# A PROGRAM FOR STABLE MIGRATION

*Bob Godfrey and Bert Jacobs*

Migration in a laterally varying medium with assured stability is discussed in another paper in this report, "Bullet-Proofing the Code for the 45-Degree Equation" (Jacobs, Godfrey and Claerbout). Equation (9) of that paper,

$$\frac{d}{dz} q = - \left\{ i\omega(\Lambda - X) + V^{-1/2} \left[ \frac{VD_x D_x^t V}{2i\omega I + \frac{VD_x D_x^t V}{2i\omega}} \right] V^{-1/2} \right\} q$$

$$D_x D_x^t = \frac{T}{\Delta x^2 (I - \gamma T)} \qquad (1)$$

$$\frac{1}{-i\omega + \varepsilon} = \text{causal integration}$$

was found to conserve $q^*q$ as a function of depth. All symbols are as before: $T$ is a real, symmetric, tridiagonal, second-differencing matrix; $V$ is a real, diagonal velocity matrix; $\Lambda = V^{-1}$; and $S$ takes care of retardation and splitting. The $VD_x D_x^t V$ in (1) were implemented because the discrete approximation to $D_x D_x^t$ is well understood. The alternative is to use $D_x^t V^2 D_x$ in place of $VD_x D_x^t V$.

If dip filtering is added to Equation (1), we get

$$\frac{d}{dz} q = - \left\{ (i\omega + \varepsilon_0)(\Lambda - S) + V^{-1/2} \left[ \frac{VD_x D_x^t V}{2(i\omega + \varepsilon_1) I + \frac{VD_x D_x^t V}{2(i\omega + \varepsilon_2)}} \right] V^{-1/2} \right\} q \qquad (2)$$

where the signs of the $\varepsilon$'s are chosen for the migration of upcoming waves. The discrete approximation to (2) is in Equation (11) of "Bullet-Proofing ..." We choose to multiply through the operators by $(i\omega + \varepsilon_2)$ and avoid complex divisions. This leads to

$$[D_3 + TD_4]q' = [D_1 + TD_2]q(z)$$

$$D_1(\omega,\Delta z) = (i\omega + \varepsilon_2)(i\omega + \varepsilon_1)\Lambda^{1/2}[I - (i\omega + \varepsilon_0)\frac{\Delta z}{2}(\Lambda - S)]$$

$$D_2(\omega,\Delta z) = \left[\frac{1}{4\Delta x^2}V^2 - \gamma(i\omega + \varepsilon_2)(i\omega + \varepsilon_1)I\right]\Lambda^{1/2}$$
$$\left[I - (i\omega + \varepsilon_0)\frac{\Delta z}{2}(\Lambda - S)\right] - \frac{\Delta z}{4\Delta x^2}(i\omega + \varepsilon_2)V^{1/2}$$

$$D_3(\omega,\Delta z) = D_1(\omega,-\Delta z)$$

$$D_4(\omega,\Delta z) = D_2(\omega,-\Delta z)$$

*Fourier transform sign conventions*

If the Fourier transform sign convention is the opposite of that in Equation (1), then all we have to do is change the sign of $\omega$ in our discrete operators. This conclusion is a consequence of the Hermitian property of the transforms of real series.

It is easy to demonstrate this result. Let R denote the quantity in braces in Equation (2), Q(t) stand for the input data, and $s(\omega) = q(-\omega)$. We get the migration equation

$$s(\omega) = q(-\omega) = \int_0^\infty dt\; Q(t)\; e^{-i\omega t}$$

$$\frac{d}{dz}s(\omega) = \frac{d}{dz}q(-\omega) = -R(-i\omega + \varepsilon)\; q(-\omega)$$

$$= -R(-i\omega + \varepsilon)\; s(\omega) \tag{4}$$

and imaging conditions

$$Q(0) = 2 \text{ Re} \int_0^\infty q(\omega)\, d\omega = 2 \text{ Re} \int_0^\infty s(-\omega)\, d\omega$$

$$= 2 \text{ Re} \int_0^\infty s^*(\omega)\, d\omega$$

$$= 2 \text{ Re} \int_0^\infty s(\omega)\, d\omega \tag{5}$$

In other words, we get the same migration by imaging $s(\omega)$ with (4) that we get by imaging $q(\omega)$ with (2).

