물리탐사
Vol. 1, No. 2, 1998, p. 116~126

Datuming by Wavefield Depth Extrapolation

Jun, Ji[1)]

파동장 외삽을 이용한 데이터밍

지 준

Abstract : I present a datuming scheme for poststack data that uses wavefield depth extrapolation. The method I have developed allows the use of any depth extrapolation technique, such as phase-shift, split-step, and finite-difference extrapolation. I derive the datuming algorithms by transposing and taking the complex conjugate (i.e. taking adjoint) of the corresponding forward modeling operator, which does upward extrapolation from a flat surface to an irregular surface. The exact adjoint relation between the forward modeling operator and the datuming operator is demonstrated algebraically. Testing the poststack datuming algorithms with synthetic data, using several depth extrapolation algorithms, has shown that the method works well.

요 약 : 본 논문은 파동장의 심도방향으로의 외삽(extrapolation) 을 사용한 데이터밍 기법을 소개한다. 개발된 기법은 phase-shift, split-step, 또는 유한차분과 같은 다양한 파동장 외삽기법들을 사용할 수 있다. 데이터밍 알고리즘을 유도하기 위해, 우선 평면에 정의 되어 있는 파동장을 임의의 굴곡을 갖는 면으로 외삽을 수행하는 모델링 연산자를 대수학적으로 구한 후, 본 모델링 연산자에 어드조인트(adjoint) 관계에 있는 연산자를 대수학적으로 구하여 데이터밍 연산자를 얻었다. 다양한 외삽방법을 사용한 데이터밍 알고리즘의 실험에서 매우 만족스러운 결과를 얻을 수 있었다.

Seismic data acquired in areas with irregular topography introduce a numerical problem for migration algorithms that are based on depth extrapolation. Since numerically efficient migration schemes are usually based on depth extrapolation algorithms which allow the extrapolation of a wavefield from a flat surface to another flat surface, datuming is required prior to migration. Datuming is a method of processing to extrapolate a known wavefield at a specified datum of arbitrary shape to another specified datum, also of arbitrary shape. For small differences in elevation and slow velocity variations between the input datum and the output datum, static shifting is a sufficiently accurate datum correction procedure. However, for significant differences in elevations and a more complicated velocity model, the accuracy of the static solution may prove to be insufficient; and a more exact method should be used.

Berryhill (1979, 1984) presented a wave equation datuming scheme for poststack and prestack data using the Kirchhoff integral method. Following his work, various forms of the Kirchhoff integral solution to the wave equation have been used by different authors for migration (Wiggins, 1984; Shtivelman and Canning, 1988) and layer replacement (Yilmaz and Lucas, 1986; Berryhill, 1986). The Kirchhoff approach is very expensive and cannot be used for a variable velocity medium.

An elegant and simple technique to correct for the error caused by the static time shift was introduced by the "zero-velocity layer" concept (Beasley and Lynn, 1989). Not only is the static shift required before the migration, but this technique cannot even be applied to the computationally attractive phase-shift algorithms (Gazdag, 1978), because it includes the nonphysical characteristic of zero velocity.

This paper describes a datuming algorithm which can uses any depth extraoplation technique. In the following section, the datum algorithm is explained in algebraic representation using various depth extrapolation schemes such as the phase-shift (Gazdag, 1978), split-step (Stoffa *et al.*, 1990), and finite-difference methods (Claerbout, 1984).

Forward Operator and its Adjoint

In this paper, the basic strategy for finding a datuming operator is first formulating a forward modeling operator and then deriving the corresponding datuming operator by transposing and taking the complex conjugate (i.e. taking

*1998년 8월 3일 접수
1) 한성대학교 정보공학부(Department of Information Eng., Hansung University)

adjoint) of the forward modeling operator. For poststack or zero offset sections, the forward modeling is an algorithm that extrapolates the wavefield upward from a flat surface to an irregular surface by the one-way wave equation using half the velocity of the medium. The modeling algorithm is derived so that it can be used with any depth extrapolation technique, including the phase-shift, split-step Fourier, and finite-difference methods.

Forward modeling operator

In forward modeling, the wavefield recorded at each geophone along an irregular surface is the wavefield propagated up to the depth level where the geophones are located. It is necessary to stop the wavefield propagation after recording because the reflection coefficient at surface is almost -1 (Ji and Claerbout, 1992). To do so, I formulate the forward modeling operator by propagating the wavefield upward with a filter between extrapolation steps to stop the wavefield propagation where it is recorded. Then all wavefields from all depth levels where the geophones are located are summed together to produce the wavefield along the irregular surface. In order to explain the algorithm clearly and schematically, I use a simple topography model that has only eight geophone groups on an irregular surface, as illustrated in Fig. 1.

Fig. 2 schematically describes the forward modeling algorithm for the simple model. W_i represents upward extrapolation at the i-th depth level and F_1, F_2, and F_3 are spatial filters for grabbing the wavefield where the geophones are located at the corresponding depth levels. The operators I-F_3 and I-F_2-F_3 in Fig. 2 stop the wavefield at the locations where it is recorded below or at the corresponding depth level and pass the wavefield at the locations where it is not yet recorded. Each small rectangle in Fig. 2 represents an abstract vector that contains wavefields at the corresponding space location. The wavefield along the irregular surface is obtained by summing the wavefields that are grabbed at the various depth levels.

The forward modeling scheme shown in Fig. 2 can be algebraically generalized, if we divide the topography into

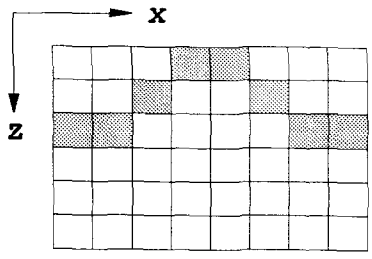

Fig. 1. Synthetic surface recording geometry. Solid squares represent geophone location on an undulating surface.

