Stanford Exploration Project, Report 75, November 17, 1997, pages 203–234

Stanford Exploration Project, Report 75, November 17, 1997, pages 203–234

Wave-equation datuming in variable velocity media

Alexander M. Popovici[1]

keywords: *algorithm, velocity, modeling, Fourier transform*

ABSTRACT

Wave equation datuming represents the process of upward or downward continuing seismic data to a new reference surface. Usually datuming algorithms are based on the Kirchhoff integral solution to the wave equation. I present a new wave-equation datuming algorithm based on wavefield continuation through Fourier methods which handles laterally and depth varying velocity models. I show datuming results for different velocity models and the migration results before and after datuming.

INTRODUCTION

Seismic data recorded on a nonflat surface topography presents several problems to routine imaging processing. The standard, optimized migration algorithms assume data is recorded on a flat surface. The static correction routinely applied to compensate for topography is inaccurate for waves which don't propagate vertically. Shtivelman and Canning (1988) show the accuracy limitations of static shifting and the need to apply wave-equation datuming when differences in elevation are significant and the velocity model is complicated.

The concept of wave-equation datuming was first presented by Berryhill (1979) and applied to zero-offset (poststack) data. The method employed by Berryhill is based on an extrapolating scheme using the Kirchhoff integral solution to the scalar wave-equation. Berryhill (1984) generalizes his method to prestack sections by applying the same extrapolation algorithm to common-source gathers and then to common-receivers gathers. Wiggins (1984) presents the same concept of Kirchhoff integral extrapolation of prestack data but adds an imaging step to the algorithm. We can use the same algorithm to:

1. Downward extrapolate data from a datum to a flat surface.

2. Continue to downward extrapolate and image for migration.

[1]**email:** not available

Yilmaz and Lucas (1986) demonstrate how datuming can be used twice to replace a layer of arbitrary shape and known velocity to another layer with a different velocity in order to eliminate raypath bending in the area with severe velocity contrast. They call the technique prestack layer replacement. The method uses wave-equation datuming twice; first to downward continue the wavefield from an initial surface to a datum, using a known velocity, and second to upward continue the wavefield from the datum to the initial surface using a different velocity. For prestack data the sources and the receivers are downward continued independently to the datum and subsequently upward continued.

Reshef (1991) presents a datuming and migration principle to be used in conjunction with phase-shift methods to directly migrate prestack data. He performs downward extrapolation from a flat datum and adds data to the extrapolated wavefield each time the topographic surface is intersected. This method allows direct prestack migration of data recorded on a nonflat topographic surface.

Ji and Claerbout (1992) also present an improved datuming and migration algorithm for depth varying $v(z)$ velocity media. Bevc (1992) improves the datuming and migration algorithm via an antialiasing Kirchhoff method. Both papers use the conjugate transpose concept to examine the algorithm.

In this paper I present a method to perform zero-offset datuming and migration for any velocity model and any topographic surface. The algorithm is based on the phase-shift extrapolation of the wavefield. For laterally varying velocity I use a Phase-Shift Plus Interpolation (PSPI) and a Split-Step method. Datuming is performed in both directions, from an uneven topographic datum to a flat surface and from a flat surface up to a topographic datum. The two algorithms are conjugate transpose to each other. To complement the two datuming algorithms I use another pair of conjugate operators to perform migration and modeling via a PSPI or Split-Step algorithm. I show results on different velocity models for the two datuming algorithms and the accompanying migration and modeling algorithms.

WAVE-EQUATION DATUMING ALGORITHM

Wave-equation datuming is the 2-D equivalent of the 1-D static shifting. It takes seismic data from a given surface and transports it to another surface. We hope that in the transportation process nothing is lost, and seismic data will look exactly as if it is recorded on the secondary surface.

We can formulate the datuming algorithm in a similar way we can formulate the migration algorithm being the conjugate transpose of the modeling algorithm. Given the data recorded on a non-level surface, we want to continue the data to a level surface. We can define the direct problem as a wave equation extrapolation from a level surface to an irregular surface. The conjugate transpose to this problem will bring the data from the irregular topographic surface to the level surface.

The direct problem is easier to formulate than the conjugate. Suppose we record a wavefield in a zero-offset experiment on a flat surface. The direct problem is to upward continue the wavefield to the uneven surface, similar to the one in Figure **??**. I propose a scheme in which the wavefield is propagated upward using a phase-shift or PSPI method and the values of the wavefield are extracted at each height corresponding to the datum. The advantage of the method is that it allows us to use simple and fast algorithms to extrapolate the wavefield while the disadvantage is that the algorithm is only an approximation to the exact solution.

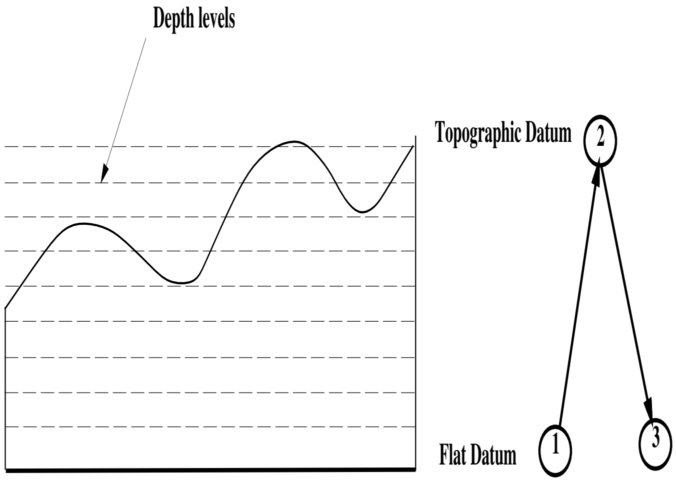


Figure 1: Data recorded on the flat datum (1) is extrapolated to the datum (2) and then, using the conjugate transpose algorithm it is extrapolated back to the flat datum(3). mihai2-Datumdraw [NR]

The conjugate transpose algorithm starts by downward extrapolating the data from the highest point on the topographic datum. Each time the wavefield reaches the datum, data is inserted into the downward propagated wavefield.

The mathematical formulation of the direct datuming algorithm is simpler in the case of depth variable velocity. However I should note that for computational purposes there is almost no speed difference between a depth varying velocity $v(z)$ medium and a laterally and depth varying velocity $v(x,z)$ medium. The wavefield $p(x, z = 0, t)$ is Fourier transformed in time to have $P(x, z = 0, \omega)$ and each frequency can be independently upward extrapolated. By Fourier transforming the horizontal space variable, the wave equation can be written as

$$\frac{\partial^2 P(k_x, z, \omega)}{\partial z^2} = (k_x^2 - \frac{\omega^2}{v^2(z)})P(k_x, z, \omega) \tag{1}$$

valid for all values of k_x and ω. For a constant velocity medium we introduce a constant k_z as

$$k_z = \pm[\frac{\omega^2}{v^2} - k_x^2]^{\frac{1}{2}}. \tag{2}$$

k_z is constant for two given values of k_x and ω. Equation (1) becomes an ordinary differential equation

$$\frac{\partial^2 P}{\partial z^2} = -k_z^2 P. \tag{3}$$

For a constant k_z it has the analytic solution

$$P(k_x, z_0 + z, \omega) = P(k_x, z_0, \omega) e^{ik_z z}. \tag{4}$$

For depth varying velocity $v(z)$, k_z is considered approximately constant for small depth intervals (Δz) where the velocity is considered constant. Therefore equation (4) can be written

$$P(k_x, z = z_0 + \Delta z, \omega) = P(k_x, z = z_0, \omega) e^{ik_z \Delta z}. \tag{5}$$

This form of the equation can be used to downward or upward extrapolate the wavefield for a small depth interval.

Thus the forward datuming algorithm can be summarized in four steps.

- Fourier transform the wavefield $P(x, z = z_0, \omega)$ over the space variable x.
- Upward propagate the wavefield $P(k_x, z = z_0, \omega)$ to the next depth level $P(k_x, z_0 - \Delta z, \omega)$.
- Forward transform the wavefield $P(k_x, z = z_0 - \Delta z, \omega)$ over the space variable x
- Extract the traces corresponding to a depth level on the topographic surface.

