Reverse-time migration as the conjugate of forward modeling

Jun Ji [1]

ABSTRACT

This paper presents a reverse-time migration algorithm as the conjugate of the forward time-extrapolation modeling. The conjugate operator is obtained by formulating the forward operator explicitly and by transposing the forward operator. The algorithm is tested by a numerical experiment for a 1-D layered model. A method to suppress artifacts, which are caused by the intrinsic non-unitary property of the operator, is introduced using the least-squares optimization technique.

INTRODUCTION

Reverse-time migration (McMechan, 1983) has been widely used in seismic exploration imaging because of its ability to accurately implement the two-way wave equation. The most popular way to implement reverse time migration is time-extrapolation from the maximum time to the minimum time with the recorded wave field as boundary condition. This algorithm does not pass Claerbout's dot-product test (Claerbout, 1992), which means that this algorithm is not the conjugate of the time-extrapolation modeling algorithm. This paper describes the implementation of a reverse-time migration algorithm as an exact conjugate operator of the forward time-extrapolation modeling.

1-D TIME EXTRAPOLATION AND ITS CONJUGATE

In order to find a conjugate operator, the forward modeling operator must be formulated explicitly. The acoustic wave equation in the one dimensional space is

$$\frac{\partial^2 P}{\partial t^2} = v^2 \frac{\partial^2 P}{\partial z^2}, \tag{1}$$

[1] **email:** Author has no known email address.

where P is pressure field, v is the velocity, t and z are time and depth axis, respectively.

Approximating the derivatives by finite differences, equation (1) becomes

$$\frac{1}{\Delta t^2}(P_{i-1,j} - 2P_{i,j} + P_{i+1,j}) = \frac{v_j^2}{\Delta z^2}(P_{i,j-1} - 2P_{i,j} + P_{i,j+1}), \tag{2}$$

where i and j represent time and depth indices, respectively. By rearranging equation (2) for time-extrapolation, we get the following equation:

$$P_{i+1,j} = 2(1-\alpha_j)P_{i,j} + \alpha_j(P_{i,j-1} + P_{i,j+1}) - P_{i-1,j} \tag{3}$$

$$\alpha_j = \frac{v_j^2 \Delta t^2}{\Delta z^2}$$

The differencing star in equation (3) covers the (t, z) space along the z-axis to extrapolate in time as shown schematically by Figure 1.

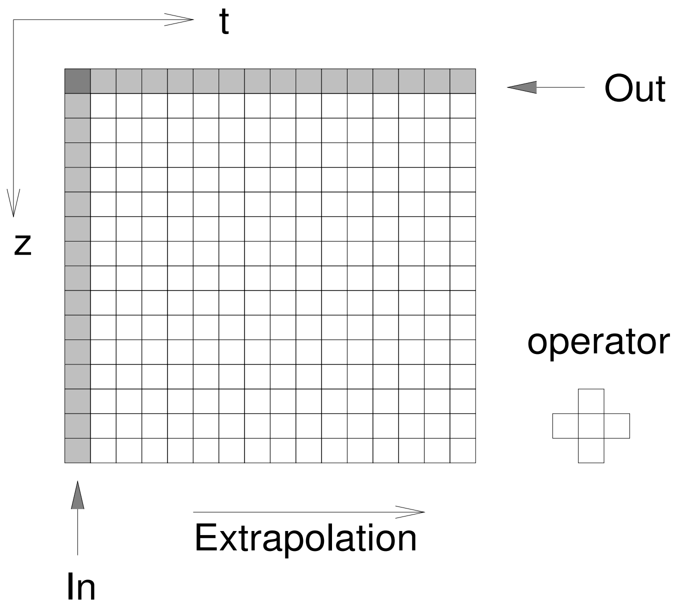

Figure 1: Schematic for time extrapolation modeling for 1-D acoustic wave equation. The input is a source function at $t = 0$ and the output is a trace at $z = 0$. (rt1dmdl) [NR]

