Elastic finite differences with convolutional operators

Peter Mora

ABSTRACT

The solution of the two dimensional elastic wave equation is well known by the finite difference method. Practical points of interest are how to parametrize the elastic wave equation and how to design the spatial derivative operator. A perfect spatial derivative operator can be achieved by Fourier transforming, multiplying by ik and inverse transforming. This requires the minimum computer storage. Alternatively, the derivative can be computed by a convolution in the space domain. As the length of the space domain derivative operator becomes greater, the accuracy improves and there is less grid dispersion. The cost of this improved accuracy is more CPU and finally, there is a length where the convolutional operator becomes slower than the Fourier derivative. If the size of the problem is fixed, say a simulation in a 4km by 2km block with velocities and frequencies such that the minimum wavelength is .1 km, then to obtain a desired accuracy there is some hardware dependent optimal derivative operator. On the CRAY-1 and CONVEX-C1 the optimum is a convolutional operator that is about eight points long (the Fourier derivative method is significantly slower). I present an easy way of designing convolutional derivative operators *not* based on Taylor's expansions. Also, I describe a method of elastic finite differences based on the stress equations of Kosloff et al. but where the derivative operators are centered half way between gridpoints enabling more accurate short derivative operators to be designed. Centering half way between gridpoints is also useful when using perfect Fourier derivatives because the space domain representation is more local thereby decreasing annoying spatial Nyquist energy.

Introduction

The 2D elastic wave equation can be solved numerically using an explicit finite difference (f.d.) scheme where the spatial derivatives are computed by either convolution with a space domain derivative operator or by Fourier transforming over space, multiplying by ik and inverse transforming (Kosloff, Reshef and Loewenthal, 1984 use the Fourier method). The advantage of using Fourier derivatives or long convolutional operators is that the spatial derivatives are accurate to about the spatial Nyquist so the grid size required for the finite difference computations is greatly reduced making larger problems feasible. For example, if a two point operator were used then at least 10 gridpoints per wavelength are required to achieve an adequate accuracy while if Fourier derivatives or a long convolutional operator were used only about 2.5 gridpoints per wavelength are necessary. Hence, for a two-dimensional problem the two point scheme requires 16 times more memory. As the operator length is increased the CPU increases but this is roughly in proportion to the increase in accuracy at least up to a length of eight points. Actually, an eight point operator takes four times the CPU but is only 3 times as accurate so performing a derivative takes about 33% longer than a two point operator scheme. However, when the operator is short, the time differencing takes a significant portion of the CPU. Therefore, two point operator f.d. schemes are probably slower than the eight point operator f.d. schemes on most computers. Furthermore, as the grid spacing Δx is decreased, the time step Δt must also be decreased to achieve stability so more time steps may be required for a two point scheme to solve the same forward problem. To summarize, relative than a two point scheme, an eight point scheme requires 16 times less memory and is normally faster. Therefore, eight point schemes are superior. Note that they are also faster than the Fourier method on the CONVEX-C1 and CRAY-1S computers.

Convolutional operators can easily be designed by inverse transforming the perfect frequency domain operator ik, truncating to the desired length, and multiplying by a Gaussian curve to taper the edges. This is simpler than the Taylor's expansion approach which requires some tedious algebra. The following paper describes an elastic finite difference scheme and illustrates the use of convolutional operators designed by the above method.

2D elastic wave equation

The basic equations are describing conservation of momentum in a 2D elastic medium are

$$\frac{1}{\rho}(\sigma_{xx,x} + \sigma_{xz,z} - f_x) = \ddot{u}_x \tag{1a}$$

$$\frac{1}{\rho}(\sigma_{xz,x} + \sigma_{zz,z} - f_z) = \ddot{u}_z \tag{1b}$$

where subscripts after the comma indicate differentiation. Using the relationship between strain and displacement given by

$$e_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}) \tag{2}$$

we obtain

$$(\frac{1}{\rho}\sigma_{xx,x} - \frac{1}{\rho}f_x)_{,x} + (\frac{1}{\rho}\sigma_{xz,z} - \frac{1}{\rho}f_x)_{,x} = \ddot{e}_{xx} \tag{3a}$$

$$(\frac{1}{\rho}\sigma_{xz,x} - \frac{1}{\rho}f_z)_{,z} + (\frac{1}{\rho}\sigma_{zz,z} - \frac{1}{\rho}f_z)_{,z} = \ddot{e}_{zz} \tag{3b}$$

$$(\frac{1}{\rho}\sigma_{xx,x} + \frac{1}{\rho}\sigma_{xz,z} - \frac{1}{\rho}f_x)_{,z} + (\frac{1}{\rho}\sigma_{xz,x} + \frac{1}{\rho}\sigma_{zz,z} - \frac{1}{\rho}f_z)_{,x} = 2\ddot{e}_{xz} \tag{3c}$$

