*Stanford Exploration Project, Report SERGEY, November 9, 2000, pages 13–**??***

Least-square inversion with inexact adjoints. Method of conjugate directions: A tutorial

Sergey Fomel[1]

ABSTRACT

This tutorial describes the classic method of conjugate directions: the generalization of the conjugate-gradient method in iterative least-square inversion. I derive the algebraic equations of the conjugate-direction method from general optimization principles. The derivation explains the "magic" properties of conjugate gradients. It also justifies the use of conjugate directions in cases when these properties are distorted either by computational errors or by inexact adjoint operators. The extra cost comes from storing a larger number of previous search directions in the computer memory. A simple ratfor program and three examples illustrate the method.

INTRODUCTION

This paper describes the method of conjugate directions for solving linear operator equations in Hilbert space. This method is usually described in the numerous textbooks on unconstrained optimization as an introduction to the much more popular method of conjugate gradients. See, for example, *Practical optimization* by Gill et al. (1995) and its bibliography. The famous conjugate-gradient solver possesses specific properties, well-known from the original works of Hestenes and Stiefel (1952) and Fletcher and Reeves (1964). For linear operators and exact computations, it guarantees finding the solution after, at most, n iterative steps, where n is the number of dimensions in the solution space. The method of conjugate gradients doesn't require explicit computation of the objective function and explicit inversion of the Hessian matrix. This makes it particularly attractive for large-scale inverse problems, such as those of seismic data processing and interpretation. However, it does require explicit computation of the adjoint operator. Jon Claerbout (1985; 1994) shows dozens of successful examples of the conjugate gradient application with numerically precise adjoint operators.

The motivation for this tutorial is to explore the possibility of using different types of preconditioning operators in the place of adjoints in iterative least-square inversion. For some linear or linearized operators, implementing the exact adjoint may pose a difficult problem. For others, one may prefer different preconditioners because of their smoothness (Claerbout, 1995a; Crawley, 1995a), simplicity (Kleinman and van den Berg, 1991), or asymptotic properties (Sevink and Herman, 1994). In those cases, we could apply the natural generalization

[1] **email:** sergey@sep.stanford.edu

of the conjugate gradient method, which is the method of conjugate directions. The cost difference between those two methods is in the volume of memory storage. In the days when the conjugate gradient method was invented, this difference looked too large to even consider a practical application of conjugate directions. With the evident increase of computer power over the last 30 years, we can afford to do it now.

I derive the main equations used in the conjugate-direction method from very general optimization criteria, with minimum restrictions implied. The textbook algebra is illustrated with a ratfor program and three simple examples.

IN SEARCH OF THE MINIMUM

We are looking for the solution of the linear operator equation

$$\mathbf{d} = \mathbf{A}\,\mathbf{m}\,, \tag{1}$$

where **m** is the unknown model in the linear model space, **d** stands for the given data, and **A** is the forward modeling operator. The data vector **d** belongs to a Hilbert space with a defined norm and dot product. The solution is constructed by iterative steps in the model space, starting from an initial guess $\mathbf{m}_0$. Thus, at the n-th iteration, the current model $\mathbf{m}_n$ is found by the recursive relation

$$\mathbf{m}_n = \mathbf{m}_{n-1} + \alpha_n \mathbf{s}_n\ , \tag{2}$$

where $\mathbf{s}_n$ denotes the step direction, and α_n stands for the scaling coefficient. The residual at the n-th iteration is defined by

$$\mathbf{r}_n = \mathbf{d} - \mathbf{A}\,\mathbf{m}_n\ . \tag{3}$$

Substituting (2) into (3) leads to the equation

$$\mathbf{r}_n = \mathbf{r}_{n-1} - \alpha_n \mathbf{A}\,\mathbf{s}_n\ . \tag{4}$$

For a given step $\mathbf{s}_n$, we can choose α_n to minimize the squared norm of the residual

$$\|\mathbf{r}_n\|^2 = \|\mathbf{r}_{n-1}\|^2 - 2\alpha_n\,(\mathbf{r}_{n-1}, \mathbf{A}\,\mathbf{s}_n) + \alpha_n^2\,\|\mathbf{A}\,\mathbf{s}_n\|^2\ . \tag{5}$$

The parentheses denote the dot product, and $\|\mathbf{x}\| = \sqrt{(\mathbf{x},\mathbf{x})}$ denotes the norm of x in the corresponding Hilbert space. The optimal value of α_n is easily found from equation (5) to be

$$\alpha_n = \frac{(\mathbf{r}_{n-1}, \mathbf{A}\,\mathbf{s}_n)}{\|\mathbf{A}\,\mathbf{s}_n\|^2}\ . \tag{6}$$

Two important conclusions immediately follow from this fact. First, substituting the value of α_n from formula (6) into equation (4) and multiplying both sides of this equation by $\mathbf{r}_n$, we can conclude that

$$(\mathbf{r}_n, \mathbf{A}\,\mathbf{s}_n) = 0\ , \tag{7}$$

which means that the new residual is orthogonal to the corresponding step in the residual space. This situation is schematically shown in Figure 1. Second, substituting formula (6) into (5), we can conclude that the new residual decreases according to

$$\|\mathbf{r}_n\|^2 = \|\mathbf{r}_{n-1}\|^2 - \frac{(\mathbf{r}_{n-1}, \mathbf{A}\,\mathbf{s}_n)^2}{\|\mathbf{A}\,\mathbf{s}_n\|^2}\,, \tag{8}$$

("Pythagoras's theorem"), unless $\mathbf{r}_{n-1}$ and $\mathbf{A}\,\mathbf{s}_n$ are orthogonal. These two conclusions are the basic features of optimization by the method of steepest descent. They will help us define an improved search direction at each iteration.

