

# **Inversion of Teleseismic Receiver Functions via Evolutionary Programming**

Chris Lynch  
September 20, 1999

Southern California Earthquake Center  
1999 Summer Internship

Advisor  
Dr. Steven Day  
Department of Geological Sciences  
San Diego State University

## Table of Contents

Introduction	3
Teleseismic Receiver Functions	4
Inverse Geophysical Modeling	5
Evolutionary Programming	6
Approach and Goals	9
Results	10
Verification	10
Synthetic Data Study	11
Conclusions	19
Acknowledgements	21
References	22

## Introduction

The inversion of teleseismic receiver functions is a seismological method that can be used to help constrain estimates of the velocity structure of the Earth's crust. Evolutionary Programming (EP) is a global optimization technique that uses a simplified model of the evolutionary process to search a solution space for the optimal solution, often referred to as the global minimum. This study explores the use of evolutionary programming techniques for the inversion of teleseismic receiver function data. Application of this technique in Southern California may contribute to The Southern California Earthquake Center's (SCEC) goal of developing a master model of seismic hazard through the determination of seismic wave velocities. Better models of seismic wave velocities can then be used to create theoretical seismograms, which are useful in the prediction of ground motion during specific potential earthquakes.

A Matlab code was developed in 1998 by Charles Hoelzer at San Diego State University (SDSU) based on work by Bernard Minster and Nadya Williams (UCSD). That code combines evolutionary programming techniques with a simulated annealing-type cost function to produce an evolving family of models of seismic velocity structure. The models are initially distributed randomly over the solution space. Synthetic receiver functions computed from the models are compared with receiver function data and are rated according to some measure of fitness. The best fitting half of the population is then selected. The program then perturbs the parameters of each parent model to generate an offspring, producing a subsequent generation of the model population. Stopping conditions can be a specified fitness tolerance or an arbitrary number of iterations.

Over the course of two semesters (Fall 1998, Spring 1999), I combined my interests in geology and computer science by investigating and developing a more efficient and faster code. This work was done under the guidance of Steven Day, (Geology, SDSU), later joined by Richard Frost, (Mathematics and Computer Science, SDSU), and produced a translation into Fortran 90 of the original Matlab code. This Fortran 90 program is named Midcrust. The goal of this internship has been to extend this project by improving the efficiency of the code and the scientific techniques it employs.

#### Teleseismic Receiver Functions

Teleseismic waves are seismic waves recorded at distances greater than 1600km ( $16^\circ$  of arc) from an earthquake's epicenter. The first arrival is a refracted P wave (P or PKP). Teleseismic P waves that are incident upon a crustal boundary below a receiver station yield P to S wave conversions and multiple reverberations within the shallow layering. By treating the vertical component of a teleseismic seismic signal as a source function, and deconvolving from the horizontal component, S wave conversions and reverberations can be isolated. This is because the horizontal components contain most of the energy of the S wave conversions. This technique removes the obscuring effects of source function and instrument response and is called a receiver function. These receiver functions can then be inverted, producing a model of the shear velocity structure. This provides a means for determining crustal velocity structure. This method can be stable for regions where the layers are nearly horizontal [Ammon et al., 1990, Lay and Wallace, 1995]. The information gained by this method spans a region around the receiver station, the radius of which is approximately equal to the depth of the deepest reflector, usually the Moho [Ammon et al., 1990].

## Inverse Geophysical Modeling

Forward modeling takes model parameters as input, applies them to a mathematical model, and generates predicted (measurable) data. Inverse modeling takes observed data as input and estimates model parameters from it.

Midcrust represents crustal structure with a set of horizontal layers with four parameters per layer, P and S wave velocities, layer thickness and density. In this implementation only layer thicknesses for P wave velocities ( $\alpha$ ) are independently estimated. S wave velocities ( $\beta$ ) are calculated from P wave velocities using the equation,

$$\beta = \alpha \sqrt{0.5 - \sigma / 1 - \sigma}$$

and density is estimated using Birch's Law, with Poisson's ratio estimated at 0.25 for average crystalline rock [Nava and Brune, 1982]. The number of layers in the models is a user-defined parameter, as are a set of a priori layer velocity and thickness constraints.

For the inversion of teleseismic receiver functions, in Midcrust, forward modeling is required to make receiver function time series in order to evaluate model fitness. This is synthetic data, or the data that would be observed for a crustal structure defined by models parameters. Ammon et al, 1990, designed the code used for this operation. The result of inverting a teleseismic receiver function is a P wave velocity structure for the region under the receiver. Midcrust manufactures a population of these velocity structure models and evolves them in search of the global minimum, the optimal solution for the reference receiver function.

## Evolutionary Programming

Evolutionary programming attempts to mimic the process by which species evolve. EP was first developed by Fogel (1962). The algorithm used in Midcrust was first implemented by Hoelzer (1998) for inversion of receiver functions, and is based upon work by Minster et al. (1995), and Groot-Hedlin and Vernon, (1998). I have made some minor modifications in the course of translating the original Matlab code into Fortran 90. This algorithm begins with a receiver function and proceeds as follows:

- (1) Produce a population of velocity/thickness models randomly distributed over a chosen solution space,
- (2) Calculate the fitness of each model by comparing the receiver function data with a theoretical receiver function calculated for the model.
- (3) Compete each model against a given number of randomly chosen members of the population and rank them according to a cost function,
- (4) Select the half of the population with the highest score to serve as the parental models,
- (5) Generate an offspring from each parent model by perturbing the parameters of a copy of the parent,
- (6) Iterate through (2) - (6), new population consisting of parents and offspring,
- (7) Quit after the desired number of iterations have been completed.

