ then we only have to compute $n(n+1)/2$, elements where $n$ is the dimension of the target zone depth samples.

- **The band diagonal structure of the Hessian.** In practice the Hessian is a matrix with a few diagonal bands. That is, it is very sparse, so we can choose to compute only the non-zero or relatively large components.

The algorithms and examples for implementations of the Hessian are outside of the scope of this document. The reader is encouraged to look at Alejandro Valenciano’s Ph.D. thesis. Valenciano has plenty of examples, figures and illustrations of the concepts discussed here. Of course equation 7.5.16 for finite dimensional spaces (the discrete) is a product of matrices, which results a collection of sums as those in Valenciano’s equation (2.8).

An interesting case happens when we only consider the diagonal of the Hessian (commonly used to balanced wave equation migration algorithms amplitudes). Equation 7.5.16 turns out into

$$H(x, y) = F(\omega)^2(\omega^4 \int_{\Sigma} d\xi |G(x_s, y, \omega)|^2 |G(y, y_g, \omega)|^2$$

$$= \int_{\Sigma} d\xi \ [(F(\omega)\omega^2|G(x_s, y, \omega)|][F(\omega)\omega^2|G(y, y_g, \omega)|]^2$$

A wave-equation migration program can compute this diagonal on the fly by observing that this is the imaging condition amplitude squared and scaled by $\omega^2$ and $F(\omega)^2$. So at the same time that the imaging condition is applied an accumulator could be put in place to add (over all source and receivers) the amplitude squared of the green functions. The square amplitudes of the Green’s functions can be interpreted as energy. In this, if we ignore the off–diagonal components of the Hessian we can get an energy normalized image $\hat{m} = H^{-1}m$ with a single division by the diagonal entries (illumination) of the Hessian.

---

18For a 3D survey with a grid of $10^3$ points on each direction. That is a model size of $10^9$ points, the size of $H$ is $(10^3)^2 = 10^{18}$, which is 1 exa-flop, or 4 exa-bytes.

7.5 The Seismic Problem

7.5.3 Kirchhoff Inversion

Introduction

The Kirchhoff method is borrowed from the field of wave optics. Gustav Kirchhoff\(^{20}\) is mostly known from his Kirchhoff circuit laws but his contributions go much further. In the context of seismic imaging, the geophysics community borrowed the Kirchhoff Diffraction Formula\(^{21}\) to obtain forward modeling equations and use them to invert for reflectivity. While Kirchhoff’s derivations were done in the context of light propagation, where the velocity is assumed constant, the theory can be extended to variable velocity media (and even anisotropic elastic) by the use of general Green’s functions which could be computed numerically with finite difference algorithms and/or ray tracing algorithms. These last technique assumes high frequency asymptotic approximations including the Wentzel, Krammers, Brillouin, and Jeffrey's WKBJ approximation\(^{22}\) which is also borrowed from the field of quantum mechanics.

A little history:

- **Newton**: Seventeen century. While the corpuscular theory of light\(^ {23}\) is not attributed to Newton, he embraced it and this was, in a way, unfortunate because it slowed down the development of diffraction theory, which picks up again more than 100 years after his death.

- **Huygens 1678**: Stated that light is a wave\(^ {24}\). Huygens’ theory was in hold for more than 100 years due to Newton’s reputation, until Thomas Young provided his experimental demonstration.

- **Young 1803**: Thomas Young designed the double slit experiment\(^ {25}\) which shows that light has the property of interference, as known in water waves. This was a big blow for the corpuscular theory of light, which has been, at this point the accepted theory for more than 100 years.

- **Fresnel, Aragon, Poisson 1818**: Acceptance of the wave theory of light. After Young’s experiment the battle between the corpuscular and wave theories of light became quite interesting. Augustin Fresnel\(^ {26}\) used Huygens’ principle to provide the first integral formulation of diffraction of light, known as the Huygens-Fresnel principle\(^ {27}\). He presented his formula in a 1818 competition organized by the French Academy of Sciences to explain the properties of light.

\(^{22}\)http://en.wikipedia.org/wiki/WKB_approximation
\(^{24}\)http://en.wikipedia.org/wiki/Treatise_on_Light
\(^{25}\)http://en.wikipedia.org/wiki/Young%27s_interference_experiment
\(^{26}\)http://en.wikipedia.org/wiki/AugustinJean_Fresnel
\(^{27}\)http://en.wikipedia.org/wiki/Huygens%E2%80%93Fresnel_principle
Among two of the members of the committee were Poisson \(^{28}\) and Arago \(^{29}\). Poisson was a theoretician, and Arago an experimentalist. Poisson, who was in favor of the particle theory of light, dedicated his efforts to prove that Fresnel was wrong. He designed the mathematical model of a small disc which he will illuminate, and predicted that a spot of light would be at the other on the axis which goes perpendicular through the middle of the disc. Then the he claimed that Fresnel’s formula was wrong, because, how come a spot of light would come in the other side where the light should be obstructed by the small disc? Then Arago (the other member of the committee) who was an experimentalist, set up the experiment and found the point of light at the other end of the disc, as predicted by Fresnel’s equation. This was a blow for Poisson and a great hit for the wave theory of light. After this there were no more discussions. From this time, the spot on Arago’s experiment is known as Arago spot \(^{30}\). Why not Poisson spot? It was believed to be the last nail in the coffin for the corpuscular theory of light, until 1905.

• **Maxwell Equations 1861**: The history \(^{31}\) as well the impact of Maxwell equations is of outstanding importance. While Maxwell has today the credit for the four equations that describe the electromagnetic theory, Hertz, Gibbs, and Heaviside, among others, contributed to the reduction of the 20 equations published by Maxwell to just 4 equations. We suggest to read, at least, the paragraph under the title “The term Maxell’s equations” on the previous link. Not only Maxwell equations unify the electric and magnetic fields, but they show that light is a wave far beyond Young’s experiments and Fresnel mathematical formulations.

While the wave equations derived from Maxwell’s equations are vectorial, an approximation (by considering only a component of that vector) to scalar works good under many situations and Kirchhoff took advantage of that.

• **Kirchhoff 1883**: It was amazing how Fresnel got such a precise formula without using formal theory. The mathematical work of the derivation from Fresnel, of the The Huygens-Fresnel principle, using Huygen’s principle, is shown in Appendix E (following Born and Wolf’s \(^{32}\) Principles of Optics classical book. Then we verify that from the Huygens-Fresnel principle, by using stationary phase, a point in space is the superposition of previous sources as indicated by Huyghens. This talks about the great intuition that Fresnel had. Kirchhoff, using a scalar wave equation \(^{33}\) derived by Hermann von Helmholtz \(^{34}\), to-

\(^{28}\)http://en.wikipedia.org/wiki/Sim%C3%A9on_Denis_Poisson
\(^{29}\)http://en.wikipedia.org/wiki/Fran%C3%A7ois_Arago
\(^{30}\)http://en.wikipedia.org/wiki/Arago_spot
\(^{32}\)https://books.google.com/books?isbn=0521784492
\(^{33}\), which is an approximation of the vector wave equation derived from Maxwell’s equations
\(^{34}\)http://en.wikipedia.org/wiki/Helmholtz_equation
together with Green’s second identity \(^{35}\) found an analytic closed formula which is equivalent to Huygens-Fresnel formula derived by Fresnel based on Huygens principle. This formula is known as the Kirchhoff-Fresnel equation \(^{36}\).

- **Poincaré, Sommerfeld 1892, 1894**: Poincaré in 1892 showed that the boundary conditions used by Kirchhoff were inconsistent. That is, for a second order differential equation you cannot have the field and its first derivative vanishing in a continuum piece of surface. Otherwise it will vanished everywhere. Kirchhoff assume a diffraction aperture (hole) where the light goes through (transmission coefficient of 1), and outside the aperture some material that blocks the light (transmission coefficient of 0). He assumed that not only the wave does not goes through the area away from the aperture, but its first derivative vanishes as well. Sommerfeld, who introduced the term “diffraction” proved (1894) the same results obtained by Poincaré two years before. Kirchhoff accepted that his formula was an approximation, which is valid for small incident and transmission (obliquity) angles.

Sommerfeld (see Goodman’s Fourier Optics book listed in the next section) fixed Kirchhoff-Fresnel formula by choosing a Green’s function using the method of images. That is two Green’s function located at points in a way that the aperture surface is a mirror. This is known as the Rayleigh-Sommerfeld integral (not in Wikipedia yet).

- **Einstein 1905**: Einstein was awarded the Nobel Price in Physics, no for the relativity theory, but for the photoelectric effect \(^{37}\). The corpuscular theory of light came alive again by consider light as a package of photons. This does not mean that light is not a wave. Light under quantum mechanical systems can be seen both as a wave and as a particle.

- **Hagedoorn 1954**: Hagedoorn is to seismic like Huygens is to optics. The method of “diffraction stack” is based on the idea of adding amplitudes along a diffraction surface. These ideas were put into mathematical formulation first by Bill Schneider in 1978.

- **Schneider 1978**: The term “Kirchhoff based migration” was introduced by Schneider in its 1978 Geophysics paper “Integral formulation for migration in two and three dimensions. Schneider’s idea was to provide mathematical foundation to the diffraction stack which was commonly done in processing. His integral formulation is for poststack data and it is migration algorithm.

- **Bleistein 1984, 1987**: The Kirchhoff approximation. Based on ray tracing we say that the scattered field is equal to the incident field times the oblique reflection

\(^{35}\)http://en.wikipedia.org/wiki/Green’s_identities  
\(^{36}\)http://en.wikipedia.org/wiki/Kirchhoff’s_diffraction_formula  
\(^{37}\)http://en.wikipedia.org/wiki/Photoelectric_effect
CHAPTER 7. SEISMIC INVERSION IN THE GENERALIZED LEAST SQUARE (GLS) CONTEXT

coefficient. That is

\[ u_S = Ru_I. \]

This observation, and the additional assumption

\[ \frac{\partial u_S}{\partial n} = -R \frac{\partial}{\partial n} u_I. \]

serve as new boundary conditions for the Helmholtz-Kirchhoff boundary value problem. Bleistein’s Mathematical Methods for Wave Phenomena book \(^{38}\) formulated these new boundary conditions.

In other words, these boundary conditions, together with the Sommerfeld radiation conditions, when applied to the Helmholtz-Kirchhoff theorem \(^{39}\), provide a unique solution of the Helmholtz equation with is consistent with ray ray tracing solutions. The following useful features are consequences of Bleistein’s boundary conditions:

- Provide a unique solution to the Helmholtz equation for acoustic seismic data.
- Introduce the reflection coefficient into the acoustic seismic waves modeling equation, extending an optics problem to the field of seismic.
- Link Kirchhoff inversion with finite difference migration through the deconvolution imaging condition

\[ R(x_s, x_g, x, \omega) = \frac{U(x, \omega)}{D(x, \omega)} \]

where \( U(x, \omega) \) is the upgoing (scattered) wavefield, and \( D(x, \omega) \) is the incident wavefield. Docherty, 1991. \(^{40}\)

- Provide a way to invert for reflectivity. However, It took Bleistein more than a decade to invert for reflectivity from the Kirchhoff modeling equation. He inverted using the Born modeling formula which was defined in a volume. The Kirchhoff modeling equation is defined in a surface. In 1997 he found a way to invert directly from Kirchhoff by extending the Kirchhoff surface integral into a volume integral by introducing the singular function of the surface \( \delta(\Sigma) \). In 1987 Bleisten published his \(^{41}\) On the imaging of reflectors of the Earth paper which is referred as “true amplitude migration” in the industry. Here Bleistein did not invert from a

\(^{38}\)https://books.google.com/books?isbn=0080916953
\(^{40}\)http://geophysics.geoscienceworld.org/content/56/8/1164.abstract
\(^{41}\)http://www.cwp.mines.edu/ norm/Herman/input1/node17.html
modeling formula. He just posed an equation (the Kirchhoff inversion) for a function \( \beta(x) \) which, when applied to Kirchhoff modeling data will provide the oblique reflection coefficient. The way that he know the factors involved in \( \beta \) is by applying the stationary phase method to the cascade of the migration with the Kirchhoff modeling formula which asymptotically would result in the reflectivity function \( R(x_s, x_g, x) \).

In any case, the inversion considered here is based on the least squares method, so we will not deal with the type of inversion done by Bleistein. Still, need the forward modeling equation.

**Forward Modeling**

The Kirchhoff modeling formula has Kirchhoff at the start. A great reference for this topic is the *Fourier Optics* textbook by Goodman. Gookman, section 3.4 is titled “The Kirchhoff Formulation of Diffraction by a Planar Screen”. This solution was borrowed from optics into seismic as it is.

We start with the Green’s theorem. That is, assume that \( U(x) \) and \( G(x) \) are two complex-value functions. Assume a closed volume \( V \). If \( U \) and \( G \), and their second partial derivatives are single–value and continuous within and on the surface \( S \) surrounding the volume, then

\[
\int_V dx \left( U \nabla^2 G - G \nabla^2 U \right) = \int_S dS \left( U \frac{\partial G}{\partial n} - G \frac{\partial U}{\partial n} \right)
\]  

(7.5.17)

where \( \partial / \partial n \) derivative along the normal to the surface.

The idea is to pick \( G \) which satisfies the Helmholtz equation. That is

\[
\left[ \nabla^2 + \frac{\omega^2}{v^2} \right] G(x, x_g, \omega) = -\delta(x - x_g)
\]  

(7.5.18)

where \( x_g \) is a point inside the volume \(^{42}\). On the other hand, choose \( U(y, x, \omega) \) which satisfies the homogeneous Helmholtz equation. That is, we assume that this is the wavefield response at \( y \) due to a scatterer at \( x \). It is homogenous because the scatterer is a secondary source and not a true source (think of an internal reflection).

\[
\left[ \nabla^2 + \frac{\omega^2}{v^2} \right] U(x, y, \omega) = 0
\]  

(7.5.19)

Multiply equation (7.5.18) by \( U \) and equation (7.5.19) by \( G \) and substract. The cross-scalars with factors \( GU \) cancel, the result is that

---

\(^{42}\) we think about Green’s functions at the receiver, which is convenient to get the seismic experiment match the optics work
\[ U \nabla^2 G - G \nabla^2 U = \delta(x - x_g)U(y, x, \omega). \]

and since

\[ \int_V d\mathbf{x} \delta(x - x_g)U(y, x, \omega) = U(y, x_g, \omega), \]

we find, using the Green’s identity 7.5.17 that

\[ U(y, x_g, \omega) = \int_{S_c} dS_c \left( U \frac{\partial G}{\partial n} - G \frac{\partial U}{\partial n} \right) \] (7.5.20)

This is the so called Helmholtz and Kirchhoff theorem \(^{43}\), because Kirchhoff derived this equation based on Helmholtz formulation of the scalar electromagnetic wave equation on the context of light propagation. However there are no approximations so far considering just that \( G \) is a Green function that satisfies the acoustic wave equation (with variable velocity) and \( U \) is a wavefield that also satisfies the acoustic variable velocity acoustic wave equation.

The Rayleigh Sommerfeld Approximation: When Kirchhoff solved equation 7.5.20, he failed to provide correct boundary conditions. Sommerfeld came up with a way to solve Kirchhoff’s error. This is based on the method of images. We modify Kirchhoff’s formula to make it fit our seismic experiment. We should define the surface the source \( S_c \) enclosing the volume \( V \). The theory says that the surface should be closed. Figure 7.5.3 shows how this should be done. Assume that the reflecting surface is the red curve in the figure, which we call just \( S \). We want to extend to surface far from the edges to avoid edge effects, or taper the edges. This is represented by the dotted black lines in the bottom attached to the reflector. The blue circle is considered so far that the radiation conditions indicate that the field should reduce to 0 at infinity and we can assume that the contribution is

We then claim that the integration 7.5.20 over all closed surface can be approximated to the integral over the red and green pieces of surface. The figure is not drawn at scale and we assume that the black and blue boundaries are away from the reflector \( S \). This is the so called Sommerfeld radiation conditions. What it means is that the product of the field \( U \) and the normal derivative of the Green’s function shrink as \( 1/r^2 \), (this is to expect if both are of the type \( A(x \exp(i \mathbf{k} \cdot (x - x_0)) r \) with \( r = \| x - x_0 \| \)). Since the differential of the surface integral \( dS_c \) shrinks as \( 1/r \), (think that for a sphere \( dS_c = 4\pi r \)) then the product of the integrand with the surface element \( dS \) shrink as \( 1/r \). That is, it goes to zero as \( r \to \infty \). The same argument applies for the product of the Green’s function \( G \) with the normal derivative of \( U \).

We still need to eliminate the contribution of the field due to the green (top) boundary $\Sigma$. This is precisely the problem that Sommerfeld solved to fix Kirchhoff boundary conditions, by picking a Green function using the method of images. The idea is the following. In Figure 7.5.3 we assume that the medium on top of the surface $\Sigma$ has symmetrically the same properties as the medium below. That is, that if we call $z$ the vertical direction, then $v(x, y, z) = v(x, y, -z)$, and the same will happen to the Green’s functions.

We want to solve the Homogeneous Helmholtz equation

$$\nabla^2 U(x, \omega) + \frac{\omega^2}{v^2} = 0$$

with two boundary conditions. The condition that

(i) $U(x, \omega)$ is zero in the surface $\Sigma$.

$$U(x, \omega) = 0, \quad x \in \Sigma$$

(ii) The normal derivative is non zero in $\Sigma$.

$$\frac{\partial U(x, \omega)}{\partial n} \neq 0, \quad x \in \Sigma$$

Provided that the solution exists, it should be unique. Let us use two sources and solve the problem.
with the sources as indicated in Figure 7.5.3. Away from the sources the Dirac Deltas are zero and the Green’s function $G$ satisfies the homogeneous Helmholtz equation. Now, the derivation of equation 7.5.20 is still valid for the chosen Green’s function. The added source $x'$ is outside of the volume of integration, so the volume integral that produced $U(y, x_g, \omega)$ in equation 7.5.20 will not produce anything additionally (the integration of a Dirac delta with the source outside is zero). The Green’s function $G(x, \omega)$ for the two Dirac deltas in equation 7.5.21 still satisfies the homogeneous equation equation. Here is the important property. Along the boundary $\Sigma$, $-\delta(x-x_g)$ produces the Green’s function $G(x', x, \omega)$, and below the boundary $\Sigma$, $\delta(x-x_g)$ produces the function $-G(x_g, x, \omega)$. (because the change of sign in the Dirac delta). Due to the symmetry between of the two Green functions with respect to the boundary, we have that

$$G(x, x_g', \omega) = G(x, x_g, \omega)$$

and so

$$G(x, x_g', \omega) - G(x, x_g, \omega) = 0.$$ 

So the Green function defined as

$$G(x, \omega) = G(x, x_g', \omega) - G(x, x_g, \omega).$$

Satisfies the first boundary condition. In the case of the normal derivative, due to the symmetry and the fact that the normal to points to the inside of the volume, the normal derivative of the Greens function associated with the source $G(x, x_g)$ is positive, (or negative) while the derivative of the Green’s function associated with the source $x'_g$ has the opposite sign. That is it is negative, (or positive). In any case the subtraction of the two derivatives will double one of them. We can say that (the sign, is not an issue) that

$$\frac{\partial G(x, \omega)}{\partial n} = 2 \frac{\partial G(x, x_g, \omega)}{\partial n} \neq 0$$

along the boundary $\Sigma$. It is not identically 0 because otherwise there would no sources, and we know that there is energy here.

The uniqueness theorem for the solution of partial differential equations assures that this is the unique solution. Hence in equation 7.5.20 can be written as

$$U(x_g, y, \omega) = \int_S dS \left( U \frac{\partial G}{\partial n} - G \frac{\partial U}{\partial n} \right) = -2 \int_S dS U(x, y, \omega) \frac{\partial G(x_g, x, \omega)}{\partial n} \quad (7.5.22)$$
as the solution of our problem, where we can say safely say that the integration over the surface \( \Sigma \) is zero. This is known in optics as the first Rayleigh-Sommerfeld solution. The second solution is obtained by choosing the sources as in the next problem:

\[
\nabla G(x_g, x, \omega) + \frac{\omega^2}{c^2} = -\delta(x - x_g) - \delta(x - x'_g) \quad (7.5.23)
\]

This will annihilate the normal derivatives along the boundary \( \Sigma \) and double the function \( G \) along the same boundary. We will use the first Rayleigh-Sommerfeld solution here.

It is interesting to see that the equation 7.5.22 can be derived by using the concept of angular spectrum (the \( k \) spectrum). Appendix D shows this derivation. The idea is that by knowing the \( k \) spectrum (along with the \( \omega \) spectrum) by just inverse Fourier transform we can recover the data. The derivation uses the method of stationary phase for surface integrals derived in Appendix D.C.

If we use the Green’s function associated with the subtraction of the two sources as implied by the equation 7.5.23, we would obtain instead

\[
U(x_g, y, \omega) = \int_S dS \left( U \frac{\partial G}{\partial n} - G \frac{\partial U}{\partial n} \right) = 2 \int_S G(x_g, x, \omega) \frac{\partial U(x, y, \omega)}{\partial n} \quad (7.5.24)
\]

which is known as the second Rayleigh-Sommerfeld solution.

The Kirchhoff Approximation: We believe that there is a historical misunderstanding about this name. Kirchhoff had the wrong boundary conditions to obtain a solution which was pretty good for small scattering angles. Sommerfeld and Rayleigh fixed Kirchhoff’s problem by choosing the method of images on the Green’s functions. Bleistein 1984 and Bleistein et. al., 2000 provide a new approximation. In their 2000 book, Bleistein et. al., they claim that the name “Kirchhoff Approximation” is suggested by the text of Goodman and the text of Born and Wolf (both referred here). We scanned for this and could not find a hint about the so called Kirchhoff approximation. This approximation has nothing to do with Kirchhoff. It has all to do with make the modeling formula consistent with ray tracing. We could call it, the Bleistein approximation, but history will prevail.

Here are the main steps assumed during the Kirchhoff approximation:

- **Redefine the volume of integration:** Bleistein et. al., do not appeal to the method of images for Green’s functions. Instead they redefine the volume as shown in Figure 7.5.3

We start all over with the Helmholtz–Kirchhoff theorem equation 7.5.20, over the surface shown in Figure 7.5.3. This problem is easier at this point because

---

44https://books.google.com/books?isbn=0080916953
45https://books.google.com/books?isbn=1461300010
we consider just two surfaces. The reflecting surface $S$ with the dotted black line (and extension), and the spherical cap. As the radius goes to infinity the integral (due to the Sommerfeld radiation conditions) goes to zero. Then we can rewrite the Helmholtz-Kirchhoff integral as

$$U(y, x_g, \omega) = \int_S dS \left( U \frac{\partial G}{\partial n} - G \frac{\partial U}{\partial n} \right)$$

where $S$ is the bottom surface.

- **Decompose the wavefield in incident, transmitted, and reflected** That is, we can write

$$U = u_I + u_S \quad \text{above the surface}$$
$$U = u_T \quad \text{below the surface}$$

With this, it is easy to see, since all the components of these wave fields satisfy the Helmholtz equation, that we can rewrite the Helmholtz-Kirchhoff integral as

$$U_s(y, x_g, \omega) = \int_S dS \left( u_S \frac{\partial G}{\partial n} - G \frac{\partial u_S}{\partial n} \right)$$

(7.5.25)
• **Boundary Conditions:** The Kirchhoff approximation assumes that the scattering wave field (the one we are interested on) \( u_S(x, x_g, \omega) \) is proportional to the incident wave field \( u_I(x, x_g, \omega) \), with \( R(x, x_g) \) being the constant of proportionality. In addition, along the reflecting boundary, since the direction of the incident and the scattering wavefields are opposite, we apply the approximation

\[
    u_S = Ru_I, \quad \frac{\partial u_S}{\partial n} = -R \frac{\partial u_I}{\partial n}
\]

along the reflecting surface \( S \).

• **Find the forward modeling integral:** With the conditions above, the integrand on equation 7.5.25 becomes

\[
    u_S \frac{\partial G}{\partial n} - G \frac{\partial u_S}{\partial n} = Ru_I \frac{\partial G}{\partial n} + R \frac{\partial u_I}{\partial n} = R \frac{\partial u_I G}{\partial n}.
\]

Then by defining the model as \( m(x_s, x_s, x_g) = R(x, x_s) \), the data as \( d(x_s, x_g, \omega) = u_S(x_s, x_g, \omega) \), and \( y = x_s \) we write

\[
    d(x_s, x_g, \omega) = \int_S dx \ m(x, x_s, x_g) \frac{\partial}{\partial n} u_I(x_s, x, \omega) G(x, x_g, \omega), \quad (7.5.26)
\]

where \( x_s \) is the source location, \( x_g \) is the receiver location, \( \omega \) is the temporal circular frequency, \( u_I \) is the incident wavefield at the reflector, \( n \) is a normal to the reflecting surface \( S \). The model parameter to invert is the reflectivity \( m(x_s, x_g, \omega) = R(x, \theta) \), with \( \theta \) the angle between the normal \( n \) to the reflector \( S \) and the incident ray.

This is sometimes called the Kirchhoff approximation formula\(^{46}\).

It has a great value, not only to provide seismic simulations, but it can be used to invert for reflectivity, as well as to check for inversion formulas using the method of stationary phase.

In the evaluation of this formula, the WKBJ approximation for incident and Green functions is assume. That is

\[
    u_I(x_s, x, \omega) \approx F(\omega) A(x_s, x) e^{i\omega r(x_s, x)} = F(\omega) G(x_s, x, \omega) \quad \text{and} \quad G(x_s, x_g, \omega) \approx A(x_s, x) e^{i\omega r(x_s, x_g)}.
\]

with \( F(\omega) \) the spectrum of the source signature. The WKBJ is a high frequency approximation. We now make another high frequency approximation to compute the normal derivative \( \partial / \partial n \). That is, when we take the normal derivative we retain only terms with the factor \( \omega \) in front.

CHAPTER 7. SEISMIC INVERSION IN THE GENERALIZED LEAST SQUARE (GLS) CONTEXT

For example, for the incident wavefield $u_I$, 
\[ \frac{\partial u_I(x_s, x, \omega)}{\partial n} \approx i\omega \nabla \tau(x, x_s) \cdot n u_I \]
and similarly for the normal derivative of the Green’s function $G(x, x_g)$. Then we can rewrite the integral 7.5.26 as
\[ d(x_s, x_g, \omega) = i\omega \int_S dx \ m(x, x_s, x_g) \left[ \nabla \tau(x_s, x) + \nabla \tau(x, x_g) \right] \cdot n \ u_I(x_s, x, \omega) G(x, x_g, \omega). \]
We recognize that
\[ \left[ \nabla \tau(x_s, x) + \nabla \tau(x, x_g) \right] \cdot n = p_s \cdot n + p_g \cdot n = \frac{\cos \theta_s}{c(x)} + \frac{\cos \theta_g}{c(x)} = \frac{\cos \theta_s + \cos \theta_g}{c(x)} \]
where $p_s$ and $p_g$ are the slowness vectors from the ray and receiver respectively at the reflection point $x$. The WKBJ assumption implies that the travel time function $\tau$ satisfies the Eikonal equation and this implies the last equality, where $\theta_s$ and $\theta_g$ are the angles that the direction rays make with the normal to the reflector. So the normal derivative in equation 7.5.26 is the obliquity factor. We can write this also in terms of wavenumbers. That is if $k_s = \omega p_s$, and $k = k_s + k_g$, then we can write the integral as
\[ d(x_s, x_g, \omega) = F(\omega) \int_S dx \ (k \cdot n) m(x, x_s, x_g) u_I(x_s, x, \omega) G(x, x_g, \omega). \]
where $k = \|k\|$. This vector $k$ is responsible for the loss of resolution in large offsets. As $\theta_s$ and $\theta_g$ approach $\pi/2$, $k$ approaches 0. The maximum resolution is obtained at $\theta_s = \theta_g$ which is zero offset. Inversion of zero offset (postack) is a much easier and inexpensive process. We recognize the similarity of this equation with the Born inversion formula 7.5.15, and the following important differences:

- In the Born forward modeling formula $m(x)$ is the perturbation of the inverse slowness. In the Kirchhoff formula above the model $m(x_s, x_g, x)$ is the reflectivity function.

- In the Born forward modeling formula, the model is independent of the recording geometry. In the Kirchhoff formula the model is dependent on the recording geometry. In this way we classify the Kirchhoff modeling formula as a pre-stack formula. However the Born modeling formula can be processed as prestack but the model will be redundant on each prestack inversion, except for aperture differences.
• The factor $k \cdot n$ is not present in the Born modeling formula. This makes the Kirchoff modeling formula more complicated since this factor depends on the source/receiver geometry.

• The Born formula is well suited to be computed with a FD algorithm since the Green functions are computed during the numerical wave propagation and not extra factors are needed. On the other hand the Kirchhoff modeling formula requires the computation of $k \cdot n$ which is not provided by a FD code. This needs to be computed using a ray tracing algorithm. The derivation of the asymptotic Kirchhoff inversion formula is presented in Bleisten et al. 47

47http://www.cwp.mines.edu/mmsimi
Chapter 8

Non-linear Inversion : Full Waveform Inversion (FWI)

8.1 introduction

The FWI inversion is rooted into the (ASM) Adjoint State Method technology. While numerical algorithms based on the (ASM) are becoming increasingly more popular in the geophysical community, the underlying mathematics is much less understood. The ideas behind the algorithm are hidden in complicated algebraic manipulations of differential equations. The purpose of this document is to provide an easy-to-understand introduction of the fundamentals of the ASM and show derivations based on implicit differentiation and Lagrange multipliers. Once understood, we demonstrate how to apply the ASM algorithm to Geophysical problems under more complicated and realistic situations.