To make the operator small enough to show on a page, I choose a small space like $(t, z) = (4, 5)$. The time extrapolation shown in equation (3) can be formulated algebraically as follows

$$\begin{bmatrix} \mathbf{S_1} & \mathbf{S_2} & \mathbf{S_3} & \mathbf{S_4} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} & \mathbf{T} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ -\mathbf{I} & \mathbf{T} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{T} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{m} \end{bmatrix} = \begin{bmatrix} \mathbf{d} \end{bmatrix}, \tag{4}$$

where $\mathbf{m}$ is the model vector at $t = 1$ with length 5, $\mathbf{d}$ is the data vector with length 4, and $\mathbf{I}$ and $\mathbf{0}$ represent 5 by 5 identity and zero matrices. The matrix $\mathbf{T}$ in equation (4) represents tridiagonal matrix which looks like:

$$\begin{bmatrix} 2(1-\alpha_1) & \alpha_1 & 0 & 0 & 0 \\ \alpha_2 & 2(1-\alpha_2) & \alpha_2 & 0 & 0 \\ 0 & \alpha_3 & 2(1-\alpha_3) & \alpha_3 & 0 \\ 0 & 0 & \alpha_4 & 2(1-\alpha_4) & \alpha_4 \\ 0 & 0 & 0 & \alpha_5 & 2(1-\alpha_5) \end{bmatrix}, \tag{5}$$

where α_j was shown in equation (3). The matrix $\mathbf{S}_i$ samples the wave field on the surface at i-th time and has the following form:

$$\mathbf{S_1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{S_2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \ldots. \tag{6}$$

Now, reverse-time migration as the conjugate operator to the forward time-extrapolation becomes clear. It is obtained by transposing the operator in equation (4) and has the following form :

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{T}^* & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{T}^* & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{T}^* \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{S_1^*} \\ \mathbf{S_2^*} \\ \mathbf{S_3^*} \\ \mathbf{S_4^*} \end{bmatrix} \begin{bmatrix} \mathbf{d} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{m}} \end{bmatrix} \tag{7}$$

where $\mathbf{T}^*$ and $\mathbf{S}_i^*$ are transpose matrices of $\mathbf{T}$ and $\mathbf{S}_i$. It is interesting that the conjugate algorithm follows the reverse-time extrapolation but with an operator which differs from the forward-time extrapolation. Instead of using wave fields from the two previous time steps together for an extrapolation, the conjugate operator uses the wave fields of the two previous time steps separately. The code for this algorithm, which has passed the dot-product test (Claerbout, 1992), is listed in the Appendix.

Example

In this example the conjugate algorithm is tested for a simple layered model. Figure 2 shows the velocity model and the source function at time equals zero. The result of the forward time-extrapolation is shown in Figure 3, on the top of the snap shots. It contains the direct arrival, primary reflection, ghost and multiple reflection events. In order to avoid the inclusion of the boundary reflections, time extrapolation was performed before the arrival time of the boundary reflections (see the snap shots in Figure 3). Figure 4 shows the result of the reverse-time migration using the algorithm presented in this paper. It can be seen that the conjugate operator does a good job. The backprojected source wave field recovered closely matching the original source function except for some artifacts. These artifacts are mainly caused

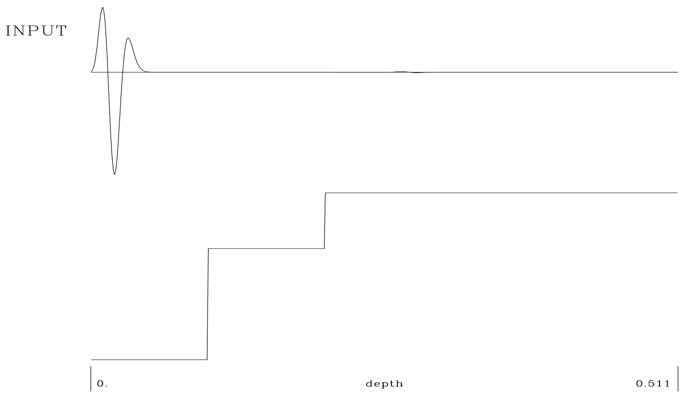

Figure 2: Input source wave field at time equals zero and velocity function used for the model. (model) [NR]

by the wave fields propagated downward, and can be suppressed using a least-squares optimization technique (Ji, 1992) by solving an objective function as follows:

$$\min_{\mathbf{m}} ||\mathbf{d} - \mathbf{W}\mathbf{m}||^2 \tag{8}$$

where $\mathbf{W}$ represents the forward time-extrapolation operator in equation (4), and $\mathbf{d}$ and $\mathbf{m}$ are the wave fields at depth equals zero and at time equals zero, respectively. The result of the least-squares inversion is obtained after 10 iterations using the conjugate gradient algorithm and is shown in Figure 5 along with the original wave field and the wave field obtained by the conjugate operator. All the artifacts are well reduced in the least-squares inversion result.

