Stanford Exploration Project, Report 70, December 18, 1997, pages 1–156

Simplifying 3-D migration operators using Givens rotations

Steve Cole[1]

keywords: *velocity, three-dimensional, algorithm*

ABSTRACT

The pentadiagonal form of the 2-D Laplacian makes 3-D migration schemes inaccurate (with splitting) or expensive. Using Givens rotations, and assuming that velocity is constant or varies only with depth, it is easy and inexpensive to reduce the pentadiagonal system to a tridiagonal form, since the algorithm lends itself well to a parallel implementation. Unfortunately, for the case of lateral velocity variation, the matrix rapidly becomes less sparse as rotations are performed, making the method too expensive. But, even in the presence of lateral velocity variation, this scheme may be a useful preconditioner for other solution methods.

INTRODUCTION

Claerbout (1985) showed that the two-dimensional Laplacian operator, which appears in 3-D finite-difference migration, has the form shown in Table 1. The addition of the crossline or y direction has added two diagonals to what is, in 2-D migration, a tridiagonal matrix. This addition presents some practical problems. Direct methods for solving this pentadiagonal system of equations have costs proportional to the square of the model size n (which is 4 in the case of Table 1), as compared to solution methods of order n for the tridiagonal case. A common compromise is splitting, also described by Claerbout, where the pentadiagonal system is split into two tridiagonal systems that are solved successively. Splitting's drawback is that it yields effective migration operators that are anisotropic; the response is particularly poor for energy propagating at angles far from either the inline or crossline directions.

In SEP–60, (Cole, 1989) I examined using iterative schemes such as Jacobi and Gauss-Seidel to solve the pentadiagonal system. I found the methods to be unconditionally stable for the 15 degree equation, but the 45 degree scheme was unstable at low frequencies (and where stable, very slow to converge for the low frequencies).

When Dave Nichols (Nichols, 1991) began working on his predictor-corrector

[1]**email:** not available

−4	1	·	·	1	·	·	·	·	·	·	·	·	·	·	·
1	−4	1	·	·	1	·	·	·	·	·	·	·	·	·	·
·	1	−4	1	·	·	1	·	·	·	·	·	·	·	·	·
·	·	1	−4	·	·	·	1	·	·	·	·	·	·	·	·
1	·	·	·	−4	1	·	·	1	·	·	·	·	·	·	·
·	1	·	·	1	−4	1	·	·	1	·	·	·	·	·	·
·	·	1	·	·	1	−4	1	·	·	1	·	·	·	·	·
·	·	·	1	·	·	1	−4	·	·	·	1	·	·	·	·
·	·	·	·	1	·	·	·	−4	1	·	·	1	·	·	·
·	·	·	·	·	1	·	·	1	−4	1	·	·	1	·	·
·	·	·	·	·	·	1	·	·	1	−4	1	·	·	1	·
·	·	·	·	·	·	·	1	·	·	1	−4	·	·	·	1
·	·	·	·	·	·	·	·	1	·	·	·	−4	1	·	·
·	·	·	·	·	·	·	·	·	1	·	·	1	−4	1	·
·	·	·	·	·	·	·	·	·	·	1	·	·	1	−4	1
·	·	·	·	·	·	·	·	·	·	·	1	·	·	1	−4

Table 0.1: The two-dimensional matrix of coefficients for the Laplacian operator. Taken from Claerbout (1985).

method using conjugate gradients, I decided that the iterative schemes I had studied deserved another look. Like conjugate gradients, they operate via convolution in 2-D, which performs very well on the Connection Machine. It would be interesting to see how the two methods compared.

I sought a way to stabilize the 45 degree Jacobi iterative scheme. The problem is that the matrix is not sufficiently diagonally dominant. My hope was that a scheme that attacked the off-diagonal elements would make the matrix sufficiently well-conditioned that the Jacobi iterative scheme would be stable. Givens rotations are a tool to accomplish such an elimination of off-diagonal elements.

As a side benefit, such an improvement in the diagonal dominance of the matrix would accelerate convergence both for the Jacobi iterative scheme and conjugate gradients. In both these schemes, the convergence rate depends on the largest eigenvalue of the matrix, which is related to the diagonal dominance. This implies that Givens rotations might be a useful preconditioner for Jacobi iterations and conjugate gradients.