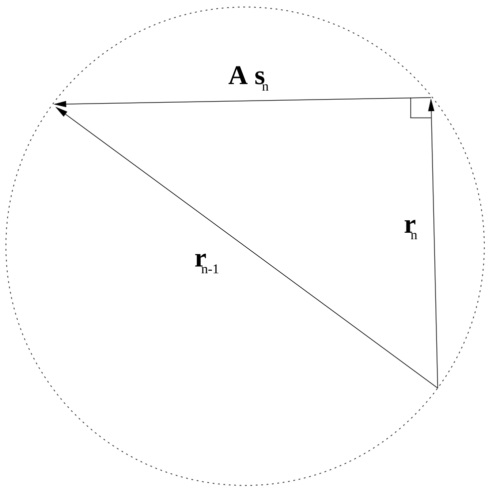


Figure 1: Geometry of the residual in the data space (a scheme). $\boxed{\text{cdstep-dirres}}$ [NR]

IN SEARCH OF THE DIRECTION

Let's suppose we have a generator that provides particular search directions at each step. The new direction can be the gradient of the objective function (as in the method of steepest descent), some other operator applied on the residual from the previous step, or, generally speaking, any arbitrary vector in the model space. Let us denote the automatically generated direction by $\mathbf{c}_n$. According to formula (8), the residual decreases as a result of choosing this direction by

$$\|\mathbf{r}_{n-1}\|^2 - \|\mathbf{r}_n\|^2 = \frac{(\mathbf{r}_{n-1}, \mathbf{A}\,\mathbf{c}_n)^2}{\|\mathbf{A}\,\mathbf{c}_n\|^2}\,. \tag{9}$$

How can we improve on this result?

First step of the improvement

Assuming $n > 1$, we can add some amount of the previous step $\mathbf{s}_{n-1}$ to the chosen direction $\mathbf{c}_n$ to produce a new search direction $\mathbf{s}_n^{(n-1)}$, as follows:

$$\mathbf{s}_n^{(n-1)} = \mathbf{c}_n + \beta_n^{(n-1)}\,\mathbf{s}_{n-1}\,, \tag{10}$$

where $\beta_n^{(n-1)}$ is an adjustable scalar coefficient. According to to the fundamental orthogonality principle (7),

$$(\mathbf{r}_{n-1}, \mathbf{A}\,\mathbf{s}_{n-1}) = 0\,. \tag{11}$$

As follows from equation (11), the numerator on the right-hand side of equation (9) is not affected by the new choice of the search direction:

$$\left(\mathbf{r}_{n-1}, \mathbf{A}\,\mathbf{s}_n^{(n-1)}\right)^2 = \left[(\mathbf{r}_{n-1}, \mathbf{A}\,\mathbf{c}_n) + \beta_n^{(n-1)}\,(\mathbf{r}_{n-1}, \mathbf{A}\,\mathbf{s}_{n-1})\right]^2 = (\mathbf{r}_{n-1}, \mathbf{A}\,\mathbf{c}_n)^2\,. \tag{12}$$

However, we can use transformation (10) to decrease the denominator in (9), thus further decreasing the residual $\mathbf{r}_n$. We achieve the minimization of the denominator

$$\|\mathbf{A}\,\mathbf{s}_n^{(n-1)}\|^2 = \|\mathbf{A}\,\mathbf{c}_n\|^2 + 2\,\beta_n^{(n-1)}\,(\mathbf{A}\,\mathbf{c}_n, \mathbf{A}\,\mathbf{s}_{n-1}) + \left(\beta_n^{(n-1)}\right)^2\,\|\mathbf{A}\,\mathbf{s}_{n-1}\|^2 \tag{13}$$

by choosing the coefficient $\beta_n^{(n-1)}$ to be

$$\beta_n^{(n-1)} = -\frac{(\mathbf{A}\,\mathbf{c}_n, \mathbf{A}\,\mathbf{s}_{n-1})}{\|\mathbf{A}\,\mathbf{s}_{n-1}\|^2}\,. \tag{14}$$

Note the analogy between (14) and (6). Analogously to (7), equation (14) is equivalent to the orthogonality condition

$$\left(\mathbf{A}\,\mathbf{s}_n^{(n-1)}, \mathbf{A}\,\mathbf{s}_{n-1}\right) = 0\,. \tag{15}$$

Analogously to (8), applying formula (14) is also equivalent to defining the minimized denominator as

$$\|\mathbf{A}\,\mathbf{c}_n^{(n-1)}\|^2 = \|\mathbf{A}\,\mathbf{c}_n\|^2 - \frac{(\mathbf{A}\,\mathbf{c}_n, \mathbf{A}\,\mathbf{s}_{n-1})^2}{\|\mathbf{A}\,\mathbf{s}_{n-1}\|^2}\,. \tag{16}$$

Second step of the improvement

Now let us assume $n > 2$ and add some amount of the step from the $(n-2)$-th iteration to the search direction, determining the new direction $\mathbf{s}_n^{(n-2)}$, as follows:

$$\mathbf{s}_n^{(n-2)} = \mathbf{s}_n^{(n-1)} + \beta_n^{(n-2)}\,\mathbf{s}_{n-2}\,. \tag{17}$$

