1D and 1.5D internal multiple prediction in MatLab

Matthew Eaid, Jian Sun, Scott Keating and Kris Innanen

ABSTRACT

An inverse scattering series approach to internal multiple prediction was developed by Weglein et al. in the late 1990's. Their method exploited the idea that all multiples can be constructed from a combination of primary events and other multiples, in a fully data driven manner (Weglein et al., 1997). Innanen (2015) presented the mathematics behind the inverse scattering series approach in the time domain. Sun and Innanen (2014) show the prediction algorithm in the planewave domain, while Pan and Innanen (2013) explore the prediction in the frequency-wavenumber domain. This paper is meant as a companion to the 2016 CREWES MatLab toolbox release, it will summarize the key ideas behind an inverse scattering series approach to internal multiple prediction in each domain listed above. The implementation of each algorithm will be reviewed and synthetic examples will be provided. Adaptive subtraction is also reviewed with synthetic examples.

INTRODUCTION

In the late 1990's Weglein et al. presented a fully data driven method for attenuating internal multiples. Their idea focuses around the fact that the traveltime of any multiple is simply a combination of traveltimes from the primaries that make up the multiple. The original algorithm derived by Weglein et al., (1997) predicted multiples in the frequency-wavenumber domain. Their algorithm searches for combinations of subevents containing reflections that obey a deeper-shallower-deeper relationship and then combines these events to predict internal multiples.

Although it is required that the algorithm searches for subevents in time or depth, the domains in which the prediction occurs can vary widely. In recent years it has been a budding research interest of CREWES to explore internal multiple prediction in various domains to optimize the environment in which multiples are predicted.

Pan and Innanen (2013) explored internal multiple prediction in the frequency-wavenumber domain. Sun and Innanen (2014) adapted the algorithm for the planewave domain, while Innanen (2015) examined a time domain version of internal multiple prediction. As will be discussed later, while internal multiple prediction is well tuned to predicting the traveltime of multiples, the raw predictions contain phase and amplitude errors. Keating et al., (2015) presented a method of adaptive subtraction to deal with these errors. Their method creates a filter based on the optimization of a hybrid L_1/L_2 norm, when this filter is convolved with the internal multiple trace the phase and amplitude errors are corrected for in a least squares approach.

This paper will begin with a review of the algorithm presented by Weglein et al., and show how it can be reduced to the 1.5D wavenumber-frequency and 1D frequency domain versions of the algorithm. Adaption of the algorithm to other domains will also be discussed. Finally, a review of adaptive subtraction will be given, along with a discussion of how to implement each in MatLab.

REVIEW: INTERNAL MULTIPLES PREDICTION

The inverse scattering series approach at its core is a data driven method of combining subevents in the data in a such a way that internal multiples are predicted. In one dimension it is easy to show that any first order internal multiple will arrive at the same time as the sum of the traveltimes of two primaries, minus the traveltime of a third primary. By combining traveltimes in this way, internal multiples can be predicted.

In two dimensions the prediction algorithm (Weglein et al., 1997) is:

$$b_3(k_g,k_s,\omega)=\frac{1}{(2\pi)^2}\iint_{-\infty}^{\infty}dk_1e^{-iq_1(\epsilon_g-\epsilon_s)}dk_2e^{iq_2(\epsilon_g-\epsilon_s)}\times\varphi \tag{1}$$

where

$$\varphi(k_g,k_1,k_2,k_s|\epsilon)=\int_{-\infty}^{\infty}dz\,e^{i(q_g+q_1)z}b_1(k_g,-k_1,z)$$
$$\times\int_{-\infty}^{z-\epsilon}dz'\,e^{-i(q_1+q_2)z'}b_1(k_1,-k_2,z')\int_{z'+\epsilon}^{\infty}dz''\,e^{i(q_2+q_s)z''}b_1(k_2,-k_s,z'') \tag{2}$$

and where

$$q_x=\frac{\omega}{c_o}\sqrt{1-\frac{k_x^2c_o^2}{\omega^2}} \tag{3}$$

Equation (3) represents the lateral wavenumbers pertaining to the vertical wavenumbers and the reference velocity, c_o.

Preparation of data

Equation (2) shows that the inputs to the integration is a prepared form of the data and not the acquired data itself. In the case of the wavenumber-frequency prediction, the data is prepared in the following manner. First, the data is Fourier transformed over all three variables.

$$d(x_g,x_s,t)\rightarrow D(k_g,k_s,\omega) \tag{4}$$

Next, a change of variables is made from ω to k_z.

$$D(k_g,k_s,\omega)\rightarrow D(k_g,k_s,k_z) \tag{5}$$

where $k_z=q_g+q_s$ and

$$q_g=\frac{\omega}{c_o}\sqrt{1-\frac{k_g^2c_o^2}{\omega^2}},\ q_s=\frac{\omega}{c_o}\sqrt{1-\frac{k_s^2c_o^2}{\omega^2}} \tag{6}$$

the data is then scaled by an obliquity factor (Weglein et al., 2003).

$$B_1(k_g,k_s,k_z)=-i2\,q_s\,D(k_g,k_s,k_z) \tag{7}$$

Finally, the data is inverse Fourier transformed over the k_z variable

$$B_1(k_g, k_s, k_z) \rightarrow b_1(k_g, k_s, \omega) \tag{8}$$

Equation (8) represents the input to equation (2), where the data has been prepared for use in the wavenumber-frequency domain prediction. While the data is prepared in a different manner for each prediction domain, the data preparation steps are fairly similar in each domain. A brief description of data preparation will be given for each algorithm.

Reduction to 1D and 1.5D

The original algorithm presented by Weglein et al., reduces to its one dimensional form when:

$$k_g = k_s = 0 \tag{9}$$

Equation (9) represents a condition of normal incidence, equation (1) then reduces to,

$$b_3(\omega) = \int_{-\infty}^{\infty} dz\, e^{i2\frac{\omega}{c_o}z} b_1(z) \int_{-\infty}^{z-\epsilon} dz'\, e^{-i2\frac{\omega}{c_o}z'} b_1(z') \times \int_{z'+\epsilon}^{\infty} dz''\, e^{i2\frac{\omega}{c_o}z''} b_1(z'') \tag{10}$$

Comparing (10) to (1) it is obvious that the computational difficulty has greatly decreased. In (1) $\varphi(k_g, k_1, k_2, k_s|\epsilon)$ is computed for every k_1, k_2 and summed, then this is repeated for every pair of k_g and k_s. In (10) the integration is carried out for every depth and then repeated for every frequency. Equation (10) represents the 1D frequency form of the inverse scattering internal multiple algorithm.