When we take courses in Inversion Theory, we learn about the linear inversion problem

\[ Au = f \]  

(8.1.1)

where \( A \) is a matrix and \( u \) is a vector. At that time, we understood that if we knew \( A \) and \( f \) then we could find \( u \). If \( A \) is non-singular then

\[ u = A^{-1}f. \]

But in general \( A \) is not even a square matrix. After this we learned about the least square method to solve 8.1.1, and solutions based on the steepest descent, conjugate gradients and other methods.

We were taught that \( u \) belongs to the model space and \( f \) to the data space.

Now, working on a Full Waveform Inversion (FWI) problem we found that \( u \) is in the data space. Where is the model?

The problem that we are trying to solve here is a bit more interesting than the problem on equation 8.1.1. It is

\[ A(p)u = f \]  

(8.1.2)
where now $A(p)$ is a matrix function of some model parameters $p$. Now, in problem 8.1.2 the data space is the space of $u$’s and the model space is the space of $p$’s.

Whether $A$ is a matrix or a more general operator the problem in equation 8.1.2 in abstract is the same. We present the problem in two ways, the matrix version (section 8.2) and a differential-equation based version (section 8.3). From this presentation a general template is extracted that could be used in general representation, a generally-applicable template is extracted that can be applied to practical applications currently deployed by the seismic exploration community. Finally, some numerical computations demonstrate the validity of the method. Two appendices are included to make the tutorial self-contained. One on the Fréchet and Gâteaux derivatives, which might not be familiar to geophysicists, and another in adjoint operators. The results on both of the appendices are used frequently in the main part of the document.

According to Yedlin, [?] the ASM started with Lagrange as early as 1760, who introduced the adjoint operator. We show here how the method can be derived from Lagrange multipliers. Among many contributors to the invention of the method we acknowledge Maurice René Fréchet and René Gâteaux (see Appendix G which shows an important part of their contributions). The method as we know it today was developed by Marchuk [?].

In Geophysics, the first important work relevant to seismic exploration is by Lailly, 1983 in the Conference on Inverse Scattering Theory Memories

Lailly [?] in 1983. Lailly’s paper was framed in a mathematical language which is not well suited for geophysicists. Tarantola, 1984 [2] published the method of inversion using the ASM method in a simpler language. After this, it was clear that finding the gradient of descent was much like performing Reverse Time Migration (RTM) on residual data. However it took significant advances in computer science and hardware to make this approach practical. Only recently, more than 20 years after the first derivation of the seismic inversion problem, can we apply this compute-intensive method in both time and frequency domain implementations. Pratt, [?], provides a good overview of publications since Lailly’s original work. There are many publications more recent contributions, but in this context, we only mention Plessix [?], and Fichtner et. al., [?] who present some good theoretical foundations.

### 8.2 The Matrix Version

In this section, we show how obtain an approximate solution for equation 8.1.2 for $p$, provided we know $A$ and $f$ and a physical realization of the experiment that is captured by equation 8.1.2. The solution of this problem illustrates the ASM.

We focus in the problem

$$A(p)u = f$$

\[(8.2.3)\]
where \( f \) is a source term, \( u \) is a forward modeled field and \( A(p) \) is a modeling equation. For this document we assume that the scalars are all real (no complex). So we think of \( A(p) \) as a matrix in the space \( \mathbb{R}^{m \times n} \), \( u \in \mathbb{R}^n \), and \( f \in \mathbb{R}^m \). We assume that a physical system records data \( d \) which corresponds to this modeling experiment and we only have one instance of that data \( d \). We want to know how to find \( p \) with the knowledge of \( A, f \) and the data realization \( d \).

For example, we can say that \( A(p) \) is given in \( \mathbb{R}^{2 \times 2} \)

\[
A(p) = \begin{pmatrix}
3p_1 & \exp(p_2) \\
\log p_3 & 5
\end{pmatrix}
\]

and equation 8.2.3 takes the specific form

\[
\begin{pmatrix}
3p_1 & \exp(p_2) \\
\log p_3 & 5
\end{pmatrix}
\begin{pmatrix}
u_1 \\
u_2
\end{pmatrix} =
\begin{pmatrix}
f_1 \\
f_2
\end{pmatrix} 
\]

(8.2.4)

The question here is that without knowing \( u \), we try to find \( p = (p_1, p_2, p_3) \). At first glance, this looks hopeless, but recall that we also have some data measurements \( d = (d_1, d_2)^T \) obtained from the correct model \( p \). These data should be close to the forward modeled field \( u \), assuming the modeling equations captured in \( A \) are correct.

- Does this system has a solution for \( p \)?
- If it does, is it unique?

It is obvious that even in this particular toy example, the system is highly non-linear as a function of \( p \). It is a short toy matrix. We observe that if a system is linear, then there would be a unique solution (non–singular matrix of coefficients), no solution (the right term is not in the range of the matrix) or an infinite number of solutions (the null space has dimension 1 or larger). Geometrically, planes can intersect along one point (like a corner in an office room); in a line, like an edge in an office room, or not intersect at all (like two parallel walls in an office room). However when the equation is non–linear, the planes and lines turn to curved surfaces and wiggles. These objects can intersect in a finite discrete set of points, in only 1 point, in a discrete infinite set of points or in a continuum set of points. The concept of existence and uniqueness is not obvious in this context. Non–linear equations are not as well understood as linear equations.

The problem is more complex when the dimension of the spaces under consideration grow. Typical geophysical problems deal with vector spaces of dimensions of the order of millions and more.

For problems this complicated, unfortunately, we only can say that since we have some instance \( d \) of a physical realization, we can play with the \( u \)'s and find some parameters \( p \) which are consistent, up to a “small” error, with system 8.2.3.

In this way we want to set up an optimization problem. The most obvious optimization problem is the least square problem where the objective (cost) function is
given by
\[ C(p) = \frac{1}{2} \| u(p) - d, u(p) - d \|^2 = \frac{1}{2} \langle u(p) - d, u(p) - d \rangle. \] (8.2.5)

where it is obvious that \( u \) depends (implicitly) on \( p \) in equation 8.2.3 and the data space is a Hilbert space where we can take inner products.

So, the most we can do (assuming no other constraints) is to minimize the objective function \( C(p) \). But the question here is

- Is the objective function \( C(p) \) convex?

Convexity of the objective function translates into uniqueness of the solution (which would be good news). Unfortunately since in general \( u \) is non-linear as function of \( p \), then we can foresee that \( C(p) \) is highly non-linear as a function of \( p \). Non-linearity creates wiggles with peaks and valleys. Clearly \( C(p) \) is now non-convex and does not have an obvious unique minimum. Additional constraints help to reduce the space of solutions. One special situation we want to comment on is a sinusoidal dependency of the modeling equation on a parameter \( p \), for example terms such as \( \sin \omega p_i \). This is bad news, because sinusoidal functions wiggle for ever and could oscillate very rapidly (for high frequencies). Fortunately for our geophysical applications the parameters in the analytical representations are low-order polynomials or simple algebraic multipliers or dividers.

From calculus, if we want to find a minimum, we take the gradient and make it zero. That is, we find the Fréchet derivative (see Appendix G) of the cost function 8.2.5 and make it zero, or
\[ \frac{dC(p)}{dp} = \nabla_p C(p) = \langle u_p , u - d \rangle = 0. \]

with \( u_p = du/dp \).

However in practice the data are noisy, the modeling equations inadequately describe the physical system and the computer evaluations have limited accuracy. In other words, we cannot match the data perfectly to the modeled field \( u \). Besides, we do not know \( p \) to start with, and if we use an initial guess of \( p_0 \), the chances to get this \( p_0 \) to fit the data are small, so the best we can do is a descent method, where we start somewhere along the hypersurface representing the cost function and descent toward the minimum. If that minimum happens to be the wrong local minimum we are out of luck and should perturb the initial guess \( p_0 \) to move to another convex zone. Of course, this assumes that the surface (or hypersurface) does not oscillate too rapidly as for example close to zero for a function of the form \( \sin(1/x) \). Fortunately, this is typically not the case.

Once the gradient \( \nabla_p C(p_0) \) is found, then it can be used to move our initial guess \( p_0 \) to a “closer” point \( p_1 \) by using optimized descent methods which are outside of the scope of this document. We will only focus on the gradient and the process of how to obtain it.
So, let us forget about the gradient being zero and compute

\[
\frac{dC(p)}{dp} = \nabla_p C(p) = \langle u_p, u - d \rangle.
\]

(8.2.6)

To find the gradient, we need to know \( u_p \). We can find \( u_p \) in two different ways

- Find \( u \) from equation 8.2.3. That is, if \( A(p) \) is non–singular ( if \( A^{-1}(p) \) exists)

\[
u = A(p)^{-1} f
\]

and from here take the derivative of \( du/dp \). However \( A(p) \) could be singular, so we have no way to directly compute \( u_p \). In wave propagation we can find \( u \) as a solution of the wave equation using finite differences or ray tracing methods. However \( u \) has a domain which in practice is a huge grid and each grid point allocates a component of the parameter vector \( p \). If we want to compute the gradient, we need to know the change along each individual cell, but this change needs to be computed by modeling at least two instances (for the central finite difference star) per cell. A 2D velocity model of \( 1000 \times 1000 \) a million cells will need 2 million modeling experiments. New parameters (such as density \( \rho \) for example) will increase the size of these computations linearly with the number of parameters. A 3D experiment would be prohibitively expensive. We need to find alternative methods to find the gradient and this is the main motivation behind the ASM. Which we illustrate next.

- Take the implicit derivative in 8.2.3. This should always work. That is

\[
A_p u + A u_p = 0,
\]

so

\[
A u_p = -A_p u.
\]

In this equation we are free to take inner product with whatever function we want in the Hilbert space of \( u \)'s. That is, we can think about a “test” function \( v \) such that

\[
\langle Au_p, v \rangle = \langle -A_p u, v \rangle
\]

and if the adjoint of \( A \) exists (it is the transposed in the case of real matrices), and we call it \( A^* \) then

\[
\langle u_p, A^* v \rangle = \langle -A_p u, v \rangle
\]

(8.2.7)

So, by looking to the gradient in equation 8.2.6, we reduce the problem to
(i) Find $u$ such that
\[ Au = f. \]

The forward propagating problem.

(ii) Find $v$ such that
\[ A^*v = u - d \]  
\[ (8.2.8) \]

The backwards propagating problem. This is the key step in the Adjoint State method. If we can solve this system, then we can completely avoid the computation of the costly derivative $u_p$. Instead, we evaluate the right side of equation $8.2.7$

(iii) and apply this $v$ to
\[ \langle -A_p u, v \rangle. \]  
\[ (8.2.9) \]

Note that since $p = (p_1, \ldots, p_n)$ is a vector, we have that
\[
\nabla_p A = A_p = \begin{pmatrix} A_{p_1} \\ A_{p_2} \\ \vdots \\ A_{p_n} \end{pmatrix}
\]

Here only two systems of the same order of the system $Au = f$ have to be solved (instead of $2$ million for the $1000 \times 1000$ 2D velocity model. Note that the least amount of terms to compute a derivative using central finite differences is $2$.). In section 8.5 we show an implementation of the algorithm using the acoustic wave equation with constant density which demonstrates numerically the validity of the method.

In example 8.2.4, all the elements to compute $du/dp$ are easy to find. For example
\[
A_{p_2} = \begin{pmatrix} 0 & \exp(p_2) \\ 0 & 0 \end{pmatrix}
\]

and
\[
A^*(p) = \begin{pmatrix} 3p_1 & \log p_3 \\ \exp(p_2) & 5 \end{pmatrix}
\]

So we need to solve first for $v$ in
\[ A^*(p)u = \begin{pmatrix} 3p_1 & \log p_3 \\ \exp(p_2) & 5 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} u_1 - d_1 \\ u_2 - d_2 \end{pmatrix} \]

and then compute the gradient as
\[
\frac{\partial C}{\partial p_i} = \langle -A_p u, v \rangle.
\]
8.2 The Matrix Version

8.2.1 The Lagrange Multipliers Approach

We can write the optimization problem as follows:

\[
\minimize \text{ for } p \\
C_L(p, \lambda) = \alpha \langle u(p) - d, u(p) - d \rangle + \beta \langle L(p), \lambda \rangle 
\]  

(8.2.10)

\[
L(p) = A(p)u - f \equiv 0. 
\]  

(8.2.11)

Since \( L(p) \equiv 0 \), \( \beta \) and \( \lambda \) could have any value. Now, any value of \( p \), which minimizes a function, should minimize that function when scaled by \( \alpha > 0 \). So we are not changing the solution by imposing the constants \( \alpha > 0 \) and \( \beta \). However, as we show below, we are changing the gradient (up to a scale factor).

Then, the stationary condition on the extended objective function implies that

\[
\frac{\partial C_L}{\partial p} = 0, \quad \frac{\partial C_L}{\partial \lambda} = 0. 
\]

The right equation does not provide any new information. It just produces \( \lambda \equiv 0 \) which we already know. From the left equation we find

\[
\alpha \langle u_p, u - d \rangle + \beta \langle Apu, \lambda \rangle + \beta \langle Au_p, \lambda \rangle = 0. 
\]  

(8.2.12)

Because we have freedom to choose \( \lambda \), we can choose \( \lambda \) such that

\[
A^* \lambda = u - d 
\]  

(8.2.13)

then from the previous two equations:

\[
0 = \alpha \langle u_p, u - d \rangle + \beta \langle Apu, \lambda \rangle + \beta \langle Au_p, A^* \lambda \rangle \\
= (\alpha + \beta) \langle u_p, u - d \rangle + \beta \langle Apu, \lambda \rangle
\]

so

\[
\langle u_p, u - d \rangle = -\frac{\alpha}{\alpha + \beta} \langle Apu, \lambda \rangle. 
\]  

(8.2.14)

When \( \alpha = 1/2, \beta = 0 \) we have the original optimization problem, and this equation matches equation 8.2.9. Different combinations of \( \alpha > 0 \) and \( \beta \) produces different gradients but all of them have the same direction and different magnitude. In this sense the Lagrange multipliers analysis sets up a more general framework for the same problem which leads to a gamma of gradients.

In summary, the algorithm is as follows:
(i) Solve equation 8.2.11

\[ Au = f. \]

(ii) Solve the adjoint state equation 8.2.13

\[ A^*\lambda = u - d, \]

for \( \lambda \) and

(iii) find

\[ -\frac{\alpha}{\alpha + \beta} \langle A_pu, \lambda \rangle, \quad (8.2.15) \]

which according to equation 8.2.14 is the same as

\[ \langle u_p, u - d \rangle = \nabla_p C \]

So we found that the Lagrange multipliers \( \lambda \) correspond with the adjoint state variable \( v \) above. By comparing equations 8.2.15 and 8.2.9. We see that 8.2.9 can be obtained by simply choosing \( \alpha = 1/2, \beta = 0 \) in the cost function \( C_L \) defined in equation 8.2.10.

In what follows we will use the optimization function 8.2.10 with \( \alpha = 1/2 \) and \( \beta = 0 \).

In practice Lagrange multipliers are used when additional constraints are added to the cost function. For example constraints on smoothness, mask functions or preconditioning; such as for example, adding well logs information to the seismic (Wang et. at., [?]).

### 8.3 The Differential Equation Version

#### 8.3.1 The ASM by Example

Here is the simplest example we can think of. Imagine we have a machine that can model some data \( d \) provided some source \( s \) on an object with some unknown parameters \( p \). Let us assume that the experiment can be mathematically described by the equation

\[ p(x) \frac{du(x)}{dx} = s(x). \quad (8.3.16) \]

with boundary conditions

\[ u(a) = u(b) = 0. \]
where \( a < x < b \).

Now, whatever the physics behind this toy problem, the idea is to perturb the parameter \( p \) so that with one instance of the measured data \( d \) we can estimate the actual \( p \) in the real model. This is an optimization problem. We define a simple objective (cost) function

\[
C(p) = \frac{1}{2} \| u(p) - d \|^2 = \frac{1}{2} \langle u(p) - d, u(p) - d \rangle.
\]

Which represents the misfit between the acquired data \( d \) and the synthetic data \( u(p) \) for a model with parameters \( p \). Note that we eliminate the dependence on \( x \) and show that \( u \) will depend on the model parameter \( p \), through the differential equation 8.3.16. This does not mean that \( u \) does not depend on \( x \) but the optimization is with respect to \( p \) and not with respect to \( x \).

Optimization problems are seen usually as minimization problems. We want to minimize the cost function \( C(p) \) as a function of \( p \). If we can get \( C(p) = 0 \) that would ideal, we have found a set of parameters \( p \) such that they model the data perfectly. In reality this does not happen, so we start with some initial parameter \( p_0 \) and through a gradient technique (the gradient of \( C(p) \)) perturb it as a sequence of linear perturbations until find the best match (the minimum of \( C(p) \)).

Note that \( C \) is a functional. That is, \( C \) is a function from the parameter space of functions to the real numbers. Here

\[
p : A = [a, b] \subset \mathbb{R} \rightarrow \mathbb{R}
\]

\[
x \mapsto p(x).
\]

For example \( p \) could represent the linear density along a string, and

\[
\mathcal{C} : U \rightarrow \mathbb{R}
\]

\[
p \mapsto C(p)
\]

where \( U \) is the space of \( p \) functions of a one–dimensional real variable into the reals. So \( \mathcal{C} \) lies in a space of functionals. Taking the derivative of \( u \) with respect to a point \( p \in \mathcal{C} \) is not a common derivative with respect to a simple variable in \( \mathbb{R} \). That is why we need the Fréchet derivative extension in Banach spaces. Appendix G shows some basic theory on Fréchet and Gâteaux derivatives, for which we will refer often in this document and are not assumed to be known by geophysicists.

We want to evaluate, the Fréchet derivative

\[
DC(p) = \frac{dC}{dp} = \frac{1}{2} \langle u_p , u - d \rangle + \frac{1}{2} \langle u - d , u_p \rangle = \langle u_p , u - d \rangle
\]

(8.3.17)

with

\[
u_p = \frac{\partial u}{\partial p}
\]
as the Gâteaux derivative of \( u \) seen as a function of \( p \), and we use commutation of the inner product of functionals in the real space (in the complex space we would need to conjugate numbers, but again, we want to simplify as much as possible).

So the evaluation of the gradient rest upon the evaluation of the Gâteaux derivative \( u_p \) at each point of the model. This operation (if computed by finite differences for example) is very expensive because the model space is usually large and we need to solve for \( u \) every time. Section 8.5 illustrates the details about why the computation of the Gâteaux derivative is so expensive.

The implicit derivative is a powerful tool in calculus because we can find derivatives in complicated relations (equations) without having to solve for the dependent variable \( (y) \) as a function of the independent variable \( (x) \). If solving algebraic equations could be hard, think about solving differential equations. This is precisely the point. We will find the derivative \( u_p \) without having to solve the differential equation for \( u \) and then take the derivative of this \( u \). We look at \( u \) as a function of \( p \) where \( p \) is seen as a real variable (when indeed it is a function) and since the definition of the Gâteaux and Fréchet derivatives has the same functional structure as those in real variables, we can use all the nice properties that we learned in calculus about taking implicit derivatives including the chain rule, the product rule, etc.

Then, to find \( u_p \), we use equation 8.3.16 and take implicit differentiation by defining

\[
f(p) = p \frac{du(p)}{dx} - s = 0.\]

So

\[
\frac{df}{dp} = \frac{du(p)}{dx} + p \frac{du_p(p)}{dx} = 0 \tag{8.3.18}
\]

As in implicit differentiation in algebraic equations, we still have to solve an equation after the implicit differentiation and, there, we hoped for an easier equation. Here we have to solve equation 8.3.18 for \( u_p \), and it is apparent that we converted a relatively simple problem into a more complicated one. Here is where the ASM plays the protagonist role.

Let us assume a differentiable function \( \phi = \phi(x) \). This function is the adjoint state. We take the inner product of equation 8.3.18 with \( \phi(x) \) and obtain

\[
\left\langle \frac{du(p)(x)}{dx} , \phi(x) \right\rangle + \left\langle p \frac{du_p(p)(x)}{dx} , \phi(x) \right\rangle = 0,
\]

\(^3\)Note that we choose the partial derivative sign instead of the total derivative. Not, because \( u \) depends on \( p \) and \( x \). They leave in different spaces. The reason is that in practice \( p \) is a vector of parameters and we should think about partial derivatives with respect to \( p_i \). In addition we assume that we are operating in the field of real numbers. Otherwise the equation above would be

\[
DC(p) = \text{Re}(u_p , u - d)
\]

where “\text{Re}” means the real part of the complex number it acts on.
8.3 The Differential Equation Version 159

where now we look at $u(p)(x)$ as a function of $x$ for each parameter $p$. Here we will drop the argument $p$ when referring to $u$ and to $u_p$. The inner product in function spaces is an integral with respect to $x$. Doing integration by parts (this is the way to find the adjoint) along the interval $[a, b]$, (and since $u(a) = u(b) = 0$ ) we find

$$-\left\langle u(x), \frac{d\phi(x)}{dx} \right\rangle - \left\langle u_p(x), \frac{d}{dx}[p(x) \phi(x)] \right\rangle = 0$$

(8.3.19)

Since the function $\phi(x)$ is differentiable, but other than that, arbitrary; we can assume that there is some $\phi$ which satisfies the differential equation

$$-\frac{d}{dx}[p(x) \phi(x)] = u(x) - d,$$

Now from equation 8.3.19, finally we have an explicit equation in $u_p$. Even better, compare the second term on this equation with the gradient of the cost function 8.3.17 and we find

$$DC(p) = \nabla_P C = \langle u_p, u - d \rangle = -\left\langle u, \frac{d\phi}{dx} \right\rangle.$$ 

(8.3.20)

In conclusion, the gradient of the objective function is the inner product (zero lag cross–correlation) of the solution of the differential equation with the derivative of the solution $\phi$ of the equation

$$-\frac{d}{dx}[p(x) \phi(x)] = u(x) - d,$$

We see that for the differential operator $Lu = p(x)du(x)/dx$, with the operator $L(p) = p$, the adjoint operator is $L^*u = -d[u(x)p(x)]/dx$, so the factor function $\phi$ is determined by solving the problem

$$L^*\phi(x) = u(x) - d.$$ 

(the adjoint state equation)

Now, the operator acting on $\phi = \phi(x)$ is $d/dx$ (for this toy problem), which in general is

$$d/dx = dL/dp,$$

so in this case we can write that the ASM

$$L(p)u = s$$

in three steps

- Solve the adjoint equation $L^*\phi(x) = u(x) - d$
- find the solution of $Lu = s$
- Find the inner product $-\left\langle u, \frac{dL}{dp}\phi \right\rangle$.

We show next that this is the case in general.
8.3.2 The Problem in Abstract

The problem can be divided in several subproblems

- Set up the modeling equation with boundary conditions.
- Define the objective function.
- Find the gradient of the objective function in several steps.
  - Take implicit differentiation.
  - Instead of solving explicit for the partial derivatives in a complicated implicit equation, use the adjoint method to move the differential operators into test functions and so uncover the desired operator \( u_p \) from its derivatives.

We will follow those steps in what follows.

8.3.3 The modeling equations

The problem is posted by some differential equation, call it

\[
Lu(x) = s(x),
\]

where \( L \) is

\[
L \equiv L \left( (\partial_x)^{(k)}, p(x) \right),
\]

Here \( k \), the order of the derivative, can be any positive integer, but for the wave equation it is \( k = 2 \), \( p(x) \) is a vector of scalar functions representing the model parameters, and \( s(x) \) is a source term. The vector \( x \) contains the spatial coordinates and time; that is, \( x \) is located in the \( n + 1 \) dimensional real space \( \mathbb{R}^{n+1} \).

This differential equation is subjected to some boundary and initial conditions

\[
B_i(\Sigma(x)) = 0
\]

\( i = 1, \ldots n \), where \( \Sigma \) is some manifold wrapping the domain volume, in time and space and \( B_i \) is a function of the first \( n \) partial derivatives and \( p(x) \). That is,

\[
B_i = B_i(\partial_x^i, p(x)).
\]

8.3.4 The Objective Function

Equation 8.3.21 implicitly defines \( u \) as a function of the vector of parameter scalar functions \( p \). That is, \( u = u(p) \).
Since we know the modeling problem \( Lu = s \), we can create synthetic data \( u \) and match it with some acquired data \( d \) in order to minimize the mismatch error. We choose the following simple objective (cost) function
\[
C(p) = \frac{1}{2} \langle u(p) - d, u(p) - d \rangle.
\]
which solves the least squares minimization problem. In practice the data \( d \) are limited in many aspects. For example:

- To fit the problem, data should be recorded in a closed volume. In the case of surface seismic, data are acquired in only one of 6 faces of an imaginary rectangular prism of earth. This lack of aperture reduces spatial resolution on the seismic images.

- Even if data are collected in a closed surface, they are collected in samples and no continuously. Loss of resolution due to coarse sampling is an inherent problem to any inversion. It can be addressed by constraints and interative application, but this is out–of–scope for this tutorial.

- The amount of data could be insufficient to solve for the parameters \( p \). This could increase the space of possible solutions. To remedy this problem, additional constraints should be added. These could be added by including a weight into the norm definition. This is the role of the covariance matrix into the least squares weight for matrices. Preconditioning in the form of additional terms into the cost function further could reduce the space of solutions. For example adding smoothness to the solution implies simultaneous minimization of the function and its derivatives up to a certain order. That is, we think about a norm in a Sobolev space. This new problem with constraints can be seen as a Lagrange Multiplier’s problem. We discussed this in section 8.2.1.

To find the minimum usually a descent method is used, and this implies finding the Fréchet derivative of \( C \) as a function of the parameter vector \( p \). Since the cost function is a functional (its range is in the real numbers) this Fréchet derivative is a gradient.

8.3.5 The Gradient of the Objective Function

\[
DC(p) = \nabla_p C(p) = \langle u_p, u(p) - d \rangle,
\]
where we used the rules of differentiation for inner products. The problem here is that we need to find \( u_p \) and a naive approach could be to solve the differential equation \( Lu = s \) for \( p \) and then take the Gâteux derivative of \( u \) as a function of \( p \). As discussed above, this could be a very expensive process due to the large dimension of the parameter space for practical problems.

We will take implicit differentiation of the objective function, which does not imply to solve any system.
Implicit Differentiation

To compute $u_p(p)$ we build the following operator

$$F(u(p), p) = Lu - s \equiv 0$$

and take the total (Fréchet) derivative of $F$ with respect to $p$. That is

$$\frac{dF}{dp} = \frac{\partial L}{\partial p} u + L u_p = 0.$$ (8.3.23)

So we see that $u_p$ is under the operator $L$. To remove the operator $L$ from $u_p$, we use the ASM.

The Adjoint Method

To find the gradient of the objective function, we take the inner product of equation 8.3.23 with a differentiable function $\phi = \phi(x)$. That is

$$\left\langle \frac{\partial L}{\partial p} u, \phi \right\rangle + \left\langle Lu_p, \phi \right\rangle = 0,$$

and from the definition of adjoint applied to the second term on the left, we rewrite this equation as

$$\left\langle \frac{\partial L}{\partial p} u, \phi \right\rangle + \left\langle u_p, L^* \phi \right\rangle = 0.$$ (8.3.24)

where $L^*$ is the adjoint of the operator $L$.

The key trick now is to specify a test function. Since, to find the gradient in equation 8.3.22 we need to evaluate

$$\left\langle u_p, u(p) - d \right\rangle$$

by finding $\phi$ such that

$$L^* \phi = u(p) - d,$$

and from equation 8.3.24 we see that

$$DC(p) = \nabla_p C(p) = \left\langle u_p, u(p) - d \right\rangle = -\left\langle \frac{\partial L}{\partial p} u, \phi \right\rangle$$

So the recipe to find the gradient is as follows
8.4 Examples

8.3.5.0.1 Template

(i) Find the adjoint $L^*$ and from this $\phi$ such that

$$L^* \phi = u(p) - d \quad (8.3.25)$$

This, in waveform inversion, is the reverse propagation from the residual between the synthetic data $u$ and the acquired data $d$.

(ii) Find the forward solution $u$ of the equation

$$Lu = s$$

and then compute

$$\frac{\partial L}{\partial p} u.$$  

(iii) Find the negative of the zero lag cross-correlation between the forward field solution under the operator $(\partial L/\partial p)$ and the adjoint state variable $\phi$. That is

$$DC(p) = \nabla_p C(p) = -\left\langle \frac{\partial L}{\partial p} u , \phi \right\rangle. \quad (8.3.26)$$

In the next section we show a few classical examples and how they can be solved in a simple way by applying the template 8.3.5.0.1.

8.4 Examples

In all the examples shown here, we follow the same template shown in the section 8.3.5.0.1.

8.4.1 The homogeneous–isotropic wave equation

Let us assume density $\rho = 1$, and write the wave equation as

$$Lu = \left( \frac{1}{v^2} \frac{\partial}{\partial t^2} - \nabla^2 \right) u = s.$$  

We should write the corresponding equations with 8.3.25 and 8.3.26. Here $p = v$. We impose the boundary and initial conditions

$$u(x, 0) = 0 \quad \frac{\partial u(x, 0)}{\partial t} = 0$$

and the Sommerfeld radiation conditions

$$u(x, t) \bigg|_{x \to \infty} \to 0.$$
(i) The operator

\[ L = \frac{1}{v^2} \frac{\partial}{\partial t^2} \nabla^2 \]

is self-adjoint since \( v \) is independent of time, so \( L = L^* \). Now we should find \( \phi(x, t) \) such that

\[ L^* \phi(x, t) = u(x, t) - d(x, t) \]

(ii) Step two is

\[ \frac{\partial L}{\partial v} = -\frac{2}{v^3} \frac{\partial}{\partial t^2} \]

and step three

(iii)

\[ DC(v) = \nabla \cdot C(v) = -\left\langle \frac{\partial L}{\partial v} u, \phi(x, t) \right\rangle = \frac{2}{v^3} \left\langle \frac{\partial^2 u}{\partial t^2}, \phi(x, t) \right\rangle. \]

8.4.2 Tarantola’s 1984 inversion for \( K, \rho, \) and source \( s \)

Tarantola [?] uses the acoustic wave equation

\[ Lu = \left[ \frac{1}{K(x)} \frac{\partial^2}{\partial t^2} - \nabla \cdot \left( \frac{1}{\rho(x)} \nabla \right) \right] u(x, t) = s(x, t). \]

Where we impose the boundary and initial conditions

\[ u(x, 0) = 0 \]
\[ \frac{\partial u(x, 0)}{\partial t} = 0 \]

and the Sommerfeld radiation conditions

\[ u(x, t) \bigg|_{x \to \infty} \to 0. \]

(i) As we showed in equation H.0.14 the operator \( L \) is self-adjoint. That is,

\[ L = L^* \]

From the point of view of analysis, self-adjoint operators have great advantages over non-self adjoint operators. For example the eigenvalues are real and the
Examples

operator can be diagonalized. However, this is aside from the flow in this document. From the point of view of numerical implementations self–adjoint operators have a great advantage, since the same code used to compute the forward wavefield through $L$ can be recycled to use the backward propagation wavefield through $L^*$.