We can deduce that after the second change, the value of numerator in equation (9) is still the same:

$$\left(\mathbf{r}_{n-1}, \mathbf{A}\,\mathbf{s}_n^{(n-2)}\right)^2 = \left[(\mathbf{r}_{n-1}, \mathbf{A}\,\mathbf{c}_n) + \beta_n^{(n-2)}\,(\mathbf{r}_{n-1}, \mathbf{A}\,\mathbf{s}_{n-2})\right]^2 = (\mathbf{r}_{n-1}, \mathbf{A}\,\mathbf{c}_n)^2\,. \tag{18}$$

This remarkable fact occurs as the result of transforming the dot product $(\mathbf{r}_{n-1}, \mathbf{A}\,\mathbf{s}_{n-2})$ with the help of equation (4):

$$(\mathbf{r}_{n-1}, \mathbf{A}\,\mathbf{s}_{n-2}) = (\mathbf{r}_{n-2}, \mathbf{A}\,\mathbf{s}_{n-2}) - \alpha_{n-1}\,(\mathbf{A}\,\mathbf{s}_{n-1}, \mathbf{A}\,\mathbf{s}_{n-2}) = 0\,. \tag{19}$$

The first term in (19) is equal to zero according to formula (7); the second term is equal to zero according to formula (15). Thus we have proved the new orthogonality equation

$$(\mathbf{r}_{n-1}, \mathbf{A}\,\mathbf{s}_{n-2}) = 0 \,, \tag{20}$$

which in turn leads to the numerator invariance (18). The value of the coefficient $\beta_n^{(n-2)}$ in (17) is defined analogously to (14) as

$$\beta_n^{(n-2)} = -\frac{\left(\mathbf{A}\,\mathbf{s}_n^{(n-1)}, \mathbf{A}\,\mathbf{s}_{n-2}\right)}{\|\mathbf{A}\,\mathbf{s}_{n-2}\|^2} = -\frac{(\mathbf{A}\,\mathbf{c}_n, \mathbf{A}\,\mathbf{s}_{n-2})}{\|\mathbf{A}\,\mathbf{s}_{n-2}\|^2} \,, \tag{21}$$

where we have again used equation (15). If $\mathbf{A}\,\mathbf{s}_{n-2}$ is not orthogonal to $\mathbf{A}\,\mathbf{c}_n$, the second step of the improvement leads to a further decrease of the denominator in (8) and, consequently, to a further decrease of the residual.

Induction

Continuing by induction the process of adding a linear combination of the previous steps to the arbitrarily chosen direction $\mathbf{c}_n$ (known in mathematics as the *Gram-Schmidt orthogonalization process*), we finally arrive at the complete definition of the new step $\mathbf{s}_n$, as follows:

$$\mathbf{s}_n = \mathbf{s}_n^{(1)} = \mathbf{c}_n + \sum_{j=1}^{j=n-1} \beta_n^{(j)}\,\mathbf{s}_j \,. \tag{22}$$

Here the coefficients $\beta_n^{(j)}$ are defined by equations

$$\beta_n^{(j)} = -\frac{\left(\mathbf{A}\,\mathbf{c}_n, \mathbf{A}\,\mathbf{s}_j\right)}{\|\mathbf{A}\,\mathbf{s}_j\|^2} \,, \tag{23}$$

which correspond to the orthogonality principles

$$\left(\mathbf{A}\,\mathbf{s}_n, \mathbf{A}\,\mathbf{s}_j\right) = 0 \,, \;\; 1 \le j \le n-1 \tag{24}$$

and

$$\left(\mathbf{r}_n, \mathbf{A}\,\mathbf{s}_j\right) = 0 \,, \; 1 \le j \le n \,. \tag{25}$$

It is these orthogonality properties that allowed us to optimize the search parameters one at a time instead of solving the n-dimensional system of optimization equations for α_n and $\beta_n^{(j)}$.

ALGORITHM

The results of the preceding sections define the method of conjugate directions to consist of the following algorithmic steps:

1. Choose initial model $\mathbf{m}_0$ and compute the residual $r_0 = \mathbf{d} - \mathbf{A}\,\mathbf{m}_0$.

2. At n-th iteration, choose the initial search direction $\mathbf{c}_n$.

3. If n is greater than 1, optimize the search direction by adding a linear combination of the previous directions, according to equations (22) and (23), and compute the modified step direction $\mathbf{s}_n$.

4. Find the step length α_n according to equation (6). The orthogonality principles (24) and (7) can simplify this equation to the form

$$\alpha_n = \frac{(\mathbf{r}_{n-1}, \mathbf{A}\,\mathbf{c}_n)}{\|\mathbf{A}\,\mathbf{s}_n\|^2}\ . \tag{26}$$

5. Update the model $\mathbf{m}_n$ and the residual $\mathbf{r}_n$ according to equations (2) and (4).

6. Repeat iterations until the residual decreases to the required accuracy or as long as it is practical.

At each of the subsequent steps, the residual is guaranteed not to increase according to equation (8). Furthermore, optimizing the search direction guarantees that the convergence rate doesn't decrease in comparison with (9). The only assumption we have to make to arrive at this conclusion is that the operator $\mathbf{A}$ is linear. However, without additional assumptions, we cannot guarantee global convergence of the algorithm to the least-square solution of equation (1) in a finite number of steps.