Equation (1) reduces to a 1.5D algorithm when,

$$k_g = k_s \tag{11}$$

Equation (11) represents the case of horizontally layered strata. When the condition of (11) is met, then (1) becomes,

$$b_3(k_g, \omega) = \int_{-\infty}^{\infty} dz\, e^{ik_z z} b_1(k_g, z) \int_{-\infty}^{z-\epsilon} dz'\, e^{-ik_z z'} b_1(k_g, z') \times \int_{z'+\epsilon}^{\infty} dz''\, e^{ik_z z''} b_1(k_g, z'') \tag{12}$$

Equation (12) is similar to equation (10), however, it is repeated for every value of k_g (trace).

1D FREQUENCY DOMAIN PREDICTION IN MATLAB

Input Data

For 1D internal multiple prediction it is assumed that a single normal incidence trace is available from a one-dimensional earth. It is important when performing internal multiple prediction that the data has been deghosted, that free surface multiples (FSM) have been removed and that the direct arrivals have been muted. Since internal multiple prediction is achieved by combining the traveltimes of subevents, if the data is not prepared in this way, then the algorithm will predict artifacts arriving with traveltimes that are combinations of

the traveltime of the undesired events (ghosts, FSM, and direct arrivals), and the internal multiples and primaries.

Preparation of input data

The preparation of data for the 1D frequency domain prediction is quite trivial. The first step is to Fourier transform the normal incidence trace.

$$d(x_g, t) = D(k_g, \omega) \tag{13}$$

However, in the case of normal incidence $k_g = 0$ and (13) reduces to $D(\omega)$. The conversion is then made from frequency to wavenumber.

$$k_z = \frac{2\omega}{c_0} \tag{14}$$

Time must also be converted to psuedodepth in order to acquire the Fourier conjugate to k_z.

$$z = \frac{c_0 t}{2} \tag{15}$$

where c_0 is the reference velocity, which is 1500 m/s for marine surveys, and should be taken as the velocity around the geophones in land applications.

Prediction of internal multiples

The role of the Heaviside step function

Although the 1D frequency domain prediction is already computationally efficient, the algorithm can be improved through the use of an identity that makes use of the Heaviside step function.

$$\int_{-\infty}^{\infty} dt f(t) \int_{-\infty}^{t} dt' g(t') = \int_{-\infty}^{\infty} dt f(t) \int_{-\infty}^{\infty} dt' H[t - t'] g(t') \tag{16}$$

where

$$H[t - t'] = \begin{cases} 1, t > t' \\ 0, t' > t \end{cases} \tag{17}$$

is the Heaviside step function, and works to replace the integration limits of (16).

$$\int_{-\infty}^{\infty} dt f(t) \int_{-\infty}^{\infty} dt' H[t - t'] g(t') = \iint_{-\infty}^{\infty} dt f(t) dt' H[t - t'] g(t') \tag{18}$$

$$\iint_{-\infty}^{\infty} dt f(t) dt' H[t - t'] g(t') = \int_{-\infty}^{\infty} dt' g(t') \int_{-\infty}^{\infty} dt H[t - t'] f(t) \tag{19}$$

$$\int_{-\infty}^{\infty} dt' g(t') \int_{-\infty}^{\infty} dt H[t - t'] f(t) = \int_{-\infty}^{\infty} dt' g(t') \int_{t'}^{\infty} dt f(t) \tag{20}$$

By using the proof in equations (16-20), equation (10) reduces accordingly.

$$b_3(\omega) = \int_{-\infty}^{\infty} dz\, e^{i2\frac{\omega}{c_o}z} b_1(z) \left[\int_{z'+\epsilon}^{\infty} dz''\, e^{i2\frac{\omega}{c_o}z''} b_1(z'')\right]^2 \qquad (21)$$

In equation (21), one of the three integrals has effectively been removed. Now only two integrals have to be computed for every frequency, where the second integral is now squared. The computational problem has now been greatly reduced. This will be especially useful when the algorithms are expanded to 1.5 dimensions.

The prediction Integrals

Prediction of internal multiples in the 1D frequency domain case can be realized through the use of two nested loops. The first loops over every positive $k_z(ii)$ value, at every value of k_z the algorithm searches for subevents to combine, by searching through every depth $z(jj)$.

Table 1. 1D Frequency domain prediction integral pseudocode

```
                    For ii = kzB:kzE
                Ipos = exp(i * kzPos(ii) * z;
               Ineg = exp(−i * kzPos(ii) * z;
                    intPos = b1z.* Ipos;
                    intneg = b1z.* Ineg;

                                                    For jj = zB: zE − ϵ
                                              Inner = sum(intPos(kk + ϵ: zE));
                                          predP = predP + intNeg(kk) * Inner^2;
                                                    END

                        END
```

The first for loop loops through every wavenumber, for every wavenumber, the inner for loop calculates the prediction integrals of (21) at every depth. The variables k_zB and k_zE represent the minimum and maximum wavenumbers respectively, while zB and zE represent the minimum and maximum depths to search through.

Synthetic example

Figure 1a shows the three interface velocity model used to create a zero offset, normal incidence trace for use in the 1D frequency domain prediction algorithm. In figure 1b, the reflectivity series containing all primaries and first order internal multiples for this velocity model is illustrated. Note that the bottom layer has been treated as a basal half space. The reflectivity series was created using the CREWES function makeTraceWithIm.m, this function takes in the velocity model and then calculates, with transmission loses, the reflectivity series and trace with all primaries and first order internal multiples.

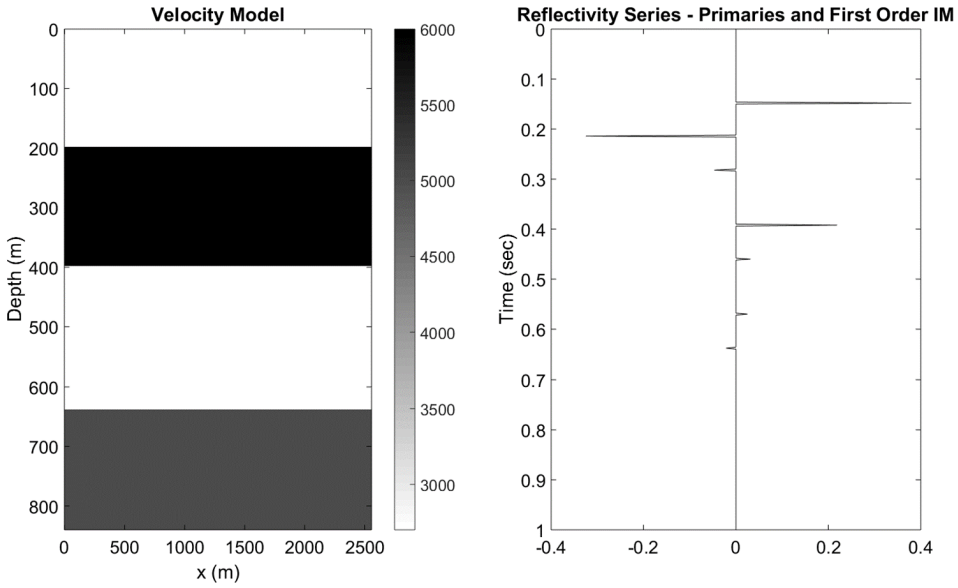

FIG. 1. Velocity model (left), normal incidence trace containing primaries and first order internal multiples (right).

