Linearized Inversion for 2D Viscoacoustic Media

Joakim O. Blanch* **William W. Symes†**

Abstract

Seismic waves attenuate and disperse while propagating through the earth. These anelastic effects can be large, and can adversely affect the results of imaging and inversion if neglected. Two dimensional viscoacoustics provides a reasonably realistic model for multidimensional simulation and inversion in the presence of attenuation and dispersion. The adjoint state method leads to an effective algorithm for linear least squares inversion of short wavelength model components. However the nonreversibility of the viscoacoustic system complicates its efficient implementation, as fields must be accessed in both forward and backward order in time during the adjoint state calculation. For 2D examples of dimensions typical of exploration seismology these fields are far to large too store in fast memory. A minor modification of an algorithm due to Griewank [2] optimally balances the storage and computation resources needed to complete the adjoint state calculation.

A synthetic 2D linearized viscoacoustic inversion example illustrates the method and its efficiency.

1 Introduction

We will present an algorithm for 2D viscoacoustic linearized inversion. The algorithm is devised to estimate short scale fluctuations in bulk modulus (K) and density (ρ) about smooth reference or background fields, as well as an attenuation parameter (τ) (roughly proportional to Q^{-1}) and source parameters. We will start by briefly describing least squares linearized inversion in general and then specializing to 2D viscoacoustics. We give both continuum and discretized descriptions of the adjoint state method for computation of the gradient of the mean square error function. Then we will motivate and describe an algorithm due to Griewank [2] for implementation of the adjoint state method. This algorithm avoids the apparent (and completely impractical) requirement of storing the entire space-time history of the system for a modest computational price. We imbed this computation of the gradient in a conjugate gradient iteration for minimization of the mean square misfit. We end with a numerical example, in which we recover a bulk modulus perturbation from synthetic data.

*The Rice Inversion Project, Department of Gology and Geophysics, Rice University, Houston TX 77251-1892

†The Rice Inversion Project, Department of Computational and Applied Mathematics, Rice University, Houston TX 77251-1892

2 Linearized Inversion

The object of viscoacoustic inversion is to find an earth model, $\vec{\mu} = (K, \rho, \tau)$, which yields the best fit between the simulated seismogram, $F(\vec{\mu})$, and the measured seismogram, F_0^{obs}. Measuring the fit with the L^2 norm leads to minimizing the following functional J,

$$\begin{aligned} J &= \frac{1}{2}(F(\vec{\mu}) - F_0^{obs}, F(\vec{\mu}) - F_0^{obs}) \\ &\equiv \frac{1}{2}\int\int (F(\vec{\mu}) - F_0^{obs})^2\, dt d\vec{x}. \end{aligned} \tag{2.1}$$

$F(\vec{\mu})$ is usually a nonlinear function of the parameters $\vec{\mu}$. Assuming there exists a reasonably good estimate, $\vec{\mu}_0$, it is justified to linearize $F(\vec{\mu})$ to get a linearized map $DF(\delta\vec{\mu})$ and use it instead of F in the functional J, $\vec{\mu} = \vec{\mu}_0 + \delta\vec{\mu}$. It is of course also necessary to change F_0^{obs} to $F^{obs} = F_0^{obs} - F(\vec{\mu}_0)$. This yields,

$$J = \frac{1}{2}(DF[\delta\vec{\mu}] - F^{obs}, DF[\delta\vec{\mu}] - F^{obs})\,, \tag{2.2}$$

where square brackets are used to emphasize that $DF[\delta\vec{\mu}]$ is linear in $\delta\vec{\mu}$. Since J is a positive semi-definite form it will have a minimum when a perfect fit occurs. The gradient of J with respect to $\delta\vec{\mu}$ will therefore vanish at this point. A small variation of J yields,

$$dJ = (DF[d\delta\vec{\mu}], DF[\delta\vec{\mu}] - F^{obs})\,, \tag{2.3}$$

since DF is linear. $d\delta\vec{\mu}$ is a variation of $\delta\vec{\mu}$. By defining the adjoint of DF as DF^*, it is possible to rewrite equation (2.3) as,

$$dJ = (d\delta\vec{\mu}, DF^*[DF[\delta\vec{\mu}] - F^{obs}])\,, \tag{2.4}$$

Now it is possible to identify the gradient of J with respect to $\delta\vec{\mu}$ as,

$$Grad\, J = DF^*[DF[\delta\vec{\mu}] - F^{obs}]\,. \tag{2.5}$$

Setting this gradient to zero yields the normal equations,

$$DF^*[DF[\delta\vec{\mu}]] = DF^*[F^{obs}]\,. \tag{2.6}$$

Solving the normal equation (2.6) yields the $\delta\vec{\mu}$ that minimizes J. Since any discretization will result in a large dimensional problem, it is natural to expect to use an iterative method, for example conjugate gradient iteration. Iteration will require repeated application of the linearized forward map and corresponding adjoint, so we must decribe algorthims to accomplish these ends. We devote the next two sections to this description. The first concerns the continuous equations and second explains necessary modification for a suitable discretization.