Solve for

$$L^* \phi = L \phi = u(x, t) - d(x, t).$$

(ii) Step two: Find $u(x, t)$ such

$$Lu(x, t) = s(x, t).$$

The gradient of $L$ with respect to the model parameters.

$$\frac{\partial L}{\partial K} = -\frac{1}{K^2} \frac{\partial^2}{\partial t^2}$$

$$\frac{\partial L}{\partial \rho} = \frac{\partial}{\partial \rho} \left[ \nabla \left( \frac{1}{\rho} \right) \cdot \nabla \right] = -\nabla \frac{1}{\rho^2} \cdot \nabla$$

(iii) Step three:

$$DC(v) = \nabla v C(v) = \begin{pmatrix} C_K \\ C_\rho \end{pmatrix}$$

With

$$C_K = \left\langle \frac{\partial L}{\partial K} u, \phi(x, t) \right\rangle = -\frac{1}{K^2} \left\langle \frac{\partial^2 u}{\partial t^2}, \phi(x, t) \right\rangle$$ (8.4.27)

$$C_\rho = \left\langle \frac{\partial L}{\partial \rho} u, \phi(x, t) \right\rangle = -\left\langle \nabla \frac{1}{\rho^2} \cdot \nabla u, \phi(x, t) \right\rangle = -\left\langle u, \nabla \frac{1}{\rho^2} \cdot \nabla \phi \right\rangle$$ (8.4.28)

Where in the last equality we used the fact that the operator acting on $u$ is self–adjoint as shown in Appendix H. See table H.1 line 4 (labeled “Mixed”).

Equations 8.4.27 and 8.4.28 match the first two equation of Tarantola [?] (see his equation (A-6)) except that in Tarantola the density factor $1/\rho^2$ is outside of the $\nabla$ operator and in my case it is inside the $\nabla$ operator. Either Tarantola or myself have an error here.
8.4.2.0.1 Inversion for Source Signature: Tarantola also inverts for source signature. To invert for source signature we redefine the problem as follows:

\[
L(K, \rho)(u) - s = \left[ \frac{1}{K(x)} \frac{\partial^2}{\partial t^2} - \nabla \cdot \left( \frac{1}{\rho(x)} \nabla \right) \right] u(x, t) - s(x, t) \equiv 0. \tag{8.4.29}
\]

We assume that

\[
s(x, t) = \delta(x - x_s)w(t)
\]

where \(w(t)\) is some wavelet that we want to invert for. We rewrite equation 8.4.29 as

\[
\left[ \frac{1}{K(x)} \frac{\partial^2}{\partial t^2} - \nabla \cdot \left( \frac{1}{\rho(x)} \nabla \right) \right] u(x, t) - \delta(x - x_s)w(t) \equiv 0.
\]

Doing implicit differentiation in this equation with respect to \(w\) we find

\[
\left[ \frac{1}{K(x)} \frac{\partial^2}{\partial t^2} - \nabla \cdot \left( \frac{1}{\rho(x)} \nabla \right) \right] u_w - \delta(x - x_s) = 0
\]

As done before, we take inner product with a differentiable function \(\phi\)

\[
\langle Lu_w(x, t), \phi(x, t) \rangle = \langle \delta(x - x_s), \phi(x, t) \rangle = \phi(x_s, t).
\]

We write

\[
\langle u_w(x, t), L^* \phi(x, t) \rangle = \phi(x_0, t)
\]

So the gradient with respect to the source signature \(w = w(t)\) is given by

\[
DC(w) = \nabla_w \mathcal{C} = \langle u_w, u(w) - d \rangle = \phi(x_s, t)
\]

where we assume that

\[
L^* \phi(x, t) = u(x, t) - d(x, t). \tag{8.4.30}
\]

So to find the source signature, we should just solve the adjoint equation 8.4.30 and evaluate the solution at \(x = x_0\). In fact this makes sense since equation 8.4.30 is back–propagating the wavefield from the receivers and collapsing at the source location \(x_s\). No modeling is required (as in RTM) because the modeling implies some source and we do not care about a synthetic source which has nothing to do with the field data.
Since the field $\phi$ in all cases is the solution to the wave operator

$$L^* \phi(x, t) = u(x, t) - d(x, t)$$

The inversion for all parameters could be done simultaneously. The cross-correlation or signature extraction of the field changes on each case and could be done individually to build the global gradient with respect to all model parameters. The perturbation could be done individually (for each parameter) or simultaneously. These are details that pertain to the code implementation and will not be discussed here.

### 8.4.3 Elastic Wave Equation

The general anisotropic elastic wave equation is represented by the operators

$$L_{ik} u_k = \left[ \rho \frac{\partial^2}{\partial t^2} - \frac{\partial}{\partial x_j} \left( c_{ijkl} \frac{\partial}{\partial x_l} \right) \right] u_k = f_i$$

Where we can label $L_{ik}$ as the operator acting on the wavefield component $u_k$, excited by the source function $f_i$. Usually the source is written in terms of Dirac delta distributions, such as for example $A_i \delta(t - t_0) \delta(x - x_s)$ for a point source starting at time $t = t_0$ and located at the point $x = x_s$. Sometimes, instead of $\delta(t - t)$ we find the use of wavelets (Ricker or other).

The arguments for the wavefield $u$ are $x$ and $t$ and the arguments for the density $\rho$ and the stiffness tensor components $c_{ijkl}$ are $x$. We assume Einstein’s notation of sum over repeated indices. For the derivation of this equation see Aki and Richards [?].

For the adjoint operator we use zero boundary conditions

$$u_k(x, 0) = 0$$
$$\frac{\partial u_k(x, 0)}{\partial t} = 0$$

and the Sommerfeld radiation conditions

$$u_k(x, t) \bigg|_{x \to \infty} \rightarrow 0.$$

Before we apply the template 8.3.5.0.1 to solve the problem let us note that $L_{ik}$ is really a $3 \times 3$ matrix of operators. Each operator $L_{ik}$ solves a problem of a source $f_i$ polarized in some direction $e_i$, for a receiver $u_k$ recorded with a sensor responding to another direction $e_k$. The $u_k$ are known as 9-component data, because the 9 possible combinations. Now, due to reciprocity only 6 combinations are independent. There should be a total of 9 data sets that we can use to compute the gradient. For the
description below, each data set could be treated as a separated problem. Having a
gradient for the 6 possible independent combinations of source and receiver polarization
directions is still an expensive process in terms of the dimension of the combined
vector space for all components. The signal–to–noise ratio of all data sets is different.
Usually the best signal to ratio is for vertical components both at the source and at the
receiver location. If done individually, then we can find 9 different gradients with respect
to the same parameters. Indeed each cost function is different, because both the
data and the modeling equations are different. Hence we do not have to expect the
same gradients. However after updating the parameters they should be close to each
other in all the 9 set of parameters. Redundancy is statistically good, but the stacking
of those might have to be done with a weighted average. A problem of averages is
that they tend to eliminate the high frequencies and in WFI we want to enhance the
high frequencies of the parameters. Hence, even though the one–by–one treatment
is easier to understand, to formulate and perhaps to implement, we will treat all the
$L_{ij}$s as a part of a single operator $L$.

We write the problem in matrix form as

$$Lu = \begin{pmatrix} L_{11} & L_{12} & L_{13} \\ L_{21} & L_{22} & L_{23} \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} = f$$

This is an interesting problem because involves the matrix approach and the differ-
tential operator approach simultaneously. Here the inner product is both discrete and
continuous. That is, for example for two functions $f$ and $g$ in the space of solutions,
the inner product is given by

$$\langle f, g \rangle = \int_V f_i g_i \, dx \, dt$$

(see equation H.0.7), where the sum along the dummy $i$ index is from 1 to 3. Let us
choose a test function $\Phi(x, t) = (\phi_1(x, t), \phi_2(x, t), \phi_3(x, t))^T$.

$$\langle Lu, \Phi \rangle = \langle L_{ik} u_k, \phi_i \rangle = \langle u_k, L^*_k \phi_i \rangle = \langle u, L^* \phi \rangle$$

where

$$(L^*)_{ik} = L^*_{ki} = L_{ki}. \quad \text{(see table H.1)}$$

Note that $L_{ik}$ does not have to be equal to $L_{ki}$ since $c_{ijkl}$ is not necessarily equal to $c_{kjil}$. However from the properties shown in appendix B we see that $L_{ik}$ is self–adjoint. That is

$$L_{ik} = L^*_{ik}.$$ 

Let us apply the template:
(i) Find $\phi_k = \phi_k(x, t), k = 1, 2, 3$ such that

$$L_{ik}^* \phi_i = L_{ik} \phi_i = u_k(x, t) - d(x, t).$$

(ii) Find $u_k(x, t)$ such that

$$L_{ik} u_k(x, t) = f_i(x, t).$$

The gradient of $L_{ik}$ with respect to the model parameter $\rho$ is given by

$$\frac{\partial L_{ik}}{\partial \rho} = \frac{\partial}{\partial t^2}$$

Next, we want to find the gradient with respect to the stiffness parameters $c_{ijkl}$. If the $c_{ijkl}$ were mutually independent then the operator $\partial L_{ik}/\partial c_{mnop}$ would be easy to find. That would be

$$\frac{\partial L_{ik}}{\partial c_{mnop}} = -\frac{\partial}{\partial x_j} \left( \delta_{im} \delta_{jn} \delta_{ko} \delta_{lp} \frac{\partial}{\partial x_l} \right) = -\frac{\partial}{\partial x_n} \left( \delta_{im} \delta_{ko} \frac{\partial}{\partial x_p} \right)$$

which is relatively simple as compared, for example, with the corresponding orthorombic equations. However we know that these parameters are not independent due to symmetries on the elastic tensor.

For the reader who is not familiar with Kronecker delta manipulations under Einstein’s contraction notation, here is a tip that could be useful for syntax debugging. Dummy indices should appear exactly twice. If they appear either one, three or more times, check your derivation. Dummy indices should disappear after contraction with Kronecker deltas. In all analysis from here down, the free variables are only $k$ (for $u_k$) and $i$ (for the source $f_i$). Free variables only appear once. If two or three check your algebra. This is like dimensional analysis, here for indices, instead of units.

There are up to only 21 independent stiffness constants which can be independent, so a maximum of 22 gradient components.\(^4\)

In general for a given parameter $p$ we should find:

$$\frac{\partial L_{ik}}{\partial p} = -\frac{\partial}{\partial x_j} \left( \frac{\partial c_{ijkl}}{\partial p} \frac{\partial}{\partial x_l} \right)$$

For example, for isotropic media (Aki and Richards, [?])

$$c_{ijkl} = \lambda \delta_{ij} \delta_{kl} + \mu (\delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk}),$$

(8.4.31)

\(^{4}\)When we make the model discrete, these 22 translate into 22 $nx*ny*nz$ where $nx$, $ny$, $nz$ are the dimensions of the velocity model.
hence
\[
\frac{\partial L_{ik}}{\partial \lambda} = -\frac{\partial}{\partial x_j} \delta_{ij} \delta_{kl} \frac{\partial L_{kl}}{\partial \lambda} = -\frac{\partial}{\partial x_i} \frac{\partial}{\partial x_k}.
\]
\[
\frac{\partial L_{ik}}{\partial \mu} = -\frac{\partial}{\partial x_j} \left( \delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk} \right) \frac{\partial}{\partial x_l} = -\delta_{ik} \frac{\partial}{\partial x_l} \frac{\partial}{\partial x_i} - \frac{\partial}{\partial x_i} \frac{\partial}{\partial x_k}.
\]

(iii) For any medium
\[
C_{\rho} = \left\langle \frac{\partial L_{ik}}{\partial \rho} u_k, \phi_i \right\rangle = \left\langle \frac{\partial^2 u_i}{\partial^2 t}, \phi_i \right\rangle
\]

For the isotropic case we found
\[
C_{\lambda} = -\left\langle \frac{\partial}{\partial x_i} \frac{\partial}{\partial x_k} u_k, \phi_i \right\rangle
\]
\[
C_{\mu} = -\left\langle \delta_{ik} \frac{\partial}{\partial x_l} \frac{\partial}{\partial x_l} u_k, \phi_i \right\rangle - \left\langle \frac{\partial}{\partial x_i} \frac{\partial}{\partial x_k} u_k, \phi_i \right\rangle.
\]

8.4.4 Transversely Isotropic Media with a Vertical axis of Symmetry: VTI

It is common to define the stiffness tensor in terms of the Voigt coefficients. Here
\[
C_{I,J} = c_{ijkl}
\]
with the mapping
\[
ij \mapsto I , \quad kl \mapsto J :
\]
\[
i1 \mapsto 1 , \quad 22 \mapsto 2 , \quad 33 \mapsto 3
\]
\[
23, 32 \mapsto 4 , \quad 13 \text{ and } 31 \mapsto 5 , \quad \text{and } 12 \text{ and } 21 \mapsto 6.
\]
The Voigt notation has the advantage of reducing the number of coefficients from 81 to 36 (from which at most 21 are different). The new matrix \( C_{I,J} \) is symmetric.

The relation between the stiffness coefficients \( c_{ijkl} \) and the Voigt coefficients \( C_{I,J} \) for VTI is given by
\[
c_{ijkl} = \lambda_{VTI} \delta_{ij} \delta_{kl} + \mu_{VTI} \left( \delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk} \right) + (C_{11} + C_{33} - 2C_{13} - 4C_{44}) \delta_{ij} \delta_{kl} \delta_{kl} \delta_{k3} \delta_{l3} + (C_{13} - C_{11} + 2C_{66})(\delta_{ij} \delta_{kl} + \delta_{ij} \delta_{kl} \delta_{k3} \delta_{l3}) + (C_{44} - C_{66})(\delta_{ij} \delta_{ji} \delta_{kl} + \delta_{ij} \delta_{ji} \delta_{k3} \delta_{l3} + \delta_{ij} \delta_{ji} \delta_{k3} \delta_{l3} + \delta_{ij} \delta_{ji} \delta_{k3} \delta_{l3}),
\]

(8.4.32)

with
\[
\lambda_{VTI} = C_{11} - 2C_{66}
\]
\[
\mu_{VTI} = C_{66}.
\]
Ikelle and Amundsen, [?]. This equation assumes that $x_3$ points along the symmetry axis.

While equation 8.4.32 is long, it is very convenient. It shows a linear relationship between the Voigt coefficients $C_{1j}$ and the stiffness coefficients $c_{ijkl}$. This will make the algebra simple when finding $\partial L_{ik}/\partial p$. In fact the gradient for the 5 VTI parameters: $C_{11}$, $C_{13}$, $C_{33}$, $C_{44}$ and $C_{66}$ is found from the following simple derivations:

\[
\frac{\partial L_{ik}}{\partial C_{11}} = -\frac{\partial}{\partial x_j}(\delta_{ij} \delta_{kl} + \delta_{i3} \delta_{j3} \delta_{k3} \delta_{l3} - \delta_{i3} \delta_{j3} \delta_{kl} - \delta_{ij} \delta_{k3} \delta_{l3}) \frac{\partial}{\partial x_1}
\]

\[
= -\frac{\partial}{\partial x_i} \frac{\partial}{\partial x_k} - \delta_{i3} \delta_{k3} \frac{\partial^2}{\partial^2 x_3} + \delta_{i3} \frac{\partial}{\partial x_3} \frac{\partial}{\partial x_k} + \delta_{k3} \frac{\partial}{\partial x_3} \frac{\partial}{\partial x_i}
\]

\[
\frac{\partial L_{ik}}{\partial C_{13}} = -\frac{\partial}{\partial x_j}(-2\delta_{i3} \delta_{j3} \delta_{k3} \delta_{l3} + \delta_{i3} \delta_{j3} \delta_{kl} + \delta_{ij} \delta_{k3} \delta_{l3}) \frac{\partial}{\partial x_1}
\]

\[
= 2\delta_{i3} \delta_{k3} \frac{\partial^2}{\partial^2 x_3} - \delta_{i3} \frac{\partial}{\partial x_3} \frac{\partial}{\partial x_k} - \delta_{k3} \frac{\partial}{\partial x_3} \frac{\partial}{\partial x_i}
\]

\[
\frac{\partial L_{ik}}{\partial C_{33}} = -\frac{\partial}{\partial x_j}(\delta_{i3} \delta_{j3} \delta_{k3} \delta_{l3}) \frac{\partial}{\partial x_1} = -\frac{\partial^2}{\partial^2 x_3}
\]

\[
\frac{\partial L_{ik}}{\partial C_{44}} = -\frac{\partial}{\partial x_j}((\delta_{i3} \delta_{j3} \delta_{k3} + \delta_{i3} \delta_{j3} \delta_{kl} + \delta_{ik} \delta_{j3} \delta_{l3} + \delta_{i3} \delta_{k3} \delta_{j3})) \frac{\partial}{\partial x_1}
\]

\[
= -\delta_{k3} \frac{\partial}{\partial x_3} \frac{\partial}{\partial x_i} - \delta_{i3} \frac{\partial}{\partial x_k} \frac{\partial}{\partial x_3} - \delta_{ik} \frac{\partial^2}{\partial^2 x_3} - \delta_{i3} \delta_{k3} \frac{\partial}{\partial x_3} \frac{\partial}{\partial x_i}
\]

\[
\frac{\partial L_{ik}}{\partial C_{66}} = -\frac{\partial}{\partial x_j}((\delta_{i3} \delta_{j3} - \delta_{i3} \delta_{k3} \delta_{l3} - \delta_{i3} \delta_{l3} \delta_{jk} - \delta_{ik} \delta_{j3} \delta_{l3} - \delta_{i3} \delta_{k3} \delta_{jl})) \frac{\partial}{\partial x_1}
\]

\[
= \delta_{k3} \frac{\partial}{\partial x_3} \frac{\partial}{\partial x_i} + \delta_{i3} \frac{\partial}{\partial x_k} \frac{\partial}{\partial x_3} + \delta_{ik} \frac{\partial^2}{\partial^2 x_3} + \delta_{i3} \delta_{k3} \frac{\partial}{\partial x_3} \frac{\partial}{\partial x_i} - \delta_{ik} \frac{\partial}{\partial x_3} \frac{\partial}{\partial x_i}
\]

\[
= \delta_{k3} \frac{\partial}{\partial x_3} \frac{\partial}{\partial x_i} + \delta_{i3} \frac{\partial}{\partial x_k} \frac{\partial}{\partial x_3} + \delta_{ik} \frac{\partial^2}{\partial^2 x_3} + (\delta_{i3} \delta_{k3} - \delta_{ik}) \frac{\partial}{\partial x_3} \frac{\partial}{\partial x_i}.
\]

These relationships are relatively simple since all derivatives are taken on linear relations. Modeling in terms of the $c_{ijkl}$ or the $C_{ij}$ coefficients is simpler than using other coefficients (such as Thomsen’s parameters for example). In addition, the gradient computation turn out to be simpler this way, since the relationship between the $c_{ijkl}$ and the $C_{ij}$ is linear while the relationship between any of these and Thomsen’s parameters (for example) is not. Here are the relations between the $C_{ij}$’s and
the Thomsen’s parameters.

\[ V_{PO} = \sqrt{\frac{C_{33}}{\rho}} \]
\[ V_{SO} = \sqrt{\frac{C_{44}}{\rho}} \]
\[ \epsilon = \frac{C_{11} - C_{33}}{2C_{33}} \]
\[ \delta = \frac{(C_{13} + C_{44})^2 - (C_{33} - C_{44})^2}{2C_{33}(C_{33} - C_{44})} \]
\[ \gamma = \frac{C_{66} - C_{44}}{2C_{44}}. \]

The success on Thomsen’s notation comes from the geophysical interpretation of his parameters, however for numerical purposes the relation between the stiffness and the Voigt coefficients is more convenient. Thomsen’s equations are easy to invert for \( C_{ij} \) in terms of \( V_{PO}, V_{SO}, \epsilon, \delta \) and \( \gamma \). If those are the data that are given to FWI, then they can be converted simply to \( C_{ij} \) form (using for example Seismic Unix “suop” and “suop2”).

The final example is the transversely isotropic media with a horizontal axis of symmetry.

### 8.4.5 Transversely Isotropic Media with a Horizontal Axis of Symmetry: HTI

The relation between the stiffness coefficients \( c_{ijkl} \) and the Voigt coefficients \( C_{ij} \) for HTI with the symmetry axis pointing in the \( x_1 \) direction is given by

\[
c_{ijkl} = \lambda_{HTI} \delta_{ij} \delta_{kl} + \mu_{HTI} (\delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk})
+ (C_{11} + C_{33} - 2C_{13} - 4C_{66}) \delta_{i1} \delta_{j1} \delta_{k1} \delta_{l1}
+ (C_{13} - C_{33} + 2C_{44}) (\delta_{i1} \delta_{j1} \delta_{kl} + \delta_{ik} \delta_{jl} \delta_{i1} \delta_{l1})
+ (C_{66} - C_{44}) (\delta_{il} \delta_{j1} \delta_{k3} + \delta_{il} \delta_{t1} \delta_{jk} + \delta_{ik} \delta_{j1} \delta_{l1} + \delta_{i1} \delta_{k1} \delta_{j1}),
\]

with

\[ \lambda_{HTI} = C_{33} - 2C_{44} \]
\[ \mu_{HTI} = C_{44}, \]

which look almost like the VTI but switching the index 3 by the index 1 (Ikelle and Amundsen [?]). As with the previous example the gradient computation is obtained after finding the partial Gâteaux derivatives:
8.5 Numerical Test

\[
\begin{align*}
\frac{\partial L_{ik}}{\partial C_{11}} &= -\frac{\partial}{\partial x_j}(\delta_{i1}\delta_{j1}\delta_{k1}\delta_{l1})\frac{\partial}{\partial x_l} = -\delta_{i1}\delta_{k1}\frac{\partial^2}{\partial x_1^2} \\
\frac{\partial L_{ik}}{\partial C_{13}} &= -\frac{\partial}{\partial x_j}(\delta_{i1}\delta_{j1}\delta_{k1}\delta_{l1}) + \delta_{i1}\delta_{j1}\delta_{k1} + \delta_{ij}\delta_{k1}\delta_{l1})\frac{\partial}{\partial x_l} \\
&= 2\delta_{i1}\delta_{k1}\frac{\partial^2}{\partial x_1^2} - \delta_{i1}\frac{\partial}{\partial x_1}\frac{\partial}{\partial x_k} - \delta_{k1}\frac{\partial}{\partial x_i}\frac{\partial}{\partial x_k} \\
\frac{\partial L_{ik}}{\partial C_{33}} &= -\frac{\partial}{\partial x_j}(\delta_{i1}\delta_{j1}\delta_{k1}\delta_{l1} - \delta_{i1}\delta_{j1}\delta_{k1} - \delta_{ij}\delta_{k1}\delta_{l1} + \delta_{ik}\delta_{jl})\frac{\partial}{\partial x_l} \\
&= -\delta_{i1}\delta_{k1}\frac{\partial^2}{\partial x_1^2} + \delta_{i1}\frac{\partial}{\partial x_1}\frac{\partial}{\partial x_k} + \delta_{k1}\frac{\partial}{\partial x_i}\frac{\partial}{\partial x_k} + \frac{\partial}{\partial x_i}\frac{\partial}{\partial x_k} \\
\frac{\partial L_{ik}}{\partial C_{44}} &= -\frac{\partial}{\partial x_j}(2(\delta_{i1}\delta_{j1}\delta_{k1} + \delta_{ij}\delta_{k1}\delta_{l1})) - (\delta_{il}\delta_{jl}\delta_{k3} + \\
&\quad \delta_{il}\delta_{l1}\delta_{jk} + \delta_{ik}\delta_{j1}\delta_{l1} + \delta_{i1}\delta_{l1}\delta_{jl} + \delta_{ik}\delta_{jl} + \delta_{il}\delta_{jl})\frac{\partial}{\partial x_l} \\
&= \delta_{k3}\frac{\partial}{\partial x_i}\frac{\partial}{\partial x_k} + \delta_{il}\frac{\partial}{\partial x_i}\frac{\partial}{\partial x_k} + \delta_{ik}\frac{\partial^2}{\partial x_1^2} + \delta_{i1}\frac{\partial^2}{\partial x_1^2} \\
&\quad \delta_{ik}\frac{\partial}{\partial x_1}\frac{\partial}{\partial x_k} - 2\delta_{il}\frac{\partial}{\partial x_1}\frac{\partial}{\partial x_k} - 2\delta_{i1}\frac{\partial}{\partial x_1}\frac{\partial}{\partial x_k} \\
\frac{\partial L_{ik}}{\partial C_{66}} &= -\frac{\partial}{\partial x_j}(\delta_{il}\delta_{i1}\delta_{k3} + \delta_{i1}\delta_{k1}\delta_{j1}\delta_{l1} + \\
&\quad \delta_{i1}\delta_{k1}\delta_{jl} - 4\delta_{i1}\delta_{jl}\delta_{k1}\delta_{l1})\frac{\partial}{\partial x_l} \\
&= 4\delta_{i1}\delta_{k1}\frac{\partial^2}{\partial x_1^2} - \delta_{k3}\frac{\partial}{\partial x_1}\frac{\partial}{\partial x_k} - \delta_{i1}\frac{\partial}{\partial x_k}\frac{\partial}{\partial x_1} - \delta_{ik}\frac{\partial^2}{\partial x_1^2} - \delta_{i1}\delta_{k1}\frac{\partial}{\partial x_1}\frac{\partial}{\partial x_k}.
\end{align*}
\]

Since the wave operators are self-adjoint then the steps to compute \(\phi\) and \(u\) are done with the same anisotropic modeling equation. The only algebraic computations that we have to do are those for \(\partial L_{ik}/\partial p\) as shown here.

The same type of computations could be carried out for orthorombic media, but the algebra is more complex.

Next, we show a numerical example on the acoustic wave equation to test the validity of the results proved on this document.

8.5 Numerical Test

For the numerical tests, we evaluate the gradient of the objective function corresponding to the acoustic isotropic, constant density wave equation, a generalization of the homogeneous case discussed in section 8.4.1.

The information that we need is an initial guess \(p_0\) which in this case is the initial velocity model shown in Figure 8.1.
We modeled 79 shot gathers, each with 240 receivers. On each receiver we computed a synthetic trace with 1000 samples, each sample with a sampling rate of 1 millisecond. Figure 8.3 shows a shot record simulated on the right side of the model. The distance between shots is 100 m, the distance between receivers is 50 meters in a split–spread acquisition pattern. The source used for the modeling is a Ricker wavelet with 10 Hz central frequency and a maximum frequency of 20 Hz. The grid size for the modeling is of $10 \times 10 \text{m}^2$.

We computed directional derivatives using two different methods and compare the results. One of the methods is the ASM, which is the topic of this document. Figure 8.4 shows the result of computing the gradient using the ASM.
8.5 Numerical Test

The other method is to use a Gâteaux central difference equation

$$\frac{C(v + s \delta v_i) - C(v - s \delta v_i)}{2s}$$

(8.5.33)

where $v$ is the initial guess shown in Figure 8.1 and $\delta v_i$ is the following vector (matrix, or better, the tensor direction)

$$\delta v_i = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

(8.5.34)

The number of columns is $n_x=120$ (the number of traces of the velocity model, sampled at 100 m) and the number of rows is $n_z=400$ (number of depth samples with
sampling rate of 10 m each). The only raw with a 1 in each column is the $i$–th row. All other rows are zero. We need $n_z$ of these vectors to expand our space of interest.

Ideally we would like to use a perturbation matrix $\delta v_{jk}$ with entries $\delta_{jk}$, that is, all zeroes except for a single 1 in the $jk$ entry. However this would imply that to compute all central derivatives we would need to run $4*n_x*n_z$, that is, 192000 modeling problems (the cost of 96000 RTMs).

Precisely, we want to test the validity of the following formula

$$DC(v)(\delta v_i) \approx \frac{C(v + s \delta v_i) - C(v - s \delta v_i)}{2s}$$

where the left symbol $DC(v)(\delta v_i)$ represents the Fréchet derivative of the cost function $C(v)$ on the initial guess velocity model $v$ in Figure 8.1 and evaluated (inner product with) at the increment vector $\delta v_i$ in equation 8.5.34, while the right expression is the central difference Gâteaux derivative. We choose $s = 1, 10, 20, 30, 40$ in this experiment. In order to compare the two differential approaches we need to take the dot product of the Fréchet derivative shown in Figure 8.4 with the vector $\delta v_i$ for each depth. This will provides us with 401 points (to cover depths from 0 to 4000 m). At each point the value is the differential computed by the two methods. Figure 8.5 shows the comparison of the differential found using the Fréchet derivative (the dark brown circles) with the Gâteaux derivative for $s = 1$ (red crosses), $s = 10$ (green “x”s), $s = 20$ (dark blue stars *) $s = 30$ (purple squares) and $s = 40$ (light blue solid squares).