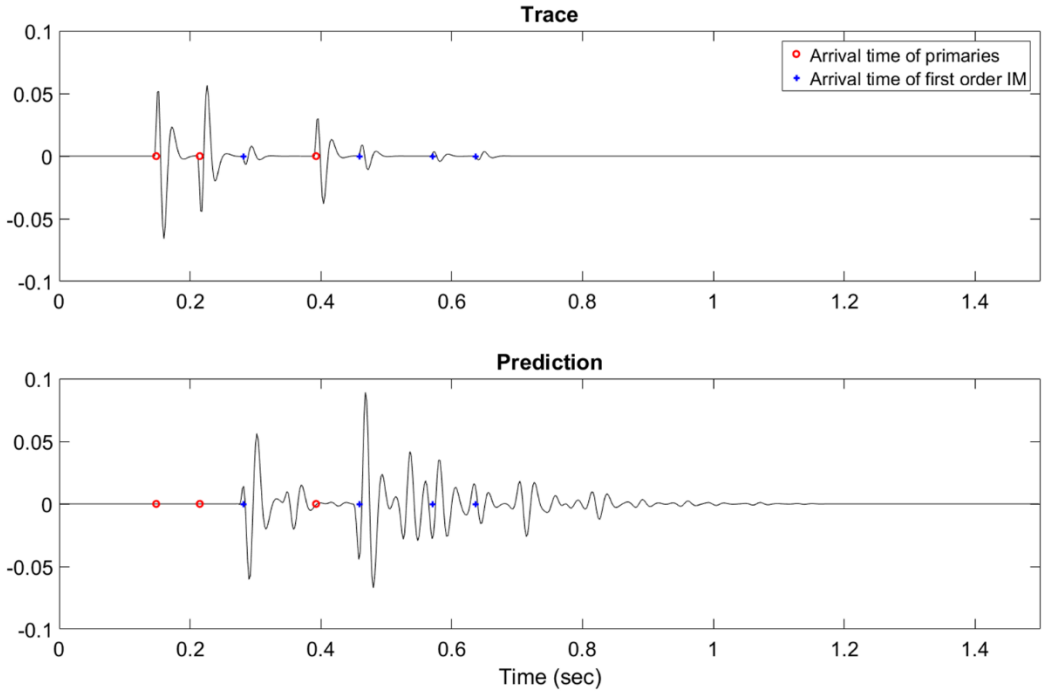

FIG. 2. Trace created by convolving reflectivity series with a 40 Hz minimum phase wavelet (top) result of the 1D frequency domain internal multiple prediction (bottom). Red circles show arrival time of primaries, while blue crosses show arrival time of first order internal multiples.

Figure 2a shows the trace created by convolving the reflectivity series with a 40 Hz minimum phase wavelet. Figure 2b illustrates the result of the internal multiple prediction using (21). Note that the red circles and blue crosses of 2a, and 2b illustrate the arrival times of the primaries and first order internal multiples respectively.

The results of this synthetic example show that the internal multiple prediction scheme of equation (21) is a very robust algorithm for the prediction of internal multiples on a single normal incidence trace. Figure 2b shows that the internal multiples have been predicted with correct traveltime but with erroneous amplitudes, in addition, none of the primary energy has been predicted. These erroneous amplitudes are corrected through adaptive subtraction, which will be discussed in a following section. The algorithm has also predicted some 'extra' energy where no events appear to be. This is due to the fact that the algorithm predicts multiples by combining subevents because the input data contained primaries and first order internal multiples, these two types of events were combined and second order internal multiples were predicted. Some of the energy predicted in figure 2b, that does not align with a blue cross, can be interpreted to second order internal multiples.

1D TIME DOMAIN PREDICTION IN MATLAB

The requirements of the input data for time domain prediction are similar to those of the frequency domain prediction. The major difference is that in the time domain prediction algorithm there is not data preparation step, the prediction can simply be carried out on an input trace, provided that trace is a normal incidence trace.

Internal multiple prediction in the time domain

It can be shown that the prediction algorithm used in free surface multiple elimination (SRME), can be achieved through simple auto-convolution of the data. In order to extend SRME to internal multiple prediction an autocorrelation step is introduced, in fact, equation (10) can be visualized as partial auto-convolution, followed by an autocorrelation. Innanen (2015) presents a formula for time domain internal multiple prediction with artifacts.

$$IM(t) + artifacts = \int_{-\infty}^{\infty} dt'\, s(t'-t) \int_{-\infty}^{\infty} dt''\, s(t'-t'')s(t'') \qquad (22)$$

The artifacts are a result of the fact that equation (22) contains a full convolution followed by a correlation. In order to correctly predict the artifact free, internal multiples, the convolution must be transformed into a partial convolution. Innanen (2015) shows that this can be achieved through the use of a masking function.

$$O(t,t',t'') = H[t'' - \alpha(t,t')]H[\beta(t) - t''] \qquad (23)$$

where,

$$\alpha(t,t') = t' - (t - \epsilon_2)$$
$$\beta(t) = t - \epsilon_1 \qquad (24)$$

The mask function invokes limits on the convolution that force the algorithm to combine events obeying the lower-higher-lower relationship required for internal multiple prediction. With this in mind the matrix form of the 1D time domain prediction (Innanen, 2015) becomes:

$$IM(t_j) = M_R(j,:)\left[O(t_j,\epsilon)\cdot M_C\right]s \qquad (25)$$

The masking function "O" is effectively blocking a portion of the convolution matrix from being involved in the prediction. The portion of the convolution matrix included in the prediction increases with time.