2-D TIME EXTRAPOLATION AND ITS CONJUGATE

The acoustic wave equation in two dimensional space is

$$\frac{\partial^2 P}{\partial t^2} = v^2 \left(\frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial z^2} \right) \tag{9}$$

where x and z are the surface and depth axis, respectively. By approximating the derivatives by finite differences, equation (8) becomes

$$\frac{1}{\Delta t^2}(P_{i-1,j,k} - 2P_{i,j,k} + P_{i+1,j,k})$$

$$= \frac{v_{j,k}^2}{\Delta z^2}(P_{i,j-1,k} - 2P_{i,j,k} + P_{i,j+1,k}) + \frac{v_{j,k}^2}{\Delta x^2}(P_{i,j,k-1} - 2P_{i,j,k} + P_{i,j,k+1}) \tag{10}$$

where i, j and k represent the time, depth and surface indices, respectively. By rearranging for time-extrapolation, we get the following equation

$$P_{i+1,j,k} = 2(1 - 2\alpha_{j,k})P_{i,j,k} + \alpha_{j,k}(P_{i,j-1,k} + P_{i,j+1,k} + P_{i,j,k-1} + P_{i,j,k+1}) - P_{i-1,j,k} \tag{11}$$

$$\alpha_{j,k} = \frac{v_{j,k}^2 \Delta t^2}{\Delta z^2}$$

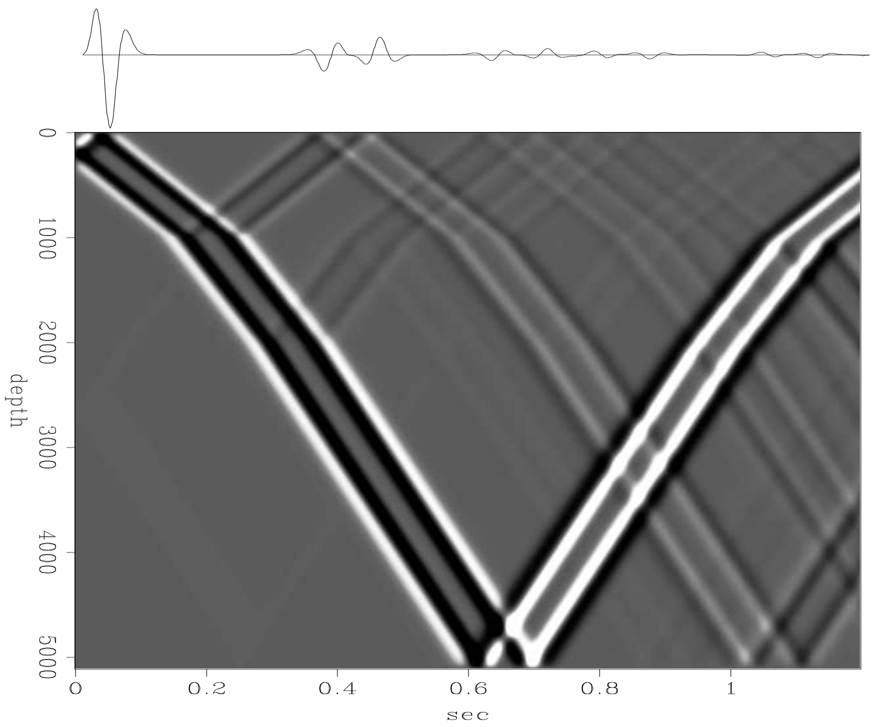

Figure 3: Forward time extrapolation experiment: The time trace along with snap shots. $\boxed{\text{frd}}$ [NR]