*Examples*

Figures 1-4 show the result of migrating a time section consisting of an impulse located at $x = 32\Delta x$, $t = 32\Delta t$. The grid size was $(64 \times 64)$ and a total of 64 $\Delta z$-steps were taken. We note that one-half the energy of the input time section was evanescent; hence, dip filtering is especially important. In fact, a blob of energy, corresponding to low $\omega$ evanescent energy, persists in Figure 1, which is a constant velocity migration $(v = 1)$ with $\alpha = 0.25$. In Figure 2, a vertical velocity boundary was placed at $x = 32\Delta x$. To the right of the boundary, $v = 1.0$ and to the left $v = 1.5$. The refraction leaves the boundary at $z = 48\Delta z$, as predicted by ray theory. Moving the velocity boundary to $x = 22\Delta x$ gave the result in Figure 3. Again, ray theory indicates that both reflection and refraction are correctly positioned. Figure 4 is the same as Figure 3, plotted at one-half the clip level.

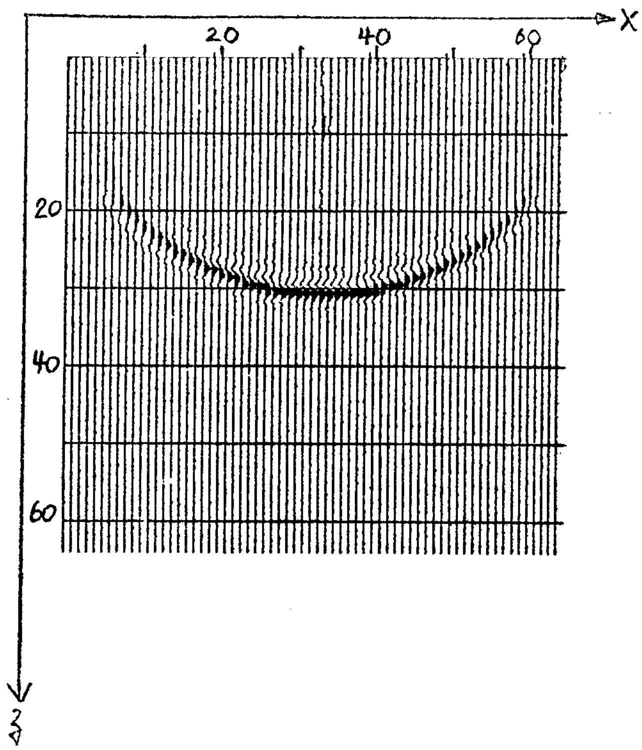


FIGURE 1.--Migrated depth section. Input was an impulse at $x = 32\Delta x$, $t = 32\Delta t$ and 64 $\Delta z$-steps were taken. Dip filtering applied, constant velocity (=1.).