WHAT ARE ADJOINTS FOR? THE METHOD OF CONJUGATE GRADIENTS

The adjoint operator $\mathbf{A}^T$ projects the data space back to the model space and is defined by the dot product test

$$(\mathbf{d}, \mathbf{A}\,\mathbf{m}) \equiv \left(\mathbf{A}^T\,\mathbf{d}, \mathbf{m}\right) \tag{27}$$

for any $\mathbf{m}$ and $\mathbf{d}$. The method of conjugate gradients is a particular case of the method of conjugate directions, where the initial search direction $\mathbf{c}_n$ is

$$\mathbf{c}_n = \mathbf{A}^T\,\mathbf{r}_{n-1}\ . \tag{28}$$

This direction is often called the *gradient,* because it corresponds to the local gradient of the squared residual norm with respect to the current model $\mathbf{m}_{n-1}$. Aligning the initial search direction along the gradient leads to the following remarkable simplifications in the method of conjugate directions.

Orthogonality of the gradients

The orthogonality principle (25) transforms according to the dot-product test (27) to the form

$$\left(\mathbf{r}_{n-1}, \mathbf{A}\,\mathbf{s}_j\right) = \left(\mathbf{A}^T\,\mathbf{r}_{n-1}, \mathbf{s}_j\right) = \left(\mathbf{c}_n, \mathbf{s}_j\right) = 0\;,\;\; 1 \le j \le n-1\;. \tag{29}$$

Forming the dot product $\left(\mathbf{c}_n, \mathbf{c}_j\right)$ and applying formula (22), we can see that

$$\left(\mathbf{c}_n, \mathbf{c}_j\right) = \left(\mathbf{c}_n, \mathbf{s}_j - \sum_{i=1}^{i=j-1} \beta_n^{(i)}\,\mathbf{s}_i\right) = \left(\mathbf{c}_n, \mathbf{s}_j\right) - \sum_{i=1}^{i=j-1} \beta_n^{(i)}\left(\mathbf{c}_n, \mathbf{s}_i\right) = 0\;,\;\; 1 \le j \le n-1\;. \tag{30}$$

Equation (30) proves the orthogonality of the gradient directions from different iterations. Since the gradients are orthogonal, after n iterations they form a basis in the n-dimensional space. In other words, if the model space has n dimensions, each vector in this space can be represented by a linear combination of the gradient vectors formed by n iterations of the conjugate-gradient method. This is true as well for the vector $\mathbf{m}_0 - \mathbf{m}$, which points from the solution of equation (1) to the initial model estimate $\mathbf{m}_0$. Neglecting computational errors, it takes exactly n iterations to find this vector by successive optimization of the coefficients. This proves that the conjugate-gradient method converges to the exact solution in a finite number of steps (assuming that the model belongs to a finite-dimensional space).

The method of conjugate gradients simplifies formula (26) to the form

$$\alpha_n = \frac{\left(\mathbf{r}_{n-1}, \mathbf{A}\,\mathbf{c}_n\right)}{\|\mathbf{A}\,\mathbf{s}_n\|^2} = \frac{\left(\mathbf{A}^T\,\mathbf{r}_{n-1}, \mathbf{c}_n\right)}{\|\mathbf{A}\,\mathbf{s}_n\|^2} = \frac{\|\mathbf{c}_n\|^2}{\|\mathbf{A}\,\mathbf{s}_n\|^2}\;, \tag{31}$$

which in turn leads to the simplification of formula (8), as follows:

$$\|\mathbf{r}_n\|^2 = \|\mathbf{r}_{n-1}\|^2 - \frac{\|\mathbf{c}_n\|^4}{\|\mathbf{A}\,\mathbf{s}_n\|^2}\;. \tag{32}$$

If the gradient is not equal to zero, the residual is guaranteed to decrease. If the gradient is equal to zero, we have already found the solution.

Short memory of the gradients

Substituting the gradient direction (28) into formula (23) and applying formulas (4) and (27), we can see that

$$\beta_n^{(j)} = \frac{\left(\mathbf{A}\,\mathbf{c}_n, \mathbf{r}_j - \mathbf{r}_{j-1}\right)}{\alpha_j\,\|\mathbf{A}\,\mathbf{s}_j\|^2} = \frac{\left(\mathbf{c}_n, \mathbf{A}^T\,\mathbf{r}_j - \mathbf{A}^T\,\mathbf{r}_{j-1}\right)}{\alpha_j\,\|\mathbf{A}\,\mathbf{s}_j\|^2} = \frac{\left(\mathbf{c}_n, \mathbf{c}_{j+1} - \mathbf{c}_j\right)}{\alpha_j\,\|\mathbf{A}\,\mathbf{s}_j\|^2}\;. \tag{33}$$

The orthogonality condition (30) and the definition of the coefficient α_j from equation (31) further transform this formula to the form

$$\beta_n^{(n-1)} = \frac{\|\mathbf{c}_n\|^2}{\alpha_{n-1}\|\mathbf{A}\,\mathbf{s}_{n-1}\|^2} = \frac{\|\mathbf{c}_n\|^2}{\|\mathbf{c}_{n-1}\|^2}\;, \tag{34}$$

$$\beta_n^{(j)} = 0\;,\;\; 1 \le j \le n-2\;. \tag{35}$$

Equation (35) shows that the conjugate-gradient method needs to remember only the previous step direction in order to optimize the search at each iteration. This is another remarkable property distinguishing that method in the family of conjugate-direction methods.