1D time domain prediction in MatLab

Step 1: Create time vectors

We start with the assumption that the input to the 1D time domain prediction is taken to be a trace in the form of an (n x 1) column vector. This trace is then padded to include negative times to create a (2n x 1) column vector. Thus the convolution and mask matrices must have a size that is (2n-1 x n) and the correlation matrix must have a size that is (3n-2 x 2n-1). The output of the prediction algorithm will, therefore, be a (3n-2 x 1) column vector. With this in mind, the first step is to create time vectors to aid in the initialization of the correlation and convolution matrices. Three time vectors need to be created, one that is twice the length of the input time, and is to be padded with negative times (t_{p1}), the second will have a size ($2t_{p1} - 1$) and will be used to create the trace used in the convolution (t_{p2}). The third time vector will have a size ($3t_{p1} - 2$), this vector will be used to initialize the trace in the correlation matrix (t_o).

Step 2: Initialize convolution and correlation matrices

The first step is to create a trace that is the same length as t_{p1} that is padded with zeros for negative times ($trace_{p1}$), and a time reversed version of this trace ($trace_{p1R}$). These two traces ($trace_{p1}, trace_{p1R}$) will be used as the input to build the convolution and correlation matrices respectively. Figure 3 shows a schematic of how steps 1 and 2 are realized in practice. It is important to note that in figure 3, the mask matrix and the convolution matrix (CNV) are multiplied together element wise. In the three middle matrices, the white areas indicate regions of zeros, whereas the gray areas indicate areas of non-zero data. The convolution matrix is built to have a number of columns equal to the length of t_{p1}, while the correlation matrix has a number of columns equal to the length of t_{p2}.

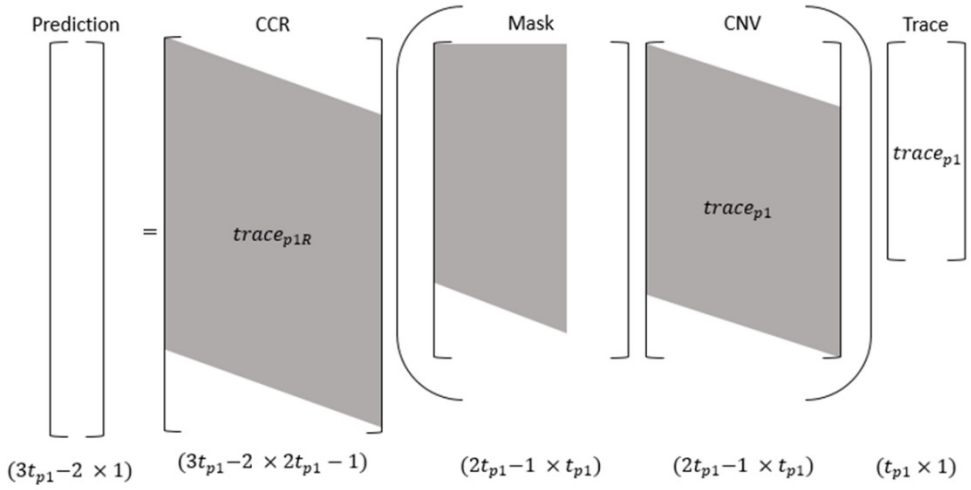

FIG. 3. Schematic of matrix multiplication for 1D time domain prediction

Step 3: The masking matrix

The masking matrix enforces the limits on the integration such that the lower-higher-lower relationship is satisfied. If the x coordinate of the mask matrix is taken to be t" then the limit invoked by the second step function of (23) becomes a vertical straight line. When the y coordinate is taken to be t', then the limit invoked by the first step function of (23) becomes a sloped straight line.

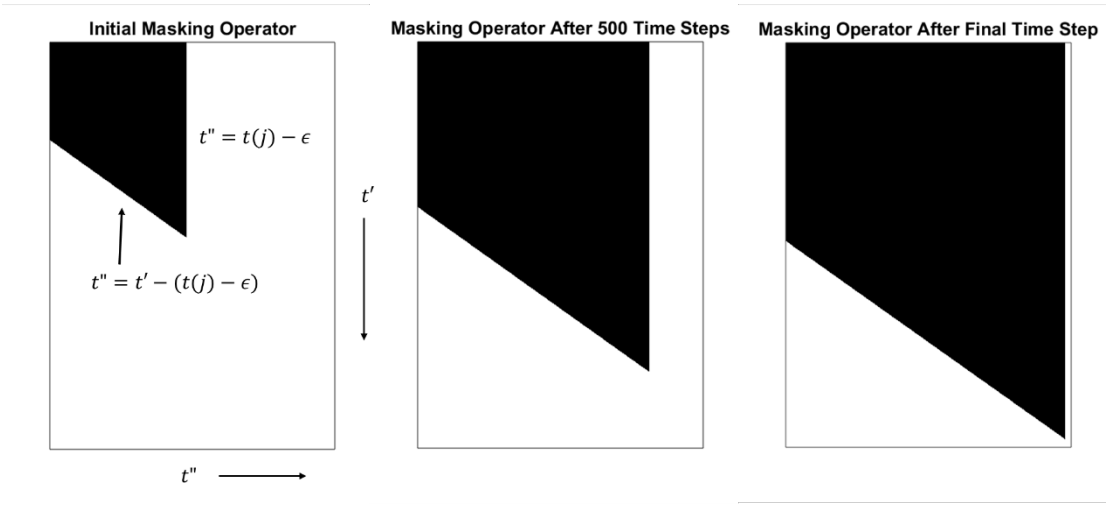

FIG. 4. Masking operator growth with time

In figure 4 the section shown in black is a region of unity, while the section in white indicates a region of zeros. When multiplied by the convolution matrix the masking operator restricts the portion of the convolution that contributes to the prediction, which is the portion obeying a lower-higher-lower relationship. Figure 4 shows how the masking operator grows with increasing time.

The masking operator is created using the CREWES function *makeMask.m*, as its input it takes in a *tIndex* and the length of the t_{p1} time vector *Nt.* The first input *tIndex* is equivalent to t(j) of figure 4, *tIndex* starts at a value equal to the length of the input trace minus the value of epsilon, and increases by 1 for each time step. The second input *Nt* is equivalent to the length of t_{p1}. Construction of the masking operator takes place within a for loop, for the first column of the mask matrix every value from the first row up until the row defined by *tIndex* is initialized to one. Then with every iteration of the for loop, the range of ones is extended by one row, and one column, creating a slanted line with a slope of one. This step is forming the condition of $t" = t' - (t(j) - \epsilon)$ shown in figure 4. Outside of the for loop, every value to the right of *tIndex* is then set to zero, creating the vertical line limit.