GIVENS ROTATIONS

Givens rotations annihilate off-diagonal matrix elements. They are often used in solving the symmetric eigenvalue problem, and have received greater attention recently because they lend themselves well to a parallel implementation.

The basic idea in Givens rotations is to annihilate a particular off-diagonal element of a matrix (and its symmetric pair). We can zero the elements a_{pq} and a_{qp} of matrix A by applying a coordinate rotation given by:

$$\begin{pmatrix} b_{pp} & 0 \\ 0 & b_{qq} \end{pmatrix} = \begin{pmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{pmatrix}^T \begin{pmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{pmatrix} \begin{pmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{pmatrix} \tag{1}$$

Effectively we have rotated the coordinate system by an angle θ, and chosen θ such that in the new coordinate system the desired off-diagonal elements will be zero. The diagonal elements a_{pp} and a_{qq} are modified by this transformation.

Golub and Van Loan (1989) show that θ is given by:

$$\theta = \frac{a_{qq} - a_{pp}}{2a_{pq}}. \tag{2}$$

Following Press et al (1986), after rotation the new values are given by:

$$b_{pp} = a_{pp} - ta_{pq} \tag{3}$$

$$b_{qq} = a_{pp} + ta_{pq}, \tag{4}$$

where

$$t = \frac{sgn(\theta)}{|\theta| + \sqrt{\theta^2 + 1}}. \tag{5}$$

On a larger scale, the operation that we perform on the full matrix A is:

$$B = J(\theta)^T AJ \tag{6}$$

where $J(\theta)$ is given by:

$$J(\theta) = \begin{pmatrix} 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \dots & cos\theta & \dots & sin\theta & \dots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \dots & -sin\theta & \dots & cos\theta & \dots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{pmatrix}. \tag{7}$$

This technique is one method traditionally used to find the eigenvalues and eigenvectors of a matrix. If we annihilate all the off-diagonal elements, we are left with a diagonal matrix B that contains the eigenvalues of A. The rotation matrices $J(\theta)$ needed to perform the annihilations, when cascaded together, yield the eigenvectors of the matrix A. When Givens rotations are used in this way to diagonalize a matrix, the method is known as a Jacobi transformation (unrelated to the Jacobi iterative scheme I mentioned earlier). Givens rotations are also known as Jacobi rotations for this reason.

This method of diagonalizing a matrix is obviously quite expensive in the general case, since there are $(n-1)*(n-2)/2$ annihilations to perform for an nxn matrix. But in our case, A is very sparse, and we want only to attack the outer diagonals of Table (1), leaving a tridiagonal system that we can solve using traditional methods. The cost should be quite small.

It would be straightforward if this operation zeroed the two off-diagonal elements and left the rest of the matrix unchanged. Unfortunately this is not the case. When we pre-multiply A by $J(\theta)^T$ and post-multiply by $J(\theta)$, what is modified is not just the two diagonal and two off-diagonal elements we are interested in, but the whole two rows and two columns that contain these elements.

In other words, as we move through the matrix, annihilating off-diagonal elements, we create non-zero elements elsewhere, and occasionally fill in previously-created zeros. For this reason, the scheme is not direct but iterative. One pass through the matrix increases the diagonal dominance, but leaves some artifacts. Additional passes gradually knock down these artifacts, and at some point everything off the diagonal is small enough that it can be ignored. Press et al. suggest 6-10 passes are needed for the general case of diagonalizing a matrix. The action of the method is best illustrated by an example, as in the following section.

The method lends itself well to a parallel implementation because a given rotation modifies only two rows or two columns of the matrix. If only one column or row were to be modified in each rotation, we would need only two passes – one for modifying the rows and one for the columns – in order to attack an entire diagonal in parallel. The fact that two rows or columns are modified simultaneously means that four passes (two for rows, two for columns) are required. This is still very few, and suggests that parallel implementations should be extremely fast.

Application to constant-velocity case

In the constant-velocity case, the coefficients in finite-difference migration operators will not vary laterally. In other words, the matrix from Table 1 will contain a constant value along each of its five diagonals. The Givens rotation scheme described above is particularly well-behaved in such a case.

Figure **??** shows a plot of such a matrix after several passes of a Givens rotation scheme. Here I have used just the constant values of 1 and -4 from Table 1. In the migration case these coefficients will be complex, but the real values used are more suitable for a demonstration.

