Two Dimensional Modeling and Inversion of the Acoustic Wave Equation in Inhomogeneous Media

John Fawcett

Abstract

In this paper we outline a fast and efficient method for the finite difference calculation of solutions to the two dimensional acoustic wave equation in an inhomogeneous medium. We then implement our forward modeling scheme in an inversion procedure to determine the two dimensional velocity variation. A numerical example is given.

Introduction

The initial-boundary value problem that is central to the work in this paper is:

$$\frac{1}{c^2(x,z)}\frac{\partial^2 P}{\partial t^2} - \nabla^2 P = \delta(\vec{X}-\vec{X_0})S(t) \tag{1a}$$

$$P(x,z=0,t) = 0 \tag{1b}$$

$$\frac{\partial P}{\partial z}(x,z=L,t) = 0 \tag{1c}$$

$$P(x,z,t=0) = 0 \qquad P_t(x,z,t=0) = 0 \tag{1d}$$

$$S(t) = 0 \; t<0 \qquad \vec{X},\vec{X_0} \; \varepsilon \; \mathbf{R}\times[0,L] \tag{1e}$$

In terms of acoustics, (1b) represents a pressure release surface and (1c) a rigid bottom. However, the methods which we shall describe, apply equally as well for other choices of boundary conditions (1b) and (1c).

The inverse problem is the estimation of $c(x,z)$ from a knowledge of $\frac{\partial P}{\partial z}(x,z=0,t;\vec{X_0})$ where $\vec{X_0}$ represents the source position. In our inversion method we shall parametrize $c(x,z)$ in terms of bicubic splines. The unknown parameters, $\vec{p}$,will be the values of $c(x,z)$

at the splines' nodes. We shall suppose that we have a set of discrete observations, $\vec{d} \equiv \frac{\partial P}{\partial z}(x_i, z=0, t_j; \vec{X_0})$. Our method will be a straight- forward implementation of non-linear least squares theory:

$$\min_{\vec{p}} \| \vec{d}_{ij} - L(\vec{p}) |_{i,j} \|_2 \tag{2}$$

where L is the solution of (1). The non-linear least squares algorithm we shall employ is equivalent to solving a sequence of linearized least squares problems. Hence, our method is an iterative application of a linearized method. Tarantola [9] has previously discussed in theory such methods.

At each iteration we must solve the problem (1) for the current estimate of $c(x,z)$, and in fact, we sometimes solve (1) several times to calculate an approximate Jacobian $\frac{\partial L(\vec{p})}{\partial \vec{p}}$. Hence, our forward modeling algorithm must be fast and accurate. We shall first discuss the details of our finite difference modeling program, and then discuss our inversion procedure.

Finite Difference Modeling of the Acoustic Wave Equation

We consider (1) on a finite domain $x \varepsilon [0,X]$ $z \varepsilon [0,L]$, and we discretize (1) in space and time to obtain the following explicit scheme:

$$P^{n+1}(x_i,z_j) - 2P^n(x_i,z_j) + P^{n-1}(x_i,z_j) = \frac{c^2(x_i,z_j)}{12} r^2 \times \tag{3}$$

$$(16(P^n(x_{i+1}, + P^n(x_i,z_{j+1}) + P^n(x_{i-1},z_j) + P^n(x_i,z_{j-1}))$$

$$- (P^n(x_{i+2},z_j) + P^n(x_i,z_{j+2}) + P^n(x_{i-2},z_j) + P^n(x_i,z_{j-2})))$$

$$- 5c^2(x_i,z_j)r^2 P^n(x_i,z_j) + S(t_n)\delta(l,m)c^2(x_i,z_j)r^2 + O(h^4+(\Delta t)^2)$$

$$\delta(i,j) = 1 \text{ if } i=l, j=m: \quad = 0 \quad otherwise.$$

We have taken the x and z discretizations both equal to h and $r = \frac{\Delta t}{h}$, where Δt is the time discretization. The Courant-Friedrich-Lewy (CFL) condition for the stability of the scheme (3) is that $r\ c \leq (\frac{3}{8})^{\frac{1}{2}}$ (Alford et al [1]), where we will take $c \geq \max_{i,j} c(x_i,z_j)$. As is indicated in (3) our scheme is spatially fourth order accurate. The source function $S(t)$ in (1a) determines which wavelengths will dominate in the solution $P(x,z,t)$. A rule of thumb suggested by Alford et al [1] is that for the discretization (3), five grid points per upper half

power wavelength are needed to properly resolve the solution. Here the upper half-power wavelength refers to the upper wavelength $\lambda=\frac{c}{f}$,where the source's power spectrum $|S(f)|^2$ has reached half its maximum value.

The only unusual term in (3) is the source term $S(t_n)\delta(l,m)\ c^2(x_i,z_j)r^2$. Previous authors have suggested more complicated procedures for the inclusion of sources, but the straightforward inclusion of the source at a single grid point works perfectly well. Intuitively, if the source point $\vec{X}_0$ is located at the grid point (l,m) then from the discretization of (1a), we would expect the discretized source to have the form:

$$\frac{1}{N}(S(t_n)\delta(l,m)c^2(x_i,z_j)(\Delta t)^2) \tag{4}$$

where N is some normalization. From the definition of a two dimensional delta function $\delta(\vec{X}-\vec{X}_0)$ we know that :

$$(\delta,\Psi(\vec{X})) = \Psi(\vec{X}_0) \quad where \tag{5a}$$

$$(\delta,\Psi) \equiv \int_0^X \int_0^L \delta(\vec{X}-\vec{X}_0)\Psi(\vec{X})dydx. \tag{5b}$$

A consistent inner product to use with our grid is :

$$(f,g) \equiv \sum_i \sum_j h^2 f(i,j)\ g(i,j) \tag{6}$$

If we take $\delta(\vec{X}-\vec{X}_0)$ as approximated by $\frac{\delta(l,m)}{N}$, then N must be equal to h^2 for (6) to be consistent with (5a) and (5b). With $N=h^2$, we obtain the source term in (3). We can evaluate $P(x,z,t;\vec{X}_0)$ analytically for $S(t) = t(t-t_c);0\le t\le t_c$, $S(t)=0\ t\ge t_c$ and $\frac{\partial P}{\partial z}(x,z=0,t)=0$, $c(x,z)\equiv 1$. In Figure 1, we show the analytical solution for the first arrival, and the numerical solution using scheme (3). Here we are looking at a trace a distance , $|\vec{X}-\vec{X}_0|=.2864$ and we used a grid of 101x51 points with 400 time steps ($\Delta x=\Delta y=.02$,$\Delta t=.005$). (There are two traces here, but they are indistinguishable.)