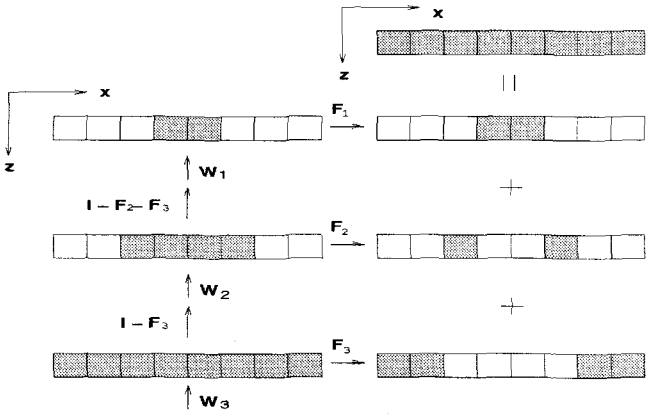

Fig. 2. Forward modeling scheme: the schematic diagram for forward depth extrapolation when the surface is not flat. W_i represents the upward extrapolation operator at the i-th depth level. F_1, F_2, and F_3 are spatial filters shown in the text, and I is the identity matrix.

z levels, as follows:

$$[d_0] = [1\ 0]\, G_1 E_1 G_2 E_2 \cdots G_{z-1} E_{z-1} G_z \begin{bmatrix} 0 \\ 1 \end{bmatrix} [d_z] \tag{1}$$

$$E_i = \begin{bmatrix} 1 & 0 \\ 0 & W_i \end{bmatrix} \tag{2}$$

$$G_i = \begin{bmatrix} 1 & F_i \\ 0 & K_i \end{bmatrix} \tag{3}$$

where

$$K_i = I - \sum_{j=1}^{z-i} F_{z-j}$$

In Eq. (1), d_0 and d_z are wavefields on the irregular surface and the datum level, respectively. The extrapolation operator E is followed by the spatial filter G at every depth level. We can see that the upward extrapolation operator W_i is applied to the wavefield that does not arrived at the surface because the operator K_{i-1} remove the wavefield if it has arrived at any previous depth level. All wavefields that arrive at the surface are saved by the operator F_{i-1} for the final output. For the simple geometry shown in Fig. 1, F_1, F_2, and F_3 are just diagonal matrices whose elements are 1 where the geophones are located and 0 elsewhere. Thus their diagonal elements are as follows:

diag(F_1) = [0 0 0 1 1 0 0 0]
diag(F_2) = [0 0 1 0 0 1 0 0]
diag(F_3) = [1 1 0 0 0 0 1 1]

The operator W_i in Eq. (2) can be any extrapolation

scheme including the Kirchhoff, phase-shift, split-step, or finite-difference method. If we use the phase-shift extrapolation algorithm for W_i, we need an additional inverse Fourier transform in every extrapolation step because the operator G is in the space domain. However, all other algorithms, such as the Kirchhoff, split-step, and finite-difference methods, don't need any additional computation except the operation by G, which is the multiplication of the extrapolated wavefield by the zero/one filter.

Datuming operator

Now the datuming operator can be easily found by transposing and taking the complex conjugate of each matrix in the forward modeling operator shown in Eq. (1). The datuming operator for a poststack data set gathered on an irregular surface is thus

$$[d_z] = [0\ 1]\, G_z^T E_{z-1}^T G_{z-1}^T \cdots E_2^T G_2^T E_1^T G_1^T \begin{bmatrix} 1 \\ 0 \end{bmatrix} [d_0] \tag{4}$$

$$E_i^T = \begin{bmatrix} 1 & 0 \\ 0 & W_i^T \end{bmatrix} \tag{5}$$

$$G_i^T = \begin{bmatrix} 1 & 0 \\ F_i & K_i \end{bmatrix} \tag{6}$$

In Eq. (4), we can see that the downward extrapolation E^T is preceded by the filter G^T at every depth level. We then apply the downward extrapolation operator W_i^T to the wavefield that was introduced by the operator F_i up to a given depth level. The portion of the wavefield that is not introduced until a given depth level is removed by the operator K_i. Fig. 3 shows this datuming operator when the topography is given by Fig. 1.

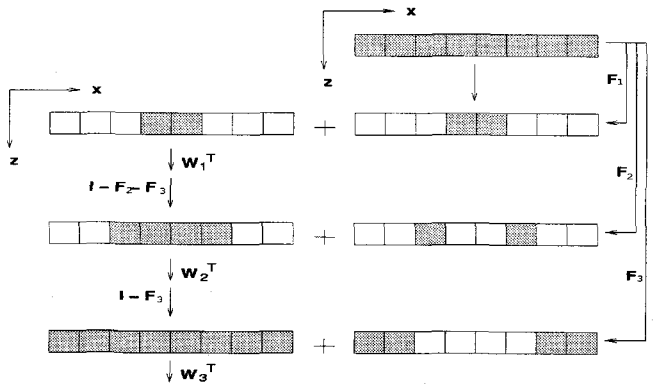

Fig. 3. Datuming scheme: schematic diagram for the datuming operator as the adjoint to the forward extrapolation scheme, when the surfaces are irregular. W_1^T represents the downward propagation operator at each depth level.

Examples

Synthetic examples

I test this datuming scheme with phase-shift, split-step, and finite-difference depth extrapolation algorithms. The codes used for the datuming are checked the adjointness with dot-product test (Claerbout, 1992) and listed in the Appendix at the end of this paper.

For testing, I use a simple model, Fig. 4, that has a syncline reflector under an undulating surface. The irregular surface is modeled to have not only gradual topographic changes but also a discontinuity.

