GLOBAL OPTIMIZATION ALGORITHMS
FOR SEISMIC INVERSION

A DISSERTATION SUBMITTED TO THE GRADUATE DIVISION OF
THE UNIVERSITY OF HAWAII IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

GEOLOGY AND GEOPHYSICS

MAY 1995

By

Bertram Nolte

Dissertation Committee:

L. Neil Frazer, Chairperson
Eduard Berg
Gerard J. Fryer
Roy H. Wilkens
Eric Firing
We certify that we have read this dissertation and that, in our opinion, it is satisfactory in scope and quality as a dissertation for the degree of Doctor of Philosophy in Geology and Geophysics.

DISSERTATION COMMITTEE

L. Neil Fraser
Chairperson

[Signatures]

ii
ACKNOWLEDGMENTS

I thank my advisor, Neil Frazer, for all his advice, support and encourage-
ment, for his interest in my research, and for all the things he taught me
about science. I am sure I will miss Neil's advice in the future. I thank
Eduard Berg for teaching me a great deal of geophysical theory, and for
many helpful discussions about my research, which took a lot of his time.
Thanks also go to Gerard Fryer and Roy Wilkens for their interest in my
research, their helpful comments and their encouragement. Eric Firing
kindly agreed to serve on my committee, and I thank him for his interest
in my work.

Patricia Cooper provided me with the ODP dataset discussed in
Chapter 3.

Many past and present SOEST people have helped me over the years,
especially Joe Dellinger, Patricia Berge, Subhashis Mallick, and all the
seismology graduate students.

Finally and most of all I thank my parents, Helmut and Gabriele
Nolte, who gave me the opportunity to receive a good education, which in
the long run made this dissertation possible.
ABSTRACT

In this dissertation I investigate the applicability of global optimization methods to seismic inversion. The methods I use are genetic algorithms (GA) and simulated annealing (SA). I first describe a new GA inversion technique and apply it to a vertical seismic profile (VSP) inversion problem where the goal is to recover slowness and impedance profiles. The algorithm consists of one long and a series of short GA runs. After each run the problem is reparameterized using the matrix of partial second derivatives in a neighborhood of the best model found in that run. The new unknown parameters are more nearly independent than the old ones, thus successive GA runs find better models faster, in accordance with the principle of meaningful building blocks.

I then apply both GA and SA to VSP inversion for slownesses and reflection coefficients. For a 45-layer synthetic problem, neither algorithm has trouble finding a solution close to the true model, and SA is somewhat faster than the GA. For a real-data problem of the same size the algorithms compare to each other similarly as for the synthetic problem. However, different inversion runs tend to converge to different solutions for both SA and GA. Therefore, even though both models have a similar potential to find good models, the method that is preferable is the one that is better suited for estimation of the a posteriori probability distribution. This method is clearly SA.
## TABLE OF CONTENTS

ACKNOLEDGMENTS ........................................... iii

ABSTRACT ......................................................... iv

LIST OF TABLES ........................................... vii

LIST OF FIGURES ............................................. viii

CHAPTER 1: INTRODUCTION ......................................... 1

1.1 INVERSION AND GLOBAL OPTIMIZATION ................. 1

1.2 APPLICATION OF GLOBAL OPTIMIZATION TO VSP ... 2

1.3 REFERENCES ...................................................... 3

CHAPTER 2: VERTICAL SEISMIC PROFILE INVERSION WITH GENETIC ALGORITHMS ................................. 4

2.1 SUMMARY ......................................................... 4

2.2 INTRODUCTION ...................................................... 5

2.3 FUNDAMENTALS OF GENETIC ALGORITHMS ............... 7

2.4 THE GENETIC ALGORITHM FOR THE VSP PROBLEM ............... 9

2.4.1 Computation of the fitness function ................. 9

2.4.2 Parameterization of the model ...................... 15

2.4.3 Selection algorithm ....................................... 16

2.4.4 Crossover algorithm ..................................... 17

2.5 INVERSION OF NOISE-FREE SYNTHETICS ............ 18

2.5.1 Reparameterization ....................................... 20

2.5.2 Update algorithm ......................................... 27
# TABLE OF CONTENTS

2.5.3 Inversion with new model parameters .................. 28

2.6 INVERSION OF NOISY SYNTHETICS ....................... 31

2.7 DISCUSSION AND CONCLUSIONS .......................... 39

2.8 REFERENCES ............................................. 53

CHAPTER 3: COMPARISON OF SIMULATED ANNEALING
AND GENETIC ALGORITHMS FOR A SEISMIC
INVERSION PROBLEM ......................................... 56

3.1 SUMMARY .................................................. 56

3.2 INTRODUCTION ............................................. 57

3.3 THEORY ..................................................... 60

3.3.1 Fitness function ........................................ 60

3.3.2 Inversion algorithms .................................... 63

3.4 A SYNTHETIC EXAMPLE .................................... 64

3.4.1 SA inversion ............................................ 67

3.4.2 GA inversion ............................................ 70

3.4.3 Comparison of SA and GA ............................... 73

3.5 INVERSION OF REAL DATA ................................. 74

3.5.1 Comparison of SA and GA ............................... 74

3.5.2 PPD estimation .......................................... 79

3.5.3 Comparison with physical properties measurements ... 85

3.6 DISCUSSION AND CONCLUSIONS ............................ 88

3.7 REFERENCES ............................................... 89
LIST OF TABLES

TABLE 2.1 Eigenvalues of the eigenvectors in Fig. 2.6 ............... 25

TABLE 2.2 Eigenvalues of the eigenvectors in Fig. 2.16 ............... 38
LIST OF FIGURES

FIGURE 2.1 Model for synthetics ......................................... 10
FIGURE 2.2 Synthetic data ................................................... 11
FIGURE 2.3 Synthetic data (upgoing wavefield) .......................... 13
FIGURE 2.4 Cooling schedule for the genetic algorithm ................. 15
FIGURE 2.5 Results of the first inversion run ............................ 19
FIGURE 2.6 Eigenvectors for the true model and the best model of Fig. 2.5 .......................................................... 24
FIGURE 2.7 Schematic of increased search space after reparameterization .......................................................... 26
FIGURE 2.8 Results of the second inversion run .......................... 29
FIGURE 2.9 Eigenvectors for the best model of Fig. 2.8 ................. 30
FIGURE 2.10 Results of the third inversion run ........................... 32
FIGURE 2.11 Results for the new parameters ............................. 33
FIGURE 2.12 Flow chart of our algorithm ................................ 34
FIGURE 2.13 Noisy synthetics ................................................ 35
FIGURE 2.14 Noisy synthetics (upgoing wavefield) ............... 36
FIGURE 2.15 Results of the first inversion run ........................... 37
FIGURE 2.16 Eigenvectors for the best model of Fig. 2.15 ........... 39
FIGURE 2.17 Results of the second inversion run ....................... 40
FIGURE 2.18 Eigenvectors from the 6th inversion run ............... 41
FIGURE 2.19 Results of the 7th inversion run ............................ 42
FIGURE 2.20 Results for the new parameters ........................... 43
# LIST OF FIGURES

FIGURE 2.21 Results of 20 independent inversion runs ................. 45

FIGURE 2.22 Results of Fig. 2.20 with corrections for the two smallest eigenvalues ........................................ 52

FIGURE 3.1 Model for synthetics ........................................ 65

FIGURE 3.2 Synthetic data ............................................. 66

FIGURE 3.3 SA results .................................................. 69

FIGURE 3.4 GA results .................................................. 72

FIGURE 3.5 Fitness versus number of forward computations .......... 73

FIGURE 3.6 Measured data ............................................. 76

FIGURE 3.7 SA results from 3 independent runs ....................... 77

FIGURE 3.8 GA results from 3 independent runs ....................... 78

FIGURE 3.9 Fitness versus number of forward computations .......... 79

FIGURE 3.10 Marginal PPDs from three independent runs with 600 sweeps each .................................................. 82

FIGURE 3.11 Marginal PPDs from six independent runs with 200 sweeps each .................................................. 83

FIGURE 3.12 Marginal PPDs from one run with 1000 sweeps .......... 84

FIGURE 3.13 Average models from Fig. 3.10 and 3.11 .................. 86

FIGURE 3.14 Comparison to physical properties ....................... 87
Chapter 1

Introduction

1.1 INVERSION AND GLOBAL OPTIMIZATION

One of the goals of seismic inversion is to obtain from seismic measurements \( d \) a model \( m \) that describes the elastic behavior of the earth. Part of an inversion algorithm is a forward modeling routine which enables one to compute synthetic data \( s \) for any model \( m \). A model \( m \) is evaluated by either minimizing the misfit or maximizing the correlation of \( s(m) \) and \( d \).

Conventional inversion techniques (Menke 1984, Tarantola, 1987) improve a starting model iteratively. At each step the model is updated by using the derivatives of the objective function with respect to the model parameters. Methods that use the first derivatives only are called gradient methods, and methods that use both the first and second derivatives are called Newton methods. The disadvantage of all these methods is that in most cases they require a good starting model, as they tend to find the local extremum of the objective function closest to the starting model, rather than its global extremum.

Global optimization methods, like simulated annealing (Kirkpatrick et al. 1983) and genetic algorithms (Holland 1975) do not use derivatives and thus have the potential to work without a good starting model. Simulated annealing (SA) simulates
a *heat bath* of a solid, in which the solid is heated above its melting point so that the atoms arrange themselves freely. The melt is then cooled slowly enough for the atoms to arrange themselves in the minimum energy state, the crystal lattice. The model parameters in SA are the analog to the atoms, and the objective function is the analog to energy.

Genetic Algorithms (GA) simulate biological evolution. Models are coded as strings of (usually binary) numbers called *chromosomes*. In analogy with genetics, three main operations, *reproduction*, *crossover* and *mutation*, are performed on a *population* of chromosomes, resulting in a new generation of chromosomes. The idea behind GA is that a population will eventually evolve to be optimally adapted to the environment, which corresponds to the objective function in GA.

### 1.2 APPLICATION OF GLOBAL OPTIMIZATION TO VSP

In this dissertation I investigate both SA and GA for a vertical seismic profile (VSP) inversion problem. In contrast to surface seismics, in a VSP receivers are located in a borehole and record the motion at different depths. I use stratified models, i.e. models in which the elastic properties are a function only of depth. For this type of model the exact computation of the complete wavefield, i.e. all multiple arrivals, can be performed rapidly. For the forward modeling I use the global matrix method (Schmidt & Tango 1986) which computes the wave amplitudes at all depths simultaneously by solving all boundary conditions simultaneously. It is thus well suited for VSP modeling.

In Chapter 2 I apply GA to the inversion of synthetic VSPs. I use a simple 11-layer earth model and develop an algorithm that, for this model, is always able to find the global extremum of the objective function. I describe the algorithm in detail and demonstrate its performance with several numerical examples.

In Chapter 3 I apply both the GA of Chapter 2 and an SA algorithm to the
inversion of synthetic data, and compare the results in terms of both computational effort and quality of the results. The synthetics are computed for a 45-layer model that is in many ways more realistic than the model of Chapter 2. Finally, I compare both algorithms for a real-data inversion.

1.3 REFERENCES


Chapter 2

Vertical seismic profile inversion with genetic algorithms

2.1 SUMMARY

We describe a new genetic algorithm (GA) inversion technique and apply it to a vertical seismic profile (VSP) inversion problem where the goal is to recover slowness and impedance profiles. Our algorithm consists of one long (300 generations) and a series of short (50 generations) GA runs. After each run the problem is reparameterized using the matrix of partial second derivatives in a neighborhood of the best model found in that run. The new unknown parameters are more nearly independent than the old ones, thus successive GA runs find better models faster, in accordance with the principle of meaningful building blocks.

For the VSP problem we use a fitness function that is the weighted sum of the crosscorrelation between the measured and synthetic total wavefields and the crosscorrelation between the measured and synthetic upgoing wavefields. The measured upgoing wavefield is obtained from the measured total wavefield by f-k filtering.

Key words: Non-linear inversion, genetic algorithms, VSP
2.2 INTRODUCTION

Seismic waveform inversion is a non-linear problem. The conventional inversion approach minimizes a misfit function in a least-squares sense. This misfit or objective function is approximated to first or second order around a given starting model, and the minimum is approached iteratively. The disadvantage of conventional methods is the need for a good starting model. Often this a priori information is not available, and then the inversion may converge to a local minimum of the misfit function, rather than converging to the global minimum. The results obtained from conventional methods are thus always biased by the starting model.

