Stanford Exploration Project, Report 75, November 17, 1997, pages 143–146

A Gazdag routine passing the dot-product test

Jon F. Claerbout[1]

keywords: *sources, diffraction*

ABSTRACT

We examine differential equations of the form $du/dz + i\alpha\, u = s(z)$ and find the pair of conjugate operators relating $s(z)$ and $u(z=0)$. This leads up to a Gazdag subroutine that passes the dot-product test.

INTRODUCTION

None of my tutorial literature develops Gazdag migration and modeling code as conjugate pairs. The closest thing to it is PVI pages 122-3 (Claerbout, 1992). Here I present a constructive proof of the conjugacy leading up to clean beginners code that passes the dot product test. (Passing the dot-product test means that the constructed conjugate really is conjugate (Claerbout, 1992).)

THE DIFFERENTIAL EQUATION

Many situations in physics are expressed by the differential equation

$$\frac{du}{dz} \;-\; i\alpha\, u \quad = \quad s(z) \tag{1}$$

In the migration application, $u(z)$ is the up-coming wave, $\alpha = \sqrt{\omega^2/v^2 - k_x^2}$, $s(z)$ is the "exploding-reflector" source, and $\tilde{s}(z)$ (summed over frequency) is the derived image. We take the medium to be layered (v constant in layers) so that α is constant in a layer, and we put the sources at the layer boundaries. Thus within a layer we have $du/dz - i\alpha\, u = 0$ which has the solution

$$u(z_k + \Delta z) \quad = \quad u(z_k)\; e^{i\alpha\Delta z} \tag{2}$$

For convenience, I introduce the "delay" operator in the k-th layer $Z_k = e^{-i\alpha\Delta z}$ so the delay of upward propagation is expressed by $u(z_k) = Z_k\, u(z_k + \Delta z)$. Besides crossing

[1]**email:** not available

layers, we must cross layer boundaries where the (reflection) sources add in to the upcoming wave. Thus we have

$$u_{k-1} \quad = \quad Z_k u_k + s_{k-1} \tag{3}$$

Recursive use of equation (3) across a medium of three layers is expressed in matrix form as

$$\mathbf{M}\,\mathbf{u} \quad = \quad \begin{bmatrix} 1 & -Z_0 & . & . \\ . & 1 & -Z_1 & . \\ . & . & 1 & -Z_2 \\ . & . & . & 1 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{bmatrix} \quad = \quad \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \quad = \quad \mathbf{s} \tag{4}$$

A recursive solution begins at the bottom with $u_3 = s_3$ and propagates upward.

The conjugate of the delay operator Z is the time advance operator $\bar{Z}$. The conjugate operation to that in equation (4) is given by

$$\mathbf{M}'\,\tilde{\mathbf{s}} \quad = \quad \begin{bmatrix} 1 & . & . & . \\ -\bar{Z}_0 & 1 & . & . \\ . & -\bar{Z}_1 & 1 & . \\ . & . & -\bar{Z}_2 & 1 \end{bmatrix} \begin{bmatrix} \tilde{s}_0 \\ \tilde{s}_1 \\ \tilde{s}_2 \\ \tilde{s}_3 \end{bmatrix} \quad = \quad \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \end{bmatrix} \quad = \quad \mathbf{u} \tag{5}$$

The conjugacy of equation (4) and (5) seems obvious, but it is not quite the elementary form we are so familiar with because the matrix multiplies the output (instead of the customary matrix multiplying the input). To prove the conjugacy, notice that equation (4) is equivalent to $\mathbf{u} = \mathbf{M}^{-1}\mathbf{s}$ whose conjugate, by definition, is $\tilde{\mathbf{s}} = (\mathbf{M}^{-1})'\mathbf{u}$ which is $\tilde{\mathbf{s}} = (\mathbf{M}')^{-1}\mathbf{u}$ (because of the basic mathematical fact that the conjugate of an inverse is the inverse of the conjugate) which gives $\mathbf{M}'\tilde{\mathbf{s}} = \mathbf{u}$ which is equation (5).

We observe the wavefield only on the surface $z = 0$, so the conjugacy of equations (4) and (5) is academic because it relates the wavefield at all depths with the source at all depths. Thus we need to truncate $\mathbf{u}$ to its first coefficient u_0. That will change the conjugate relation. Remember that the conjugate to truncation is zero padding. Thus the conjugate to solving equation (4) and then abandoning the (unobservable) waves beneath the surface is the solution to

$$\mathbf{M}'\,\tilde{\mathbf{s}} \quad = \quad \begin{bmatrix} 1 & . & . & . \\ -\bar{Z}_0 & 1 & . & . \\ . & -\bar{Z}_1 & 1 & . \\ . & . & -\bar{Z}_2 & 1 \end{bmatrix} \begin{bmatrix} \tilde{s}_0 \\ \tilde{s}_1 \\ \tilde{s}_2 \\ \tilde{s}_3 \end{bmatrix} \quad = \quad \begin{bmatrix} u_0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6}$$

which amounts to a recursion beginning from $\tilde{s}_0 = u_0$ and continuing downward with

$$\tilde{s}_k \quad = \quad \bar{Z}_{k-1}\,\tilde{s}_{k-1} \tag{7}$$

A final feature of the migration application is that the image is formed from $\tilde{\mathbf{s}}$ by summing over all frequencies. The proposed subroutine is

```
# Phase shift modeling and migration.   (Warning: destroys its input!)
#
subroutine gazconj( conj, dt,dx, v,nt,nx, modl,         data )
integer             conj,         nt,nx,                          iw, ikx, iz,nz
complex iktau,                            modl(nt,nx),  data(nt,nx)
real                      dt,dx, v(nt),                 pi, w,w0,dw, kx,kx0,dkx
temporary complex               cs(nt)
call conjnull(     conj,  0,              modl,2*nt*nx,  data,2*nt*nx )
pi = 4.*atan(1.);       w0  = -pi/dt;   dw = 2.*pi/(nt*dt)
nz = nt;                kx0 = -pi/dx;   dkx= 2.*pi/(nx*dx)
if( conj == 0)  call ft2axis( 0, -1., nz, nx, modl)
else    {       call ft2axis( 0, -1., nt, nx, data)
                call ft1axis( 0,  1., nt, nx, data)
                }
do ikx  = 2, nx {       kx = kx0 + (ikx-1) * dkx
do iw   = 2, nt {       w  = w0  + (iw -1) * dw
        if( conj== 0) { data(iw,ikx) = modl(nz,ikx)
                do iz = nz-1, 1, -1
                        data(iw,ikx) = data(iw,ikx) * iktau(dt,w,v(iz)*kx,dw) +
                                       modl(iz,ikx)
                }
        else {          cs(   1) =     data(iw,ikx)
                do iz = 1, nz-1
                        cs(iz+1) = cs(iz) *   conjg(  iktau(dt,w,v(iz)*kx,dw) )
                do iz = 1, nz
                        modl(iz,ikx) = modl(iz,ikx)  + cs(iz)
                }
        }}
if( conj == 0)  { call ft1axis( 1,  1., nt, nx, data)
                  call ft2axis( 1, -1., nt, nx, data) }
else            { call ft2axis( 1, -1., nz, nx, modl) }
return; end

complex function iktau( dt, w, vkx, nyep)
real                    dt, w, vkx, nyep        # nyep = nyquist * epsilon
iktau = cexp( - dt * csqrt( cmplx( nyep, w) ** 2   +   vkx * vkx /4. ) )
return; end
```

REFERENCES

Claerbout, J. F., 1992, Earth Soundings Analysis: Processing Versus Inversion: Blackwell Scientific Publications.