The forward modeling experiment was done using the algorithm explained in Fig. 2 for a constant velocity; Fig. 5(a) shows the result. Datuming was then performed using the algorithm shown in Fig. 3 with the phase-shift extrapolation as the depth extrapolation operator W. The result appears in Fig. 5(b); the exact bow-tie shaped wavefield is the characteristic of the syncline reflector on a flat datum. I then applied the same algorithm with the other depth extrapolation schemes. Fig. 5(c) and (d) show the datumed results for the split-step and the 45-degree finite-difference methods, respectively. When the velocity is constant, the split-step algorithm is identical to the phase shift algorithm. Therefore we can see that the datumed wavefields in Fig. 5(b) and (c) are the same. The result of the finite-difference method, Fig. 5(d), also shows a correctly located bow-tie shaped wavefield except very weak artifacts in the region under the undulating surface. These artifacts can be explained as the energy from the evanescent region which has not removed.

To illustrate the effectiveness of the datuming algorithm, a velocity function which varies in depth and lateral extent is tested for the same reflector and topographic model shown in Fig. 4. The velocity model used in this

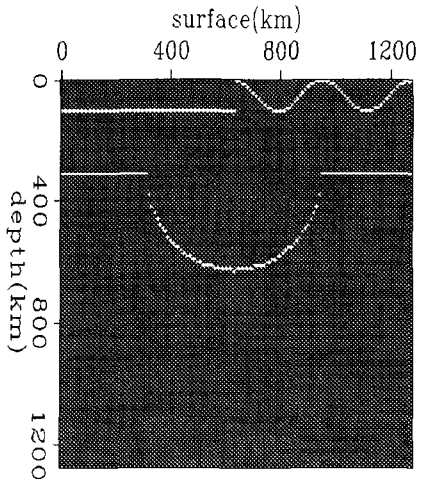

Fig. 4. Synthetic model with a syncline reflector image (lower) under an undulating surface (upper).

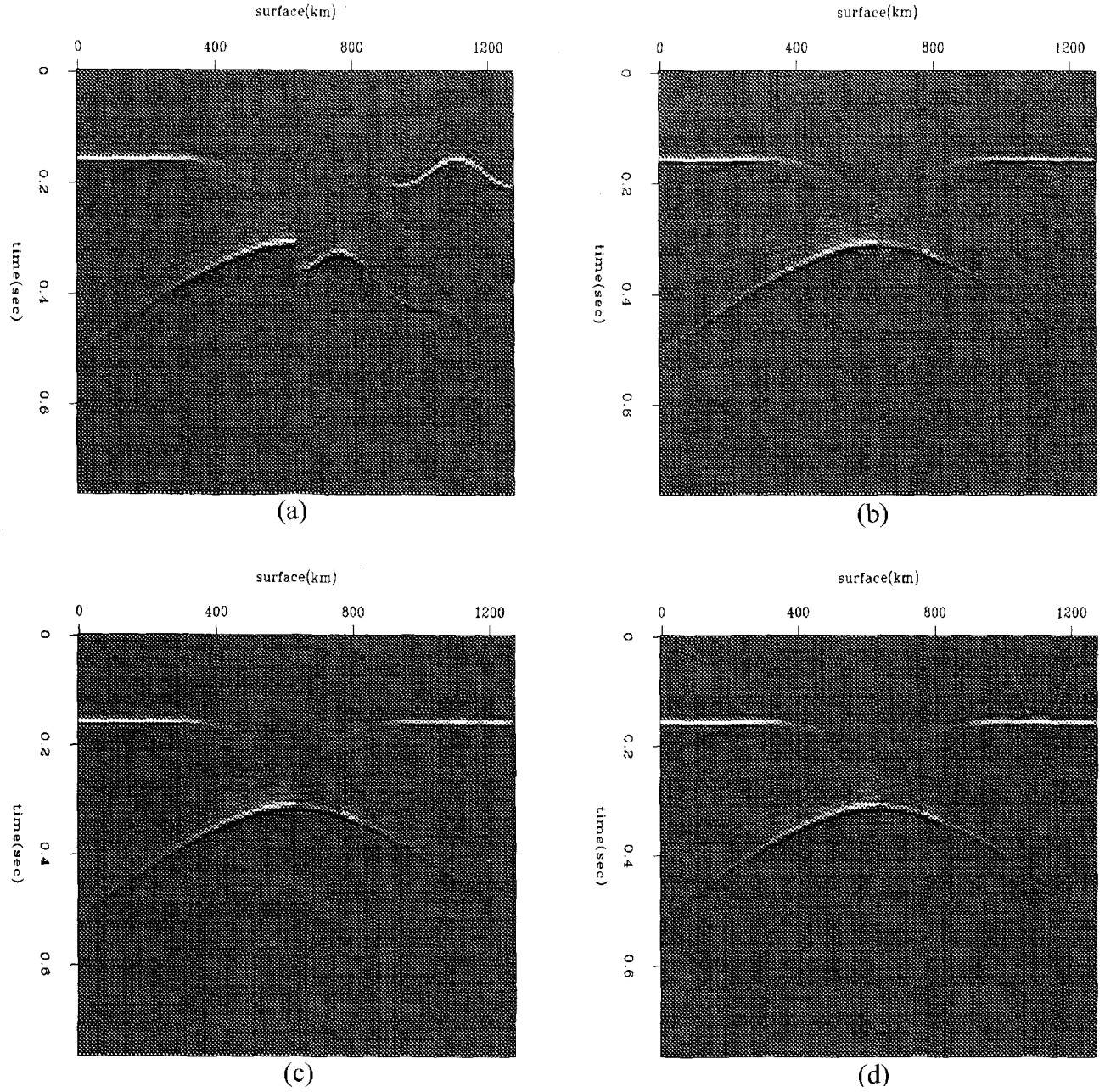

Fig. 5. (a) Wavefield recorded on the irregular surface using phase-shift extrapolation scheme. (b) Datumed wavefield using phase-shift algorithm. (c) Datumed wavefield using split-step algorithm. (d) Datumed wavefield using 45-degree finite-difference algorithm.

experiment has linear increase both in depth and laterally: $v(x, z)=1500.+0.2x+0.2z$ (Fig. 6(a)). The zero-offset data are modeled using split-step extrapolation; Fig. 6(b) shows the result. The datuming algorithm is then applied to the data using the split-step extrapolation; the result is shown in Fig. 7(a). Fig. 7(b) shows the migrated image. By comparing Fig. 4 with Fig. 7(b) we can see the effect of the datuming algorithm. The datuming algorithm using the finite-difference extrapolation are also tested for the same data Fig. 6(b); the datumed wavefield is shown in Fig. 7(c) and the migrated image is shown Fig. 7(d). In Fig. 7(d) we can see the reflector is imaged correctly execpt the steep dip portion, which is limited by the 45-degree wave equation.