To implement (3) on the array processor (Floating Point AP 120B) we consider (thanks to Peter Mora) the discrete Laplacian in (3) as a sum of convolutions in the x and z directions. We define the vector $\vec{b}$:

$$\vec{b} \equiv [\frac{-r^2}{12},\frac{16r^2}{12},-2.5r^2,\frac{16r^2}{12},\frac{-r^2}{12}] \tag{7}$$

and we can write that :

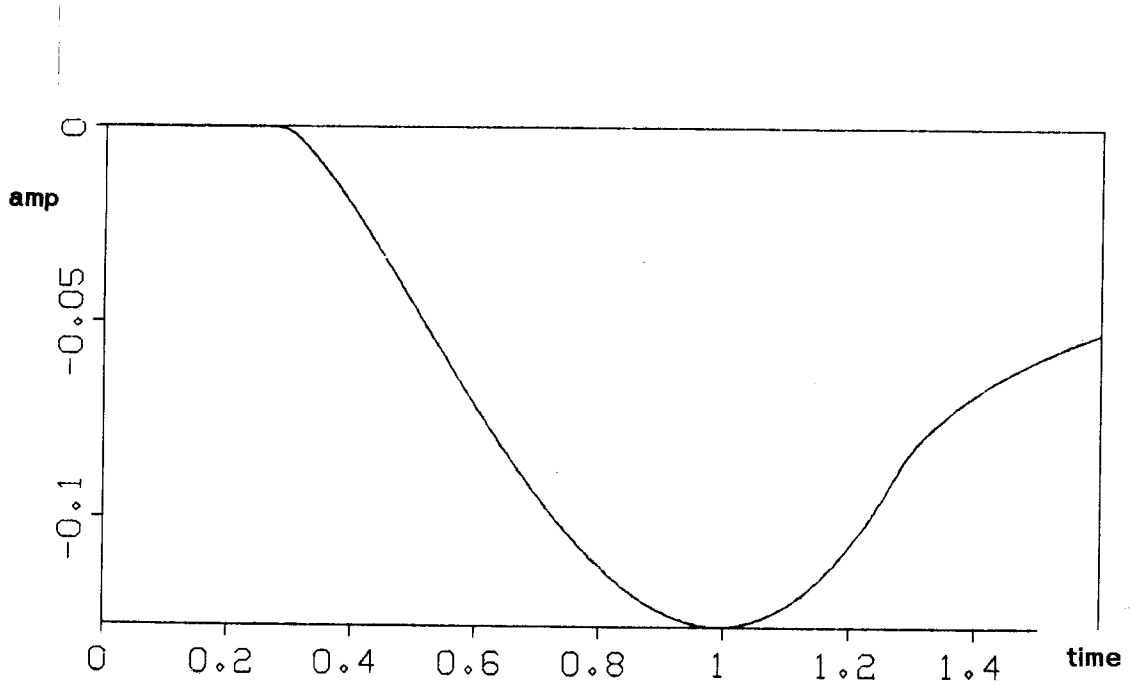

Figure 1. Analytic and Numerical Response to Source $t_c = 1$

$$\nabla^2_{i,j} = \vec{b}(j) * P(i,j) + \vec{b}(i) * P(i,j) \tag{8}$$

This convolution, and all vector/matrix operations are done quickly in the array processor. The field resulting from this convolution is then multiplied by the corresponding elements of the velocity matrix (actually, square of the velocity) to produce $\tilde{P}$. Then $\tilde{P}$, P^n, P^{n-1} , and the source contribution for the particular time level are added together to produce P^{n+1} . We then move the elements of P^{n+1} corresponding to the values of the pressure field at selected traces at $z = \Delta z$ into a storage area in the array processor. P^{n-1} is then overwritten by P^n, P^n overwritten by P^{n+1}, and we start again for the next time step. This whole procedure, for all time steps, is carried out with one call to the array processor. For a small grid (e.g., 51×26 with 200 time steps) we have a saving of a factor of 8 to 10 times over the VAX, using the array processor; for larger grids the savings is even more substantial. At the end of all the time steps, we transfer $P(x_j, z = \Delta z, t)$ to the VAX , and use $\frac{P(x_j, z = \Delta z, t)}{\Delta z}$ to represent $\frac{\partial P}{\partial z}(x_j, z = 0, t)$.

Finally, we must address the question of implementing boundary conditions. Our boundary conditions in the vertical direction (1b) and (1c) have physical significance and are easy to implement. We can consider our grid to have 2 rows appended to the top, and 2 rows appended to the bottom. We then simply extend the solution oddly and evenly, respectively, about the two boundaries. However, in our finite domain, the horizontal boundaries have no physical significance, but numerically we must impose some boundary condition. Many authors have suggested various **absorbing boundary conditions** which in some way simulate a transparent boundary. Clayton and Engquist [3] have given schemes based upon one-way wave operator approximations. Here the accuracy (in particular the range of incident angles over which the approximation is valid) and the complexity of the scheme depends upon the order of the one-way operator used

In our problems, energy will be reflected off the top and bottom boundaries and there will be in general, energy incident upon the side boundaries at a large range of incident angles. Also, although probably possible to implement in the array processor , we desire , for our first attempts at inversion , simple boundary conditions which can be easily **internally** implemented. A "brute force " method is to make the horizontal size of the grid large enough so that reflections from the sides would have no effect on the traces of interest in the relevant time window. However, due to storage limitations within the array processor, we cannot make the grid too large. Our method is to use a grid twice as large horizontally as vertically, and we then calculate the pressure field twice , once for zero boundary conditions on the sides and once for zero-slope conditions. We then average the two solutions(see also Smith [8]). This effectively gives us a grid four times as wide as deep. Perhaps in the future, we shall examine the more effective use of side boundary conditions.

