Constrained inversion for finite-difference operators

Joe Dellinger

ABSTRACT

In SEP–50 I presented an inversion algorithm for finding accurate finite-difference operators. Each element of the finite-difference operator was a free parameter in the inversion. The resulting objective function was unsuitable for conjugate gradient techniques. The algorithm can be constrained easily by constructing the operators as weighted sums over a set of basis functions. The inversion determines the weights of the basis functions, and not the actual elements of the operator. This space of basis function weights is found to be much more suitable for the use of conjugate gradient techniques than was the descent space of the original problem.

REVIEW OF INVERSION

Conjugate gradients is a general technique for solving matrix equations of the form

$$Ax = b,$$

where A is a known matrix, b is a known vector, and x is the vector to be solved (read inverted) for. Like all optimization techniques, conjugate-gradients attempts to minimize the norm of the error

$$Ax - b = \text{residuals}.$$

The conjugate-gradients method used here does not need to know A explicitly, but only needs to have available a subroutine which calculates Ax given x and another which calculates $A^t b$ given b. Thus we can easily extend this linear technique to nonlinear problems. Starting from a nonlinear equation $A(x) = b$ which we wish to solve, we linearize about some point (x, b) for small perturbations δx:

$$A(\delta x; x, b) = \delta b. \tag{1}$$

A is a linear function of δx, but a nonlinear function of (x, b). For each fixed (x, b), there is an *exact* linear transpose of $A(\delta x; x, b)$, which defines

$$A^t(\delta b; x, b) = \delta \hat{x}. \tag{2}$$

Note that $\delta\hat{x}$ does not have the same units as δx. We can now use the same conjugate-gradient algorithm as before. All we need are two subroutines, one that returns $A(\delta x; x, b)$ and another that returns $A^t(\delta b; x, b)$. If all goes well, conjugate-gradients will find the globally best solution of $A(x) - b \approx 0$.

REVIEW OF SEP–50

In SEP–50, Peter Mora and I showed how to use standard inversion techniques to find good finite-difference operators. Consider the acoustic wave equation

$$\ddot{p} - \partial_{xx} p = f, \tag{3}$$

where p is the pressure field and f is the forcing function, both functions of space and time. We can use finite differences to solve this equation for $p(x,t)$ given $f(x,t)$ and ∂_{xx}. Here we will take the unusual step of considering $f(x,t)$ to be *constant*, and ∂_{xx} to be the *free variable*. Given a forcing function $f(x,t)$ and the ideal resulting wavefield $p(x,t)$, the technique of the preceding section can then be used to invert for the best possible representation of ∂_{xx} within the limits imposed by the discretization of the problem.

The advantage to this method is that the finite-difference operator ∂_{xx} can be optimized for a given irregular grid, boundary condition, or source wavelet, with no preconceptions as to what sort of operator should be best. An representation of ∂_{xx} is "best" if the wavefield that results when it is used in the wavefield extrapolation operator differs by the minimum possible amount from the desired "perfect" answer.

In SEP–50, the parameters of the inversion were the elements of the finite-difference operator. Although this choice of parameters makes for the simplest theory and code, it is not a very good choice. Most of the effort of the inversion gets wasted in re-discovering such simple principles as "the sum of all the elements in the operator should be zero" or "the operator should be symmetric". Compared to the errors induced by violating these general principles, the slight improvements made by getting exactly the right operator are insignificant. Local minima abound, frustrating efforts to find the global solution.

The solution is to force some constraints upon the inversion, limiting the search to a space of "reasonable" possibilities. Laws such as "the sum of all the elements in the operator should be zero" should be worked into the algorithm from the beginning, leaving behind a more tractable search space of lower dimension.

INVERSION WITH CONSTRAINTS

We will now rederive the inversion exactly as in the previous paper, but with the additional complexity of constraints. Instead of letting the individual elements of the operator be the parameters, we will build up the operator out of a weighted sum of basis operators. This restricts the search space. For example, if each of these basis operators has zero sum, so must any linear combination of them.