Figure 8.5: Differential along the vector $\delta v$, evaluated using the central differences 8.5.33 (Gâteaux derivative) for $s = 1, 10, 20, 30, 40$ and from the Fréchet derivative.

More precisely, here is a pseudo–code of the procedure to compute the data in Figure 8.5.
Compute the Frechet derivative (Gradient) in Figure 4

for each z (index i=0,400)
{
    /* inner product of tensors Frechet with dvi=(0,0,....1,1,....0,0) */
    left[i] = DC(V) . dv[i] /* sum DC[V[i]) along row "i"

    for each s (1,10,20,30,40)
    {
        /* compute the Gateaux Derivative */
        right[i][s] = ( C(V + s dv[i]) - C(V)/(2 s)
    }
}

The plot corresponds to the 401 points left[i] of the Fréchet derivative and the same number of points right[i][s] for each s = 1, 10, 20, 30 and 40 on the Gâteaux derivatives.

While the Fréchet derivative in this example is a tensor of rank 2, we are more familiar with gradients which are tensors of rank 1. That is, we know that the partial derivative could be computed from the formula

\[
\frac{\partial C}{\partial v_i} = \lim_{s \to 0} \frac{C(V + se_i) - C(V - se_i)}{2s},
\]

where \(e_i = (0, \ldots, 0, 1, 0 \cdots 0)\). Where the “1” is in the \(i\)-th position. If we have the gradient (Fréchet), then another way of computing the partial derivative is by taking the dot product with the gradient. That is,

\[
\frac{\partial C}{\partial v_i} = \nabla C(v) \cdot e_i.
\]

However the gradient allows us to compute more than that. We can find the derivative along any arbitrary direction \(u\) by just having the dot product of the gradient with this \(u\). That is, the directional derivative along \(u\) can be written as

\[
\frac{\partial C}{\partial u} = \lim_{s \to 0} \frac{C(V + su) - C(V - su)}{2s},
\]

where \(u\) is a direction vector in the parameter space, or

\[
\frac{\partial C}{\partial u} = \nabla C(v) \cdot u.
\]

For parameters defined in a 3D space, the Fréchet derivative is a rank 3 tensor and the directional (Gâteaux) derivative is a tensor contraction.

The power of the Fréchet derivative (gradient) is that it needs to be computed only once, and then we can find derivatives along any vectorial or tensor direction. However to compute the Gâteaux derivative without the use of the gradient, the problem has to be solved every time. In this way, finding the gradient is like finding a Green’s function in a differential equation, where we just need to convolve with a source term.
to get a new solution without having to solve the differential equation again. It is like having $A^{-1}$ when solving for $Ax = b$. We only have to pre-multiply the source $b$ with $A^{-1}$. The gradient is the “genetic code” of any descent/ascent method. The power of the ASM is that it provides a way to find the actual Fréchet derivative which fits all directional derivatives with only two modeling–like computations (one “forward” and one “backwards”). It is a one–fits–all solution.

We have several observations on the numerical results of this experiment:

- We had to scale down the Fréchet derivative computation by $10^{-6}$. The sampling rate is $10^{-3}$ seconds and a second derivative with respect to time would have $10^{-6}$ unit scaling. However, ignoring this scaling would make the data only $10^6$ smaller, no larger. So, we are not sure if there is an error in the source code or there is something in the mathematical physics that we are not accounting for. Still, we are more interested on the direction of the gradient and not its magnitude, since the optimization problem will scale the gradient based on its own criteria.

- The gradient computation using the ASM (the Fréchet derivative) matches well the computations done using central differences Gâteaux derivatives, except at the top of the model.

- The dispersive error observed closed to the top of the model is worse as $s$ becomes larger. For $s = 1$ the error is very small and the matching between the gradient using the ASM and that using the central differences Gâteaux derivatives is excellent.

- Since the gradient using the ASM was computed for a grid spacing of 10 m, then it is understandable that the level of precision is higher than that of the Gâteaux derivatives for $s > 10$.

- We believe that the error is larger in the top of the model than at the bottom because close to the source the wave equation does not work correctly.

- It is really amazing that the computation of the differentials through the Fréchet derivative (the ASM) is 96000 faster than the computation of the differentials using the Gâteaux central difference if we really want to perturb each cell individually. The gain is much more, the larger the model. In fact the cost of computing the Gâteaux derivatives for three–dimensional models is prohibitive. More precisely, in 3D the number of RTM–like processes run, if the Gâteaux derivative will be evaluated (for each grid point) would be $nx*ny*nz$, Where $nx$, $ny$ and $nz$ are the number of grids on the velocity model along the $x$, $y$ and $z$ directions respectively. Still the number of RTM–like processes needed to find the Fréchet derivative is 1.
• The experiment proposed here would be of great value in evaluating the fidelity of numerical FWI software, such as computing gradients in HTI models, VTI models or any other anisotropic models.

### 8.6 Conclusions

In this document we derived the ASM from simple examples, and then from first principles in general linear partial differential equations, matrices of functions and matrices of operators. The derivation shown here has the advantage the it does not depend on solving complicated differential equations and it is general. It applies to any linear partial differential equation or a matrix of functions or operators. The derivation offers a template that can be used in gradient–based inversion problems. This template is posted in terms of three simple steps. We applied the template to show how to find the gradient using the ASM in the case of two simple acoustic wave equations and two elastic wave equations (VTI and HTI). Because the method is general it can be applied to any operator with explicit dependence on the model parameters and that is solvable for the wavefield $u$. We also show how the Lagrange multipliers method, which offers a more general framework that the traditional framework of optimizing a least square function, can be reduced to a adjoint state problem. Finally, we showed a numerical example that validates the theory.

In addition, we proposed an experiment that could be of great value for testing development software on FWI. For example a new wave–inversion code for HTI models. The computation of the gradient is the most expensive and complicated part of a FWI algorithm and guaranteeing a good gradient computation is having a big part of code in place.
Appendices
Appendix A

Convexity and inequalities used on this document

The optimization methods explained in this book are mainly for convex or local convex functions. We define convexity and list a few inequalities which are important for the derivations on this document.

A.1 Convex Functions

Let $I \subset \mathbb{R}$ be an interval. A function $f : I \rightarrow \mathbb{R}$ is called convex if

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y). \tag{A.1.1}$$

for all $x, y \in I$ and all $\lambda \in [0, 1]$.

It is interesting to observe that if the function $f$ is linear then

$$f(\lambda x + (1 - \lambda)y) = \lambda f(x) + (1 - \lambda)f(y).$$

which is always convex. But a concave (or convex down) function satisfies the inequality $f(\lambda x + (1 - \lambda)y) \geq \lambda f(x) + (1 - \lambda)f(y)$. So a linear function is simultaneously convex and concave. A function $f$ is convex if and only its negative $-f$ is concave.

Figure A.1 shows an illustration of a convex function.

The interesting functions in this area are functions which satisfy strictly inequality. That is, they have curvature. and we prove first that for a convex function, the second derivative is always greater or equal to 0 (equal if it is linear), or in symbols $f''(z) \geq 0, \forall z \in I$. The proof of this is easy. Pick

$$x = z + h \quad y = z - h \quad \lambda = 1/2$$

in equation A.1.1, and find

$$f((z + h)/2 + (z - h)/2) \leq f(z + h)/2 + f(z - h)/2$$
that is

\[ f(z + h) - 2f(z) + f(z - h) \geq 0 \]

and dividing by \( 2h \)

\[ \frac{f(z + h) - 2f(z) + f(z - h)}{2h} \geq 0 \]

we take the limit as \( h \to 0 \) and find

\[ f''(z) \geq 0. \]

Since the second derivative of \( -\log x \) is \( 1/x^2 \) then \( -\log x \) is convex in the interval \((0, \infty)\). We now prove the reverse. That is if \( f''(x) \geq 0 \) for all \( x \in I \) where \( I \) is an interval over the real numbers, then \( f \) is convex in \( I \). Let us first prove that if \( f'(x) \) is monotonically increasing. From the mean value theorem, pick \( x < y \) in \( I \) we know that there is a \( c \in [x, y] \) such that

\[ f''(c) = \frac{f'(y) - f'(x)}{y - x} \]

and since \( f''(c) \geq 0 \), and \( y > x \) then

\[ f'(y) \geq f'(x). \]

So \( f' \) is monotonically increasing. Now, let us choose \( s = \lambda x + (1-\lambda)y \). Then \( x < s < y \). By the mean value theorem exists \( c_1, x < c_1 < s \) and, \( c_2, s < c_2 < y \) such that

\[ f'(c_1) = \frac{f(s) - f(x)}{s - x} \quad f'(c_2) = \frac{f(y) - f(s)}{y - s}, \]
since $f'$ is monotonically increasing and $c_1 \leq c_2$ then
\[
\frac{f(s) - f(x)}{s - x} \leq \frac{f(y) - f(s)}{y - s}
\]
\[
\frac{f[\lambda x + (1 - \lambda)y] - f(x)}{\lambda x + (1 - \lambda)y - x} \leq \frac{f(y) - f(\lambda x + (1 - \lambda)y)}{y - (\lambda x + (1 - \lambda)y)}
\]
\[
\lambda f[\lambda x + (1 - \lambda)y] - \lambda f(x) \leq (1 - \lambda) f(y) - f[\lambda x + (1 - \lambda)y] + \frac{\lambda f[\lambda x + (1 - \lambda)y]}{\lambda(y - x)}
\]
\[
f[\lambda x + (1 - \lambda)y] \leq \lambda f(x) + (1 - \lambda) f(y).
\]

So $f$ is convex.

Next, we show a variation of Jensen’s inequality \(^1\), for a discrete set.

### A.2 Jensen’s Inequality

Suppose $I \in \mathbb{R}$ is an interval and $f : I \to \mathbb{R}$ is convex. Then for all $n \in \mathbb{N}$, all $\lambda_i \geq 0$, $i = 1, \cdots, n$ and $\sum_{j=1}^{n} \lambda_j = 1$, and all $z_i \in I$,
\[
f \left( \sum_{j=1}^{n} \lambda_j z_j \right) \leq \sum_{j=1}^{n} \lambda_j f(z_j)
\]

That is, the function is “under-linear” for more than just two points. It is like the definition of convexity extended to many points.

We use induction. For $n = 1$ it is trivial that $f(z_1) \leq f(z_1)$. Let us assume that up to some $n \geq 2$ the inequality is valid. Then we want to evaluate
\[
f \left( \sum_{j=1}^{n+1} \lambda_j z_j \right).
\]

The idea is to split the sum in two sets of points each with less than $n + 1$ elements. By assumption
\[
\sum_{i=1}^{n+1} \lambda_i = 1
\]

If any of the $\lambda_i$ is equal to 1, all others have to be equal to 0 because $\lambda_i \geq 0$, $i = 1, \cdots, n + 1$. So, necessarily $\lambda_i < 1$, for all $i$’s. Let us separate $\lambda_1$ from the rest of $\lambda_i$’s and write
\[
\sum_{j=1}^{n+1} \lambda_j z_j = \lambda_1 z_1 + \sum_{j=2}^{n+1} \lambda_j z_j
\]

\(^1\)http://en.wikipedia.org/wiki/Jensen%27s_inequality
Since \( z_1 \) has already a coefficient \( \lambda_1 \), we want to write the second sum as some number \( u \) with coefficient \( 1 - \lambda_1 \). That is, we call
\[
(1 - \lambda_1)u = \sum_{j=2}^{n+1} \lambda_j z_j,
\]
or define
\[
u = \frac{\sum_{j=2}^{n+1} \lambda_j z_j}{1 - \lambda_1} = \sum_{j=2}^{n+1} \frac{\lambda_j z_j}{1 - \lambda_1}.
\]

We obtained what we wanted. That is,
\[
\sum_{j=1}^{n+1} \lambda_j z_j = \lambda_1 z_1 + (1 - \lambda)u
\]
and we can apply the convexity of \( f \) to these two terms to obtain
\[
f \left( \sum_{j=1}^{n+1} \lambda_j z_j \right) = f(\lambda_1 z_1 + (1 - \lambda_1)u) \leq \lambda_1 f(z_1) + (1 - \lambda_1) f(u).
\]

Now we should expand \( f(u) \). We note that the coefficients of \( u \) add to 1, because
\[
\sum_{i=1}^{n+1} \lambda_i = 1 \quad \Rightarrow \quad \lambda_1 = 1 - \sum_{i=2}^{n+1} \lambda_i \quad \Rightarrow \quad \sum_{i=2}^{n+1} \lambda_i = 1 - \lambda_1
\]
and so
\[
\sum_{i=2}^{n+1} \frac{\lambda_i}{1 - \lambda_1} = \frac{1 - \lambda_1}{1 - \lambda_1} = 1,
\]
and then by the induction hypothesis
\[
f(u) \leq \sum_{i=2}^{n+1} \frac{\lambda_i}{1 - \lambda_1} f(z_i),
\]
and finally
\[
f \left( \sum_{j=1}^{n+1} \lambda_j z_j \right) = f(\lambda_1 z_1 + (1 - \lambda_1)u) \leq \lambda_1 f(z_1) + \frac{1}{1 - \lambda_1} \sum_{j=2}^{n+1} \lambda_i f(z_i),
\]
that is
\[
f \left( \sum_{j=1}^{n+1} \lambda_j z_j \right) \leq \sum_{j=1}^{n+1} \lambda_i f(z_i),
\]
which proves the theorem.

The Jensen’s inequality is useful to prove other inequalities. This is the case of the following inequality.
A.3 Weighted geometric/arithmetic mean inequality

Suppose $\sum_{j=1}^{n} \lambda_j a_j$ is a convex combination of non-negative numbers $a_1, \ldots, a_n$. Then

$$a_1^{\lambda_1} a_2^{\lambda_2} \cdots a_n^{\lambda_n} \leq \sum_{j=1}^{n} \lambda_j a_j.$$

What this is telling us is that the weighted geometric mean is smaller than the weighted arithmetic mean. We assume that $0^0 = 0$ by definition, so if one of the $a_i = 0$, then the inequality is obvious and so we assume that all $a_i > 0$. We use Jensen’s inequality applied to the convex function $f(x) = -\log x$, on the interval $I = (0, \infty)$. That is

$$-\log \left( \sum_{j=1}^{n} \lambda_j a_j \right) \leq - \sum_{j=1}^{n} \lambda_j \log a_j = - \log (a_1^{\lambda_1} \cdots a_n^{\lambda_n})$$

and so, multiplying by $-1$ and taking the exponential function we observe the inequality. In particular when each $\lambda_i = 1/n$, we find

$$(a_1 a_2 \cdots a_n)^{1/n} \leq \frac{1}{n} \sum_{j=1}^{n} a_j \quad \text{(A.3.2)}$$

which is the classical geometric/arithmetic mean inequality.

A.4 Hölder’s inequality

For $x, y \in \mathbb{C}^n$ and $1 \leq p \leq \infty$

$$\sum_{j=1}^{n} |x_j y_j| \leq \|x\|_p \|y\|_q, \quad \text{where} \quad \frac{1}{p} + \frac{1}{q} = 1.$$

For any $1 \leq p \leq \infty$ the norm $\|\cdot\|_p$ is defined as

$$\|x\|_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

and

$$\|x\|_\infty = \max_{i=1}^{n} |x_i|.$$

When $p = 2$, by default nothing is written as sub-index. That is $\|\cdot\|_2 = \|\cdot\|.$
Let us first prove this inequalities for the extrema values \( p = 1 \) and \( p = \infty \). If \( p = 1 \), then \( q = \infty \) we have

\[
\sum_{j=1}^{n} |x_j y_j| \leq \max_{i=1}^{n} |y_i| \sum_{j=1}^{n} |x_j| = \|x\|_1 \|y\|_\infty.
\]

The case \( p = \infty \) and \( q = 1 \) is the equivalent to this (just change \( x \) by \( y \) and being those vectors arbitrary we do not have to repeat that proof here.

Let us then assume that \( 1 < p < \infty \). If one of or both vectors \( x \) and \( y \) are zero the inequality turns into the equality \( 0 \leq 0 \). So we assume that both \( x \) and \( y \) are non-zero vectors.

For any \( a_j, b_j \geq 0 \) the weighted arithmetic/geometric mean inequality indicates that

\[
a_j^{\frac{1}{p}} b_j^{\frac{1}{q}} \leq \frac{a_j}{p} + \frac{b_j}{q}.
\]

We set

\[
a_j = \frac{|x_j|^p}{\|x\|^p} \quad b_j = \frac{|y_j|^q}{\|y\|^q}
\]

and so

\[
\left( \frac{|x_j|^p}{\|x\|^p} \right)^{\frac{1}{p}} \left( \frac{|y_j|^q}{\|y\|^q} \right)^{\frac{1}{q}} \leq \frac{1}{p} \frac{|x_j|^p}{\|x\|^p} + \frac{1}{q} \frac{|y_j|^q}{\|y\|^q}.
\]

We add over \( j = 1, \ldots, n \), and find

\[
\sum_{i=1}^{n} \left( \frac{|x_j|^p}{\|x\|^p} \right)^{\frac{1}{p}} \left( \frac{|y_j|^q}{\|y\|^q} \right)^{\frac{1}{q}} \leq \sum_{i=1}^{n} \frac{1}{p} \frac{|x_j|^p}{\|x\|^p} + \frac{1}{q} \frac{|y_j|^q}{\|y\|^q}.
\]

We write the left sum as

\[
\sum_{i=1}^{n} \left( \frac{|x_j|^p}{\|x\|^p} \right)^{\frac{1}{p}} \left( \frac{|y_j|^q}{\|y\|^q} \right)^{\frac{1}{q}} = \frac{1}{\|x\|_p \|y\|_q} \sum_{i=1}^{n} |x_j| |y_j|
\]

and the right side as

\[
\sum_{i=1}^{n} \frac{1}{p} \frac{|x_j|^p}{\|x\|^p} + \frac{1}{q} \frac{|y_j|^q}{\|y\|^q} = \frac{1}{p} \sum_{j=1}^{n} |x_j|^p + \frac{1}{q} \sum_{j=1}^{n} |y_j|^q + \frac{1}{p} + \frac{1}{q} = 1
\]

and so, we showed that

\[
\frac{1}{\|x\|_p \|y\|_q} \sum_{i=1}^{n} |x_j| |y_j| \leq 1,
\]

and from here

\[
\sum_{i=1}^{n} |x_j| |y_j| \leq \|x\|_p \|y\|_q
\]

which is the Hölder’s inequality. If \( p = q = 2 \) we find the famous
A.5 Minkowski’s inequality

Let $x, y \in \mathbb{C}^n$ and $1 \leq p \leq \infty$. Then

$$\|x + y\|_p \leq \|x\|_p + \|y\|_p.$$ 

Let us start with $p = 1$ here

$$\|x + y\|_1 = \sum_{i=1}^{n} |x_i + y_i| \leq \sum_{i=1}^{n} |x_i| + \sum_{i=1}^{n} y_i = \|x\|_1 + \|y\|_1.$$ 

and if $p = \infty$

$$\|x + y\|_\infty = \max_{i=1}^{n} |x_i + y_i| \leq \max_{i=1}^{n} |x_i| + \max_{i=1}^{n} |y_i| = \|x\|_\infty + \|y\|_\infty.$$ 

Let us now assume that $1 < p < \infty$. We write

$$\|x + y\|_p^p = \sum_{j=1}^{n} |x_j + y_j|^p$$

$$= \sum_{j=1}^{n} |x_j + y_j||x_j + y_j|^{p-1}$$

$$\leq \sum_{j=1}^{n} |x_j||x_j + y_j|^{p-1} + \sum_{j=1}^{n} |y_j||x_j + y_j|^{p-1}. \quad (A.5.3)$$

We apply the Hölder’s inequality to each of the terms above. That is

$$\sum_{j=1}^{n} |x_j||x_j + y_j|^{p-1} \leq \|x\|_p \left( \sum_{j=1}^{n} |x_j + y_j|^{(p-1)q} \right)^{1/q}$$

and from

$$\frac{1}{p} + \frac{1}{q} = 1 \quad \Rightarrow \quad (p - 1)q = p,$$

we see that

$$\sum_{j=1}^{n} |x_j||x_j + y_j|^{p-1} \leq \|x\|_p \|x + y\|_p^{p/q} = \|x\|_p \|x + y\|_p^{p-1}. \quad (A.5.4)$$
Similarly

\[ \sum_{j=1}^{n} |y_j|x_j + y_j|^{p-1} \leq \|y\| \|x + y\|^{p/q} \|y\| \|x + y\|^{p-1}. \tag{A.5.5} \]

Combining A.5.3, A.5.4, and A.5.5, we find

\[ \|x + y\|^p \leq (\|x\| + \|y\| \|x + y\|^{p-1} \]

which after dividing by \( \|x + y\|^{p-1} \) is

\[ \|x + y\| \leq \|x\| + \|y\|. \]

This is the triangular inequality for vectors in \( \mathbb{R}^n \). Together with the homogeneity and no-negativity properties of \( \|\cdot\|_p \), this proves that the \( p \)-norm is actually a norm in the space \( \mathbb{R} \).

### A.6 Kantarovich Inequality

The Kantarovich inequality is useful to prove bounds of iterative methods such as the Steepest Descent (SD) and the Conjugate Gradient (CG) methods. Given \( A \) positive definite and symmetric with largest eigenvalue \( M \) and smallest eigenvalue \( m \), then for any \( y \neq 0 \),

\[ 1 \leq \frac{(y^T Ay)(y^T A^{-1} y)}{(y^T y)^2} \leq \frac{(M + m)^2}{4Mm} \tag{A.6.6} \]

We observe that if \( A \) is the identity matrix we have the equality (to 1) on each of the tree terms.

Let \( (\lambda_j, v_j), j = 0, \ldots, n - 1 \), be the orthonormal eigenpairs of \( A \). It is known that \( (\lambda_j^{-1}, u_j) \) are the orthonormal eigenpairs of \( A^{-1} \). Let

\[ y = \sum_{j=0}^{n-1} c_j v_j \]

be the representation of \( y \) in the basis \( \{v_j\} \). We evaluate all expressions represented in the inequality. Since the vectors \( v_j \) are orthonormal \( v_j^T v_i = \delta_{ij} \), so

\[ y^T y = \sum_{j=0}^{n-1} c_j^2 \]

\[ y^T Ay = y^T A \sum_{i=0}^{n-1} c_i v_i = y^T \sum_{i=0}^{n-1} c_i \lambda_i v_i = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} c_j c_j^T c_i \lambda_i v_i = \sum_{j=0}^{n-1} c_j^2 \lambda_j \]

\[ y^T A^{-1} y = \sum_{j=0}^{n-1} c_j^2 \lambda_j^{-1}, \]
then by defining \( t_i = \frac{c_i^2}{\sum_{j=0}^{n-1} c_j^2}, i = 0, \ldots, n-1 \), we have that \( \sum_{i=0}^{n-1} t_i = 1 \). We define \( a \) and \( b \) as

\[
a = \frac{y^T Ay}{y^T y} = \sum_{i=0}^{n-1} t_i \lambda_i, \quad b = \frac{y^T A^{-1} y}{y^T y} = \sum_{i=0}^{n-1} \frac{t_i}{\lambda_i}
\]

and find lower and upper bounds for the product \( ab \). From the Cauchy-Schwartz inequality we find

\[
1 = \left( \sum_{i=0}^{n-1} t_i \right)^2 \leq \left( \sum_{i=0}^{n-1} (t_i \lambda_i)^{1/2} (t_i / \lambda_i)^{1/2} \right)^2 \leq \left( \sum_{i=0}^{n-1} t_i \lambda_i \right) \left( \sum_{i=0}^{n-1} t_i / \lambda_i \right) = ab
\]

so that we proved the lower bound of 1 for the product \( ab \). To prove the upper bound we use the geometric/arithmetic mean inequality A.3.2. That is,

\[
\sqrt{ab} = \frac{\sqrt{ac} + b/c}{2} \leq \frac{1}{2} \sum_{i=0}^{n-1} \left( \lambda_i c + \frac{1}{\lambda_i c} \right) = \frac{1}{2} \max_{mc \leq x \leq Mc} f(x).
\]

Now \( f''(x) \geq 0 \). In section A.1 we showed that \( f''(x) \geq 0 \) implies that the function \( f \) is convex. That is the function is below the segment line that joints the points \((mc, f(mc))\) and \((Mc, f(Mc))\), so the minimum of the function is below those two points and particularly

\[
\sqrt{ab} \leq \frac{1}{2} \max\{f(mc), f(Mc)\}. \tag{A.6.7}
\]

We are free to pick any positive number \( c \). Let us choose

\[
c = \frac{1}{\sqrt{mM}},
\]

so

\[
f(mc) = f(Mc) = \sqrt{\frac{M}{m}} + \sqrt{\frac{m}{M}} = \frac{M + m}{\sqrt{mM}}.
\]

wich by squaring A.6.7 is

\[
ab = \frac{(y^T Ay)(y^T A^{-1} y)}{(y^T y)^2} \leq \frac{(M + m)^2}{4Mm}.
\]

This concludes the upper bound proof and the proof of the inequality.
A.7 A rational versus exponential inequality

We prove the algebraic inequality

$$\frac{x - 1}{x + 1} < e^{-2/x}$$

Let us define $y = 1/x$, and from $2^k/k! = 2, k = 1, 2$, and $2^k/k! < 2$ for $k > 2$, we find by using the infinite series representation for the exponential function

$$e^{2/x} = e^{2y} = \sum_{k=0}^{\infty} \frac{(2y)^k}{k!} < 1 + 2y \sum_{k=0}^{\infty} y^k = 1 + \frac{2y}{1-y} = \frac{x+1}{x-1}. \quad (A.7.8)$$

A.8 Error bound for steepest descent

The SD algorithm, to find a solution of the linear system $Ax = b$, establishes that at each iteration $k$, the current point $x_k$ can be found in terms of the previous point $x_{k-1}$, the previous residual $r_{k-1}$ and a scalar $t$ using the formula $x_k = x_{k-1} - tr_{k-1}$. The residual $r_k$ and scalar $t$ are given by

$$r_k = Ax_k - b \quad , \quad t = \frac{\|r_k\|^2}{r_k^T A r_k}.$$

The bounds for the SD method are given by the following inequality

$$\|x^* - x_k\|_A \leq \left( \frac{\kappa - 1}{\kappa + 1} \right)^k \leq e^{-\frac{1}{2}k} \quad , \quad k \geq 0. \quad (A.8.9)$$

where $x^*$ is the exact solution and $x_k$ is the current point location.

The error vector, of exact solution relative to the current point location is given by

$$e_k = x^* - x_k$$

We show that

$$\frac{\|e_{k+1}\|^2_A}{\|e_k\|^2_A} \leq \left( \frac{\kappa - 1}{\kappa + 1} \right)^2 \quad , \quad k = 0, 1, \cdots, n - 1.$$

for then $\|e_k\|_A \leq \left( \frac{\kappa - 1}{\kappa + 1} \right) \|e_{k-1}\|_A \leq \cdots \leq \left( \frac{\kappa - 1}{\kappa + 1} \right)^k \|e_0\|_A$. Using the recursion formula $x_{k+1} = x_k - tr_k$, we see that

$$x^* - x_{k+1} = x^* - x_k + tr_k \quad \Rightarrow \quad e_{k+1} = e_k + tr_k.$$

Now, since $A = A^T$,

$$\|e_k\|^2_A = e_k^T A e_k = (x^* - x_k)^T A A^{-1} A (x^* - x_k) \quad \text{from} \quad I = A A^{-1}$$

$$= (b - r_k - B_k)^T A^{-1} (b - r_k - B) \quad \text{from} \quad A x^* = b$$

$$= r_k^T A^{-1} r_k.$$
In addition
\[ \| \epsilon_{k+1} \|_A^2 = (\epsilon_k + t r_k)^T A (\epsilon_k + t r_k) \]
\[ = \epsilon_k^T A \epsilon_k + 2 t r_k^T A \epsilon_k + t^2 r_k^T A r_k \]
\[ = \epsilon_k^T A \epsilon_k + 2 \frac{\| r_k \|^2}{r_k^T A r_k} r_k^T A r_k + \left( \frac{\| r_k \|^2}{r_k^T A r_k} \right)^2 r_k^T A r_k \] (A.8.10)

Now,
\[ A \epsilon_k = A x^* - A x_k = b - A x_k = -r_k, \]
so, we simplify the middle term, with coefficient 2, to
\[ -2 \frac{\| r_k \|^2 r_k^T A r_k}{r_k^T A r_k} = -\frac{2\| r_k \|^4}{r_k^T A r_k}. \]

We simplify equation A.8.10 as
\[ \| \epsilon_{k+1} \|_A = \| \epsilon_k \|_A - \frac{\| r_k \|^4}{r_k^T A r_k}. \]

Next, we use this, and the Kantorovich inequality A.6.6, and as follows:
\[ \frac{\| \epsilon_{k+1} \|_A^2}{\| \epsilon_k \|_A^2} = 1 - \frac{\| r_k \|^4}{(r_k^T A r_k)(r_k^T A^{-1} r_k)} \leq 1 - \frac{4mM}{(m + M)^2} = \left( \frac{M - m}{M + m} \right)^2. \]

Now since the condition number is defined as \( \kappa = m/M \), we see that
\[ \frac{\| \epsilon_{k+1} \|_A^2}{\| \epsilon_k \|_A^2} \leq \left( \frac{\kappa - 1}{\kappa + 1} \right)^2. \]

This proofs the first inequality on A.8.9. To prove the second inequality we use the equation A.7.8, with \( x = \kappa \). That is
\[ \frac{\kappa - 1}{\kappa + 1} < e^{-2/\kappa}, \quad \kappa > 1. \]

and so
\[ \frac{\kappa - 1}{\kappa + 1} \leq e^{-2k/\kappa}, \quad \kappa > 1. \]

where the equality happens at \( \kappa = 1 \) (the identity matrix). Recall that \( \kappa \geq 1 \) is the ratio of largest to smallest eigenvalue and so it is larger or equal than one.
Appendix B

Auxiliary Programs:

This appendix shows a few program implementations for several of the algorithms illustrated in this document.