Optimization Procedure

As we mentioned above, we will assume that $c(x,z)$ is parametrizable in terms of bicubic splines. This is most suited for the situation where $c(x,z)$ is smooth. The use of one-dimensional spline approximations for one-dimensional inverse problems has previously been studied by Banks et al [2]. The algorithm we use for two dimensional spline interpolation is based on that of de Boor [4]. Here, we suppose we have a grid with $c_{i,j}$ the value of $c(x,z)$ at $x=x_i, z=z_j$. Then one dimensional cubic splines are found in the x and z directions. This procedure determines $\frac{\partial c}{\partial x}$ and $\frac{\partial c}{\partial z}$ at the nodes. Fitting bicubic splines along each column of the grid (or row) determines $\frac{\partial^2 c}{\partial x \partial z}$ at the node points. The 4 quantities $\left[c_{i,j}, \frac{\partial c_{i,j}}{\partial x}, \frac{\partial c_{i,j}}{\partial z}, \frac{\partial^2 c_{i,j}}{\partial x \partial z}\right]$ at 4 corners uniquely determines the bicubic spline

$\varphi(x,z) \equiv \sum_{m=0}^{3} \sum_{n=0}^{3} \Gamma_{m,n}{}^{i,j} (x-x_i)^m (z-z_j)^n$ within the rectangle bounded by the corners. With the one-dimensional splines, there is always the problem of what to do at the boundaries. We used de Boor's [5] " not - a- knot " condition. This is a condition that replaces the need for derivative information at the endpoints with a condition that requires the two cubics on the last 2 intervals to have continuous third derivative at their common node (i.e., they are the same cubic).

The parameters, $\vec{p}$, to determine are the values of $c(x,z)$ at the nodes. Once we have found $c(x,z)$ at the nodes, then from bicubic spline interpolation,, $c(x,z)$ is known everywhere, within this approximation. Our problem is :

$$\min_{\vec{p}} \| \vec{d} - L(\vec{p}) \|_2 \tag{9}$$

We will employ the Gauss-Newton method for non-linear least squares. If we denote $\vec{d} - L(\vec{p})$ as $\vec{F}$ and $\Phi(\vec{p}) \equiv \frac{1}{2} \vec{F}^T \vec{F}$ then (9) is equivalent to finding $\vec{p}_0$ such that $\nabla\Phi = \vec{0}$ where $\vec{p}_0$ is a minimum point. To find $\vec{p}_0$,Newton's method would be:

$$\vec{p}^{\,n+1} = \vec{p}^{\,n} + \Delta\vec{p}^{\,n} \quad where: \tag{10}$$

$$\Delta\vec{p}^{\,n} = \left(-\nabla^2\Phi\right)^{-1} \nabla\Phi(\vec{p})$$

For problems, where $\vec{F}(\vec{p}_0) \approx \vec{0}$, then the Gauss- Newton method approximates $\nabla^2\Phi$ by $\underline{J}^T\underline{J}$ where :

$$J_{i,j} \equiv \frac{\partial F_i}{\partial p_j} \quad i=1,...,m \quad j=1,...,n$$

Thus, we can write (10) as:

$$\Delta\vec{p}^{\,n} = \left(-\underline{J}^T\underline{J}\right)^{-1} \underline{J}^T \vec{F}(\vec{p}) \tag{11}$$

The vector, $\Delta\vec{p}^{\,n}$, is also the solution of the linear least squares problem

$$\min_{\vec{p}} \| \underline{J}\Delta\vec{p} + \vec{F} \|_2 \tag{12}$$

It is often advantageous to " damp" the step of (11), and use instead :

$$\Delta\vec{p}^{\,n} = -\left(\underline{J}^T\underline{J} + \lambda_n I\right)^{-1} \underline{J}^T \vec{F}(\vec{p}) \tag{13}$$

Now $\Delta\vec{p}$ is the solution of the linear minimization problem with the following coefficient matrix :

$$\begin{matrix} \mathbf{m} \\ \mathbf{n} \end{matrix}\begin{bmatrix} \underline{J} & -\vec{F} \\ \lambda_n I & \vec{0} \end{bmatrix} \tag{14}$$

Our choice of λ_n will be such that $||\vec{F}^{n+1}||_2 < ||\vec{F}^{n}||_2$. For λ_n sufficiently large, this condition will be true, as $\Delta\vec{p}$ becomes aligned with $-\nabla\Phi(\vec{p})$ for $\lambda_n \to \infty$. To find the minimal length solution of (14), we employed Golub's [6] QR method of solution. We employed it once to reduce (14) to the form:

$$\begin{matrix} \mathbf{n} \\ \mathbf{n} \end{matrix}\begin{bmatrix} \underline{T} & -\underline{Q}\vec{F} \\ \lambda_n I & \vec{0} \end{bmatrix} R_1 \tag{15}$$

where $\underline{T}$ is upper triangular, and then storing R_1 ,we repeatedly solve (15) (usually once or twice) to find a good choice of λ_n (see Osborne [7]).