3 The Continuous System

The full 2-D viscoacoustic wave equation can be written as,

$$
\text{(3.1)} \qquad \begin{cases} p^a_{,t} = -K^a(1+\tau^a)\vec{\nabla}\cdot\vec{v}^a - r^a + f(\vec{x}_s,t) \\ r^a_{,t} = -\dfrac{1}{\tau_\sigma}(r^a + K^a\tau^a\vec{\nabla}\cdot\vec{v}^a) \\ \vec{v}^a_{,t} = -\dfrac{1}{\rho^a}\vec{\nabla}p^a \end{cases} .
$$

Linearizing the equation, i.e., $K^a = K_0 + K$ etc., yields,

$$
\text{(3.2)} \qquad \begin{cases} p_{0,t} = -K_0(1+\tau_0)\vec{\nabla}\cdot\vec{v}_0 - r_0 + f(\vec{x}_s,t) \\ r_{0,t} = -\dfrac{1}{\tau_\sigma}(r_0 + K_0\tau_0\vec{\nabla}\cdot\vec{v}_0) \\ \vec{v}_{0,t} = -\dfrac{1}{\rho_0}\vec{\nabla}p_0 \end{cases} ,
$$

and

$$
\text{(3.3)} \qquad \begin{cases} p_{,t} = -K_0(1+\tau_0)\vec{\nabla}\cdot\vec{v} - r - [K_0\tau + r_K K_0(1+\tau_0)]\vec{\nabla}\cdot\vec{v}_0 \\ r_{,t} = -\dfrac{1}{\tau_\sigma}(r + K_0\tau_0\vec{\nabla}\cdot\vec{v} + K_0[\tau + r_K\tau_0]\vec{\nabla}\cdot\vec{v}_0) \\ \vec{v}_{,t} = -\dfrac{1}{\rho_0}\vec{\nabla}p + \dfrac{r_\rho}{\rho_0}\vec{\nabla}p_0 \end{cases} .
$$

where $r_K = K/K_0$ and $r_\rho = \rho/\rho_0$. The field p is sampled at points x_r to yield output, $p(t,x_r) \equiv u_s$. Now define $u = (p, r, \vec{v})$, $\mu = (K, \tau, \rho)$, $u_0 = (p_0, r_0, \vec{v}_0)$, and $\mu_0 = (K_0, \tau_0, \rho_0)$. The operators A and B are defined through equation (3.3) and equation (3.4),(3.5),

$$
\text{(3.4)} \qquad A(u_0,\mu_0)u = B_1(u_0,\mu_0)\mu
$$

$$
\text{(3.5)} \qquad A(u_0,\mu_0)u = B_2(\mu,\mu_0)u_0,
$$

where equation (3.5) can be used for source estimation.

The linearized model DF yields the output $u_s = p(t,x_r)$ from both/either linearized parameters and/or source. Now to find the adjoint DF^* for the input being linearized parameters or source, and the output seismogram, it is necessary to set up a scalar product between two output like quantities and rewrite the quantities so that a scalar product between two input parameters appear.

$$(DF\mu, u_s) = (\mu, DF^* u_s) \tag{3.6}$$

or

$$(DFf, u_s) = (f, DF^* u_s) \tag{3.7}$$

for source estimation. This yields,

$$\begin{aligned}
(DF\mu, u_s) &= \sum_r \sum_s \int_0^T \Delta u_s u_s \, dt \\
&= \sum_s \int_0^T \int_\Omega \sum_r \Delta u(\vec{x}) u_s \delta(\vec{x}_r - \vec{x}) \, d\vec{x} dt \\
&= \sum_s \int_0^T \int_\Omega \Delta u(\vec{x}) A^* w \, d\vec{x} dt \\
&= \sum_s \int_0^T \int_\Omega w A \Delta u(\vec{x}) \, d\vec{x} dt \\
&= \sum_s \int_0^T \int_\Omega w B_1 \mu \, d\vec{x} dt \\
&= \sum_s \int_0^T \int_\Omega \mu B_1^* w \, d\vec{x} dt \\
&= \int_\Omega \mu \left[\sum_s \int_0^T B_1^* w \, dt \right] d\vec{x} \\
&= (\mu, DF^* u_s)
\end{aligned}$$