Step 4: Prediction of internal multiples

As discussed previously internal multiple prediction in the time domain can be viewed as partial convolution followed by correlation. In order to invoke a partial convolution, a full convolution matrix of the zero padded trace is created, the Hadamard product of this convolution matrix with the masking matrix creates the partial convolution matrix. The

Table 2. Pseudocode for creation of mask matrix

For ii = 1:length(t_{p1})
mask(1:tIndex+ii-1,ii) = 1;
END
mask(:,tIndex:length(t_{p1})) = 0;

zero padded trace is then time reversed and used to create the cross correlation matrix. Each row of the correlation matrix is then multiplied by the masked convolution matrix, the result of which is multiplied by the input trace.

$$predt(ii) = CCR(ii,:)[mask \odot CONV](trace_{p1}) \quad (26)$$

Where ($\odot$) represents the Hadamard product. The prediction vector will have a length of $(3t_{p1} - 2)$, therefore it is important to set the final prediction to be equal to $predt(3Nt: 3Nt + Nt - 1)$.

Synthetic Example

The requirements for the 1D time domain prediction are the same as for the 1D frequency domain prediction. That is a single normal incidence trace is assumed as the input to the algorithm. For consistency and simplicity, the same velocity model and input of figure 1 will be utilized.

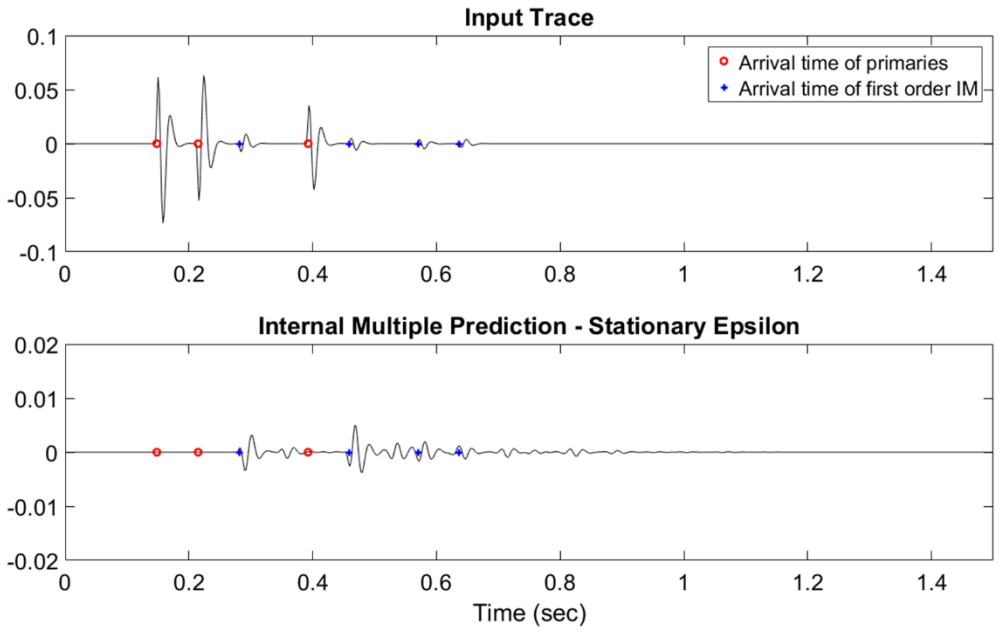

FIG. 5. Trace created by convolving reflectivity series with a 40 Hz minimum phase wavelet (top), the result of the 1D time domain internal multiple prediction (bottom). Red circles show arrival time of primaries, while blue crosses show arrival time of first order internal multiples.

Comparing figure 5 to figure 2 the 1D time and frequency domain predictions produce comparable results. One noticeable difference is that the time domain prediction has predicted much smaller amplitudes than the frequency domain prediction.

1.5D FREQUENCY WAVENUMBER PREDICTION IN MATLAB

Internal multiple prediction in the frequency wavenumber domain

Equation (12) above represents the 1.5D equation for the prediction of multiples in the wavenumber-frequency domain. By letting $k_z = 2q_g$ and making use of the identity in (20), equation (12) is transformed.

$$b_3(k_g,\omega) = \int_{-\infty}^{\infty} dz'\, e^{-i2q_g z'} b_1(k_g,z')\left[\int_{z'+\epsilon}^{\infty} dz''\, e^{i2q_g z''} b_1(k_g,z'')\right]^2 \qquad (27)$$

where,

$$q_g = \frac{\omega}{c_o}\sqrt{1-\frac{k_g^2 c_o^2}{\omega^2}} \qquad (28)$$

Input Data

For 1.5D internal multiple prediction algorithms, the input data is assumed to be a split spread shot record, over a horizontally stratified geology. It is also assumed that the shot record has been deghosted, and has had free surface multiples and direct arrivals removed. Although not imperative for synthetic datasets, the application of deconvolution prior to prediction is a helpful step, in this case, deconvolution was not applied.

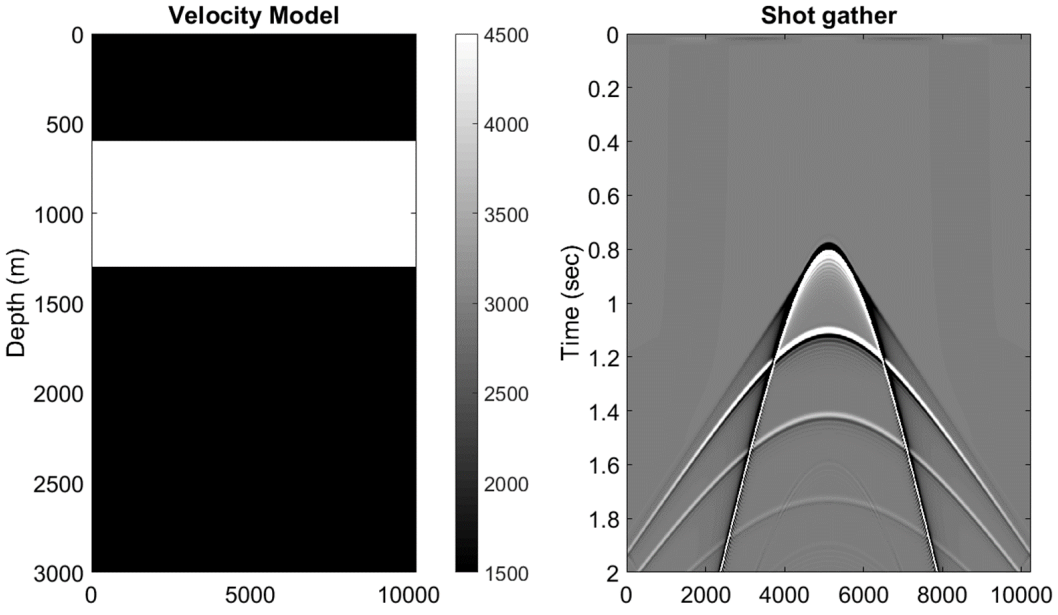

FIG. 6. Velocity model (left) and split spread shot record (right).