Two conjugate transpose datuming algorithms

We can epitomize the datuming algorithm in a depth varying velocity by writing the successive operators in matrix form. For simplicity I will take only a datum with three depth levels similar to the one in Figure **??**. The algorithm can be generalized for any number of depth levels and any datum geometry. For each value of ω we have

$$\begin{bmatrix} A & B & C \end{bmatrix} \begin{bmatrix} FT^* & & \\ & FT^* & \\ & & FT^* \end{bmatrix} \begin{bmatrix} W_1 \\ W_2 W_1 \\ W_3 W_2 W_1 \end{bmatrix} \begin{bmatrix} FT \end{bmatrix} \begin{bmatrix} P(x, z = 0, \omega) \end{bmatrix} \tag{6}$$

where $P(x, z = 0, \omega)$ represents the zero-offset data after Fourier transformation along the time axis. The matrix

$$\begin{bmatrix} FT \end{bmatrix}$$

performs the Fourier Transform matrix of the horizontal space variable. Therefore the matrix

$$\begin{bmatrix} FT^* & & \\ & FT^* & \\ & & FT^* \end{bmatrix}$$

is composed of inverse Fourier Transform block matrices, each being the conjugate transpose to the direct transformation. The matrix

$$\begin{bmatrix} W_1 \\ W_2 W_1 \\ W_3 W_2 W_1 \end{bmatrix}$$

contains three blocks corresponding to the product of diagonal matrices of the form

$$W_i = \begin{bmatrix} e^{ik_{zi1}\Delta z} & & & \\ & e^{ik_{zi2}\Delta z} & & \\ & & \ddots & \\ & & & e^{ik_{zi6}\Delta z} \end{bmatrix}. \tag{7}$$

On each diagonal are the phase-shifting exponentials necessary to upward extrapolate the wavefield to a depth level. The value of $(k_z)_{ij}$ is given by the dispersion relation:

$$(k_z)_{ij} = \sqrt{\frac{\omega^2}{v(z_i)^2} - k_{xj}^2}.$$

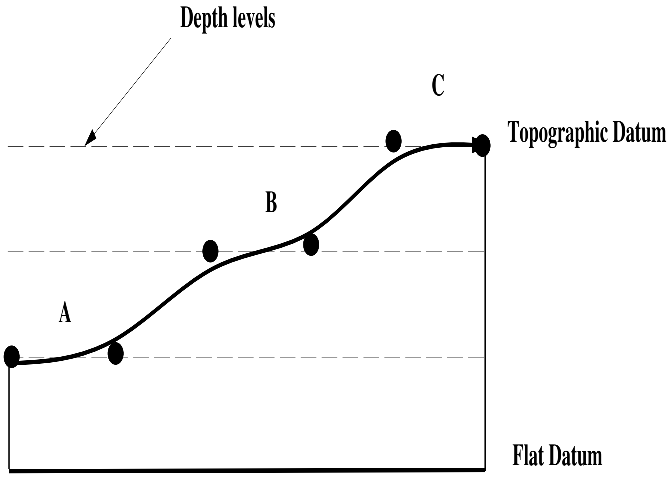


Figure 2: Datuming for only three depth levels. Data extrapolated to the first depth level is extracted by matrix **A**, to the second level by matrix **B** and to the third by matrix **C**. $\boxed{\text{mihai2-Datumatrix}}$ [NR]

The matrices **A,B,C** are matrices corresponding to the shape of the datum. For the case shown in Figure **??** the matrices **A,B,C** are

$$A = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & \cdot & & & \\ & & & \cdot & & \\ & & & & \cdot & \\ & & & & & \cdot \end{bmatrix}, B = \begin{bmatrix} \cdot & & & & & \\ & \cdot & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & \cdot & \\ & & & & & \cdot \end{bmatrix}, C = \begin{bmatrix} \cdot & & & & & \\ & \cdot & & & & \\ & & \cdot & & & \\ & & & \cdot & & \cdot \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix}$$

The conjugate transpose algorithm can be obtained by reversing the order of matrix multiplication in equation (6) and transposing-conjugating each matrix. The matrices **A,B,C** are diagonal matrices and therefore equal to the transpose. For each value of the frequency ω the conjugate datuming is

$$\begin{bmatrix} FT^* \end{bmatrix} \begin{bmatrix} W_1^* & W_1^*W_2^* & W_1^*W_2^*W_3^* \end{bmatrix} \begin{bmatrix} FT & & \\ & FT & \\ & & FT \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} \begin{bmatrix} P(x, z_{dat}, \omega) \end{bmatrix} \quad (8)$$

where $P(x, z_{dat}, \omega)$ represents the wavefield recorded on the topographic datum. The matrices W_i^* are conjugate transposed of the diagonal matrices W_i which are explicitly represented in equation (7).

Wave-equation Datuming for laterally variable velocity

For laterally variant media, Gazdag and Sguazzero (1984) propose to downward extrapolate the wavefield one depth interval at a time with several velocities. We can apply the same idea to upward propagate the wavefield with several velocities. To obtain a single upward propagated wavefield at each space point the value of the resulting wavefield is interpolated between the two wavefields with the closest velocities. Linear interpolation between two values P_1 and P_2 is an operation of the type

$$\begin{bmatrix} \tilde{P} \end{bmatrix} = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \end{bmatrix}$$

where the conjugate transpose operator is spreading a value with two weights w_1 and w_2

$$\begin{bmatrix} \tilde{P}_1 \\ \tilde{P}_2 \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \begin{bmatrix} P \end{bmatrix}.$$

The PSPI upward datuming can be written as

$$\begin{bmatrix} A & B & C \end{bmatrix} \begin{bmatrix} U_1 \\ U_2U_1 \\ U_3U_2U_1 \end{bmatrix} \begin{bmatrix} P(x, z=0, \omega) \end{bmatrix} \quad (9)$$

where the matrices U_i represent the operator for upward continuation of the wavefield to the depth level i. The matrix U_i can be further decomposed into the sequence

$$\begin{bmatrix} FT^* \end{bmatrix} \begin{bmatrix} I & I \end{bmatrix} \begin{bmatrix} W_i1 & \\ & W_i2 \end{bmatrix} \begin{bmatrix} FT & \\ & FT \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}. \quad (10)$$

The matrix

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

contains two diagonal matrices w_1 and w_2 which have constant coefficients along the diagonal. Multiplication by this matrix, which is the transpose to linear interpolation, splits the data with two different weights. The matrix

$$\begin{bmatrix} W_i1 & \\ & W_i2 \end{bmatrix}$$

contains two diagonal matrices, which perform the upward extrapolation with different velocities. Each element of the diagonal is a complex exponential as in equation (7). And finally, the matrix

$$\begin{bmatrix} I & I \end{bmatrix}$$

contains two identity matrices which add the two wavefields.

The conjugate transpose algorithm is found by transposing each matrix and reversing the multiplication order:

$$\begin{bmatrix} U_1^* \\ U_2^* U_1^* \\ U_3^* U_2^* U_1^* \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} \begin{bmatrix} P(x, z_{dat}, \omega) \end{bmatrix} \tag{11}$$

The matrices U_i^* represent the operator for downward continuation of the wavefield to the depth level i. The matrix U_i^* can be further decomposed into the sequence

$$\begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} FT^* & \\ & FT^* \end{bmatrix} \begin{bmatrix} W_i^*1 & \\ & W_i^*2 \end{bmatrix} \begin{bmatrix} I \\ I \end{bmatrix} \begin{bmatrix} FT \end{bmatrix}. \tag{12}$$

The physical interpretation of equation (12) is that the wavefield, after Fourier transformation, is split and downward continued with two different velocities. The two wavefields are independently inverse Fourier transformed and then interpolated. This sequence is then repeated for each depth level.

The Split-Step algorithm is very similar to the PSPI algorithm, with the difference being that a single average velocity is used and an extra phase-shift correction is applied. Although the up-datuming sequence is exactly the same as in equation (9), the values of U_i are different then in equation (10). The matrix U_i is decomposed in the sequence

$$\begin{bmatrix} FT^* \end{bmatrix} \begin{bmatrix} W_i \end{bmatrix} \begin{bmatrix} FT \end{bmatrix} \begin{bmatrix} S_i \end{bmatrix}. \tag{13}$$

The only new matrix here is the matrix S_i which has the form

$$S_i = \begin{bmatrix} e^{i(\frac{1}{v(x_1,z_i)} - \frac{1}{v_{med}})\omega\Delta z} & & & \\ & e^{i(\frac{1}{v(x_2,z_i)} - \frac{1}{v_{med}})\omega\Delta z} & & \\ & & \ddots & \\ & & & e^{i(\frac{1}{v(x_6,z_i)} - \frac{1}{v_{med}})\omega\Delta z} \end{bmatrix} \tag{14}$$

and is responsible for a laterally varying phase-shift correction.