Marine data examples

One of the practical applications of datuming in seismic data processing is layer replacement (Yilmaz and Lucas, 1986; Berryhill, 1986). The layer replacement refers to replacing the overburden velocity with the velocity of the substratum, thereby eliminating raypath bendings at the interface between the overburden and the substratum. Time migration after layer replacement can be a practical alternative to depth migration. Therefore, one of the most

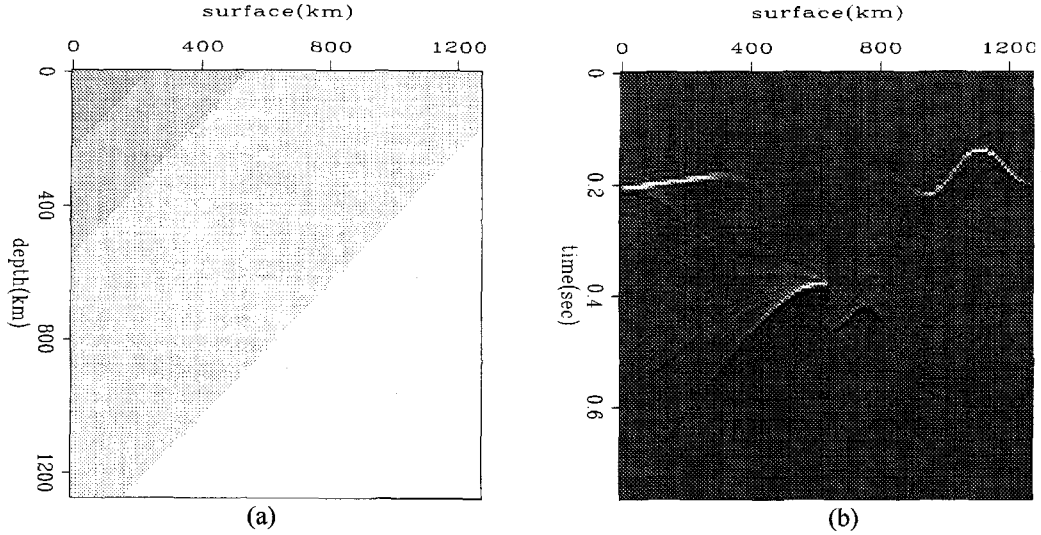

Fig. 6. (a) Velocity model ($v(x, z)$=1500.+0.2x+0.2z). (b) Wavefield recorded on the irregular surface using split-step extrapolation scheme