The initial population is created by generating N (e.g., 100) random velocity-thickness models. Each model is created by selecting a uniform random number between specified upper and lower bounds for each parameter. Bounds are selected to be reasonable constraints for the velocity and thickness of each model layer based on general seismic knowledge.

Fitness is calculated using an L1 norm, or mean absolute value of the misfit between the observed (reference) seismogram and the seismogram generated through the forward modeling of the predicted (model) parameters. This value can be described as the amount of misfit between the reference and predicted model, where a smaller value represents a closer fit.

$$misfit = \frac{\sum_{i=1}^N |observed(i) - spredicted(i)| w(j)}{\sum_{i=1}^N |observed(i) w(i)|}$$

A weighting function,  $w(j)$  is used to place greater significance on arrival time within a specified time period. To date no weighting has been used giving the entire time series equal significance and so  $w(j) = 1$ . When synthetic data with noise, and later when real data is used for the reference receiver function the weighting function can be used to place emphasis on, for example, the first arrivals of the P to S wave conversions or on the later arriving Moho free surface multiples.

Selection is done via competition between models where each member of the model population is rated based on a comparison of fitness values between itself and a specified number of different models randomly chosen from the total population. This is analogous to competition between members of a population. Competition is implemented by first calculating:  $\Delta m = (misfit[a] - misfit[b]) / misfit[a]$ , where "a" is index of current model and b is a randomly generated index for the competitor. If  $\Delta m \leq 1$  then model[a] has its score increased by one, favoring a downhill move. This means that model a is accepted unconditionally and favors finding the local minimum. If  $\Delta m > 1$  then it will be unconditionally accepted, It will be conditionally accepted if  $\exp(-\Delta m / Temp) > R$ , where R is a random number between 0 and 1 and Temp is a geometrically decreasing value that governs acceptance of uphill moves. This is an uphill move, which provides a way to break out of a local minimum and continue to explore the solution space. Temp controls the probability of accepting an uphill move and is a specified function of iteration number. This function is referred to as the cooling schedule .

Adding a perturbation value to each parent parameter makes the offspring of the population. The perturbation is calculated by multiplying the misfit value of the parent by one fifth of the difference of the upper and lower bounds of the parameter, and a normally distributed random number with zero mean and standard deviation of one. Any offspring that are outside of the a priori bounds are reset to the value of the bound that has been crossed.

The Midcrust algorithm is being tested on synthetic data computed from a known layered crustal model. We refer to this test model as our reference model to distinguish it from the models generated randomly during the solution process. The reference model used for most of the testing of Midcrust is a four-layer model that was used as the standard reference model by Hoelzer (1998) to test the original Matlab implementation. It is a challenging test case because it has a small velocity contrast between layers 2 and 3, and a low velocity zone (LVZ) just above the crust-mantle boundary, or Mohorovicic discontinuity (Moho), which is a difficult solution to find. One of the advantages of Evolutionary Programming is the ability to find unexpected model solutions, being free from the necessity of having a close starting model in order to find a LVZ, as in linear inversion methods [Groot-Hedlin and Vernon, 1998].