For an isotropic medium we have the stress strain relationship (Hooke's law)

$$\sigma_{xx} = (\lambda + 2\mu)e_{xx} + \lambda e_{zz} \tag{4a}$$

$$\sigma_{zz} = \lambda e_{xx} + (\lambda + 2\mu)e_{zz} \tag{4b}$$

$$\sigma_{xz} = 2\mu e_{xz} \tag{4c}$$

Algorithm

These equations may be solved as follows

for all time {

$\sigma(t) \rightarrow \ddot{u}(t)$ *from the elastic wave equation, eq. (1)*

$\ddot{u}(t) \rightarrow \ddot{e}(t)$ *from the strain-displacement relation, eqs. (3) and (4)*

$\ddot{e}(t) \rightarrow \ddot{\sigma}(t)$ *from Hooke's Law, eq. (4)*

$\ddot{\sigma}(t) \rightarrow \sigma(t+\Delta t)$ *step forward in time using an explicit finite difference formula*

}

Below is a detailed description of this algorithm including the place at which the horizontal and vertical displacements as well as the P and S potentials can be output. Note that (t,x,z) is used to denote functional dependencies of the variables on time and the space coordinates. Note that the functions are discretely sampled so t, x and z can take on integral values only (for example *time* $= t\ \Delta t$ where Δt denotes the time step). To avoid cluttering I only include the specific functional dependencies when they are non integral (e.g. $f(x+\frac{1}{2}) = f(t,x+\frac{1}{2},z)$ and $g = g(t,x,z)$ etc.). The algorithm is interesting because it tells how to use derivative operators that are centered half way between gridpoints.

$$\sigma_{xx} = 0 \quad , \quad \sigma_{zz} = 0 \quad , \quad \sigma_{xz}(x-\tfrac{1}{2},z-\tfrac{1}{2}) = 0$$

$$\dot{\sigma}_{xx} = 0 \quad , \quad \dot{\sigma}_{zz} = 0 \quad , \quad \dot{\sigma}_{xz}(x-\tfrac{1}{2},z-\tfrac{1}{2}) = 0$$

for all time {

$$\chi_{xx,x}(x-\tfrac{1}{2}) = \frac{1}{\rho}\hat{\partial}_x(x+\tfrac{1}{2})\sigma_{xx}$$

$$\chi_{zz,z}(z-\tfrac{1}{2}) = \frac{1}{\rho}\hat{\partial}_z(z+\tfrac{1}{2})\sigma_{zz}$$

$$\chi_{xz,z}(x-\tfrac{1}{2}) = \frac{1}{\rho}\hat{\partial}_z(z-\tfrac{1}{2})\sigma_{xz}(x-\tfrac{1}{2},z-\tfrac{1}{2})$$

$$\chi_{xz,x}(z-\tfrac{1}{2}) = \frac{1}{\rho}\hat{\partial}_x(x-\tfrac{1}{2})\sigma_{xz}(x-\tfrac{1}{2},z-\tfrac{1}{2})$$

$$\ddot{u}_x(x-\tfrac{1}{2}) = \chi_{xx,x}(x-\tfrac{1}{2}) + \chi_{xz,z}(x-\tfrac{1}{2})$$

$$\ddot{u}_z(z-\tfrac{1}{2}) = \chi_{xz,x}(z-\tfrac{1}{2}) + \chi_{zz,z}(z-\tfrac{1}{2})$$

$$\ddot{u}_z(z-\tfrac{1}{2}) = \ddot{u}_z(z-\tfrac{1}{2}) - \frac{1}{\rho(z-\frac{1}{2})}\sum_S s_z(t,S)\hat{\delta}(x-x_S, z-\tfrac{1}{2}-z_S) \qquad \textit{add vertical source}$$

$$\ddot{u}_x(x-\tfrac{1}{2}) = \ddot{u}_x(x-\tfrac{1}{2}) - \frac{1}{\rho(x-\frac{1}{2})}\sum_S s_x(t,S)\hat{\delta}(x-\tfrac{1}{2}-x_S, z-z_S) \qquad \textit{add horizontal source}$$