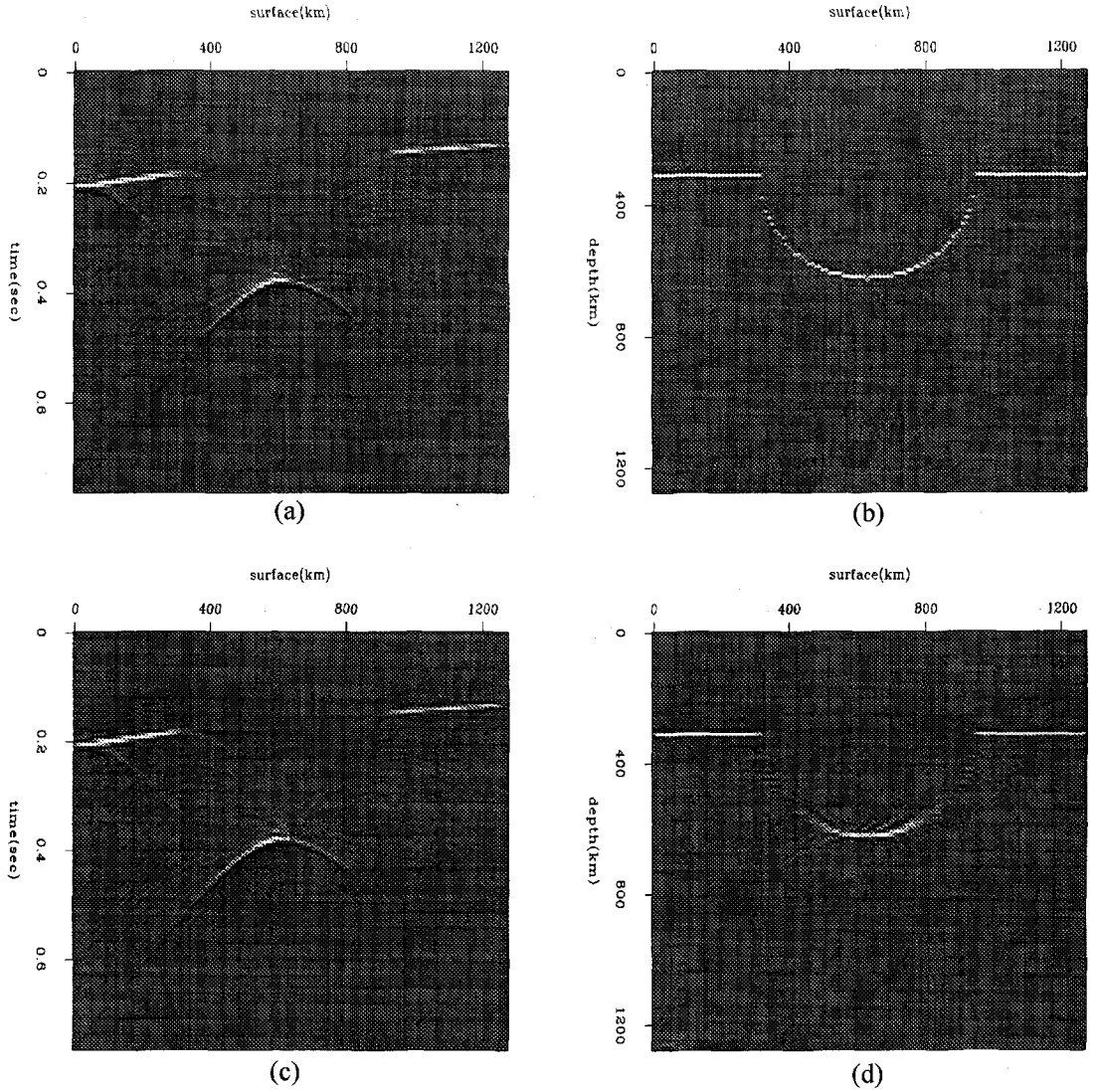

Fig. 7. (a) Datumed wavefield using split-step algorithm. (b) Migrated image using split-step algorithm. (c) Datumed wavefield using finite-difference algorithm. (d) Migrated image using finite-difference algorithm.

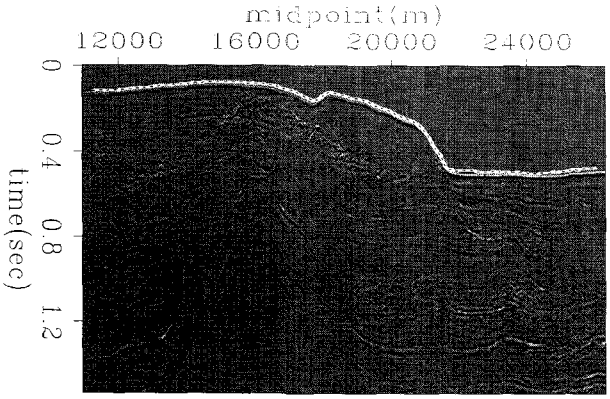

Fig. 8. Gulf of Mexico near-offset data with first break picks overlaid.

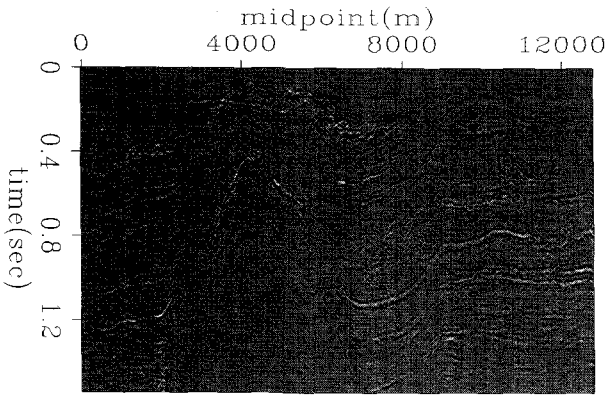

Fig. 9. Gulf of Mexico near-offset data datumed to the ocean bottom.

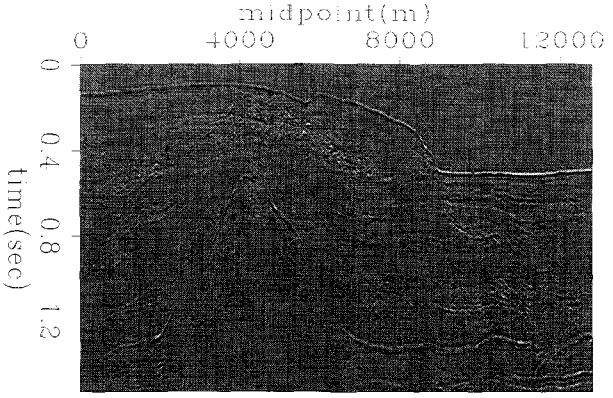

Fig. 10. Gulf of Mexico near-offset data redatumed to the sea level.

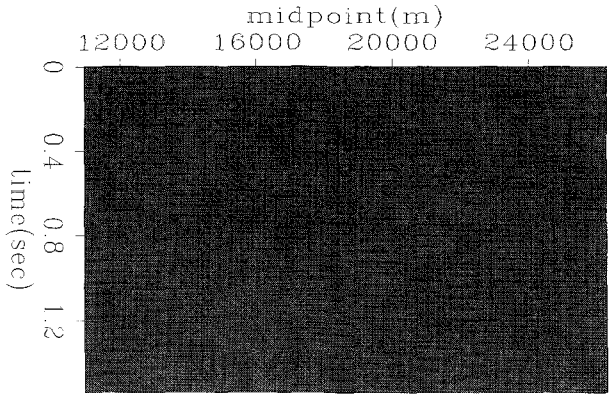

Fig. 11. The Difference between Figure 8 and Figure 10 real-sea-surface. We can see that two results are too close to show the difference in the paper printout gray scale.

important things in layer replacement might be preserving the correct spectral property of the original data. The ability to preserve the correct spectral property in layer replacement techniques can be verified by the unitary characteristic of the datuming operator used in the layer replacement.

I applied the datuming algorithm to marine data to illustrate the effectiveness of the datuming algorithm. The near-offset section of a marine data set is displayed in Fig. 8. This is the Gulf of Mexico data obtained by Amoco. The complex structure along the sea bottom causes raypath bending that induces distortions and disruptions on reflections beneath a complex structure. The dashed line in Figs. 8 represents a datum along the sea floor defined by the first break picks. The near offset is 30 meters so I treat the data as if it were zero-offset and input it to the datuming algorithm.

The datuming to the sea floor is performed using phase-shift extrapolation with water velocity, 1500. m/s, and the result is displayed in Fig. 9. To compare with the original data, I used the same water velocity in the process of upward continuing the data back to sea level (Fig. 10). Unlike the Kirchhoff datuming (Berryhill, 1979; Bevc, 1992) which loose some high frequency component, we can see that the datuming used in this paper preserve the spectral property of the original data after layer replacement. The difference between the original data and the data after layer replacement is shown in Fig. 11 and confirms the nice property of the datuming operator used.