PROGRAM

The following ratfor program, `cdstep()`, implements one iteration of the conjugate-direction method. It is based upon Jon Claerbout's `cgstep()` program (Claerbout, 1994) and uses an analogous naming convention. Vectors in the data space are denoted by double letters.

```
# A step of conjugate-direction descent.
#
subroutine cdstep( niter,iter,   n, x, g, s,   m, rr, gg, ss,ssn)
integer i,j,liter, niter,iter,   n,            m
real    x(n), rr(m),            # solution,  residual
        g(n), gg(m),            # direction, conjugate direction
        s(n,niter), ss(m,niter) # step,      conjugate step
double precision ddot, alpha, ssn(niter)

liter = min0(iter,niter)

do i= 1, n                                      # initial direction
        s(i,liter)  =  g(i)
do i= 1, m
        ss(i,liter) = gg(i)

do j= 1, liter-1 {                              # update direction
        alpha = ddot(m,gg,ss(1,j))/ssn(j)
do i= 1, n
        s(i,liter)  =  s(i,liter) - alpha *  s(i,j)
do i= 1, m
        ss(i,liter) = ss(i,liter) - alpha * ss(i,j)
}

        ssn(liter) = dmax1(ddot(m,ss(1,liter),ss(1,liter)),1.d-35)

if (liter == niter) {
        do j= 1, niter-1 {
                ssn(j) = ssn(j+1)
                do i= 1, n
                         s(i,j) =  s(i,j+1)
                do i= 1, m
                        ss(i,j) = ss(i,j+1)
        }
}

        alpha = ddot(m,ss(1,liter),rr)/ssn(liter)
do i= 1, n                                      # update solution
        x(i)  =  x(i) + alpha *  s(i,liter)
do i= 1, m                                      # update residual
        rr(i) = rr(i) - alpha * ss(i,liter)

return; end

double precision function ddot( n, x, y)
integer i, n;           real  x(n), y(n);       double precision val
val = 0.;       do i=1,n  { val = val + x(i) * y(i) }
ddot = val;     return;         end
```

In addition to the previous steps $\mathbf{s}_j$ (array `s`) and their conjugate counterparts $\mathbf{A}\mathbf{s}_j$ (array `ss`), the program stores the squared norms $\|\mathbf{A}\mathbf{s}_j\|^2$ (array `ssn`) to avoid recomputation. For practical reasons, the number of remembered iterations `niter` can actually be smaller than the total number of iterations. The value `niter=2` corresponds to the conjugate-gradient method. The value `niter=1` corresponds to the steepest-descent method. The iteration process should start with `iter = 1`, corresponding to the first steepest-descent iteration in the method of conjugate gradients.

EXAMPLES

Example 1: Inverse interpolation

Matthias Schwab has suggested (in a personal communication) an interesting example, in which the `cgstep` program fails to comply with the conjugate-gradient theory. The inverse problem is a simple one-dimensional data interpolation with a known filter (Claerbout, 1994). The known portion of the data is a single spike in the middle. One hundred other data points are considered missing. The known filter is the Laplacian $(1,-2,1)$, and the expected result is a bell-shaped cubic spline. The forward problem is strictly linear, and the exact adjoint is easily computed by reverse convolution. However, the conjugate-gradient program requires significantly more than the theoretically predicted 100 iterations. Figure 2 displays the convergence to the final solution in three different plots. According to the figure, the actual number of iterations required for convergence is about 300. Figure 3 shows the result of a similar experiment with the conjugate-direction solver `cdstep`. The number of required iterations is reduced to almost the theoretical one hundred. This indicates that the orthogonality of directions implied in the conjugate-gradient method has been distorted by computational errors. The additional cost of correcting these errors with the conjugate-direction solver comes from storing the preceding 100 directions in memory. A smaller number of memorized steps produces smaller improvements.

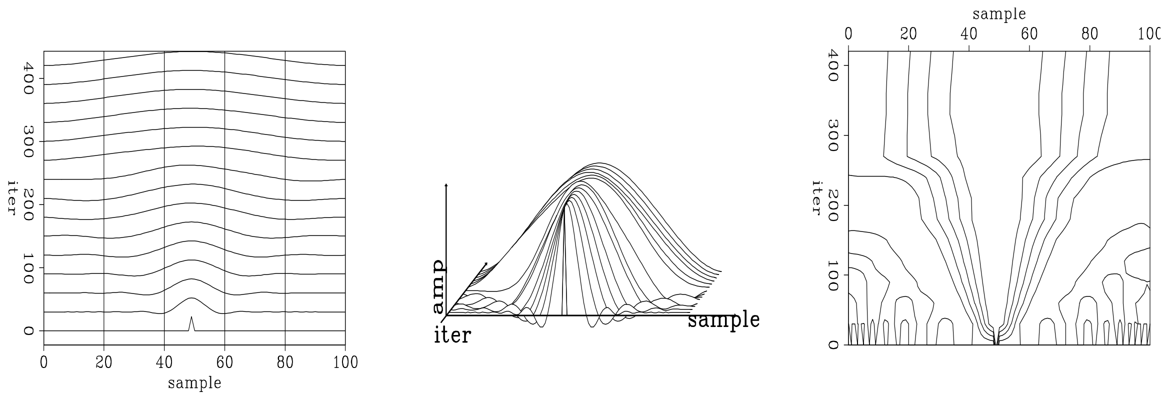


Figure 2: Convergence of the missing data interpolation problem with the conjugate-gradient solver. Current models are plotted against the number of iterations. The three plots are different displays of the same data. $\boxed{\text{cdstep-dirmcg}}$ [ER]