The conjugate transpose downward-datuming has the form given in equation (11) except that the values of U_i are found by transposing the matrices in equation (13)

$$\begin{bmatrix} S_i^* \end{bmatrix} \begin{bmatrix} FT^* \end{bmatrix} \begin{bmatrix} W_i^* \end{bmatrix} \begin{bmatrix} FT \end{bmatrix}. \tag{15}$$

Wave-equation Datuming Models

To illustrate the datuming algorithm I use several velocity models. I use the topographic surface displayed in Figure ??. The reflectivity model consists of a syncline in the middle of a horizontal layer. In order to analyze how the datuming algorithm affects the initial data I apply the conjugate pair of datuming operators and compare the resulting data with the initial data.

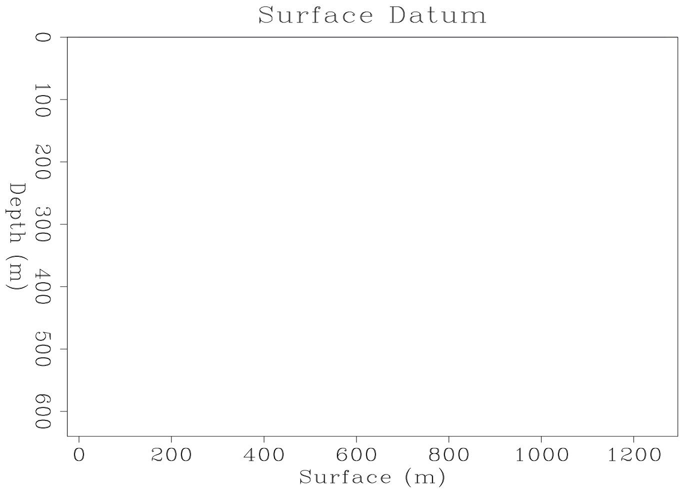


Figure 3: The surface topography (datum) used for the datuming examples. $\boxed{\text{mihai2-datum}}$ [NR]

The initial data is the result of zero-offset modeling using the syncline reflectivity model with different velocity models. The zero-offset data is obtained on a flat datum. Figure ??a shows an example for a constant velocity medium. The zero-offset data is then extrapolated to the uneven topographic datum shown in Figure ?? to obtain the data in Figure ??b. Using the conjugate transpose algorithm, data is extrapolated back to the flat datum to obtain the third panel in Figure ??c. By comparing the first and the third panel in Figure ?? we can estimate the effect of the datuming algorithm on the initial data.

The initial zero-offset data is migrated prior to the datuming step in Figure ??a and then again after the pair of conjugate datuming operators is applied. The migrated image after conjugate datuming is shown in Figure ??b.

I use several velocity models for assessing the effects of the datuming algorithm on zero-offset data. The first velocity model, used to obtain Figures ?? and ??, is just constant. The next model, used to obtain Figures ?? and ??, is a linearly increasing velocity in depth $v(z) = 1000. + 0.6z$ $[m/s]$. The third velocity model is linearly increasing both in depth and laterally $v(x, z) = 1000. + 0.6z + 0.2x$ $[m/s]$. For this case I use a PSPI and a Split-Step datuming algorithm. The last velocity model consists of three vertical areas of constant velocity $v(x) = (1000., 2000., 3000.)$ $[m/s]$.

All the conjugate operators used in this paper (Gazdag migration and modeling, Gazdag datuming, PSPI migration and modeling, PSPI datuming, Split-Step migration and modeling, Split-Step datuming) pass the Dot-Product test.

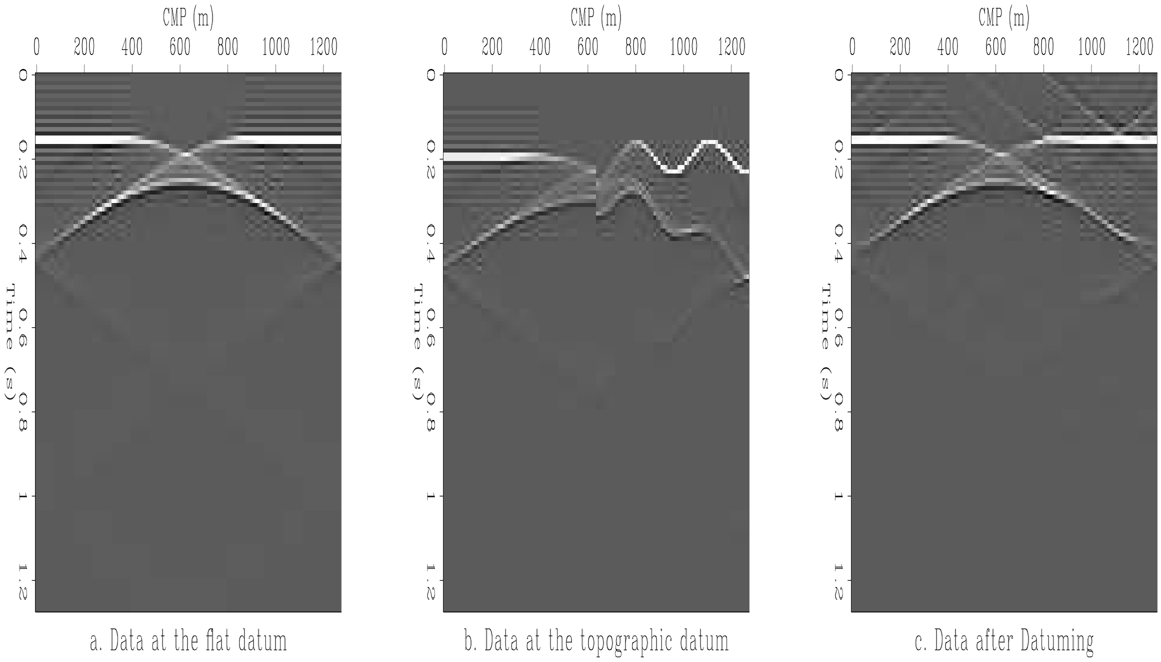


Figure 4: Effect on initial data of the datuming algorithm (constant velocity).
a. Initial data at the flat surface.
b. Data on the topographic datum.
c. Data after datuming to the topographic surface and back. mihai2-data3datum [ER]

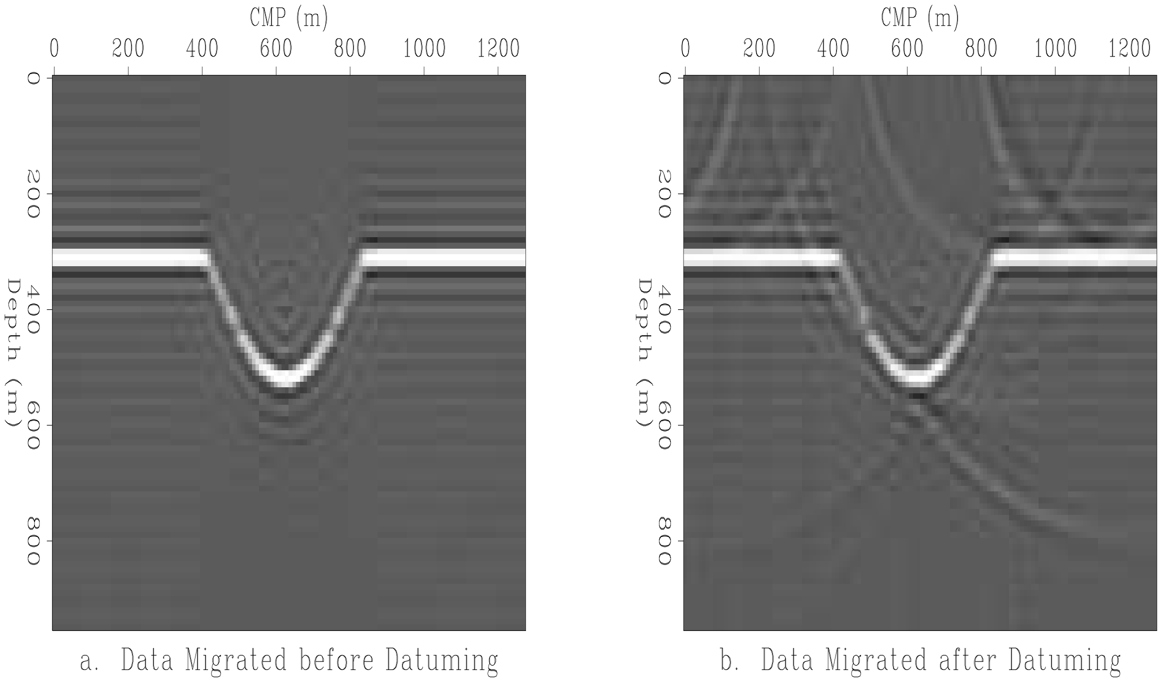


Figure 5: Migration before and after datuming in constant velocity.
a. Migration before datuming.
b. Migration after continuing the wavefield to the topographic surface and back to the flat datum. $\boxed{\text{mihai2-Mig2Datum}}$ [ER]