Another method of solution of (13) is using the singular value decomposition of the matrix $\underline{J}$:

$$\underline{J} = \underline{U}\,\Lambda\,\underline{V}^T \tag{16}$$

Here, Λ is a diagonal matrix containing the singular values of $\underline{J}$. The columns of $\underline{V}$ (the eigenvectors of $\underline{J}^T\underline{J}$) corresponding to small singular values, represent the portion of parameter space which cannot be resolved in a stable fashion. We will sometimes use (16) at our last iteration to determine what portions of the velocity field are not "well" determined.

Having the data vector , $\vec{d}$ as input to our inversion problem, and calculating $L(\vec{p})$ using our finite difference method, it is easy to calculate the residual vector $\vec{F}$. However, we must also calculate the Jacobian matrix $\frac{\partial\vec{F}}{\partial\vec{p}}$. Tarantola [9] has discussed such calculations. His method is based upon the linearization of the wave operator. If we write $n(x,z)=\frac{1}{c^2(x,z)} = n^0(x,z)+\varepsilon n^1(x,z)$ and $P=P^0+\varepsilon P^1$ where P^0 is the wave solution for the slowness field n^0 , then we obtain :

$$n^0\,P^0{}_{tt} + n^0\,\varepsilon\,P^1{}_{tt} + \varepsilon\,n^1 P^0{}_{tt} \tag{17}$$

$$+\nabla^2 P^0 + \varepsilon\nabla^2 P^1 = \delta(\vec{X}-\vec{X}_0)\,S(t) + O(\varepsilon^2)$$

Hence,

$$P^1(\vec{X},t) = -\int G^0(\vec{X},t;\vec{X}') * P^0{}_{tt}(\vec{X}',t)\,n^1(\vec{X}')\,d\vec{X}' \tag{18}$$

or:

$$\frac{\delta P}{\delta \vec{p}} = -\int G^0(\vec{X},t;\vec{X}') * P^0{}_{tt}(\vec{X}',t)\ \delta n(\delta \vec{p})\ d\vec{X}'$$

Here * denotes convolution in time and G^0 is the unperturbed Green's function. Whether to use (18) as a formula for the computation of the Frechet derivative depends upon the geometry of the problem under consideration. In general, we would have to numerically calculate $G(\vec{X},t;\vec{X}_s)$. This would mean a wavefield calculation for various source positions $\vec{X}_s$. The judicious use of a reciprocity relation could reduce the amount of calculation: however a large amount of computation still remains. On the other hand, if the data consists of the pressure field for many different source positions, then using (18) may indeed be appropriate.

We will usually be considering in our inversion examples single shot data, with various receiver positions. For these problems, it is more efficient to perturb $\vec{p}$ and form $\vec{p}' = \vec{p} + \varepsilon \vec{e}_j$ where $\vec{e}_j = [0,0,...,1,..0]^T$ (the "1" in the j'th position) and recalculate the wavefield. Hence,

$$\frac{\partial f_i}{\partial p_j} = \frac{F_i(\vec{p}') - F_i(\vec{p})}{\varepsilon} + O(\varepsilon) \qquad (19)$$

Numerical Examples

Example 1

For this example the source is located at a horizontal distance of 2.5 km. and at a depth of .3 km. The velocity field was $c(x,z) = 1.5 \times (1 + .2\sin(12x) \times \sin(12z))km/s$. To generate the synthetic data, we did the finite difference calculations completely within the VAX with a grid size of 501 horizontal points, 51 depth points, and 400 time steps with $\Delta t = .0025$. Our source function was $S(t) = .5H(t)e^{-1000(t-.15)^2}$. The Fourier Transform, $\tilde{S}(\omega)$ defined as $\int_{-\infty}^{\infty} S(t)e^{-i\omega t}\,dt$ is approximately equal to $.5\sqrt{\frac{\pi}{1000}}e^{\frac{-\omega^2}{4000}}e^{.15i\omega}$. This is a broadband source which has half-power at about 6 Hz. Using $c \approx 1.5km/s$, then the half-power wavelength is $\lambda \approx .25km$. Thus we are , from the rule discussed above, using a sufficient number of grid points.

For the inversion we shall use for data, traces from 16 offsets : x_i=.02+(i−1)×.06 i=1,...,16 ,with 50 time samples, t_j=.25+.01×(j−1) j=1,...,50 for each of these traces. This data for all 401 time samples is shown in Figure 2. For the finite difference scheme in the inversion , we use a coarser grid than in the data generation; we use 51 horizontal points and 26 vertical points. However, we still use 400 time steps, so that during the iterations the velocity can increase to approximately $5km/sec$ without the CFL stability requirement being violated.

