Stanford Exploration Project, Report 89, November 12, 1997, pages 209–218

Stanford Exploration Project, Report 89, November 12, 1997, pages 209–218

Approximate vs. exact adjoints in inversion

Sean Crawley[1]

keywords: *inversion, velocity, approximation, wave equation, stacking, modeling*

ABSTRACT

Gradient-based inversion methods employ adjoint operators to find the gradient direction in the objective function. Generally we assume that the exact adjoint is the optimal choice. In this paper I discuss one type of approximate adjoint, called a pull adjoint because of the way it discretizes the output space. For operators which sum and spray data along curved trajectories, such as Kirchoff and move-out operators, pull adjoints more closely approximate an operator in a continuous space. This characteristic is important because we use these operators to emulate the wave equation, which is a continuous and unitary operator. This paper presents a theoretical argument for the use of pull adjoints in some inversions, and presents a simple inversion example where the use of pull adjoints is of benefit.

INTRODUCTION

Gradient-based inversion methods often require the use of adjoint operators (Claerbout, 1994). In programming an operator and its adjoint, we generally loop over the same space independent of whether we desire the forward or adjoint operation. This means that in one of the two cases a subroutine/operator loops over its input space and in the other it loops over its output space. For many operators, this detail can have a significant negative effect on the output. For example, we can neither stack reflections nor collapse diffractions perfectly even with exact velocities. A physical explanation for this shortcoming is found in Ji (1992): the wave equation is a unitary operator that we measure in continuous space. However, our operators must work in discrete and finite space. An important factor in approximating a continuous space with a discrete one is evenness of sampling, and we will see that designing operators to loop over input or output space has a major impact on how the output of the operator samples the input.

For wave-equation based operators, it seems always preferable to loop over the output space. This means that each bin in the output space scans the inputs for the appropriate value or values to *pull* in. The number of inputs per output varies

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

smoothly across the output space. Looping the other way, *pushing* values from input bins into whatever outputs are appropriate, carries with it the inevitability that the number of input values per output will not be a smooth function, and the possibility that it may actually be zero in some areas. However, looping over inputs is a natural consequence of using adjoints to find the gradient in an inverse problem, and so seems more or less unavoidable.

The question arises whether it is better to use an approximate adjoint, which will not generally provide the exact gradient direction for the inversion, but which will have smoothly sampled outputs in both the forward and adjoint operations. We can recode our operators so that they loop over the output space in both the forward and adjoint directions. Claerbout (1995) labels such an operator a *pull adjoint.* Such an operator does not precisely fit our conjugate gradient optimization theory, but is attractive because it more closely approximates an operator in a continuum, and is closer to being a unitary operator like the wave equation.

In this paper, I describe some of the details behind implementing pull adjoints, and present examples which highlight their 0.

DESIGNING PULL OPERATORS

A couple of implementation details are worthy of discussion. In particular, how does one go about designing operators to always loop over inputs or outputs, and how does one design weighting functions so that the operator pairs are as near to exact adjoints as possible?

It is worth pointing out that, though this paper speaks of the attractiveness of approximate adjoints, the goal is not to design an operator pair which fails the dot product test, but rather to design an operator pair which more closely approximates the continuous and unitary properties of the wave equation. Changing the operator changes the objective function, and the hope is that by recoding operator pairs so that they will always operate as pull operators, we will end up with an objective function that is easier to minimize. The idea is analogous to changing the elliptical objective function for a two parameter problem to a spherical one. The number of iterations required in theory is unchanged, but in practice it is halved.

Discretizing the output space

To clarify all this talk of operators which push and pull, here are a few samples of code to show exactly what pushing and pulling mean from an operator design perspective. Below is the computational part of the subroutine `nmo1` from Jon's book *Basic Earth Imaging.*

```
do iz= 1, n {z = t0 + dt*(iz-1)
```

```
         xs = x * slow(iz)
         t = sqrt ( z * z + xs * xs ) + 1.e-20
         wt = z/t * (1./sqrt(t))    #weighting function
         it = 1 + .5 + (t - t0) / dt
         if( it <= n )
           if( conj == 0 )
               tt(it) = tt(it) + zz(iz) * wt
           else
               zz(iz) = zz(iz) + tt(it) * wt
        }
```

One can see that in both the forward (`conj == 0`) and adjoint operation, a loop over `z` (which represents vertical travel time τ) is performed. This means that each sample along the zero offset trace is visited once. Time t is calculated and nearest-neighbor interpolation is performed to find the correct sample along the non-zero offset trace. A given sample along the non-zero offset trace may be visited multiple times or not at all.