Optimization methods like simulated annealing or genetic algorithms are attempts to obtain good solutions without the requirement of a good starting model. Simulated annealing has recently been applied to seismic waveform inversion (Basu & Frazer 1990, Frazer & Basu 1990, Sen & Stoffa 1991). Genetic algorithms have been applied to many different problems in engineering, computer science, and other fields (Goldberg 1989), and also to geophysical problems. Wilson & Vasudevan (1991) applied GA to the problem of residual statics estimation. Jin & Madariaga (1993) used GA for background-velocity inversion of seismic reflection data, and they applied their algorithm to a seismic profile from the North Sea. Gallagher et al. (1991) compared the performance of GA and Monte Carlo techniques and found GA to be superior. Stoffa & Sen (1991, 1992) and Sen & Stoffa (1992) applied GA to waveform inversion with the goal of estimating the a posteriori probability density function (PPD) in model space. Sambridge & Drijkoningen (1992) used GA for the inversion of marine refraction data and compared the performance of GA to both Monte Carlo and conventional linearized inversion procedures. They found that GA was clearly superior to Monte Carlo and possibly also superior to linearized inversion schemes that require a good starting model. Scales et al. (1992) compared GA and simulated annealing for waveform inversion, using reflection coefficients as unknown model pa-
rameters. They found that both methods performed well for problems with a small number of free parameters, but if the number was increased GA became superior.

A GA is a simulation of natural genetics that works simultaneously with a set of models called a population. Each model is coded as a string (chromosome), where model parameters are segments of the string. Each substring consists of a sequence of characters (genes). The GA is linked to the problem that is to be solved through a fitness function, which measures how well a model satisfies the data. From one generation to the next the GA usually increases the fitness of the best model and the average fitness of the population.

With a proper choice of parameters a GA can be made to find good models rather rapidly. However, there are two problems. First, the desired starting-model independence is usually not achieved in GA inversions. Except possibly for the algorithm of Berg (1990, 1991) different starting populations often lead to different results. Second, a GA cannot be made to sample from the PPD determined by the data, because in each generation the exploration of the model space is biased toward the fittest members of the current population. Thus, even though genetic algorithms are good at finding one particular good model, there is at present no theoretical foundation for using them to compute the PPD. Nevertheless Stoffa & Sen (1991, 1992) attempted to estimate the PPD with many independent GA runs with different starting populations, and then used a graph-binning technique to estimate the marginal PPD's. Their assumption was that through these repeated runs they had sampled the most significant part of the model space and that their display was a good estimate of the true marginal PPD's. However, it seems to be difficult if not impossible to tell how many independent runs have to be performed in order for this assumption to be true. Given the nature of GA it is likely that even if GA were started at every point in a (discrete) model space one would still not obtain a good estimate of the PPD. Graph-binning of GA solutions is useful, but the results should be treated cautiously and not confused with PPD's.
Consequently, in this paper we do not focus on the computation of a PPD over
the model space, though this is often the goal of inversion. Instead we design a method
that gives a model near the true model and, at the same time, a considerably reduced
search space. We test this method on a one-dimensional VSP inversion problem.
Unlike a simple GA, our method appears to be starting-population independent. Our
procedure is thus a rapid and powerful “preprocessor” either for itself, when used
iteratively, or for a more conventional inversion technique.

2.3 FUNDAMENTALS OF GENETIC
ALGORITHMS

We briefly review the fundamentals of genetic algorithms. More comprehensive treat­
ments of this subject can be found in Goldberg (1989) and Davis (1991). As noted
above, in a GA each model is coded as a string, called a chromosome in analogy with
natural genetics. Model parameters are substrings that sit next to each other on the
chromosome and consist of characters (genes). If binary coding is used then each bit
in the binary representation of the parameter is a gene. Possible values of a gene (1
and 0 in the case of binary coding) are often called alleles. We use binary coding here,
and therefore limit our discussion to this case. If $N_b$ is the number of bits assigned
to a parameter, then this parameter can take $2^{N_b}$ different values. These values lie
between upper and lower bounds that define the search interval for that particular
parameter. If we have $N_m$ model parameters to each of which $N_b$ bits are assigned,
the length of the chromosome will be $N_m \times N_b$.

Genetic algorithms work simultaneously with a group of different strings called
a population. The starting population is usually selected randomly, i.e. the value
of every gene of every chromosome is randomly set to either 0 or 1. Then, for each
string the fitness is calculated. The fitness of a string is a measure of performance
of the model represented by that string. Maximizing a fitness function is similar to
minimizing a misfit function, a familiar concept in inverse theory.
After the population is initialized, new descendant populations are iteratively created, with the goal of an overall fitness increase from generation to generation that may finally lead to a population rich in models near to the global maximum. Each new generation is created from the prior one by three main operations: selection, crossover, and mutation.

Selection assigns each member of the population a probability of advancing to the next generation that is proportional to its relative fitness within the current population. Examples of selection algorithms are stochastic selection with replacement, stochastic remainder selection without replacement, and deterministic remainder selection (Goldberg 1989). We found the latter to be the most efficient algorithm, and we explain it in more detail later. Selection ensures that models with a higher fitness have a better chance to advance to the next generation, but selection alone produces no new models and thus does not search the model space. Therefore additional operations, crossover and mutation, are needed for the exploration of the model space.

The simplest crossover algorithm is single-point crossover: The members of the population are randomly paired. For each pair, crossover is performed with probability $p_c$. If it is decided (using $p_c$) that crossover is to be performed for a certain pair, a crossover point is randomly chosen between the first and the last string position, and bits thereafter are exchanged between the two strings. In this way a new population of strings is created by combining substrings from the old population. Other algorithms are multiple-point crossover (e.g. Goldberg 1989), where several crossover points are selected and thus several substrings are exchanged within a pair, and uniform crossover (Syswerda 1989) where bits are exchanged randomly within a pair.

Another operation, mutation, is necessary to guarantee diversity in the population. Mutation, usually carried out after crossover, is just a random change of the value of a gene from 0 to 1 or from 1 to 0. It is applied with a low probability $p_m$ to every gene of each string. The purpose of mutation is to make sure that no ge-
netic possibility is irretrievably lost, as that would reduce the model space for future searches.

After a new population, the next generation, has been created through selection, crossover, and mutation, the old population is discarded and the process is repeated. The combined effect of these operations is to increase the fitness of the population from one generation to the next.

2.4 THE GENETIC ALGORITHM FOR THE VSP PROBLEM

We develop a GA to invert VSPs in stratified media for both slowness (inverse wavespeed) and impedance. We solve the forward problem with a frequency-wavenumber modeling code that is based on a global matrix method (Schmidt & Jensen 1985, Schmidt & Tango 1986). This exact technique computes the complete wavefield, i.e., all multiple arrivals, and is attractive for VSP computation because the wavefield is obtained simultaneously in all layers.

Figure 2.1 shows the slowness and impedance profiles of our test model. The layers at the bottom and the top of the stack are halfspaces. Figure 2.2 shows 100 noise-free synthetic seismograms for vertically traveling plane waves computed for the model in Fig. 2.1, over a depth range of 1000 m with a receiver spacing of 10 m. Our goal is to estimate the slowness and impedance profiles in Fig. 2.1 using the data in Fig. 2.2.

2.4.1 Computation of the fitness function

As our synthetics are computed in the frequency domain it is efficient to use a frequency domain fitness function. Following Stoffa & Sen (1991), we define the fitness function as the real part of the normalized crosscorrelation of the synthetics with the data:
Figure 2.1: Model for the computation of the synthetic data.
Figure 2.2: Synthetic data used for the inversion.
Here $N_z$ is the number of receivers, while $d_{z,\omega}$ and $s_{z,\omega}$ denote, respectively, the spectra of the data and the synthetics; $\ast$ denotes a complex conjugate. The fitness function $f$ is normalized to take values between -1 and 1. It is a function of only the slownesses and impedances of the model used to compute the synthetics. To evaluate $f$ the data must be Fourier transformed whereas the synthetics, as mentioned above, are computed directly in the frequency domain.

The seismograms in Fig. 2.2 are dominated by the direct arrivals, which contain mainly information about the slowness structure, but little information about the impedance structure. To make our inversion more sensitive to the impedances we give the upgoing waves a stronger weight in the fitness function than the downgoing waves. The upgoing wavefield can be extracted from the data in several ways, some of which require the picking of first arrivals. We choose f-k filtering (e.g. Hardage 1983), which does not require arrival time picking and can be applied to data with equally spaced receivers if the spacing is small enough to prevent spatial aliasing. Figure 2.3 shows the upgoing wavefield that was obtained from the data in Fig. 2.2 through f-k filtering.

We now modify the criterion for model fitness by adding weight to the upward traveling waves. We use equation (1) to compute both a fitness function $f_c$ for the complete wavefield and another fitness function $f_u$ for the upgoing wavefield. Our new fitness is then given by

$$f = a f_c + (1 - a) f_u,$$

where $a$ is a number between 0 and 1 that controls the relative weight of $f_c$ and $f_u$.

We could have used the downgoing wavefield instead of the complete wavefield $f_c$; however, we prefer $f_c$, because it makes our results less susceptible to the artifacts.
Figure 2.3: Upgoing wavefield obtained from the data in Fig. 2.2 with f-k filtering.
of f-k filtering. In practice it does not make much difference if we use the complete or
the downgoing wavefield, since the additional parameter $a$ controls the relative weight
of the upgoing wavefield in the fitness function. Note that we have to filter only our
data; we directly calculate both the complete and the upgoing synthetic wavefields. It
would be preferable to also extract the upgoing synthetic wavefield from the complete
synthetic wavefield by f-k filtering, for then the artifacts of the filtering will be similar
in both data and synthetics. This would require f-k filtering the synthetics from each
model generated, which is computationally more expensive.

As has been suggested by others (e.g. Goldberg 1989) it is useful to introduce
some kind of "stretching" of the fitness function. Our stretched fitness function $g$ is
given by Frazer et al. (1990) and Stoffa & Sen (1991, 1992) as

$$g = \exp(f/T),$$

where $T$ is a control parameter which we call temperature in analogy to simulated
annealing (e.g. Basu & Frazer 1990). A high temperature will discriminate little
among models with different fitnesses, a desirable property in the beginning stage
of the inversion. In early generations a fit individual should not immediately domi­
inate the population, as that would limit the exploration of new model space, with
the consequence that the population would quickly converge to that particular fit
individual. In later stages of the inversion a low temperature is more advantageous.
At this point there are presumably many good models with slightly different values
of $f$. A low $T$ will amplify this difference, and thus enable the GA to discriminate
among these models. Accordingly, $T$ should be a function of the generation number
analogous to a cooling schedule in simulated annealing. Figure 2.4 shows the cooling
schedule that we use in the following. For the first $N_1$ generations $T$ is kept at a
rather high constant value $T_1$. At generation number $N_1$ the temperature begins to
decrease linearly until temperature $T_2$ is reached at generation number $N_2$. For the
remaining generations the temperature is kept constant at this level.
2.4.2 Parameterization of the model

If we knew the number of different layers and the layer thicknesses in the medium from which our data were obtained, then our model would consist of the slowness and impedance values for each of these layers of known thickness. Here we assume no prior knowledge of layer thicknesses; rather, we use layers of constant thickness slightly less than the theoretical depth resolution of our data, i.e., we overparameterize. With many layers the inversion program tends to find solutions in which slowness and impedance oscillate from layer to layer but have fitness values similar to smoother models. As we usually prefer smooth models over highly oscillatory models (in the absence of any information that helps us discriminate between two models, we prefer the simpler model) we want to decrease the fitnesses of the latter relative to the former by using a roughness penalty function, as in Basu & Frazer (1990). They used a penalty function of the form
\[ p(m) = \frac{\sum_{i=2}^{N_z-1} |m_{i+1} - 2m_i + m_{i-1}|}{2(N_z - 2)(m_{\text{max}} - m_{\text{min}})}, \]

where \( m = [s, \eta] \) denotes the vector of model parameters. We calculate one penalty function \( p(s) \) for the slownesses \( s \) and another function \( p(\eta) \) for the impedances \( \eta \), and decrease \( f \) of equation (2) by some fraction \( \epsilon \) of these functions:

\[ \tilde{f} = f - \epsilon p(s) - \epsilon p(\eta). \]

We then use \( \tilde{f} \) in equation (3) rather than \( f \).