$$\ddot{u}_z(z-\tfrac{1}{2}) = \ddot{u}_z(z-\tfrac{1}{2}) - \frac{1}{\rho(z-\frac{1}{2})}\sum_S s_P(t,S)\partial_z\hat{\delta}(x-x_S, z-\tfrac{1}{2}-z_S) \qquad \textit{add pressure source}$$

$$\ddot{u}_x(x-\tfrac{1}{2}) = \ddot{u}_x(x-\tfrac{1}{2}) - \frac{1}{\rho(x-\frac{1}{2})}\sum_S s_P(t,S)\partial_x\hat{\delta}(x-\tfrac{1}{2}-x_S, z-z_S) \qquad \textit{add pressure source}$$

$$\ddot{u}_z(z-\tfrac{1}{2}) = \ddot{u}_z(z-\tfrac{1}{2}) + \frac{1}{\rho(z-\frac{1}{2})}\sum_S s_S(t,S)\partial_x\hat{\delta}(x-x_S, z-\tfrac{1}{2}-z_S) \qquad \textit{add shear source}$$

$$\ddot{u}_x(x-\tfrac{1}{2}) = \ddot{u}_x(x-\tfrac{1}{2}) - \frac{1}{\rho(x-\frac{1}{2})}\sum_S s_S(t,S)\partial_z\hat{\delta}(x-\tfrac{1}{2}-x_S, z-z_S) \qquad \textit{add shear source}$$

$$\textit{output} \quad \left\{ \ddot{u}_x = \ddot{u}_x(x-\tfrac{1}{2}) * \hat{\delta}(x-\tfrac{1}{2}),\ \ddot{u}_z = \ddot{u}_z(z-\tfrac{1}{2}) * \hat{\delta}(z-\tfrac{1}{2}) \right\}$$

$$\ddot{u}_{x,x} = \hat{\partial}_x(x-\tfrac{1}{2})\ddot{u}_x(x-\tfrac{1}{2})$$

$$\ddot{u}_{x,z}(x-\tfrac{1}{2}, z-\tfrac{1}{2}) = \hat{\partial}_z(z+\tfrac{1}{2})\ddot{u}_x(x-\tfrac{1}{2})$$

$$\ddot{u}_{z,z} = \hat{\partial}_z(z-\tfrac{1}{2})\ddot{u}_z(z-\tfrac{1}{2})$$

$$\ddot{u}_{z,x}(x-\tfrac{1}{2}, z-\tfrac{1}{2}) = \hat{\partial}_x(x+\tfrac{1}{2})\ddot{u}_z(z-\tfrac{1}{2})$$