$\Rightarrow$

$$DF^* = \sum_s \int_0^T B_1^* w \, dt \tag{3.8}$$

For source estimation a similar calculation yields,

$$DF^* = w_0(\vec{x}_s, t) \,. \tag{3.9}$$

The fields w and w_0 are defined through,

$$A^* w = u_s \tag{3.10}$$

and

$$A^* w_0 = B_2^* w \,. \tag{3.11}$$

w_0 is an extra adjoint field that has to be defined to calculate the adjoint for sources inserted into the linearized equation.

Now it is necessary to find the explicit adjoints of A and B. The explicit expression for A is,

$$A = \begin{pmatrix} \frac{\partial}{\partial t} & 1 & K_0(1+\tau_0)\vec{\nabla}\cdot \\ 0 & \frac{\partial}{\partial t} + \frac{1}{\tau_\sigma} & \frac{1}{\tau_\sigma} K_0 \tau_0 \vec{\nabla}\cdot \\ \frac{1}{\rho_0}\vec{\nabla} & 0 & \frac{\partial}{\partial t} \end{pmatrix} \tag{3.12}$$

This matrix is actually a 4×4 matrix since $u = (p, r, \vec{v})$ and $w = (q, s, \vec{u})$. This yields,

$$A^* = -\begin{pmatrix} \frac{\partial}{\partial t} & 0 & \vec{\nabla} \cdot \frac{1}{\rho_0} \\ -1 & \frac{\partial}{\partial t} - \frac{1}{\tau_\sigma} & 0 \\ \vec{\nabla} K_0(1+\tau_0) & \vec{\nabla}\frac{1}{\tau_\sigma} K_0 \tau_0 & \frac{\partial}{\partial t} \end{pmatrix} \tag{3.13}$$

B_1 is equal to,

$$B_1 = -\begin{pmatrix} K_0(1+\tau_0)(\vec{\nabla}\cdot\vec{v}_0) & K_0(\vec{\nabla}\cdot\vec{v}_0) & 0 \\ \frac{1}{\tau_\sigma} K_0 \tau_0 (\vec{\nabla}\cdot\vec{v}_0) & \frac{1}{\tau_\sigma} K_0 (\vec{\nabla}\cdot\vec{v}_0) & 0 \\ 0 & 0 & -\frac{1}{\rho_0}(\vec{\nabla} p_0) \end{pmatrix}, \tag{3.14}$$

which adjoint is clear. The linearized parameters are here defined as $r_K = K/K_0$ and $r_\rho = \rho/\rho_0$. The explicit form for B_2 is,

$$B_2 = -\begin{pmatrix} 0 & 0 & K_0\left[r_K(1+\tau_0)+\tau\right]\vec{\nabla}\cdot \\ 0 & 0 & \frac{1}{\tau_\sigma} K_0\left[r_K \tau_0 + \tau\right]\vec{\nabla}\cdot \\ -\frac{r_\rho}{\rho_0}\vec{\nabla} & 0 & 0 \end{pmatrix}, \tag{3.15}$$

and the adjoint B_2^* is clearly,

$$B_2^* = \begin{pmatrix} 0 & 0 & \vec{\nabla}\cdot -\frac{r_\rho}{\rho_0} \\ 0 & 0 & 0 \\ \vec{\nabla} K_0\left[r_K(1+\tau_0)+\tau\right] & \vec{\nabla}\frac{1}{\tau_\sigma} K_0\left[r_K\tau_0+\tau\right] & 0 \end{pmatrix}. \tag{3.16}$$

3.1 Boundary and Initial (Final) Conditions

The boundary conditions for A have so far been ignored. A standard Dirichlet boundary is used for all boundaries except $z = 0$, where a free fluid surface boundary condition is used. For a 2-D model $\vec{x} = (x, z)$, this yields

$$p = r = v_x = 0 \tag{3.17}$$

$$v_{z,z} = 0 \tag{3.18}$$

at $z = 0$. The $z = 0$ boundary condition is easily found through the method of images. The boundary conditions can for the corresponding adjoint fields, for instance, be,

$$q = s = u_x = 0 \tag{3.19}$$

$$u_{z,z} = 0 \tag{3.20}$$

at $z = 0$ and Dirichlet on the other boundaries. There are several other choices for the boundary conditions for the continuous system, but the above boundary condition is necessary for the discrete adjoint system (discussed below).