In Figure **??**, one pass of Givens rotations (upper right) has turned the outer diagonals into a series of diagonals, all with amplitudes much smaller than the original. After three passes (lower right), these artifacts have effectively been suppressed. The matrix is now tridiagonal, and I can use traditional methods for solving the system.

Note that the rotation matrices that have been applied need also to be applied

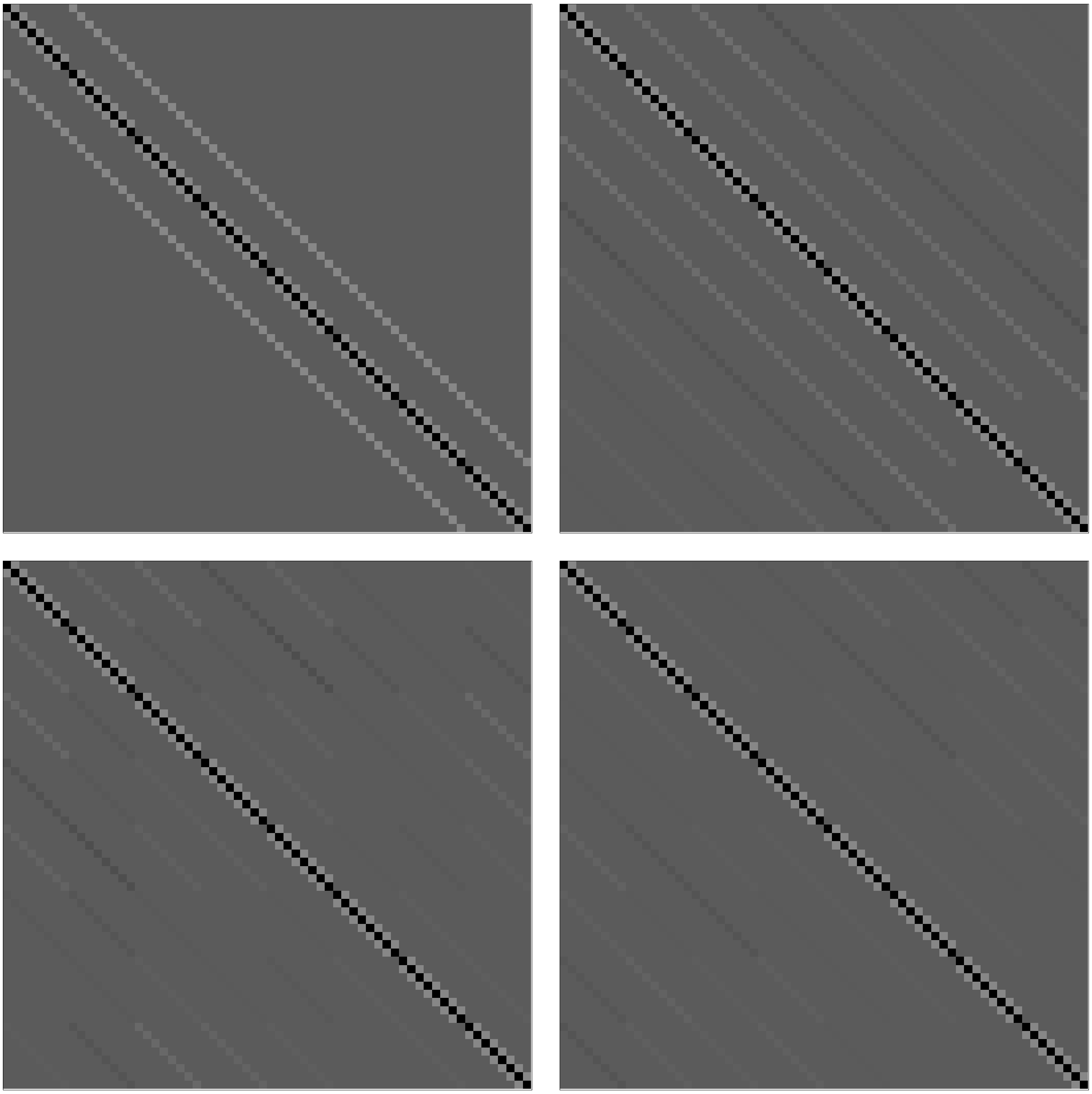

Figure 1: Migration operators before (upper left) and after each of three passes of the Givens rotation scheme. The first pass creates some diagonal artifacts, but these are suppressed by later passes. steve2-give [NR]

to the right-hand side of the system of equations being solved. Because the matrix remains sparse throughout the procedure, this method is rather inexpensive.