We use the five traces at depths of 100, 300, 500, 700, and 900 m for the inversion and invert for 11 layers each of which is 100 m thick. Using all traces would increase the computation time and is unnecessary for this study. As we want to determine two parameters for each layer, we have \( N_m = 22 \) model parameters. We code each of them as a binary number with \( N_b = 6 \) bits. Thus each parameter can take 64 possible values, and the total length of the chromosome is \( 6 \times 22 = 132 \) bits. The first 66 bits define the slownesses and the last 66 bits define the impedances. The search intervals are 0.4 s km\(^{-1}\) to 0.67 s km\(^{-1}\) for the slownesses and \( 2.25 \times 10^6 \) kg m\(^{-2}\) s\(^{-1}\) to \( 5.75 \times 10^6 \) kg m\(^{-2}\) s\(^{-1}\) for the impedances.

### 2.4.3 Selection algorithm

We tried different selection algorithms. The simplest algorithm is stochastic selection with replacement, also called roulette-wheel selection (Goldberg 1989). Each new individual is drawn from the entire old population, where the probability that a particular individual will be selected is proportional to its fitness. This is repeated until the new population has the same size as the old one. The obvious disadvantage of this method is its susceptibility to stochastic errors. Unless the population is very large it is highly unlikely that the number of instances of an individual in the new population will be proportional to its fitness. We found that this type of selection
makes the results strongly dependent on the set of random numbers that was used for the GA.

Two better selection algorithms are stochastic remainder selection without replacement and deterministic remainder selection (Goldberg 1989). In both methods, for each individual \( j \) an expected number of copies \( E_j \) is calculated as

\[
E_j = N_{pop} \frac{f_j}{\sum_{i=1}^{N_{pop}} f_i}.
\]

\( f_i \) is the fitness of individual \( i \), and \( N_{pop} \) is the population size.

First, each member \( j \) of the population receives \( N_j = \text{int}(E_j) \) copies. The remaining individuals are selected according to the fractional part of \( E_j \). In stochastic remainder selection without replacement they are drawn with a probability proportional to the fractional part; however, in contrast to roulette wheel selection, an individual that is selected once cannot be selected again. In deterministic remainder selection the population is sorted according to the fractional part of \( E_j \) and the individuals needed to fill up the population are drawn from the top of the list. Note that no random number generator is used for deterministic remainder selection.

Although both stochastic remainder selection without replacement and deterministic remainder selection work well, we found that the latter is usually slightly better. As a random number generator is used only for crossover and mutation, but not for selection, the dependence of the inversion algorithm on the particular set of random numbers is somewhat reduced. All the examples that we show in this paper have been computed with deterministic remainder selection.

### 2.4.4 Crossover algorithm

After selection has been performed the individuals in the population are randomly paired. Crossover is then performed for each pair with probability \( p_c \).

There have been several studies of crossover algorithms (e.g. Eshelman et al.,
1989) and there seems to be no general agreement among authors on which algorithm is preferable. The simplest crossover is single-point crossover, in which only one crossover point is specified and the two resulting substrings are exchanged between the two individuals. In multiple-point crossover several smaller substrings are exchanged. Usually the crossover points in the chromosomes are selected randomly. Sen & Stoffa (1991) used multiple-point crossover, but with the restriction that each parameter substring contains exactly one crossover point. Syswerda (1989) introduced a uniform crossover, where bits are randomly exchanged between strings and claimed it to be superior to one- and two-point crossover.

In our case single- and multiple-point crossover gave equally good results, whereas the performance of uniform crossover was slightly poorer. For all computations shown in this paper we have used single-point crossover.

2.5 INVERSION OF NOISE-FREE SYNTHETICS

The GA parameters for the inversion of the data in Fig. 2.2 and Fig. 2.3 were chosen after several trials as $N_{\text{pop}} = 50$, $p_c = 0.9$, and $p_m = 0.01$. The cooling schedule was chosen as in Fig. 2.4, with $T_1 = 1.5$, $T_2 = 0.05$, $N_1 = 40$, and $N_2 = 240$. We computed 300 generations. We used this rather slow cooling so as to decrease the likelihood of premature convergence, i.e. convergence to a local maximum of the fitness function. We chose a value of $a = 0.8$ for the weight of the complete wavefield in equation (2), and a value of $\epsilon = 0.2$ for equation (5). We assumed that the source signature is known. For the computation of the fitness function we used only 20 frequencies linearly spaced from 8 to 20 Hz, where our signal has its maximum values. Using more frequencies does not improve the results and increases the computation time.

Figures 2.5 (a) and (b) show the results after 300 generations. The ranges of slownesses and impedances that are displayed represent the corresponding search spaces. All the models in the last generation are displayed together using a graph-
Figure 2.5: Slowness (a) and impedance (b) models obtained in the first inversion run. The solid line represents the true model, and the dashed line represents the best model in the population. All the models of the last generation are displayed in a grey-scale as explained in the text.
binning technique similar to the one described by Basu & Frazer (1990). For all 50 models the occurrences of all possible values for each of the 22 parameters are counted and displayed in a grey-scale. Each possible value is displayed as a rectangle, whose width is the discretization interval of the slownesses and impedances, respectively. The darker a rectangle is, the more often that value is encountered. If a rectangle is black the parameter takes that value in every model. If it is white, that value is never taken. As noted above, these figures do not represent a PPD, because a GA does not sample from the PPD. The figures rather represent one particular set of good models and show the fitnesses of the models relative to each other within this set. Note that we do not weight the graphs by their fitnesses, as they are already selected by fitness. The solid line shows the true model parameters of the data. The dashed line shows the best model, i.e., the one with the highest fitness, in the last generation.

Some of the resulting slownesses are not far from the true values (layers 1, 4, 5, 7, 8, 9, and 10). Others, especially layer 11, are less well resolved. None of the impedances are resolved particularly well, and they also have converged to a lesser extent than the slownesses.

2.5.1 Reparameterization

What can we do to improve the results? If we just continue the inversion at the low temperature $T_2$ the population is likely to converge to the best model (the dashed lines in Figs. 2.5 (a) and (b)), and thus no further improvement is achieved. Our approach is to reparameterize the model and then to continue the GA inversion using the last generation from our previous run as a starting population.

It is well known that a GA works better if the model parameters are as independent of each other as possible. This is known as the principle of meaningful building blocks (Goldberg 1989, p. 80). By this principle one should select a coding that puts dependent parts of the model on genes that are close to each other on the chromosome. The principle is a consequence of the schema theorem, also called the
fundamental theorem of genetic algorithms, which forms the mathematical foundation of genetic algorithms. We do not discuss this theorem here, but refer the reader to Goldberg (1989, Chapter 2).

Our next task is to find model parameters that are less dependent on each other than slownesses and impedances. The model parameters that we will use are linear combinations of slownesses and impedances. Neglecting the roughness penalty function, for our idealized problem the fitness will be a maximum at the true model \( f(m_{\text{true}}) \), and as \( f \) is normalized to take values between 1 and \(-1\) its maximum is

\[
f(m_{\text{true}}) = 1.
\]  

Let \( m \) denote a model vector that is close to \( m_{\text{true}} \) and write

\[
m = m_{\text{true}} + \delta m.
\]  

Expanding \( f(m) \) in a Taylor series around the maximum \( m_{\text{true}} \) and neglecting terms of order higher than two gives

\[
f(m) - f(m_{\text{true}}) = -\frac{1}{2}\delta m^T H \delta m,
\]  

or with equation (7)

\[
f(m) - 1 = -\frac{1}{2}\delta m^T H \delta m.
\]  

Here \( H \) is the Hessian matrix, the matrix of the second derivatives of \( f \) with respect to the model parameters. The negative signs on the right hand sides of equations (9) and (10) make \( H \) positive definite. The first derivatives vanish, because \( f \) takes its maximum value at \( m_{\text{true}} \). We can write \( H \) as

\[
HW = WA,
\]  

21
where $\Lambda$ is the diagonal matrix of eigenvalues and $W$ is the matrix of normalized eigenvectors. Applying the coordinate transformations

$$m = W\tilde{m}, \quad \delta m = W\delta \tilde{m},$$

(12)

and substituting in equation (9) gives

$$f(W\tilde{m}) - 1 = -\frac{1}{2}\delta \tilde{m}^T \Lambda \delta \tilde{m}.$$  

(13)

The new model parameters $\tilde{m}$ are maximally independent in the vicinity of the true solution. Frazer & Basu (1993) suggest a similar transformation for use in simulated annealing except that they diagonalize a covariance matrix based on SA-generated samples from the PPD, while our Hessian is based on derivatives at a point.

Of course, in real problems the true solution $m_{true}$ is unavailable, and so our procedure is to compute $W$ for the best solution from the most recent inversion run. To obtain this matrix we first compute synthetic seismograms for our best model. Then we vary one model parameter by a small step, compute synthetics for this model and compute the fitness function from the correlation of these synthetics with the one for our best model. From this fitness function we obtain one element in the diagonal of $H$. We continue this procedure for all the parameters and obtain the complete diagonal of $H$. The off-diagonal elements are then obtained by varying two parameters at a time, and $W$ is obtained through diagonalizing $H$.

Note that we calculate $H$ and $W$ from ideal synthetic seismograms computed for our best model rather than from our data, so that equation (7) is always true, and $H$ is always positive definite. The above derivation would not be correct if $H$ was computed from the data, because $H$ would not be positive definite in that case, and thus the first order terms would have to be included in equation (9). For the same reason we omit the roughness penalty function of equation (4) when we compute the eigenvectors. The matrix $W$ transforms our model parameters to parameters that are nearly independent of each other in the vicinity of our best solution. The parameters
will generally not be nearly independent in the vicinity of the unknown true solution, but they will be less dependent than our original parameters if, as we assume, \( W \) is similar in both cases.

We can test this conjecture on \( W \) by comparing the \( W \) of our best model (dashed line in Fig. 2.5) to the \( W \) of the true model (solid line in Fig. 2.5). Figure 2.6 shows the eigenvectors for the true model (a) and for the best inversion model (b). The fitness function \( f \) that we used in equation (9) is given by equation (2) with \( a = 0.5 \), which is the value for \( a \) that we use in the inversion runs below. The eigenvectors are ordered from left to right in the order of increasing corresponding eigenvalues. The eigenvalues are given in Table 2.1.

Figures 2.6 (a) and (b) show the weight that each original parameter receives in each eigenvector. The 22 original parameters are ordered in the following way. The topmost 11 parameters are the slownesses of the layers, where the top parameter is the slowness of the topmost layer, and the 11th parameter is the slowness of the halfspace. The bottommost 11 parameters are the impedances ordered in the same way. As can be seen by comparing Figs. 2.6 (a) and (b) the eigenvectors are very similar for the true model and our best model. The eigenvectors corresponding to the five largest eigenvalues (eigenvectors 18 to 22 from the left) are almost identical in both cases. Most other eigenvectors are also nearly identical up to a sign change (eigenvector 2 and eigenvectors 5 to 13). A few eigenvectors (1,3,4,14 to 17) do not match quite as well; however, they are similar inasmuch as they still weight the same parameters relative to each other in a similar way.

Figures 2.6 (a) and (b) show that the eigenvectors that correspond to the largest 10 eigenvalues resolve the slownesses of the first 10 layers only. The slowness of the lower halfspace and all the impedances correspond to smaller eigenvalues. The largest eigenvalues resolve mainly the low wavenumber part of the slowness-depth function; the greatest eigenvalue corresponds mainly to the average of the first 8 slownesses. With decreasing eigenvalues the new parameters resolve the higher wavenumber in-
Figure 2.6: Eigenvectors computed for the true model (a) and the best model from the inversion (b). The figures show the relative weights that the slownesses and impedances receive in each eigenvector.
formation in the slowness profile, namely differences between groups of layers. For the impedance this behavior is opposite: The larger eigenvalues resolve mainly the high wavenumber part of the impedance profile. The parameter with the smallest eigenvalue is mainly the average of the impedances.