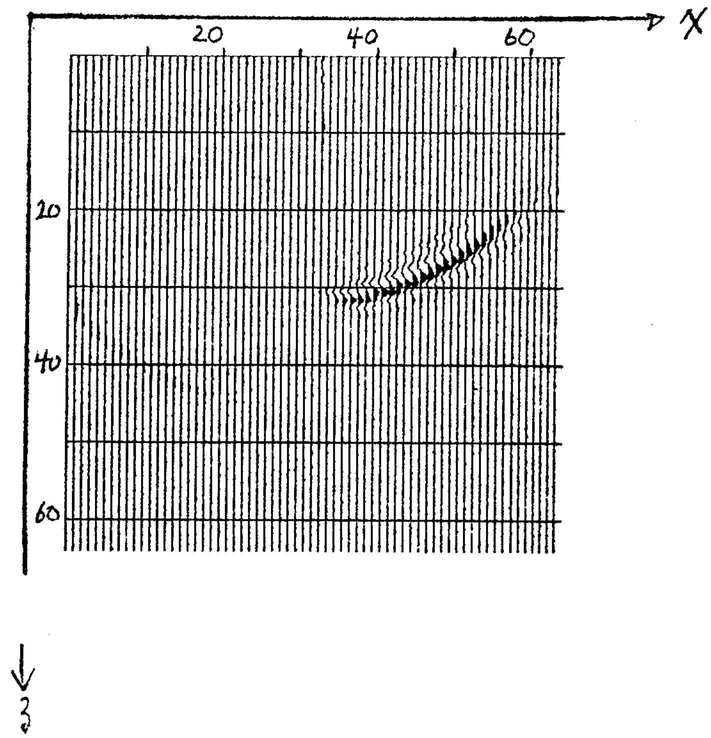


FIGURE 2.--Vertical velocity variation: $x < 32\Delta x$, $v = 1.5$; $x \geq 32\Delta x$, $v = 1.0$. The refraction ($x < 32\Delta x$) is correctly positioned but hardly visible.

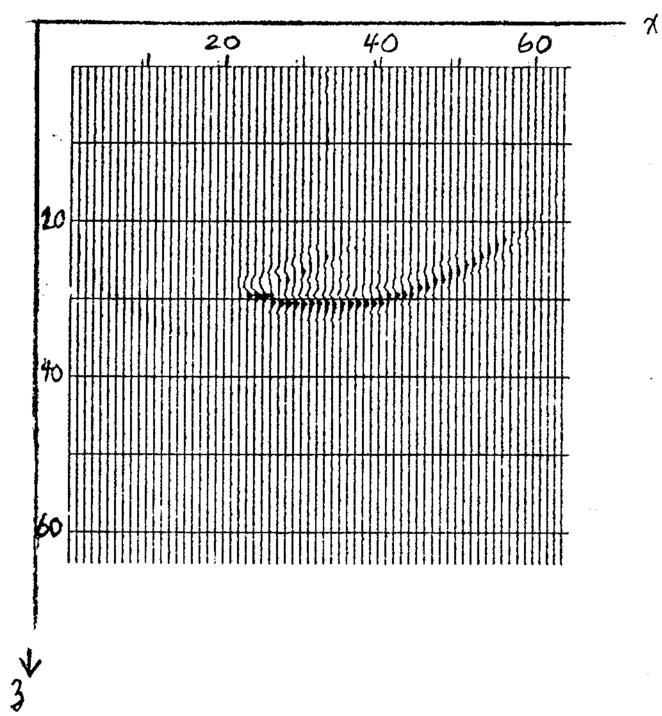


FIGURE 3.--Vertical velocity variation: $x < 22\Delta x$, $v = 1.5$; $x \geq 23\Delta x$, $v = 1.0$. Both refraction and reflection are correctly positioned with reflection having higher amplitude.