We remind the reader that absorbing boundaries are easily and effectively modeled simply by reducing Q (increasing τ) around the (Dirichlet) boundaries.

A corresponds to an initial value problem and A^* to a final value problem, i.e. the fields u_0 and u (associated with A) are solved forward in time and the fields w and w_0 (associated with A^*) are solved backward in time. Thus the initial/final conditions are $u = u_0 = 0$ for $t \ll 0$ and $w = w_0 = 0$ for $t \gg T$ (the fields u and u_0 are solved up to time T, see also Equation (3.8)).

4 The Discrete System

The only real difficulty in switching from continuous to discrete equations is the operator A. For the discrete system it is possible to write A as,

$$A = \begin{pmatrix} D_{\Delta t} & M_{1/2} & K_0(1+\tau_0)D_{\Delta x} & K_0(1+\tau_0)D_{\Delta z} \\ 0 & D_{\Delta t} + \frac{1}{\tau_\sigma}M_{1/2} & \frac{1}{\tau_\sigma}K_0\tau_0 D_{\Delta x} & \frac{1}{\tau_\sigma}K_0\tau_0 D_{\Delta z} \\ \frac{1}{\rho_0}D_{\Delta x} & 0 & D_{\Delta t} & 0 \\ \frac{1}{\rho_0}D_{\Delta z} & 0 & 0 & D_{\Delta t} \end{pmatrix}, \tag{4.1}$$

where $D_{\Delta t}$ is a discrete approximation to a time derivative, $D_{\Delta x}$, $D_{\Delta z}$ are discrete approximations to spatial derivatives, and $M_{1/2}$ is an averaging operator between two time levels. It is only necessary to find the adjoint of the individual components of the discrete A. As a matter of fact all the components have identical counter parts for 1-D and the problem is already solved in 1-D [1]. The only remaining problem is the free surface boundary condition. The boundary condition is solved for the forward model by the method of images. Thus, the fields for p, r, and v_x are found to be odd functions with the respect to $z = 0$ and v_z an even function with respect to $z = 0$. This applies to the discrete fields as well and has the effect for the adjoint fields $w = (q, s, u_x, u_z)$ that u_z is even, and q, s, and u_x are odd functions with respect to $z = 0$ [3]. The spatial derivatives in B_1 and B_2 are natuarally replaced by appropriate combinations of discrete approximations ($D_{\Delta x}$ and/or $D_{\Delta z}$).

5 Practical Implementation Considerations

The most difficult practical problem arising in calculating the adjoint (3.8) is the necessity to simultaneously (in time) know both the adjoint fields w and the background forward fields u_0 (see

Equations (3.8) and (3.14)). Note that the former is computed backwards in time, whereas the latter is computed forwards in time. Conventional time-marching schemes to compute these fields store only a few time levels, which will only rarely coincide in time in the usual arrangement of these loops. Therefore evaluation of the right-hand side of (3.8) is impossible without some change of approach.

There are three obvious ways to solve the problem, each of course having both drawbacks and advantages,

- Compute the background fields and save every time level. Thus no time levels need be recomputed. We will use this "store it all" algorithm as a unit of CPU cost, against which to compare the computational cost of other approaches. A *lot* of memory is needed, so that this scheme is really only practical when the storage cost for a single time level of the fields is approximately zero (i.e. in problems with one space dimension). Sparse storage schemes have been used to reduce the required memory for this approach, but to keep the threshhold high enough to significantly reduce storage seems likely to degrade the accuracy of the computed gradient to the extent that it can no longer be used reliably in conjugate gradient iteration, for example.

- Compute the background fields forward in time up to the desired time and then compute the background fields backwards in time simultaneously with the adjoint fields. This is an elegant solution that demands an energy conserving operator A. In fortunate situations (very little energy dissipation) it might be possible to proceed in this fashion with non-conservative systems. However we shall see below that this does not appear to be the case for viscoacoustics with realistic parameters.

- Compute the background fields and save them at a few (but well chosen!) time levels. During the calculation of the adjoint these *snapshots* can be used as initial conditions to recompute the background fields up to a desired time level.

To save every time level of the computation demands extreme amounts of memory. For example, a reasonable 2D experiment, with a grid consisting of 700x1000 gridpoints and where the fields are computed for 1000 time levels, would require approximately 10 GB of memory (4 fields need to be saved in each time level). Evidently analogous 3D computations are altogether out of the question (distinctly not the case for elasticity, where the second alternative mentioned above is available).