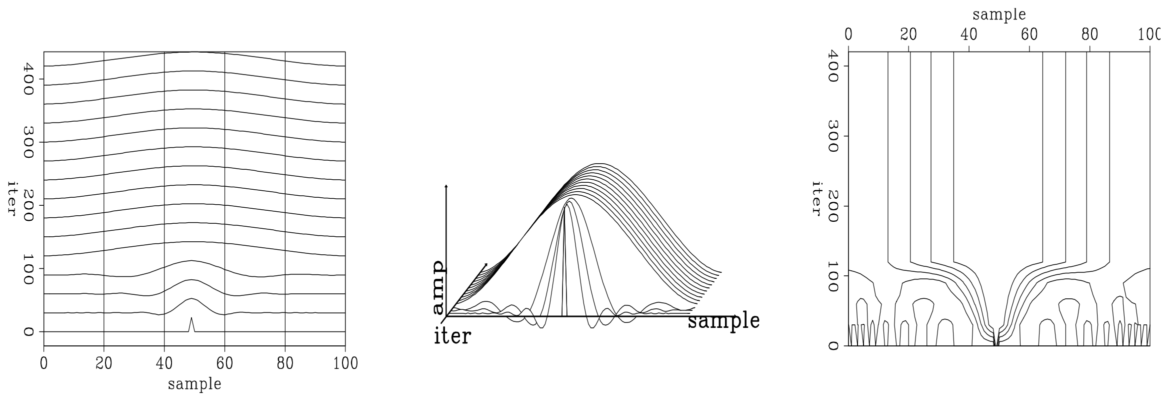


Figure 3: Convergence of the missing data interpolation problem with the long-memory conjugate-direction solver. Current models are plotted against the number of iterations. The three plots are different displays of the same data. cdstep-dirmcd [ER]

Example 2: Velocity transform

The next test example is the velocity transform inversion with a CMP gather from the Mobil AVO dataset (Nichols, 1994; Lumley et al., 1994; Lumley, 1994). I use Jon Claerbout's `veltran` program (Claerbout, 1995b) for anti-aliased velocity transform with rho-filter preconditioning and compare three different pairs of operators for inversion. The first pair is the CMP stacking operator with the "migration" weighting function $\left(w = \frac{(t_0/t)}{\sqrt{t}}\right)$ and its adjoint. The second pair is the "pseudo-unitary" velocity transform with the weighting proportional to $\sqrt{|s\,x|}$, where x is the offset and s is the slowness. These two pairs were used in the velocity transform inversion with the iterative conjugate-gradient solver. The third pair uses the weight proportional to $|x|$ for CMP stacking and $|s|$ for the reverse operator. Since these two operators are not exact adjoints, it is appropriate to apply the method of conjugate directions for inversion. The convergence of the three different inversions is compared in Figure 4. We can see that the third method reduces the least-square residual error, though it has a smaller effect than that of the pseudo-unitary weighting in comparison with the uniform one. The results of inversion after 10 conjugate-gradient iterations are plotted in Figures 5 and 6, which are to be compared with the analogous results of David Lumley (1994) and Dave Nichols (1994).

Example 3: Leveled inverse interpolation

The third example is the linearized nonlinear inversion for interpolating the SeaBeam dataset (Claerbout, 1994; Crawley, 1995b). This interpolation problem is nonlinear because the prediction-error filter is estimated simultaneously with the missing data. The conjugate-gradient solver showed a very slow convergence in this case. Figure 7 compares the results of the conjugate-gradient and conjugate-direction methods after 2500 iterations. Because of the large scale of the problem, I set `niter=4` in the `cdstep()` program, storing only the three preceding steps of the conjugate-direction optimization. The acceleration of convergence produced