Conclusion

This paper describes a poststack datuming scheme that use wavefield depth extrapolation. The method allows the use of any depth extrapolation technique such as phase-shift, split-step, or finite-difference extrapolation. The derivation of the datuming algorithms is obtained by transposing and taking the complex conjugate of the corresponding forward modeling operator which does upward extrapolation from a flat surface to an irregular surface. The exact adjoint relation between the forward modeling operator and the datuming operator is demonstrated algebraically and confirmed by the dot-product test. Several tests of the datuming algorithms with synthetic data and real data show that the method works well.

References

1. Beasley, C. J. and Lynn, W., 1989, The zero-velocity layer: Migration from irregular surfaces: Presented at the 59th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, 1179-1183.
2. Berryhill, R. T., 1979, Wave-equation datuming: Geophysics, **44**, 1329-1344.
3. Berryhill, R. T., 1984, Wave-equation datuming before stack: Geophysics, **49**, 2064-2066.
4. Claerbout, J. F., 1985, Imaging the Earth's Interior: Blackwell Scientific Publications.
5. Claerbout, J. F., 1992, Earth Soundings Analysis: Processing versus Inversion: Blackwell Scientific Publications.
6. Ji, J. and Claerbout, J. F., 1992, Phase shift datuming and migration from an irregular surface: SEP **75**, 157-170.
7. Ji, J., 1992, Least squares imaging, datuming, and interpolation using the wave equation: SEP **75**, 121-134.
8. Shtivelman, V. and Canning, A., 1988, Datum correction by wave equation extrapolation: Geophysics, **53**, 1311-1322.
9. Stoffa, P. L., Fokkema, J. T., de Luna Freire, R. M. and Kessinger, W. P., 1990, Split-step Fourier migration: Geophysics, **55**, 410-421.
10. Wiggins, J. W., 1984, Kirchhoff integral extrapolation and migration of nonplanar data: Geophysics, **49**, 1239-1248.
11. Yilmaz, O. and Lucas, D., 1986, Prestack layer replacement: Geophysics, **51**, 1355-1369.