Figure **??** illustrates what is happening during one pass of the Givens rotation technique. The snapshots here are all from the first pass. It can be seen that artifacts are introduced as the diagonal elements are zeroed, and that the zeroed elements are filled in again (with smaller values) as the algorithm proceeds.

Snapshots from first pass of Givens rotation scheme. As the algorithm steps down along the main diagonal, the elements of the outermost diagonal are zeroed. The artifacts can be eliminated by additional passes.

Lateral velocity variation

When velocity varies laterally, the coefficients of the migration operator will vary along the diagonals. This has some adverse effects. Figure **??** shows the result after several passes where there has been a random 4% variation in coefficient amplitude introduced along the main diagonal and a 1% variation along the other diagonals. The first pass of Givens rotations, instead of introducing artifacts along a number of new diagonals, introduces these artifacts over several fairly broad bands.

This has serious consequences. Later passes spread these values out even further. After three passes, the matrix contains a large diagonal and a noisy background. At first glance, this might seem like a good result; the artifacts have dissipated rather than remaining as discrete events. The problem is that the number of operations needed to attenuate the artifacts becomes large very rapidly, so large that the cost of such an operation is prohibitive (order n^2 rotations instead of order n for attacking a single diagonal).

When the coefficients vary along each diagonal (as would be the case with a lateral velocity variation), the artifacts are spread out over wide bands instead of being located at discrete distances from the main diagonal (upper right). Successive iterations spread these values further. After three passes (lower right) the matrix is filled with a low-amplitude background. This slows the algorithm down considerably because all non-zero values must be annihilated.

The problem in Figure **??** comes from the random variation of the coefficient values. When we compute the angle θ for Givens rotation, we get slightly different answers at each point because of the varying coefficients. The angles that we compute are correct in that they do the best possible job of attacking the off-diagonal elements. However, the creation of artifacts is also controlled by the rotation angles. Different angles mean that artifacts are created in different places. Thus the 1% (4% on the main diagonal) variation in coefficients introduces artifacts over ranges of positions rather than at discrete locations. Again, this is a bad idea because it means we have to do prohibitively many operations to arrive at a tridiagonal form.

One alternative is to use a constant (mean) value of rotation angle to perform the