Comparison of the eigenvalues to each other (Table 2.1) shows that there is a large difference in magnitude between them. The smallest eigenvalue is negative for the true model, and for our best model the smallest two eigenvalues are negative. Ideally this should not be the case since the matrix \( H \) should be positive definite. It is a result of numerical inaccuracy in the diagonalization and is an indication that

<table>
<thead>
<tr>
<th>True Model</th>
<th>Best Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.0061</td>
<td>-0.0024</td>
</tr>
<tr>
<td>0.0001</td>
<td>-0.0001</td>
</tr>
<tr>
<td>0.0028</td>
<td>0.0021</td>
</tr>
<tr>
<td>0.0035</td>
<td>0.0026</td>
</tr>
<tr>
<td>0.0146</td>
<td>0.0112</td>
</tr>
<tr>
<td>0.0267</td>
<td>0.0176</td>
</tr>
<tr>
<td>0.0402</td>
<td>0.0289</td>
</tr>
<tr>
<td>0.0554</td>
<td>0.0436</td>
</tr>
<tr>
<td>0.0879</td>
<td>0.0601</td>
</tr>
<tr>
<td>0.1246</td>
<td>0.0847</td>
</tr>
<tr>
<td>0.1990</td>
<td>0.1337</td>
</tr>
<tr>
<td>0.2624</td>
<td>0.3338</td>
</tr>
<tr>
<td>0.3471</td>
<td>0.4336</td>
</tr>
<tr>
<td>2.7083</td>
<td>2.8172</td>
</tr>
<tr>
<td>2.7685</td>
<td>2.9609</td>
</tr>
<tr>
<td>3.1620</td>
<td>3.6633</td>
</tr>
<tr>
<td>3.7758</td>
<td>3.9118</td>
</tr>
<tr>
<td>5.0290</td>
<td>5.2784</td>
</tr>
<tr>
<td>13.6752</td>
<td>13.9914</td>
</tr>
<tr>
<td>19.2167</td>
<td>19.2157</td>
</tr>
<tr>
<td>27.4103</td>
<td>27.3522</td>
</tr>
<tr>
<td>291.6871</td>
<td>300.1709</td>
</tr>
</tbody>
</table>
Figure 2.7: After reparameterization it is usually necessary to increase the search space. We show a two-parameter case. The black square is the search space for the original parameters $m_1$ and $m_2$. The grey square is the search space for the transformed parameters $\tilde{m}_1$ and $\tilde{m}_2$.

The values are negligible compared to the larger eigenvalues. It can also be seen that the largest eigenvalue that defines impedances is a factor of $10^3$ smaller than the largest eigenvalue that defines slownesses. Parameters with larger eigenvalues are far more likely to be determined correctly in the inversion. For example we expect to obtain better results for most of the slownesses than for the impedances, which makes sense intuitively, as the former are determined through both traveltime and amplitude information whereas the latter are determined only through amplitude information.

In the inversion for the transformed model parameters the search space for the model parameters has to be increased, as sketched in Fig. 2.7 for the simplest case of two model parameters, $m_1$ and $m_2$. If we originally limit the search space to the black square in Fig. 2.7 then we have to increase it by the grey areas when we invert for the new model parameters $\tilde{m}_1$ and $\tilde{m}_2$. If we do not increase the search space we
will lose potential solutions of our original search space. Note that, even though the original parameters $m_1$ and $m_2$ in Fig. 2.7 were all positive, the inversion for the new parameters might actually result in solutions that correspond to negative $m_2$. This is an undesirable effect, for our model parameters are slownesses and impedances, for both of which negative values are physically meaningless. One way around this is to have the program check for models with negative values $s_i$ and $\eta_i$, and to assign the lowest possible fitness value to these models rather than computing their fitness. In our case negative values of the model parameters do not have to be excluded because they automatically result in low fitness values and will be eliminated in the selection process.

We obtain the bounds of the new search interval for the transformed parameters from the upper and lower bounds of the old parameters as

$$b_i = \sum_k (W^T)_{ik} a_k.$$  \hspace{1cm} (14)

Here, $W^T$ is the transpose of $W$, which is equal to its inverse, $b_i$ denotes the upper/lower bound of the $i$th transformed parameter, and $a_k$ is the upper/lower bound of the original parameter if $(W^T)_{ik}$ is positive and is the lower/upper bound if $(W^T)_{ik}$ is negative.

Even though we did not find it necessary to explicitly exclude negative values, we found that the inversion program works better if we apply another penalty function to models that lie outside of our original bounds. It especially prevents large shifts in the impedance-depth function which tend to occur without a penalty function due to the small eigenvalues that correspond to absolute impedance values.

### 2.5.2 Update algorithm

At this point we will also use an update algorithm, similar to that of Stoffa and Sen (1992). The purpose of this update algorithm is to give fit individuals a better chance to progress to the next generation, instead of being destroyed by crossover.
We compare each parent to one of their two children. The other parent is compared to the other child. If the child is fitter than the parent, we keep the child. If the parent is fitter, we keep the child with probability

\[ p_c = \exp\left(\frac{f_{\text{child}} - f_{\text{parent}}}{T_{up}}\right), \tag{15} \]

where \( f_{\text{child}} \) and \( f_{\text{parent}} \) are the fitness of the child and the parent, respectively, and \( T_{up} \) is a parameter that controls the probability of a child being replaced by its parent. The lower \( T_{up} \) the more likely it is that the parent is kept. We use a value of \( T_{up} = 0.1 \).

Our experience is that an update algorithm is always useful if the goal is to reach a high fitness value within a small number of generations, which was the goal of Stoffa & Sen (1991, 1992). In our problem we have not found it useful in our first inversion run, i.e. when we start from scratch, as it usually made our population converge too fast, giving much worse results than the ones displayed in Fig. 2.5. In subsequent steps, however, we find an update algorithm useful. Reasons for this are both our reparameterization and the fact that we do not select our first generation randomly but instead use the last generation from our previous run. Thus we start with a population whose average fitness is already high.

2.5.3 Inversion with new model parameters

The high fitness of our new starting population not only makes an update algorithm useful, but also enables us to do without a cooling schedule and use the constant low temperature \( T = 0.05 \) throughout the run. Furthermore, we use \( \epsilon = 0.1 \) instead of \( 0.2 \) in equation (5), and \( a = 0.5 \) in equation (2). The penalty function can be lowered because our starting population is already biased towards a non-oscillatory model. The new value for \( a \) gives more weight to the upgoing wavefield than the previous one, and leads to better final results then \( a = 0.8 \). In the noise-free case we have also tried to use \( a = 0.5 \) in our first run, and have obtained equally good results. The reason we used \( a = 0.8 \) is to be consistent with the case of noisy synthetics, discussed
Figure 2.8: Results of the second inversion run for slownesses (a) and impedances (b). They are displayed in the same way as in Fig. 2.5.
Figure 2.9: Eigenvectors computed for the best model in Fig. 2.8 and displayed in the same way as in Fig. 2.6.

For noisy synthetics a value of $a = 0.8$ works better in the first run.

For our second run we compute only 50 generations. Figure 2.8 shows the resulting slownesses and impedances. Note that the search space was actually larger than the one displayed in Fig. 2.8. Also the discretization interval is now different for each model parameter, and the possible values of the original parameters are not evenly spaced, because the inversion was done for the new parameters. The width of the boxes used for the graph binning does not represent the discretization interval of this inversion run, but the same interval as before. Comparison with Fig. 2.5 shows that the results have improved, especially for the impedances. The slownesses have slightly improved, with the exception of the last slowness, which as noted above, is not very well determined by the data.

Now we repeat the procedure: compute new eigenvectors and new parameters and then restart the inversion in the updated parameters. The GA control parameters
\(e, a, \text{ and } T\) are the same as in the previous run. The new eigenvectors are shown in Fig. 2.9, and the results of the inversion after 50 more generations are shown in Fig. 2.10. Comparison of Fig. 2.9 with Fig. 2.6 (a) shows that all eigenvectors, with the exception of eigenvectors 1, 3, and 4, match the eigenvectors for the true model very well. The results from the inversion have improved slightly for the slownesses and are about the same for the impedances. Subsequent inversion runs do not change the results significantly. We therefore stop the inversion here.

Figure 2.11 shows the results for the updated parameters. Eigenvalues increase from top to bottom of the figure. The width of the plot corresponds to the search space, and the solid line represents the true values of the parameters. The width of the rectangles corresponds to the discretization interval in this plot. It can be seen that all the parameters are resolved rather well. The majority of models take values that are within three steps of the true values.

Our algorithm is summarized in Fig. 2.12. First we perform an initial run with slownesses and impedances as model parameters. Then we do a series of reparameterizations and inversion runs for the new parameters. We stop the inversion when the results do not differ significantly from the previous ones.

\section*{2.6 INVERSION OF NOISY SYNTHETICS}

We now add random noise to the synthetics in Fig. 2.2. The resulting seismograms are displayed in Fig. 2.13. The noise level, which we define as the ratio of the rms of the noise to the maximum of the signal amplitude, is about 5\%. The upgoing wavefield obtained from the synthetics of Fig. 2.13 by f-k filtering is shown in Fig. 2.14. As the amplitudes of the upgoing waves are smaller than those of the direct waves, the signal-to-noise ratio is lower in Fig. 2.14 than in Fig. 2.13. The noise level for the upgoing wavefield is about 15\% by our definition. This means that the fitness \(f_u\) for the upgoing waves is much more susceptible to errors than the fitness \(f_c\) for the complete wavefield. For this reason we use a factor \(a = 0.8\) in equation (2) in the
Figure 2.10: Results after the third inversion run for slownesses (a) and impedances (b).
Figure 2.11: Results of Fig. 2.10 displayed for the parameters that were used in the inversion. The parameters correspond to increasing eigenvalues from top to bottom. The range displayed corresponds to the search interval for each parameter.
first inversion run, and \( a = 0.5 \) in the later runs.

Figures 2.15 (a) and (b) show the slownesses and impedances obtained from the first GA run using slowness and impedance parameters. The results are not very good. To improve them we proceed as in the noise-free case above, computing the eigenvalues and eigenvectors of \( W \) around our best solution. The eigenvalues are given in Table 2.2, and the eigenvectors are shown in Fig. 2.16. The eigenvectors do not match those of the true model (Fig. 2.6 (a)) as well as they did in the noise-free case, but the overall structure is still similar, and for some eigenvectors (e.g. the four eigenvectors with the largest eigenvalues) the match is quite good.

We restart the inversion with the new model parameters using \( T = 0.05 \) and \( a = 0.5 \). The results after 50 generations are displayed in Figs. 2.17 (a) and (b). Comparison with Fig. 2.15 shows that both the slownesses and the impedances have improved only slightly. However, if we keep reparameterizing and running the genetic algorithm as before we slowly get improvement. Figure 2.18 shows the eigenvectors
Noisy Synthetics

Figure 2.13: Synthetic data of Fig. 2.2 after 5% random noise was added.
Figure 2.14: Upgoing wavefield obtained from the data in Fig. 2.13 with f-k filtering.
Figure 2.15: Results for slownesses (a) and impedances (b) after the first inversion of the noisy synthetics.
Table 2.2: Eigenvalues of the eigenvectors in Fig. 2.16.

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.0008</td>
</tr>
<tr>
<td>0.0001</td>
</tr>
<tr>
<td>0.0015</td>
</tr>
<tr>
<td>0.0037</td>
</tr>
<tr>
<td>0.0112</td>
</tr>
<tr>
<td>0.0198</td>
</tr>
<tr>
<td>0.0331</td>
</tr>
<tr>
<td>0.0596</td>
</tr>
<tr>
<td>0.0707</td>
</tr>
<tr>
<td>0.0922</td>
</tr>
<tr>
<td>0.1513</td>
</tr>
<tr>
<td>0.2791</td>
</tr>
<tr>
<td>0.5539</td>
</tr>
<tr>
<td>1.7752</td>
</tr>
<tr>
<td>3.1112</td>
</tr>
<tr>
<td>3.4771</td>
</tr>
<tr>
<td>3.9736</td>
</tr>
<tr>
<td>5.2687</td>
</tr>
<tr>
<td>13.3228</td>
</tr>
<tr>
<td>19.2871</td>
</tr>
<tr>
<td>27.3509</td>
</tr>
<tr>
<td>299.2619</td>
</tr>
</tbody>
</table>

that are computed after four more runs with 50 generations each, and Figs. 2.19 (a) and (b) show the models obtained after another run using these parameters. The eigenvectors in Fig. 2.18 match the true eigenvectors much better than the ones in Fig. 2.16. Note that eigenvector 12 corresponds to eigenvector 11 in Fig. 2.6 (a) and vice versa, which is not surprising, as the eigenvalues are similar. The same is true for eigenvectors 14 and 15. The main discrepancy with Fig. 2.6 (a) is for eigenvectors 1, 3, and 4.

The slownesses (Fig. 2.19 (a)) are resolved rather well, and the impedances (Fig. 2.19 (b)) have improved. Overall the impedance differences are resolved better than the impedances themselves, which is what we expected from the eigenvectors.
Figure 2.16: Eigenvectors computed for the best model in Fig. 2.15.

Figure 2.20 shows the results for the new parameters. They are displayed in the same way as in Fig. 2.13. The figure shows that most new parameters are close to their true values. The main error is in the two smallest eigenvalues.