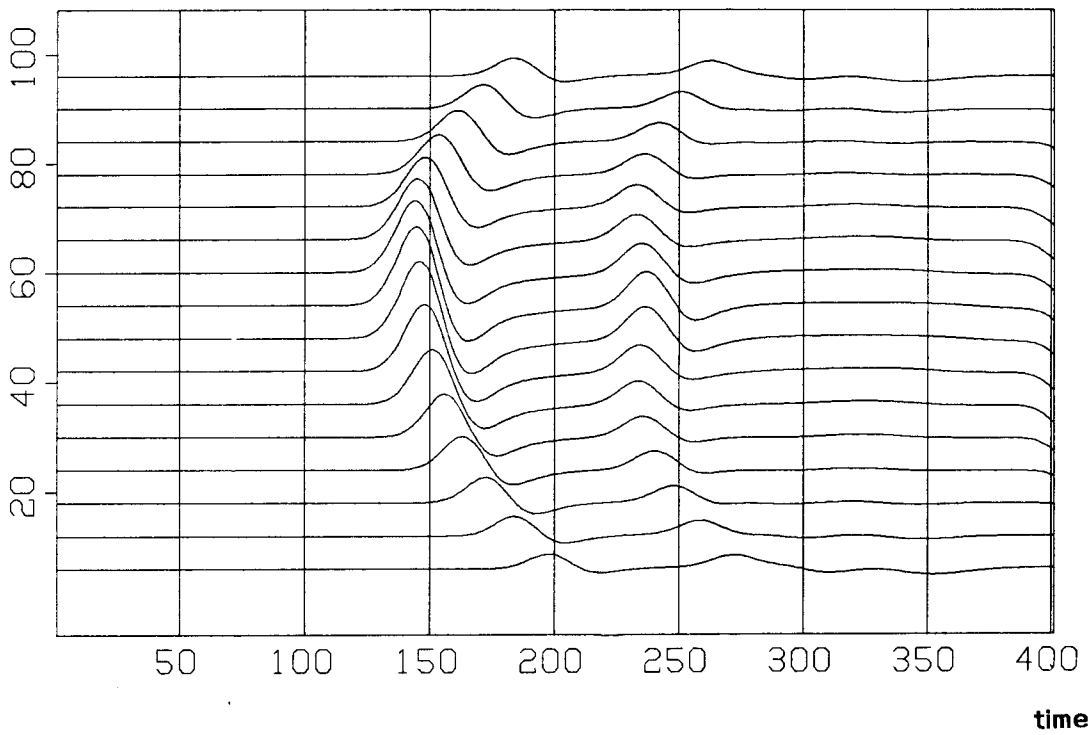

Figure 2. The Sixteen Traces used for Inversion

Along the top row, we will assumes that the velocity is known; it is $1.5km/s$. We will have 18 parameters to determine. We take as an initial guess : c_{ij}(i=1,6 j=2,4) = $1.6km/s$. In this run, we did not allow λ_n to fall below 1. In Figure 3 we show a comparison of the input data and calculated data sets for the initial guess. In Table 1 , we show the residual $||\vec{F}||_2$, and the magnitude of the gradient vector $||\underline{J}^T\vec{F}||_2$

at each iteration. We also show the resulting velocity field at selected iterations in Figure 4a-4d. In Figure 4e, we plot the true field $c(x,z)$ for comparison. The values of c_{ij} , after the final iteration are given in Table 2. In Figure 5, we show the data and calculated fields after the second iteration.

Table 1 Iterations and Convergence

it	$\|\|\vec{F}\|\|_2$	$\|\|\underline{J}^T\vec{F}\|\|_2$
1	9.30004	141.865
2	4.16730	52.5459
3	2.36897	47.1720
4	1.74593	16.3310
5	1.53574	6.23639
6	1.46706	1.32478
7	1.43601	0.47162
8	1.41921	0.30535
9	1.40903	0.23679

The final residual, for the velocity estimate after the ninth iteration was $||\vec{F}||_2 = 1.401723$.

Table 2 Final Values at Nodes

1	2	3	4	5	6
1.500	1.500	1.500	1.500	1.500	1.500
0.752	1.647	1.307	1.402	1.522	0.824
1.861	1.030	1.738	1.908	1.003	2.172
1.579	1.832	1.462	0.905	1.335	1.439

We note that the true and inverted velocity fields qualitatively agree. However, it was necessary to use $\lambda_n \equiv 1$ throughout the calculation as otherwise "wild" estimates for the nodes in the outer portions of the grid were obtained. Simple ray considerations, using our source and receiver locations, show that very little energy propagates through these portions of the grid. Thus, indeed, this portion of the velocity field is not well constrained from the data. We note that the 4 vertical node points determine one cubic interpolant. We show this cubic, determined by the inversion, and the true vertical profile for the centre trace ($n_x = 26$) in Figure 6.