Appendix

This appendix contains the codes in ratfor for datuming using phase shift, split step and 45-degree finite difference extrapolation methods. Some subroutines that are not listed here can be found in the Claerbout's recent publication (1992).

```
#
#       Datuming using Gazdag algorithm.
#
subroutine gazdtm( adj, nt,nx,nz, dt,dx,dz, datain,dataout, v,geoz,datum )
integer            adj, nt,nx,nz, iw,ikx,iz,ix,                    datum
complex                                  datain(nt,nx),dataout(nt,nx)
real      pi,w0,dw, kx0,dkx,       dt,dx,dz,                v(nz),geoz(nx)
temporary real     passed( nx)
temporary complex                           tdata(nt,nx)

call conjnull( adj, 0, datain,2*nt*nx, dataout,2*nt*nx )
pi = 4.*atan(1.);       w0  = -pi/dt;   dw  = 2.*pi/(nt*dt)
                        kx0 = -pi/dx;   dkx = 2.*pi/(nx*dx)

if( adj == 0) {  do ix=1,nx {   passed(ix) = 1. }}
else          {  call zero( nx, passed)          }

if( adj == 0) {
        call copy(          2*nt*nx, datain, tdata)
        call ft1axis( 0,  1., nt,nx,         tdata)
        call gate(  adj,      nt,nx, dataout,tdata, datum, geoz,passed)
        call ft2axis( 0, -1., nt,nx,         tdata)

        do iz = datum-1, 1, -1{
                call phsh(  adj,      nt,nx,   tdata, w0,kx0,dw,dkx,dz,v(iz+1))
                call ft2axis( 1, -1., nt,nx,         tdata)
                call gate(  adj,      nt,nx, dataout,tdata, iz, geoz,passed)
                call ft2axis( 0, -1., nt,nx,         tdata)
                }
        call ft1axis( 1, 1., nt,nx, dataout)
        }
else    {
        call copy(          2*nt*nx, dataout, tdata)
        call ft1axis( 0,  1., nt,nx,         tdata)
        call gate(  adj,      nt,nx,         tdata, datain, 1,geoz,passed)
        call ft2axis( 0, -1., nt,nx,                datain)

        do iz = 2, datum{
                call phsh(  adj,      nt,nx,datain, w0,kx0, dw,dkx,dz,v(iz))
                call ft2axis( 1, -1., nt,nx,       datain)
                call gate(  adj,      nt,nx, tdata,datain, iz,geoz,passed)
                call ft2axis( 0, -1., nt,nx,       datain)
                }
        call ft1axis( 1,  1., nt,nx, datain)
        call ft2axis( 1, -1., nt,nx, datain)
        }
return: end
```

```
#
#       Datuming using Split Step algorithm
#
subroutine splitdtm( adj, nt,nx,nz, dt,dx,dz, datain,dataout, vel,geoz,datum)
integer              adj, nt,nx,nz, it,ix,iz,                   iw,ikx,datum
real    pi, dw,dkx,w0,kx0,          dt,dx,dz, avgvel,vel(nz,nx),geoz(nx)
complex                                       datain(nt,nx),dataout(nt,nx)
temporary complex                              tdata(nt,nx)
temporary real tvel(nx), sdiff(nx),passed(nx)

pi = 4.*atan(1.);        w0 = -pi/dt;   dw  = 2.*pi/(nt*dt)
                        kx0 = -pi/dx;   dkx = 2.*pi/(nx*dx)

call conjnull( adj, 0, datain,2*nt*nx, dataout,2*nt*nx )

if( adj == 0 ) { do ix=1,nx { passed(ix) = 1. }}
else           { call zero(nx,passed)       }
if( adj == 0 ) {
        call copy(        2*nt*nx, datain, tdata)
        call ft1axis( 0, 1.,nt,nx,         tdata)
        call gate(  adj,    nt,nx, dataout,tdata, datum,geoz,passed)
        call ft2axis( 0,-1.,nt,nx,         tdata)

        do iz = datum-1,1,-1 {
                do ix = 1,nx {     tvel(ix) = vel(iz+1,ix) }
                call avgndiff( nx, tvel,avgvel,sdiff )
                call phsh(  adj,      nt,nx, tdata, w0,kx0, dw,dkx,dz,avgvel)
                call ft2axis( 1, -1., nt,nx, tdata )
                call addsh( adj,      nt,nx, tdata, w0,     dw,    dz,sdiff)
                call gate(  adj,     nt,nx, dataout,tdata, iz,geoz,passed)
                call ft2axis( 0, -1.,nt,nx,          tdata)
                }
        call ft1axis( 1, 1., nt,nx,dataout)
        }
else    {
        call copy(         2*nt*nx, dataout, tdata)
        call ft1axis( 0, 1., nt,nx,          tdata)
        call gate(  adj,     nt,nx,          tdata, datain, 1,geoz,passed)

        do iz = 2,datum {
                do ix = 1,nx {  tvel(ix) = vel(iz,ix) }
                call avgndiff(  nx,tvel,avgvel,sdiff )
                call addsh( adj,      nt,nx, datain, w0,     dw,    dz,sdiff)
                call ft2axis( 0, -1., nt,nx,       datain)
                call phsh(  adj,      nt,nx, datain, w0,kx0, dw,dkx,dz,avgvel)
                call ft2axis( 1, -1., nt,nx,       datain)
                call gate(  adj,      nt,nx, tdata,datain, iz,geoz,passed)
                }
        call ft1axis( 1,  1., nt,nx,datain)
        }

return; end


#
#       Datuming using wxz45 algorithm.
#
subroutine wxz45dtm( adj, nt,nx,nz, dt,dx,dz, datain,dataout,vel,geoz,datum)
integer              adj, nt,nx,nz, iw,ix,iz,                          datum
real   pi, alpha,beta, dw,w0,       dt,dx,dz,         vel(nz,nx),geoz(nx)
complex                                          datain(nt,nx),dataout(nt,nx)
temporary complex tdata( nt,nx)
temporary real tvel(nx),passed(nx)

call conjnull( adj, 0, datain,2*nt*nx, dataout,2*nt*nx )
pi = 4.*atan(1.)
w0 =   -pi/dt;           alpha = 1./(4.*dx*dx)
dw = 2.*pi/(dt*nt);       beta = dz/(4.*dx*dx)

if( adj == 0 ) {  do ix=1,nx {   passed(ix) = 1. }}
else           {  call zero( nx, passed) }
```

```
if( adj == 0 ) {
        call copy(         2*nt*nx,  datain, tdata)
        call ft1axis( 0,-1., nt,nx,          tdata)
        do iw=1,nt { if( w0+(iw-1)*dw == 0.) { do ix=1,nx { tdata(iw,ix)=0.}}}
        call gate( adj,     nt,nx, dataout, tdata, datum,geoz,passed)

        do iz=datum-1,1,-1 {
                do ix=1,nx { tvel(ix) = vel(iz+1,ix) }
                call wxz45( adj, nt,nx, tdata, dz,w0,dw, alpha,beta, tvel)
                call gate(  adj, nt,nx, dataout,tdata,iz,geoz,passed)
                }
        call ft1axis( 1,-1., nt,nx, dataout)
        }
else    {
        call copy(         2*nt*nx, dataout, tdata)
        call ft1axis( 0,-1., nt,nx,          tdata)
        do iw=1,nt { if( w0+(iw-1)*dw == 0.) { do ix=1,nx { tdata(iw,ix)=0.}}}
        call gate( adj,     nt,nx,          tdata, datain, 1,geoz,passed)

        do iz=2,datum {
                do ix=1,nx { tvel(ix) = vel(iz,ix) }
                call wxz45( adj, nt,nx, datain, dz,w0,dw,alpha,beta,tvel)
                call gate(  adj, nt,nx, tdata,datain, iz,geoz,passed)
                }
        call ft1axis( 1,-1., nt,nx,       datain)
        }
return; end


#
#       filter for get and kill traces
#
#       if( adj == 0 )                  if( adj == 1 )
#               |du| - | I F | \/ |du|          |du| - | I 0 | \/ |du|
#               |dd| - | 0 K | /\ |dd|          |dd| - | F K | /\ |dd|
#
#
subroutine gate( adj, nt,nx, dataup,datadown,     depth,geoz,    passed)
integer          adj, nt,nx,                  i,j, depth
complex                     dataup(nt,nx), datadown(nt,nx)
real                                                geoz(nx),passed(nx)

do j=1,nx
        {
        if( adj!=0 & passed(j) != 1. ) {  do i=1,nt { datadown(i,j) = 0. }}
        if( geoz(j) == depth*1.)
                {
                do i=1,nt
                        {
                        if(adj==0){ dataup(i,j) = dataup(i,j) + datadown(i,j) }
                        else {    datadown(i,j) = datadown(i,j) + dataup(i,j) }
                        }
                if(adj==0){ passed(j) = 0. }
                else       { passed(j) = 1. }
                }
        if( adj==0 & passed(j) != 1. ){ do i=1,nt { datadown(i,j) = 0. }}
        }

return; end


#
#       Perform phase shift
#
subroutine phsh( adj, nt,nx, data, w0,kx0, dw,dkx,dz,         vel)
integer          adj, nt,nx,       iw,ikx
real    ktau2,                     w0,kx0, dw,dkx,dz,signum, vel, w,kx,w2,kx2
complex cz,                  data(nt,nx)

do ikx = 1, nx { kx = kx0 + (ikx-1) * dkx ; kx2 = kx * kx
do iw  = 1, nt { w  = w0  + (iw -1) * dw  ;  w2 =  w *  w
                 ktau2 =  w2/(vel*vel) - kx2
             if( ktau2 > 0){ cz = cexp( cmplx(0.,signum(w)*dz*sqrt(ktau2)) ) }
             else          { cz = 0.                                         }
```

```
            if(adj==0)
                  data(iw,ikx) = data(iw,ikx) *   cz
            else
                  data(iw,ikx) = data(iw,ikx) * conjg(cz)
          }}
rn; end
```