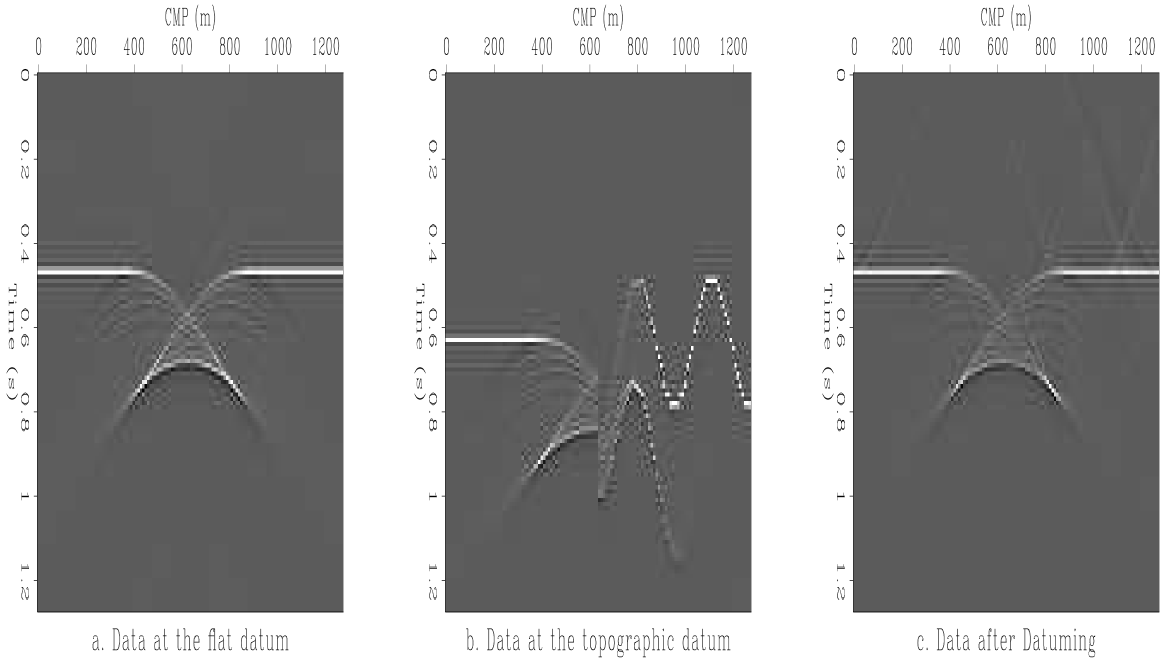
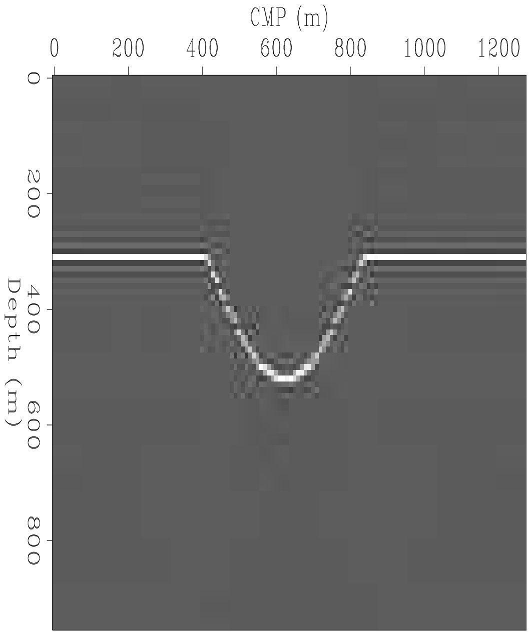
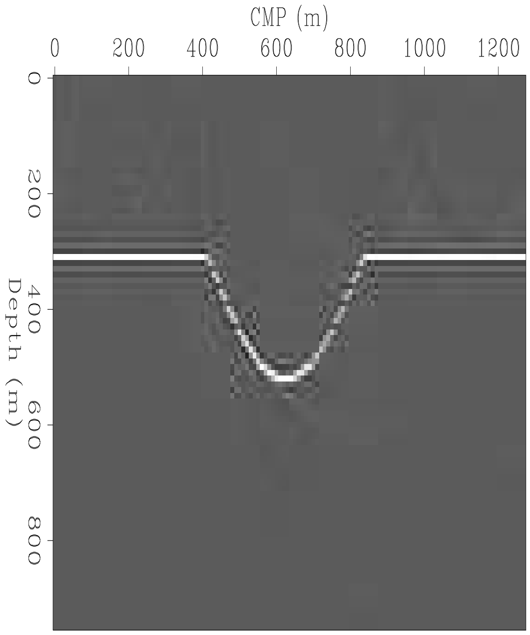


Figure 6: Effect on initial data of the datuming algorithm $v(z) = 1000 + 0.6 * z$.
a. Initial data at the flat surface.
b. Data on the topographic datum.
c. Data after datuming to the topographic surface and back. mihai2-data3datumvz [ER]



a. Data Migrated before Datuming



b. Data Migrated after Datuming

Figure 7: Migration before and after datuming in depth variable velocity v(z).
a. Migration before datuming.
b. Migration after continuing the wavefield to the topographic surface and back to the flat datum. mihai2-Mig2Datumvz [ER]

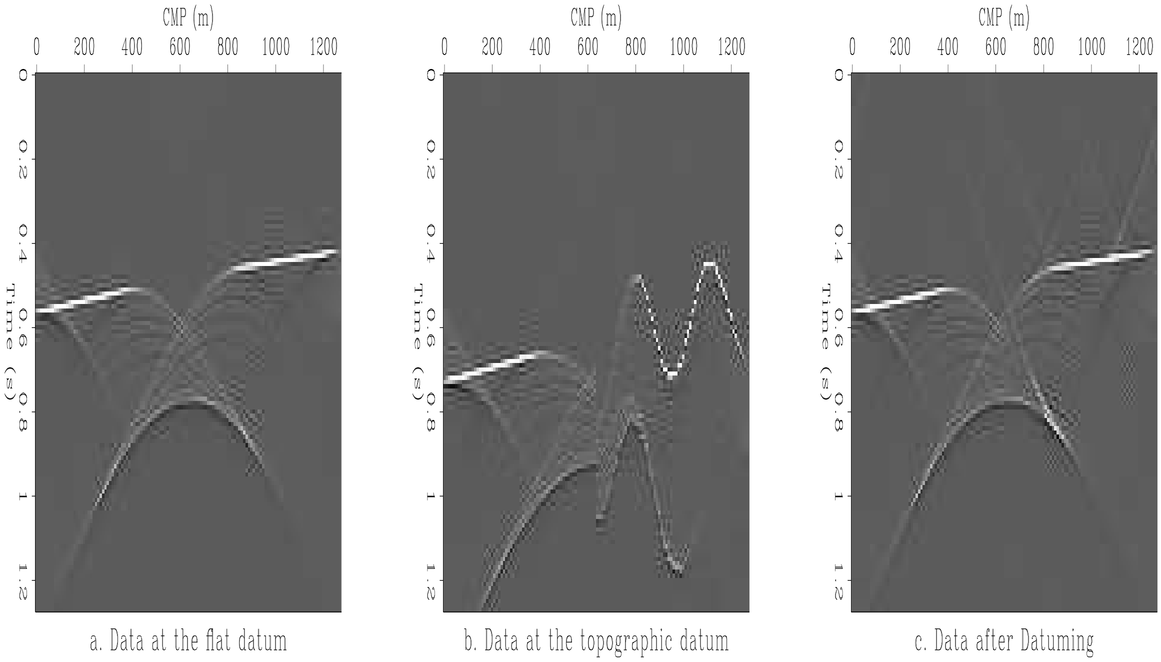


Figure 8: Effect on initial data of the datuming algorithm v(x,z). Split-Step.
a. Initial data at the flat surface.
b. Data on the topographic datum.
c. Data after datuming to the topographic surface and back. $\boxed{\text{mihai2-data3datumsplit}}$ [ER]