B.0.1 Matlab code for the steepest descent method

The following code in Matlab computes a few steepest descent iterations toward the minimum using the algorithm 3. Here the tolerance of 0.01 and a maximum of 100 iterations are hardwired.

clear all;
n=100;
tol=0.01;

% function to minimize
f=inline('5*x^2+(1/2)*y^2-x+y','x','y');

% data
x0=0;
y0=0;
b=[1,-1]';
A = [10 0; 0 1];

% initial conditions
X(:,1) = [x0 y0]';
z(1)=f(x0,y0);
R(:,1) = -A*X(:,1)+b;

% steepest descent algorithm
i=2;
while norm( R(:,i-1)) > tol && i<n,
    D=R(:,i-1);
    t=dot(D,D)/dot(A*D,D);
    X(:,i)=X(:,i-1)+t*D;
    R(:,i) = -A*X(:,i)+b;


\[ z(i) = f(X(1,i), X(2,i)) \]
\[ i = i + 1; \]
end

figure;
hold
% draw the contours
Mx = max(X(1,:));
My = max(X(2,:));
mx = min(X(1,:));
my = min(X(2,:));
ngrids = 100;
dgx = (Mx - mx) / ngrids;
dgy = (My - my) / ngrids;
[U, V] = meshgrid(mx:dgx:Mx, my:dgy:My);
Z = 0.5*A(1,1)*U.^2 + 0.5*A(2,2)*V.^2 + 0.5*A(1,2)*U + 0.5*A(2,1)*V - b(1)*U - b(2)*V

[C, h] = contour(U, V, Z, z);
grid on;
axis equal;
axis tight;
xlabel('x');
ylabel('y');
title('Steepest descent on f(x,y)');

% plot zig-zag lines left right
line('Color', 'red');
Y = X';
W = reshape(Y(:,1), 2, []);
W(3, :) = nan;
W = W(:, :);
Q = reshape(Y(:, 2), 2, []);
Q(3, :) = nan;
Q = Q(:, :);
% subplot(2,2,1);
plot(W, Q, 'Color', [1, 0, 0]);
% plot zig-zag lines right left
Y = circshift(X', -1);
W = reshape(Y(:, 1), 2, []);
W(3, :) = nan;
W = W(:, :);
Q = reshape(Y(:, 2), 2, []);
Q(3, :) = nan;
Q = Q(:, :);
Q = Q(1:size(Q, 1)-2);
W = W(1:size(W, 1)-2);
% subplot(2,2,1);
plot(W,Q, 'Color',[1,0,0]);

% Save the plot as eps
box on
saveas(gcf, 'steepmatlab.eps','psc2');
figure;

% Plot z values
plot(z, 'LineWidth', 2);
box on
grid on;
xlabel('iteration number');
ylabel('z');
saveas(gcf,'zvsn.eps','psc2');

% Write 3D points for gnuplot
fp = fopen('gnuplot.dat','w');
for i = 1: size(z,2)
    fprintf(fp,'%f %f %f \n', X(1,i),X(2,i),z(i));
end
fclose(fp);
Appendix C

Generalized Tikhonov Regularization

The equations for Generalized Tikhonov Regularization are shown presented in Wikipedia without derivation. Here I derive those equations. There are two cases to be considered.

C.1 Norm $\ell_2$, model centered at 0

Let us start with the minimization of the objective function

$$C(x) = \|Ax - b\|^2 + \|\Gamma x\|^2 = \langle Ax - b, Ax - b \rangle + \langle \Gamma x, \Gamma x \rangle.$$  

The gradient if $C(x)$ is

$$\nabla_x C = 2\langle A, Ax - b \rangle + 2\langle \Gamma, \Gamma x \rangle.$$  

Making the gradient equal to zero means

$$A^*(Ax - b) + \Gamma^*\Gamma x = 0.$$  

That is

$$x = (A^*A + \Gamma^*\Gamma)^{-1}A^*b.$$  

This is the result shown in Wikipedia. The matrix is non-singular provided that $\Gamma$ is positive definite. If $A^*A$ is non–singular $\Gamma$ could be 0.

\(^1\)http://en.wikipedia.org/wiki/Tikhonov_regularization
C.2 Model centered at $x_0$ and the Mahalanobis norms

C.2.1 Model centered at $x_0$

Let us assume that the model has mean (expected value) at $x_0$. Then it is convenient to shift the coordinate system to be centered around $x_0$.

Let us call the translation in the model space $y = x - x_0$, and the translation in the data space $c = b - Ax_0$. Then we claim

$$C(x) = \|Ax - b\|^2 + \|\Gamma x\|^2 \iff C(y) = \|Ay - c\|^2 + \|\Gamma y\|^2$$

Which has a solution

$$y = (A^*A + \Gamma^*\Gamma)^{-1}A^*c.$$  

That is, the problem

$$C(x) = \|Ax - b\|^2 + \|\Gamma(x - x_0)\|^2$$  \hspace{1cm} (C.2.1)

has the solution

$$x = x_0 + (A^*A + \Gamma^*\Gamma)^{-1}A^*(b - Ax_0).$$  \hspace{1cm} (C.2.2)

C.2.2 The Mahalanobis norms

The Mahalanobis norm of $x$ according to the matrix $P$ is defined as

$$\|x\|_P = x^*Px.$$  

where $P$ is a positive definite matrix. The regular $\ell_2$ norm is a Mahalanobis norm with $P = I$.

The Generalized Tikhonov Regularization problem is given by

$$C(x) = \|Ax - b\|^2_P + \|x - x_0\|^2_Q,$$  \hspace{1cm} (C.2.3)

where $P$ and $Q$ are positive definite matrices. Since $Q$ is positive definite we can write $Q = \Gamma^*\Gamma$, (Cholesky decomposition). \footnote{https://en.wikipedia.org/wiki/Cholesky_decomposition} Then the generalized problem can be written as

$$C(x) = \|Ax - b\|^2_P + \|\Gamma(x - x_0)\|^2.$$
where we recognize the second term on the right, as the second term on the right of the cost function C.2.1. Let us transform the first term. Assume that
\[
P = \Sigma^*\Sigma,
\]
from Cholesky decomposition, then
\[
\|Ax - b\|_P^2 = (Ax - p)^*P(Ax - p) = (Ax - p)^*\Sigma^*\Sigma(Ax - p) = (\Sigma Ax - \Sigma c)^*(\Sigma Ax - \Sigma p)
\]
so we can make the following notation
\[
\Sigma A = B \quad \Sigma b = c,
\]
and rewrite the first term as
\[
\|Ax - b\|_P^2 = \|Bx - c\|.
\]
Then, the Generalized Tikonov problem C.2.3 is now the minimization of the cost function
\[
C(x) = \|Bx - c\| + \|\Gamma(x - x_0)\|^2,
\]
which we know how to invert from section C.2.1 The inversion formula for this problem is from C.2.2,
\[
\begin{align*}
x &= x_0 + (B^*B + \Gamma^*\Gamma)^{-1}B^*(c - Bx_0).
\end{align*}
\]
Now,
\[
B^*B = A^*\Sigma^*\Sigma A = A^*PA, \quad \Gamma^*\Gamma = Q \quad \text{and} \quad B^*c = A^*\Sigma^*\Sigma b = A^*Pb,
\]
from which
\[
\begin{align*}
x &= x_0 + (A^*PA + Q)^{-1}A^*P(b - Ax_0). \tag{C.2.4}
\end{align*}
\]
which is the solution in Wikipedia.
Appendix D

The Rayleigh–Sommerfeld Integral derived from the Angular Spectrum

The Rayleigh–Sommerfeld formula is derived in Goodman. For a full reference see: [?]. Here we derive that formula based only on the concept of angular spectrum (also in Goodman). In addition some important formulas, needed for the main derivation, are derived. These are

(i) The 2D Stationary Phase Approximation. See Appendix F.A

(ii) The Derivation of the Fresnel Integral:
\[ \int_{-\infty}^{\infty} e^{j\omega \nu^2} d\nu, \]
using contour integration. See Appendix D.A

(iii) The Fourier Transform of the Fresnel integral kernel. See Appendix ??

D.1 The Angular Spectrum

The Fourier transform can be seen as a superposition of plane waves. Assume that the aperture, or set of sources is confined to a region of the XY plane, and they are represented by the function \( U(x, y, 0) \), indicating that we are interested on seen the propagation along the z direction.

The 2D Fourier transform for the function \( U(x, y, 0) \) is given by

\[
A(k_x, k_y, 0) = \frac{1}{2\pi} \int_{-\infty}^{\infty} dx \int_{-\infty}^{\infty} dy \: U(x, y, 0) e^{-j(k_x x + k_y y)},
\]
with \( k_x \) and \( k_y \) the wave numbers along the \( x \) and \( y \) directions respectively, and \( j = \sqrt{-1} \).
This is known as the *angular spectrum*. As said above, this is seen as a superposition of plane waves with phase

\[ \phi = k_x x + k_y y. \]

We now assume that the plane waves have propagated some distance \( z \) along the \( z \) axis. The propagation of the wave is done along the vector

\[ k = (k_x, k_y, k_z). \]

From the Helmholtz equation we find the dispersion relation

\[ k_x^2 + k_y^2 + k_z^2 = \frac{\omega^2}{c^2} \]

where \( \omega \) is the circular frequency and \( c \) is the wavespeed (assumed constant). We will refer to \( \omega/c \) as simply \( k^2 \) which is the square of the magnitude of the wavenumber \( k \), and hence we write

\[ k_x^2 + k_y^2 + k_z^2 = k^2, \]

From here we have that

\[ k_z = \pm \sqrt{k^2 - k_x^2 - k_y^2}. \]

We pick the “+” sign to consider displacement only along the positive \( Z \) direction. If the plane was displaced a distance \( z \) along the \( Z \) axis, then this is represented in the Fourier domain as \( \exp(jk_z z) \), then at that \( z \) we should have

\[ U(x, y, z) = e^{-jk_z z} U(x, y, 0), \]

with

\[ U(x, y, 0) = \frac{1}{2\pi} \int_{-\infty}^{\infty} dk_x dk_y A(k_x, k_y, 0) e^{j(k_x x + k_y y)}. \]

Hence

\[
U(x, y, z) = \frac{1}{2\pi} \int_{-\infty}^{\infty} dk_x dk_y A(k_x, k_y, 0) e^{j(k_x x + k_y y + k_z z)} \\
= \frac{1}{2\pi} \int_{-\infty}^{\infty} dk_x dk_y A(k_x, k_y, 0) e^{j(k_x x + k_y y + \sqrt{k^2 - k_x^2 - k_y^2})} \\
= \frac{1}{4\pi^2} \int_{-\infty}^{\infty} dk_x dk_y \int_{-\infty}^{\infty} d\chi d\upsilon U(\chi, \upsilon, 0) e^{-j(\chi x + \upsilon y)} e^{j(k_x x + k_y y + \sqrt{k^2 - k_x^2 - k_y^2})}
\]
That is
\[ U(x, y, z) = \frac{1}{4\pi^2} \int_{-\infty}^{\infty} d\chi d\upsilon \int_{-\infty}^{\infty} dk_x dk_y U(\chi, \upsilon, 0) e^{jk_x(x-\chi)+k_y(y-\upsilon)+\sqrt{k_x^2-k_y^2-k^2 z^2}}. \] (D.1.1)

We simplify this integral by collapsing the \( k_x, k_y \) integrals using the method of stationary phase. That is, we want to evaluate the integral
\[ I = \int_{-\infty}^{\infty} dk_x dk_y e^{jk_x(x-\chi)+k_y(y-\upsilon)+\sqrt{k_x^2-k_y^2-k^2 z^2}}. \] (D.1.2)

### D.2 The Derivation: The method of stationary phase

The method of stationary phase for 2D integrals is explained in Appendix F.A. Let us write integral D.1.2 as
\[ I = \int_{-\infty}^{\infty} dk_x dk_y e^{jk\Phi(k_x,k_y)} \]

with
\[ \Phi(k_x, k_y) = \frac{k_x}{k}(x-\chi) + \frac{k_y}{k}(y-\upsilon) + \sqrt{1 - \left(\frac{k_x}{k}\right)^2 - \left(\frac{k_y}{k}\right)^2} z. \] (D.2.3)

The method of stationary phase consist in the evaluation of the integral shown in equation D.C.14. This can be computed in three steps.

(i) Expand the phase in a Taylor series about the wave numbers (the high value parameters).

(ii) Find the stationary phase. The values of the wavenumbers that annihilate the gradient of the phase.

(iii) Find the Hessian (correction for curvature).

(iv) Evaluate the integral.

#### D.2.1 Taylor series on the phase

We want to expand the phase in a Taylor series up to second order around a point \((k_{x0}, k_{y0})\) such that \(\nabla_{(k_x,k_y)} \Phi = 0\) (the stationary point). That is,
\[ \Phi(k_x, k_y) = \Phi + \Phi_{,i}(k_{i0} - k_i) + \Phi_{,ij}(k_{i0} - k_i)(k_{j0} - k_j) + \cdots \]
where \( \Phi, i \) means derivative with respect to \( k_x \) for \( i = 1 \) and \( k_y \) for \( i = 2 \). Similarly for the second derivative tensor (Hessian) \( \Phi, ij \), with either \( i \) or \( j \) taking the values 1 and 2 for \( k_x \) and \( k_y \) respectively. Also, \( \Phi \) is evaluated at the point \((k_{x0}, k_{y0})\) in all terms. We assume sums over each repeated indices. We will only consider the expansion up to the second order.

\[
\frac{\partial \Phi}{\partial k_x} = \Phi,1 = \frac{x - \chi}{k} - \frac{k_x z}{k \sqrt{k^2 - k_x^2 - k_y^2}}
\]

\[
\frac{\partial \Phi}{\partial k_y} = \Phi,2 = \frac{y - \upsilon}{k} - \frac{k_y z}{k \sqrt{k^2 - k_x^2 - k_y^2}}.
\]

### D.2.2 The stationary phase

We want to impose \( \Phi, i = 0 \), for \( i = 1, 2 \) and find \( k_{x0} \) and \( k_{y0} \) such that this happens. Let us square both equations D.2.4 and add them together, and call \( r^2 = (x - \chi)^2 + (y - \upsilon)^2 + z^2 \). Then

\[
\frac{(x - \chi)^2 + (y - \upsilon)^2}{k^2} = \frac{z^2(k_x^2 + k_y^2)}{k^2(k^2 - k_x^2 - k_y^2)}
\]

\[
r^2 - z^2 = \frac{z^2(k_x^2 + k_y^2)}{k^2 - k_x^2 - k_y^2}
\]

So

\[
(x - \chi)^2 + (y - \upsilon)^2 = \frac{z^2(k_{x0}^2 + k_{y0}^2)}{k^2 - k_{x0}^2 - k_{y0}^2}
\]

\[
r^2 - z^2 = \frac{z^2(k_{x0}^2 + k_{y0}^2)}{k^2 - k_{x0}^2 - k_{y0}^2} \quad \text{from definition of } r \text{ above}
\]

\[
(k^2 - k_{x0}^2 - k_{y0}^2)(r^2 - z^2) = z^2(k_{x0}^2 + k_{y0}^2)
\]

\[
k^2(r^2 - z^2) = (r^2 - z^2)(k_{x0}^2 + k_{y0}^2)
\]

\[
k_{x0}^2 + k_{y0}^2 = \frac{k^2(r^2 - z^2)}{r^2}
\]

Now from the first of equations D.2.4 we see that

\[
(x - \chi)^2(k^2 - k_{x0}^2 - k_{y0}^2) = k_{x0}^2 z^2
\]

\[
(D.2.6)
\]

\[
(D.2.7)
\]
and so and using equation D.2.5 here,

\[(x - \chi)^2 \left( k^2 - \frac{k^2(r^2 - z^2)}{r^2} \right) = k^2 z_x^2 \]

\[(x - \chi)^2 \left( \frac{x^2 k^2 - k^2 r^2 + k^2 z^2}{r^2} \right) = k^2 z_x^2 \]

so

\[k_{x_0} = \pm \frac{(x - \chi)k}{r} \]

The two solutions are for two plane waves propagating in opposite directions. We pick the “+” solution. The solution for \(k_{y_0}\) is obtained similarly. That is, we find

\[k_{x_0} = \frac{(x - \chi)k}{r} \quad \text{(D.2.8)} \]

\[k_{y_0} = \frac{(y - \chi)k}{r} \quad \text{(D.2.9)} \]

From equation D.2.6 we find and the first of the previous equations

\[k^2 - k_{x_0}^2 - k_{y_0}^2 = \frac{k_{x_0}^2 z_x^2}{(x - \chi)^2} = \frac{(x - \chi)^2 k^2 z_x^2}{(x - \chi)^2 r^2} \]

We pick the positive square root

\[\sqrt{k^2 - k_{x_0}^2 - k_{y_0}^2} = \frac{k z_x}{r} \Rightarrow \sqrt{1 - \frac{k_{x_0}^2 k}{k} - \frac{k_{y_0}^2 k}{k}} = \frac{z}{r} \quad \text{(D.2.10)} \]

We now evaluate D.2.3.

\[\Phi(k_{x_0}, k_{y_0}) = \frac{k_{x_0}}{k}(x - \chi) + \frac{k_{y_0}}{k}(y - \chi) + \sqrt{1 - \frac{k_{x_0}^2 k}{k} - \frac{k_{y_0}^2 k}{k} z} \]

\[= \frac{(x - \chi)^2}{r} + \frac{(y - \chi)^2}{r} + \frac{z^2}{r} = r \]

This is

\[\Phi(k_{x_0}, k_{y_0}) = r \quad \text{(D.2.11)} \]

Which physically makes sense. This is the distance between the input source point location \((\chi, \nu, 0)\) and the observation point \((x, y, z)\).
D.2.3 The Hessian

The Hessian is usually the most cumbersome part of the method of stationary phase. In the 2D case it implies a 2x2 matrix of four partial derivatives. We already, from equation D.2.4 have the first partial derivatives.

Now,

$$\Phi_{,11} = -\frac{z}{k^2 z/r} - \frac{k_x^2 z}{k(kz/r)^3} = -\frac{r}{k^2} - \frac{k_x^2 r^3}{k^2 (kz)^2} = -\frac{r z^2 k^2 + k_x^2 r^3}{k^2 (kz)^2}$$

but we know $k_x$ from the first of equations D.2.8. Then

$$\Phi_{,11} = -\frac{r(z^2 + (x - \chi)^2)}{k^2 (kz)^2}$$

From the symmetry in the $x$ and $y$ variables we also find

$$\Phi_{,22} = -\frac{r[z^2 + (y - \upsilon)^2]}{k^2 z^2}$$

Now for the cross terms (mixed derivatives).

$$\Phi_{,12} = -\frac{k_x k_y z}{k(k^2 - k_x^2 - k_y^2)^{3/2}}$$

and in the stationary phase point this is

$$\Phi_{12} = -\frac{k_x k_y z}{k(kz/r)^3} = -\frac{r^3[(x - \chi)/r][k(y - \upsilon)/r]}{k^4 z^2} = -\frac{r k^2 (x - \chi)(y - \upsilon)}{z^2}$$

We also have $\Phi_{,12} = \Phi_{,21}$. The Hessian is

$$\det \Phi_{,ij} = \Phi_{,11} \Phi_{,22} - \Phi_{,12}^2$$

$$= \frac{r^2[z^2 + (x - \chi)^2][z^2 + (y - \upsilon)^2]}{k^2 z^4} - \frac{r^2(x - \chi)^2(y - \upsilon)^2}{k^2 z^4}$$

$$= \frac{r^2 z^4 + r^2 z^2(y - \upsilon)^2 + r^2 z^2(x - \chi)^2}{k^2 z^4}$$

$$= \frac{r^2 z^2(z^2 + (y - \upsilon)^2 + (x - \chi)^2)}{k^2 z^4}$$

$$= \frac{r^4 z^2}{k^2 z^4}$$

$$= \frac{r^4}{k^2 z^2}$$
With this, we are ready to evaluate the stationary phase formula D.C.14 in integral D.1.1. That is, since the Hessian is $r^4/z^2$ both eigenvalues are either negative or positive. Since the matrix traces is $\lambda_1 + \lambda_2 = \Phi_{,11} + \Phi_{,22} < 0$ both eigenvalues are negative and then $\text{SGN}[H] = -2$. So,

$$U(x, y, z) = \frac{k^2}{2\pi k} \int_{-\infty}^{\infty} d\chi d\nu \frac{1}{\sqrt{r^4/z^2}} U(\chi, \nu, 0) e^{ikr} e^{-j\pi/2}$$

$$= \frac{k}{2\pi} \int_{-\infty}^{\infty} d\chi d\nu \frac{z}{r} U(\chi, \nu, 0) e^{ikr} e^{-j\pi/2}$$

$$= \frac{1}{j\lambda} \int_{-\infty}^{\infty} d\chi d\nu \cos \theta U(\chi, \nu, 0) \frac{e^{ikr}}{r},$$

where $\cos \theta$ is the obliquity factor, $\theta$ being the angle between the incident normal and the transmitted normal. This is precisely the Rayleigh-Sommerfeld integral, equation (3-41) in Goodman’s book.

## D.A The Fresnel Integral

The method of stationary phase is based on the computation of the following integral

$$\int_{-\infty}^{\infty} e^{j\omega \nu^2} d\nu = 2 \int_{0}^{\infty} e^{j\omega \nu^2} d\nu. \quad \text{(D.A.12)}$$

We compute this integral (assuming $\omega > 0$ ) as a contour along the contour in Figure D.1.

From the Cauchy residue theorem the integral along the closed loop is 0. From the Jordan Lemma, the integral along the arc as the radius goes to infinity is 0, since
in the upper half plane the function $\exp(j\nu^2)$ goes to 0 when $\nu$. We then have that
\[
\int_{C_1} d\nu e^{j\omega\nu^2} = \int_{-C_3} d\nu e^{j\omega\nu^2}.
\]
Now, along the curve $-C_3$ which is the curve $C_3$ in opposite order we have
\[
\int_{-C_3} d\nu e^{j\omega\nu^2} \quad = \quad \int_0^\infty dx e^{j\omega(x+ja)^2} \\
\quad = \quad \int_0^\infty dx e^{j\omega(x^2+2jx^2-x^2)} + j\int_0^\infty dx e^{j\omega(x^2+2jx^2-x^2)} \\
\quad = \quad \int_0^\infty dx e^{-2\omega x^2} + j\int_0^\infty dx e^{-2\omega x^2}
\]
which is the integral of a Gaussian. This is a well known and can be easily computed.\(^1\) The formula for the integral of the Gaussian is
\[
\int_0^\infty e^{-ax^2} dx = \frac{1}{2}\sqrt{\frac{\pi}{a}}.
\]
So, since $C_1$ is the contour from 0 to $\infty$ we find that
\[
\int_0^\infty d\nu e^{j\omega\nu^2} = \sqrt{\frac{\pi}{2\omega}} + j\sqrt{\frac{\pi}{2\omega}} = \sqrt{\frac{2\pi}{\omega}(1+j)} = \sqrt{\frac{2\pi}{\omega}e^{j\pi/4}}.
\]
If instead we choose $\omega < 0$, we would (choosing the contours that are a reflection with respect to the x axis of the contours in Figure D.1), obtain
\[
\int_{-\infty}^\infty d\nu e^{j\omega\nu^2} = \sqrt{\frac{2\pi}{-\omega}}e^{-j\omega\pi/4}.
\]
Then we have the following important result
\[
\int_{-\infty}^\infty d\nu e^{j\omega\nu^2} = \frac{\sqrt{2\pi}}{|\omega|}e^{j\text{sgn}(\omega)\pi/4}. \tag{D.A.13}
\]
This is the key equation to solve the stationary phase method.

**D.B The Fourier Transform of the Fresnel integral kernel.**

Let us define the Fresnel kernel as
\[
f(x) = e^{jk^2x^2},
\]
\(^1\)http://en.wikipedia.org/wiki/Gaussian_integral
and the one-dimensional Fourier transform as

\[ F(k) = \int_{-\infty}^{\infty} dx \ e^{-jkx} f(x). \]

Hence, for the Fresnel kernel we have

\[ F(k) = \int_{-\infty}^{\infty} dx \ e^{jk_0^2 x^2 - (k/k_0^2)x} \]

We perform square completion in the phase. That is, we write

\[ x^2 - \frac{k}{k_0^2} x = x^2 - \frac{k}{k_0^2} x + \frac{k^2}{4k_0^4} - \frac{k^2}{4k_0^2} = \left( x - \frac{k}{2k_0^2} \right)^2 - \frac{k^2}{4k_0^4} \]

and then

\[ F(k) = \int_{-\infty}^{\infty} dx \ e^{jk_0^2 \left( x - \frac{k}{2k_0^2} \right)^2}. \]

A change of variables \( y = x - k/2k_0^2 \) leaves us with

\[ F(k) = \int_{-\infty}^{\infty} dy \ e^{jk_0^2 y^2}. \]

From equation D.A.13, and since \( \text{sgn}(k_0^2) = 1 \), we find that

\[ F(k) = \frac{e^{-jk^2/4k_0^2 \sqrt{2\pi}}}{|k_0|} \frac{\sqrt{2\pi}}{|k_0|} \sqrt{\frac{j}{2}} \]

That is,

\[ F(k) = \frac{\sqrt{j\pi}}{|k_0|} e^{-jk^2/4k_0^2} \]

Written in a different way

\[ e^{jk_0^2 x^2} \xrightarrow{FT} \frac{\sqrt{j\pi}}{|k_0|} e^{-jk^2/4k_0^2} \]

By the symmetry theorem of Fourier transforms we can write

\[ e^{jk_0^2 x_0^2} \xrightarrow{FT} 2|x_0| \frac{\sqrt{\pi}}{\sqrt{j}} e^{-jk^2/4x_0^2} \]

Note the \( k_0^2 \) which is put this way to make the argument of the exponential dimensionless.
APPENDIX D. THE RAYLEIGH–SOMMERFELD INTEGRAL DERIVED FROM
THE ANGULAR SPECTRUM

D.C The Method of Stationary Phase for 2 fold integrals

Let us assume that we want to evaluate the integral

\[ I = \int_S dx \, dy \, A(x, y) \, e^{j \Phi(x, y)} \]

where \( k \) is a “large” number (creates high oscillations). \( S \) is a surface having the stationary phase points inside (we define stationary phase points next). Let us assume that \((x_s, y_s)\) is a stationary phase point. That is,

\[ \nabla \Phi(x_s, y_s) (\Phi_x(x_s, y_s), \Phi_y(x_s, y_s)) = 0. \]

The Taylor expansion of \( \Phi(x, y) \) around \((x_s, y_s)\) is given by

\[ \Phi(x, y) = \Phi(x_s, y_s) + \frac{1}{2} \left[ a(x - x_s)^2 + 2b(x - x_s)(y - y_s) + c(y - y_s)^2 \right] \]

where

\[ a = \frac{\partial^2 \Phi}{\partial x^2} (x_s, y_s) \quad b = \frac{\partial^2 \Phi}{\partial x \partial y} (x_s, y_s) \quad a = \frac{\partial^2 \Phi}{\partial y^2} (x_s, y_s) \]

We substitute the Taylor expansion into our integral and find

\[ I \approx A(x_s, y_s) \int_S du \, dv \, e^{j \frac{1}{2} \left[ au^2 + 2buv + cv^2 \right]} \]

with \( u = x - x_s \), and \( v = y - y_s \).

We want to decouple the quadratic formula in the phase. The Hessian matrix is

\[ H = \begin{pmatrix} a & b \\ b & c \end{pmatrix} \]

The phase can be written as

\[ \frac{1}{2} U^T H U \]

with

\[ U = \begin{pmatrix} u \\ v \end{pmatrix} \]

To decouple the equation (put it as a sum of squares) we have to diagonalize the matrix \( H \). Since the matrix is symmetric it can be diagonalized.
shows two different proofs for this important theorem called The principal axis or Spectral theorem. The meaning of diagonalization is that $H$ can there is a diagonal matrix $D$ and a non-singular matrix $Q$ such that

$$QHQ^{-1} = D.$$ 

Since $H$ is symmetric there is an additional property. This is, $Q$ is orthogonal. The matrix $D$ is the matrix of eigenvalues of $H$ and the columns of $Q$ are the eigenvectors of $H$ corresponding to different eigenvalues of $H$ or spanning a space for repeated eigenvalues of $H$. It is said that the matrix $H$ is semi-simple. This is really the key to decouple the stationary phase method from multiple dimensions to a product of integrals on one dimension each. The idea is that a change of variables from the original axis $(u, v)$ to a new axis such that the phase term is decoupled is orthonormal. That is by picking the new axis along the columns of the matrix $Q$ the phase is automatically decoupled. Even better, since $\det Q = 1$ the Jacobian of the transformation is 1, and so we just can write

$$I \approx A(x_s, y_s) e^{jk\Phi(x_s, y_s)} \int_S d\xi_1 d\xi_2 e^{j\frac{k}{2}\left[\lambda_1 \xi_1^2 + \lambda_2 \xi_2^2\right]}$$

where we recognize that each of the integrals at the end correspond with the one-dimensional case. We still have to consider an important detail. Whether the sign of $\lambda_i$ is positive or negative we pick the contour in Figure D.1 or its mirror image with respect to the x axis for the computation of the Fresnel integral. That is we find (see equation

$$\int_{-\infty}^{\infty} d\nu \ e^{\pm j\lambda \nu^2} = \sqrt{\frac{2\pi}{|\lambda|}} e^{\text{sgn}(\lambda) j\pi/4}.$$ 

We are ready to compute the value of integral $I$. That is, the product is

$$\int_S dx \ dy \ A(x, y) e^{-jk\Phi(x, y)} = A \frac{2\pi}{k \sqrt{|\det H|}} e^{-jk\Phi} e^{\text{SGN}[H] j\pi/4}. \quad (D.C.14)$$

Where $\det H$ is the product $\lambda_1 \lambda_2$ is the Hessian (determinant of $H$) and SGN is the signature of the matrix (number of the difference between positive and negative eigenvalues). The functions $A, H,$ and $\Phi$ are assumed to be evaluated at the (only) stationary phase point $(x_s, y_s).$ This is the stationary phase formula for 2D. It is generalized to n dimensions along the same lines.