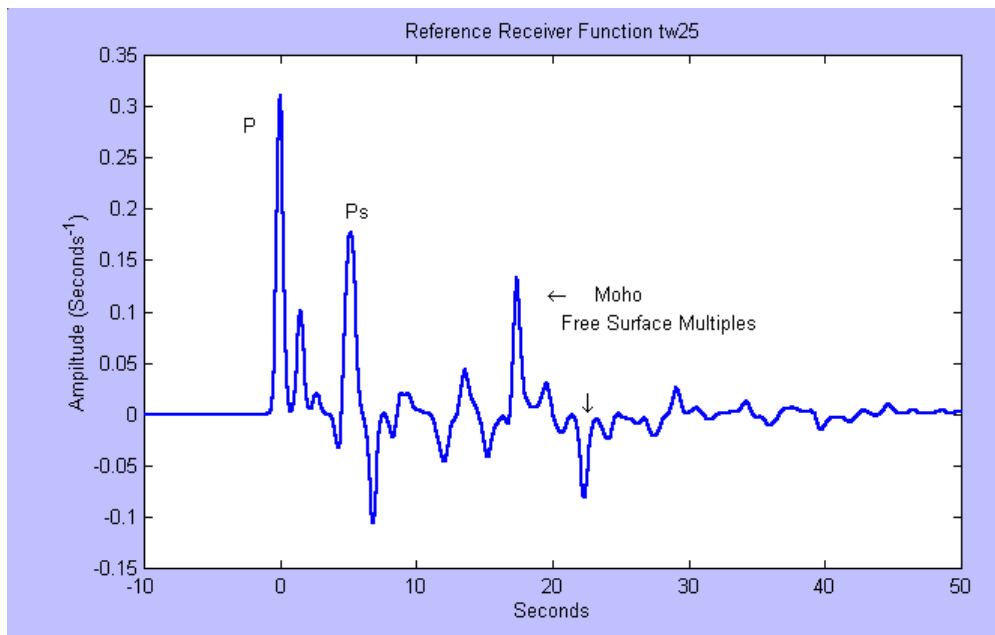


Figure 1. Reference Model Receiver Function

## Approach and Goals

The first step of this project was to complete development of the uni-processor implementation of the original Matlab driver into Fortran 90. This code is used in combination with a collection of Fortran 77 seismic solvers and their helper subroutines that were used in the original Matlab code. These perform the forward modeling and other required computational tasks. Then the ability of the Fortran 90 code to drive these solvers correctly was verified. After verification, a synthetic data study was initiated to evaluate the quality of the model solutions. The data study was designed to gain an understanding of the effects of the main evolutionary parameters and how they affect the final population. This in turn would enable tuning the code to produce optimal solutions. Next, a parallel implementation would be undertaken to provide a much more efficient program, thereby allowing a greater number of runs to be conducted within a given period of time, and allowing larger sized problems to be solved. An additional study objective is to test the code's ability to generate good solutions given known synthetic data with different levels of noise added.

Accomplishing the above would provide the starting point for studies with real data, generating possible solutions to crustal structure in the Peninsular Ranges Batholith (PRB) in Southern California. These solutions can then be compared with solutions determined from other methodologies including seismic tomography, magnetotellurics, reflective seismology, and field geology. Success in these goals along with steady improvements to the scientific techniques used in the code should yield a production level code. This code can then be used in projects like SCEC's Master Seismic Hazard Model of the Los Angeles Basin by helping to determine seismic wave velocities.

## Results

The uni-processor code has been completed and verified to drive the F77 solvers correctly. This verification consisted of producing a variant of the Midcrust code to output a synthetic receiver function time series from a set of model parameters as input.

A substantial decrease in processing time has been achieved. A normal run of 100 iterations, 100 models and competition with 20 different models takes around an hour compared with between 2 and 3 hours for the original code.

A preliminary study with synthetic data has been conducted. The results of this study have been used to do discover errors in the implementation and to start tuning the Midcrust application, and have provided several opportunities for improvements to the evolutionary technique.

## Verification

Objective: To verify that Midcrust produces correct forward models for input data. This would prove that the Midcrust program is driving the Fortran 77 seismic routines correctly. This can be shown by generating receiver function time series forward models for several unique input models and then comparing them to independently generated receiver functions.

The velocity and thickness parameters that were used to construct the different reference seismograms would be used as the input model for the forward modeling codes and the time series generated from these would be compared with the reference time series. These reference seismograms were made using a different suite of Fortran 77 codes in combination with some Seismic Analysis Code (SAC) routines, and so were independent of the Midcrust routines.

Three distinct models and matching time series were used. The Midcrust program was able to produce virtually identical time series from these model parameters. In Figure 3, the independently generated reference model is shown plotted along with the one made by Midcrust.

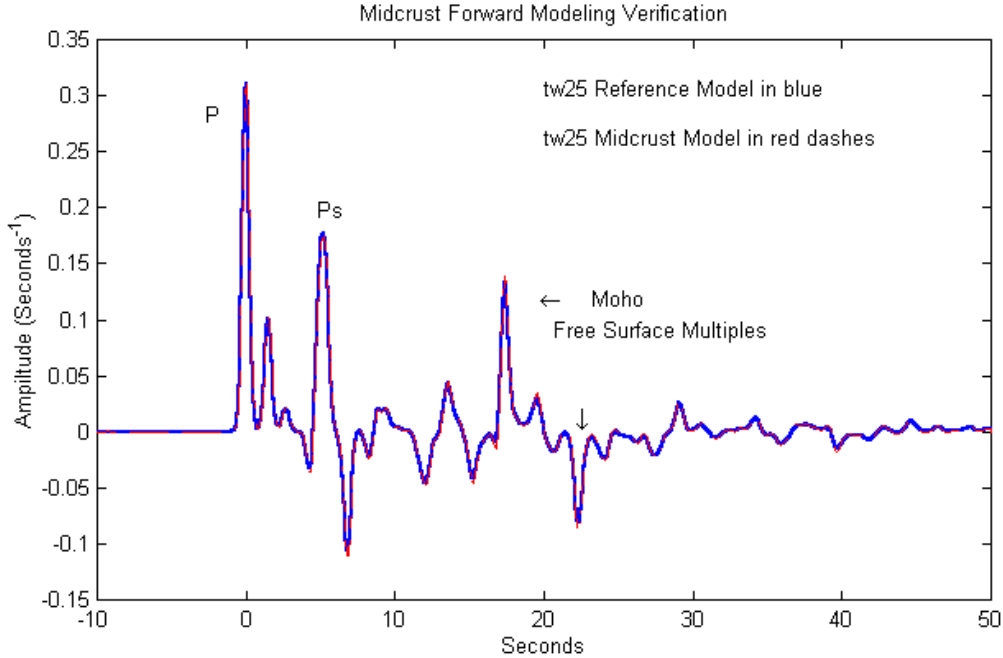


Figure 2. Comparison of reference model receiver function to one produced by Midcrust with the same model parameters.