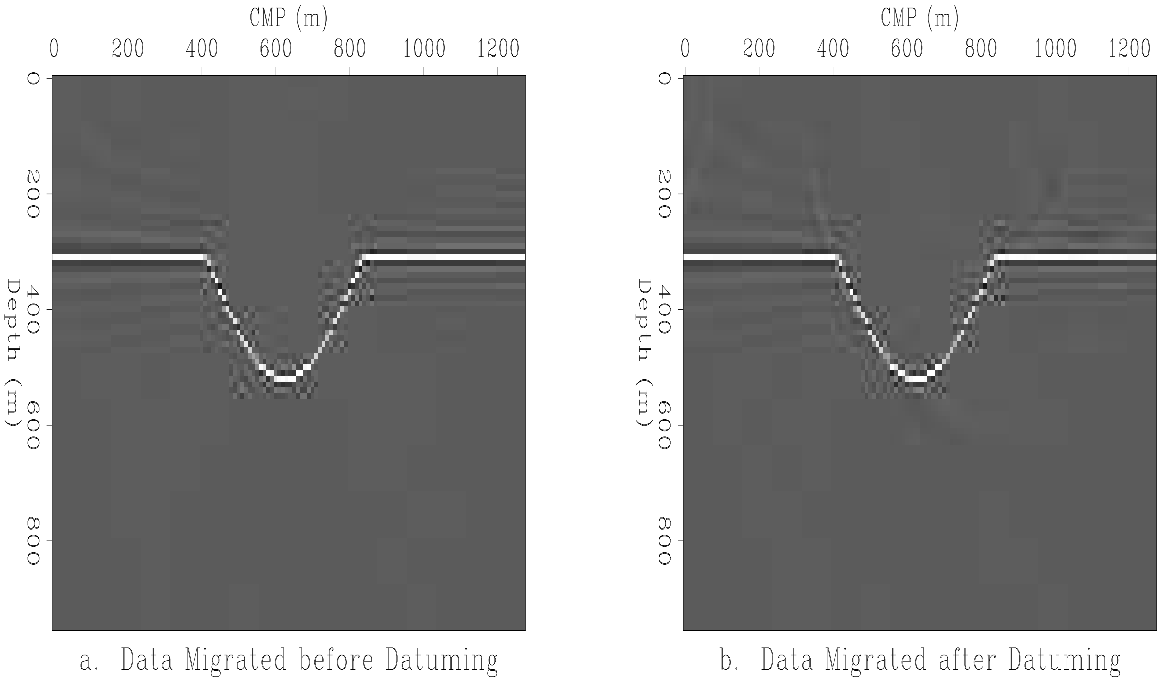


Figure 9: Migration before and after datuming. $v(x,z) = v_0 + ax * x + az * z$.
a. Migration before datuming (Split-Step).
b. Migration after continuing the wavefield to the topographic surface and back to the flat datum (Split-Step). mihai2-Mig2Datumsplit [ER]

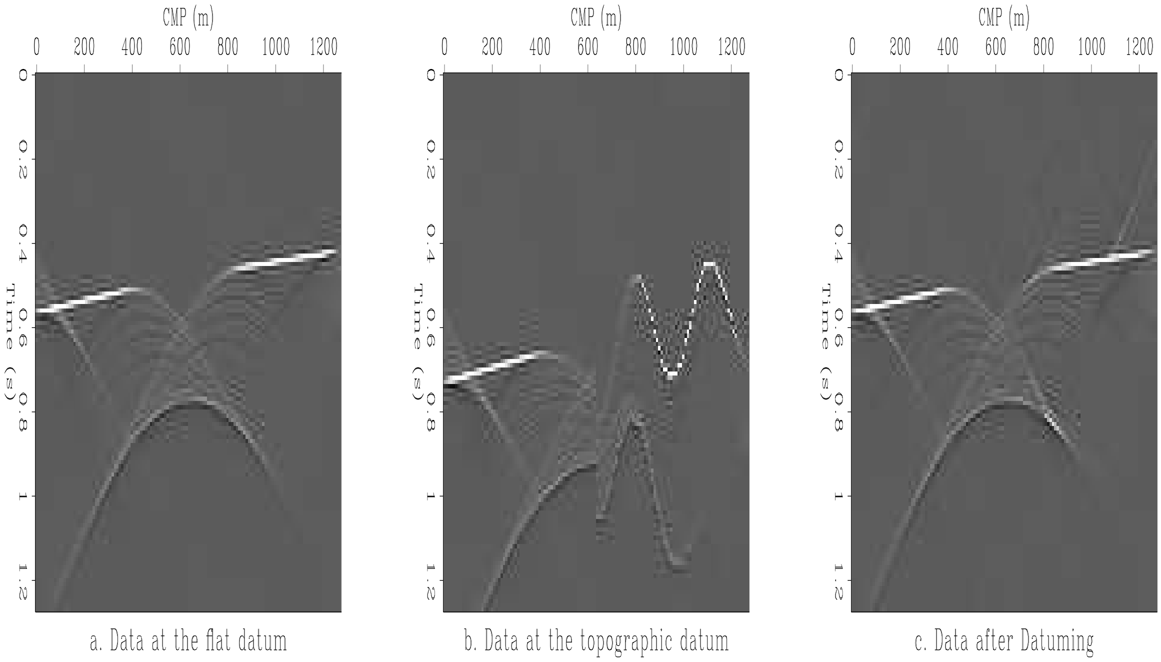
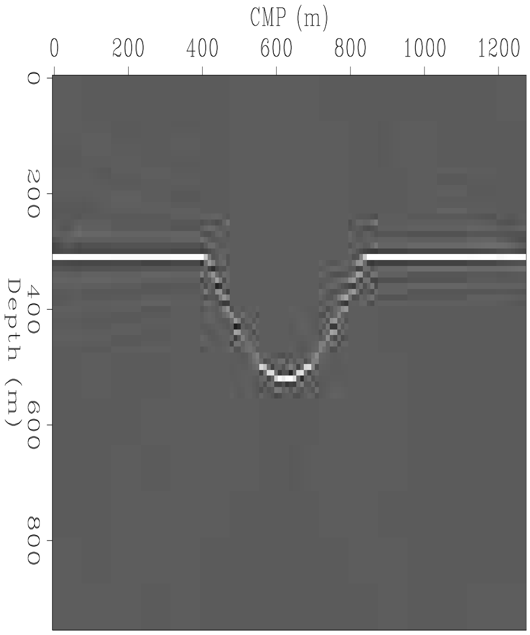
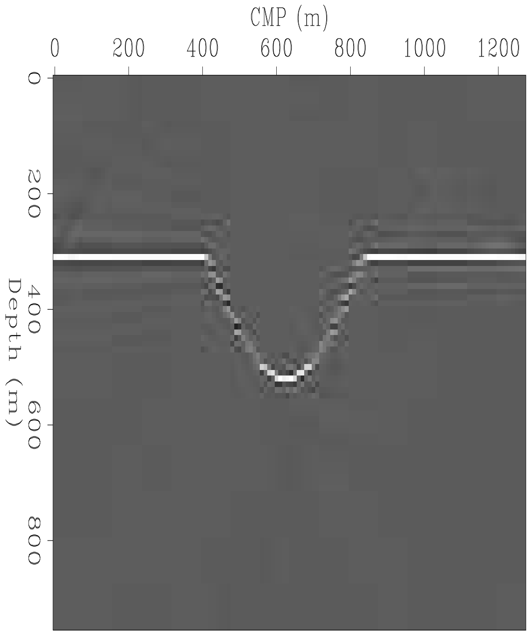


Figure 10: Effect on initial data of the datuming algorithm v(x,z). PSPI datuming.
a. Initial data at the flat surface.
b. Data on the topographic datum.
c. Data after datuming to the topographic surface and back.
mihai2-data3datumpspi [ER]



a. Data Migrated before Datuming



b. Data Migrated after Datuming

Figure 11: Migration before and after datuming. $v(x, z) = v_0 + ax * x + az * z$.
a. Migration before datuming (PSPI).
b. Migration after continuing the wavefield to the topographic surface and back to the flat datum (PSPI). $\boxed{\text{mihai2-Mig2Datumpspi}}$ [ER]