Appendix E

The Huyghens-Fresnel Principle

In the first part of this appendix, we show a derivation of the Huyghens-Fresnel principle based on the classical Born and Wolf’s “Principles of Optics” classical book. In the second part we verify that the this principle indeed accomplishes what Huyghens predicted, by using the method of stationary phase.

E.1 The Principle

These notes are taken from the classical Principles of Optics classical book.

Referring to figure E.1 the Huygens principle states that the energy recorded at point of a propagating wave, is due to secondary sources all located to a previous wavefront (the blue wavefront). Fresnel went a bit further and used the interference principle (superposition) to add the contributions in a mathematical way to obtain an integral representation.

We assume that the initial wave is a pure point source with a given spectrum $F(\omega)$ and at any point in a distance from the source we have the wavefield described by the Green’s function

$$W_0 = A \exp (-j\omega t_0) F(\omega) \frac{\exp(jkr_0)}{r_0} \quad \text{(E.1.1)}$$

The factor $A$ is a constant due to the source strength, and we will omit $\exp (-j\omega t_0) F(\omega)$ because it will not be used for any of the computations here. This could be inserted at the end of all the computations.

Now the share of this source in the total wavefield in $P$ is given by

$$W_0 K(\chi) \frac{\exp(jks)}{s}.$$  

where $W_1$ given by equation E.1.1 is the strength of the secondary source and $K(\chi)$ is an angle dependent weight factor that is initially assume unknown. This is the so-called obliquity factor.
Figure E.1: A wave starts at a point source $P$. After sometime, a wavefront is located in the blue sphere (only a main cross–section in the $xy$ plane is shown). A disturbance at point $P$, according to Huygens principle is understood as the superposition of all disturbances starting at the wavefront.

The integral becomes

$$U(P) = \int_S W_0 K(\chi) \frac{\exp(jks)}{s} dS$$

$$= \int_S K(\chi) \exp(jkr_0) \frac{\exp(jks)}{s} dS$$

$$= A \exp(jkr_0) \int_S K(\chi) \frac{\exp(jks)}{s} dS.$$  \hspace{1cm} (E.1.2)

Fresnel decided to integrate the effect of all sources in zones as shown in the figure. The zone $Z_1$ is the piece of sphere which is between the nearest wavefront point and half wavelength $\lambda/2$ farther. In the figure these are the points along the sphere between distances $b$ and $b + \lambda/2$ from $P$, with $b = CP$ being the the distance between the point $P$ and the wavefront. The second zone $Z_2$ contains the peace of wavefront which is between the distances $b + \lambda/2$ and $b + \lambda$. The constructions continues in this way so that the zone $Z_i$ is between the distances $b + (i – 1)(\lambda/2)$ and $b + i\lambda/2$. The partition in these type of slices is quite interesting as we will observe.

One assumption for the evaluation of the integral is that in each slice the obliquity
factor remains constant. Hence the integral for the \(j\)–th slice is
\[
U_j(P) = A \frac{\exp jkr_0}{r_0} K(\chi) \int_{Z_j} \frac{\exp (jks)}{s} r_0^2 \sin \theta d\theta d\phi.
\]

where we use the transformation into polar spherical coordinates with
\[
dxdy = Jd\theta d\phi,
\]
with \(J = r_0^2 \sin \theta\), \(\theta\) is the polar angle and \(\phi\) is the azimuthal angle. Since the integral
is azimuthally invariant
\[
U_j(P) = 2\pi A \frac{\exp jkr_0}{r_0} K(\chi) \int_{Z_j} \frac{\exp (jks)}{s} r_0^2 \sin \theta d\theta.
\] (E.1.3)

From the cosine law we see that
\[
s^2 = r_0^2 + (r_0 + b)^2 - 2r_0(r_0 + b) \cos \theta,
\]
and so
\[
sds = r_0(r_0 + b) \sin \theta d\theta,
\]
so
\[
\sin \theta d\theta = \frac{sds}{r_0(r_0 + b)},
\]
and integral E.1.4 becomes
\[
U_j(P) = 2\pi A \frac{\exp ikr_0}{r_0} K_j(\chi) \int_{b+(j-1)\lambda/2}^{b+j\lambda/2} \frac{\exp (iks)}{s} r_0^2 \frac{sds}{r_0(r_0 + b)}
\]
\[
= 2\pi \frac{Ar_0}{r_0 + b} \frac{\exp ikr_0}{r_0} K_j(\chi) \int_{b+(j-1)\lambda/2}^{b+j\lambda/2} \exp (iks) ds
\]
\[
= -\frac{2\pi i A \exp ik(r_0 + b)}{k} \frac{r_0}{r_0 + b} K_j(\chi) \exp (ikj\lambda/2)[1 - \exp (-ik\lambda/2)].
\]

Since \(k\lambda = 2\pi\), the last two factors reduce to
\[
\exp (jk\lambda/2)[1 - \exp (-jk\lambda/2)] = \exp (jj\pi)(1 - \exp [-j\pi]) = 2(-1)^j
\]
so
\[
U_j(P) = 2j\lambda(-1)^{j+1} \frac{A \exp jk(r_0 + b)}{r_0 + b} K_j(\chi).
\] (E.1.4)
From now on, to simplify notation, we write $K_j$ instead of $K_j(\chi)$. The total field is the sum of all contributions $U_j(P)$ with alternating signs. That is

$$U(P) = \sum_{j=1}^{n} U_j(P)$$

$$= 2j\lambda(-1)^{j+1}\frac{A\exp jk(r_0 + b)}{r_0 + b}\sum_{j=1}^{n} K_j.$$  \hspace{1cm} (E.1.5)

What is $n$? This is a source of controversy. Fresnel claimed that all he needed is to cover the forward (we mean to the right in the figure) propagating front. That is $0 \leq \chi \leq \pi/2$. This was put in question for several scientists since the waves propagating to the left also send pulses to the receiver and since the receiver is static they should arrive at some time and interference should occur at that time. For example this link discusses in detail the issue of the “left propagating” wavefront. With that said, the number $n$ should be such that $n\lambda/2 \geq r_0$ so that all sphere is covered.

We will evaluate the series

$$\sigma = \sum_{j=1}^{n} (-1)^{j+1} K_j.$$  \hspace{1cm} (E.1.6)

Born and Wolf use a method due to A. Schuster\(^1\) to evaluate this sum. They first write E.1.6 as follows

$$\sigma = \frac{K_1}{2} + \left( \frac{K_1}{2} - K_2 + \frac{K_3}{2} \right) + \left( \frac{K_3}{2} - K_4 + \frac{K_5}{2} \right) + \cdots + \text{Tail}.$$  \hspace{1cm} (E.1.7)

The Tail is

$$\text{Tail} = \begin{cases} 
\frac{K_n}{2} & \text{if } n \text{ odd} \\
\frac{K_{n-1}}{2} - K_n & \text{if } n \text{ even}
\end{cases}$$

Born and Wolf refer to a law that says that “the directional variation is such that $K_j$ is greater than the arithmetic mean of its two neighbors $K_{j-1}$ and $K_{j+1}$”. we have no clue where this law comes from, but the $\cos\chi$ ($0 \leq \chi \leq \pi$) function satisfies such a condition as it does any concave down function, since the average is a linear interpolation and a line is under any concave down curve. If this is the case then any partial sum in parenthesis in E.1.7 is negative, so

$$\sigma \leq \begin{cases} 
\frac{K_1}{2} + \frac{K_n}{2} & \text{for } n \text{ odd} \\
\frac{K_1}{2} + \frac{K_{n-1}}{2} - K_n & \text{for } n \text{ even}
\end{cases}$$

---

\(^1\)A. Schuster, *Phil. Mag.* (5), 31 (1891), p. 77. At this time we do not know what Fresnel did because he died in 1827, so obviously he did not know A. Schuster work.
We can also write E.1.6 in the form
\[ \sigma = K_1 - \frac{K_2}{2} - \left( \frac{K_2}{2} - K_3 + \frac{K_4}{2} \right) - \left( \frac{K_4}{2} - K_5 + \frac{K_6}{2} \right) - \cdots + \text{Tail} \]
with
\[ \text{Tail} = \begin{cases} 
-\frac{K_{n-1}}{2} + K_n & \text{for } n \text{ odd} \\
-\frac{K_n}{2} & \text{for } n \text{ even}
\end{cases} \]

Hence
\[ \sigma \geq \begin{cases} 
K_1 - \frac{K_2}{2} - \frac{K_{n-1}}{2} + K_n & \text{for } n \text{ odd} \\
K_1 - \frac{K_2}{2} - \frac{K_n}{2} & n \text{ for } n \text{ even}
\end{cases} \]

We found then that
\[ K_1 - \frac{K_2}{2} - \frac{K_{n-1}}{2} + K_n \leq \sigma \leq \frac{K_1}{2} + \frac{K_n}{2} \quad \text{for } n \text{ odd} \]
\[ K_1 - \frac{K_2}{2} - \frac{K_n}{2} \leq \sigma \leq \frac{K_1}{2} + \frac{K_{n-1}}{2} - K_n \quad \text{for } n \text{ even} \]

Next, is a kind of high frequency assumption. If the wavenumber is large, then \( \lambda \ll 1 \) and \( K_i \approx K_{i+1} \), so we can approximate
\[ \sigma = \begin{cases} 
\frac{K_1}{2} + \frac{K_n}{2} & \text{for } n \text{ odd} \\
\frac{K_1}{2} - \frac{K_n}{2} & \text{for } n \text{ even}
\end{cases} \]

Born and Wolf proceed to weaken the argument about the convexity of the directivity factor (\( \cos \chi \) like behavior) by saying that even if the directivity function is concave up the conclusion would be the same and further that even if the difference of a \( K_j \) with the average of its neighbors oscillate the argument will still apply.

we do not think there is much point here. An approximation is an approximation, an assumption is an assumption. We know from the more rigorous mathematical (Kirchhoff–Sommerfeld using Helmholtz equation) approach that the directivity is a \( \cos \chi \)–like function. The purpose of the analysis shown here is more historical and to see how the big scientists approached the problem. With this last result, applied to equations E.1.5 and E.1.4 we find
\[ U(P) = i\lambda \frac{A \exp [ik(r_0 + b)]}{r_0 + b} (K_1 \pm K_n) = \frac{1}{2} [U_1(P) + U_n(P)] \quad (E.1.8) \]
where the “+” is for odd \( n \) and the “-” is for even \( n \).

The last \( K_n \) is such that \( \chi = \pi/2 \). In this case the ray path is tangent to the sphere and it can not reach \( P \), so \( K_n = 0 \). Then from equation E.1.8

\[
U(P) = i\lambda K_1 \frac{A \exp [i k (r_0 + b)]}{r_0 + b} = \frac{1}{2} U_1(P)
\]

A major finding here is that “the total disturbance at \( P \) is equal to half the disturbance due to the first zone”\(^2\).

The next step on the development is crucial and it will show that the 90 degrees phase shift is needed to correctly predict the Huygens principle according to the wave equation. To validate Huygens principle we should have

\[
K_1 = -\frac{i}{\lambda} = \exp \left[ -i \pi/2 \right] \frac{\lambda}{\lambda}.
\]

If this is the case, then

\[
U(P) = \frac{A \exp [i k (r_0 + b)]}{r_0 + b},
\]

which is exactly what the wavefield would be for a spherical wave propagating in free space. This validates the Huygens principle. This is reaffirmed by using the method of stationary phase in my notes with title “The Huygens Principle” (2013).

Note that these does not provide information about what the wavefield above or below \( P \), and the point \( P \) should be such that the intersecting spheres from \( P \) to the wavefront \( S \) should have \( P_0 P \) as the main axis. That is, there is totally symmetry on the figure with respect to the angle \( \theta \).

The Fresnel zone partition provides a way to know how much contribution to the total wavefield we would obtain by putting obstructing circular zones of different radii along the \( P_0 P \) axis.

For example, if all zones are present, except the first zone \( Z_1 \), then

\[
U(P) = \frac{2i\lambda K_1 A \exp [i k (r_0 + b)]}{r_0 + b} = 2 \frac{A \exp [i k (r_0 + b)]}{r_0 + b},
\]

and the intensity \( I(P) = |U(P)|^2 \) is four times larger than if the screen were absent.

Now, if we allow light to go through the first two zones \( Z_1, Z_2 \) and obstruct the other zones, the amplitude should be almost zero, since \( K_1 \approx K_2 \) and they have opposite signs. The amplitude will grow again (but not that much) when letting three zones illuminate the object in \( P \) (that is zones \( Z_1, Z_2, Z_3 \) and will oscillate as more zones are added.

\(^2\)Now known as first Fresnel zone
Here is a prediction that made a strong impression in Fresnel contemporaries. What if only the first zone is obstructed and all other are open for light through. Then we would have

\[ U(P) = \frac{2i\lambda A \exp[ik(r_0 + b)]}{r_0 + b} \left[-K_2 + K_3 - K_4 + \cdots \right], \]

and since the missing \( K_1 \) is approximately half of the alternating sum of the rest of its \( K_j \) partners, with \( K_1 = \frac{1}{i\lambda} \) then

\[ U(P) = \frac{A \exp[ik(r_0 + b)]}{r_0 + b} \]

as if there would not be any block to light. The important thing about this contribution is that it validated the wave theory of light which was rejected by Newton about 200 years before that. Here is the footnote in Born and Wolf’s book: "That a bright spot should appear at the centre of the shadow of a small disc was deduced from Fresnel’s theory by S. D. Poisson in 1818. Poisson who was a member of the committee of the French Academy which reviewed Fresnel’s prize memoir, appears to have considered this conclusion contrary to experiment and so refuting Fresnel’s theory. However, Arago, another member of the committee, performed the experiment and found that the surprising prediction was correct. A similar observation had been made a century earlier by Maraldi but had been forgotten."

**E.2 The Huygens Principle**

Here we try to explain, using the method of stationary phase, why the Huygens Principle makes sense. We start with the Huygens-Fresnel integral representation in equation E.1.2, where \( A = -\frac{i}{\lambda} \), and we call

\[ U_0(x, y, 0) = \frac{\exp ik_0}{r_0}, \]

the obliquity factor

\[ K(\chi) = \frac{1}{2}(\cos \xi + \cos \chi), \]

and \( r = s \) the vector from the aperture to the observation point \( Q \). Please refer to Figure E.2 for this.

Then, using the method of stationary phase, prove that the energy in a point in the wavefront can be seen as a summation of contributions originated from sources on a previous wavefront.
E.2.1 The principle in words

In words it says that each point of a wavefront, is a center of a new disturbance; the advancing wave as a whole may be regarded as the sum of all the secondary waves with sources in a previous wavefront.

This might sound confusing, but if you are reading this is because you already have an idea about what is the Huygens principle.

E.2.2 The Integral Formulation

This Huygens-Fresnel principle establishes that

$$U_{HF}(x_Q, y_Q, z_2) = -\frac{i}{\lambda} \int_{\text{Area}} U_0(x, y, 0) \frac{\exp(ikr)}{r} \frac{1}{2} (\cos\xi + \cos\chi) \, dx \, dy. \tag{E.2.9}$$

We explain the symbols in this equation. Starting with Figure E.2. The figure is a cross-section of the actual experiment. The $y$ axis points out of the visual plane. The $z$ axis is along the horizontal and the $x$ axis is along the vertical. We assume a point source $P$ and a wavefront hitting a barrier, with a small aperture. The purpose of this barrier is to isolate a little piece of wavefront. The area $\text{Area}$ is the piece of spherical surface (we assume constant velocity for simplicity) drawn by the aperture in the $xy$ plane. The integral has as support the $\text{Area}$ surface. The point $Q$ is the output point.
That is \( Q = (x', y', z = z_2) \). We show the rays from the source \( P \) to the aperture part of the wavefront, and the rays from the aperture to the output point \( Q \). The point \( O \) is to reference one typical ray and the associated radius \( r = OQ \), the distance between the \( O \) and \( P \) points, and the two obliquity angles \( \xi \) and \( \chi \). The blue arrow is drawn only for illustrating the angles. We picked the last ray to avoid clustering of symbols, but the illustration should be understood for any ray on the package beam. The distance \( r_P = P0 \) is not shown in the formula explicitly but is part of it. We assume that the wavefront is “flat” due to the large value of the radius \( r_P \), so that \( z_1 \) is constant, this is a high frequency approximation. A curve front will require a change of coordinates (from cartesian to polar) in our integral equation. We are lazy to go in that direction. Then, we can think of that wavefront as an approaching plane wave. \( \lambda \) and \( k \) are related by the formula

\[
\lambda = \frac{2\pi}{k}.
\]

Finally, the field \( U_0(x, y, 0) \) represents the energy along the wavefront through the point \((x, y)\). For a perfect impulsive source (a Dirac delta) this is given by

\[
U_0(x, y, 0) = \frac{\exp(i k r_P)}{r_P}
\]

Hence we can write equation E.2.9 as

\[
U_{HF}(x_Q, y_Q, z_2) = -\frac{i}{\lambda} \int_{\text{Area}} \frac{\exp[i k (r + r_P)]}{r r_P} \frac{1}{2} (\cos \xi + \cos \chi) \, dx \, dy.
\] (E.2.10)

We want to evaluate this formula using the method of stationary phase. The formula for stationary phase is as follows.

Given

\[
I(k) = \int_S A(x, y) \exp[i k \phi(x, y)] \, dx \, dy,
\]

Where \( k \) is large (we will not explain what large means. Bleistein claims that \( K \) large if \( \lambda = 2\pi/k > 3L \), where \( L \) is the “natural” scale of the problem...)

Then \( I(k) \) can be approximated using the stationary phase formula ([?])

\[
I(k) \approx \frac{2\pi}{k \sqrt{|H(x_0, y_0)|}} A(x_0, y_0) \exp[i k \phi(x_0, y_0) + i \text{sgn} H \pi/4].
\] (E.2.11)

with

\[
H(x_0, y_0) = \det \phi_{xy} = \phi_{xx} \phi_{yy} - \phi_{xy}^2.
\]
is the Hessian for the matrix of second partial derivatives of the form

\[
\begin{align*}
\phi_{xy} &= \frac{\partial^2 \phi}{\partial x \partial y} \\
\phi_{xx} &= \frac{\partial^2 \phi}{\partial x^2} \\
\phi_{yy} &= \frac{\partial^2 \phi}{\partial y^2}
\end{align*}
\]

All of this expressions are evaluated at the point \((x_0, y_0)\) such that

\[\nabla \phi(x, y) = 0.\]

If such a point does not exist, then the formula can be evaluated by integration by parts.

We evaluate the formula in pieces.

- **Find the stationary phase points.** The phase function is given by

\[\phi(x, y) = r_P + r.\]

Now \(P = (x_P, y_P, 0)\) is the source location so

\[r_P = \sqrt{(x - x_P)^2 + (y - y_P)^2 + z_1^2},\]  

(E.2.12)

similarly

\[r = \sqrt{(x - x_Q)^2 + (y - y_Q)^2 + (z_2 - z_1)^2},\]  

(E.2.13)

with \(Q = (x_Q, y_Q, z_2)\).

The partial derivatives of the distances are:

\[
\begin{align*}
\frac{\partial r}{\partial x} &= \frac{x - x_P}{r} \\
\frac{\partial r}{\partial y} &= \frac{y - y_P}{r}, \\
\frac{\partial r_P}{\partial x} &= \frac{x - x_P}{r_P} \\
\frac{\partial r_P}{\partial y} &= \frac{y - y_P}{r_P}.
\end{align*}
\]

so

\[
\begin{align*}
\phi_x &= \frac{\partial \phi}{\partial x} = \frac{x - x_Q}{r} + \frac{x - x_P}{r_P} = \frac{r_P(x - x_Q) + r(x - x_P)}{rr_P} \\
\phi_y &= \frac{\partial \phi}{\partial y} = \frac{y - y_Q}{r} + \frac{y - y_P}{r_P} = \frac{r_P(y - y_Q) + r(y - y_P)}{rr_P}.
\end{align*}
\]
and from
\[ \nabla \phi(x, y) = 0 \]

We find:
\[ r_P(x - x_Q) + r(x - x_P) = r_P(y - y_Q) + r(y - y_P) = 0 \]

That is
\[ \frac{x - x_Q}{x - x_P} = \frac{y - y_Q}{y - y_P} = -\frac{r}{r_P} \tag{E.2.15} \]

Now, from E.2.15 into E.2.12.
\[ r_P = \sqrt{(x - x_Q)^2 \frac{r^2}{r_P^2} + (y - y_Q)^2 \frac{r^2}{r_P^2} + z_1^2} \]

so
\[ r = \sqrt{(x - x_Q)^2 + (y - y_Q)^2 + z_1^2 \frac{r^2}{r_P^2}} \]

and from E.2.13 we find that
\[ z_1^2 \frac{r^2}{r_P^2} = (z_1 - z_2)^2. \]

That is
\[ \frac{r}{r_P} = \pm \frac{z_2 - z_1}{z_1}. \]

We pick the “−” sign (see figure) and
\[ \frac{z_1 - z_2}{z_1} = \frac{r}{r_P}. \]

Then from this equation and equation E.2.15 we find
\[ \frac{y - y_Q}{x - x_Q} = \frac{y - y_P}{x - x_P} = \frac{z_1 - z_2}{z_1} = \frac{r}{r_P}. \]

The direction of the two rays is the same. The only ray pair, with that direction is the red pair drawn in red in figure E.2. That is a straight path between P and Q. This is in fact a restatement of Fermat’s principle, given that the velocity is constant in both medios (before and after the barrier).
The specular ray is the direct ray. Other rays have lower order contributions. If there is an \((x, y)\) in the aperture such that this happens, that is the stationary phase point. Otherwise all rays bend and this is the classical case of diffraction as “going around a corner”. In that case since there is no \((x, y)\) such that \(\nabla \phi(x, y) = 0\), the integral can be evaluated by integration by parts and the leading order will indicate the dominant (asymptotic) wavefield for large wavenumbers.

Let us then assume that there is an \((x, y)\) such that the ray can travel straight from \(P\) to \(Q\) along the aperture and call it \((x_0, y_0)\) (the intersection of the wavefront with the red ray in the picture).

We will not find the explicit solution in algebraic form of this point, since it is not needed for our purpose.

- **The Hessian.** The Hessian is always the most combersome computation for stationary phase evaluation. The Hessian is defined by equation E.2.12 which we rewrite here.

\[
H(x_0, y_0) = \det \Phi = \phi_{xx}\phi_{yy} - \phi_{xy}^2. \tag{E.2.16}
\]

In general, let us now compute the crossing term \(\phi_{xy} = \phi_{yx}\).

From equation E.2.14 we find

\[
\phi_{xy} = \frac{rr_P[(x - x_Q)(r_P)_y + (r)_y(x - x_P)] - [r_P(x - x_Q) + r(x - x_P)][(r)_yr_P + r(r_P)_y]}{(r r_P)^2}
\]

From the definition of \(r\) and \(r_P\) (see formulas E.2.13 and E.2.12) we find \((r)_x\) and \((r_P)_x\).

\[
(r)_y = \frac{y - y_Q}{r} \quad (r_P)_y = \frac{y - y_P}{r_P}
\]

So the numerator for \(\phi_{xy}\) expands to

\[
r r_P[(x - x_Q)(y - y_P)/r_P + (x - x_P)(y - y_Q)/r] - T
\]

\[
= r(x - x_Q)(y - y_P) + r_P(x - x_P)(y - y_Q) - T
\]

\[
= r(x - x_Q)(y - y_P) + r_P(x - x_Q)(y - y_P) - T
\]

\[
= (x - x_Q)(y - y_P)(r + r_P) - T
\]

where

\[
T = [r_P(x - x_Q) + r(x - x_P)] \left[ \frac{(y - y_Q)r_P}{r} + \frac{r(y - y_P)}{r_P} \right]
\]

\[
= \frac{r^2}{r} (x - x_Q)(y - y_Q) + r_P(x - x_P)(y - y_Q) + r(x - x_Q)(y - y_P) + \frac{r^2}{r} (x - x_P)(y - y_P)
\]

\[
= 0
\]
The cancelations are done by using \( x - x_P = -(r_P/r)(x - x_Q) \).

So

\[
\phi_{xy} = \frac{(x - x_Q)(y - y_P)(r + r_P)}{(rr_P)^2}.
\]  

(E.2.17)

Let us find \( \phi_{xx} \). This would be given by

\[
\phi_{xx} = \frac{r r_P[(x - x_Q)(r_P x) + r_P(x - x_P) + r] - [r_P(x - x_Q) + r(x - x_P)][(r_P x_P + r(r_P x)]}{(rr_P)^2}.
\]

As done before, with the help of

\[
(r)_x = \frac{x - x_Q}{r} \quad (r_P)_x = \frac{x - x_P}{r_P}
\]

the numerator is

\[
\begin{align*}
rr_P[(x - x_Q)(x - x_P)/r + (x - x_P)(x - x_Q)/r] &+ r r_P(r_P + r) - T \\
&= r(x - x_Q)(x - x_P) + r_P(x - x_P)(x - x_Q) + r r_P(r_P + r) - T \\
&= (r + r_P)(x - x_Q)(x - x_P) + r r_P(r_P + r) - T
\end{align*}
\]

with

\[
T = [r_P(x - x_Q) + r(x - x_P)] \left[ \frac{(x - x_Q)r_P}{r} + \frac{r(x - x_P)}{r_P} \right]
\]

\[
= \frac{r_P^2}{r}(x - x_Q)^2 + r(x - x_Q)(x - x_P) + r_P(x - x_Q)(x - x_P) + \frac{r^2}{r_P}(x - x_P)^2
\]

\[
= 0.
\]

The cancelations are done after using \( x - x_P = -(r_P/r)(x - x_Q) \). Then

\[
\phi_{xx} = \frac{(r + r_P)(x - x_Q)(x - x_P) + r r_P(r_P + r)}{(rr_P)^2}
\]

(E.2.18)

Similarly

\[
\phi_{yy} = \frac{(r + r_P)[(y - y_Q)(y - y_P) + r r_P]}{(rr_P)^2}
\]

(E.2.19)
From equations E.2.18 and E.2.19

\[
\phi_{xx}\phi_{yy} = \left(\frac{(r + r_P)(x - x_Q)(x - x_P) + r^2 r_P}{(r r_P)^2}\right) \left(\frac{(r + r_P)(y - y_Q)(y - y_P) + r^2 r_P}{(r r_P)^2}\right)
\]

\[
= \frac{(r + r_P)^2(x - x_Q)(x - x_P)(y - y_Q)(y - y_P) + (r + r_P)^2 r r_P(x - x_Q)(x - x_P)}{(r r_P)^4}
\]

\[
+ \frac{(r + r_P)^2 r r_P(y - y_Q)(y - y_P) + (r + r_P)^2}{(r r_P)^4}
\]

\[
= \frac{(r + r_P)^2(x - x_Q)^2(y - y_P)^2}{(r r_P)^4} - \frac{(r + r_P)^2 r^2 r_P(x - x_Q)^2}{(r r_P)^4}
\]

\[
- \frac{(r + r_P)^2 r^2 r_P(y - y_Q)^2}{(r r_P)^4} + \frac{(r + r_P)^2}{(r r_P)^2}.
\]

(E.2.20)

and from equation E.2.17

\[
\phi_{xy}^2 = \frac{(x - x_Q)^2(y - y_P)^2(r + r_P)^2}{(r r_P)^4}.
\]

(E.2.22)

Then this equation and E.2.21

\[
H(x_0, y_0) = \frac{(r + r_P)^2}{(r r_P)^2} - \frac{(r + r_P)^2 r^2 r_P(x - x_Q)^2}{(r r_P)^4} - \frac{(r + r_P)^2 r^2 r_P(y - y_Q)^2}{(r r_P)^4}
\]

\[
= \frac{(r + r_P)^2}{(r r_P)^2} \left(1 - \frac{r^2 r_P}{(r r_P)^2} [x - x_Q]^2 + (y - y_Q)^2]\right).
\]

we can not ignore this :(.

We only can predict the appropriate Green’s function if \(x = x_Q\) and \(y = y_Q\) which corresponds to figure E.3

\[
\sqrt{|H(x_0, y_0)|} = \sqrt{\phi_{xx}\phi_{yy} - \phi_{xy}^2} = \frac{r + r_P}{r r_P}
\]

with signature of the Hessian is 2.

So, since the stationary point is such that the rays connecting \(P\) with \(Q\) across the aperture are colinear, then

\[
\xi = \chi = 0.
\]
Figure E.3: This picture corresponds to the experiment that predicts the correct interpretation of the Huygens principle

\[ P = (x_P, y_P, z_P) \]
\[ Q = (x_Q, y_Q, z_2) \]

So, the amplitude factor is

\[ A(x_0, y_0) = -\frac{i}{\lambda} \frac{1}{rr_P} \frac{1}{2} (1 + 1) = -\frac{i}{\lambda} \frac{1}{rr_P} \]

- We apply the stationary phase formula E.2.11. We find, since \( \lambda = 2\pi/k \),

\[
I(k) \approx -\frac{2\pi}{k} \frac{r + r_P}{(r + r_p) X_{EPF}} \frac{i}{\lambda} \exp(ik(r + r_P))] \exp(i\pi/2)
\]
\[
= -\frac{ii}{r + r_P} \exp(ik(r + r_P)]
\]
\[
= \frac{1}{r + r_P} \exp(ik(r + r_P)]
\]

Which predicts the Green’s function obtained as if there is no barrier, and the wave would travel directly from the source at \( P \) to the receiver at \( Q \).
Appendix F

Comparison of Born, Rytov, and De Wolf Approximations

F.1 Introduction

We offer a few derivations to compute relationships between wavefields and scatterers that are useful for modeling and inversion in fields such as medical imaging, non-destructive testing, and seismic modeling and inversion. These derivations are in no way new. We include a few references but the list is far from being fair.