#### Synthetic Data Study

An examination of population misfit value distributions vs. temperature per iteration revealed very little change in misfit and competition values. The original temperature schedule was then adjusted (diminished) to allow greater competition over a longer duration of the simulation. Originally the cooling schedule (T2) was

$$\begin{aligned} \text{Temp}[0] &= \text{maxval}[\text{misfit}] \\ \text{Temp}[i] &= \text{Temp}[0] * 2^{-i} \end{aligned}$$

where  $i$  = iteration number. The probability of making an uphill move,  $p(c, \text{Temp}[i]) = \exp(-c/\text{Temp}[i])$ , where  $c$  is the percent of change. The probability for making a 1% change has dropped to 0.0194 by the 13th iteration, and the probability of making a 25% change is 0.0021 at the 9<sup>th</sup> iteration. This explains why little change was observed in model populations after a dozen iterations.

The new cooling schedule (TE) is

$$\begin{aligned} \text{Temp}[0] &= \text{maxval}[\text{misfit}] \\ \text{Temp}[i] &= \text{Temp}[i-1]/A \end{aligned}$$

where

$$A = \text{Temp}[0] * \exp\left(\frac{\ln(-1\% / (T[0] * \ln(0.01)))}{-N}\right)$$

and N is the total number of iterations selected for a program run.

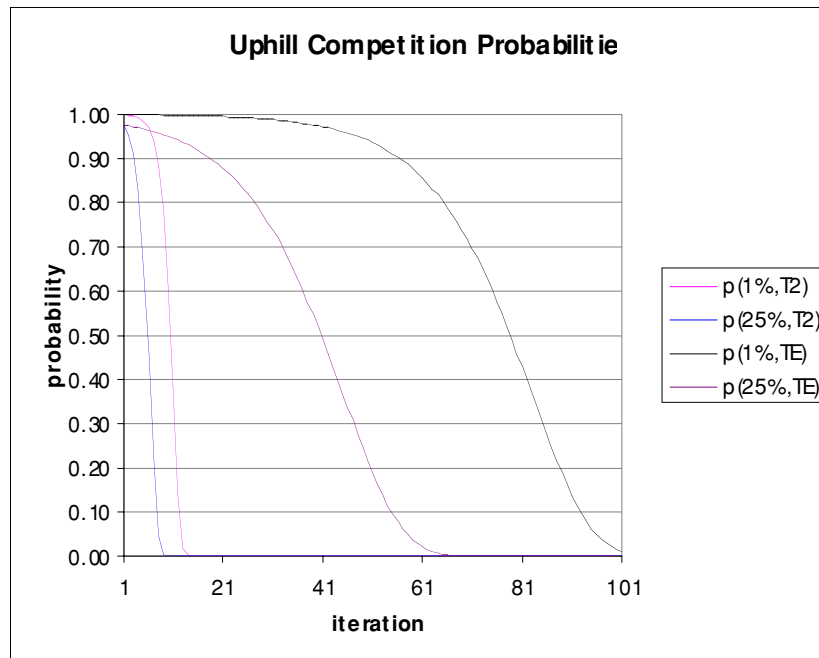


Figure 3. Probability of making an up hill move for the two cooling schedules, T2 is the old and TE the new schedule.

As can be seen from Figure 3 the new cooling schedule offers a greater probability of making uphill moves for more iterations.

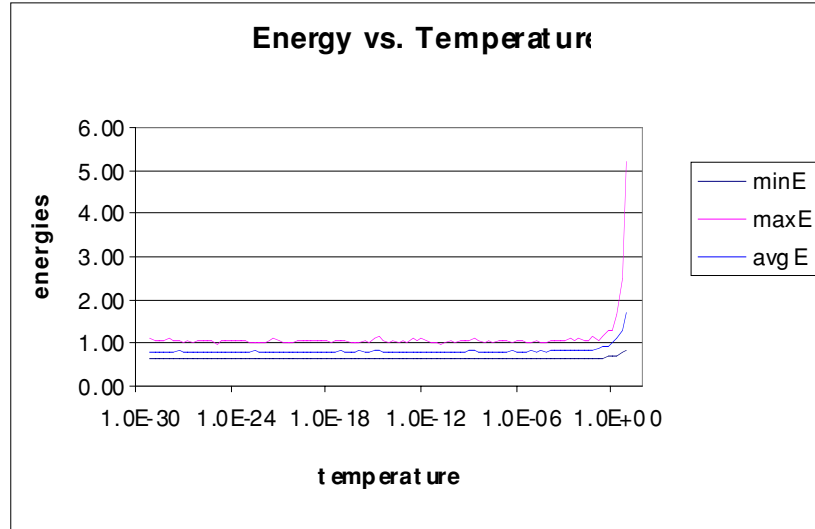


Figure 4. Model Energy with Temperature Schedule T2.

Model energy is an alternative term for fitness, often referred to as misfit, since this value quantifies the difference between a model prediction and the data. The term energy indicates the likelihood of an offspring model ending up a good distance away from its parent if plotted in the multidimensional parameter or solution space. This space is the multi-axis coordinate system defined by the model parameters. In Figure 4 the long plateau in energy across magnitudes of temperature indicates that the population has become trapped in local minima. This would be fine if the energy surface were concave, but it is known that in this type of problem the energy surface usually contains many hills and valleys [Groot-Hedlin and Vernon, 1998]. Note: in these plots, temperature decreases from right to left with iteration.