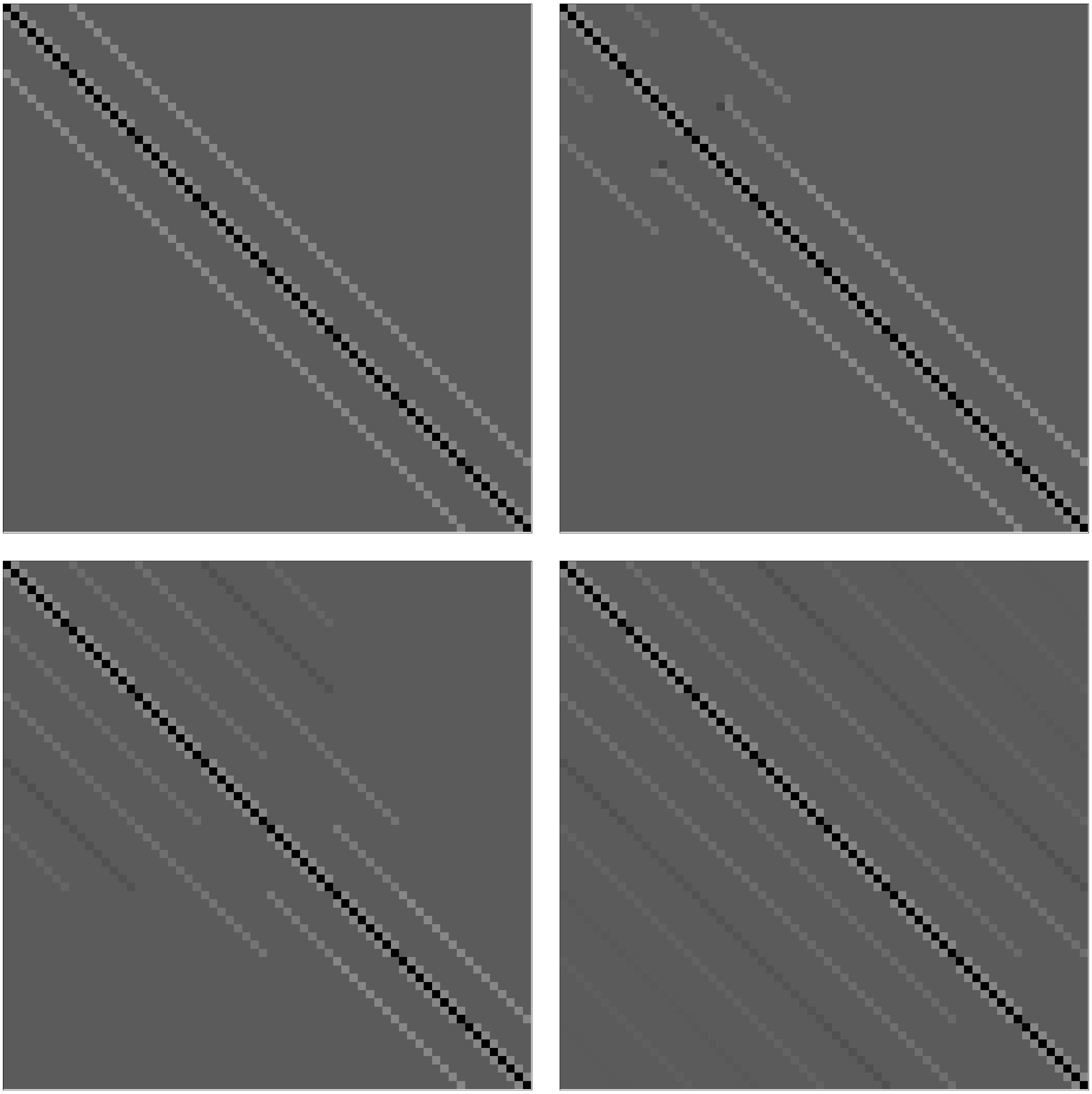

Figure 2: givestep $\boxed{\text{steve2-givestep}}$ [NR]