F.2 General Theory

In general, in the context of perturbation theory, a linear problem is written which operates in two different objects. A background object and a perturbed object. The background object is usually defined with constant velocity (or known velocity, which could be non-constant) \(^1\) and the perturbed media with a perturbed velocity. The difference of the two linear operators is a new linear operator with the same shape but where the source is now some function of the perturbation. The solution in terms of Green’s functions creates a non-linear integral equation (Fredholm type in Born and Riccati type in Rytov, for example). The integral equation could be solved iteratively by starting with a linear assumption and plugging back the result into a new background field.

The standard way to solve inverse scattering problems is by formulating a linear operator

\[
Lu = f
\]  

where \(f\) is a source function and \(u\) is a wavefield. The operator \(L\) is written as the sum of a background operator \(L_0\) and a perturbed operator \(L_1\). The solution \(u\) is

\(^1\)In the seismic experiment velocity is the most common parameter used
also written as the sum of two functions (or product of two functions in the case of
the Rytov approximation, for example). In the context of the Born approximation we
write

$$L = L_0 + L_1 \quad u = u_0 + u_s$$ \hspace{1cm} (F.2.2)

We assume that we know how to solve for the background operator $L_0$. That is, for
the case of the background field $u_0$ and the background operator $L_0$ we know

$$u_0 = L_0^{-1} f.$$ \hspace{1cm} (F.2.3)

We substitute F.2.3 and F.2.2 into F.2.1 and find

$$(L_0 + L_1)(u_0 + u_s) = f \Rightarrow L_0 u_s + L_1 u_0 + L_1 u_s = 0,$$ \hspace{1cm} (F.2.4)

from which

$$u_s = -L_0^{-1} L_1 u_0 - L_0^{-1} L_1 u_s = -L_0^{-1} (L_1 u_0 + L_1 u_s) = -L_0^{-1} L_1 (u_0 + u_s).$$ \hspace{1cm} (F.2.5)

This equation provides the scattering field $u_s$ as a function of the incident field $u_0$
and the scattering field itself $u_s$. Moreover, the inversion is that operator $L_0$ which
we know how to invert for. The Born approximation is the suppression of the second
term on the right, assuming that $u_s \ll u_0$.

We now translate the operator $L$ and $L_0$ into wave equation operators. That is

$$L = \left( \nabla^2 + \frac{\omega^2}{c^2(r)} \right) \quad L_0 = \left( \nabla^2 + k^2 \right)$$

with $k = \omega^2/c_0^2$, from which

$$L_1 = L - L_0 = \omega^2 \left( \frac{1}{c^2(r)} - \frac{1}{c_0^2} \right) = k^2 \left( \frac{c_0^2}{c^2(r)} - 1 \right) = k^2 \epsilon(r)$$ \hspace{1cm} (F.2.6)

Now the operator $L_0^{-1}$ is given by the convolution of the Green’s function $G(r', r)$
such that

$$L_0 G(r, r') = -\delta(r - r').$$

and the source (whatever is on the right side). So replacing the operators accordingly
in F.2.5 we find

$$u_s = -L_0^{-1} L_1 (u_0 + u_s) = k^2 \int_V dr' G(r, r') \epsilon(r')(u_0 + u_s).$$ \hspace{1cm} (F.2.7)

and after dropping the scattering field from the right hand side,

$$u_s = -L_0^{-1} L_1 (u_0 + u_s) = k^2 \int_V dr' G(r, r') \epsilon(r') u_0$$ \hspace{1cm} (F.2.8)

Equation F.2.8 is known as the Born approximation, while equation F.2.7 is know
as the Lippmann-Schwinger integral equation and it is a Fredholm type two integral
equation. This equation, in the context of quantum mechanics is known as the Lippmann-
Schwinger \(^2\) equation. See also Weglein et. al, 1997. \(^3\) show applications of the

\(^2\)http://en.wikipedia.org/wiki/Lippmann%E2%80%93Schwinger_equation
\(^3\)http://www.slb.com/ /media/Files/technical_papers/140/1444298.pdf
Neumann series for attenuation of multiples in seismic reflection data. We prefer the Fredholm name which has a much broader scope (just mathematics, far beyond quantum mechanics).

Note that the scattering field $u_s$ is both in the left side and inside the integral. In this way the equation is non-linear. One way to see this problem is by writing as an eigenvalue/eigenvector problem. Let us see: By defining the operator

$$T = -L_0^{-1}L_1$$

and a new source term

$$g = Tu_0,$$  \hspace{1cm} (F.2.9)

we might write the (in operator form) equation F.2.7 as

$$u_s = g + Tu_s$$  \hspace{1cm} (F.2.10)

For $g = 0$ we see that 1 is an eigenvalue of the operator $T$ and $u_s$ is an eigenvector. Reordering terms in F.2.10 we find

$$(1 - T)u_s = g.$$  \hspace{1cm} (F.2.11)

If $(1 - T)^{-1}$ exists (and this is conditioned to the norm of the operator $\|T\| < 1$). In this case we can expand

$$(1 - T)^{-1} = \frac{1}{1 - T} = 1 + T + T^2 + \cdots$$

With the help of this series we can write

$$u_s = (1 + T + T^2 + \cdots)g.$$  \hspace{1cm} (F.2.12)

This expression is sometimes known as the multiple scattering series the Neumann series and the operator $(1 - T)^{-1}$ as the resolvent operator. This is not the first time we discuss this ideas in this book. We discussed this in section 7.5.2 but it is important to write this again to make emphasis on its importance.

F.3 The Rytov Approximation

The strategy of the Rytov approximation [?] is a bit different form the one in Born approximations. Here Rytov writes the wave field as product of an “amplitude” term with a complex exponential “phase”. Where the phase is the actual perturbation and the amplitude factor is the background field.
From equation F.2.4 we observe that

\[ L_0 u_s = -L_1 u; \]

and from using equation F.2.6

\[ L_0 u_s = -L_1 u = -k^2 \epsilon(r) u(r). \]

Hence,

\[ (\nabla^2 + k^2) u(r) = -k^2 \epsilon(r) u(r). \] (F.3.13)

where

\[ \epsilon(r) = \frac{c_0^2}{c_0^2(r) - 1} \]

is the model perturbation (slowness square perturbation. Which is obtained by subtracting the exact from the background wave equations).

The first Rytov expression is to write the field \( u(r) \) as an amplitude phase complex number that is

\[ u(r) = u_0(r)e^{\psi(r)}, \] (F.3.14)

where \( u_0(r) \) is the background wavefield and all other contributions to the number are stacked into the phase \( \psi(r) \). Then from the sub–appendix equation F.A.27

\[ \nabla^2 u(r) = e^{\psi(r)} \left[ u_0(r) \left( \nabla^2 \psi(r) + (\nabla \psi(r))^2 \right) + 2 \nabla u_0(r) \cdot \nabla \psi(r) + \nabla^2 u_0(r) \right]. \] (F.3.15)

From equation F.3.13

\[ (\nabla^2 + k^2) u_0(r)e^{\psi(r)} = -k^2 \epsilon(r) u_0(r)e^{\psi(r)} \Rightarrow \]

\[ \nabla^2 u(r) = -k^2 e^{\psi(r)} u_0(r)(\epsilon(r) + 1), \] (F.3.16)

and matching right hand sides of the equations F.3.15 and F.3.16, and cancelling the common factor \( e^{\psi(r)} \), we find

\[ u_0(r) \left( \nabla^2 \psi(r) + (\nabla \psi(r))^2 \right) + 2 \nabla u_0(r) \cdot \nabla \psi(r) + \nabla^2 u_0(r) = -k^2 u_0(r)(\epsilon(r) + 1). \]

Now, because the amplitude term \( u_0(r) \) is the background wave field it satisfies the homogeneous equation (this is the second version of the wave equation)

\[ (\nabla^2 + k^2) u_0(r) = 0. \] (F.3.17)
we get to
\[ u_0(r) \left( \nabla^2 \psi(r) + (\nabla \psi(r))^2 \right) + 2 \nabla u_0(r) \cdot \nabla \psi(r) = -k^2 u_0(r) \epsilon(r) \]

Moving terms around
\[ 2 \nabla u_0(r) \cdot \nabla \psi(r) + u_0(r) \nabla^2 \psi(r) = -u_0(r) \left[ (\nabla \psi(r))^2 + k^2 \epsilon(r) \right] \]

The first two terms on the left ask for a completion of squares. That is we add the missing \( \psi(r) \nabla^2 u_0(r) \) and see
\[ \nabla^2 (u_0(r) \psi(r)) = -u_0(r) \left[ (\nabla \psi(r))^2 + k^2 \epsilon(r) \right] + \psi(r) \nabla^2 u_0(r) \]  \hspace{1cm} (F.3.18)

From F.3.17
\[ \psi(r) \nabla^2 u_0(r) = -k^2 u_0(r) \psi(r), \]

so equation F.3.18 can be written as
\[ \left( \nabla^2 + k \right) u_0(r) \psi(r) = -u_0(r) \left[ (\nabla \psi(r))^2 + k^2 \epsilon(r) \right] \]

This is the wave equation for the combined field
\[ u_0(r) \psi(r) \]

and a source term
\[ -u_0(r) \left[ (\nabla \psi(r))^2 + k^2 \epsilon(r) \right] \]

for which the solution in terms of Green’s functions is
\[ u_0(r) \psi(r) = \int_V d^3r G(r; r') u_0(r') \left[ (\nabla \psi(r'))^2 + k^2 \epsilon(r') \right] \]  \hspace{1cm} (F.3.19)

with \( G(r; r') \) the Green’s function of the background medium, \( u_0(r') \) and \( u_0(r) \) are the incident fields at \( r' \) and \( r \) respectively.

At this point there are not yet approximations. The integral equation F.3.19 is a Riccati integral equation. If the square gradient function \( (\nabla \psi(r'))^2 \) is small compared to the perturbation \( k^2 \epsilon(r') \) then it can be dropped from the integral. This is the:

\section*{F.3.1 The Rytov Approximation}

The Rytov approximation is the truncation of integral F.3.19 by dropping the gradient square function of the phase. This is
\[ \psi(r) = \frac{1}{u_0(r)} \int_V d^3r G(r; r') u_0(r') k^2 \epsilon(r') \]  \hspace{1cm} (F.3.20)
assuming $\|\nabla \psi\|^2 \ll k^2\epsilon(r')$.

What is the meaning of $\|\nabla \psi\|^2 \ll k^2\epsilon(r')$? From equation F.3.14, we see that $\phi(r)$ measures the perturbation in phase. The gradient $\nabla \psi(r)$ measures the change on this perturbation. For example, a sudden change on velocity would give a bending of the ray trajectory so a phase change is correlated with changes in the direction of propagation. In this way the Rytov approximation constrains the change on angle propagation from the background to the scatterer volume. It is in this sense that the Rytov approximation is better for transmission that for reflection, since in reflection a large change on angle (and phase) is assumed. Wu et. al, 2007 make the following quantitative analysis. Assume that the observed total field after interacting with the heterogeneities is nearly a plane wave:

$$u = Ae^{i k_1 \cdot r},$$

The incident wave is

$$u_0 = A_0 e^{i k_0 \cdot r}.$$  

Write

$$u = u_0 e^{\psi(r)},$$

and taking natural logarithm ln to the $u/u_0$ ratio,

The phase $\psi$ can be written as

$$\psi = \ln \left( \frac{A}{A_0} \right) + i (k_1 - k_0) \cdot r,$$

and

$$\nabla \psi = \nabla \ln \left( \frac{A}{A_0} \right) + i (k_1 - k_0)$$

so

$$\nabla \psi \cdot \nabla \psi = \left| \nabla \ln \left( \frac{A}{A_0} \right) \right|^2 - |k_1 - k_0|^2 + 2i(k_1 - k_0) \cdot \nabla \ln \left( \frac{A}{A_0} \right).$$

Normally wave amplitudes vary much slower than phases, so the major contribution to $\nabla \psi \cdot \nabla \psi$ is given by the term $|k_1 - k_0|^2$. So the Rytov approximation $\|\nabla \psi\|^2 \ll k^2\epsilon(r')$ implies

$$|k_1 - k_0|^2 = |k_1|^2 + |k_0|^2 - 2k_1 \cdot k_0$$

$$= 2k^2 - 2k^2 \cos 2\theta$$

$$= 2k^2 - 2k^2(\cos^2 \theta - \sin^2 \theta)$$

$$= 2k^2(1 - \cos^2 \theta) + 2k^2 \sin^2 \theta$$

$$= 4k^2 \sin^2 \theta \ll k^2 \sqrt{|\epsilon|}$$
F.4 Comparison between Rytov and Born

where $k = \omega/v$, $v$ is the wavespeed at point $r$, and $\theta$ is the aperture angle between $k_1$ and $k_2$. Then

$$2\sin \theta \ll \sqrt{|\epsilon|}$$

or

$$\sin \theta \ll \frac{1}{2} \sqrt{\left| c_0^2 - c^2(r) \right| c^2(r)}.$$

If we require a small contrast between the background and the heterogeneities, then $\sin \theta \ll 1$. That is, the Rytov approximation is a small angle approximation. If the energy is backscattered, then $2\theta = \pi$ and so $\sin \theta = 1$ which violates the small angle assumption.

### F.4 Comparison between Rytov and Born

- In the Born approximation the wavefield is additive. That is,

$$u(r) = u_0(r) + u(r)$$

where $u(r)$ is the propagating wavefield, $u_0(r)$ is the background wavefield and $u(r)$ is the perturbed wavefield.

- In the Rytov approximation the wavefield is multiplicative. That is,

$$u(r) = u_0(r)e^{\psi(r)}$$

where $u(r)$ is the propagating wavefield, $u_0(r)$ is the background wavefield and $\psi(r)$ is a complex phase function that make up for the rest of the wavefield. That is this could be computed using logarithms as

$$\psi(r) = \ln u(r) - \ln u_0(r).$$

A reason for the exponential function is because it fits well the case where $\psi(r) = 0$, returning $u_0(r)$ as the solution of the background problem. Any other exponential would do, but anything more natural than the Euler constant?

The Rytov assumption about the wavefield as a multiplication of an amplitude factor times a phase factor (complex exponential) reflects in the behavior of the WKBJ Green’s functions and emphasize in the approximation of the phase rather than in the whole Green’s function.
• By expanding $e^\psi$ into a power series we see

$$u - u_0 = u_0(e^\psi - 1) = u_0\psi + \frac{1}{2}u_0\psi + \cdots$$

When $\psi \ll 1$, that is, when the accumulated phase change is than one radian (corresponding to about one sixth of the wave period), we can neglect high order terms in after $\psi^2$, so

$$u(r) = u_0(r) + u_0(r)\psi(r) = u_0(r) + k^2 \int_V d^3r G(r, r')\epsilon(r')u_0(r'),$$

which is the Born approximation F.2.8. That is, for $\phi \ll 1$, the Rytov approximation reduces to the Born approximation. In this sense, it is customary to think that the Rytov approximation has a broader scope than the Born approximation, but this is no necessarily the case for Brown, although Oristaglio thinks his numerical evidence shows that the Rytov approximation is better and that the Born approximation is just the first order term of an infinite series of the Rytov approximation. Piperakis and friends also agree about the better approximation of the Rytov approximation over the Born approximation, for computing travel times.

• By the nature of the Born approximation and that of the Rytov approximation, as indicated in section F.3.1 we have that

  – In the Born approximation we might expect better behavior of reflected (backscattered) energy. Because the scattered wavefield with respect to the incident wavefield should be small, the propagation distance is either short (in which the incident wavefield is large) and/or the heterogeneities are weak. By observing integral F.2.8 we see that the integral adds up the incident energy $u_0(r')$ constructively along the transmitted energy without bounds (because the $u_0(r')$ is not recomputed. This makes of the transmission problem for the Born approximation a poor choice.

  – As discussed in section F.3.1, the Rytov approximation has poor accuracy for backscattered energy.

F.5 The de Wolf Approximation

Due to the limitations of Born for forward scattering and Rytov for backward scattering some new ideas have came through. A basic idea is to include more terms in the Neumann series. However de Wolf presented the article Electromagnetic reflection from an extended turbulent medium: Cumulative forward-scatter single-backscatter approximation in 1971 That is, by choosing the scattering direction so that there could any number of forward scattered energy but only one backscattered at most. This is called Multiple Forward–scattering Single Backscattering (MFSB).
F.5 The de Wolf Approximation

From equations F.2.11 and F.2.9 we have that

\[ u_s = (1 - T)^{-1} g \]

where \( g \) is the incident field that we call \( u_f \).

Also, \( T = -L_0^{-1} L_1 \), \( L_1 = k^2 \epsilon \). To simplify notation we can call \( L_0^{-1} = -G_0 \) as the Green’s function for the background problem, so

\[ u_s = (1 - k^2 G_0 \epsilon)^{-1} u_f, \]

(F.5.21)

We can write

\[ \epsilon = \epsilon_f + \epsilon_b \]  

(F.5.22)

where \( \epsilon_f \) is the forward perturbation and \( \epsilon_b \) is the backward perturbation. We can think of a model with many interfaces, and at each interface the acoustic wave will split into two new waves. The transmitted wave with perturbation value \( \epsilon_f \), and a backscattered wave with a perturbation value \( \epsilon_b \).

The idea behind the de Wolf’s approximation is to choose, from all the Neumann series terms, only those that correspond to all forward coefficients and just one backward. That is, pick only primaries. We expand the operator into its geometrical (Neumann) series,

\[ u_s = u_f + k^2 G_0 \epsilon u_f + k^2 G_0 k^2 \epsilon G_0 \epsilon u_f + \cdots (k^2 G_0 \epsilon)^n u_f + \cdots \]  

(F.5.23)

At this point we replace \( \epsilon \) from equation F.5.22 into F.5.23. The \( n \)-th term of equation F.5.23 is (we omit \( u_f \) since this is the same at the end of each term)

\[ [k^2 G_0 (\epsilon_f + \epsilon_b)]^n \]

since we can not assume commutations, the expansion of the \( n \)-th term has \( 2^n \) products. The idea is that from all those products we want to pick only those that have one factor \( \epsilon_b \) while the other factors have \( \epsilon_f \). That is, a factor such as

\[ (k^2 G_0 \epsilon_f) \cdots (k^2 G_0 \epsilon_f) (k^2 G_0 \epsilon_b) (k^2 G_0 \epsilon_f) \cdots (k^2 G_0 \epsilon_f), \]

only backscatter

or, for each \( i = 0, 1, \ldots, n - 1 \),

\[ (k^2 G_0 \epsilon_f)^i (k^2 G_0 \epsilon_b) (k^2 G_0 \epsilon_f)^{n-i-1} \]

Wu et. al, 2007⁴ provide practical implementations of de Wolf’s ideas into seismic.

F.6 In Between Born and Rylov

Daniel L. Marks considers an interesting idea that creates an infinite intermediate approximations between the Born and the Rylov. His idea is simple. The Rylov approximation is based on the assumption $u = u_0 e^{\psi}$. Now from Euler’s formula

$$\lim_{n \to \infty} \left(1 + \frac{x}{n}\right)^n = e^x$$

he writes

$$u = u_0 \lim_{n \to \infty} \left(1 + \frac{\psi}{n}\right)^n$$

When $n = 0$ we have that the perturbation is zero $u = u_0$, when $n = 1$ we find

$$u = u_0 + u_0 \psi$$

where we identify $u_s = u_0 \psi$, and when $n \to \infty$ we find the Rylov approximation. So with this formula, we can think of an infinite number of approximations in between Born and Rylov, some of which could better suited for a particular problem.

F.A The Laplacian of a Product of Functions

We justify the Laplacian of the product used in equation F.3.15. We use the Laplacian of the product of two functions:

$$\nabla(fg) = \nabla \cdot (\nabla f g + g \nabla f) = f \nabla^2 g + 2 \nabla f \cdot \nabla g + g \nabla^2 f$$

(F.A.24)

Now, let us assume that $u = u_0(r) e^{\psi(r)}$, so

$$\nabla^2 u(r) = \nabla^2 (u_0(r) e^{\psi(r)})$$

and

$$f = u_0(r) \quad g = e^{\psi(r)}$$

Then by applying equation F.A.24

$$\nabla^2 (u_0(r) e^{\psi(r)}) = u_0(r) \nabla^2 e^{\psi(r)} + 2 \nabla u_0(r) \cdot \nabla e^{\psi(r)} + e^{\psi(r)} \nabla^2 u_0(r)$$

(F.A.25)

We are no done. Let us find

$$\nabla^2 e^{\psi(r)} = \nabla \cdot \nabla e^{\psi(r)} = \nabla \cdot (e^{\psi(r)} \nabla \psi(r)) = e^{\psi(r)} \nabla^2 \psi(r) + e^{\psi(r)} (\nabla \psi(r))^2$$

(F.A.26)

with $(\nabla f)^2 = (\nabla f) \cdot (\nabla f)$. We apply F.A.26 into F.A.25 and find

$$\nabla^2 (u_0(r) e^{\psi(r)}) = e^{\psi(r)} \left[ u_0(r) (\nabla^2 \psi(r) + (\nabla \psi(r))^2) + 2 \nabla u_0(r) \cdot \nabla \psi(r) + \nabla^2 u_0(r) \right].$$

(F.A.27)
Appendix G

The Fréchet and Gâteaux Derivatives

First, I introduce the concepts of Fréchet derivatives and Gâteaux derivatives starting with the traditional calculus definitions on real variables, then generalize the concept to Banach spaces where the variables can go beyond numbers to functionals.

The concept of derivative (we use the word gradient because our objective functions go from Banach spaces into the real numbers and so, as we will see, the derivatives are actually gradients) is at the heart of linear perturbation theory.

The idea is that we have a function $f$ of some parameters $p$ and want to find out $p$ from perturbing the function and observations of that perturbation away from an initial guessed solution $p_0$. To the first order

$$f(p + \delta p) = f(p) + f'(p) \delta p.$$  

that is, the perturbation of the function is, to linear order, that of the parameters times the derivative, or in other words:

$$\delta f(p) = f'(p) \delta p.$$  

G.0.1 Review of derivatives in calculus

In real spaces if a function $f$ is defined on $A$ (an open set in $R^n$),

$$f : A \subset R^n \rightarrow R^m$$

$$x \rightarrow f(x).$$

In calculus we say that if

$$\lim_{h \rightarrow 0} \frac{1}{\|h\|} [f(x + h) - f(x)]$$  

for all $h$ near zero exists, then the function is differentiable at the point $x$ and its derivative is given by

$$f'(x) = \lim_{h \rightarrow 0} \frac{1}{\|h\|} [f(x + h) - f(x)]$$
This definition is not very useful. But it can be shown that if this derivative exists, then by writing
\[ h = \sum_{j} h_j e_j \]
and using the notation
\[ f_{i,j} = \frac{\partial f_i}{\partial x_j} = \lim_{h_j \to 0} \frac{1}{h_j} [f_i(x + h_j e_j) - f_i(x)], \]
i = 1, \ldots, m, j = 1, \ldots, n.

Note that we could define the mapping.
\[ f' : A \to \mathbb{R}^{m \times n} \]
\[ x \mapsto f_{i,j}(x) = J(x) \]
where the notation \( J(x) \) stands for Jacobian matrix.

However this mapping is not very interesting in practical applications (the Jacobian is, but not the mapping). Usually when solving inverse problems we want to stay in a fixed point \( x \), and based on the derivative move along a direction. For this the differential is a more interesting concept. The definition of the differential is given by
\[ Df(x) : \mathbb{R}^n \to \mathbb{R}^m \]
\[ v \mapsto Df(x)(v) = J(x)v \]
Note that the symbol \( Df(x) \) does not show the domain of dependence (the space of \( v \)). To see that this operator is well defined we observe that \( J(x) \in \mathbb{R}^{m \times n} \) and \( v \in \mathbb{R}^n \), so the product is a well defined operation of a matrix times a vector.

The differential has some good attributes. A few of them are:

1. It is a linear operator. That is
\[ Df(x)(v + w) = Df(x)(v) + Df(x)(w) \] and
\[ Df(x)(\alpha v) = \alpha Df(x)(v). \]
This can be shown from the definition.

2. It lets us compute some variation of a function along any direction, as we will see below.

3. In integration theory, as well as in sensitivity analysis, \( v \) is understood as \( dx \) and so we can write
\[ Df(x)dx = J(x)dx \]
which works well to estimate small displacements and differential forms for integration.
4. The differential of a linear function is itself. The proof of this is simple. Assume

\[ f(x) = Ax \]

(since we are in real spaces \( A \) is a matrix and \( x \) is a vector.) We know that

\[ f_{i,j}(x) = \frac{\partial f_i}{\partial x_j} = a_{ij} \]

Now by definition

\[ Df(x) : \mathbb{R}^n \rightarrow \mathbb{R}^m \]

\[ v \mapsto Df(x)(v) = Av, \]

so \( Df(x) = f(x) \), for \( f \) linear.

5. In 1D, the differential represents the tangent line at the point \((x, f(x))\) of the curve. That is, if we write for a one dimensional space

\[ Df(x) : \mathbb{R} \rightarrow \mathbb{R} \]

\[ dx \mapsto f'(x)dx \]

It is like saying

\[ f(x + h) = f(x) + f'(x)h \]

where the translation from the origin to the point along the curve, is explicitly achieved.

One motivation behind the focus on differentials as going through the \((0, 0)\) point is because in general (Hilbert or Banach) spaces linear operators go through the point \((0, 0)\).

If \( n = 2 \) the tangent plane (normal to the gradient vector at the point) and in general it is a tangent hyperspace. So it is the simplest object that approximates (of course linearly) a curve (surface or hypersurface) close to its point of contact. Linearity is important, not only because the integration theory plays with things like \( f(x)dx \) and not with things like \( f(x)d^2x \) or other monstrosities, but because non–linear problems often are seen as a sequence of linear problems. Also the theory for linear problems is the simplest and most well studied and known of all other (non–linear) theories.

The name of the operator \( Df(x) \) is the Fréchet derivative, but this name will be more useful when we are not working on real spaces but in spaces of functions where we will extend the concept of derivative with respect to objects different from coordinate points, such as wavefields, vector fields or scalar fields for example.
As a simple example, if \( m = 1 \) then \( Df(x) \) is given by
\[
Df(x)dx = J(x)dx = \sum_i \frac{\partial f}{\partial x_i} dx_i = \nabla f \cdot dx
\]
which is the well known differential from calculus. So in general for real spaces of the type \( \mathbb{R}^{m \times n} \) the Fréchet derivative is the Jacobian matrix \( J(x) \) and if \( m = 1 \) it is simply the gradient \( \nabla f(x) \). The differential at a point \( v \) happens to be the product (inner) of the Fréchet derivative with that point \( v \) seen as a linear operator acting on \( v \). This form of a differential is actually also known as the Gâteaux derivative.

Before we define derivative in more general spaces, let us introduce the concept of directional derivative. This is given by
\[
D_v f(x) = \lim_{h \to 0} \frac{1}{h} [f(x + hv) - f(x)] \tag{G.0.2}
\]
where \( h \) is a scalar. This is the derivative of \( f \) along the direction \( v \). Observe that the differential and the Gâteaux derivative coincide.

If \( f \) is differentiable, then the directional derivative exists along any direction. The contrary is not true (search for examples in calculus books or the Internet). The directional derivative generalize the concept of partial derivatives. A \( i \)-th partial derivative is a directional derivative when \( u = e_i \).

The directional derivative is known as the Gâteaux derivative, but again this is more important when we move to more general functions which do not depend directly on real variables (such as functionals for example, which are functions from a function space into real variables).

**G.0.2 What is differentiation on general operators?**

Now, we have a function
\[
f : A \to B \\
x \mapsto f(x)
\]
Here \( A \) and \( B \) are more general spaces than just \( \mathbb{R}^n \) or \( \mathbb{R}^m \). What is the meaning of the derivative of \( f \)? what is the meaning of the differential of \( f \)? I will not use bold symbols because this is general and if I have to distinguish between scalars and vectors of the space, the context will tell.

We borrow all the elements above to re-formulate the problem in a more general context, but before we do that we have to put some restriction on the spaces \( A \) and \( B \). They cannot be, for example, oranges. How close is an orange from another? That
is, they cannot be any arbitrary space. We have to be able to measure distances. That is a norm has to be defined on them. The norm does not necessarily have to be the same $A$ that in $B$, but it should satisfy (of course) the axioms of the norm in functional analysis (positive or zero if vector is zero, scaling, triangular inequality). The context for the spaces $A$ and $B$ is known as Banach spaces. Not only they are normed vector spaces, but they are complete. This means that every Cauchy sequence\(^1\) converge on them.

I start with the Wikipedia\(^2\) definition of Fréchet derivative and show that that definition reduces to the calculus definition G.0.1.

With this the Fréchet derivative is defined as a linear $L$ operator on the same spaces $A$ and $B$\(^3\) such that

$$
\lim_{h \to 0} \frac{\|f(x + h) - f(x) - L(h)\|}{\|h\|} = 0.
$$

If such a linear operator $L$ exists then it is the Fréchet derivative and it its noted as $L(h) = Df(x)(h)$.

Another way to say this is that if we define a residual

$$
R(x, h) = f(x + h) - f(x) - L(h),
$$

such that an $L$ is some linear operator with

$$
\lim_{\|h\| \to 0} \frac{R(x, h)}{\|h\|} = 0
$$

then that $L$ is the differential explicitly written as

$$
L(h) = f(x + h) - f(x) + R(x, h).
$$

This is why in a sloppy way, many authors look at the Fréchet derivative simply as

$$
L = \delta f = f(x + \delta x) - f(x).
$$

It is, however, correct to say that

$$
f(x + h) = f(x) + L(x) + O(\|h\|^2).
$$

So it should be obvious, from here, that

$$
\lim_{\|h\| \to 0} \frac{f(x + h) - f(x)}{\|h\|} = L(x) + \lim_{\|h\| \to 0} O(\|h\|) = 0 = L(x).
$$

\(^1\)A Cauchy sequence is one were the elements get closer together as they get farther from the first element. We will not use this requirement here but it is something needed for the Banach spaces to work “fine”.