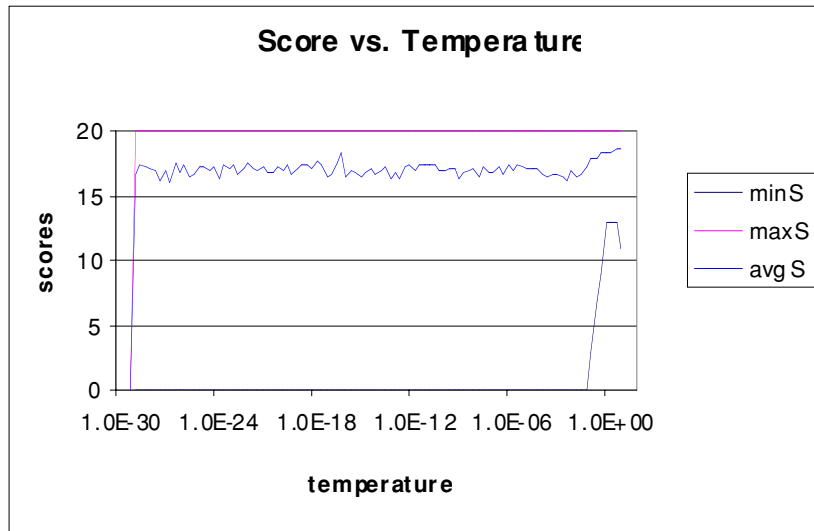


Figure 5. Model scores from competition stage compared to temperature for T2.

For schedule T2 the average model scores remain high throughout the run. It would be preferred if they would drop to a low plateau when the temperature has cooled to the last order of magnitude.

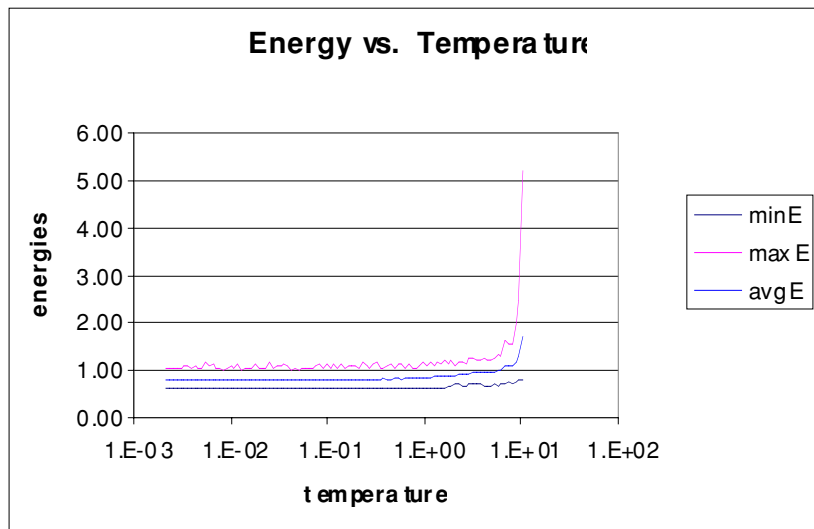


Figure 6. Energy vs. temperature for cooling schedule TE.

With the new cooling schedule (TE) more energy remains in the system a little longer. This means more large misfit values, but also that the program explores the solution space over a longer number of iterations. As mentioned, it would be better if the energy did not plateau until the last order of temperature magnitude. This is still not the case and is an issue to be addressed in the continuing development of the code.

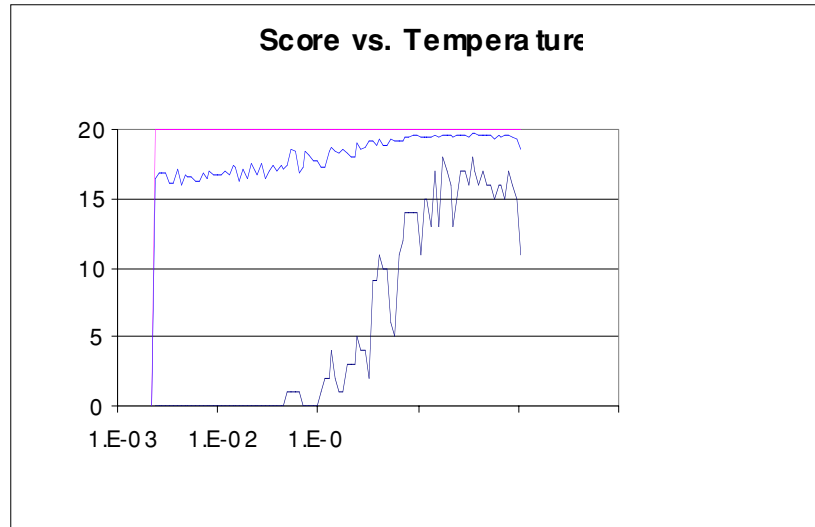


Figure 7. Scores vs. temperature for cooling schedule TE.

There is little change in the average or maximum scores from T2 to TE. However, significantly, the minimum score remains much higher for several orders of magnitude. This shows that less really poor models are being made.