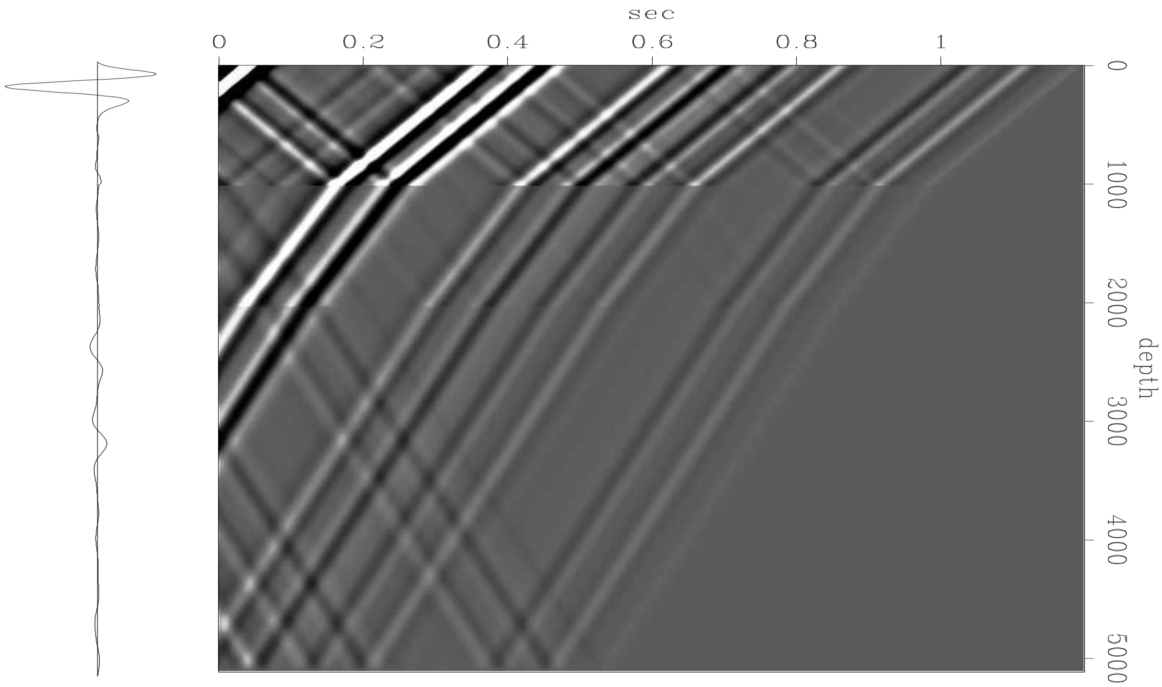

Figure 4: Reverse time migration using the algorithm shown in equation (2): The depth section along with snap shots. $\boxed{\text{rt}}$ [NR]

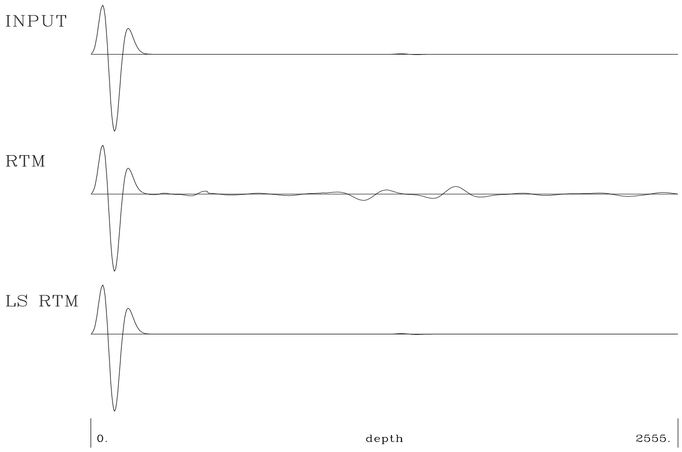

Figure 5: Comparison of wave fields: Top is the original wave field. Middle is the result of conjugate operator. Bottom is the result of the least-squares inversion (after 10 iteration). rtinv [NR]

where I assumed $\Delta x = \Delta z$.

The matrix representation of this operator is the same as equation(4) except that the size of each matrix is now larger than the corresponding matrix in equation(4) because I need to express the two-dimensional wavefield with an abstract vector form. For a space of $(z, x) = (5, 6)$, the matrix $\mathbf{T}$ becomes a block tridiagonal matrix as follows :

$$\mathbf{T} = \begin{bmatrix} \mathbf{R_1} & \mathbf{A} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A} & \mathbf{R_2} & \mathbf{A} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} & \mathbf{R_3} & \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A} & \mathbf{R_4} & \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A} & \mathbf{R_5} & \mathbf{A} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A} & \mathbf{R_6} \end{bmatrix}, \tag{12}$$

where $\mathbf{A}$ is a diagonal matrix having the following elements

$$\alpha_{j,k} = \frac{v_{j,k}^2 \Delta t^2}{\Delta z^2}.$$

The matrix $\mathbf{R_k}$ represents a tri-diagonal matrix :

$$\mathbf{R}_k = \begin{bmatrix} 2(1-2\alpha_{1,k}) & \alpha_{1,k} & 0 & 0 & 0 \\ \alpha_{2,k} & 2(1-2\alpha_{2,k}) & \alpha_{2,k} & 0 & 0 \\ 0 & \alpha_{3,k} & 2(1-2\alpha_{3,k}) & \alpha_{3,k} & 0 \\ 0 & 0 & \alpha_{4,k} & 2(1-2\alpha_{4,k}) & \alpha_{4,k} \\ 0 & 0 & 0 & \alpha_{5,k} & 2(1-2\alpha_{5,k}) \end{bmatrix} \tag{13}$$

The conjugate operator can easily be obtained by substitution of the transposed form of the new matrix $\mathbf{T}$ into the equation(7). The code for 2-D algorithm, which has passed dot-product test, can be found in the CD-rom version of this report.