\(^2\)http://en.wikipedia.org/wiki/Fr échet_derivative

\(^3\)In practice $B = \mathbb{R}$. If $A$ is the space of functions then any function from $A$ to $B = \mathbb{R}$ is a functional.
So the Fréchet derivative reduces to the same definition G.0.1 used for real spaces. That is, the Fréchet derivative is that linear operator $L$ such that the limit

$$L(x) = \lim_{\|h\| \to 0} \frac{f(x + h) - f(x)}{\|h\|}$$

exists. This definition is very convenient because the algebraic rules (chain rule, rules for differentiation of sums, products, quotients, etc. are based on this definition and they will inherit the same behavior).

The operator is linear by definition and naming it as $L(x) = f'(x)$ we find the usual expression

$$f(x + h) = f(x) + f'(x)(h) + O(\|h\|^2) \quad (G.0.3)$$

The Taylor series theory could be developed in this way defining a second differential, however we will not proceed any further on this direction.

Let us assume that $f$ is linear. That is

$$f(x) = Ax$$

for $A$ independent of $x$. Then

$$f(x + h) - f(x) = A(x + h) - Ax = Ah$$

so if $L(h) = Ah$ then

$$\lim_{h \to 0} \frac{\|f(x + h) - f(x) - L(h)\|}{\|h\|} = 0,$$

so $L = f = A$. That is, if $f$ is linear then the Fréchet derivative is the same linear operator, as shown above where the spaces $A$ and $B$ are real spaces. If in particular $B$ is a subset of $\mathbb{R}$ then $Df(x)$ is the gradient $\nabla_x f$ and the Fréchet derivative operator is defined as

$$Df(x) : A \to B$$

$$v \mapsto Df(x)(v) = \langle \nabla_x f, v \rangle$$

which is the differential as shown above.

It should be clear now that even in spaces which are no real spaces (spaces of functions) we can extend the concept of differentiation, and we can talk about perturbation theory. In exactly the same way in which we perturb functions of a real variable (or complex) we can perturb operators of objects (functions for example) using the same theory with the same algebra, but being careful of identifying the context. That
is, the space that we use to make perturbations. I will be more specific about this in the examples below.

Now, the Gâteaux derivative is defined by the equation G.0.2

\[ D_u f(x) = \lim_{h \to 0} \frac{1}{h} \left[ f(x + hu) - f(x) \right] \]

which it is an equivalent version of the Gâteaux derivative.

While using the definition could be convenient sometimes, I will stay way from using the definition in the case outlined below. It can be shown (since the definition is the same as in regular calculus, the chain rule, the derivative of products, quotients, sums scalings, they all behave the same) that the algebra used for regular partial derivatives will operate the same here.

\[ D_u f(x) = \lim_{h \to 0} \frac{1}{2h} \left[ f(x + hu) - f(x - hu) \right] \]

4In the numerical implementation I used the central difference representation
Appendix H

Adjoint of partial differential operators

Given a linear operator \( L \) in a Hilbert space, its adjoint is that operator \( L^* \) such that for each pair of elements \( u \) and \( v \) in the Hilbert space

\[
\langle Lu, v \rangle = \langle u, L^*v \rangle.
\]

where the inner product \( \langle \cdot, \cdot \rangle \) binary operator is defined on this context by equation H.0.6. I will not discuss any issues of existence and uniqueness here.

Multi-dimensional differential operators come in many flavors. For example, while in one dimension we only have a first derivative operator, in several dimensions we find for scalar fields gradient (\( \nabla \)) and for vector fields divergence (\( \nabla \cdot \)) and curl (\( \nabla \times \)). All of them use first order partial derivatives. In the case of second order I only explore the Laplacian operator (\( \nabla^2 \)). The trick to find adjoints is always integration by parts. Integration by parts comes from the derivative of the product of two functions. In symbols, if

\[
d(uv) = u \, dv + v \, du,
\]

then

\[
uv = \int (u \, dv + v \, du)
\]

or

\[
\int u \, dv = uv - \int v \, du.
\]

I will use bold capital letters for vector fields and non-bold non-capital letters for scalar fields. I assume vector fields from \( \mathbb{R}^n \) into \( \mathbb{R}^m \) (\( m \) could be equal to \( n \)) and scalar fields from \( \mathbb{R}^n \) into \( \mathbb{R} \).
Therefore the following set of equations will be of great use in the following sections:

\[
\nabla \cdot (f \mathbf{G}) = G \cdot \nabla f + f \nabla \cdot \mathbf{G} \quad (H.0.1)
\]

\[
\nabla \cdot (\mathbf{F} \times \mathbf{G}) = G \cdot (\nabla \times \mathbf{F}) - \mathbf{F} \cdot (\nabla \times \mathbf{G}) \quad (H.0.2)
\]

This list could be made larger but what we are shown here is what is most commonly used. The reader could check for example the Wikipedia site.\(^1\) We add that

\[
\nabla \cdot \nabla f = \nabla^2 f. \quad (H.0.3)
\]

The Laplacian \(\nabla^2 f\) is sometimes noted as \(\Delta f\). If in equation H.0.1 we change \(\mathbf{G}\) by \(\nabla g\), we find:

\[
\nabla \cdot (f \nabla g) = f \nabla \cdot \nabla g + \nabla f \cdot \nabla g. \quad (H.0.4)
\]

That is

\[
\nabla \cdot (f \nabla g) = f \nabla^2 g + \nabla f \cdot \nabla g. \quad (H.0.5)
\]

The identities in this section are commonly used to reduce dimensionality of problems. For example to reduce an integral over a volume to the integral over its surface (the boundary). That is the case of the Green’s and Stoke’s theorems. With Green identities, also built from the identities in this section, Green’s functions for wave operators are built. Here we are somehow going the other way around. We want data that vanishes in the boundaries in order to find the adjoint operator in acting on the whole volumes.

Before we start applying the tools at hand, we should understand inner products in functions of several variables. Assume that \(f\) and \(g\)

\[
f, g : V \subseteq \mathbb{R}^n \rightarrow \mathbb{R}
\]

are scalar differentiable fields, in some open domain \(V\). We define the inner product \(\langle \cdot, \cdot \rangle\) of \(f\) with \(g\) as

\[
\langle f, g \rangle = \int_V f g \, dx, \quad (H.0.6)
\]

where \(x\) rolls over all points of the set \(V\), given that this multi–dimensional integral exists.

Now let us define the corresponding inner product for vector fields. Assume that \(\mathbf{F} = \{f_i\}\) and \(G = g_i, i = 1, 2, .., n\).

\[
\mathbf{F}, \mathbf{G} : V \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^m
\]

\(^1\)http://en.wikipedia.org/wiki/Vector_calculus_identities
are vector differentiable fields, in some domain \( V \). We define the inner product \( \langle ., . \rangle \) of \( f \) with \( g \) as

\[
\langle F, G \rangle = \int_V \sum_{i=1}^m f_i g_i \, dx, = \int_V F \cdot G \, dx.
\]  

(H.0.7)

where \( x \) rolls over all points of the set \( V \), given that this multi–dimensional integrals exists.

**H.0.0.0.1 The gradient and its adjoint the divergence** Assume that \( f \) is a scalar differentiable field in \( V \subseteq \mathbb{R}^n \). Let us call the Hilbert space \( \mathcal{H}_1 \) the space of scalar field functions

\[ f : V \subseteq \mathbb{R}^n \to \mathbb{R} \]

which, where we can take as many we need partial derivatives (\( C^\infty \) if needed) and \( \mathcal{H}_2 \) the space of vector field functions

\[ G : V \subseteq \mathbb{R}^n \to \mathbb{R}^n \]

where also we can take as many derivatives as needed. Call \( L \) the gradient operator

\[
L : \mathcal{H}_1 \to \mathcal{H}_2
\]

\[
f \mapsto L(f) = \begin{pmatrix}
\frac{\partial f}{\partial x_1} \\
\frac{\partial f}{\partial x_2} \\
\vdots \\
\frac{\partial f}{\partial x_n}
\end{pmatrix}
\]

we want to find \( L^* \). Let us take a vector field \( G \) in \( \mathcal{H}_2 \). Then, from the inner product definition H.0.7 we have:

\[
\langle L(f), G \rangle = \int_V \nabla f \cdot G \, dx
\]

\[
= \int_V [\nabla \cdot (f G) - f \nabla \cdot G] \, dx
\]

\[
= \int_V f G \cdot d\Sigma - \int_V f \nabla \cdot G \, dx.
\]

The second equation comes from identity H.0.1 and the third equation comes from the divergence theorem. Here \( \Sigma \) is the boundary of the volume \( V \). Assuming that the product \( f G \) vanishes in the boundary \( \Sigma \) we find

\[
\langle L(f), G \rangle = -\int_V f \nabla \cdot G \, dx = \langle f, -\nabla \cdot G \rangle.
\]
where the second inner product is in the space of scalar functions (see H.0.6). That is, provided that the product \( f \, G \) vanishes in the boundary we found that

\[
\langle L(f), G \rangle = \langle f, L^*(G) \rangle
\]

with

\[
L^* : \mathcal{H}_2 \to \mathcal{H}_1 \\
G \mapsto L^*(G) = -\nabla \cdot G.
\]

We say that the adjoint of the gradient is the negative divergence. Since the adjoint operation is symmetric we can say that the adjoint of the divergence is the negative of the gradient.

H.0.0.0.2 Product of a scalar function and a gradient: Making the previous example a bit more general, let us assume that

\[
L : \mathcal{H}_1 \to \mathcal{H}_2 \\
f \mapsto a(x) \nabla f,
\]

then, given a vector field \( G \) in \( \mathcal{H}_2 \)

\[
\langle L(f), G \rangle = \int_V a(x) \nabla f \cdot G \, dx \\
= \int_V \nabla f \cdot a(x) G \, dx \\
= \int_V [\nabla \cdot (f a(x) G) - f \nabla \cdot a(x) G] \, dx \\
= \int_{\Sigma} (f a(x) G) \cdot d\Sigma - \int_V f \nabla \cdot (a(x) G) \, dx
\]

The second equation is the commutation of the scalar under the dot product, the third equation comes from identity H.0.1 and the fourth equation comes from the divergence theorem. Here \( \Sigma \) is the boundary of the volume \( V \). Assuming that the product \( f \, a(x) \, G \) vanishes in \( \Sigma \) we find

\[
\langle L(a(x) f), G \rangle = -\int_V f \nabla \cdot (a(x) G) \, dx = \langle f, -\nabla \cdot (a(x) \, G) \rangle.
\]

\[
(H.0.9)
\]

where the second inner product is in the space of scalar functions (see H.0.6). That is, provided that the product \( f \, G \) vanishes in the boundary we found that

\[
\langle L(f), G \rangle = \langle f, L^*(G) \rangle
\]
with

\[ L^*: \mathcal{H}_2 \rightarrow \mathcal{H}_1 \]
\[ G \mapsto L^*(G) = -\nabla \cdot [a(x) G] \]

In words, we say that the adjoint maps gradient scaled into scaled divergence. That is, a factor outside the gradient will be sent inside the divergence symbol under the adjoint operation.

**H.0.0.3 The curl:** The curl can be defined in \( n \)-dimensional spaces but here we will only use the three-dimensional space version, in order to simplify the development and because it is the most commonly used. The definition of curl is a linear operator \( \textbf{L} \) from \( \mathcal{H}_2 \) to \( \mathcal{H}_2 \) such that:

\[ \textbf{L}: \mathcal{H}_2 \rightarrow \mathcal{H}_2 \]
\[ \textbf{G} \mapsto \textbf{L}(\textbf{G}) = \nabla \times \textbf{G} = \begin{pmatrix} \partial g_2 / \partial x_3 - \partial g_3 / \partial x_2 \\ \partial g_3 / \partial x_1 - \partial g_1 / \partial x_3 \\ \partial g_1 / \partial x_2 - \partial g_2 / \partial x_1 \end{pmatrix} \]

provided that all partial derivatives exist.

We now find the adjoint of the curl. Using the inner product definition H.0.7 we have that for any vector field function \( \textbf{F} \) in \( \mathcal{H}_2 \)

\[ \langle L \textbf{G}, \textbf{F} \rangle = \int_V (\nabla \times \textbf{G}) \cdot \textbf{F} \, dx \]
\[ = \int_V -\nabla \cdot (\textbf{F} \times \textbf{G}) + \textbf{G} \cdot (\nabla \times \textbf{F}) \, dx \]
\[ = -\int_\Sigma (\textbf{F} \times \textbf{G}) \, d\Sigma + \int_V \textbf{G} \cdot (\nabla \times \textbf{F}) \, dx \]
\[ = -\int_\Sigma (\textbf{F} \times \textbf{G}) \, d\Sigma + \langle \textbf{G}, L(\textbf{F}) \rangle. \]

Here in the second step we used the identity H.0.2 and in the third step the divergence theorem. So, provided the boundary condition \( \textbf{F} \times \textbf{G} = 0 \) on \( \Sigma \), we have that the curl operator is self-adjoint.

\[ ^2 \text{Strictly speaking this should be from } \mathcal{H}_2 \text{ to a less smooth space } \mathcal{H}_3, \text{ since those functions with derivatives up to an order } n \text{, get reduced to spaces with derivatives up to an order } n - 1 \text{ when the curl is operated on them. However due to symmetry theorem of adjoint operators, the adjoint of the adjoint is the original operator, so we need the range to be in } \mathcal{H}_2. \text{ That is, strictly speaking the space of functions for these operators should be } C^\infty. \text{ We will ignore this mathematical detail through the rest of the document because this is irrelevant for the purposes here.} \]
**Product of a scalar function and a curl:** In the previous problem let us multiply the curl by scalar field $a$; that is

$$L : \mathcal{H}_2 \to \mathcal{H}_2$$

$$G \mapsto L(G) = a(x) \nabla \times G.$$ 

Give any vector field $F$ in $\mathcal{H}_2$ we find:

$$\langle L G, F \rangle = \int_V (a(x) \nabla \times G) \cdot F \, dx$$

$$= \int_V (\nabla \times G) \cdot (a(x) F) \, dx$$

$$= \int_V \nabla \cdot (a(x) F \times G) + G \cdot (\nabla \times (a(x) F))$$

$$= \int_\Sigma (a(x) F \times G) \, d\Sigma + \int_V G \cdot (\nabla \times (a(x) F))$$

$$= \int_\Sigma ((a(x) F) \times G) \, d\Sigma + \langle G, L^*(F) \rangle.$$

with

$$L^* : \mathcal{H}_2 \to \mathcal{H}_2$$

$$F \mapsto L^*(F) = \nabla \times (a(x) F).$$

So, provided the boundary condition $(a(x) F) \times G = 0$ in $\Sigma$ we have that the adjoint of the product of a scalar field and a curl is the curl of the product of the scalar field and its vector field argument.

**The Laplacian:** is a linear operator $L$ from $\mathcal{H}_1$ to $\mathcal{H}_1$ such that:

$$L : \mathcal{H}_1 \to \mathcal{H}_1$$

$$f \mapsto L(f) = \nabla^2 f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x^2}.$$ 

provided that all partial derivatives exist.

We want to find the adjoint of the Laplacian. We use the results found above. That is, since, from H.0.0.3 $L(f) = \nabla^2 f = \nabla \cdot \nabla f$, we can write Laplacian as the cascade of two operators:

$$S(f) = \nabla f \quad \text{and} \quad T(S) = \nabla \cdot S$$
so,

\[ \langle L(f), g \rangle = \langle TSf, g \rangle = \langle Sf, T^*g \rangle \quad \text{definition of adjoint} \]
\[ = \langle Sf, -\nabla g \rangle \quad \text{see comment after H.0.8} \]
\[ = \langle f, -S^*\nabla g \rangle \quad \text{definition of adjoint} \]
\[ = \langle f, \nabla \cdot \nabla g \rangle \quad \text{property H.0.8} \]
\[ = \langle f, \nabla^2 g \rangle \]
\[ = \langle f, Lg \rangle. \]

This is an interesting way to prove that the Laplacian is a self–adjoint operator, however it does not explicitly show the boundary conditions for this to happen. Therefore I will try this again using the (already familiar) methodology (integration by parts) used on this document.

\[ \langle L(g), f \rangle = \int_V f \nabla^2 g \, d\mathbf{x} \]
\[ = \int_V \nabla \cdot (f \nabla g) - \nabla f \cdot \nabla g \, d\mathbf{x} \quad \text{see equation H.0.4} \]
\[ = \int_{\Sigma} f \nabla g \cdot d\Sigma - \int_V \nabla f \cdot \nabla g \, d\mathbf{x} \quad \text{divergence theorem} \]
\[ = \int_{\Sigma} f \nabla g \cdot d\Sigma - \int_V [\nabla \cdot (g \nabla f) - g \nabla^2 f] \, d\mathbf{x} \quad \text{H.0.4 again but change } f \text{ by } g \]
\[ = \int_{\Sigma} (f \nabla g - g \nabla f) \cdot d\Sigma + \int_V g \nabla^2 f \, d\mathbf{x} \quad \text{divergence theorem} \]
\[ = \int_{\Sigma} (f \nabla g - g \nabla f) \cdot d\Sigma + \langle g, L(f) \rangle. \]

So we showed again that the adjoint of the Laplacian operator is itself (the Laplacian operator is self–adjoint) and exposed the boundary condition. We require

\[ f \nabla g - g \nabla f = 0 \quad \text{(H.0.10)} \]

in the boundary \( \Sigma \). This proof could have been written in a couple of lines by using the second Green’s identity. Actually we proved the second Green’s identity

\[ \int_V (f \nabla^2 g - g \nabla^2 f) \, d\mathbf{x} = \int_{\Sigma} (f \nabla g - g \nabla f) \cdot d\Sigma. \quad \text{(H.0.11)} \]

**H.0.0.0.6 Product of a scalar function and a Laplacian:** Let assume that the operator \( L \) is such that

\[ L : \mathcal{H}_1 \rightarrow \mathcal{H}_1 \]
\[ f \mapsto L(f) = a(\mathbf{x})\nabla^2 f. \]
As usual we write, for any given scalar field $g$ in $\mathcal{H}_1$

$$
\langle L(f), g \rangle = \int_V a \nabla^2 f \, g \, d\mathbf{x} = \int_V a \, g \, (\nabla^2 f) \, d\mathbf{x}
= \int_V f \, [\nabla^2 (a \, g)] + \int_{\Sigma} [f \, \nabla (a \, g) - (a \, g) \, \nabla f] \cdot d\Sigma.
= \langle f, L^*(g) \rangle + \int_{\Sigma} [f \, \nabla (a \, g) - (a \, g) \, \nabla f] \cdot d\Sigma.
$$

In the third line we used the second Green’s identity H.0.11. So provided the boundary condition:

$$f \, \nabla (a \, g) - (a \, g) \, \nabla f = 0$$
in $\Sigma$, we find that the adjoint of $L$ is:

$$L^*: \mathcal{H}_1 \rightarrow \mathcal{H}_1 \quad g \mapsto L^*(g) = \nabla^2 (a \, f).$$

**Observation** We found that in all cases of product of scalars with operators ($\nabla$, $\nabla \cdot$, $\nabla \times$ and $\nabla^2$), the scalar switches to the inside if it is outside, and vice-versa when finding the adjoint.

**The dot product of a vector function and a gradient. The general first order linear differential operator:** We can write a general first order linear differential operator acting on a scalar field $f \in \mathcal{H}_1$ as

$$L f = \sum_i b_i(x) \frac{\partial f}{\partial x_i} + (c \, f)(x) = B \cdot \nabla f + (c \, f)(x), \quad (H.0.12)$$

for scalar fields $b_i(x)$ and $c(x)$ all in $\mathcal{H}_1$. Here $B$ is a vector with components $b_i(x)$.

The last term (a factor of a scalar field by $f$) is obviously self-adjoint and non-interesting. We are after the adjoint of the first term, which is the dot product of the vectors $B$ and $\nabla f$ and we call $L_B$. 

As usual, let us pick an arbitrary scalar function \( g \in \mathcal{H}_1 \). Then,

\[
\langle L_B(f), g \rangle = \int_V (B \cdot \nabla f) \, g \, dx \\
= \int_V (g B) \cdot \nabla f \, dx \\
= \int_V \nabla \cdot (g f \cdot B) - f \nabla \cdot g B \, dx \\
= \int_{\Sigma} f g B \cdot d\Sigma - \int_V f \nabla \cdot g B \, dx \\
= \int_{\Sigma} f g B \cdot d\Sigma - \langle f, L^* g \rangle.
\]

Second line is commutation, third line is due to identity H.0.1, and fourth line is due to the divergence theorem. Given the boundary condition

\[ f g B = 0, \]

in \( \Sigma \), the adjoint operator \( L^* \) is given by:

\[
L^*_B : \mathcal{H}_1 \rightarrow \mathcal{H}_1 \\
g \mapsto L^* g = -\nabla \cdot (g B).
\]

So the adjoint of the second order operator \( L \) is given by

\[
L^*(g) = L^*_B(g) + c g(x) \\
= -\nabla \cdot g B + c g(x) \\
= -\sum_{i=1}^n \frac{\partial}{\partial x_i} gb_i(x) + (c g)(x). 
\] (H.0.13)

with the boundary condition:

\[ (f g b_i)(x) = 0, \]

for any \( x \) in \( \Sigma \).

It is common on wave equations to find the operator defined by

\[
L u(x) = \nabla \cdot (a(x) \nabla u(x)).
\]

By the rules of differentiation of the divergence we see that

\[
L u(x) = \nabla a(x) \cdot \nabla u(x) + a(x) \nabla^2 u(x).
\]
and from the previous derivation with $B = \nabla a(x)$ and the adjoint of the product of a scalar function and a Laplacian we have

$$L^*(g) = -\nabla \cdot [g\nabla a] + \nabla^2(a g)$$

$$= -\nabla a \cdot \nabla g - g\nabla^2 a + \nabla^2(a g)$$

$$= -\nabla a \cdot \nabla g - g\nabla^2 a + (a\nabla^2 g + 2\nabla a \cdot \nabla g + g\nabla^2 a)$$

$$= \nabla a \cdot \nabla g + a\nabla^2 g$$

$$= \nabla \cdot (a \nabla g)$$

So $L = L^*$ and $L$ is self-adjoint.

In particular Tarantola’s [?] example on the acoustic wave equation.

$$Lu = \left[ \frac{1}{K(x)} \frac{\partial^2}{\partial t^2} - \nabla \cdot \left( \frac{1}{\rho(x)} \nabla \right) \right] u(x, t) = s(x, t). \quad \text{(H.0.14)}$$

is self-adjoint since $K(x)$ is time independent.

Table H.1 shows a summary of the results obtained so far, assuming the proper boundary conditions are chosen so that the surface integral contribution is zero.

<table>
<thead>
<tr>
<th>Gradient/Divergence</th>
<th>$a(x)\nabla u(x)$</th>
<th>$-\nabla \cdot [a(x)G(x)]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Curl</td>
<td>$a(x) \nabla \times F(x)$</td>
<td>$\nabla \times [a(x)F(x)]$</td>
</tr>
<tr>
<td>Laplacian</td>
<td>$a(x)\nabla^2 u(x)$</td>
<td>$\nabla^2[a(x)g(x)]$</td>
</tr>
<tr>
<td>Mixed</td>
<td>$\nabla \cdot a(x)\nabla u(x)$</td>
<td>$\nabla \cdot a(x)\nabla g(x)$</td>
</tr>
</tbody>
</table>

A general linear second order differential operator: A general linear second order differentiator operator in $\mathcal{H}_1$ can be written as

$$L : \mathcal{H}_1 \rightarrow \mathcal{H}_1$$

$$f \mapsto L(f) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \frac{\partial^2 f}{\partial x_i \partial x_j} + \sum_{i=1}^{n} b_i \frac{\partial f}{\partial x_i} + c f. \quad \text{(H.0.15)}$$

Let us find the adjoint $L^*$. The first and zero order differentials are trivial and considered in previous examples. Let us focus in the first double sum. That is, let us define

$$L_A : \mathcal{H}_1 \rightarrow \mathcal{H}_1$$

$$f \mapsto L_A(f) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \frac{\partial^2 f}{\partial x_i \partial x_j} +$$
We are tempted to start applying the definition of inner product and integration by parts to get \( L_A^* \). However I do not know about any integration by parts that will take care of the whole matrix (of \( a_{ij} \)) to matrix (of second order partial derivatives) inner product \(^3\). Based on the observation in the observation H.0.0.0.7 above, I guess how \( L^* \) should look like and then verify that it is indeed the adjoint we are looking for.

I claim that

\[
L_A^* : \mathcal{H}_1 \rightarrow \mathcal{H}_1 \quad \ni \quad g \mapsto L_A^*(g) = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2(a_{ij} g)}{\partial x_i \partial x_j} + (H.0.16)
\]

Then I compute

\[
\langle L_A f, g \rangle - \langle L_A^* g, f \rangle
\]

and show that it is an exact differential, which after applying the divergence theorem produces the condition in the boundary \( \Sigma \). We see that:

\[
\langle L_A f, g \rangle - \langle L_A^* g, f \rangle = \int_{V} \sum_{i=1}^{n} \sum_{j=1}^{n} \left( a_{ij} g \frac{\partial^2 f}{\partial x_i \partial x_j} - f \frac{\partial^2(a_{ij} g)}{\partial x_i \partial x_j} \right) \, dx
\]

The expression in the parenthesis can be written as

\[
a_{ij} g \frac{\partial^2 f}{\partial x_i \partial x_j} - f \frac{\partial^2(a_{ij} g)}{\partial x_i \partial x_j} = a_{ij} g \frac{\partial^2 f}{\partial x_i \partial x_j} + \frac{\partial(a_{ij} g)}{\partial x_i} \frac{\partial f}{\partial x_j} - \frac{\partial(a_{ij} g)}{\partial x_j} \frac{\partial f}{\partial x_i}
\]

\[
= \frac{\partial}{\partial x_i} \left( a_{ij} g \frac{\partial f}{\partial x_j} \right) - \frac{\partial}{\partial x_j} \left( f \frac{\partial(a_{ij} g)}{\partial x_i} \right).
\]

Now, since

\[
\sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial}{\partial x_j} \left( f \frac{\partial(a_{ij} g)}{\partial x_i} \right) = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial}{\partial x_i} \left( f \frac{\partial(a_{ij} g)}{\partial x_j} \right),
\]

we find that

\[
\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} g \frac{\partial^2 f}{\partial x_i \partial x_j} - f \frac{\partial^2(a_{ij} g)}{\partial x_i \partial x_j} = \sum_{i=1}^{n} \frac{\partial}{\partial x_i} \left[ \sum_{j=1}^{n} a_{ij} g \frac{\partial f}{\partial x_j} - f \frac{\partial(a_{ij} g)}{\partial x_i} \right].
\]

\(^3\)Actually \( A \) is a tensor operator and in Einstein’s notation, \( a_{ij} \partial_i \partial_j f \) is a zero rank contracted tensor. This belongs to the field of differential forms under the mathematical branch of differential geometry.
and so
\[ \langle L_A f, g \rangle - \langle L_A^* g, f \rangle = \int_V \sum_{i=1}^{n} \frac{\partial}{\partial x_i} \left[ \sum_{j=1}^{n} a_{ij} g \frac{\partial f}{\partial x_j} - f \frac{\partial (a_{ji} g)}{\partial x_j} \right]. \]
\[ = \int_V \nabla \cdot p_i d\mathbf{x} \]

Is an exact differential with
\[ p_i = \sum_{j=1}^{n} a_{ij} g \frac{\partial f}{\partial x_j} - f \frac{\partial (a_{ji} g)}{\partial x_j}, \]

and we can apply the divergence theorem to finally get
\[ \langle L_A f, g \rangle - \langle L_A^* g, f \rangle = \int_{\Sigma} p_i \cdot d\Sigma. \]

So provided that \( p_i = 0 \) in the boundary \( \Sigma \), the adjoint of \( L_A \) exists and it is given in equation H.0.16. Combining this result with the adjoint of the general linear first order differential operator in equation H.0.13 we find that the adjoint of the general linear second order partial differential operator H.0.15 is
\[ L^* : \mathcal{H}_1 \rightarrow \mathcal{H}_1 \]
\[ g \mapsto L(g) = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2 (a_{ij} g)}{\partial x_i \partial x_j} - \sum_{i=1}^{n} \frac{\partial (b_i g)}{\partial x_i} + c g. \] (H.0.17)

with the boundary condition
\[ p_i = \sum_{j=1}^{n} a_{ij} g \frac{\partial f}{\partial x_j} - f \frac{\partial (a_{ji} g)}{\partial x_j} + b_i f g = 0 \]

for all \( x \) in \( \Sigma \).

With this adjoint we cover all second order linear partial differential equations including all wave equations.

We use the results on this section to derive the adjoints of the wave equation operators in the examples for this document.
Index

Armijo, 7
ascent direction, 7
bisection, 7
Cholesky decomposition, 200
conjugate gradient, 7
descent methods, 7
eigenvalues, 11
eigenvectors, 11
Gauss-Siedel method, 16
golden rule, 7
gradient descent, 19
Jacobi method, 15
Krylov subspace, 51
least square, 20
least squares problem, 16
line search, 7, 21
Mahalanobis norm, 200
Newton methods, 7
parabolic interpolation, 7
positive definite, 19
quasi-Newton methods, 7
singular value decomposition, 15
steepest descent, 7
steepest descent method, 19
Successive over-relaxation method, 16
Tikhonov regularization, 16
trust region, 7