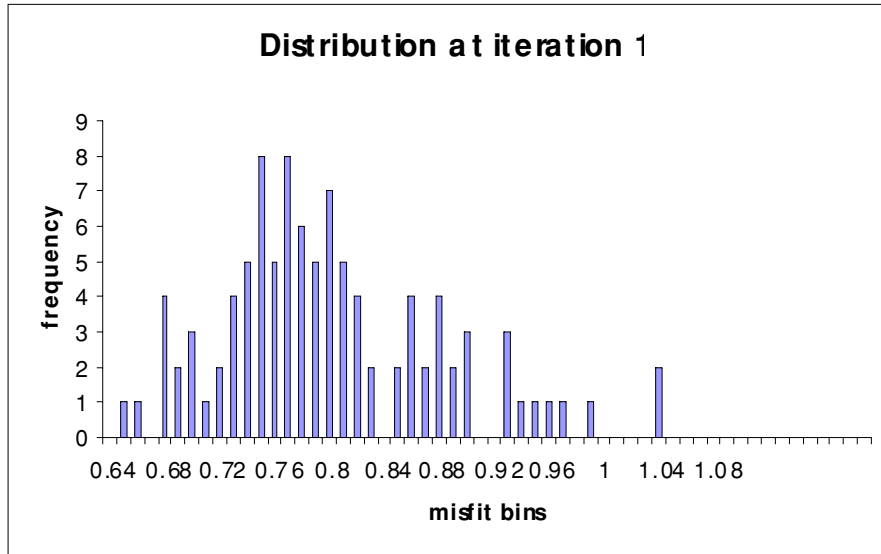


Figure 8. Histogram of misfit values after 100 iterations

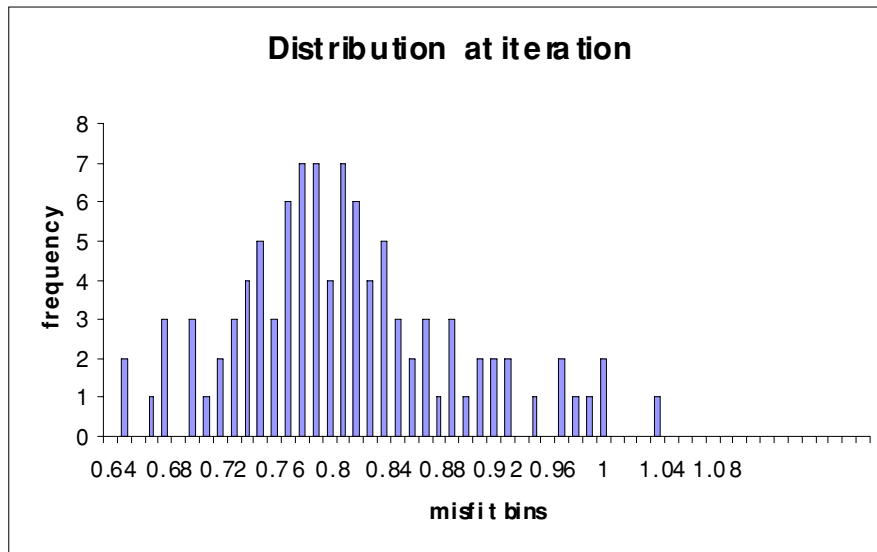


Figure 10. Histogram of misfit values at iteration 53.

The best model overall was found to be at iteration 53 (model #92), and has a misfit value  $6.42053E-01$  compared to  $6.48278E-01$ , the smallest misfit after iteration 100 (model #13). The histogram of misfit values at iteration 53 (Fig. 10) is shown along with the one after iteration 100 (Fig. 9). These histograms, showing the distribution of misfit values, exhibit a fairly bell shaped curves, and are not too badly skewed.

The time series for the best model overall, based on misfit value, (Fig. 10), shows that this model does a good job of fitting the first arrivals but fails to pick up the Moho free surface multiples.