The best solution is clearly to compute the fields backwards in time, as is possible for example for $Q = \infty$. The viscoelastic wave equation is not energy conserving, though, so this is not possible. Figure 5.1 shows a 1D example where the viscoelastic wave equation has been solved backwards in time. The actual wave has only increased by 20 % (Figure 5.2), but the very high frequency error components are obviously exploding. This occurs because the memory variable (r) is rapidly decaying, i.e. rapidly increasing when solved backwards, and is strongly coupled to the high frequency wave components. Thus, when the equation for the memory variable is solved backwards it explodes and contaminates the other fields (p, $\vec{v}$) as well.

The only remaining alternative is to save some time levels and recompute the rest. It is not an easy task to decide how many time levels to save and how many to recompute, however, to optimize both memory use and CPU cycles. To illustrate, suppose that one needs $n+1$ time levels,

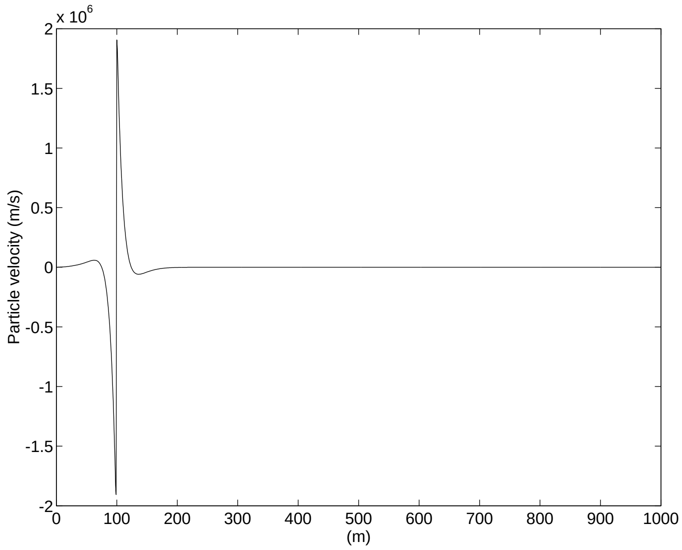

Figure 5.1: The background (particle velocity) field solved backwards, after 700 time levels (140 ms). The wave was inserted at 100 m. It is hardly possible to see the correct wave located at approximately 350 m. In the simulation the gridspacing was 1 m, the time level 0.2 ms, $Q \approx 100$, the wave velocity 2.0 km/s, and the density 2600 kg/m^3.

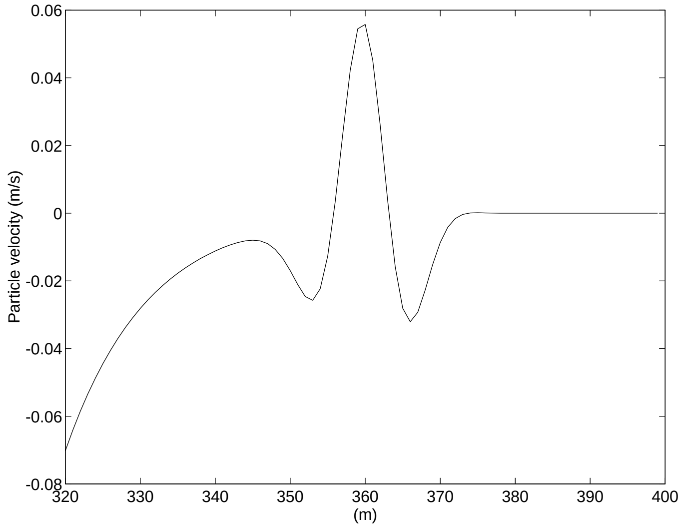

Figure 5.2: A close-up of the correct wave. It is approximately 20 % larger than when it was originally inserted. The field is starting to explode after the wave has passed (in the interval 320-340 m).

and computes them in the forward direction, saving levels

$$n+1-\frac{k(k+1)}{2},\ k=1,2,3,...$$

One easily verifies that there are $O(\sqrt{n})$ saved levels in such a scheme. The access of the field backward in time (for example in evaluating the right hand side of (3.8)) proceeds as follows:

1. use level n;
2. recompute level $n-1$ starting with initial condition at level $n-2$, store over level n;
3. use levels $n-2$, $n-1$;
4. recompute levels $n-4$, $n-3$, starting with initial condition at level $n-5$, store over $n-2$, $n-1$;
5. use levels $n-5$, $n-4$, $n-3$;
6. etc. etc.