$$\left.\begin{array}{l} \textit{output} \quad \left\{ \phi_p \;=\; \nabla\cdot\ddot{\mathbf{u}} \;=\; \ddot{u}_{x\,,x} \,+\, \ddot{u}_{z\,,z} \right\} \\ \textit{output} \quad \left\{ \phi_s \;=\; \nabla\times\ddot{\mathbf{u}} \;=\; (\ddot{u}_{x\,,z}(x-\tfrac{1}{2},z-\tfrac{1}{2}) - \ddot{u}_{z\,,x}(x-\tfrac{1}{2},z-\tfrac{1}{2}))^{**}\,\hat{\delta}(x-\tfrac{1}{2},z-\tfrac{1}{2}) \right\} \\ \\ \ddot{e}_{xx} \;=\; \ddot{u}_{x\,,x} \\ \ddot{e}_{zz} \;=\; \ddot{u}_{z\,,z} \\ \ddot{e}_{xz}(x-\tfrac{1}{2},z-\tfrac{1}{2}) \;=\; \dfrac{1}{2}(\ddot{u}_{x\,,z}(x-\tfrac{1}{2},z-\tfrac{1}{2}) + \ddot{u}_{z\,,x}(x-\tfrac{1}{2},z-\tfrac{1}{2})) \\ \\ \ddot{\sigma}_{xx} \;=\; (\lambda+2\mu)\ddot{e}_{xx} \,+\, \lambda\ddot{e}_{zz} \\ \ddot{\sigma}_{zz} \;=\; \lambda\ddot{e}_{xx} \,+\, (\lambda+2\mu)\ddot{e}_{zz} \\ \ddot{\sigma}_{xz}(x-\tfrac{1}{2},z-\tfrac{1}{2}) \;=\; 2\mu(x-\tfrac{1}{2},z-\tfrac{1}{2})\ddot{e}_{xz}(x-\tfrac{1}{2},z-\tfrac{1}{2}) \\ \\ \dot{\sigma}_{xx}(t+\tfrac{1}{2}) \;=\; \dot{\sigma}_{xx}(t-\tfrac{1}{2}) + \Delta t\,\ddot{\sigma}_{xx} \\ \\ \dot{\sigma}_{zz}(t+\tfrac{1}{2}) \;=\; \dot{\sigma}_{zz}(t-\tfrac{1}{2}) + \Delta t\,\ddot{\sigma}_{zz} \\ \\ \dot{\sigma}_{xz}(t+\tfrac{1}{2},x-\tfrac{1}{2},z-\tfrac{1}{2}) \;=\; \dot{\sigma}_{xz}(t-\tfrac{1}{2},x-\tfrac{1}{2},z-\tfrac{1}{2}) + \Delta t\,\ddot{\sigma}_{xz}(x-\tfrac{1}{2},z-\tfrac{1}{2}) \\ \\ \sigma_{xx}(t+1) \;=\; \sigma_{xx} \,+\, \Delta t\,\dot{\sigma}_{xx}(t+\tfrac{1}{2}) \\ \\ \sigma_{zz}(t+1) \;=\; \sigma_{zz} \,+\, \Delta t\,\dot{\sigma}_{zz}(t+\tfrac{1}{2}) \\ \\ \sigma_{xz}(t+1,x-\tfrac{1}{2},z-\tfrac{1}{2}) \;=\; \sigma_{xz}(x-\tfrac{1}{2},z-\tfrac{1}{2}) + \Delta t\,\dot{\sigma}_{xz}(t+\tfrac{1}{2},x-\tfrac{1}{2},z-\tfrac{1}{2}) \\ \\ \textit{absorbing boundaries} \\ \\ \textit{free surface} \end{array}\right\} \qquad (5)$$

Note that the values $s_x(t,S)$, $s_z(t,S)$, $s_P(t,S)$ and $s_S(t,S)$ are the source time functions respectively for vertical, horizontal, pressure and shear sources for shot number S located at (x_S, z_S). Note also that the output obtained by algorithm (5) is actually for the second time derivative of displacement, $\ddot{u}$ rather than displacement. This is most easily taken into account by using the second integral of the desired source time history as the source function, i.e. use $\int dt \int dt s_\alpha(t,S)$ rather than $s_\alpha(t,S)$ in algorithm (5) (where $\alpha = x, z, P$ or S).

Sources

The introduction of the approximate δ-function sources $\hat{\delta}$ in algorithm (5) requires some special care. If a force is applied at a single point on the finite difference mesh we are really applying a spatially band-limited source with sharp cutoffs at the spatial Nyquist frequency. This is undesirable because the sharp spatial frequency cutoff results in a non-local *sinc* source distribution and consequently the appearance of significant Nyquist energy. This effect is aggravated because the δ-functions in algorithm (5) are located half way between gridpoints and derivatives of delta functions are required for pressure and shear sources. Kosloff et al. 1984 suggest using a bell-shaped source distribution as a band limited approximation to the δ-function's to decrease this undesirable Nyquist energy and make the source

$$\delta(x,z) = \exp -\beta^2(x^2+z^2) \tag{6}$$

Kosloff et al. 1984 recommend a value of β=1. An alternative is a cubic spline impulse response which is very local and has a fairly flat amplitude spectrum almost to Nyquist.

Spatial derivative operators

The approximate derivative operators $\hat{\partial}$ in algorithm (5) should be chosen depending on hardware to maximize efficiency. They may be either the perfect Fourier derivative operators, i.e.

$$\hat{\partial}_j = \partial_j * = F^{-1} i k_j F \quad , \tag{7a}$$

where F denotes Fourier transformation or they may be convolutional operators, i.e.

$$\hat{\partial}_j = \hat{\hat{\partial}}_j * \quad . \tag{7b}$$

An easy way to design convolutional operators without tedious Taylor's expansions is as follows: Start with the ideal frequency domain response ik. Inverse transform to the

space domain and truncate to the desired length. The corresponding frequency domain response $i\hat{k}$ is obtained by convolving ik by a sinc function. Therefore, $i\hat{k}$ looks like ik with bumps in it. These bumps would cause significant grid dispersion in any finite difference scheme (and even instability) unless the truncation length was very large. The velocity as a function of frequency is $v(k) = \hat{k}/k$ so we desire $v(k) \approx 1$ for minimal grid dispersion. The bumps can be smoothed out by "low pass filtering" the bumpy $i\hat{k}$. In other words, the space domain operator must be multiplied by a smooth Gaussian curve to taper the edges.