Figure 6 shows the velocity model that will be used for the 1.5D predictions and the resulting shot record created by the CREWES finite difference algorithm.

Data preparation

Step 1: 2D Fourier Transform

The input data as seen by figure 6 is acquired with the coordinates of horizontal geophone location (x_g) and the vertical coordinate of time. Equation (12) shows that we require the data in the (k_g, z) domain, thus the first step in the data preparation phase is to perform a two dimensional Fourier transform taking the data to the wavenumber-frequency domain.

$$d(x_g, t) \rightarrow D(k_g, \omega) \tag{29}$$

Step 2: Change of variables $\omega \rightarrow k_z$

The next step that is performed is to make a change of variables from ω to k_z.

$$D(k_g, \omega) \rightarrow D(k_g, k_z) \tag{30}$$

Step 2a: Resampling

Since k_z is equal to twice q_g, it is easy to see through equation (28) that this change of variables is not a linear operation. Since this change of variables is non-linear, a simple swap of variables would result in an irregularly sampled (k_g, k_z) grid. To overcome this hurdle, a regular (k_g, k_z) grid is computed, and then the irregular frequency values that adhere to this new grid are computed. The data is then linearly interpolated to fit the regular (k_g, k_z) grid.

Starting with,

$$k_z = 2q_g = \frac{2\omega}{c_o}\sqrt{1 - \frac{k_g^2 c_o^2}{\omega^2}} \tag{31}$$

solving for ω and making the conversion to frequency, the irregular frequency values are,

$$f = \frac{c_o k_z}{2\pi}\sqrt{1 + \frac{k_g^2}{k_z^2}} \tag{32}$$

Using these irregular frequency values, the data on the irregular (k_g, k_z) grid, is linearly interpolated onto a regular (k_g, k_z) grid.

$$\frac{y - y_L}{x - x_L} = \frac{y_H - y_L}{x_H - x_L} \tag{33}$$

Equation (33) represents the standard linear interpolation formula, changing the placeholder variables to the variables of interest,

$$\frac{data - data_L}{freq - freq_L} = \frac{data_H - data_L}{freq_H - freq_L} \tag{34}$$

Solving for the interpolated data,

$$data = (freq - freq_L)\left(\frac{data_H - data_L}{freq_H - freq_L}\right) + data_L \quad (35)$$

Equation (35) is carried out for every (k_x, k_z), the term to the left of the addition must be computed for both the real and imaginary portions of the data.

Table 3. Pseudocode for the resampling of data to a regular grid.

```
For ii = Nx/2 + 2: Nx − 1
                                    For jj = Nz/2 + 2: Nz − 1
                                    freq = irregFreq(jj, ii)
                                index = find((f < freq + df)&((f > freq − df))
                                    indexLow = index(1);
                                    indexHigh = max(index)
                                    data_L = data(indLow, ii)
                                    data_H = data(indHigh, ii)
                                    freq_L = freq(indLow)
                                    freq_H = freq(indHigh)
                                data(jj,ii) =(freq − freq_L)((data_H − data_L)/(freq_H − freq_L)) + data_L
                                    END

    END
```

Step 3: Data Scaling

The data are then scaled by the obliquity factor $-i2q_s$ due to the fact that the 1.5D internal multiple prediction is basically an un-collapsed Stolt migration (Weglein et al., 2003).

$$D(k_g, k_z) = -2iq_s D(k_g, k_z) \quad (36)$$

Step 4: Inverse Fourier transform

The last step in the data preparation phase is to inverse Fourier transform the data over the k_z variable.

$$D(k_g, k_z) \rightarrow b_1(k_g, z) \quad (37)$$

Figure 7 below shows the result of performing the four data preparation steps on the data in figure 6 (right).

Internal multiple prediction

Internal multiple prediction in 1.5 dimensions is very similar to the prediction in 1 dimension, except the prediction integral is repeated over a given variable. In this case, the prediction is carried out for every positive lateral wavenumber.

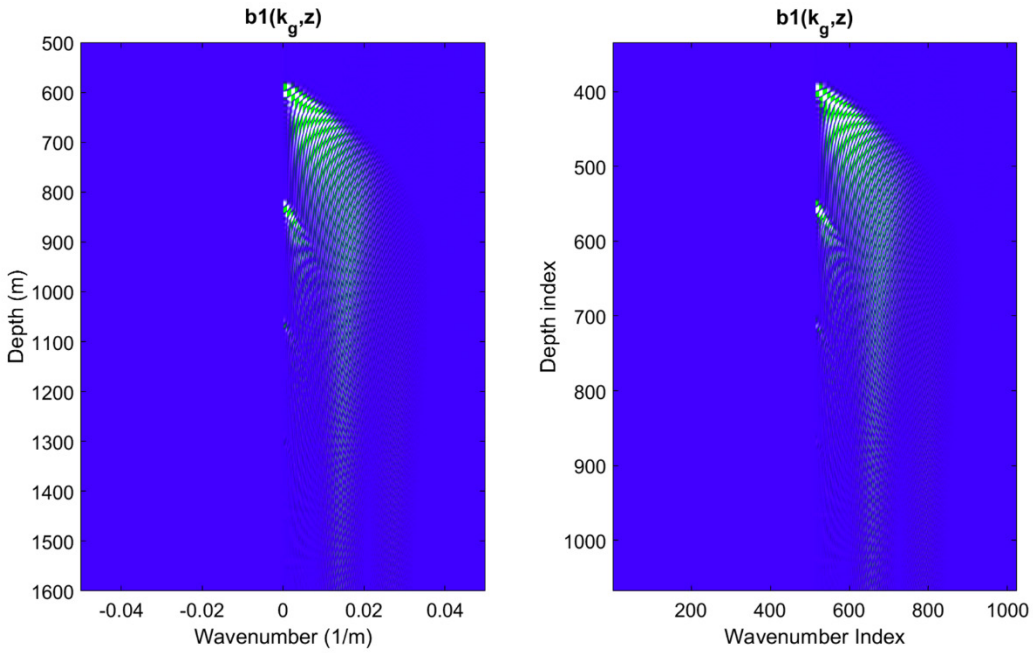

FIG. 7. Prepared input data $b_1(k_g, z)$ to be used in the 1.5D wavenumber frequency internal multiple prediction algorithm (left), same data plotted against vector index (right).