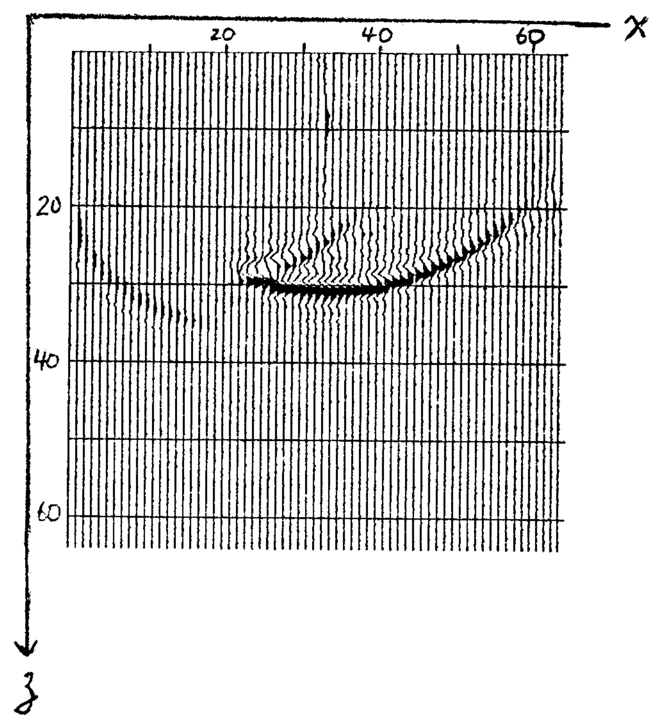


FIGURE 4.--Same as Figure 3 except plotted at a clip level = 1/2 maximum value. (All other plots are clipped at the maximum value of the plot.)

```
c       Stable 45 degree migration.
c       When the positive real paremeter eps=0, the algorithm
c       conserves p(*)p, where p(*)=conjugate transpose.
c       Otherwise, algorithm is dissipative (neglecting round-off).
c       Sign convention:
c               F.T. defined by: P(x,w,z)=sum[P(x,t,z)exp(+iwt)]
c               eps>0
c               dp/dz=function(-iw+eps)
c
c       Input wavefield: P(x,w,z=0)
c       Output wavefield: P(x,t=0,z)
c
        complex cmplx,cexp,carg,cpsh(64)
        complex conjg
        complex ciad,cciad,csc1,csc2,c1,c2
        complex cp(64),cd(64),cta(64),ctb(64),ctc(64),cd1(64),
     *           cd3(64),cd2(64),cd4(64),ca1(64),cb1(64),
     *           cc1(64),ca2(64),cb2(64),cc2(64),cwrap(64)
        complex cep(64),cf(64)
        real v(64),slowp(64),rmig(64)
c       Define file system:
c               pf:P(x,w,z=0)
c               sc:P(x,w,z>0), each old z-level is destroyed
c                       as new z-level is produced.
c               mg:P(x,t=0,z)
        logical*1 pf(50),filsc(50),filmg(50)
        data pf/'pf'/
        data filsc/'sc'/
        data filmg/'mg'/
c       Read-in field and processing parameters
        call readin(nx,nt,nw,ist,nz,dx,dt,vmin,samp,
     *    idip,dip)
        write(6,30)nx,nt,nw,ist,nz,idip,dx,dt,vmin,samp,dip
30      format('nx,nt,nw,ist,nz,idip=',6(i5,1x),/,
     *   'dx,dt,vmin,samp,dip=',5(e11.4,1x))
31      format('z level=',i5)
c
c       Explanation of processing parameters
c
c       nx=number of x-grid points
c       nt=number of t-grid points
c       nw=number of frequencies to sum
c       ist=start summing frequency at ist
c       nz=number of dz-steps
c       dx=grid interval in x
c       dt=sampling interval in t
c       vmin=lowest velocity in model
c       samp=degree of over-sampling in t .ie. w(max)=w(nq)/samp
c       idip=1==>dip filtering
c       dip=dip filtering coefficient .ie. eps2=dip*w
c
c       Open and create the file systems.
c       The function icreat(filename,6*64+4*8+4) creates a file
c       so that other users can't write on it.  Function
c       iclose(file) closes the file.  Function iopen(file,2) opens
c       the file so that the user can both read & write on it.
```

```
c       Subroutine setfil(11,file,512) assigns the number 11 to the
c       file for use in unformatted write statements.
c
        call setfil(11,'/scr/bob/vel',512)
        ifilmg=icreat(filmg,6*64+4*8+4)
        ncl=iclose(ifilmg)
        nop=iopen(filmg,2)
        ifilsc=icreat(filsc,6*64+4*8+4)
        ncl1=iclose(ifilsc)
        nop1=iopen(filsc,2)
        ifilcp=iopen(pf,2)
c       Define constant table:
c       (1) gam:-Dxx=(I-gam*T)'T where T=(...-1,2,-1,...)
c               where A'=inverse of A
c       (2) alf,beta:(1+x)**1/2=(1+alfp*x)/(1.+betap*x)
c               alf=alfp-betap
c               beta=betap
c
        dx2=dx*dx
        gam=1./6.
        alf=0.5
        beta=+.250
c       Additional constants:
        pi=4.*atan(1.)
        rnw=float(ist+nw)