An alternative is to use two subroutines. The first performs the adjoint operation exactly as above (I will avoid reprinting it and just say that the differences are that there is no test for `conj`, and the weighting function is different). The second subroutine performs the forward operation. In the forward operation, discretizing in the output space means looping over time t.

```
do it= 2, n {   t = t0 + dt*(it-1)
                xs = x * slow(it)
                z = sqrt ( t * t - xs * xs )
                iz = 1 + .5 + (z - t0) / dt
                if( iz >=1 )
                        tt(it) = tt(it) + zz(iz) * z/t
            }
```

Here the non-zero offset trace is sampled evenly, just as the zero-offset trace is above. There is, however, one serious complication to the computation shown directly above. Calculating `z` requires knowing the velocity as a function of travel time t rather than vertical travel time τ. The equation for moveout in a vertically varying velocity field doesn't permit an explicit solution for $v(t)$. This is an important future project. For now I stick to constant velocity.

Weighting functions

Generally, whatever weight is chosen for the forward operator, is also applied to the adjoint. However, because we have changed the way we design our operator, we must rethink our weighting. We can derive the proper weighting function for our Kirchoff and moveout operators in a continuum and apply it to our operator (see (Jedlicka, 1989)). This means that our approximate pull adjoint is an exact adjoint in a continuum.

We can think of a single trace of data $\tilde{d}(t)$, a model trace $m(\tau)$, scalar weights w and $\tilde{w}$, and an operator which performs the change of variables (this could be NMO, for example):

$$m(\tau) = wd\left(t\left(\tau\right)\right) \tag{1}$$

$$\tilde{d}(t) = \tilde{w}\tilde{m}\left(\tau\left(t\right)\right) \tag{2}$$

We can multiply the two equations and integrate both sides with respect to model variable τ:

$$\int m\tilde{m}\, d\tau = \int [\frac{w}{\tilde{w}}] d\tilde{d}\, d\tau \tag{3}$$

and change the variable of integration:

$$\int m\tilde{m}\, d\tau = \int [\frac{w}{\tilde{w}}\frac{d\tau}{dt}] d\tilde{d}\, dt \tag{4}$$

The statement of the dot product test in a continuum is

$$\int m\tilde{m}\, d\tau = \int d\tilde{d}\, dt \tag{5}$$

which means that

$$\frac{w}{\tilde{w}}\frac{d\tau}{dt} = 1 \tag{6}$$

From our NMO and Kirchoff equations,

$$t^2 = \tau^2 + \frac{x^2}{v^2} \tag{7}$$

$$tdt = \tau d\tau \tag{8}$$

$$\frac{dt}{d\tau} = \frac{\tau}{t} \tag{9}$$

This tells us the ratio of one weighting function to the other. However, some experimentation may be required to find the optimal choices. I find good results using $w = z/t$, $\tilde{w} = 1$, but there may be more effective combinations.

NORMAL MOVEOUT

A simple example of an adjoint with rough output is normal moveout. Performing normal moveout on a trace means looping over values of vertical travel time τ and solving for the appropriate travel time t. The trace, collected at some offset, is stretched upward to simulate a zero offset trace. Each output bin collects one value from the input space. Performing the adjoint means pushing each input value into an output bin. Multiple inputs will end up in some output bins. The roughness with which the output space samples the input space is dictated by the velocity.

Here is a simple inversion, following an example from TDF (Claerbout, 1994). Begin with a single seismic trace, which we label the model, m. We also have a solution trace, labeled x, which we wish to construct from a synthetic gather $d = Mm$, where M is a normal moveout 0 operator, and M' its adjoint stacking operator. Thus our regression is:

$$d \approx Mx \tag{10}$$

which is solved by variation of x:

$$\Delta x = M'r \tag{11}$$

where r is the residual, $d - Mx$.

We have some freedom in the way we define M and M'. M is our 0 operator, but we can choose it to sample the input trace or to sample the output gather, and the same is true of the stacking operator M'. In order to be precisely adjoint to each other, both M and M' must sample the same space. Further, the mathematics of normal moveout recommend always sampling the trace, because of the explicit relationship between vertical travel-time τ and velocity. So initially we choose our NMO operator to model by *pushing* values from an input trace into the output gather, and to stack by *pulling* values into the trace. This gives an exact adjoint.

I use this operator pair to solve the above regression and get results as seen in Figure 1. We see that the original waveforms are reconstructed almost exactly after just four or five iterations.