In this way, by overwriting time levels already used in (3.8) with time levels recomputed from earlier, saved ones, each time level is computed exactly twice, and the total storage required is $O(\sqrt{n})$.

Although much better than the $O(n)$ storage required by the naive "store it all" scheme, the algorithm just described still generates formidable demands for memory resources in problems of realistic size. Thus one naturally asks: is it possible to reduce the memory requirements still further, by exchanging storage for CPU cycles? Griewank [2] has answered this question in the affirmative. His algorithm is a refinement of the one just described. We shall not describe it in detail here. NEED TO DO!!!!!!!!!!!!!!! Griewank's algorithm yields an evaluation of the right hand side of (3.8) using $O(\log n)$ recomputations and $O(\log n)$ time levels to be stored.

The efficiency achieved by Griewank's algorithm is truly remarkable. For example, for up to roughly $n = 10,000$, storage of 10 time levels allows the evaluation of the right hand side of (3.8) with roughly 3 times the CPU cycles of the "store it all" scheme. If the available storage is reduced to 5 time levels, the number of floating point operations is still only 5 times that of the "store it all" scheme. Clearly Griewank's approach permits very efficient use of available memory resources.

Griewank's original algorithm determines the time levels to be saved dynamically during the forward computation of the reference fields. This feature is useful for example for application to variable step ODE solvers, for which the number of intermediate time levels necessary to reach a given final time is unknown before the computation is completed. A price paid for this flexibility is that only certain values of n are terminal. In contrast, the methods used in our solution of the viscoacoustic initial/boundary value problem employ fixed time steps. The knowledge of the number of time levels beforehand allows us to determine statically the time levels to be saved, hence to adjust the method to terminate at any prescribed level n.

We have implemented this modification of Griewank's algorithm in the viscoacoustic context explained above, and verified that the approximate right hand side of (3.8) so computed satisfies the necessary adjointness relations to machine precision, as is required for successful use in conjugate gradient iteration and related algorithms.

6 Example

We produced a synthetic linearized seismogram using the target bulk modulus perturbation shown in Figure 6.1. The velocity was 1.5 km/s, the density 2600 kg/M^3, and Q approximately 120. The sythetic seismogram consisted of 20 shotrecords, of which the oddnumbered shots are shown in Figure 6.2. The target bulk modulus perturbation varied laterally in magnitude. The effect of this amplitude variation is clearly evident in the seimogram. The result after 20 steps of conjugate residual is shown in Figure 6.3. The fit error decreased by 98 %. The difference in appearance between the target and estimated bulk modulus perturbations is due to the bandlimited nature of the source wavelet. Its frequency content does not match the spatial frequency of the target bulk modulus perturbation, as is indicated by smaller amplitudes and an larger relative content of high frequencies in the estimated bulk modulus perturbation, compared to the target. Of course thi inversion with a bandlimited source - hence of any realistic inversion! The general feature of the target (laterally varying magnitude) is still well reproduced in the estimated bulk modulus perturbation.

7 Conclusion

We have presented an algorithm for estimation bulk modulus, density, attenuation, and source in heterogeneous 2D viscoacoustic media. We showed an example in which we estimated short scale fluctuations in bulk modulus from synthetic data with good results.

It is clearly necessary to use a recomputation scheme to able to carry out the adjoint state method for the linearized 2D viscoacoustic wave equation. The scheme suggested by Griewank [2] can be adjusted for arbitrary snapshot size and maximum timelevel. This scheme appears to yield an optimal program of recomputation for given memory resources.

The code has performed well on IBM RS6000, CRAY YMP and CRAY C90 computers.

8 Acknowledgments

We would like to thank our friends Alain Sei for reviewing the derivation of the adjoint and Roelof Versteeg for providing us with the laterally varying bulk modulus reflectivity model. This research was supported by ONR grant N00014-89-J1115, the Schlumberger Foundation, and The Rice Inversion Project. TRIP Sponsors for 1994 are Advance Geophysical, Amoco Production Co., Conoco Inc., Cray Research Inc., Exxon Production Research Co., Interactive Network Technologies, Mobil Research and Development Corp., and Texaco Inc.

References

[1] J. O. Blanch and W. W. Symes, Linearized viscoelastic inversion,. In O. Diachok, A. Caiti, P. Gerstoft, and H. Schmidt, editors, *Full field inversion methods in ocean and seismic acoustics*, to appear. Kluwer Academic Publishers, 1994.