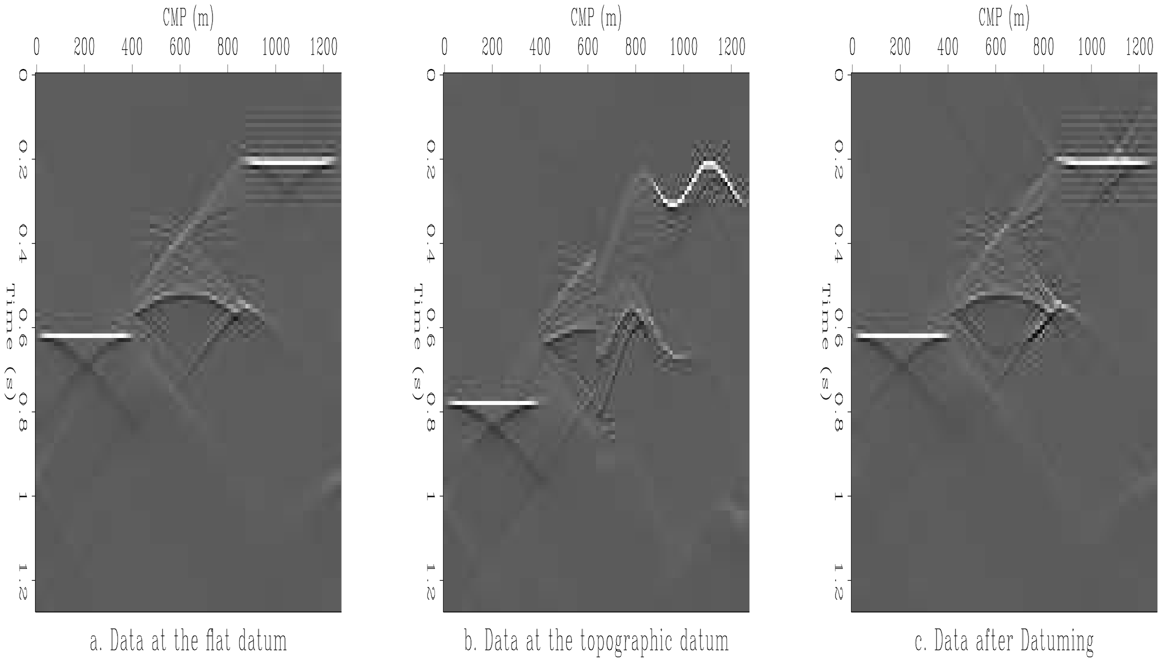
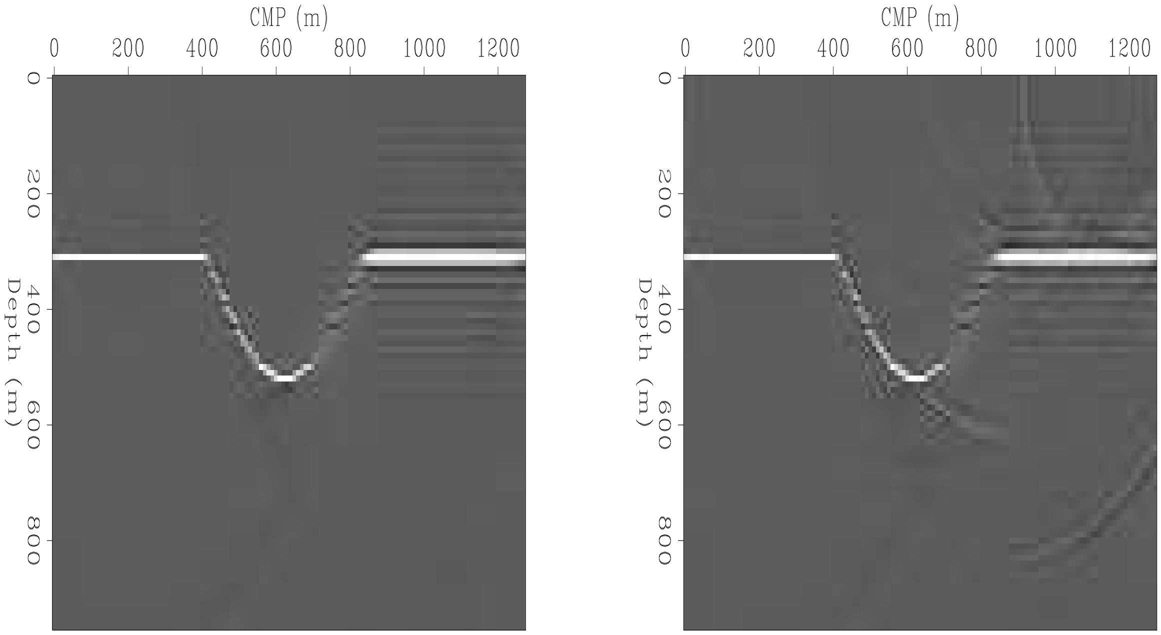


Figure 12: Effect on initial data of the datuming algorithm v(x,z). Split-Step.
a. Initial data at the flat surface.
b. Data on the topographic datum.
c. Data after datuming to the topographic surface and back. mihai2-data3datumxsplit [ER]



a. Data Migrated before Datuming

b. Data Migrated after Datuming

Figure 13: Migration before and after datuming in $v(x, z) = 1000, 2000, 3000$ velocity.
a. Migration before datuming (Split-Step).
b. Migration after continuing the wavefield to the topographic surface and back to the flat datum (Split-Step). mihai2-Mig2Datumxsplit [ER]

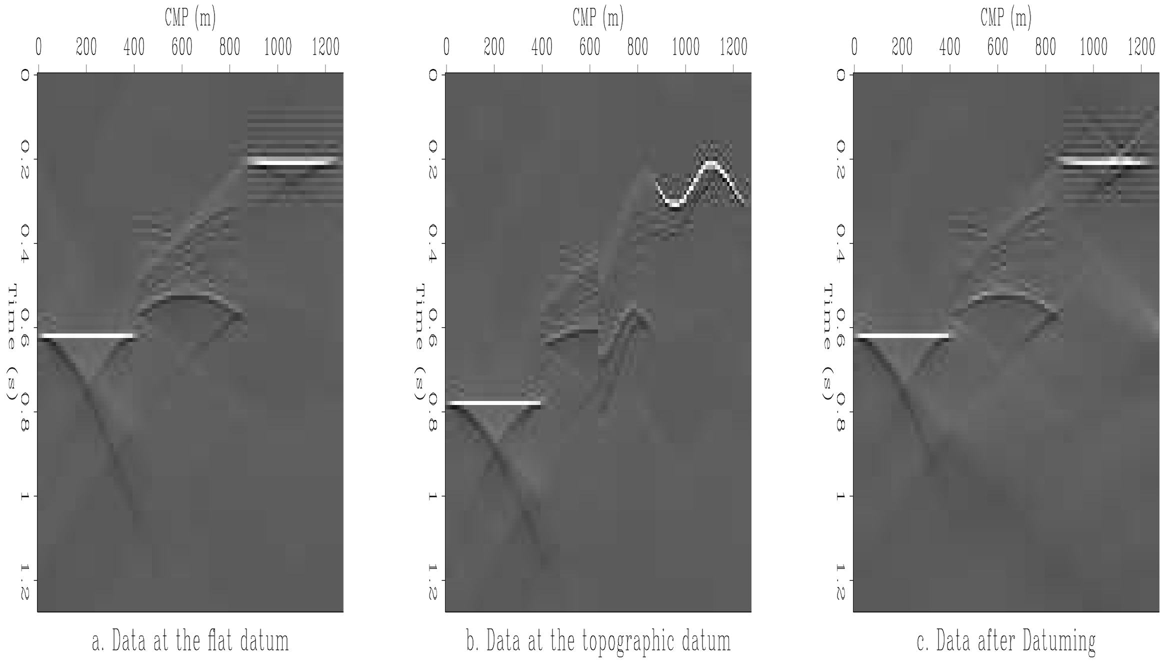


Figure 14: Effect on initial data of the datuming algorithm v(x,z). PSPI datuming.
a. Initial data at the flat surface.
b. Data on the topographic datum.
c. Data after datuming to the topographic surface and back. mihai2-data3datumxpspi [ER]