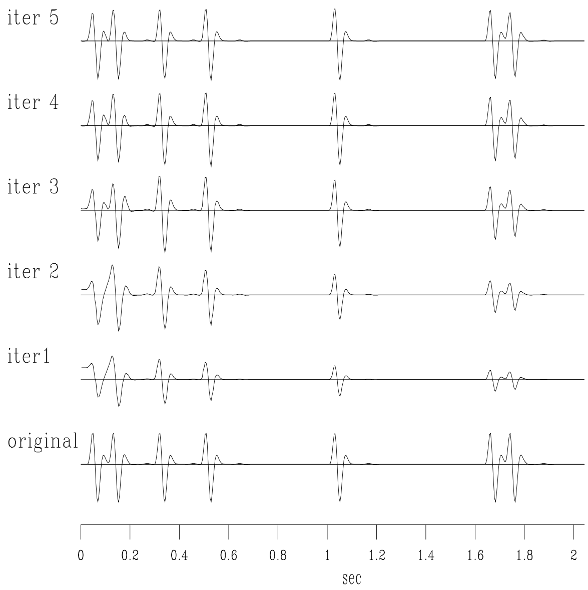

Figure 1: Iterations of the inversion $d \approx Mx$. Below is shown the original trace. Note that the original is duplicated almost exactly after just a few iterations. $\boxed{\text{sean-results}}$ [ER]

Now we redefine the operator. M' stays unchanged, but M is replaced. Maintaining similar naming conventions, we can define a stacking operator S which samples an input gather and pushes values into a stacked trace output. S' then must model by pulling values into the gather. We replace M by S' and obtain the regression:

$$d \approx S'x \tag{12}$$

and the gradient

$$\Delta x = M'r \tag{13}$$

Using this new operator pair in the inversion yields the results seen in Figure 2. Amplitudes for most events are reconstructed more quickly, after a single iteration.

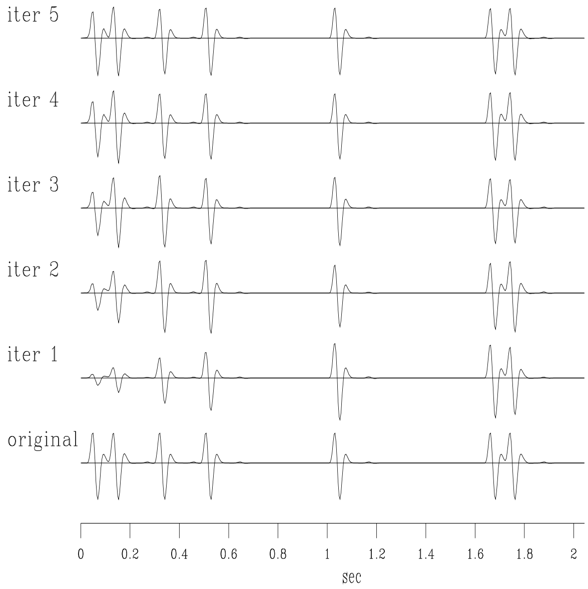

Figure 2: Iterations of the inversion $d \approx S'm$. Below is shown the original trace. Note that amplitudes on most events are recovered more quickly than with the exact adjoint. sean-appnmo [ER]

To quantify the results, Figure 3 displays the misfit of the model at each iteration with the original trace: $||m_{\text{inf}} - m_i||^2$. This misfit is the model residual, normally an unknown quantity, but a useful measure of success in synthetic problems where the exact answer is known. The inversion which uses exclusively pull operators converges more quickly than that which uses exact adjoints. For this simple example, instant convergence in the model residual would mean unitarity; accelerated convergence suggests that the pull adjoint gives something closer to a unitary operator than the exact adjoint.

To satisfy curiosity, I measured data residuals (that quantity minimized by the inversion) after several iterations at various slownesses. The results are shown in Figure 4. The smooth curve corresponds to the pull adjoint and the rough one to the exact adjoint. At all velocities, there is a noticeable advantage to using pull adjoint NMO. It is surprising that there is so little velocity dependence, because it seems the aliasing at high dips (or low velocities) should be a problem for push operators more than for pull operators. Further, the magnitude by which the operator pair which includes the pull adjoint fails the dot product test is not systematically dependent on velocity.

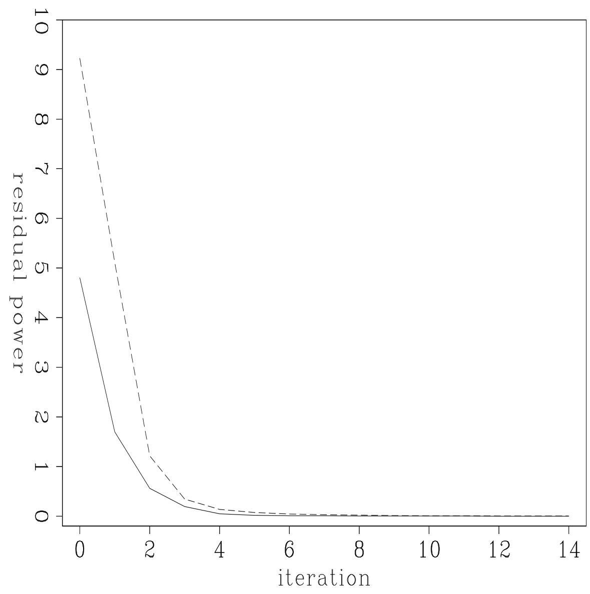

Figure 3: Residuals for the two inversions as a function of iteration. Dashed line denotes inversion with the exact adjoint, continuous line with the pull adjoint. sean-nmores [ER]