The resulting operators would still not be very accurate if the space domain representation of the ik operator was centered at x =0. This is because its space response that dies off very gradually and so there are still truncation effects, i.e. bumps in $i\hat{k}$. However, if the derivative is centered at $x = \frac{1}{2}$ the space domain response dies off much more rapidly and so is more local and hence $i\hat{k}$ would have smaller bumps. The reason is that the $\left| ik \exp(ik\,\Delta x/2) \right| = k$ all the way from zero frequency to Nyquist while $\left| ik \right| = k$ up to but not including Nyquist, the Nyquist frequency has zero energy. For example, convolution of a two point operator centered at $x = 0$ with $f(x)$ yields $(f(x+\frac{1}{2}) - f(x-\frac{1}{2}))/(2\Delta x)$ while convolution of an operator centered at $x = -\frac{1}{2}$ yields $(f(x+\frac{1}{2}) - f(x))/\Delta x$ which is accurate to twice the frequency. Therefore, the convolutional operators are centered half way between gridpoints, at either $\Delta x/2$ or $-\Delta x/2$ depending on where they are to be applied in the finite difference algorithm.

Figure 1a shows the frequency domain response of a perfect derivative operator, an 8 point operator and a conventional 2 point operator (the perfect operator is centered at $x = 0$ in the space domain while the two convolutional operators were centered at $x = \frac{1}{2}$). The 8 point operator is very accurate to about 3/4 of the Nyquist while the two point operator is only accurate to about 1/4 of the Nyquist. The same operators are shown in the space domain in figure 1b. Note that if Fourier derivatives were more efficient on a particular computer then they should also be centered at $x = \frac{1}{2}$ to make the tails die more rapidly thereby avoiding the Nyquist artifacts observed by Kosloff et al..

Boundaries

If the Fourier derivatives are used, the medium is cyclic so the only easy way to include absorbing boundaries is to have a region of absorption surrounding the zone of interest. This approach also has the advantage that there are no angle restrictions so waves entering the absorbing region at any angle are absorbed. This method is very useful even if Fourier derivatives are not used because it avoids complicated specialized coding of non-reflecting boundaries. This is a massive headache when the derivative operator may be any length (say eight points) depending on computer hardware. It is complicated enough even in the acoustic case with two point derivative operators (see Clayton and Engquist, 1980, solved this problem using the one-way wave equations of Claerbout, 1976).

For the above reasons, an absorbing region is preferred rather than an explicit boundary condition. Cerjan et al. 1985 suggest simply multiplying the variables σ and $\dot{\sigma}$ by an attenuation factor at each time step. Their attenuation function has the Gaussian form

$$A \;=\; \exp -\gamma(x - x_A)^2 \tag{8}$$

and typically grades from 1 at the start of the attenuating region x_A to a minimum value of about .85 or .9 . Tests indicate that a value of γ=.02 works reasonably well with an attenuating border region that is 20 gridpoints wide. The attenuation is applied wherever absorbing boundaries are required and so at most 40 extra gridpoints are required in both directions. Thus, CPU time increases by around 40% for typical sized problems, a small price to pay for the ease of coding (perhaps efficiency could be improved by using shorter derivative operators in the attenuation region where grid dispersion is irrelevant). Etgen (pers. comm.) suggests an improvement can be obtained by gradually decreasing the velocities in the attenuating region (while increasing the density thereby keeping the impedances fixed to minimize the weak reflection amplitudes). Then the waves would spend more time in the boundary region and so suffer a greater attenuation.