2.7 DISCUSSION AND CONCLUSIONS

The examples in this paper show that by running a series of inversions and reparameterizations we can get a model that is close to the true model even in the presence of noise. Subsequent reparameterizations counteract the tendency of the GA to converge to a suboptimal model. If in a simple GA the population has partly converged to a local fitness maximum the chances of further improvement become increasingly smaller. Our reparameterization enables the GA to search for better solutions even from a partly converged population. Because the new parameters are less dependent than the old ones, changing one new parameter (through crossover and mutation) is
Figure 2.17: Results for slownesses (a) and impedances (b) after the second run.
Figure 2.18: Eigenvectors computed for the best model after the 6th run.

much more likely to result in a higher fitness than changing one old parameter. New models that are created through crossover and mutation are therefore less likely to be rejected in the selection process if the inversion is done for the new parameters. As changing one new parameter corresponds to changing several old parameters simultaneously, it is rather easy for the GA to find new models in which the values of several original parameters are different from their values in the previous generation; thus it is easier for the GA to escape from a local fitness maximum. For example, in Fig. 2.5 (b) the true values of the impedances of layers 3 to 8 are not taken by any model in the population. After reparameterization and inversion for the new parameters, however, these values and values close to them are taken by most models in the resulting population (Fig. 2.8 (b)).

We have obtained results similar to the ones displayed in Figs. 2.10 and 2.19 with different sets of random numbers, indicating that they are largely independent of the starting population. This is illustrated in Fig. 2.21 where the best models
Figure 2.19: Results for slownesses (a) and impedances (b) after the 7th run.
Figure 2.20: Results of Fig. 2.19 displayed for the parameters used in the inversion. The display is as in Fig. 2.11.
from each of 20 independent runs for the noisy data are binned together. The dashed line represents the best model of Fig. 2.19. The slownesses (Fig. 2.21 (a)) more or less converge to the true values in all cases, with the exception of layer 11, noted above as not being well defined and, to a lesser extent, layer 10. Since the absolute values of the impedances are not well determined by the data, different independent inversion runs yield impedance functions that are shifted relative to each other but otherwise similar. As this is difficult to see in the grey-scale display used so far, we have displayed the impedances in the following way. First we calculated the mean impedance for each model and subtracted this value from the impedance values of all layers in that model. We then computed the mean of all mean impedances and added that value to every impedance value. The resulting display (Fig. 2.21 (b)) shows that the results for the impedances are also similar for different starting populations up to an additive constant.

Our algorithm makes it possible to work with a rather small population size corresponding to a small computation time. The optimum population size is different for different GAs, so that one should always experiment to find the optimum value. Generally the optimum population size is a function of the chromosome length $L_c$ in bits. Berg (1990) recommends a population size between $L_c/2$ and $L_c$ for his algorithm. For our algorithm we needed a population size of only $N_{pop} = 50$ for $L_c = 132$, which is smaller than the number recommended by Berg. A value $N_{pop} = 100$ gave no improvement over $N_{pop} = 50$, whereas smaller values gave worse results. Of course, for problems with more unknowns (larger $L_c$) $N_{pop}$ would have to be increased. We expect that $N_{pop}$ should be made a linear function of $L_c$.

Other parameters that have to be specified are the crossover probability $p_c$, the mutation probability $p_m$, the temperatures $T_1$, $T_2$, and $T_{up}$, the generation numbers $N_1$ and $N_2$ that define the linear cooling schedule, the weight of the complete wavefield in the fitness function $a$, and the factor $\epsilon$ for the penalty function. We have determined the values for most of these parameters by varying one parameter at a
Figure 2.21: The best slowness (a) and impedance (b) models from 20 independent inversions binned together. The results for the impedances (b) have been shifted, so that all 21 impedance profiles have the same mean value. The best model from Fig. 2.19 is shown as a dashed line.
time while keeping all the others fixed. The values that we have used in this paper are by no means the only possible choices, and they are most likely not the optimal choices either, since these parameters cannot be expected to be independent, and thus determining them one at a time does not enable us to find the optimum values. Thus it is almost certain that the results shown in this paper could be achieved in less computation time with a different choice of control parameters. Finding optimal GA control parameters is an optimization problem by itself. Grefenstette (1986) used another “metalevel” GA for the optimization of GA control parameters. This is of course rather time consuming and often not worth the effort, since the time spent on the optimization of the control parameters may exceed the time saved in the actual inversion with these parameters over an inversion with some suboptimal parameters.

We found that the exact values of some of the parameters are not all that important, and that good results are achieved as long as they are within a certain range. These parameters are $p_c$, $p_m$ and $T_{up}$. We obtained good results for $0.8 \leq p_c \leq 1$, $0.002 \leq p_m \leq 0.02$, and $0.05 \leq T_{up} \leq 0.5$. We do not expect that these parameters would have to be changed if the number of unknowns is increased.

The determination of $T_1$, $T_2$, $N_1$, and $N_2$ is a little trickier. The most important one of these parameters is $T_2$, because all the later GA runs are done with this temperature. In our examples the value of $T_2 = 0.05$ makes the maximum $g$ of equation (3) approximately twice as large as the average $g$ in the last generation of the first inversion run. This seems to stretch the fitness function enough to emphasize the differences in the fitnesses, so that selection becomes meaningful in the following runs. On the other hand, the fitness function appears not to be stretched too much, thus reducing the risk of the whole population converging to the best model. The parameters $T_1$, $N_1$, and $N_2$ are not that crucial for the result of the inversion, because good results are easily achieved as long as both $T_1$ and $N_2 - N_1$ are rather large, the latter meaning that the cooling is slow. It is likely that we could find values for these parameters that would speed up the inversion through reducing the number
of generations in the first run without changing the results. In fact, even the linear cooling that we use is probably not the most efficient cooling schedule. As mentioned above, we have not tried to optimize these parameters in terms of computation time. In order to improve them, one should probably carry out several test runs with different starting populations for each parameter combination. It is then always possible to tell if the cooling was too fast and/or $T_1$ was too low, because in either case different starting populations lead to different results. If similar results are obtained for different starting populations one can try to speed up the inversion by cooling faster or starting at a lower $T_1$.

The values of $a$ (in equation (2)) and $\epsilon$ (in equation (5)) are more crucial than those of the other control parameters, because the results are more sensitive to these values. As explained above, it was necessary to give the upgoing wavefield some extra weight to obtain the impedance profile from the data. Using $a = 1$ still gave good results for the slownesses, but poor results for the impedances. A fitness function with $a = 0.5$ was able to resolve both slownesses and impedances simultaneously. We also tried $a = 0.8$ but found it to be too high to recover the impedances. However a combination of $a = 0.8$ in the first run and $a = 0.5$ in all the later runs gave good results for noise-free and noisy synthetics. For the noise-free synthetics a value of $a = 0.5$ also worked well in the first run. For the noisy synthetics a value of $a = 0.5$ in the first run made more iterations necessary later on, but the end results were still good. As we have not tried $a = 0.5$ in the first run for different starting populations we cannot say that this is generally true. It is possible that on the average $a = 0.5$ works just as well as $a = 0.8$ in the first run. It seems to be rather clear, though, that $a$ should not be greater than 0.5 in the later runs. The value of $\epsilon$ in equation (5) has been determined by starting with $\epsilon = 0$ and then increasing it in subsequent test runs. Generally we want $\epsilon$ to be as low as possible. The main reason for introducing a penalty function is that only the average slowness of the layers between two receivers can be determined from the data, but not the individual slownesses. The effect of the
penalty function is to assign to each individual layer between two receivers a slowness value close to the average slowness. For the determination of $c$ one has to keep in mind the following. If $c$ is too low there will be oscillations in the slowness profile. If it is too high the inversion will try to model steps in the slowness profile by a more gradual transition, which means that one big step will be modeled by several smaller steps. The values $c = 0.2$ in the first run and $c = 0.1$ in the following runs seem to be a good choice for avoiding both of these problems for the type of model shown in this paper. We obtained equally good results with $c = 0.1$ in the first run but needed a few more iterations in that case. The impedances are not as susceptible to oscillations as the slownesses, and it might have not been necessary to use a penalty function for the impedances at all. We have done a few experiments where we applied the penalty function only to the slownesses and have obtained similar results to the ones presented here, but we have not tested this extensively with a large number of different starting populations.

We have applied our algorithm with the same set of control parameters to several different models, and it performed equally well for all of them. We note, however, that all these models had the same number of unknowns. If the algorithm is applied to a model with a different number of unknowns, $N_{pop}$ has to be changed. It is not unreasonable to assume that the values that we have determined for the other control parameters may still be good choices. There is however no guarantee for this. Thus, before the algorithm is applied to real data it is advisable to do some test runs with synthetic data for models of the same complexity to make sure the control parameters are adequate and to adjust them in case they are not.

Besides changing some of the control parameters there may be other ways to improve our algorithm, so that fewer iterations are needed to arrive at the final model. For instance, instead of using the best model in the last generation for reparameterization, we could use the best model found in all 50 generations. Also, doing the reparameterization after 50 generations is somewhat arbitrary. Doing the reparam-
eterization more often might also result in improvement. One factor that has to be
taken into account here is the relative cost of one reparameterization versus one GA
run. A reparameterization requires $N_m(N_m + 1)/2$ solutions to the forward problem
(where $N_m$ is the number of model parameters), one for each different element of the
symmetric matrix $H$. A GA run with $N_{gen}$ generations requires $N_{gen} \times N_{pop}$ solutions.
With $N_m = 22$, $N_{pop} = 50$, and $N_{gen} = 50$ one GA run takes about 10 times as long
as one reparameterization. On a Cray Y-MP the actual CPU time for a GA run with
these parameters is approximately 30 sec, and the CPU time for a reparameterization
is approximately 3 sec.

It is better for the efficiency of the algorithm if, from the first inversion run,
we obtain eigenvectors that are similar to those of the true model. Doing the repa­
parameterization too early makes the algorithm less efficient. This is because reparam­
eterization has two effects. First, the new parameters become more independent if
the eigenvectors are similar to the true ones, which, as we have shown, increases the
efficiency. At the same time, however, the original search intervals increase, as shown
in Fig. 2.7, so the computation slows down a little. If the eigenvectors are not similar
enough to the true ones the second effect may actually dominate, making the algo­

ithm less efficient than inversion for the original parameters. To obtain the first set
of usable new parameters it is essential to avoid premature convergence. Our method
of using a rather large number of generations, slow cooling, and no update works well
in this respect. Once we have found good new parameters it seems to be relatively
easy to obtain a good model through repeated reparameterization and GA runs.

It might be suggested that, after a good model has been found in the first
GA run, one might continue with a linearized inversion scheme rather than doing
reparameterization and repeated GA runs. However, there is a fundamental difference
between the two methods. Doing a GA with new parameters does not bias the results
by the model for which eigenvalues have been computed. The GA does not use any
derivative information for the new searches, and the search is still global. All we
do is introduce parameters that result in more efficiently coded chromosomes. The chromosomes will be more efficient if the eigenvectors of the best model and the true model are similar. In the problems we have discussed here we have found this to be true. Even if the eigenvalues were not similar the introduction of new parameters would not make convergence to a wrong model more likely. It is possible that the efficiency decreases, but this is only because the search intervals increase as described above, not because the new parameters introduce any bias to a particular model. Continuing with a linearized inversion, on the other hand, does introduce a bias and makes the search local. If the model that was obtained in the first GA represents some local maximum then a linearized scheme is likely to converge to that maximum. Note that even if the eigenvectors of the best model are similar to those of the true model the best model could still be closer to some local maximum than to the true model. This is because we compute the eigenvectors from the misfit of synthetics for the best model and synthetics for perturbed versions of that model. We do not compute the eigenvectors from the misfit of the synthetics and the data. The fact that the eigenvectors are similar merely tells us that similar model parameters are efficient in the vicinity of either model. It does not imply that a linearized inversion with the best model as a starting model would automatically converge to the true model, because there can be more peaks in the fitness function in the neighborhood of these models even if the eigenvectors are similar for both models. As we have not compared our method to a linearized inversion technique, we cannot rule out that the VSP problem is in fact linearizable after the first inversion run and that linearized techniques might be more efficient at this point for this particular problem. We feel, however, that since our method is generally less likely to converge to a local maximum than a subsequent linearized method, it is worthwhile to continue doing the GA inversion as long as subsequent GA runs give improvement. However, a linearized inversion might still be useful as a final step in the inversion that could be performed when GA runs do not improve the results anymore.
Figure 2.20 shows that for most models in the final population the parameters with large eigenvalues take values very close to their true values. The discrepancies between the true model and the models obtained from the inversion are mainly caused by poor determination of parameters with small eigenvalues (mainly the two parameters with the smallest eigenvalues in Fig. 2.20). This suggests that we can narrow the search intervals for a future GA inversion. For each new parameter we can specify a new search interval symmetrically around its value in our best model. As we can be confident that the parameters with large eigenvalues are already well resolved, we expect that for such a parameter a rather small search interval around the best solution will contain the true value of the parameter. On the other hand, we do not expect that a parameter with a small eigenvalue has taken a value close to its true value. For these parameters we should therefore use a larger search interval. This suggests that we should assign search intervals that increase with decreasing eigenvalue.