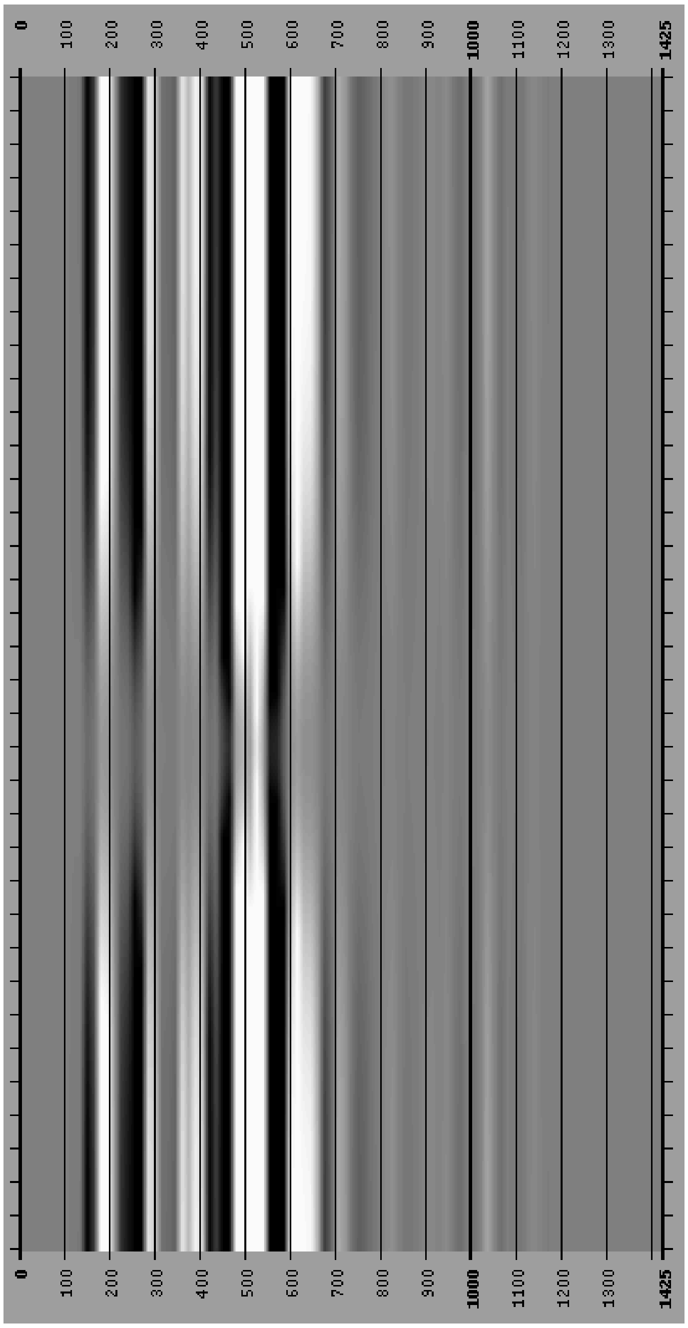

Figure 6.1: The spatially varying bulk reflectivity used in the example.