Stability, accuracy and time differencing

Kosloff et al. 1984 give the stability criterion for the perfect Fourier derivative

$$\Delta t \;\leq\; \frac{min\,(\Delta x\,,\Delta z\,)}{V_{\min}} \tag{9}$$

However, to avoid excessive numerical dispersion (finite difference inaccuracy) they

suggest a choice of Δt which is .2 of this stability criterion. Depending on the problem, it may be possible to increase Δt and hence speed computations. Perhaps other choices of time derivative operators may be more efficient though such choices are not easy because of numerical stability requirements. Like the spatial operators, the worth of such choices would depend on the computer hardware. One method is by Taylor's expansions to get higher order time differencing schemes but this would be very complicated in the elastic case (see Dablain, 1986, for the acoustic case). One other alternative may be by using something like the 1/6 trick (Claerbout, 1976). Unfortunately, while it works well for implicit finite differences, it seems to lead to an unstable explicit scheme (in the 1D acoustic case). Anyway, it is not clear how to apply this trick to inhomogeneous elastic finite differences. Implicit methods which offer unconditional stability may be the answer but are not desirable because of their great complexity in more than one dimension unless approximations such as splitting are made.

NUMERICAL EXAMPLE

A shot simulation over a simple velocity model consisting of a layer over a half-space is used to illustrate the method. The properties of the upper layer are $vp = 3$ km/sec, $vs = 1.7$ km/sec and $\rho = 2$ gm/cc while the properties of the underlying half-space are $vp = 6$ km/sec, $vs = 3.4$ km/sec and $\rho = 2$ gm/cc. A grid spacing of at the origin and was a pressure type source with a second derivative Gaussian far field wavelet (in displacement). Absorbing boundary regions were used all around the model. Figure 2a shows a snapshot of the waves at $t = .96$ seconds. The main events are the direct P, reflected P, reflected S, transmitted P, transmitted S and P and S head waves trailing from the transmitted P in the upper layer. The shot gather is shown in Figure 2b (note that due the use of approximate δ-functions, the source generated a small amount of S-wave energy). There is negligible grid dispersion despite the fact that at the 90 percentile frequency there were only 2.8 gridpoints per wavelength. The 90 percentile frequency in this example was 30 hertz and is the frequency at the 90 percentile of the amplitude spectrum. Also, note that there is no noticeable reflection from the absorbing boundaries.

CONCLUSIONS

An elastic finite difference algorithm is described based on the stress equation approach (Kosloff, et al. 1984) but using derivative operators that are centered half way between gridpoints. Because of this centering it does not have the Nyquist problems that plagued the Fourier derivative method described by Kosloff. A simple non-tedious method for designing convolutional derivative operators is described. This makes it easy to design customized operators to maximize efficiency on any given computer. On the CRAY-1S and CONVEX-C1 the optimal operator is eight points long, being, more than 5 times faster than the Fourier derivative method and slightly faster than a two point scheme (and requiring 16 times less memory).

ACKNOWLEDGMENTS

I acknowledge the support of the sponsors of the Stanford Exploration Project (SEP) and Jon Claerbout. Also, I acknowledge the support of the CNRS (C_2VR) who supplied some CRAY time to test the algorithm. Thanks to Albert Tarantola for making possible a stay at the Institut de Physique du Globe in Paris where some of the early tests of this algorithm were carried out. Thanks to Alexandre Nercessian, Jean Virieux and Jean Remy for CRAY communication help and advice.

REFERENCES

Cerjan C., Kosloff D., Kosloff R., and Reshef M., 1985, Short note: A nonreflecting boundary condition for discrete acoustic and elastic wave equations: Geophysics, **50**, 705-708.

Claerbout, J.F., 1976, Fundamentals of geophysical data processing: McGraw-Hill.

Clayton, R.W., and Engquist, B., 1980, Absorbing side boundary conditions for wave-equation migration: Geophysics, **45**, 895-904.

Dablain, M.A., 1986, The application of high-order differencing to the scaler wave equation: Geophysics, **51**, 54-66.

Kosloff D., Reshef M., and Loewenthal D., 1984, Elastic wave calculations by the Fourier method: Bulletin of the Seismological Society of America, **74**, 875-891.