We could also get improvement if we keep most of the parameters fixed at their current values and invert only for the parameters with the smallest eigenvalues. For example, we might improve the results in Fig. 2.20 if we continue the inversion for just two parameters. To show how much improvement we would get if we had resolved the two parameters with the smallest eigenvalues, we set them to their correct values while leaving all other parameters unchanged. The resulting slowness and impedance profiles are shown as dashed lines in Figs. 2.22 (a) and (b). The solid lines represent the true model, again. Comparison with Figs. 2.19 (a) and (b) shows considerable improvement for the impedances and a small improvement for the slowness of the lower halfspace. Note that the impedance of the uppermost layer is outside our original search interval. This is not surprising, since the true impedance value was the first possible value that could be taken. Assigning new search intervals, as discussed above, would have enabled us to identify that our original search interval for this impedance should be shifted to a lower value.
Figure 2.22: Results for slownesses (a) and impedances (b) obtained if the parameters with the two smallest eigenvalues in Fig. 2.20 are set to their correct values.
Convergence of our GA/reparameterization technique means that further iterations do not significantly improve the fitness of the best model in each generation. After convergence is achieved the current parameters will also be maximally independent or nearly so. Such parameters are ideal for use with conventional inversion techniques; thus, as mentioned above, it may be cost effective to use a conventional technique to try to improve our best model, once the GA inversion stops giving improvement. For example, one could search along the (current parameter) coordinate axes in model space. A graphical display of the results of such a search, fitnesses versus coordinate value for each coordinate, would also have value as an indication of resolution in model space. Note that the length of a coordinate search interval should be larger for coordinates associated with smaller eigenvalues.

2.8 REFERENCES


physicists, Sixtieth Annual International Meeting and Exposition, Sept. 23-27,
San Francisco, California, Expanded Abstracts, II, 1123-1125.

1477.

Frazer, L. N. & Basu, A., 1993. Inversion by statistical physics with an application
to offset VSP, submitted to *Geophys. J. Int.*

evolution from Monte Carlo methods for strongly non-linear geophysical optimi-


Grefenstette, J. J., 1986. Optimization of control parameters for genetic algorithms,

Geophysical Exploration, Section I. Seismic Exploration, Vol. 14 A, Geophysical
Press, London - Amsterdam.

Jin, S. & Madariaga, R., 1993. Background velocity inversion with a genetic algo-


Chapter 3

Comparison of simulated annealing and genetic algorithms for a seismic inversion problem

3.1 SUMMARY

We apply both a simulated annealing (SA) algorithm and a genetic algorithm (GA) to vertical seismic profile (VSP) inversion. We invert for slownesses (inverse wavespeeds) and reflection coefficients. For a 45-layer synthetic problem, neither algorithm has trouble finding a solution close to the true model, and SA is somewhat faster than the GA. For a real-data problem of the same size the algorithms compare to each other similarly as for the synthetic problem. However, different inversion runs tend to converge to different solutions for both SA and GA. Therefore, even though both models have a similar potential to find good models, the method that is preferable is the one that is better suited for estimation of the a posteriori probability distribution. This method is clearly SA.

Key Words: genetic algorithms, global optimization, seismic inversion, simulated annealing, vertical seismic profile
3.2 INTRODUCTION

Global optimization methods have recently become very popular in non-linear geophysical inversion, mainly because of their promise of starting-model independence. The most popular of these techniques are simulated annealing (Kirkpatrick et al. 1983) and genetic algorithms (Holland 1975). Both methods have been applied to a large variety of problems in many different fields, a review of which would be beyond the scope of this paper. For an introduction to SA and examples of its applications see Laarhoven & van Aarts (1987) and Ingber (1993), for an introduction to theory and applications of GA see Goldberg (1989) and Davis (1991).

Some publications on SA and GA of interest to seismologists are mentioned in the following. SA was first used in seismology by Rothman (1985, 1986) for residual statics estimation. Among other things Rothman found the one-step method, usually called the heat-bath algorithm (Rebbi 1984, Creutz 1984), to be superior to the more widely used two-step method, usually called the Metropolis algorithm (Metropolis et al. 1953). Landa et al. (1989) used SA for the estimation of seismic background velocity. Mosegaard & Vestergaard (1991) developed a SA for seismic trace inversion, which worked well for synthetic data. Sen & Stoffa (1991) used SA for waveform inversion, and obtained good results for a synthetic problem. Nørmark & Mosegaard (1993) applied SA to residual statics estimation focusing on the development of a method to estimate an optimal cooling schedule. Mirkin et al. (1993) compared several cooling schedules for SA, also working on residual statics estimation. Basu & Frazer (1990) gave a method to rapidly determine the critical temperature in SA and performed the inversion at constant temperature for a VSP crosshole velocity determination. Frazer & Basu (1990, 1993) showed how SA, which has traditionally been used for maximizing the objective function, can also be used for the computation of statistical quantities like a posteriori probability distribution (PPD) and model covariance. To avoid confusion below note that in this paper, we use the term objective
function or fitness function to mean a function whose maximum is sought. Some writers use the term objective function synonymously with misfit function, which is a function whose minimum is sought.


Several comparisons for different GAs and SAs have been made, either for some mathematical test functions (e.g. Ingber & Rosen 1992), which are of less interest here, or more interestingly, for some real problems in seismic inversion. The latter ones tend to favor GA over SA. Scales et al. (1992) tested GA against SA for synthetic seismic inversion problems with 15, 22 and 30 unknown parameters. While they found both methods useful for the smaller sized problems they found GA clearly superior to SA for the 30-parameter problem. Gerstoft (1994) used GA for a synthetic seismic inversion problem and found it to be superior to a SA algorithm based on the method of Szu & Hartley (1987). Stoffa & Sen (1991) and Sambridge & Drijkoningen (1992), even though they did not compare GA and SA directly, hint that GA may be superior to SA for waveform inversion.

In spite of these studies suggesting the superiority of GA, SA has some definite
advantages over GA. First, it does not require the specification of as many control parameters as GA (population size, crossover probability, mutation probability, etc.). The determination of these parameters requires experimentation that makes GA less convenient to use, and requires additional computation time. Secondly, if one adopts the inversion philosophy of Tarantola & Valette (1982), then the goal of the inversion should be to compute the PPD in model space, rather than finding its global maximum. Here SA has a definite advantage over GA because it is possible in principle to estimate marginal PPDs for each of the model parameters with SA, something that cannot be done reliably with GA. Therefore, SA would still deserve consideration for PPD computations, even if GA were found to be better in finding the global maximum of the objective function.

In this paper we will apply both SA and GA to VSP inversions of both synthetic and real data. Our SA algorithm is of the heat-bath kind. For GA we use the algorithm described in NF94. We use the full wavefield, i.e. all multiple arrivals, for the inversion.

The paper is organized as follows. In section 3.3.1 we discuss the objective function used in both inversion algorithms. We use a function that does not require knowledge of the source wavelet, and is thus particularly well suited for the inversion of real data. Moreover, it does not require knowledge of the structure between the source and the topmost receiver. In section 3.3.2 we briefly describe the inversion algorithms.

In section 3.4 we perform inversions for a 45-layer synthetic problem. The purpose of using a synthetic problem first is that the results can be verified by comparing them to the true model. In this section we focus on finding a solution close to the global fitness maximum, which is supposedly the main strength of GA. We discuss the SA inversion in section 3.4.1 and the GA inversion in 3.4.2. In section 3.4.3 we compare the two methods and show that SA is not necessarily inferior to GA but can actually be better.
In section 3.5.1 we do the same comparison for a real-data problem. It then becomes apparent that non-uniqueness can be a serious problem in real-data cases, so that the estimation of the PPD (or the marginal PPDs of the model parameters) may be more important than finding individual good solutions. We show in section 3.5.2 how the marginal PPDs of the model parameters can be estimated from the SA results, which demonstrates the real advantage that SA has over GA.

In section 3.5.3 we compare our real-data results to physical properties measurements.

3.3 THEORY

3.3.1 Fitness function

We perform the forward modeling using a code based on the global matrix method (Schmidt & Tango 1986). This code operates in the frequency-wavenumber domain, i.e. it computes the response separately for each frequency. It is thus efficient to Fourier transform the data and to compare data and synthetics in the frequency domain, because then the synthetics (computed for many different models) do not need to be Fourier transformed.

We apply an approximate geometrical spreading correction to the data by multiplying them by the time, so that we have to compute synthetics for only one horizontal wavenumber \( k_h = 0 \), i.e., we compute only vertically traveling plane waves. Computations are thus rapid enough to permit experiments with different inversion schemes.

The goal of inversion is often to minimize the misfit between the measured and theoretical data. When people speak of the objective function they usually mean this misfit function. As mentioned in the introduction, we formulate our problem as the maximization of a function, called the fitness function, in conformance with GA terminology. We use this term for both GA and SA, even though people usually do
not speak of fitness when they are talking about SA.

Following NF94 we compute the contributions of up- and downgoing waves to the fitness separately. Here, however, we transform the synthetic wavefield $F(\omega, z)$ to $F(\omega, k_z)$ for each synthetic model generated in the inversion process and perform the fitness evaluation in the $\omega$-$k_z$ domain. This is different from NF94, where we performed $f$-$k$ filtering only on the data, and evaluated the fitness in the $\omega$-$z$ domain. The method we use here is somewhat more time consuming, but it leads to more accurate results, as the numerical artifacts due to the Fourier transform are the same in both data and synthetics. We restrict our computations to the acoustic case and use slownesses and reflection coefficients as model parameters. In NF94 we used slownesses and impedances. However, since the average impedance was not determined by the data we always had at least one model parameter that we could not determine. By using reflection coefficients rather than impedances we eliminate this one unknown parameter. Our model vectors are thus given as

$$m = \left( \begin{array}{c} q \\ r \end{array} \right),$$

where $q$ and $r$ are vectors containing slownesses and reflection coefficients, respectively.

The main contribution $f_1$ to the fitness $f$ of a model is computed as the sum of a term $f_d$ for the downgoing wavefield and a term $f_u$ for the upgoing wavefield:

$$f_1 = f(m) = \frac{1}{2} (f_d(s_+, d_+) + f_u(s_-, d_-)), \quad (2)$$

with

$$f_d(s_+, d_+) = \sum_{\omega} W(\omega) \frac{|s_+^*(\omega) d_+(\omega)|}{||s_+(\omega)|| ||d_+(\omega)||}, \quad (3)$$

$$f_u(s_-, d_-) = \sum_{\omega} W(\omega) \frac{|s_-^*(\omega) d_-(\omega)|}{||s_-(\omega)|| ||d_-(\omega)||}. \quad (4)$$
Here $d_+$ and $s_+$ are vectors that contain, for a given frequency, the amplitudes of the data and synthetics for the positive vertical wavenumbers, and $d_-$ and $s_-$ are vectors that contain the amplitudes for the negative vertical wavenumbers. Thus $d_{+/−}$ and $s_{+/−}$ are vectors with one component per vertical wavenumber. $\|\|$ denotes the $L_2$ norm and $\ast$ a complex conjugate. Different frequencies can be weighted differently through the weight function $W(ω)$, e.g. frequencies for which the signal-to-noise ratio is high can be given more weight. As we want to compare fitnesses later we normalize $W(ω)$ so that $\sum_ω W(ω) = 1$. With this normalization $f_1$ in eqn. (2) varies between 0 and 1.

Equations (3) and (4) have some advantages over the fitness function used in NF94 (eqn. 1 there). Most importantly, the source wavelet need not be known, as the contribution of each frequency is computed separately. Also, any structure between the source and the topmost receiver need not be known. This can be intuitively understood, by viewing the complete downgoing signal at the topmost receiver level as a source wavelet that is propagated through the layer stack. Hence, as the source wavelet need not be known, neither does the structure above the topmost receiver. (As this theoretical result may be thought surprising, we also verified it computationally). Moreover, the absolute amplitudes of data and synthetic do not have to match, i.e. $f_1(s_{+/−}, d_{+/−}) = f_1(\ast s_{+/−}, d_{+/−})$ which is obvious from eqns. (3) and (4). All of the above makes our fitness function particularly well suited for real-data inversion.