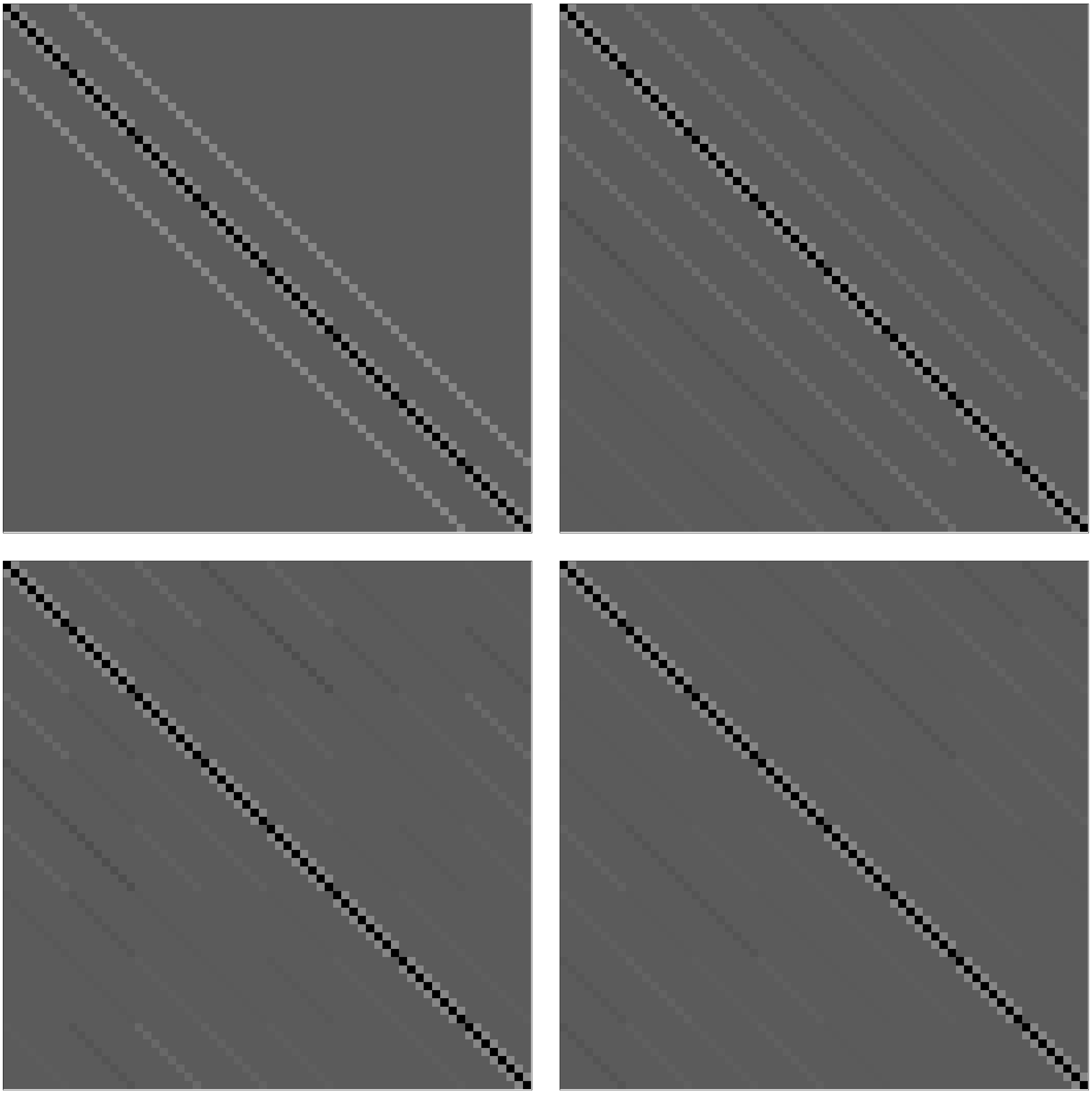

Figure 3: givenoise $\boxed{\text{steve2-givenoise}}$ [NR]

Givens rotations. This constant angle would keep artifacts confined to a few locations, and mean that attenuating the artifacts with successive passes would be much faster. In Figure **??**, I did this, using a rotation angle computed from the mean values of the diagonal coefficients. You can see that, as I predicted, the artifacts remain along the diagonals, similar to Figure **??**, rather than spreading over the entire matrix. This is good. On the down side, however, successive passes of rotation do not reduce the matrix to tridiagonal form. Compare the lower right panel of Figure **??** to that of Figure **??**, and you will see that the artifacts have not been attenuated very well. More passes do not solve the problem.

The reason this happens is because we used an average rotation angle. This angle does a relatively poor job of annihilating the off-diagonal elements, so no matter how many times we do it, these elements do not go away completely.

Here the coefficients vary along the diagonals but an average value has been used to perform the rotations. This has the advantage of confining the artifacts to narrow bands; thus the algorithm will be fast. Unfortunately, because the rotations have not been done precisely enough, the algorithm does not converge, as can be seen from the presence of artifacts after three passes (lower right).

All hope is not lost. Certainly in Figure **??** we have improved the diagonal dominance of the matrix, at little cost. We could use such an operator now in the Jacobi iterative scheme or conjugate gradients and expect convergence to be much faster. In the case of Jacobi, the 45 degree algorithm should be stable where it was not before. So even if we have not been able to reduce to tridiagonal form in the lateral velocity case, this method should still be a useful preconditioner for other methods.

CONCLUSIONS

For the constant-velocity case, we can use Givens rotations to reduce the pentadiagonal 3-D migration operator to tridiagonal form for a reasonable cost.

For the case where velocity varies laterally, this transformation is prohibitively expensive because the matrix loses its sparseness as the algorithm proceeds. However, using a mean rotation angle can significantly improve the diagonal dominance of the matrix, while keeping the matrix sparse. This improved operator can then be used in solution methods such as the Jacobi iterative scheme and conjugate gradients, and should improve the performance of these methods considerably.

More work needs to be done to determine whether Givens rotations can really aid in solving the 3-D migration problem. I need to estimate the cost of reducing the matrix to tridiagonal form and solving the resulting tridiagonal system. Then this cost should be compared with the costs of solving the original system by a good direct method, solving with iterative methods (with of without Givens rotations as a preconditioner), and solving via splitting. Only when these different methods have been implemented and the costs compared will I be able to decide what place, if any,