Problem definition

Letting p and f be discrete, we can rewrite "∂_{xx}" as a convolutional operator C. Equation 3 then becomes

$$\ddot{p} - C *_x p = f. \tag{4}$$

The subscript on the convolutional operator shows whether the convolution is over space or time. The forward problem is to find the wavefield p given the finite-difference operator C and the forcing function f. Our inverse problem is to find C given p and f. We constrain C by constructing it as a linear combination of basis functions c_n:

$$C = \sum_n A_n c_n. \tag{5}$$

Our goal is to invert for the set of real coefficients $\{A_n\}$, which will determine the finite-difference second-derivative operator C.

Linearizing the problem

Equation 4 relates C and p. We will invert this nonlinear equation by using conjugate-gradient techniques on the linearized approximation of this equation. To construct the linearized equation we perturb both the finite-difference second-derivative operator C and the wavefield p:

$$(\ddot{p} + \delta\ddot{p}) - (C + \delta C) *_x (p + \delta p) = f. \tag{6}$$

Expanding, discarding nonlinear terms, and reordering we obtain:

$$(\ddot{p} - C *_x p - f) + \delta\ddot{p} - C *_x \delta p = \delta C *_x p. \tag{7}$$

The parenthesized term is identically zero by equation 4 and so can be discarded. Defining

$$\delta f = \delta C *_x p = [\sum_n \delta A_n \, c_n] *_x p \tag{8}$$

for notational convenience equation 6 can be rewritten as

$$\delta\ddot{p} - C *_x \delta p = \delta f. \tag{9}$$

This equation is our linearized forward problem corresponding to equation 1. It is a linear equation giving δp as a function of the $\{\delta A_n\}$ for a given "current search position" (C,p).

Note that equation 9 has just the same form as equation 4. δp is just the wavefield that results when δf is used as a forcing function. The same computer subroutine that solves equation 4 for p can also be used to calculate δp in equation 9. Mathematically, if G is the Green's function such that

$$p(x_r, t) = \int dx \, G(x_r, t; x, 0) *_t f(x, t) \tag{10}$$

solves equation 4, then

$$\delta p(x_r, t) = \int dx \, G(x_r, t; x, 0) *_t \delta f(x, t) \tag{11}$$

will solve equation 9. ($G(x_r, t; x, 0)$ is the wavefield observed at x_r at time t due to a shot having been fired at location x at time 0.) In practice we need never explicitly calculate this Green's function; using it in the mathematics represents a finite-difference calculation on the computer.

We now know how to take our finite-difference modeling program and use it as the core of the first required subroutine. The conjugate-gradients algorithm also requires the second subroutine, which is the exact transpose of the first.

The transpose

To find the transpose, we first find the kernel K of the linearized forward operator. Integrating a function times this kernel is the continuous equivalent of multiplication of the "matrix kernel" times the "function vector".

Isolating the kernel K in a useful form requires a few simple manipulations. We begin by substituting the definitions of δf and C into equation 9:

$$\delta p(x_r,t) = \int dx\; G(x_r,t;x,0) *_t (p *_x \sum_n \delta A_n\; c_n).$$

Expanding out the convolution over x using the new variable X, this becomes

$$= \int dx\; G(x_r,t;x,0) *_t \int dX\; p(x-X,t) \sum_n \delta A_n\; c_n(X).$$

Regrouping to pull out the δA_n, we get

$$= \sum_n \delta A_n \Big[\int \int dx\; dX\; G(x_r,t;x,0) *_t p(x-X,t) c_n(X) \Big].$$

The term between the square brackets is the kernel K:

$$= \sum_n \delta A_n\; K(x_r,t;n) \tag{12}$$

To construct the transpose, every variable in the kernel K which is summed or integrated over must be changed to a free variable, and every free variable in K must be changed to one that is summed or integrated over. This is the continuous equivalent of transposing a matrix. The transpose is thus

$$\delta \hat{A}_n = \int \int dx_r\; dt\; \delta p(x_r,t) K(x_r,t;n). \tag{13}$$

We write $\delta \hat{A}_n$ to indicate that this does *not* have the same units as δA_n. (The difference between the transpose and the inverse in some sense is that the inverse maps back to the correct units while the transpose doesn't.)