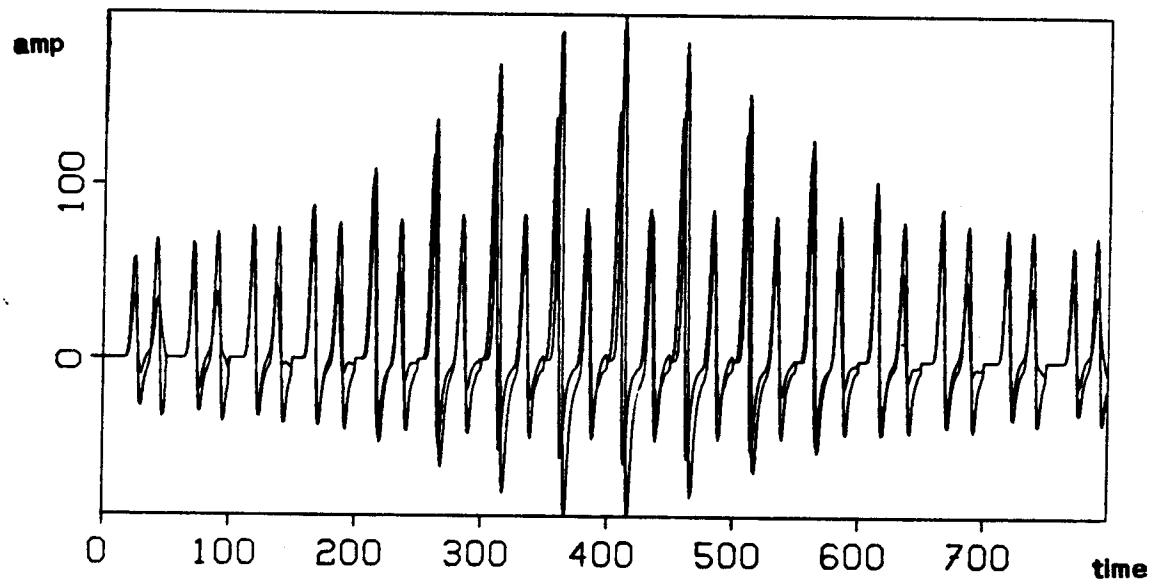

Figure 3. Discretized Data and the Generated Data for Initial Guess

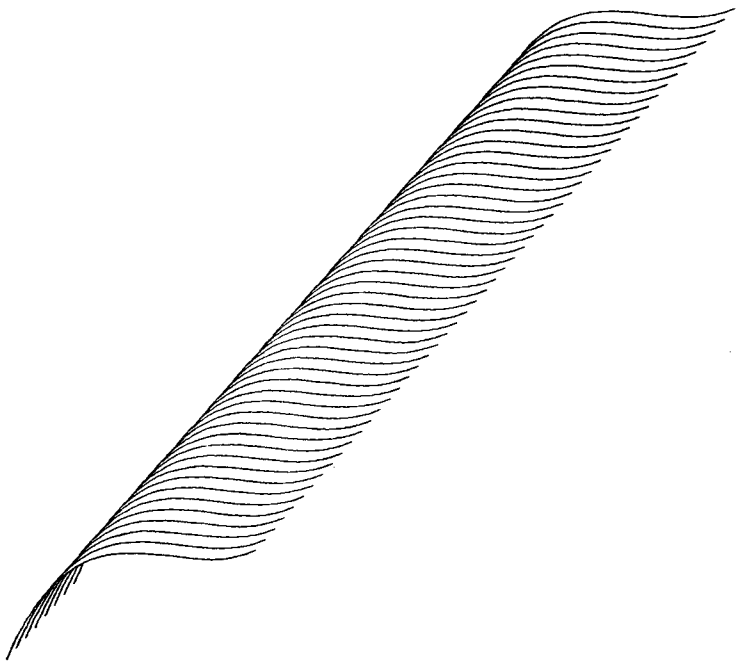

Figure 4a. Initial Velocity Field ($-1.5 km/s$)

