Stanford Exploration Project, Report 89, November 12, 1997, pages 203–208

Ellipsoids versus hyperboloids

Jon Claerbout[1]

keywords: *modeling, algorithm, conjugate gradient, approximation*

ABSTRACT

Many operators can be coded in either "push" or "pull" forms, namely, you can loop over the space of outputs and find the inputs (pull), or loop over the space of inputs and find the outputs (push). The adjoint of a push operator is a pull operator and vice versa. Migration can be done with either hyperbolas or diffractors. Inversions impose extra accuracy requirements that may lead to forward modeling being a pull operator. A DMO operator is a product of two operators, either of which is a push or pull, leading to four possible representations.

INTRODUCTION

Migration and diffraction operators can be conceived and programmed in two different ways. Let $\vec{\mathbf{t}}$ denote data and $\vec{\mathbf{z}}$ denote the depth image. We have

$$\vec{\mathbf{t}} \;=\; \mathbf{H}\,\vec{\mathbf{z}} \qquad \text{spray or push a Hyperbola into the output} \tag{1}$$

$$\vec{\mathbf{z}} \;=\; \mathbf{C}\,\vec{\mathbf{t}} \qquad \text{spray or push a semiCircle into the output} \tag{2}$$

The adjoints are

$$\vec{\mathbf{z}} \;=\; \mathbf{H}'\,\vec{\mathbf{t}} \qquad \text{sum or pull a Hyperbola from the input} \tag{3}$$

$$\vec{\mathbf{t}} \;=\; \mathbf{C}'\,\vec{\mathbf{z}} \qquad \text{sum or pull a semiCircle from the input} \tag{4}$$

Generally we can choose between the two similar operators $\mathbf{H} \approx \mathbf{C}'$. For modelling we can make $\vec{\mathbf{t}}$ from either $\mathbf{H}\vec{\mathbf{z}}$ or $\mathbf{C}'\vec{\mathbf{z}}$. For migration we can make $\vec{\mathbf{z}}$ from either $\mathbf{C}\vec{\mathbf{t}}$ or $\mathbf{H}'\vec{\mathbf{t}}$.

PUSH VERSUS PULL

A natural question is which of the possibilities is more correct or better. The question of "more correct" applies to modeling and is best answered by theoreticians (who will

[1]**email:** jon@SEP.Stanford.EDU

find more than simply a hyperbola; they will find its waveform including its amplitude and phase as a function of frequency). The question of "better" is something else. An important practical issue is that the transformation should not leave miscellaneous holes in the output. It is typically desireable to write programs that loop over all positions in the *output* space, "pulling" in whatever inputs are required. It is usually less desireable to loop over all positions in the *input* space, "pushing" or "spraying" each input value to the appropriate location in the output space. Programs that push the input data to the output space might leave the output too sparsely distributed. Also, because of gridding, the output data might be irregularly positioned. Thus, to produce smooth outputs, we usually *prefer the summation operators* $\mathbf{H}'$ for migration and $\mathbf{C}'$ for diffraction modeling. Since one could always force smooth outputs by lowpass filtering, what we really seek is the highest possible resolution.

Although practitioners prefer the summation operators, $\mathbf{H}'$ and $\mathbf{C}'$, $\mathbf{H}$ and $\mathbf{C}'$ are only approximately adjoint to each other (or $\mathbf{C} \approx \mathbf{H}'$ are approximately adjoint to each other.) This conflicts with the requirements for sparse solvers which work with an operator pair, either (H, H') or (C, C'). Paige and Saunders (1982) say that their well known LSQR program is expected to fail if the given adjoint is not truely the adjoint, such as the "smooth" pair (H', C').