CONCLUSIONS

A reverse-time migration algorithm expressed as the conjugate of forward time-extrapolation modeling is presented for both 1-D and 2-D geometries. The conjugate operator is obtained by formulating the forward operator explicitly and transposing the forward operator. The algorithm is tested by a numerical experiment for a 1-D layered model. A method to suppress artifacts, which are caused by the intrinsic non-unitary property of the operator, is introduced using a least-squares optimization technique.

REFERENCES

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

Ji, Jun, 1992, Least squares imaging, datuming, and interpolation using the wave equation: SEP–**75**, 121–134

McMechan, G. A., 1983, Migration by extrapolation of time-dependent boundary values: Geophys. Prosp., **31**, 413–420.

APPENDIX A

```
#     1D time extrapolation modeling ( conj=0 )
#     and reverse time migration ( conj=1 )
#
subroutine rt1d( conj, dt,dz, nt,nz, modl,       data,     v)
integer          conj,        nt,nz,                                   it, iz
real                                   modl(nz), data(nt), v(nz)
real                   dt,dz, dtz, a, a1,a2,a3
temporary real                        tmodl(nz,3)

call conjnull( conj, 0, modl,nz, data,nt )
call zero(     3*nz,   tmodl)

if (conj==0) {
                do iz = 1, nz { tmodl(iz,1) = 0.
                                tmodl(iz,2) = modl(iz) }
                data(1) = modl(1)
        do it = 2, nt {
        do iz = 2, nz-1 {
                a = (v(iz)*dt/dz)**2.
                tmodl(iz,3) = a*(tmodl(iz+1,2) - 2.*tmodl(iz,2)+tmodl(iz-1,2))_
                           -    tmodl(iz  ,1) + 2.*tmodl(iz,2)
                       }
                a = (v(1)*dt/dz)**2.
                tmodl(1,3) = a*(tmodl(2,2) - 2.*tmodl(1,2))_
                           -   tmodl(1,1) + 2.*tmodl(1,2)
                a = (v(nz)*dt/dz)**2.
                tmodl(nz,3) = a*(tmodl(nz-1,2) - 2.*tmodl(nz,2))_
                           -    tmodl(nz  ,1) + 2.*tmodl(nz,2)
                data(it) = tmodl(1,3)
                do iz = 1, nz { tmodl(iz,1) = tmodl(iz,2)
                                tmodl(iz,2) = tmodl(iz,3) }
              }
        }
else    {
                dtz= (dt/dz)**2.
                tmodl(1,1) = data(nt);  tmodl(1,2) = data(nt-1)
                                        tmodl(1,3) = data(nt-2)
```

```
        do it = nt-1, 1, -1 {
        do iz = 1, nz
                tmodl(iz,3) = tmodl(iz,3) - tmodl(iz,1)
        do iz = 2, nz-1{
                a1 = v(iz-1)*v(iz-1)*dtz; a2 = v(iz)*v(iz)*dtz
                a3 = v(iz+1)*v(iz+1)*dtz
         tmodl(iz,2) = a1*tmodl(iz-1,1) - 2.*a2*tmodl(iz,1) + a3*tmodl(iz+1,1)_
                     + 2.*tmodl(iz  ,1) +       tmodl(iz,2)
                        }
                a2 = v(1)*v(1)*dtz;      a3 = v(2)*v(2)*dtz
         tmodl( 1,2) = a3*tmodl(2,1) - 2.*a2*tmodl(1,1)_
                     + 2.*tmodl(1,1) +       tmodl(1,2)
                a1 = v(nz-1)*v(nz-1)*dtz; a2 = v(nz)*v(nz)*dtz
         tmodl(nz,2) = a1*tmodl(nz-1,1) - 2.*a2*tmodl(nz,1)_
                     + 2.*tmodl(nz  ,1) +       tmodl(nz,2)

        do iz = 1, nz { tmodl(iz,1) = tmodl(iz,2)
                        tmodl(iz,2) = tmodl(iz,3) }

        if( it > 2 )  { do iz = 1, nz
                                tmodl(iz,3) = 0.
                                tmodl( 1,3) = data(it-2) }
                }
        do iz = 1, nz { modl(iz) = tmodl(iz,1) }
        }

return; end
```