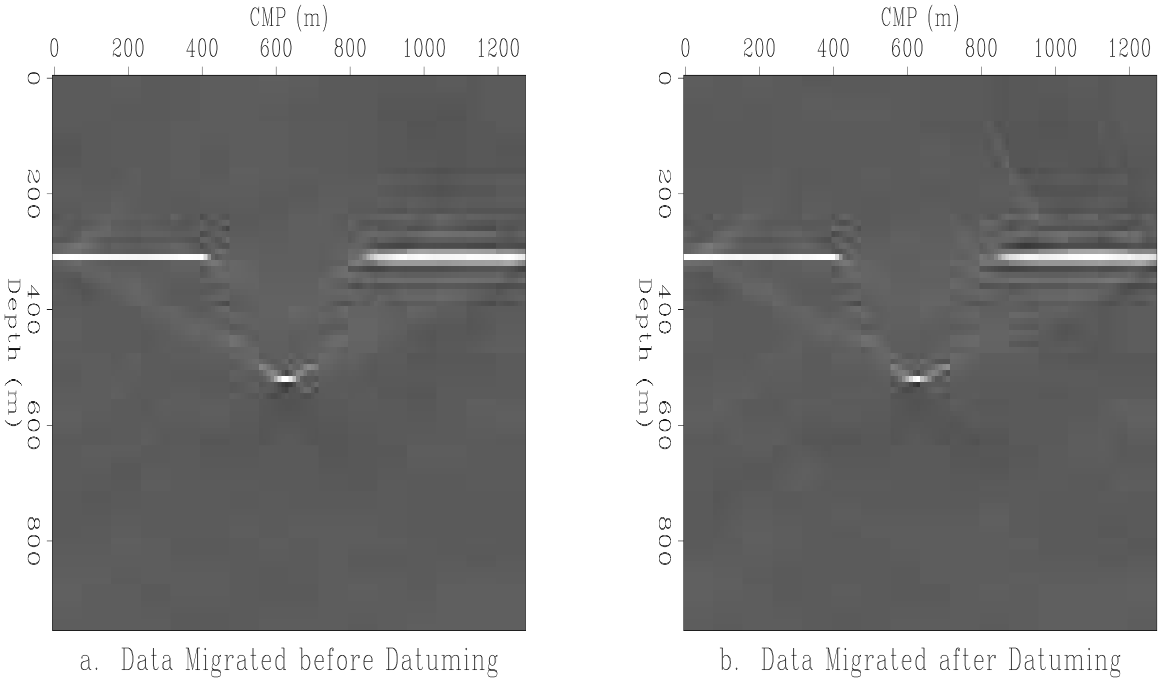


Figure 15: Migration before and after datuming in $v(x, z) = 1000, 2000, 3000$ velocity.
a. Migration before datuming (PSPI).
b. Migration after continuing the wavefield to the topographic surface and back to the flat datum (PSPI). mihai2-Mig2Datumxpspi [ER]

Future work

Berryhill (1984) generalized his earlier work for zero-offset data to constant-offset (prestack) data by downward continuing the geophones and the sources to a datum in a two pass algorithm. The same idea can be used for extrapolating data using different phase-shift algorithms for downward (or upward) continuing common-source and common-receiver sections to a datum. Moreover, due to the inherent simplicity of the algorithm the 3-D implementation is straightforward.

Another research direction was given by Yilmaz and Lucas (1986) and Berryhill (1986) who both describe a velocity replacement technique using a Kirchhoff based datuming algorithm twice. The layer replacement method consists of downward extrapolating the data to a datum using a known velocity and then upward extrapolating the data back to the surface with a different velocity in order to reduce the effect of the velocity contrast at the datum. Using the phase-shift datuming algorithm described in this paper we can apply the same technique to zero-offset data and to prestack data to obtain the same results.

REFERENCES

Berryhill, J. R., 1979, Wave-equation datuming: Geophysics, **44**, 1329–1344.

Berryhill, J. R., 1984, Wave-equation datuming before stack: Geophysics, **49**, 2064–2066.

Berryhill, J. R., 1986, Submarine canyons: Velocity replacement by wave-equation datuming before stack: Geophysics, **51**, 1572–1579.

Bevc, D., 1992, Kirchhoff wave-equation datuming with irregular acquisition topography: SEP–**75**.

Claerbout, J. C., 1985, Imaging the Earth's Interior: Blackwell Scientific Publications.

Claerbout, J. C., 1992, Earth Soundings Analysis: Boston, Blackwell Scientific Publications.

Gazdag, J., 1978, Wave equation migration with the phase shift method: Geophysics, **43**, 1342–1351.

Gazdag, J., and Sguazzero, P., 1984, Migration of seismic data by phase shift plus interpolation: Geophysics, **49**, 124–131.

Ji, J., and Claerbout, J. C., 1992, Phase shift datuming and migration from an irregular surface: SEP–**75**.

Popovici, A. M., 1992, 3-D and 2-D phase shift plus interpolation and split-step Fourier modeling: SEP–**73**, 331–343.

Shtivelman, V., Canning A., 1988, Datum correction by wave-equation extrapolation: Geophysics, **53**, 1311–1322.

Stoffa, P. L., Fokkema, J. T., de Luna Freire, R. M., and Kessinger, W. P., 1990, Split-step Fourier Migration: Geophysics, **55**, 410–421.

Wiggins, J. W., 1984, Kirchhoff integral extrapolation and migration of nonplanar data: Geophysics, **49**, 1239–1248.

Yilmaz, O., Lucas D., 1986, Prestack layer replacement: Geophysics, **51**, 1355–1369.

APPENDIX

The code for conjugate transpose PSPI and Split-Step. Both programs pass the Dot-Product test.

PSPI modeling and migration

```
C#######################################################################
C  File pspi2Dconj.ft
C  Author: Mihai Popovici (Aug 92)
C
C
C  conj=0 MODELING
C      =1 MIGRATION
C#######################################################################
        subroutine pspi2D(conj,nt,nx,nz,dt,dx,dz,data,model,vel)
        integer nt,nx,nz,conj
        real    dt,dx,dz
        complex data(nt,nx)
        complex model(nz,nx)
        real    vel(nz,nx)
        integer it,ix,iz,iw,kx,nw
        real    v(2),dv1,dv2,w0,kx0
        real    dv,rdv,rv,r2v,w,x,dw,dkx,akz
        real    PI,killkz,tscale,signum
        integer iv,nv
        temporary real akx(nx)
        temporary complex Pxyw(nt,nx,2)

        PI  =4.*atan(1.)
        nw  =nt
        dw  =2.*PI/(nt*dt)
        dkx =2.*PI/(nx*dx)
        w0  = -PI/dt
        kx0 = -PI/dx
        tscale=1./sqrt(float(nt))
C######## ######### ######### ######### ######### ######### ######### ##
C Zero the tables conj=0 data=0, conj=1 model=0
        call conjnull(conj,0,model,2*nz*nx,data,2*nt*nx)
C Compute the kx**2 values only once
        do kx=1,nx
          x=kx0+(kx-1)*dkx
          akx(kx)=x*x
        enddo
C######## ######### ######### ######### ######### ######### ######### ##
C If MODELING
      if(conj.eq.0) then

C Spread the Image (when t=0 in exploding reflectors) over the frequency range
C This is equivalent to the FFT in t of an impulse function.
        do ix=1,nx
          do iw=1,nw
```

```
            data(iw,ix)=model(nz,ix)*tscale
          enddo
        enddo
        do iz=nz,2,-1

C Find Min v(1) and Max v(2) velocity
          v(1)=vel(iz,1)
          v(2)=vel(iz,1)
          do ix=2,nx
            if(v(1).gt.vel(iz,ix) ) v(1)=vel(iz,ix)
            if(v(2).lt.vel(iz,ix) ) v(2)=vel(iz,ix)
          enddo
          dv=v(2)-v(1)
C For constant velocity in X use normal algorithm
          if(dv.eq.0.) then
            nv=1
          else
            nv=2
            rdv=1./dv
          endif

C Split with weights in w
C For constant velocity don't bother to Split
          if(nv.le.1) then
            do ix=1,nx
              do iw=1,nw
                Pxyw(iw,ix,1)=data(iw,ix)
              enddo
            enddo
          else
C For variable velocity split with weights in omega (w)
            do ix=1,nx
              dv2=v(2)-vel(iz,ix)
              dv1=vel(iz,ix)-v(1)
              do iw=1,nw
                Pxyw(iw,ix,1)=data(iw,ix)*dv2*rdv
                Pxyw(iw,ix,2)=data(iw,ix)*dv1*rdv
              enddo
            enddo
          endif
C FFT over X
          call fft2X(Pxyw,data,nt,nx,nv,0)

C Start extrapolation loop with two velocities
          do iv=1,nv
            rv=1./v(iv)
            r2v=rv*rv
            do iw=1,nw
              w=w0+(iw-1)*dw
              do kx=1,nx
                akz=w*w*r2v-akx(kx)
                killkz=sign(1.,akz)
                killkz=0.5*(1+killkz)
                akz=signum(w)*sqrt(max(0.,akz))
```

```
                Pxyw(iw,kx,iv)=Pxyw(iw,kx,iv)*
     &          cexp(cmplx(0.,(akz-w*rv)*dz))*killkz
              enddo
            enddo
          enddo

C For constant velocity don't bother to sum
          if(nv.le.1) then
            do ix=1,nx
              do iw=1,nw
                data(iw,ix)=Pxyw(iw,ix,1)
              enddo
            enddo
          else
C For variable velocity sum the wave-fields
            do ix=1,nx
              do iw=1,nw
                data(iw,ix)=Pxyw(iw,ix,1)+Pxyw(iw,ix,2)
              enddo
            enddo
          endif

C Inverse FFT in x
          call ft2axis( 1, -1., nt, nx,data)

C Multiply by exp(iw*dz/v(x,z))
          do iw=1,nw
            w=w0+(iw-1)*dw
            do ix=1,nx
              data(iw,ix)=data(iw,ix)*cexp(cmplx(0.,w*dz/vel(iz,ix)))
              data(iw,ix)=data(iw,ix)+model(iz-1,ix)*tscale
            enddo
          enddo

        enddo
C Inverse FFT in t
        call ft1axis( 1,1.,nt,nx,data)

C######## ######### ######### ######### ######### ######### ######### ##
C If MIGRATION
      else
        call ft1axis( 0,1.,nt,nx,data)
        do ix=1,nx
          do it=1,nt
            model(1,ix)=model(1,ix)+data(it,ix)*tscale
          enddo
        enddo

        do iz=2,nz

C Find Min v(1) and Max v(2) velocity
          v(1)=vel(iz,1)
          v(2)=vel(iz,1)
          do ix=2,nx
```

```
            if(v(1).gt.vel(iz,ix) ) v(1)=vel(iz,ix)
            if(v(2).lt.vel(iz,ix) ) v(2)=vel(iz,ix)
          enddo
          dv=v(2)-v(1)
C For constant velocity in X use normal algorithm
          if(dv.eq.0.) then
            nv=1
          else
            nv=2
            rdv=1./dv
          endif

C Multiply by exp(iw*dz/v(x,z))
          do iw=1,nw
            w=w0+(iw-1)*dw
            do ix=1,nx
              data(iw,ix)=data(iw,ix)*cexp(cmplx(0.,-w*dz/vel(iz,ix)))
            enddo
          enddo

C FFT in x
          call ft2axis( 0, -1., nt, nx,data)

C Start extrapolation loop with two velocities
          do iv=1,nv
            rv=1./v(iv)
            r2v=rv*rv
            do iw=1,nw
              w=w0+(iw-1)*dw
              do kx=1,nx
                akz=w*w*r2v-akx(kx)
                killkz=sign(1.,akz)
                killkz=0.5*(1+killkz)
                akz=-signum(w)*sqrt(max(0.,akz))
                Pxyw(iw,kx,iv)=data(iw,kx)*
     &          cexp(cmplx(0.,(akz+w*rv)*dz))*killkz
              enddo
            enddo
          enddo
C Inverse FFT in x
          call fft2X(Pxyw,data,nt,nx,nv,1)

C Interpolate in w
C For constant velocity don't bother to interpolate
          if(nv.le.1) then
            do ix=1,nx
              do iw=1,nw
                data(iw,ix)=Pxyw(iw,ix,1)
              enddo
            enddo
          else
C For variable velocity interpolate in omega (w)
            do ix=1,nx
              dv2=v(2)-vel(iz,ix)
```

```
            dv1=vel(iz,ix)-v(1)
            do iw=1,nw
              data(iw,ix)=(Pxyw(iw,ix,1)*dv2+Pxyw(iw,ix,2)*dv1)*rdv
            enddo
          enddo
        endif

C Image for t=0.
C For t=0 all the exponents in the Fourier summation are 1.
        do ix=1,nx
          do iw=1,nw
            model(iz,ix)=model(iz,ix)+data(iw,ix)*tscale
          enddo
        enddo

      enddo
    endif
    return
    end

C######## ######### ######### ######### ######### ######### ######### ##
      subroutine fft2X(Pxyw,data,nt,nx,nv,signft)
      integer nx,nt,nv,signft
      complex Pxyw(nt,nx,2),data(nt,nx)
      integer iv,ix,it

      do iv=1,nv
        do ix=1,nx
          do it=1,nt
            data(it,ix)=Pxyw(it,ix,iv)
          enddo
        enddo
C FFT in x
        call ft2axis( signft,-1.,nt,nx,data)
        do ix=1,nx
          do it=1,nt
            Pxyw(it,ix,iv)=data(it,ix)
          enddo
        enddo
      enddo
      return
      end
```