        dw=pi/(rnw*dt)
        nx1=nx-1
c       Set parameters for Test Velocity Model
        nxv=(nx/2)+10
        nxv1=nxv+1
        vel1=1.0
        vel2=1.5
c       Form 3 vectors of t matrix:
c               vector a located below diag vector b
c                      c         above              b
c
        cta(1)=(0.,0.)
        ctb(1)=(2.,0.)
        ctc(1)=(-1.,0.)
        do 100 ix=2,nx1
        cta(ix)=(-1.,0.)
        ctb(ix)=(2.,0.)
100     ctc(ix)=(-1.,0.)
        cta(nx)=(-1.,0.)
        ctb(nx)=(2.,0.)
        ctc(nx)=(0.,0.)
c       Copy nw frequencies (starting from ist) from input file(pf)
c       to internal file(sc)
        if(ist.eq.0)go to 107
c
c       Function iread(file,array,nbytes) reads nbytes from the
c       named file and packs them into array. Integer=2 bytes,
c       Real=4 bytes and Complex=8 bytes.
c
        do 105 iw=1,ist
        nread=iread(ifilcp,cp,8*nx)
```

```
105     continue
c
c       Function iwrite(file,array,nbytes) writes nbytes from the
c       array onto the named file.
c
107     do 110 iw=1,nw
        nread=iread(ifilcp,cp,8*nx)
        do 108 ix=1,nx
108     cp(ix)=conjg(cp(ix))
110     nwrite=iwrite(ifilsc,cp,8*nx)
c
c       Downward continue through nz levels
c
        do 1000 iz=1,nz
        write(6,31)iz
c       Rewind "sc" file system
c
c       Function iseek(file,offset,ptrname) moves the file pointer
c       to offset (in bytes) if ptrname=0. If ptrname=1, the pointer
c       is set to it's current location plus offset.
c
        nseek=iseek(ifilsc,0,0)
c       Set up velocity field either from disk or internally
c       read(11)(v(i),i=1,nx)
        do 120 ix=1,nxv
120     v(ix)=vel1
        do 140 ix=nxv1,nx
140     v(ix)=vel2
c       Set up the retardation field. Three choices:
c               (1)=constant retardation
c               (2)=smoothed version of 1./v(x)
c               (3)=1./v(x)
c       Choice (1) gives exact split. Last choices are O(dz**2)
        do 160 ix=1,nx
160     slowp(ix)=0.88
c       Choose dz so that kz(max)=vmin/w(max)
        dz=vmin*dt*samp
        a=dz/2.
c       Zero out migration accumulator
        call zero(nx,rmig)
c       Loop over nw frequencies
        do 800 iw=1,nw
        nread=iread(ifilsc,cp,8*nx)
        w=+float(iw+ist-1)*pi/((rnw-1.)*dt)
c       Set dip-filtering constants
        eps1=0.
        eps2=0.
        if(idip.eq.1)eps2=dip*w
        c1=cmplx(eps2,w)
        c2=cmplx(eps1,w)*c1
c       Compute exponential shifts.
        do 450 ix=1,nx
        aip=-w*dz*slowp(ix)
        carg=cmplx(0.,aip)
450     cpsh(ix)=cexp(carg)
c       Subtract the wrap-around field from P and shift.
```

```
c       When iz=1,the wrap-around field is identically zero
c       and has no effect on the wavefield p.
        do 500 ix=1,nx
        cp(ix)=cp(ix)-cwrap(ix)
500     cp(ix)=cp(ix)*cpsh(ix)
c       Form diagonal matrices d1,d2,d3,d4 as in Equation 3
        do 550 ix=1,nx
        v2=v(ix)*v(ix)
        vh=sqrt(v(ix))
        d=(1./v(ix))-slowp(ix)
        wad=w*a*d
        ciad=cmplx(1.,-wad)
        cciad=cmplx(1.,wad)
        cd1(ix)=c2*ciad/vh
        cd3(ix)=c2*cciad/vh
        csc1=(-c2*gam + (beta*v2/dx2))/vh