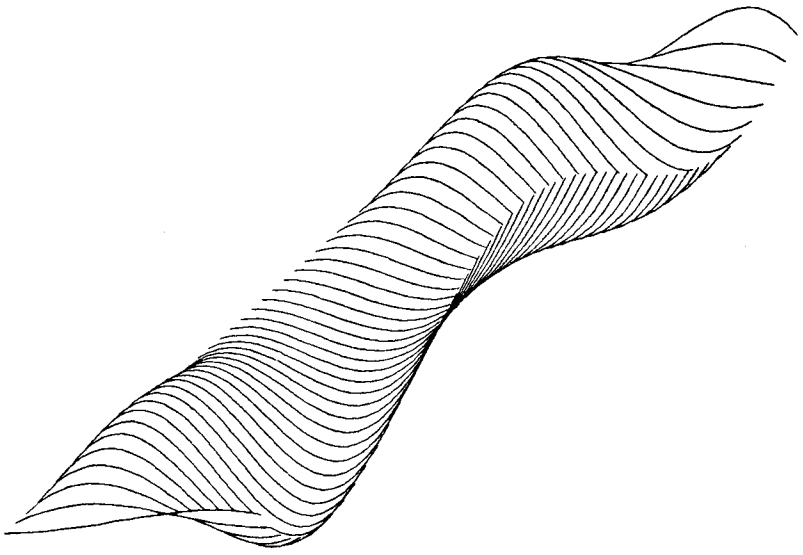

Figure 4b. Velocity Field After First Iteration

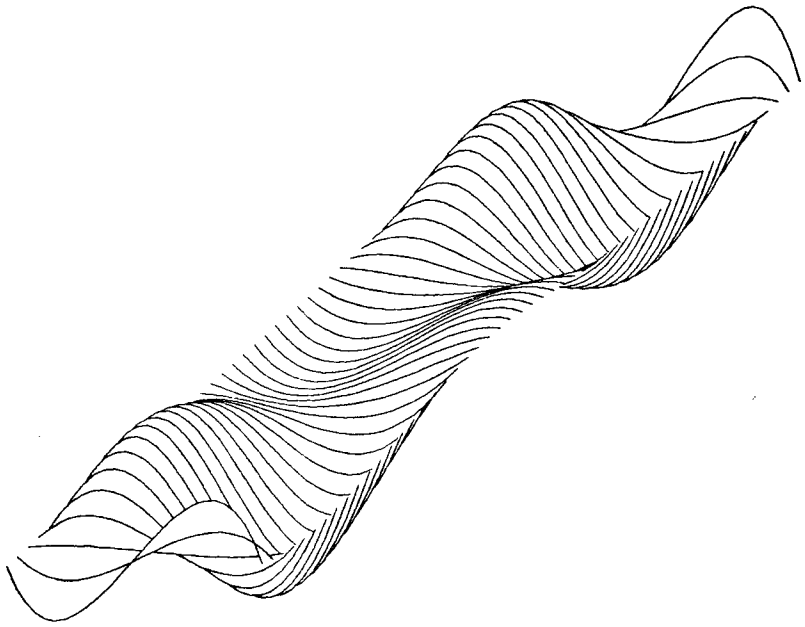

Figure 4c. Velocity Field After Second Iteration

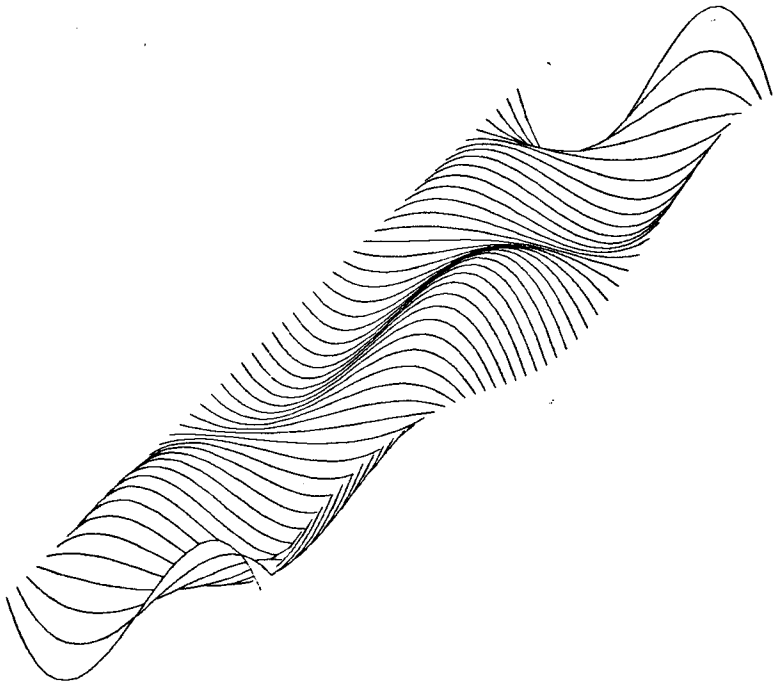

Figure 4d. Velocity Field After Final Iteration

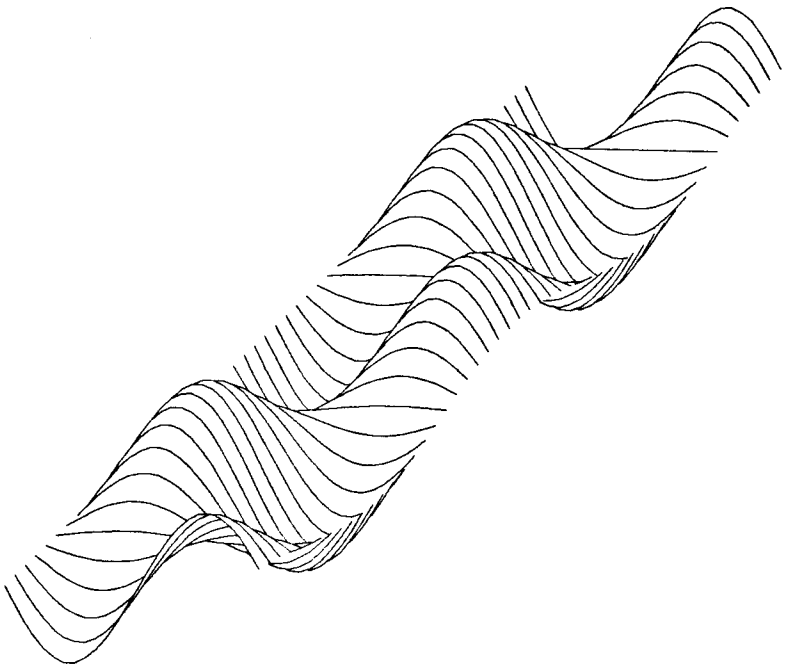

Figure 4e. True Perturbation to the Field ($1.5km/s$)