For data processing we prefer the pull operator into model space rather than a push from data space. For inversion, however, we are matching observed data to theoretical data. For this matching we need accurate (not rough) theoretical data which implies we need the pull operator into data space and the solver needs its adjoint, the push operator. The operator we do not seem to require is $\vec{\mathbf{t}} = \mathbf{H}\,\vec{\mathbf{z}}$ which sprays a model point into a possibly subsampled hyperbola. In summary:

$$\vec{\mathbf{z}} = \mathbf{H}'\,\vec{\mathbf{t}} \qquad \text{smooth data processing} \tag{5}$$
$$\vec{\mathbf{t}} = \mathbf{C}'\,\vec{\mathbf{z}} \qquad \text{smooth modeling needed for data fitting} \tag{6}$$
$$\vec{\mathbf{z}} = \mathbf{C}\,\vec{\mathbf{t}} \qquad \text{CG solver needs for fitting} \tag{7}$$
$$\vec{\mathbf{t}} = \mathbf{H}\,\vec{\mathbf{z}} \qquad \text{make aliased data} \tag{8}$$

Although the requirement of a correct adjoint can be expected of all the standard conjugate-gradient data fitting algorithms, I believe my `cgplus()` conjugate-gradient subroutine is immune to that requirement. I believe this because `cgplus()` uses the adjoint only to suggest a direction for descent using the operator itself to choose the distance moved. In principle, the search line can be random as long as the correct (signed) distance is chosen. Perhaps when used in this way, my `cgplus()` is properly called a "conjugate-direction" algorithm. This conjecture should be resolved.

Do we get better and quicker inversions by using correct adjoints for search directions? or could we get better results with pull-type adjoint approximations that do not yield rough or aliased output?

DIP MOVEOUT

Analogous to the hyperbola push operator $\mathbf{H}$ and semicircle push operator $\mathbf{C}$ defined above, let us define $\mathbf{C}_h$ and $\mathbf{H}_h$ for pushing ellipses and flat-topped hyperbolas. (Here h is half the shot-geophone offset.)

$$\vec{\mathbf{z}} \;=\; \mathbf{C}_h\,\vec{\mathbf{t}} \qquad \text{spray or push an ellipse into the output} \tag{9}$$

$$\vec{\mathbf{t}} \;=\; \mathbf{H}_h\,\vec{\mathbf{z}} \qquad \text{spray or push a flattened hyperbola into the output} \tag{10}$$

Given data on a nonzero-offset section, we seek to convert it to a zero-offset section. Conceptually the simplest approach is to first migrate the constant offset data with an ellipsoid push operator $\mathbf{C}_h$ and then take each point on the ellipsoid and diffract it out to a zero-offset hyperbola with a push operator $\mathbf{H}_0 = \mathbf{H}_{h=0}$. The product of push operators $\mathbf{R} = \mathbf{H}_0\mathbf{C}_h$ is known as Rocca's smile. This smile operator includes both normal moveout and dip moveout. (We could say that dip moveout is defined by Rocca's smile after restoring the normal moveout.)

To visualize the Rocca smile operator more clearly, I broke the continuous ellipsoid into a sequence of dots so you would be able to see individual hyperbolas as well their superposition. This is shown in Figure 1.

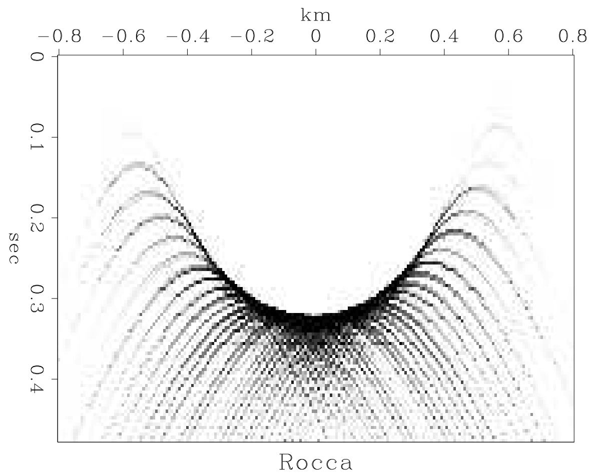

Figure 1: Rocca's smile is a superposition of hyperbolas, each with its top on an ellipse. jon2-frocca [ER]

Because of the approximation $\mathbf{H} \approx \mathbf{C}'$, we have four different ways to express the Rocca smile:

$$\mathbf{R} \;=\; \mathbf{H}_0\mathbf{C}_h \;\approx\; \mathbf{H}_0\mathbf{H}'_h \;\approx\; \mathbf{C}'_0\mathbf{C}_h \;\approx\; \mathbf{C}'_0\mathbf{H}'_h \tag{11}$$