Table 4. Frequency wavenumber prediction in pseudocode

$For\ ii = kxB{:}\,kxE$
$F = kx(ii)^2 c_o^2./\omega.^2$
$q_g = (\omega/c_o).* sqrt(1 - F)$

$For\ jj = \ \omega B{:}\,\omega E$
$IP = i * 2q_g(jj) * z;$
$IN = -i * 2q_g(jj) * z;$
$I1 = \mathrm{b1}(:, \mathrm{ii}) * \exp(IP);$
$I2 = \mathrm{b1}(:, \mathrm{ii}) * \exp(IN);$

$For\ kk = zB{:}\,zE$
$S = sum\big(I1(kk + \epsilon{:}\,zE)\big);$
$S = dz * S;$
$P = P + I2(kk) * S * S;$
END

$P = P * dz;$
END

END

Table 4 shows the prediction algorithm in pseudocode, $kxB, kxE, \omega B, \omega E, zB, and\ zE$, represent the integration limits for the wavenumber, angular frequency and depth respectively. As seen in figure 7, it is only necessary to integrate over depths and wavenumbers where meaningful data exists, thus by carefully selecting $kxB, kxE, zB, and\ zE$ computational efficiency can be improved. In table 4, "S" takes care of the integral inside the square brackets of equation 27, while I2 takes care of the second integral.

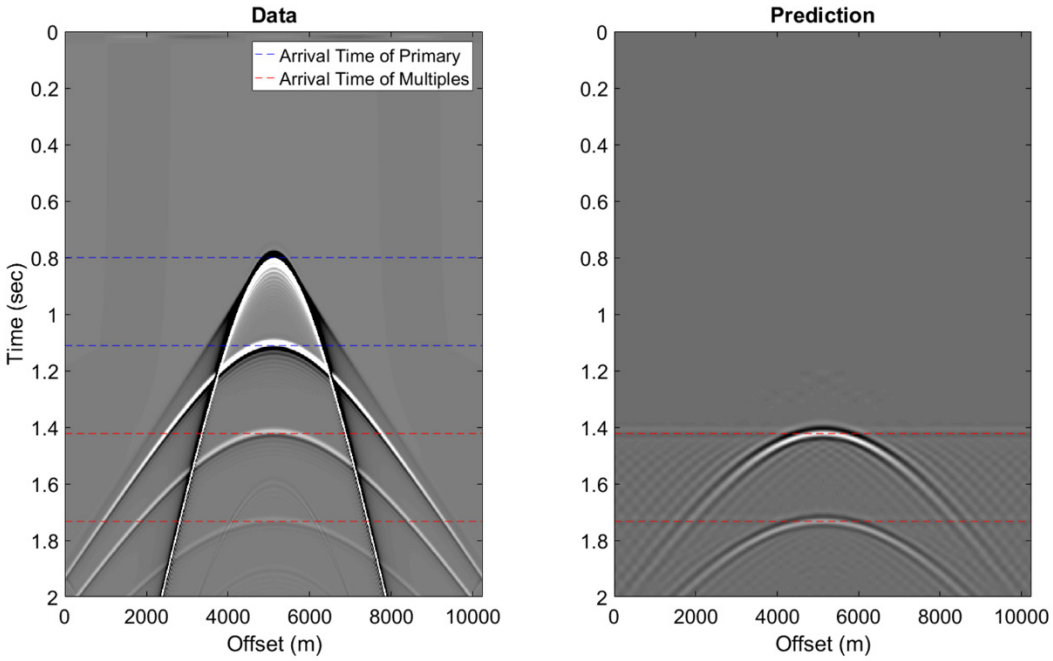

FIG. 8. Input data (left), the result of the prediction using pseudocode in table 4 (right). The blue dashed lines indicate the zero offset travel time of the two primary events. The red dashed lines indicate the zero offset travel time of a first order internal multiple, and a second order internal multiple.

Table 4 above shows the pseudocode representation of the 1.5D internal multiple prediction algorithm in the wavenumber frequency domain. Figure 8 shows the results of applying this pseudo code to the prepared data of figure 7. The algorithm has predicted the multiples at the correct traveltimes, again the amplitudes can be corrected by employing adaptive subtraction to the prediction results.

1.5D TAU-P PREDICTION IN MATLAB

Internal multiple prediction in the tau-p domain

Coates and Weglein (1996) presented a planewave (tau-p) domain version of the internal multiple prediction algorithm.

$$b_3(p,\tau) = \int_{-\infty}^{\infty} d\tau\, e^{i\omega\tau} b_1(p,\tau) \int_{-\infty}^{\tau-\epsilon} d\tau'\, e^{-i\omega\tau'} b_1(p,\tau') \times \int_{\tau'+\epsilon}^{\infty} d\tau''\, e^{i\omega\tau''} b_1(p,\tau'') \quad (38)$$

Making use of the Heaviside step identity, (38) becomes,

$$b_3(p,\tau) = \int_{-\infty}^{\infty} d\tau'\, e^{-i\omega\tau'} b_1(p,\tau') \left[\int_{\tau'+\epsilon}^{\infty} d\tau''\, e^{i\omega\tau''} b_1(p,\tau'')\right]^2 \quad (39)$$

The input data for the 1.5D tau-p prediction maintains the same assumptions as the wavenumber-frequency prediction. The data in figure 6, will also serve as the input data in the tau-p internal multiple prediction algorithm.

Data preparation

Step 1: Tau-p transform

The first step in preparing the data is performing a tau-p transform on the data. In this study, the CREWES tool box MatLab function *tptran.m* is used, which performs the transform in the frequency domain through a linear phase shift.

$$d(x_g, t) \rightarrow D(p,\tau) \quad (40)$$

Step 2: Fourier transform

The next step is to Fourier transform the data over the "τ" variable so that the data may be scaled in a following step.

$$D(p,\tau) \rightarrow D(p,\omega) \quad (41)$$

Step 3: Data scaling

The data is then scaled by an obliquity factor (Weglein et al., 2003),

$$D(p,\omega) = -i2q_s D(p,\omega) \quad (42)$$

Step 4: Inverse tau-p transform

The final step is to take the data back to the tau-p domain through the use of an inverse tau-p transform. The CREWES toolbox function *iptran.m* is used to accomplish this.

$$D(p,\omega) \rightarrow b_1(p,\tau) \quad (43)$$

Figure 9 below shows the result of carrying out equations (40-43) on the shot record in figure 6. It is important to note that in both figures 7 and 9 that only the positive horizontal slowness and wavenumbers are used. In 1.5 dimensions the algorithms only need to be calculated for positive values, the negative values can be filled in by conjugate symmetry, greatly increasing the computational efficiency of the algorithm.