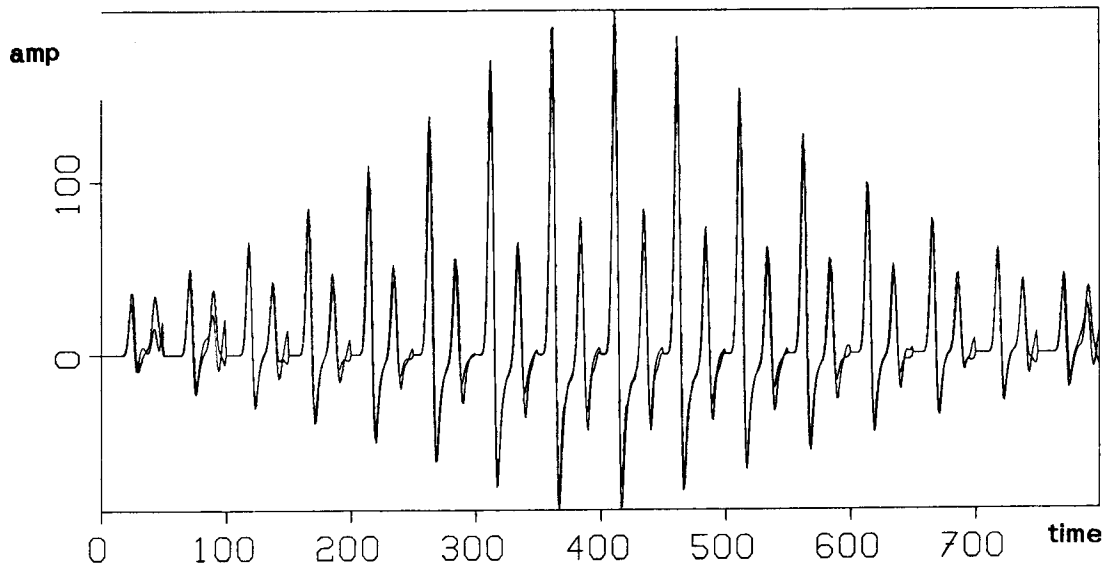

Figure 5. Discretized Data and Generated Data After Second Iteration

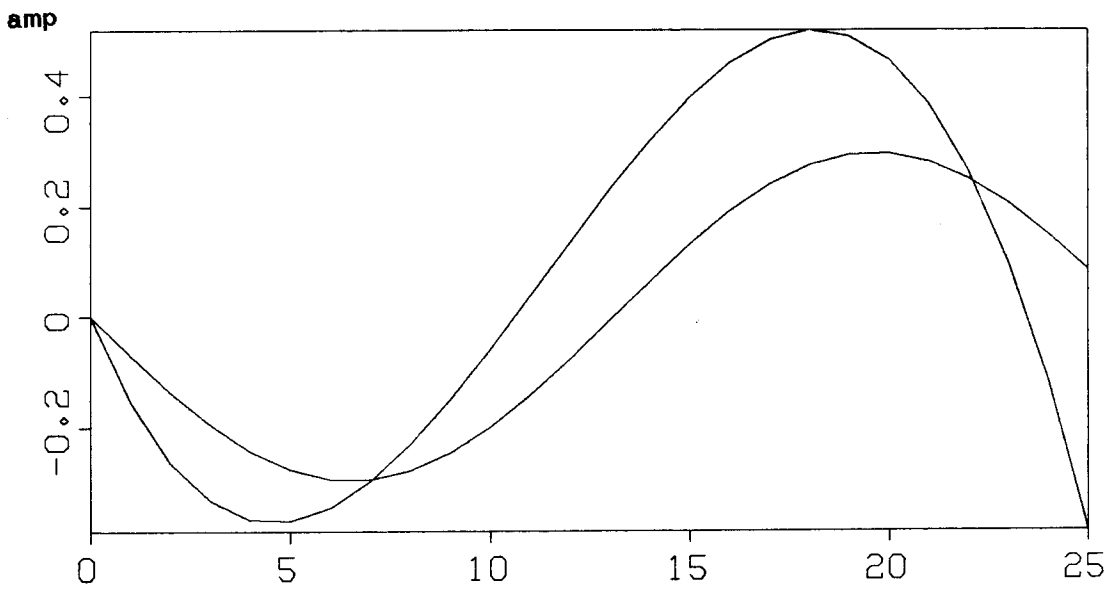

Figure 6. True and Approximate Vertical Profile for Centre Trace

Acknowledgements

I would like to thank Peter Mora and Dr. Chris Anderson of the Departments of Mathematics and Computing Science for some very helpful discussions. Also, I would like to thank Professor Joseph Keller of the Department of Mathematics for his critique of an earlier version of this report. Financial support for my tenure at Stanford comes from the Air Force Office of Scientific Research, the Army Research Office, the Office of Naval Research, and the National Science Foundation.

References

[1] R.Alford, K.Kelly, and D.Boore.,"Accuracy of Finite-Difference Modeling of the Acoustic Wave Equation", **Geophysics,** 39,No.6, P.834-842 (1974).

[2] H.Banks, K.Ito, and K.Murphy.,"Computational Methods for Estimation of Parameters in Hyperbolic Systems", **ICASE Report,** (Sept. 1983).

[3] R.Clayton and B.Engquist., "Absorbing Boundary Conditions for Acoustic and Elastic Wave Equations", **B.S.S.A.,** 67,No.6.,p.1529-1540 (1977).

[4] C.de Boor., **A Practical Guide to Splines.** Applied Mathematical Sciences, 27, Springer-Verlag, New York (1978).

[5] C. de Boor., "Bicubic Spline Interpolation", **J. Math. Phys.** ,41,p.212-218 (1962).

[6] G.Golub., "Numerical Methods for Solving Linear Least Squares Problems", **Numerische Mathematik** ,7,p206-216 (1965).

[7] M.Osborne., "Some Aspects of Non-Linear Least Squares Calculations", **Conference on Numerical Methods for Non-Linear Optimization.** ,University of Dundee, 1971 (ed. F.A.Lootsma), Academic Press, London, p.171-191 (1972).

[8] W.Smith., " A Non-Reflecting Plane Boundary For Wave Propagation Problems", **J. Comp Physics** ,15,p.492-503 (1974).

[9] A.Tarantola., "Nonlinear Inverse Problem For an Heterogeneous Acoustic Medium", preprint (1983).

Total = 387241
31454 the
11196 of
11007 is
10882 to
8340 and
6868 in
4512 be
4467 for
4106 are
4044 file
3541 if
3345 by
3158 it
3078 this
2836 or
2813 as
2709 on
2673 that
2659 with
2393 not
2287 you
2159 name
2094 which
2064 an
2057 line
1962 command
1869 will
1628 can
1449 from
1405 system
1392 files
1391 number
1384 may
1331 used
1326 input
1308 one
1299 at
1292 output
1224 when
1188 all
1136 character
1107 program
1096 also
1082 then
1058 have
1036 list
1015 no
986 see
980 each
973 text
971 use
970 first
949 set
943 lines
929 there
921 data
901 has
900 user
865 only
860 type
847 time
832 characters
822 value
817 function
816 error
801 any
774 commands
770 terminal
730 more
725 should
722 current
721 but
719 other
717 read
715 process
682 these
681 given
668 description
663 example
654 end
649 than
649 string
649 into
648 new
645 default
642 standard
641 do
612 directory
591 must
584 so
573 shell
569 two
559 its
556 expression
552 up
550 argument
544 mode
543 after
541 print
537 unix
533 arguments
525 was
525 out
520 they
518 format
516 been
515 same
512 some
511 last
506 section
505 block
504 message
501 write
501 size
500 specified
496 we
491 programs
486 synopsis
483 does
480 space
474 case
470 next
469 your
469 return
463 using
460 option
448 names
445 table
438 left
433 before
428 point
425 like
424 call
422 special
417 where
410 following
408 make
408 control
403 such
395 information
394 array
391 thus
389 entry
381 code
380 device
377 printed
377 mail
377 functions
376 page
376 editor
375 most
370 form
368 length
366 would
349 single
346 signal
346 right
345 tape
344 macro
344 buffer
339 source
336 second
335 word
333 trace
333 change
331 options
331 bugs
327 bytes
326 between
324 get
324 called
323 now
322 returns
322 possible
320 what
319 header
317 about
313 since
313 another
311 define
309 written
309 done
307 char
306 statement
298 them
298 include
297 symbol
297 many
296 while
296 variable
293 machine
293 count
290 non
290 lisp
288 way
288 just
288 free
286 numbers
285 version
282 language
281 integer
277 etc
276 long
276 causes
276 beginning
274 made
272 blocks
271 through
267 available
266 compiler
264 zero
261 group
259 begin
256 order
253 start
253 open
252 values
252 edit
252 disk
251 found
251 contains
251 address
249 normally
249 level
246 instead
245 returned
244 stack
241 above
239 useful
237 both
236 their
236 named
236 common
235 messages
233 run
233 memory
233 flag
232 executed
230 prints
230 nil
229 manual
228 followed
227 double
227 contents
225 field
225 described
224 users
223 over
223 editing
222 changed
221 want
221 result
221 processes
220 otherwise
220 defined