        csc2=a*alf*c1*vh/dx2
        cd2(ix)=(csc1*ciad)-csc2
        cd4(ix)=(csc1*cciad)+csc2
550     continue
c       Form tri-diag matrices t1=d1+t*d2
c                              t2=d3+t*d4
        call tee(cd1,cd2,cta,ctb,ctc,ca1,cb1,cc1,nx)
        call tee(cd3,cd4,cta,ctb,ctc,ca2,cb2,cc2,nx)
c       Form d=t1*p
        call teep(ca1,cb1,cc1,cp,cd,nx)
c       Form d=t2'*d
        call tricv(cc2,cb2,ca2,nx,cd,cd,cep,cf)
        fact=2.
c       Accumulate migrated field
c       P(x,t=0,z)=sum[P(x,w,z)]
c       The sum doubles real parts and zeroes imag. parts
c       The DC is added just once.
        if((iw+ist).eq.1)fact=1.
        do 600 ix=1,nx
600     rmig(ix)=rmig(ix)+fact*real(cd(ix))
c       Write downward continued field to internal file(sc)
c       after rewinding to the iz-level.
        iset=-8*nx
        nseek=iseek(ifilsc,iset,1)
        nwrite=iwrite(ifilsc,cd,8*nx)
800     continue
c       Write out migrated field
        nwrite=iwrite(ifilmg,rmig,4*nx)
c       Form wrap-around field
c       P(x,t=0,z) must be subtracted from P(x,t,z)
c                         or
c       FT[P(x,t=0,z)] must be subtracted from FT[P(x,t,z)]
c
        do 900 ix=1,nx
        rp=rmig(ix)/float(nt)
900     cwrap(ix)=cmplx(rp,0.)
1000    continue
        stop
        end
c
```

```
c
c
c
      subroutine tee(cx,cy,cta,ctb,ctc,ca,cb,cc,nx)
      complex cx(1),cy(1),cta(1),ctb(1),ctc(1),ca(1),cb(1),cc(1)
      nx1=nx-1
      do 100 ix=1,nx1
      ca(ix+1)=cta(ix+1)*cy(ix)
      cb(ix)=cx(ix)+ctb(ix)*cy(ix)
100   cc(ix)=ctc(ix)*cy(ix+1)
      cb(nx)=cx(nx)+ctb(nx)*cy(nx)
      return
      end
c
c
      subroutine teep(ca,cb,cc,cx,cy,nx)
      complex cx(1),cy(1),ca(1),cb(1),cc(1)
      nx1=nx-1
      cy(1)=cb(1)*cx(1)+cc(1)*cx(2)
      do 100 ix=2,nx1
100   cy(ix)=ca(ix)*cx(ix-1)+cb(ix)*cx(ix)+cc(ix)*cx(ix+1)
      cy(nx)=ca(nx)*cx(nx1)+cb(nx)*cx(nx)
      return
      end
c
c
      subroutine tricv(a,b,c,n,t,d,e,f)
      implicit complex ( a-h,o-z)
      dimension t(n),d(n),f(n),e(n),a(n),b(n),c(n)
      n1 = n-1
      e(1) = -a(1)/b(1)
      f(1) = d(1)/b(1)
      do 10 i = 2,n1
      den = b(i)+c(i)*e(i-1)
      e(i) = -a(i)/den
10    f(i) = (d(i) - c(i)*f(i-1))/den
      t(n) = (d(n)- c(n)*f(n1))/(b(n)+c(n)*e(n1))
      do 20 j = 1,n1
      i = n-j
20    t(i) = e(i) *t(i+1) + f(i)
      return
      end
      subroutine readin(nx,nt,nw,ist,nz,dx,dt,vmin,samp,
     *   idip,dip)
      call getint(2,nx)
      call getint(3,nt)
      call getint(4,nw)
      call getint(5,ist)
      call getint(6,nz)
      call getrel(7,dx)
      call getrel(8,dt)
      call getrel(9,vmin)
      call getrel(10,samp)
      call getint(11,idip)
      call getrel(12,dip)
      return
```

```
        end
        subroutine getint(ipos,int)
        logical*1 arg(8)
        call getarg(ipos,arg,8)
        decode(8,101,arg)int
101     format(i8)
        return
        end
        subroutine getrel(ipos,rel)
        logical*1 arg(8)
        call getarg(ipos,arg,8)
        decode(8,101,arg)rel
101     format(f8.3)
        return
        end
        subroutine zero(n,x)
        dimension x(n)
        do 10 i=1,n
10      x(i)=0.
        return
        end
```