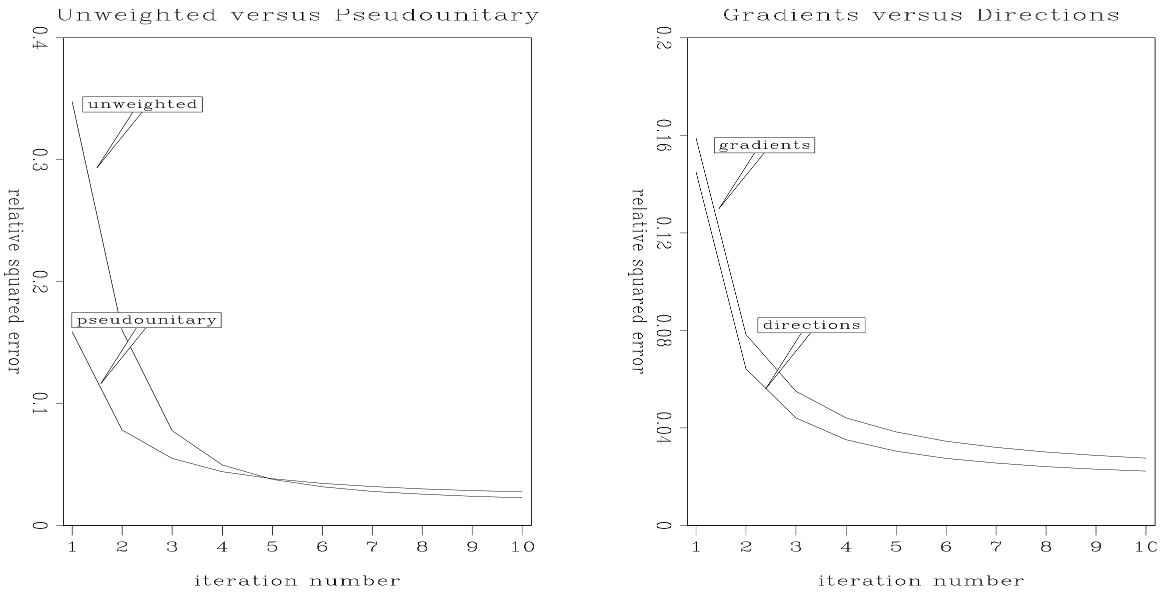


Figure 4: Comparison of convergence of the iterative velocity transform inversion. The left plot compares conjugate-gradient inversion with unweighted (uniformly weighted) and pseudo-unitary operators. The right plot compares pseudo-unitary conjugate-gradient and weighted conjugate-direction inversion. cdstep-diritr [ER]

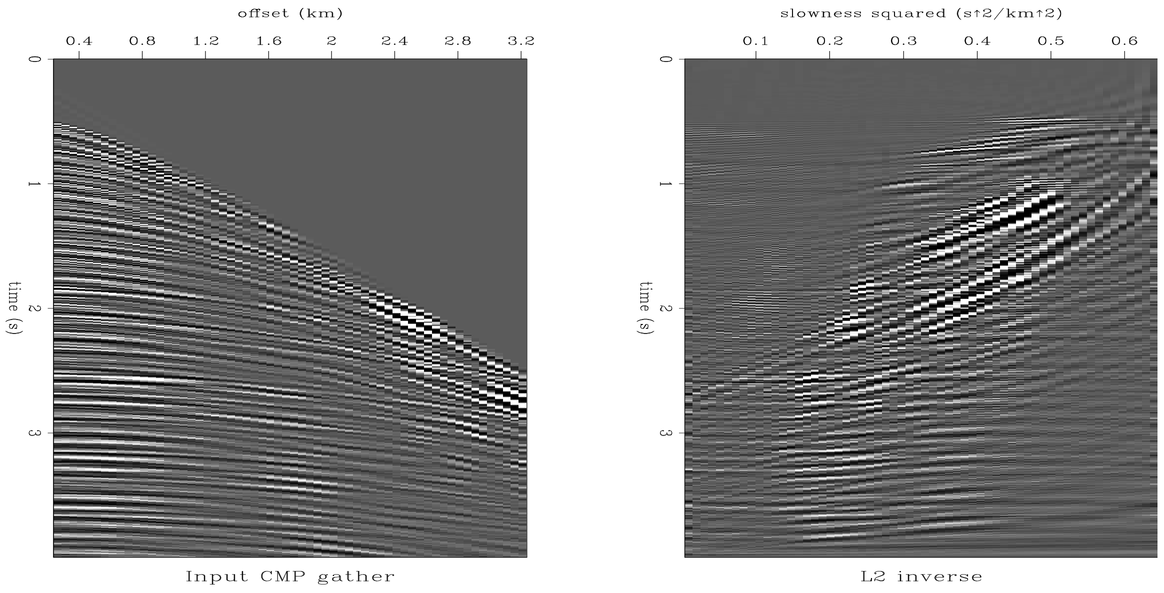
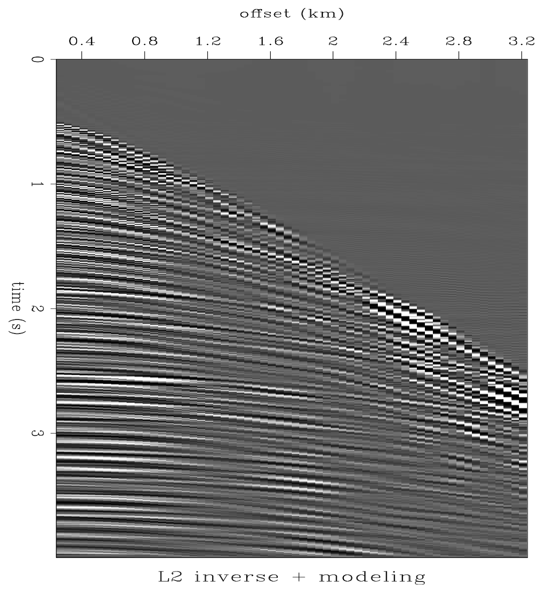


Figure 5: Input CMP gather (left) and its velocity transform counterpart (right) after 10 iterations of conjugate-direction inversion. cdstep-dircvv [ER]



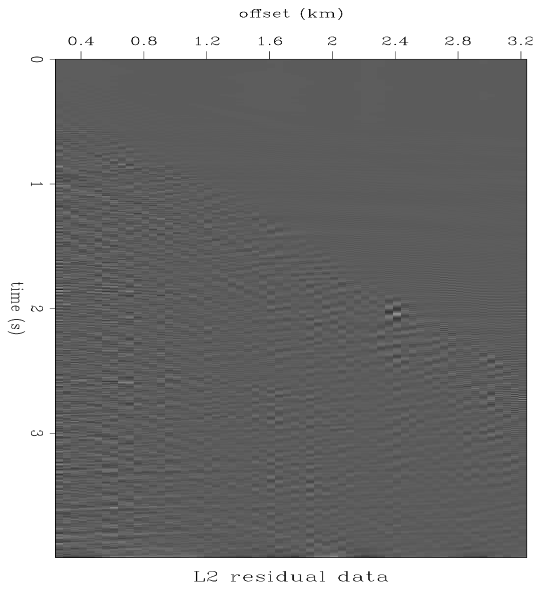


Figure 6: The modeled CMP gather (left) and the residual data (right) plotted at the same scale. cdstep-dirrst [ER]

a noticeably better interpolation, which is visible in the figure.