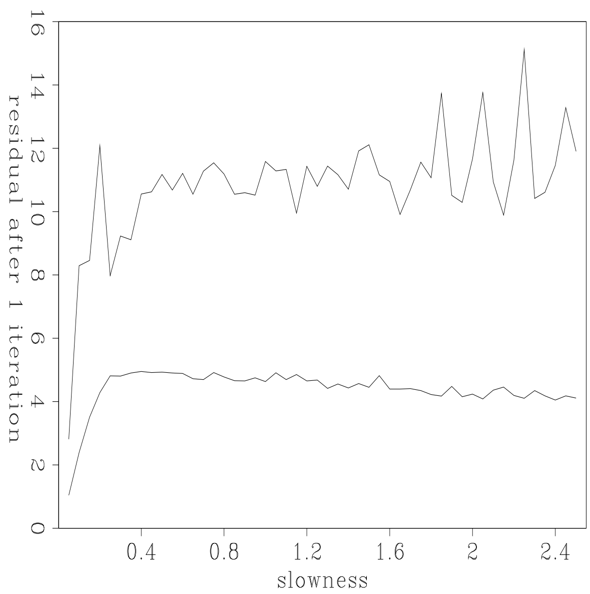

Figure 4: Comparison of residual after one iteration for the pull and exact adjoint. The smooth curve denotes the pull adjoint. Click to see movie of residuals at higher numbers of iterations. sean-slowres [ER]

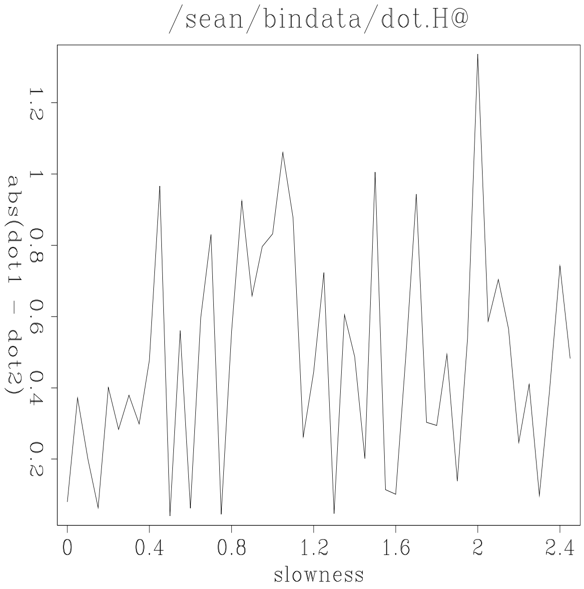

Figure 5: Error in the dot product test is independent of velocity. sean-dotprods [ER]

FUTURE WORK

An obvious necessary step is an effective method for finding the velocities at offset. In particular, the method must be able to handle the situation where velocity is multi-valued.

Also, in the simple synthetic test described here, both the exact and pull adjoint inversions converge to the same value after a sufficient number of iterations. It remains to be seen how their behaviors will differ when applied to real data (if we can change the objective function by changing the operator, perhaps we will discover noticeably higher or lower minima in some cases).

A large number of wave equation based processes seem likely candidates for further investigation of pull adjoints. Sergey Fomel has done work with Kirchoff migration, and Jon Claerbout (Claerbout, 1995) discusses the application of pull adjoints to DMO and other forms of migration.

An important future step is the investigation of whether the use of the new moveout and stacking operator pair can reduce the number of iterations required to perform inversions using more complex series of operators. This opens the possibility of accelerating and/or improving various sorts of inversions in velocity space.

CONCLUSIONS

I present a pseudounitary pull adjoint as an alternative to the exact adjoint in least squares stacking. Pull adjoints are a good choice for wave equation based operators because they more closely approximate an operator in a continuum. An operator pair which includes pull adjoints produces faster convergence in a least squares stacking example than an exactly adjoint operator pair, despite failing the dot product test. This is because they more closely approximate the continuous wave equation than an exactly adjoint operator pair.

ACKNOWLEDGEMENTS

Many of the codes used in this paper were written by Jon Claerbout. Additionally, the derivation of the weighting functions was shown to me by Sergey Fomel.

REFERENCES

Claerbout, J. F., 1994, Applications of three-dimensional filtering: SEP.

Claerbout, J., 1995, Spheroids versus hyperboloids: SEP–**89**.

Jedlicka, J., 1989, Velocity analysis by inversion: SEP–**61**, 41–68.

Ji, J., 1992, Least squares imaging, datuming, and interpolation using the wave equation: SEP–**75**, 111–124.