Calculating the transpose

This transpose must now be recast in terms of the original finite-difference equation. Substituting K back in equation 13, we get:

$$\delta \hat{A}_n = \int \int dx_r\; dt\; \delta p(x_r,t) \int \int dx\; dX\; G(x_r,t;x,0) *_t p(x-X,t) c_n(X).$$

Rearranging to pull out the c_n, this becomes

$$= \int dX\; c_n(X) \int \int \int dx_r\; dt\; dx\; \delta p(x_r,t) G(x_r,t;x,0) *_t p(x-X,t). \tag{14}$$

Applying the identity $\int dt\; h(t)[g(t) * f(t)] = \int dt\; f(-t)[g(t) * h(-t)]$,

$$\delta \hat{A}_n = \int dX\; c_n(X) \int \int dt\; dx \int dx_r\; p(x-X,-t) G(x_r,t;x,0) *_t \delta p(x_r,-t). \tag{15}$$

Define

$$\psi(x,-t) = \int dx_r \; G(x,t;x_r,0) *_t \delta p(x_r,-t). \tag{16}$$

$\psi(x,t)$ is called the "back propagated residuals": to calculate it we take the error in p, time reverse it, use our handy wave-propagation subroutine yet again, and then time reverse the output.

Equation 15 now becomes (using the reciprocity of G)

$$\begin{aligned} \delta \hat{A}_n &= \int dX \; c_n(X) \int \int dt \; dx \; p(x-X,-t)\psi(x,-t) \\ &= \int dX \; c_n(X) \int \int dt \; dx \; p(x-X,t)\psi(x,t). \end{aligned} \tag{17}$$

To evaluate this equation we take the back-propagated residuals and cross correlate them with the wavefield. Note that the boundary conditions in time work out nicely; if the wavefield extrapolation program runs from $t = 0$ to $t = T$, p is zero before $t = 0$ by causality, and the residuals are considered to be zero after $t = T$. From equation 16, ψ is zero after $t = T$ by time-reversed causality and so when the two are integrated together only the contribution from 0 to T is nonzero. The resulting operators are then dotted with the basis functions to determine the coefficients $\{A_n\}$.

SOME EXAMPLES

To test the method, we begin with the simplest possible example. The forcing function consists of a spike centered on the x axis at zero time. This forcing function is shown in the bottom right hand corner of Figure 3. The "exact" solution (the one the inversion is trying to duplicate) is the wavefield that results when the simplest possible second derivative operator, $\{1,-2,1\}$, is used. Only two basis functions are used, $\{1,-2,1\}$ and $\{1,0,-2,0,1\}$. The top part of Figure 1 shows the results for four different inversion runs, starting from four different initial positions. The contour plot shows how the norm of the residuals varies as a function of the weights of the two basis functions. The contour lines are not at all equally spaced; the norm of the residuals varies over several orders of magnitude. The region without contours in the upper left hand corner corresponds to the unstable region of the finite-difference operator, where the residuals are effectively infinity. For each inversion run, a thick line shows the descent path followed. An arrow near the beginning of the path shows the descent direction, and the final value is marked with a box.

For this forcing function, local minima were somewhat of a problem. The search converged to the correct solution for only two of the four inversions. However, there is a fairly broad and deep local minimum at the correct solution, and the two paths that fell into that region converged on exactly the correct solution.

To see if a different forcing function could produce a better result, the one shown in the bottom left hand corner of Figure 3 was used instead. This is simply Gaussian noise. Such noise should have fewer spurious correlations which result in troublesome local minima. The results are shown in the lower half of Figure 1. The norm of the residual as a function of the basis function weights is now much better behaved, and a good minima was attained for all four initial starting choices.

In Figure 2 the same two forcing functions are used as before, but the "exact" solution was obtained by interpolating the forcing function onto a tenfold finer grid, calculating