$\mathbf{H}_0\mathbf{H}'_h$ says sum over a flat-top and then spray a regular hyperbola. More interesting is the operator $\mathbf{C}'_0\mathbf{C}_h$ which says spray an ellipse and then sum over a circle. This approach, associated with Gerry Gardner, says that we are interested in all circles that are inside and tangent to an ellipse, since only the ones that are tangent will have a constructive interference. The last operator $\mathbf{C}'_0\mathbf{H}'_h$, having two pull operators should have smoothest output. Sergey Fomel (personal communication) suggests an interesting illustration of it: Its adjoint is two push operators, $\mathbf{R}' = \mathbf{H}_h \mathbf{C}_0$. $\mathbf{R}'$ takes us from zero offset to nonzero offset first by pushing a data point to a semicircle and then by pushing points on the semicircle to flat-topped hyperbolas. As before, to make the hyperbolas more distinct, I broke the circle into dots along the circle and show the result in Figure 2.

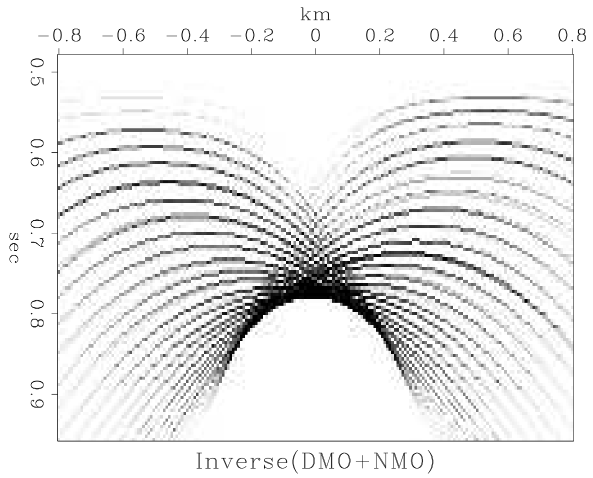

Figure 2: The adjoint of Rocca's smile is a superposition of flattened hyperbolas, each with its top on a circle. $\boxed{\text{jon2-sergey}}$ [ER]

It is worth noticing that the concepts in this section are not limited to constant velocity but apply as well to $v(z)$. The circle operator $\mathbf{C}$ presents some difficulties, however. The circle operator, a push from t to z, requires us to solve for z given t. Starting from the Dix moveout approximation, $t^2 = \tau^2 + x^2/v(\tau)^2$ we can directly solve for $t(\tau, x)$ but finding $\tau(t, x)$ is an iterative process at best. At worst, when the velocity gradient is abrupt enough, $\tau(t, x)$ is multivalued. We often see this multivalued function on raw data.

```
# Flat topped hyperbolas and constant-offset section migration
#
subroutine flathyp(   adj, add, vel    , h, t0,dt,dx, modl,nt,nx, data)
integer ix,iz,it,ib,  adj, add,                            nt,nx
real    t, amp, z,b,            vel(nt), h, t0,dt,dx, modl(nt,nx),data(nt,nx)
call adjnull(       adj, add,                         modl,nt*nx, data,nt*nx)
do ib= -nx, nx {        b = dx * ib             # b = midpt separation y-y0
    do iz= 2, nt {      z = t0 + dt * (iz-1)    # z = zero-offset time
        t = .5 * ( sqrt( z**2 +((b-h)*2/vel(iz))**2) +
                   sqrt( z**2 +((b+h)*2/vel(iz))**2)   )
        it = 1.5 + (t - t0) / dt
     if( it > nt )                     break
        amp = (z/t)/ sqrt(t)
        do ix= max0(1, 1-ib),  min0(nx, nx-ib)
                 if( adj == 0 )
                        data(it,ix+ib)= data(it,ix+ib) + modl(iz,ix   ) * amp
                 else
                        modl(iz,ix   )= modl(iz,ix   ) + data(it,ix+ib) * amp
        }
    }
return; end
```

ACKNOWLEDGMENTS

I would like to thank Christine Ecker and David Lumley for proofreading.

REFERENCES

Paige, C. C., and Saunders, M. A., 1982, Algorithm 583, LSQR: Sparse linear equations and least squares problems: Assn. Comp. Mach. Trans. Mathematical Software, **8**, 195–209.