Following Basu & Frazer (1990) and NF94, we introduce a penalty function for oscillating slowness profiles. This reduces the potentially large number number of solutions to the the inverse problem by introducing some a priori information, namely that smooth slowness profiles are more likely than highly oscillating ones. Our penalty function is

$$p(q) = \frac{\sum_{i=2}^{N_z-1} |q_{i+1} - 2q_i + q_{i-1}|}{2(N_z - 2)(q_{max} - q_{min})},$$  

(5)
where \( q_i \) is the slowness of layer \( i \) and \( N_z \) is the number of layers. For any model \( m \), we then define the fitness \( f(m) = f(q, r) \) as

\[
f(q, r) = f_1(q, r) - \epsilon p(q).
\] (6)

The factor \( \epsilon \) that determines the relative weight of the two contribution is set to \( \epsilon = 0.2 \) in the following, as in NF94.

In SA one usually defines a quantity \( E(m) \) which is called "energy" in analogy with the statistical physics of a melt, on which SA is based. We define the energy \( E(m) \) as

\[
E(m) = -f(m).
\] (7)

By the same analogy one introduces a parameter \( T \), usually called "temperature" and defines the function

\[
\rho(m) = \exp \left( \frac{-E(m)}{T} \right) = \exp \left( \frac{f(m)}{T} \right).
\] (8)

The function \( \rho(m) \) is the (unnormalized) Gibbs distribution. In both our inversion algorithms \( \rho(m) \) is the function to be maximized.

In GA the term energy is usually not used, and people prefer to speak of fitness in analogy with natural evolution. When we mention fitness in the following, we mean \( f \) of eqn. (6). Strictly speaking, we would have to call \( \rho \) of eqn. (8) the fitness, as this is the function that the GA is trying to maximize. However, we want to talk about particular values of \( f \) later on and thus find it convenient to give a name to this quantity.

### 3.3.2 Inversion algorithms

Each model parameter can then take a certain number of discrete values in a search interval. For SA we use the heat-bath method which works as follows. Starting with
a random model, all model parameters are visited in turn. Each time a parameter \( m_k \) is visited, the value of \( \rho \) in eqn. (8) is computed for all possible values of \( m_k \), while all other parameters are kept fixed at their current values. A new value for \( m_k \) is then drawn from the distribution \( \rho \), and the next parameter is visited. After each parameter has been visited once, one "sweep" has been completed. The inversion then proceeds with another sweep, usually accompanied by lowering the temperature \( T \) in eqn. (8) according to some cooling schedule.

For a detailed description of our algorithm see NF94. In a simple GA each model is coded as a "chromosome", a string of binary numbers representing the values of the model parameters. The algorithm works with a "population" of chromosomes, i.e. it works with several models simultaneously. The models in the starting population are selected randomly. After initializing the population one computes the fitness (and \( \rho \) of eqn. (8) in our case) for every model in the population. A new population is then formed from the old one, in which each member of the old population receives a number of copies proportional to its fitness (proportional to \( \rho \) in our case). This step is called selection or reproduction. Then, with a certain probability, parts of the chromosomes are exchanged between pairs of chromosomes in the new population. This operation is called crossover. Either before or after crossover, with a small probability, binary digits of each string are changed from 1 to 0 or from 0 to 1. This is called mutation. After selection, crossover, and mutation, a new generation of models is obtained, and the algorithm proceeds as before.

### 3.4 A SYNTHETIC EXAMPLE

For a synthetic test of our inversion algorithms we use the model shown in Fig. 3.1. The depth region used for the inversion is between the two dashed lines. The seismograms computed for this model are shown in Fig. 3.2. We computed 45 traces with a receiver spacing of 9.114 m (30 ft).

For the inversion we parameterize the model with 45 layers, one layer for each
Figure 3.1: Model for the computation of the synthetic data.
Figure 3.2: Synthetic data computed for the model of Fig. 3.1 and used for the inversion.
receiver, and we invert for the slowness of each layer and the reflection coefficient at each interface. We therefore have 89 unknowns, 45 slownesses and 44 reflection coefficients. For each parameter we allow 32 possible values. The total size of the model space is thus $32^{89} \approx 9 \times 10^{133}$. We invert for slownesses ranging from 0.25 s km$^{-1}$ to 0.6 s km$^{-1}$ and for reflection coefficients between -0.25 and 0.25. We use 7 frequencies ranging from 15 to 70 Hz in eqns. (3) and (4).

In NF94 we used some simplifying assumptions that we will drop here in order to simulate a real-data case. Note especially the following.

- The synthetic data in Fig. 3.2 were computed with a slowness integration, i.e. for spherical waves (NF94 used plane waves). For the inversion the approximate spherical spreading correction as described above was then applied, to allow us to compute the synthetics only for vertically traveling plane waves, i.e. only for one horizontal wavenumber ($k_h = 0$).

- The layer boundaries of the true model do not coincide with the layer boundaries of the inversion models, as they would not coincide for real data. This means that the true model cannot be recovered exactly in the inversion.

- No assumptions about the source wavelet enter the inversions.

- In the true model there are reflectors below the bottommost receivers. The corresponding reflections can be seen in Fig. 3.2. This introduces some error in the inversion (mainly at greater depths), and it simulates a real-data case where reflections from lower depths might be present.

### 3.4.1 SA inversion

First we use the method of Basu & Frazer (1990) to determine a temperature $T$, which is presumably the critical temperature, which in the analogy to the cooling of a melt corresponds to the freezing temperature. We first perform a few short
inversion runs, each with the same set of random numbers but a different $T$ that is kept constant during each run. Then $T_*$ is defined as the temperature of the run having the highest average fitness. After several short runs of 20 sweeps each, we find $T_*$ to be approximately $5 \times 10^{-5}$.

Once we have determined $T_*$, we perform 30 sweeps starting at $T = 10T_*$ and ending at $T = T_*$ with a linear cooling schedule. We then keep running the inversion at $T_*$ until the fitness does not improve anymore, but keeps fluctuating about some value. Note that this cooling schedule is not very elaborate, and in no way optimal. In spite of this we achieve very good results after relatively few sweeps, and thus find it unnecessary to optimize our cooling schedule. The reason for this may be that we estimate the temperature $T_*$ before we do the actual inversion, which permits us to start at a relatively low temperature and thus avoid a long cooling process altogether.

The results of a typical SA inversion are displayed in Fig. 3.3. Figure 3.3 (a) shows the slownesses and Fig. 3.3 (b) the reflection coefficients. The solid line represents the true model, and the dashed line the best model obtained in the inversion. This model was obtained after 40 sweeps and its fitness is 0.9812, which is a rather high value (the maximum possible fitness is 1 minus some penalty).

The number of forward computations for a SA inversion is $N_s \times N_p \times N_v$, where $N_s$ is the number of sweeps, $N_p$ is the number of parameters and $N_v$ is the number of possible values for each parameter. The number of forward computations needed to produce the model shown in Fig. 3.3 is thus 113920 ($40 \times 89 \times 32$).

The match between the true model and the model obtained from the inversion is as good as can be expected. Note, that it is impossible to obtain the true model exactly, because the layer boundaries of the true model and the inversion model do not coincide.
Figure 3.3: SA results (dashed) and true values for slownesses (a) and reflection coefficients (b).
3.4.2 GA inversion

For the GA inversion we follow a strategy similar to NF94. First, we parameterize our model with slownesses and reflection coefficients. Once we reach the point where the fitness increase in successive generations gets very small, we do a reparameterization. The reparameterization finds new parameters that are more independent of each other than slownesses and reflection coefficients themselves. Using these parameters leads to a more efficient coding of the chromosome (see NF94 for details). The new parameters are obtained as linear combinations of the old ones, and they are computed from the eigenvectors in the neighborhood of the best model found so far. We then perform an inversion for these new unknown parameters followed by another reparameterization and so on, until the models do not improve significantly anymore.

The GA requires the specification of more control parameters, such as population size, crossover probability, and mutation probability, and thus requires more experimentation before the inversion can be done. For a detailed discussion of how the values for these parameters are obtained see NF94. We started with the values recommended in NF94, but we later obtained better results after changing some of them. This is an indication that the optimal values of the control parameters are, to a certain extent, always dependent on the problem, so that there seems to be no way around doing some experiments in order to find good values. This is one feature that makes GA less attractive than SA, as for the latter only the critical temperature and a cooling schedule have to be determined.

After some experiments we developed the following GA strategy. We chose the values $N_{\text{pop}} = 230$, $p_m = 0.005$, and $p_{\text{cross}} = 0.9$ for population size, mutation probability, and crossover probability, respectively. These are fixed during the inversion. We start with temperature $T = 0.05$. This is considerably higher than the critical SA temperature, but surprisingly we found it necessary to use higher temperatures in GA. We say surprisingly, because one might think that the optimal temperatures should be similar for GA and SA. However, starting the GA at a lower temperature
always made the population converge to some poor model very soon in our experiments. For example, when the inversion was started at $T = 0.005$, only about three different individuals were typically selected from the starting population which drastically limited further exploration of the model space. It might not be the case for a larger population size, but we do not like to have a larger population size because that would considerably increase the computation time. Another possibility is to use a lower temperature in combination with a higher mutation rate. We tried this, but it gave us worse results. The reason for this seems to be that with a combination of a low temperature and a high mutation rate the exploration of the model space is mainly done with mutation, while crossover becomes insignificant.

For these reasons we use $T = 0.05$ for the first 200 generations. We then lower $T$ to $T = 0.005$ and compute another 200 generations. Lowering $T$ any further resulted in poor performance in our experiments. We therefore keep the temperature at $T = 0.005$ for the rest of the inversion. In order to improve the performance, we use an update algorithm at this point, as described in NF94, and compute another 100 generations. We then do reparameterizations followed by inversion runs with 50 generations each until we get no improvements in subsequent runs. We then stop the inversion.

Typical GA results are shown in Fig. 3.4. They were obtained after a total of 700 generations and 4 reparameterizations. They are of about equal quality to the SA results of Fig. 3.3. The fitness of the model shown in Fig. 3.4 is 0.9786 which is about the same as the fitness obtained in the SA inversion.

The number of forward computations for the GA inversion is $N_g \times N_{pop}$, where $N_g$ is the number of generations. The number of forward computations needed for each reparameterization cycle is $N_p(N_p + 1)/2$ ($N_p$ is the number of parameters). The number of forward computations needed to compute the model in Fig. 3.4 is 177020, which is more than the number needed for the SA inversion.

71
Figure 3.4: GA results (dashed) and true values for slownesses (a) and reflection coefficients (b).
3.4.3 Comparison of SA and GA

Figure 3.5 shows the fitness versus the number of forward computations for GA (solid line) and SA (dashed line) for the examples discussed above. Clearly our SA algorithm is somewhat better than our GA, as the fitness increase in SA is faster. In the GA curve there are two points where the slope increases rapidly. One is at $N_e = 46000$ where we lower the temperature to $T = 0.005$ and the other (somewhat less pronounced) at $N_e = 115000$ where we start inverting for new parameters. The slope change at $N_e = 46000$ is rather dramatic, but the curve flattens out again soon after that. This behavior illustrates one of the main problems of GA. If in one way or the other the fitness increases too rapidly it tends to converge to some value and remain stuck there soon afterwards. For example, if we use a lower temperature early on to speed up the inversion in the beginning, our fitness never gets as high as in this example. It is really an art to balance the inversion between a too rapid and a too
slow fitness increase, since the fitness increase depends on the interaction of all the GA control parameters. The SA inversion does not require much effort to maintain the middle way between these two extremes, which is apparent in Fig. 3.5. Even though we spent hardly any effort on the optimization of the algorithm the fitness goes rather smoothly to a very high value.

Similar results are obtained for both GA and SA with different random numbers. In most cases models similar to the ones in Figs. 3.3 and 3.4 are obtained, and approximately the same number of forward computations as in the examples above are needed for SA and GA. Occasionally however, both algorithms get trapped in some suboptimal solutions. Thus, even though the inversions are largely independent of the random numbers, we cannot claim that they are completely independent. This is in contrast to what we found in NF94. The reason that we did not encounter this problem there was probably that the inverse problem of NF94 was idealized in several ways. For example, the layer boundaries of the true and the inversion models coincided, and we inverted synthetic data that were computed for plane waves. Moreover, the model was considerably smaller (11 layers). There we also used a different objective function that required knowledge of the source wavelet, i.e. we assumed more information to be known beforehand than we do here. All of the above may have contributed to making the global fitness maximum considerably more pronounced than the local ones and thus to making it easier to find the global maximum.