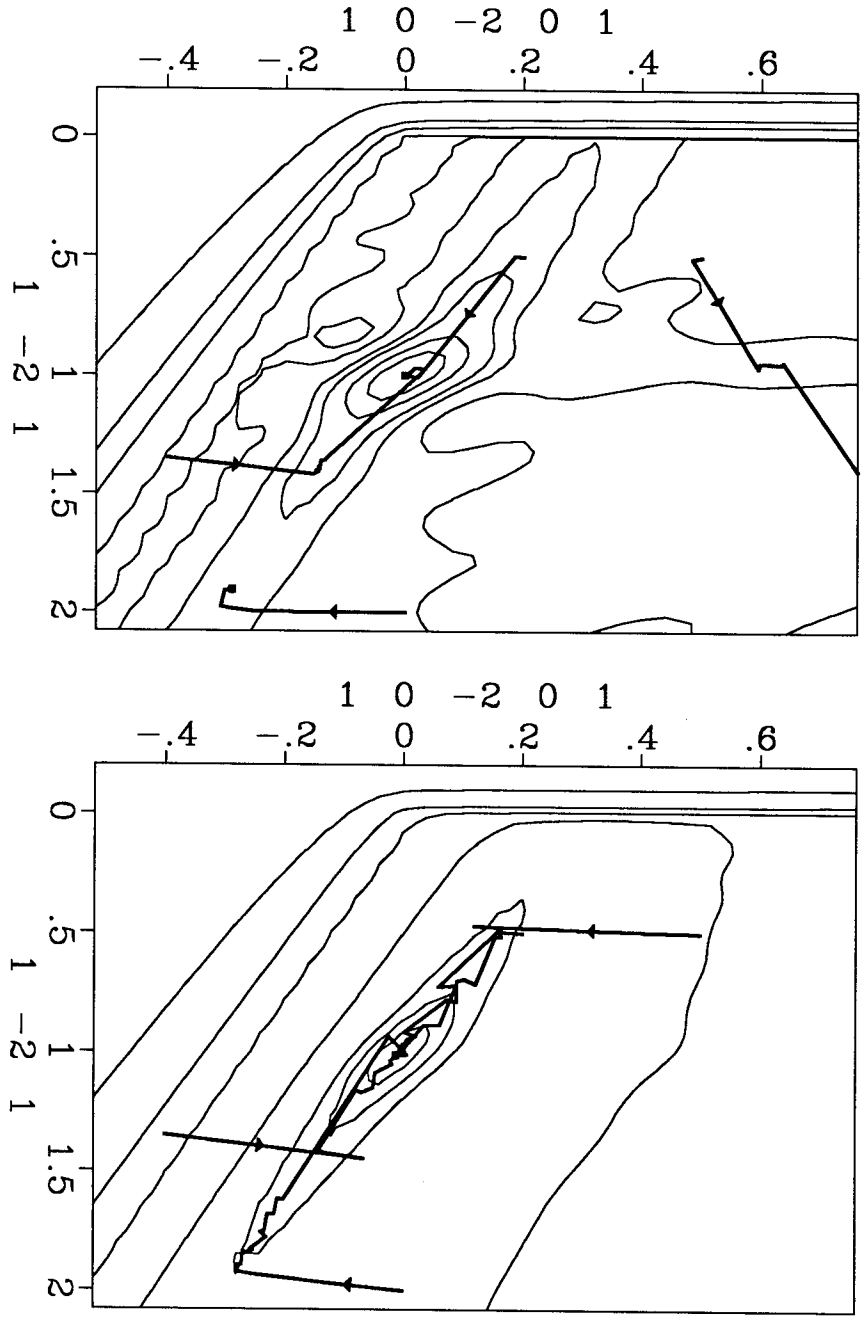


FIG. 1. Contour plots showing the search space for two different forcing functions and the histories of four different inversions on each. The axis labels show the two different basis functions, and the axes are labeled to show their weights at each point. The contour lines are not evenly spaced; several orders of magnitude are spanned by the ones shown. The finite-differencing scheme becomes unstable in the upper left hand region and as a result the residuals there are infinite.