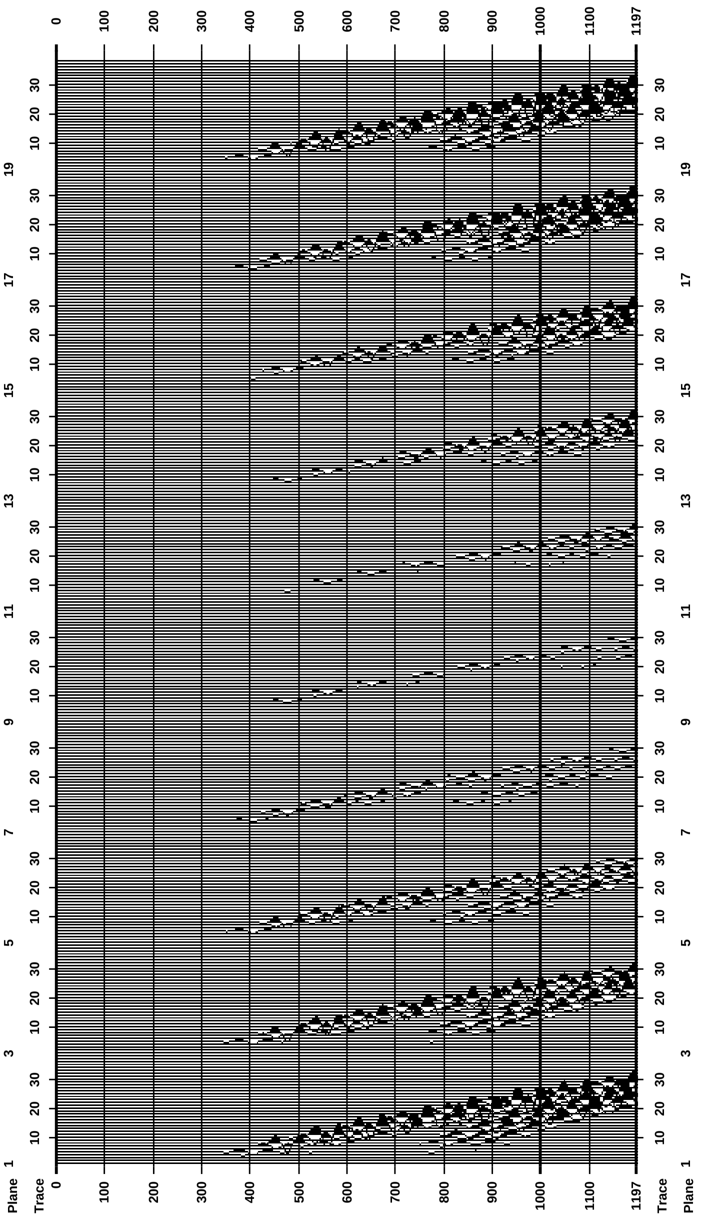

Figure 6.2: The synthetic seismogram.

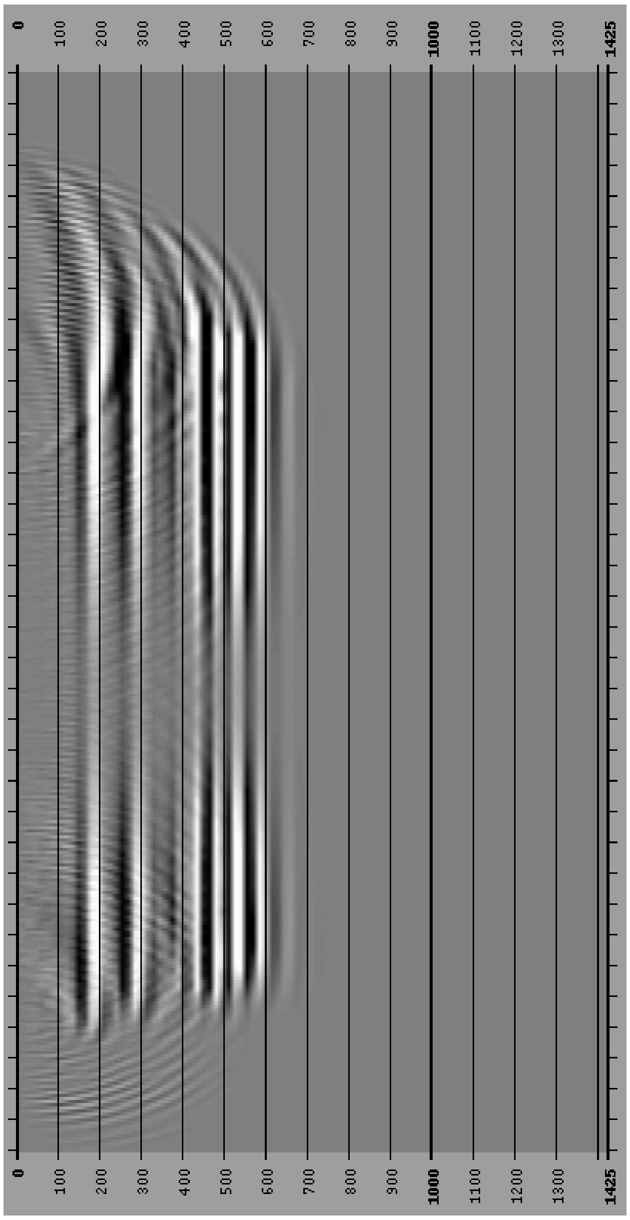

Figure 6.3: The estimated bulk reflectitivty. The difference in appearance between the original bulk reflectivity and estimated is probably due to a badly chosen source wavelet, which frequency content does not match the spatial frequency of the model reflectivity.

[2] A. Griewank, Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. *Optimization Methods and Software*, 1:35–54, 1992.

[3] A. Sei, Calculation of the gradient of the DSO objective function. *The Rice Inversion Project annual report*, 1,article 12, 1993.