Spit-Step modeling and migration

```
C######################################################################
C  File Split2Dconj.ft
C  Author: Mihai Popovici (July 92)
C
C
C  conj=0 MODELING
C      =1 MIGRATION
C######################################################################
        subroutine split2D(conj,nt,nx,nz,dt,dx,dz,data,model,vel)
        integer nt,nx,nz,conj
        real    dt,dx,dz
        complex data(nt,nx)
        complex model(nz,nx)
        real    vel(nz,nx)
        integer it,ix,iz,iw,kx,nw
        complex phase
        real    velmed,slowmed,r2v
        real    w,x,dw,dkx,rdz,akz
        real    w0,kx0
        real    PI,killkz,tscale
        real    signum
        integer esize
        temporary real akx(nx), sdiff(nx)

        esize=4
        PI=4.*atan(1.)
        nw=nt
        dw=2.*PI/(nt*dt)
        dkx=2.*PI/(nx*dx)
        w0  = -PI/dt
        kx0 = -PI/dx

        rdz=1./dz
        tscale=1./sqrt(float(nt))
C######### ######### ######### ######### ######### ######### ######### ##
C Zero the tables conj=0 data=0, conj=1 model=0
        call conjnull(conj,0,model,2*nz*nx,data,2*nt*nx)
C Compute the kx**2 values only once
        do kx=1,nx
          x=kx0+(kx-1)*dkx
          akx(kx)=x*x
        enddo

C######### ######### ######### ######### ######### ######### ######### ##
C If MODELING
      if(conj.eq.0) then

C Spread the Image (when t=0 in exploding reflectors) over the frequency range
C This is equivalent to the FFT in t of an impulse function.
        do ix=1,nx
          do iw=1,nw
            data(iw,ix)=model(nz,ix)*tscale
```

```
          enddo
        enddo
        do iz=nz,2,-1
          velmed=0.
C Find average velocity
          do ix=1,nx
            velmed=vel(iz,ix)/nx+velmed
          enddo
          slowmed=1./velmed
          r2v=slowmed*slowmed
C Find deviation from average slowness
          do ix=1,nx
            sdiff(ix)=slowmed-1./vel(iz,ix)
          enddo
          do iw=1,nw
            w=w0+(iw-1)*dw
C Apply phase shift due to perturbation in slowness.
            do ix=1,nx
              phase=cexp(cmplx(0.,-w*sdiff(ix)*dz))
              data(iw,ix)=data(iw,ix)*phase
            enddo
          enddo

C FFT over X
          call ft2axis(0,-1.,nt,nx,data)

C Start extrapolation loop
          do iw=1,nw
            w=w0+(iw-1)*dw
            do kx=1,nx
              akz=w*w*r2v-akx(kx)
              killkz=sign(1.,akz)
              killkz=0.5*(1+killkz)
              akz=signum(w)*sqrt(max(0.,akz))
              data(iw,kx)=data(iw,kx)*cexp(cmplx(0.,akz*dz))*killkz
            enddo
          enddo

C Inverse FFT in x
          call ft2axis( 1,-1.,nt,nx,data)

          do ix=1,nx
            do iw=1,nw
              data(iw,ix)=data(iw,ix)+model(iz-1,ix)*tscale
            enddo
          enddo
        enddo
C Inverse FFT in t
        call ft1axis( 1,1.,nt,nx,data)

C######## ######### ######### ######### ######### ######### ######### ##
C If MIGRATION
      else
        call ft1axis( 0,  1., nt, nx, data)
```

```
        do ix=1,nx
          do it=1,nt
            model(1,ix)=model(1,ix)+data(it,ix)*tscale
          enddo
        enddo
        do iz=2,nz
C Find average velocity
          velmed=0.
          do ix=1,nx
            velmed=vel(iz,ix)/nx+velmed
          enddo
          slowmed=1./velmed
          r2v=slowmed*slowmed
C Find deviation from average slowness
          do ix=1,nx
            sdiff(ix)=slowmed-1./vel(iz,ix)
          enddo
C FFT in x
          call ft2axis( 0, -1., nt, nx,data)

C Start extrapolation loop
          do iw=1,nw
            w=w0+(iw-1)*dw
            do kx=1,nx
              akz=w*w*r2v-akx(kx)
              killkz=sign(1.,akz)
              killkz=0.5*(1+killkz)
              akz=-signum(w)*sqrt(max(0.,akz))
              data(iw,kx)=data(iw,kx)*cexp(cmplx(0.,akz*dz))*killkz
            enddo
          enddo
C Inverse FFT in x
          call ft2axis( 1, -1., nt, nx,data)

C Apply phase shift due to perturbation in slowness
          do ix=1,nx
            do iw=1,nw
              w=w0+(iw-1)*dw
              phase=cexp(cmplx(0.,w*sdiff(ix)*dz))
              data(iw,ix)=data(iw,ix)*phase
C Image for t=0.
C For t=0 all the exponents in the Fourier summation are 1.
              model(iz,ix)=model(iz,ix)+data(iw,ix)*tscale
            enddo
          enddo

        enddo
      endif
      return
      end
```