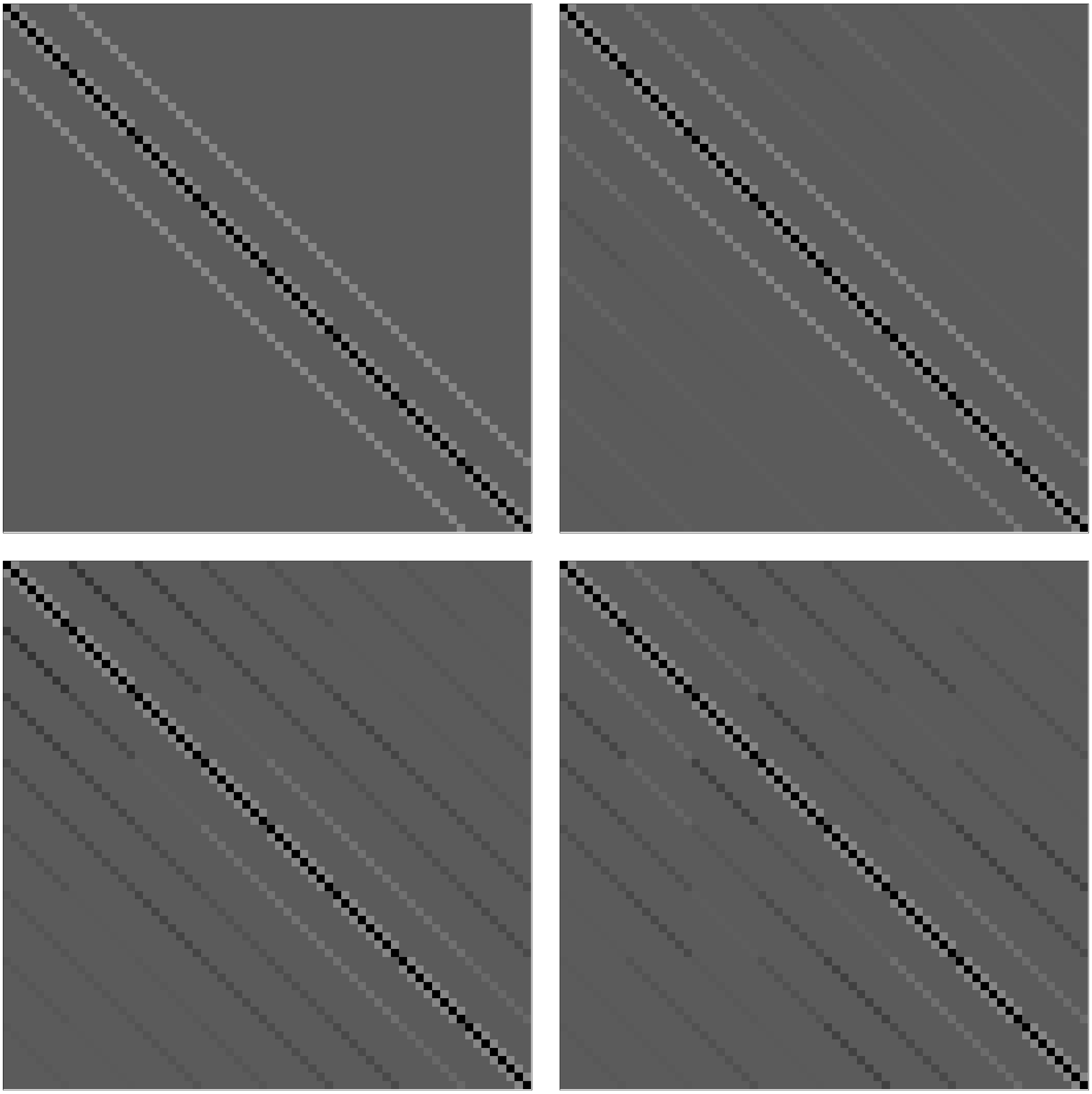

Figure 4: giveavg $\boxed{\text{steve2-giveavg}}$ [NR]

Givens rotations have in 3-D migration.

ACKNOWLEDGMENTS

Thanks to Martin Karrenbach and Dave Nichols for many helpful discussions, and to Martin for help with Mathematica.

REFERENCES

Claerbout, J.F., 1985, Imaging the Earth's Interior: Blackwell Scientific Publications.

Cole, S., 1989, Iterative methods for 3-D finite-difference migration and modeling: SEP–**60**, 149-163.

Golub, G.H., and Van Loan, C.F., 1989, Matrix Computations: The Johns Hopkins University Press.

Nichols, D., 1991, 3-D migration by a predictor-corrector method: SEP–**70**.

Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T., 1986, Numerical recipes: The art of scientific computing: Cambridge University Press.