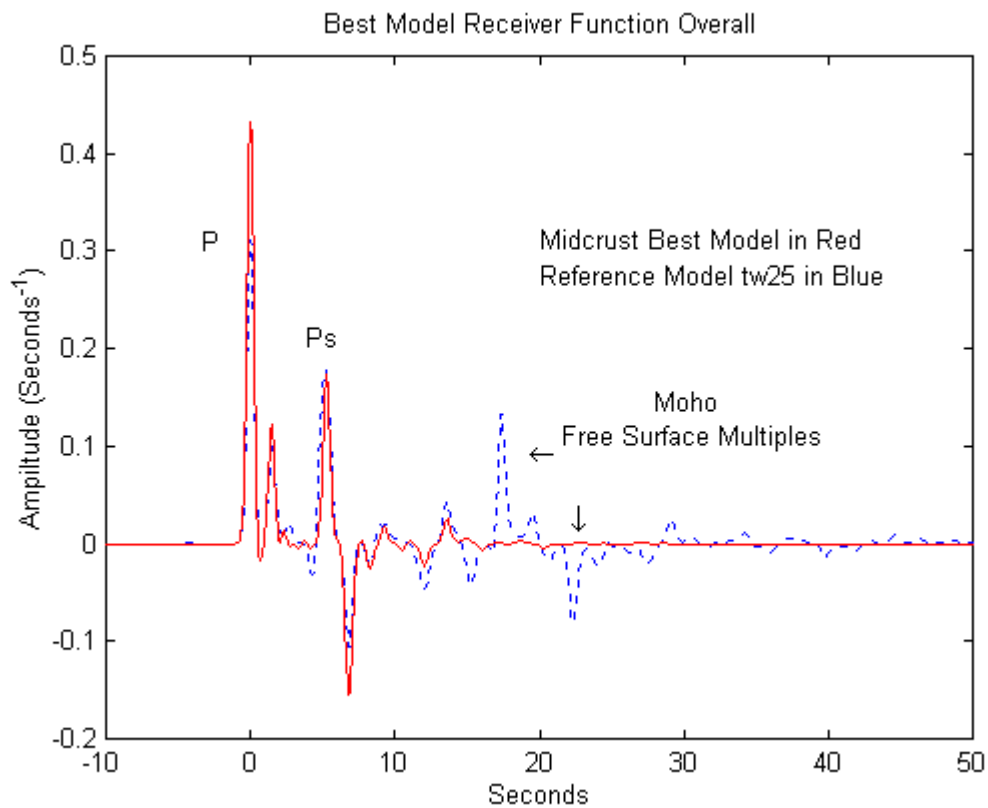


Figure 10. Time Series for Best Model after 100 Iterations

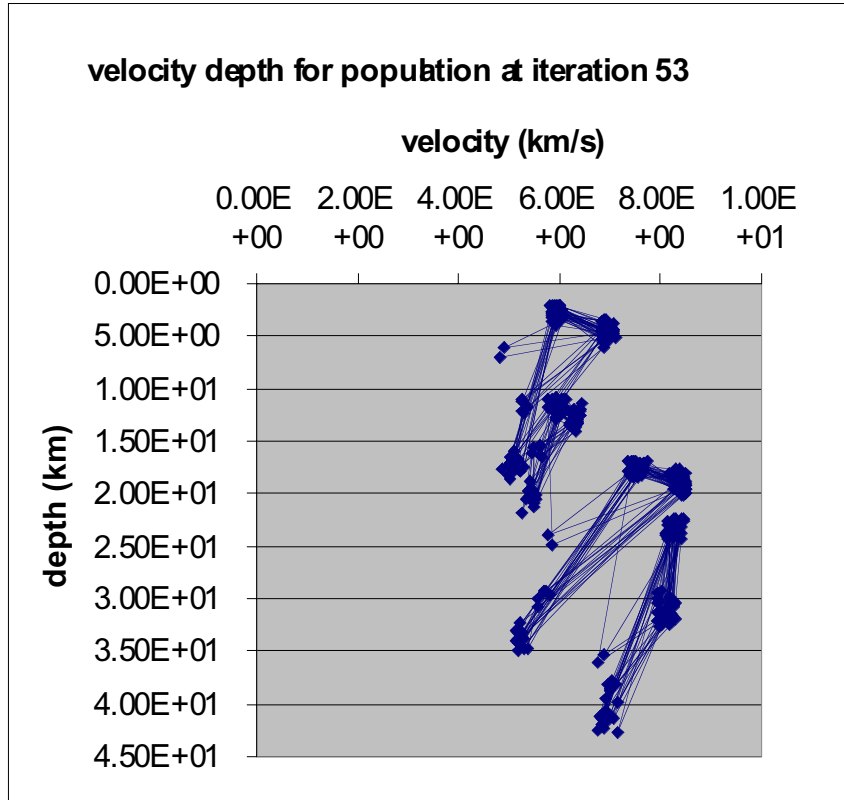


Figure 11. Velocity/thickness for population at 53<sup>rd</sup> iteration.

The velocity and depth plots for the 53<sup>rd</sup> and 101<sup>st</sup> iterations are showing some clustering of models. The distribution of the 53<sup>rd</sup> is looking a bit better than the 100th. Even in the 53<sup>rd</sup> iteration very few models are picking up the low velocity zone (6.2km/s at 35km to 40 km), and at the 101<sup>st</sup> there are none. This is clearly an important issue to address in the further development of this code.

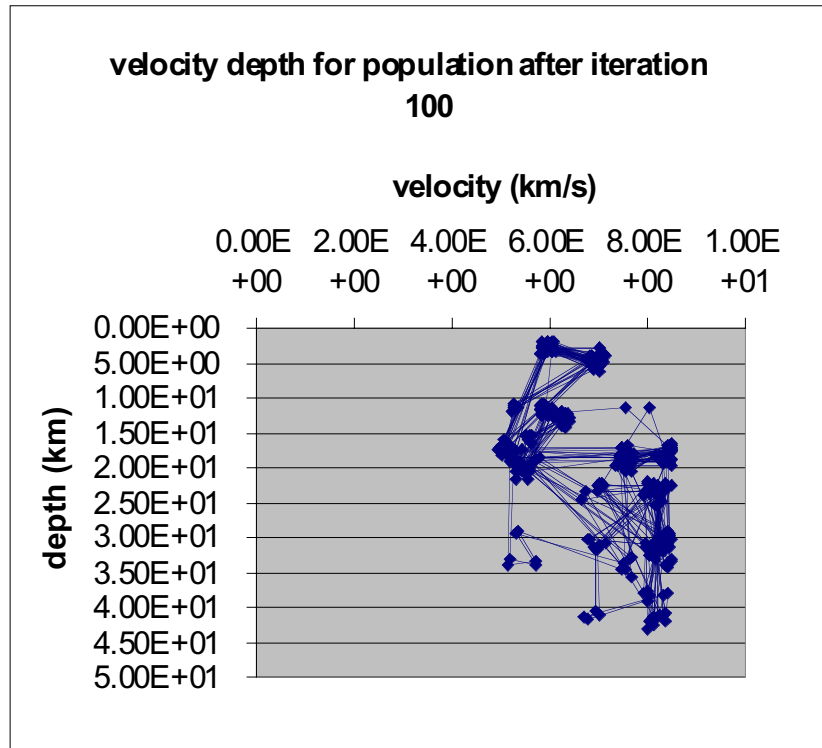


Figure 12. Velocity/thickness for population after 100<sup>th</sup> iteration.