3.5 INVERSION OF REAL DATA

3.5.1 Comparison of SA and GA

Having verified that both algorithms work very well for synthetic problems, we now use them for a real-data inversion. Our data are VSP seismograms from Ocean Drilling Program hole 792E. The dataset is described in Cooper et al. (1992). We
use the recordings from 45 depth levels from 448 m below seafloor (mbsf) to 850 mbsf, as shown in Fig. 3.6. The receiver spacing is 9.114 m (30 ft). We use the same frequencies and search intervals as in the synthetic problem.

We follow the same inversion strategies as above. For SA we find $T_0 = 5 \times 10^{-5}$, the same as for the synthetic problem. We again run 30 sweeps with cooling from $T = 10T_0$ to $T = T_0$, and then keep running the inversion at $T = T_0$ until the fitness does not improve anymore. Figure 3.7 shows the best models from 3 independent SA runs with 60 sweeps each. The fitnesses of all three models are approximately the same (0.8545, 0.8552, and 0.8546). Two of the models are rather similar. The third one (the one represented by the fine-dashed line), however, is quite different. This indicates that there seems to be no dominant peak in the fitness function, rather several peaks of approximately equal height.

Figure 3.8 shows the results of three independent GA inversions. The maximum fitnesses are 0.8511, 0.8503 and 0.8421. The three models are again different from each other. By comparing Fig. 3.8 to Fig. 3.7 it can be seen that the models represented by the fine-dashed lines in both figures are similar to each other, and that the model represented by the solid line in Fig. 3.8 is rather similar to the other two models in Fig. 3.7. The third GA model differs from both of these models.

Even though some of the GA models are similar to some of the SA models, the fitnesses of the GA models are consistently lower. This reflects one advantage that SA (at least the heat-bath algorithm) has over GA. In the later stages of the inversion, when fitnesses are generally very high, the chances for improvement in GA become small, as most of the mutations and crossovers do not lead to improvement and are therefore eliminated through selection. In a heat-bath algorithm, on the other hand, for each parameter the fitness distribution of all possible values is computed each time the parameter is visited, which results in a greater chance of improvement in the later stages of the inversion.

Figure 3.9 shows a plot of fitness versus number of forward calculations for the
Figure 3.6: Measured data used for the inversion. The depths are given in meters below seafloor (mbsf).
Figure 3.7: SA results for slownesses (a) and reflection coefficients (b) for three independent inversion runs.
Figure 3.8: GA results for slownesses (a) and reflection coefficients (b) for three independent inversion runs.
inversions that resulted in the models represented by the solid lines in Figs. 3.7 and 3.8. As in the synthetic case, our SA algorithm is somewhat better than our GA, inasmuch as it finds better models faster.

3.5.2 PPD estimation

As the results of the previous section have shown, there are several possible solutions to our real-data inverse problem, all of which tend to have very similar fitnesses. In cases like this, knowing one of the fitness maxima is less important than knowing the PPD in model space. What we would really like, therefore, is to compute the PPD or, since it is impossible to visualize a multidimensional function like the PPD, to compute the marginal PPDs of the model parameters. The marginal PPD \( \sigma_k \) of model parameter \( m_k \) is
where $\sigma(m)$ is the PPD of $m$. The sums in eqn. (9) are taken for each model parameter over all possible values of that parameter.

These marginal PPDs can be estimated by graph-binning the results from sweeps at a constant temperature $T = T_c$ (Basu & Frazer 1990, Frazer & Basu 1993) if we assume that $\sigma(m)$ is given by the Gibbs distribution of eqn.(8) with $T = T_c$, which is the distribution from which SA samples. Ideally $T_c$ should be the critical temperature $T_{crit}$. If $T_c$ is higher than $T_{crit}$ the marginal PPDs will tend to be broader, so we will lose some of the resolution that we would achieve at $T_{crit}$. If $T_c$ is too low a large number of sweeps might be needed to obtain equilibrium, and some of the local fitness maxima may not be sampled. In the extreme case of $T_c = 0$ the distributions are delta functions, and computing these distributions becomes the same problem as finding the global fitness maximum.

The results that we obtained at $T = 5 \times 10^{-8} = T_*$, our earlier estimate for the critical temperature, have a tendency to not change significantly any more once they have converged to the models shown in Fig. 3.7. For this reason PPD estimations that we obtained from sweeps at $T = T_*$ were dependent on the random numbers. This behavior indicates that $T_*$ is lower than $T_{crit}$, as there should be a reasonable chance to escape local maxima at $T = T_{crit}$. It could also indicate, and in a way this is more likely, that there is not such a thing as one critical temperature in our problem. Rather, each parameter may have its own $T_{crit}$ due to different sensitivities of different parameters. For cases like this multiple-temperature SA has been suggested (Kuperman et al. 1990, Frazer & Basu 1993). In our case, $T_*$ may be the lowest of the different critical temperatures; then most parameters would have $T_{crit}$ higher than $T_*$, so sweeping at $T = T_*$ would make it very difficult for them to escape local maxima.

We will not investigate multiple-temperature SA here, since estimating the var-
ious critical temperatures is another problem, the discussion of which is beyond the scope of this paper. We like to note, however, that techniques for doing this exist (Frazer & Basu 1993).

Rather than using multiple temperatures, we will sacrifice some of the resolution that could be obtained in principle, and sweep at a single temperature, that we want to be the highest of the various critical temperatures. We tried the two temperatures $T_c = 1 \times 10^{-4}$ and $T_c = 5 \times 10^{-4}$. At $T_c = 1 \times 10^{-4}$ the inversion still tended to stick at some particular model, and we therefore settled for $T_c = 5 \times 10^{-4} = 10T_\ast$. All the results shown below were computed with this temperature.

Figure 3.10 is a grey-scale display of the marginal PPDs of the model parameters that were obtained from three inversion runs with 600 sweeps each. For each model parameter, each possible value is displayed as a rectangle whose darkness is a measure of the number of occurrences of that value in all sweeps. It is thus proportional to the marginal PPD of the model parameter provided the number of sweeps is high enough. The dashed line represents the model with the highest fitness found in the 1800 sweeps. The solid line shows the mean model $\langle m \rangle$ computed from the PPD. Its $k$-th parameter $\langle m_k \rangle$ is computed as

$$\langle m_k \rangle = \sum_{m_k} m_k \sigma_k(m_k).$$  \hspace{1cm} (10)

If the results in Fig. 3.10 were reliable estimates of the marginal PPDs they should be independent of the random numbers used for their computation. We test this in two different ways. First we make a display like Fig. 3.10 from 6 more inversions with 200 sweeps each. They are shown in Fig. 3.11. Again, the dashed line shows the best model and the solid line the mean model. Comparison of Figs. 3.10 and 3.11 shows that both the marginal PPDs and the mean model parameters computed from them are very similar, even though they have been obtained totally independently.

For our second test we try to construct the marginal PPDs from a single inversion run with 1000 sweeps. The results are shown in Fig. 3.12. Unfortunately, the
Figure 3.10: Marginal PPDs of slownesses (a) and reflection coefficients (b) obtained from graph-binning of three independent runs with 600 sweeps each. The solid line shows the average model, and the dashed line the best model.
Figure 3.11: Marginal PPDs of slownesses (a) and reflection coefficients (b) obtained from graph-binning of 6 independent runs with 200 sweeps each. The solid line shows the average model, and the dashed line the best model.
Figure 3.12: Marginal PPDs of slownesses (a) and reflection coefficients (b) obtained from graph-binning of 1 run with 1000 sweeps each. The solid line shows the average model, and the dashed line the best model.
agreement is not as good as before. For example, the marginal PPD of the reflection coefficient at the interface at a depth of approximately 0.5 km below seafloor (kmbsf) has a rather narrow peak at approximately -0.2 in Fig. 3.12 (b), whereas the PPD is a lot wider in Figs. 3.10 (b) and 3.11 (b). Actually, the values of the best models and the mean models in Figs. 3.10 (b) and 3.11 (b) are values of low probability in Fig. 3.12 (b). It seems like this parameter has more or less converged to one value in Fig. 3.12 (b) and reaching other values seems difficult for it. We can take this as an indication that for this parameter the critical temperature is probably higher than our \( T_c \).

In order to compute totally reliable marginal PPDs there may be no alternative using multiple temperatures. We think, however, that the results shown in Figs. 3.10 and 3.11 are good estimates of the marginal PPDs, because they were obtained completely independently and are very similar in spite of that. The mean models obtained from the marginal PPDs of Figs. 3.10 and 3.11 are also very similar. This is shown in Fig. 3.13, where the dashed line shows the mean parameters of Fig. 3.10 and the solid line the mean parameters of Fig. 3.11.

### 3.5.3 Comparison with physical properties measurements

In Fig. 3.14 the marginal PPDs of the slownesses from Fig. 3.11 are compared to the physical properties measurements (solid line). The dashed line represents the mean slownesses of Fig. 3.11. The agreement is rather good for most depths. At a depth of about 0.8 kmbsf the physical properties measurements indicate very high velocities (low slownesses), which mark the top of the basement. These high values are outside of the search interval that we used for the inversion. This is, however, not the reason that we did not recover them. We have performed some inversion runs both with shifted and with increased search intervals for the lower depths, but that did not change our results. However, this discrepancy between seismic and laboratory velocity measurements for the top of the oceanic basement is commonly observed and
Figure 3.13: Average slowness (a) and reflection coefficient (b) models from Fig. 3.11 (solid) and Fig. 3.10 (dashed).
Figure 3.14: Marginal PPDs and average slowness model (dashed) of Fig. 3.11 (a) compared to physical properties measurements.
has been explained by large-scale porosity (Hyndman & Drury 1976, Schreiber & Fox 1976). *In situ* velocity measurements for these depths (see Fig. 8 of Cooper et al. 1992) are actually closer to our results than to the physical properties measurements.

### 3.6 DISCUSSION AND CONCLUSIONS

We showed that, for a synthetic 45-layer problem, both our SA and our GA were able to recover slowness and reflection-coefficient profiles close to the true ones, without the need of a good starting model. Moreover, our choice of the objective function enabled us to obtain these profiles without knowledge of the source wavelet. These results demonstrate the power of both inversion algorithms.

In both cases our SA was somewhat faster than our GA, and the highest fitnesses obtained with SA were slightly higher. This does not mean that SA is generally superior to GA in finding the global fitness maximum, as there are many possible ways of writing both a SA and a GA, and the efficiency of each algorithm may also depend on the problem to be solved. We do not claim that our GA is the best possible one, and there are certainly many ways of improving it. However, we think our results show that SA can compete with GA, and is not necessarily inferior as is sometimes assumed. Our SA, of course, is not the best possible SA, either. For example, the algorithm could be sped up by adopting a more elaborate cooling schedule, a subject to which many people have devoted a great deal of research. As a matter of fact, we spent a lot less effort on the development and optimization of our SA compared to our GA. Our GA only performed as well as it did, because we used the rather elaborate method of reparameterization, while the SA algorithm is much simpler. Therefore SA may have more potential for improvement than the GA.

In the real-data case there seem to be a whole series of possible solutions, and for both SA and GA different inversion runs with different sets of random numbers tend to converge to different models. When non-uniqueness is encountered to such a degree, the goal of finding the global fitness maximum becomes questionable, and it
is much more useful to estimate the PPD in model space, or the marginal PPDs of
the model parameters.

We showed how this can be done by graph-binning of many SA results obtained
at constant temperature. This is justified if the Gibbs distribution of eqn. (8) is
identified with the PPD. GA, on the other hand, should not be used for this purpose,
as a GA does not sample from the PPD, and any estimate of the PPD that was
obtained from graph-binning GA results, would among other things depend on the
number of generations used for the PPD estimate.

In order to obtain the PPDs with SA graph-binning in any realistic time, the
temperature must not be lower than the critical temperature. We found that the
method of Basu & Frazer (1990) of rapidly estimating the critical temperature gave us
a temperature that was too low, and we used a temperature that was higher by a factor
of 10. With this temperature we obtained very similar results in two independent PPD
computations which we believe to be good estimates for that reason. The attempt to
obtain the same marginal PPDs in a single inversion run was not totally successful.
We suggested that the reason for this might be that different parameters have different
critical temperatures, and we suggested multiple-temperature SA as a possible way
of improving the estimates.

We finally showed that our results agree reasonably well with physical properties
measurements.

3.7 REFERENCES

Basu, A. & Frazer, L. N., 1990. Rapid determination of the critical temperature in
simulated annealing inversion, Science, 249, 1409-1412.

data, Society of Exploration Geophysicists, Sixtieth Annual International