CONCLUSIONS

The conjugate-gradient solver is a powerful method of least-square inversion because of its remarkable algebraic properties. In practice, the theoretical basis of conjugate gradients can be distorted by computational errors. In some applications of inversion, we may want to do that on purpose, by applying inexact adjoints in preconditioning. In both cases, a safer alternative is the method of conjugate directions. Jon Claerbout's `cgstep()` program actually implements a short-memory version of the conjugate-direction method. Extending the length of the memory raises the cost of iterations, but can speed up the convergence.

REFERENCES

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

Claerbout, J. F., 1994, Applications of Three-Dimensional Filtering: Stanford Exploration Project.

Claerbout, J., 1995a, Ellipsoids versus hyperboloids: SEP–**89**, 201–205.

Claerbout, J. F., 1995b, Basic Earth Imaging: Stanford Exploration Project.

Crawley, S., 1995a, Approximate vs. exact adjoints in inversion: SEP–**89**, 207–215.

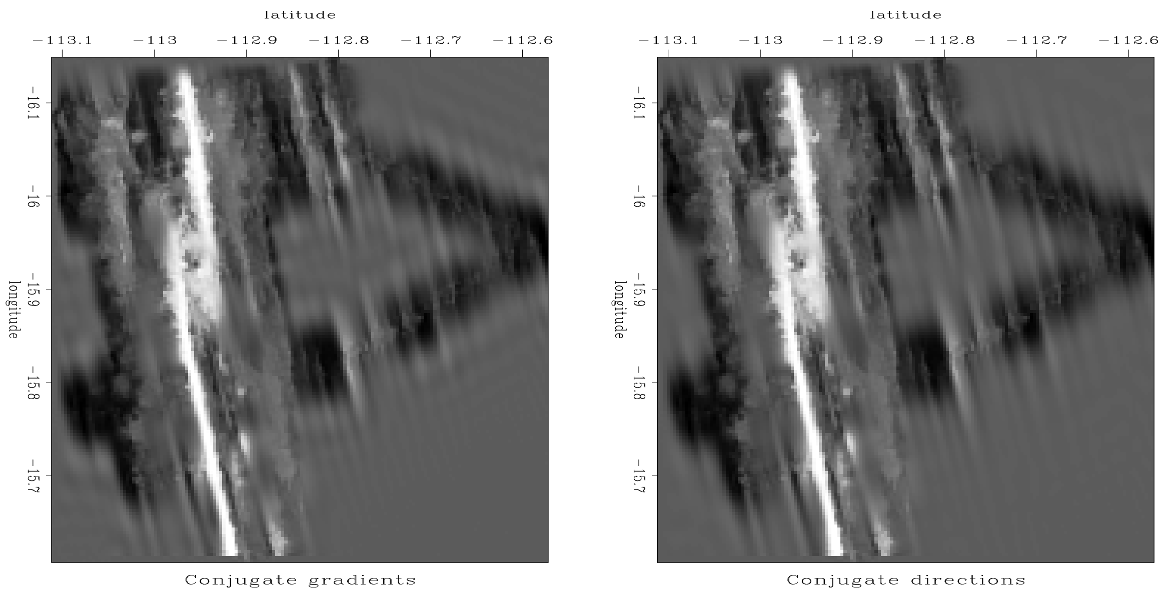


Figure 7: SeaBeam interpolation. Left plot: the result of the conjugate-gradient inversion after 2500 iterations. Right plot: the result of the short-memory conjugate-direction inversion after 2500 iterations. cdstep-dirjbm [CR]

Crawley, S., 1995b, Multigrid nonlinear SeaBeam interpolation: SEP–**84**, 279–288.

Fletcher, R., and Reeves, C. M., 1964, Function minimization by conjugate gradients: Computer Journal, **7**, 149–154.

Gill, P. E., Murray, W., and Wright, M. H., 1995, Practical optimization: Academic Press.

Hestenes, M. R., and Stiefel, E., 1952, Methods of conjugate gradients for solving linear systems: J. Res. NBS, **49**, 409–436.

Kleinman, R. E., and van den Berg, P. M., 1991, Iterative methods for solving integral equations: Radio Science, **26**, 175–181.

Lumley, D., Nichols, D., and Rekdal, T., 1994, Amplitude-preserved multiple suppression: SEP–**82**, 25–45.

Lumley, D. E., 1994, Estimating a pseudounitary operator for velocity-stack inversion: SEP–**82**, 63–78.

Nichols, D., 1994, Velocity-stack inversion using $\mathbf{L_p}$ norms: SEP–**82**, 1–16.

Sevink, A. G. J., and Herman, G. C., 1994, Fast iterative solution of sparsely sampled seismic inverse problems: Inverse Problems, **10**, 937–948.