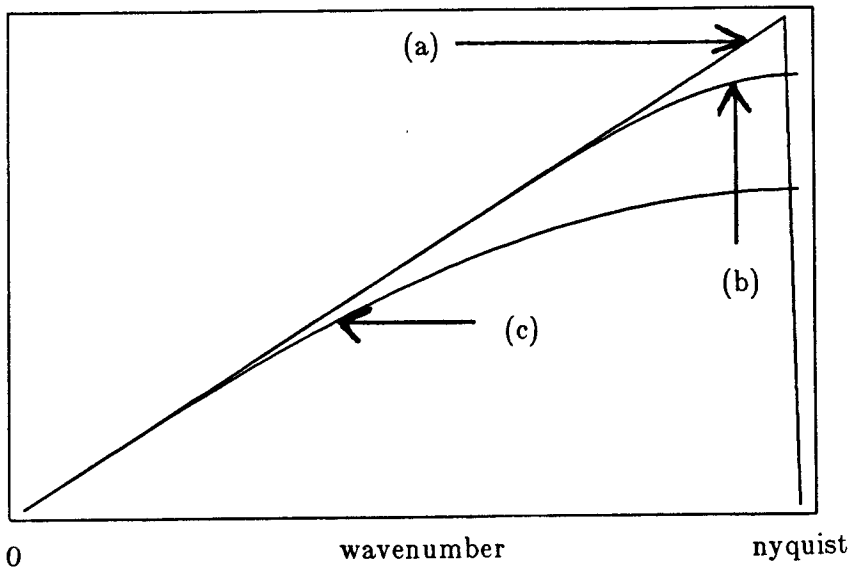

FIG. 1a. Wavenumber domain plot of three approximations for ∂_x, (a) is the perfect operator (ik in the wavenumber domain), (b) is an 8 point convolutional operator in the space domain, and (c) is the conventional 2 point finite difference operator.

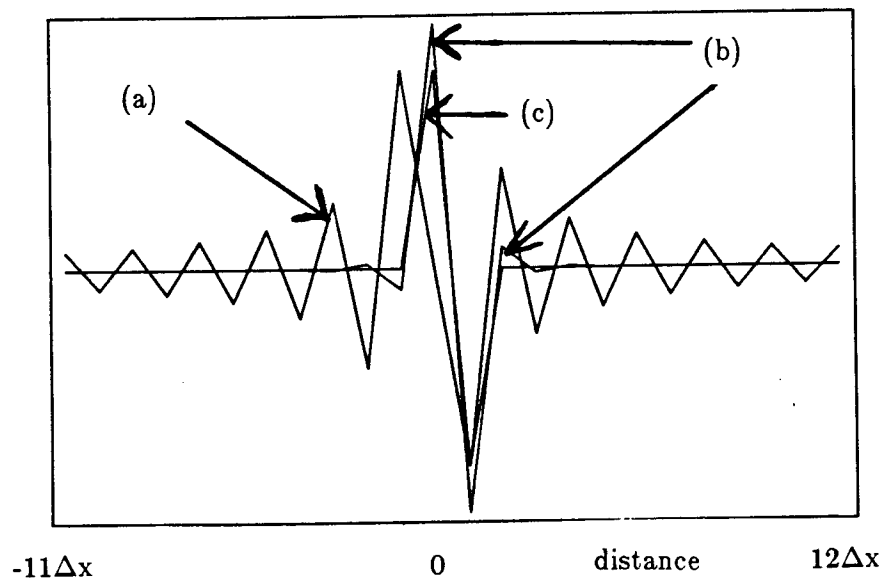

FIG. 1b. Space domain plot of three approximations for ∂_x, (a) is the perfect operator (ik in the wavenumber domain), (b) is an 8 point convolutional operator in the space domain, and (c) is the conventional 2 point finite difference operator.