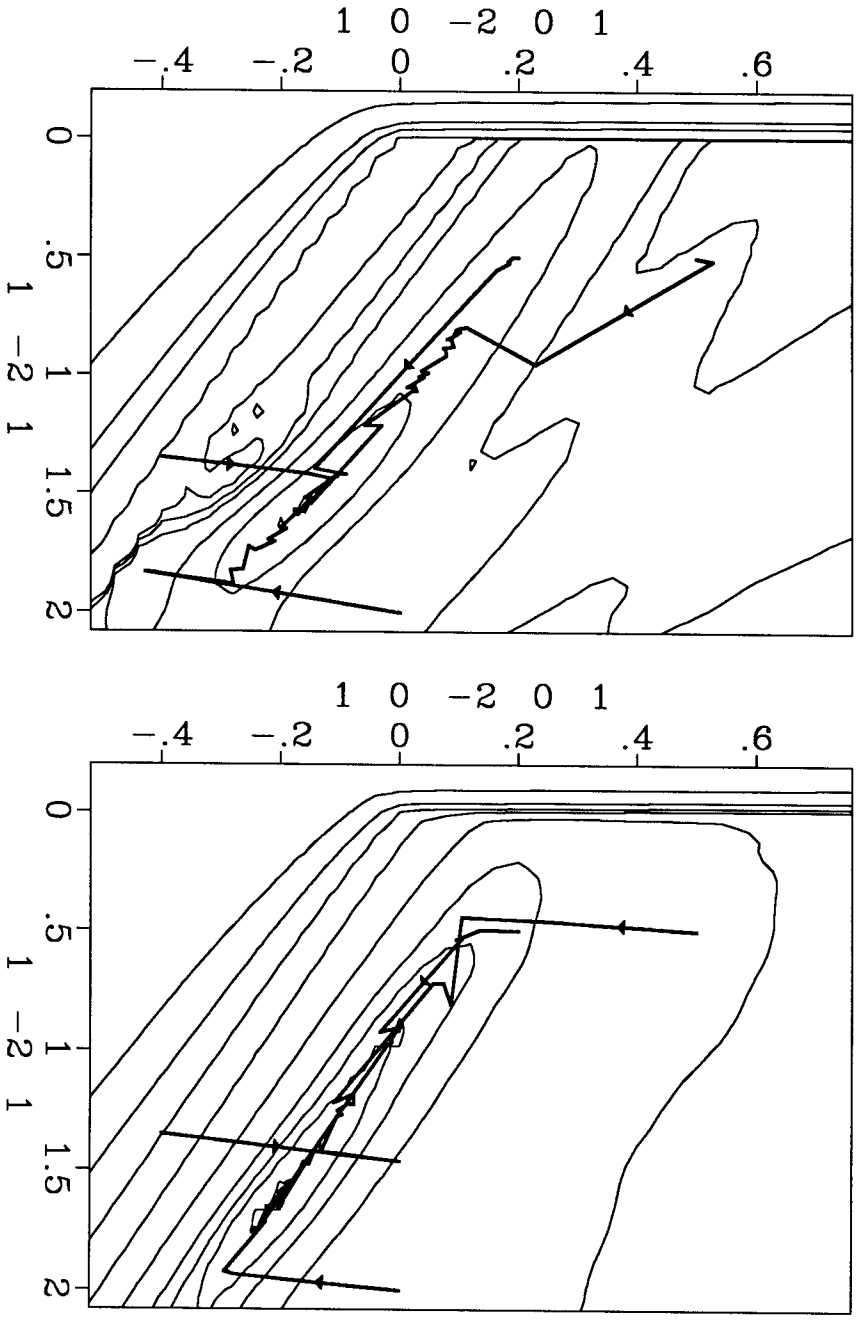


FIG. 2. Contour plots showing the search space for two different forcing functions and the histories of four different inversions on each. This figure differs from the previous one in that the descent is trying to find a five-point operator that minimizes numerical dispersion.

the wavefield there, and then subsampling back onto the original grid. This minimizes the dispersion.

For this example again the Gaussian noise proves to be a better forcing function for most of the search space. It does possess a "double minima", but since the two minima are nearly equally good it doesn't really matter which the inversion falls into. For both forcing functions, all four inversions converge onto very nearly the best five-point second derivative operator calculated by Taylor's series. This is not surprising, as the Taylor's series is the optimal low frequency solution.

CONCLUSIONS

Once constraints are added, this method seems to work well for the simple case where the same operator is used over the entire grid. The next step is to allow there to be a different operator at each position. This will allow the inversion to solve for operators that work on irregular grids, and also operators to produce absorbing boundary conditions.

REFERENCES

Dellinger, J., and Mora, P., 1986, Accurate finite-difference derivative operators by inversion: SEP–50.