### Conclusions

The new cooling schedule has improved the competition among population members but has provided little improvement in the energy evolution of the system. Determination of temperature schedules, model parameter perturbation functions, and misfit (energy) metrics for any given problem which evolve the system energy and resulting competitions in a meaningful way requires a significant amount of experimentation. Different cooling schedules can be tried, the number of competitions can be adjusted, and a better delta value for perturbing the parent parameters may be found. I also feel that there may be a better norm for the signal correlation, although an L1 norm is good at dealing with outliers [Groot-Hedlin and Vernon, 1998]. The problem being addressed has a high dimensionality, there are a lot of different, valid moves to make when perturbing a model.

Inverse modeling tends to be non-unique [Sheriff and Geldart, 1995] and materials of different density and thickness can produce the same travel times. One of the best features of EP is the ability to suggest solutions that are unexpected. This is one of the biggest challenges for modeling and simulation. We need to be able to verify that our modeling and simulations are valid and preserve the ability of the algorithms used to find unexpected solutions. This is where combining different methods for determining velocity structure, combined with a good geological understanding, and the ability to make intuitive decisions is required to meet the challenges inherent to this type of problem.

Two important areas are next on the agenda: tests with noisy data as compared to noiseless synthetics, and a parallel implementations targeting the IBM SP2 at San Diego Supercomputing Center (SDSC), and the more accessible multiprocessor Sun Enterprise Systems at SDSU. The most critical of these is the noise study, which will determine just how effective this modeling technique will be with real data. Success in this area will enable this program to make a serious contribution to determining actual velocity structure. The parallel implementation is also of great importance, however, and will contribute directly to the success of the noise study by enabling more program runs through faster execution. It will also allow larger problems to be addressed, that is to search a larger solution space, (i.e. models with more layers), and to add more parameters, thereby increasing the resolution of the models. This will be an important step toward tackling more complex inverse problems with larger data sets and multi-dimensional models.

There are several other related areas in which further work on this project will yield beneficial results. One is in the area of data management, and the other in data visualization. The ability to produce large amounts of output is very nice but without the ability to manage and communicate this data it is not of much use. Therefore a suite of Unix shell scripts (developed as a part of the ongoing data studies) will be included in the Midcrust package that will catalogue, store and filter the data. Visualization of the data has produced some challenges as well. It is not hard to interpret a couple of time series plots or velocity/depth profiles, but it is not as easy to do this with many models over many iterations. Therefore, another goal will be to explore this area of scientific visualization.

## Acknowledgements

I would like to thank Dr. Steven Day, Professor, SDSU, Geology, for bringing this project to my attention and being my mentor in this internship. His guidance and patience have been of tremendous help, both in this project and in helping me find a focus for my continued studies. I would also like to thank Richard Frost, SDSU, Mathematics and Computer Science, for being my computer science advisor in this project. And again patience and guidance are the key words in helping me with the code development and in planning my graduate studies. I also thank Charles Hoelzer, who developed the original Matlab code. His thesis and his willingness to answer so many questions, have been a great assistance. I would like to thank all the people at SCEC for making this wonderful learning experience possible, with a special thanks to Mark Benthien, Internship Coordinator, for all his assistance as well.

## References

- Ammon, C.J., The Isolation of Receiver Effects From Teleseismic P Waveforms, Bulletin of the Seismological Society of America, Vol.81, No. 6, pp 2,504-2,510, Dec. 1991.
- Ammon, C.J., Randall, G.E., and Zandt, G., On the Nonuniqueness of Receiver Function Inversions, Journal of Geophysical Research, Vol. 95, No. B10, pp 15,303-15,318, Sept. 1990.
- Fogel, D.B., Autonomous Automata, Industrial Res., Vol. 4, pp 14-19, 1962.
- Groot-Hedlin, de, C.D., Vernon, F.L., An Evolutionary Programming Method for Estimating Layered Velocity Structure, Bulletin of the Seismological Society of America, Vol. 88, No. 4, pp 1,023-1,035, 1998.
- Hoelzer, C.E., An Evolutionary Programming and Simulated Annealing Method for Nonlinear Inversion of Teleseismic Receiver Functions, Masters Thesis, SDSU, 1998.
- Minster, J.H., Williams, N.P., Masters, T.G., Gilbert, J.F., Haase, J.S., Application of Evolutionary Programming to Earthquake Hypocenter Determination, Proceedings of the Fourth Annual Conference of Evolutionary Programming, pp 3-7, 1995.
- Nava, F.A., and Brune, J.N., An Earthquake-explosion Reversed Refraction Line in the Peninsular Ranges of Southern California and Baja California Norte, Bulletin of the Seismological Society of America, Vol. 72, pp 1,195-1,206, 1982.
- Sheriff, R.E., Geldart, L.P., Exploration Seismology, Cambridge, Cambridge University Press, 1995