Additional phase shifting in Split-Step algorithm

```
outine addsh( adj, nt,nx, data,    w0,dw,dz,  sdiff)
ger           adj, nt,nx,                              iw,ix
                              w,w0,dw,dz,  sdiff(nx)
lex kz,                  data(nt,nx)

w= 1,nt {   w = w0 + (iw-1)*dw
do ix= 1,nx {   kz=cexp(cmplx(0., w*dz*sdiff(ix)))
        if( adj == 0 )
                data(iw,ix) = data(iw,ix) * kz
        else
                data(iw,ix) = data(iw,ix) * conjg(kz)
        }}

return; end


#
#       Calculate average velocity and slownwss differences from average
#                for Split-step extrapolation algorithm
#
subroutine avgndiff( n, tvel,    avgvel,sdiff )
integer           i, n
real                    tvel(n), avgvel,sdiff(n), avgslow
avgvel = 0.
do i= 1,n
        avgvel = avgvel + tvel(i)
avgslow = n/avgvel
avgvel = avgvel/n
do i= 1,n
        sdiff(i) = 1./tvel(i) - avgslow

return; end


#
#       This program does 45-degree Finite difference wavefield
#       extrapolation downward [ adj = 1 ] and
#       modeling [ adj = 0 ] in wxz domain. ( Dot product test passed )
#
subroutine wxz45( adj, nt,nx, data, dz,w0,dw,   alpha,beta, vel)
integer           adj, nt,nx,       iw,ix
real                                dz,w0,dw,w, alpha,beta, vel(nx)
complex cshift,                data(nt,nx)
temporary complex tdata(nx),cd(nx),ce(nx),cf(nx)
temporary complex aa(nx),bb(nx), a(nx),b(nx),c(nx),d(nx),e(nx),f(nx)

if( adj==0 ) {
        do iw=1,nt {    w = w0 + (iw-1)*dw;   if(w!=0.) {
        do ix=1,nx {    cshift = cexp( cmplx( 0., -w*dz/vel(ix)))
                        data(iw,ix) = data(iw,ix)*cshift
                        cd(     ix) = data(iw,ix)               }
        call fdcoeff( w,vel,nx,alpha,beta, aa,bb,a,b,c,d,e,f)
        call triadj(adj,nx,a,b,c,tdata,cd,ce,cf)
        do ix=2,nx-1
                data(iw,ix) = conjg(f(ix+1))*tdata(ix+1)
                data(iw, 1) = conjg(f(   2))*tdata(   2)
                data(iw,nx) = conjg(d(nx-1))*tdata(nx-1)
        }}
        }
else    {
```

```
        do iw=1,nt {    w = w0 + (iw-1)*dw;   if(w!=0.){
        call fdcoeff( w,vel,nx,alpha,beta, aa,bb,a,b,c,d,e,f)
        do ix=2,nx-1
                cd(ix) = d(ix) * data(iw,ix+1) + e(ix) * data(iw,ix)
                cd(1)  = d( 1) * data(iw,   2) + e( 1) * data(iw, 1)
                cd(nx) = f(nx) * data(iw,nx-1) + e(nx) * data(iw,nx)
        call triadj(adj,nx,a,b,c,cd,tdata,ce,cf)
        do ix=1,nx {    cshift = cexp( cmplx( 0., w*dz/vel(ix)))
                        data(iw,ix) = tdata(ix) * cshift }
               }}
        }

return;         end

#
#       find difference star coefficients
#
subroutine fdcoeff( w,    vel,n,  alpha,beta,  aa,   bb,   a,b,c,d,e
integer    i,                n

 real               w,w2, vel(n), alpha,beta
 complex        a(n),b(n),c(n),d(n),e(n),f(n),  aa(n),bb(n)
 w2 = w*w
        do i=1,n{
                aa(i) = alpha*vel(i)**2 / w2
                bb(i) =  beta*vel(i)*(0.,1.) / w
                 a(i) = aa(i)-bb(i); b(i) = 1.-2.*a(i); c(i) = a(i)
                 d(i) = aa(i)+bb(i); e(i) = 1.-2.*d(i); f(i) = d(i)
                }
 return; end
 #



 #
 #  Tridiagonal solver for 45-degree Finite-Difference Extrapolation
 #
 #       adj == 1   :  T y = x
 #       adj == 0   :  T'x = y
 #
 subroutine triadj(adj,n,a,b,c,x,y,e,f)
 integer n,i, adj
 complex y(n),x(n),f(n),e(n),a(n),b(n),c(n),den
 temporary complex ta(n)

 if ( adj == 1 ) {
        e(1) = -a(1)/b(1);              f(1) = x(1)/b(1)
        do i = 2,n-1 {
                den = b(i)+c(i)*e(i-1)
                e(i) = -a(i)/den
                f(i) = (x(i)-c(i)*f(i-1))/den
                }
        y(n) = (x(n)-c(n)*f(n-1))/(b(n)+c(n)*e(n-1))
        do i = n-1,1,-1
                y(i) = e(i)*y(i+1)+f(i)
                }
 else {  do i = 2, n
                ta(i) = conjg(a(i-1))
        do i = n-1, 1, -1
                a(i) = conjg(c(i+1))
        do i = 2, n
                c(i) = ta(i)
        do i = 1, n
                b(i) = conjg(b(i))
        e(1) = -a(1)/b(1);              f(1) = y(1)/b(1)
        do i = 2,n-1 {
                den = b(i)+c(i)*e(i-1)
                e(i) = -a(i)/den
                f(i) = (y(i)-c(i)*f(i-1))/den
                }
        x(n) = (y(n)-c(n)*f(n-1))/(b(n)+c(n)*e(n-1))
        do i = n-1,1,-1
                x(i) = e(i)*x(i+1)+f(i)
        }

 return: end
```