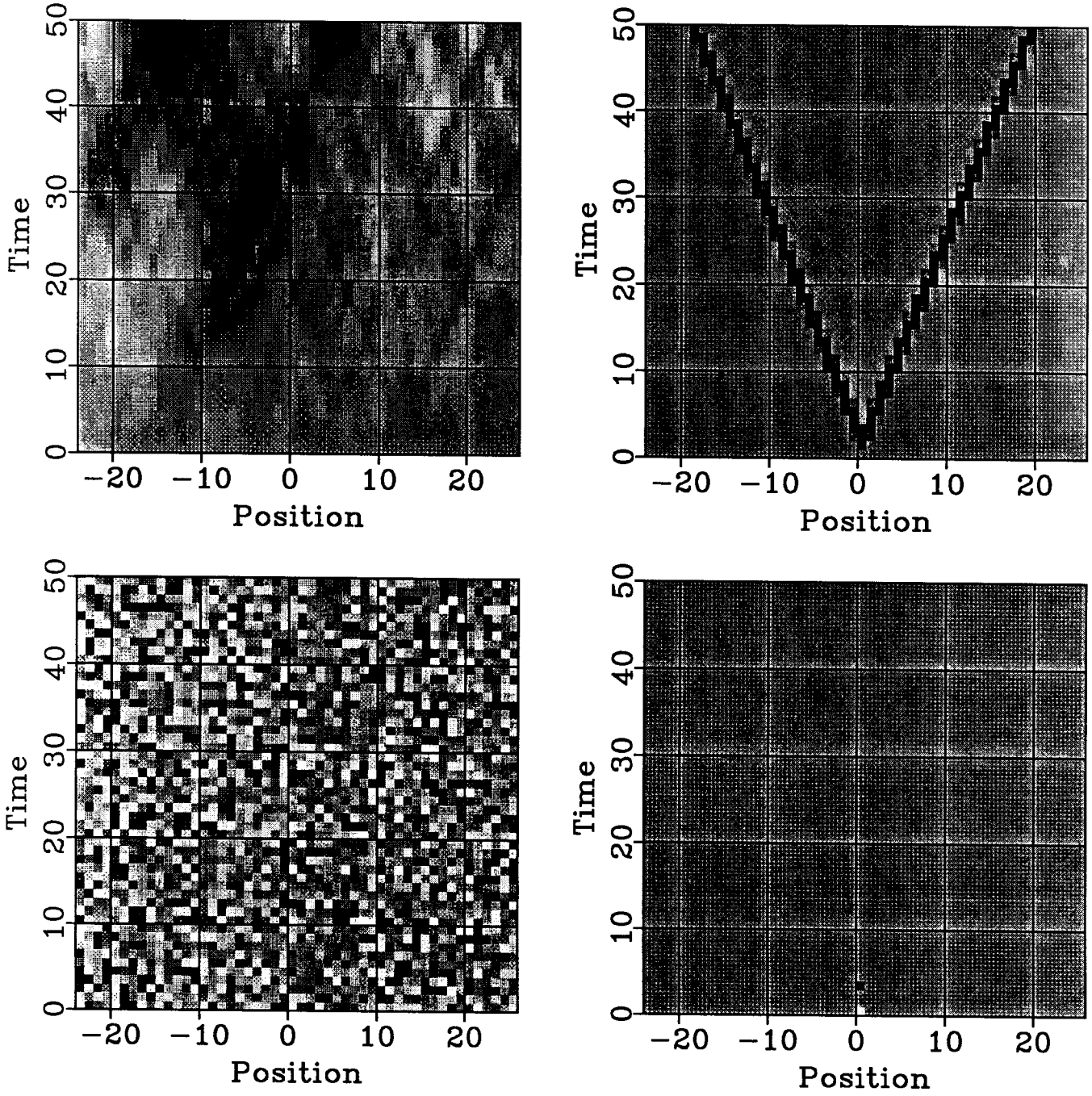
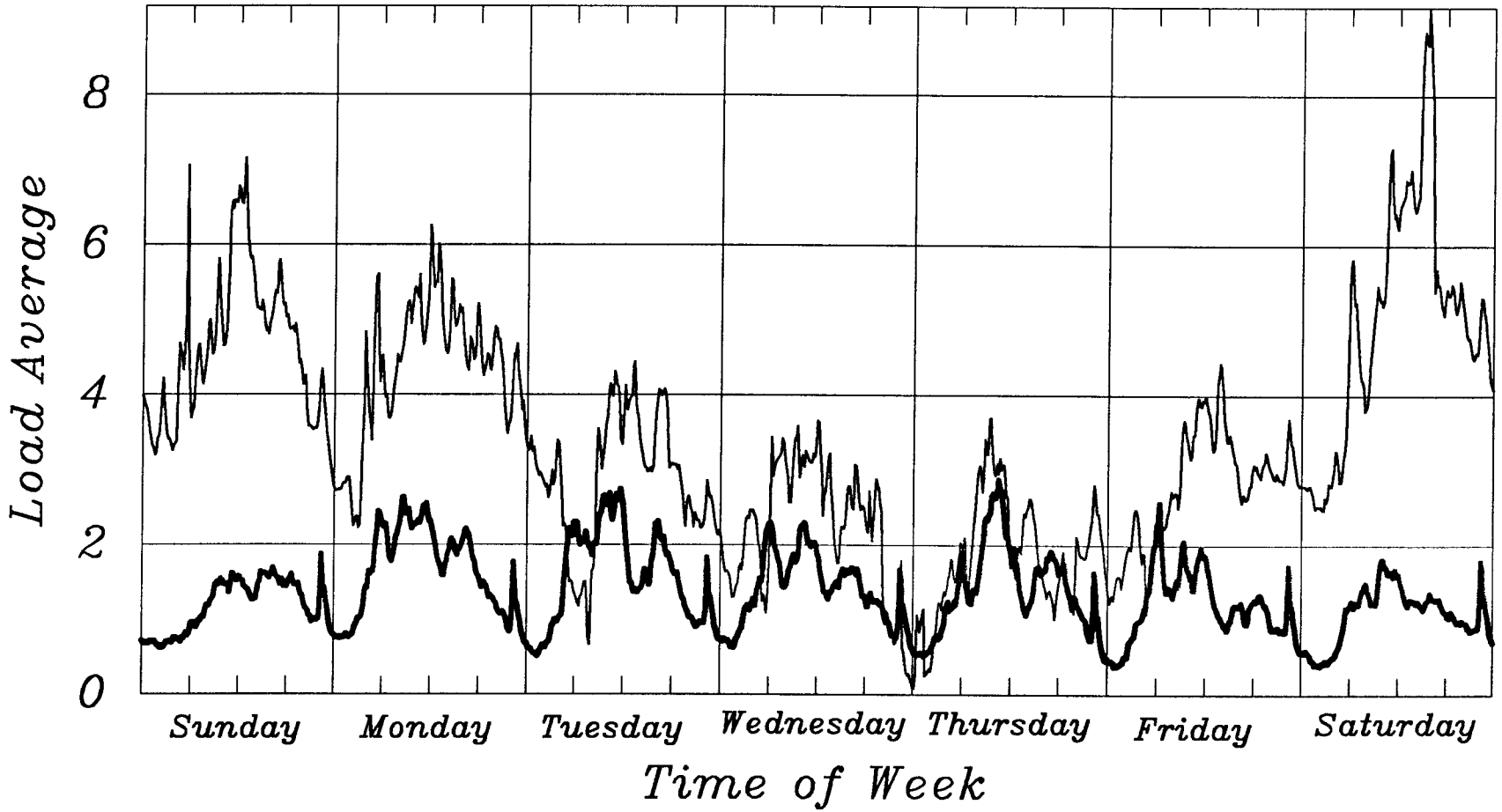


FIG. 3. The forcing functions (bottom) and "exact" wavefields (top) used for the inversions in Figure 2.

Hanauma, then and now



This plot shows the load on our Convex C−1 as a function of time of week. Each vertical bar is at 6AM. The 3 small tics mark Noon, 6PM, and Midnight.
The thick plot is for March−July 1986; The thin plot is for March−April 1987.
Several features can be discerned: The sharp spike at 4AM on all days is due to system accounting. Regularly scheduled maintenance is on Tuesday mornings now, which explains the sharp dip on Tuesday around noon. The graph from last year mostly reflects when people were actually here working; note the regular dips at lunch and dinnertime. Nowadays the load data is controlled by a few huge crunchers. The heaviest use day is now Saturday!!!!