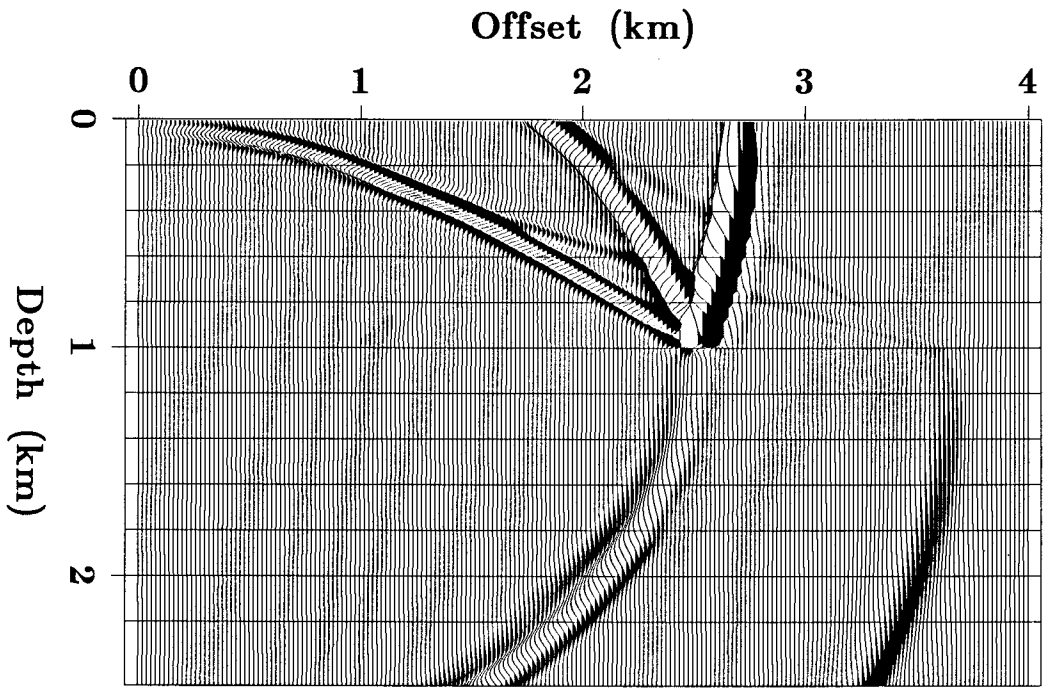

FIG. 2a. Snapshot at $t = .96$ seconds of waves propagating in the layer over a half-space model. The horizontal component of displacement is shown.

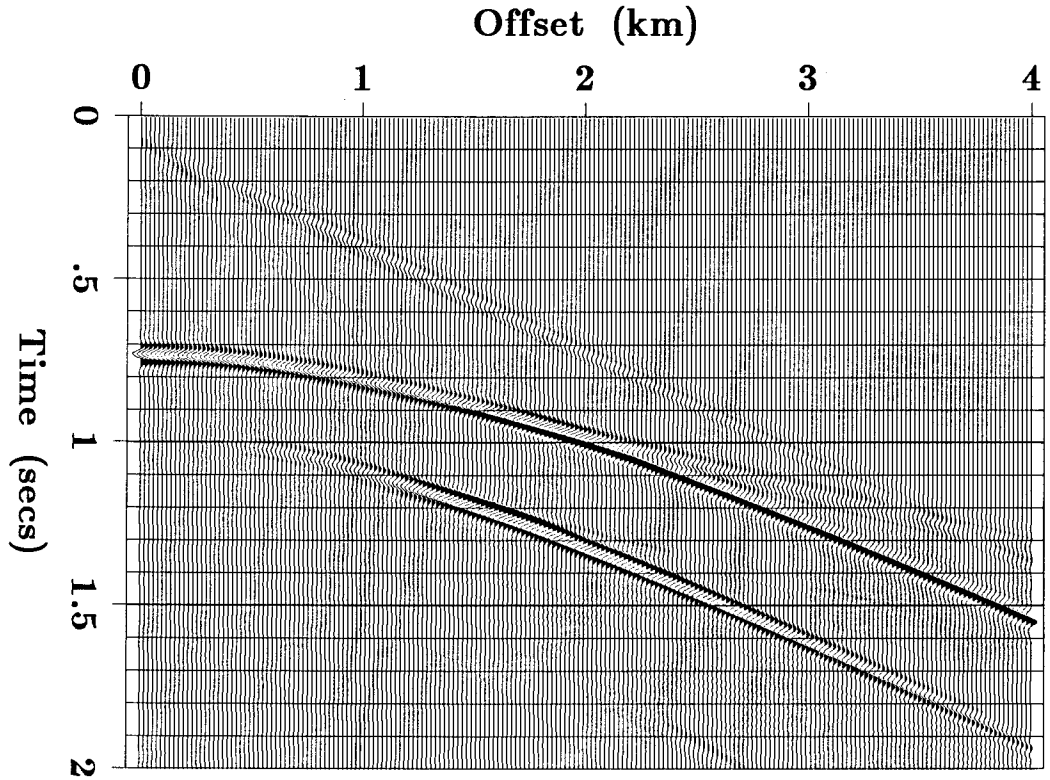

FIG. 2b. Shot gather for the layer over a half-space model. The vertical component of displacement is shown.

Date: Thu, 6 Mar 86 21:10:40 pst
From: Jon Claerbout <jon>
To: chuck
Subject: old times

I'm glad to see that our gang still has the intestinal fortitude
to bring a computer to its knees.

Date: Thu, 6 Mar 86 21:11:25 pst
From: Chuck Sword <chuck>
To: jon
Subject: Re: old times

But now we can bring it to its knees 12 to 20 times faster.