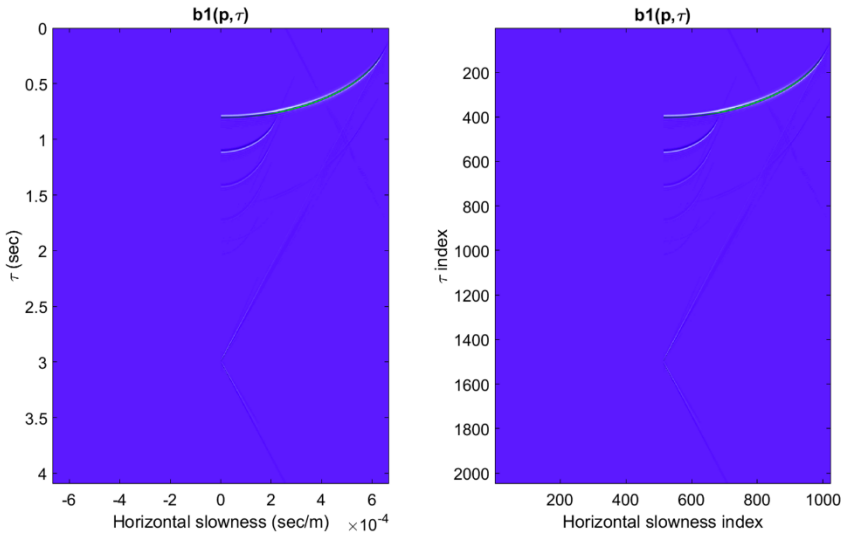

FIG. 9. Prepared input data $b_1(p,\tau)$ to be used in the 1.5D tau-p internal multiple prediction algorithm.

Internal multiple prediction

Internal multiple prediction in 1.5 dimensions is very similar to the prediction in 1 dimension, except the prediction integral is repeated over a given variable. In this case, the prediction is carried out for every positive value of the horizontal slowness.

Table 5. Tau-p prediction in pseudocode

```
For ii = pB:pE
                    For jj = ωB:ωE
                    IP = i * ω(jj) * τ;
                    IN = −i * ω(jj) * τ;
                    I1 = b1(:, ii) * exp(IP);
                    I2 = b1(:, ii) * exp(IN);
                                        For kk = τB:τE
                                        S = sum(I1(kk + ϵ:τE));
                                        S = dτ * S;
                                        P = P + I2(kk) * S * S;
                                        END
                    P = P * dτ;
                    END
END
```

Table 5 shows the prediction algorithm in pseudocode, $pB, pE, \omega B, \omega E, \tau B, \tau E$, represent the integration limits for the horizontal slowness, angular frequency and traveltime respectively. Once again $pB, pE, \tau B, and\ \tau E$ can be selected to improve the

computational efficiency. In table 5, "S" takes care of the integral inside the square brackets of equation 39, while I2 takes care of the second integral.

Figure 10 shows the results of applying this pseudo code to the prepared data of figure 9. The algorithm has predicted the multiples at the correct traveltimes, again the amplitudes can be corrected by employing adaptive subtraction to the prediction results.

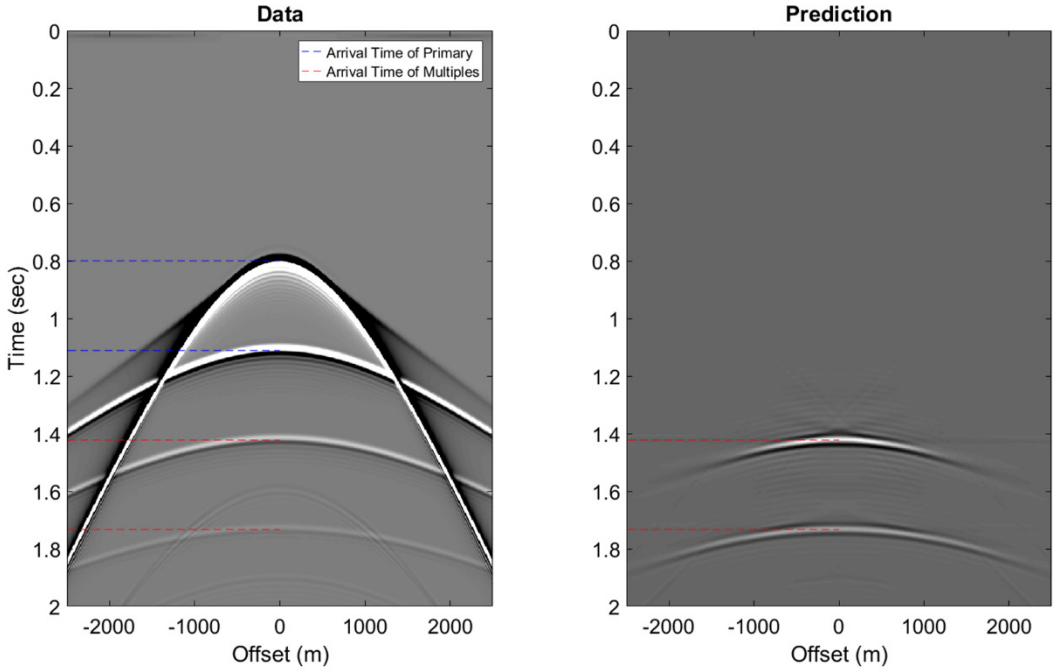

FIG. 10. Input data (left), the result of the prediction using pseudocode in table 5 (right). The blue dashed lines indicate the zero offset travel time of the two primary events. The red dashed lines indicate the zero offset travel time of a first order internal multiple, and a second order internal multiple.

ADAPTIVE SUBTRACTION

Inverse scattering series internal multiple prediction is a very robust algorithm for accurately predicting the travel times of internal multiples. However, as seen in the previous examples the prediction suffers from both amplitude and phase differences. Series truncation, incomplete deconvolution of the source wavelet, noise, and residual ghosts all contribute to the erroneous results. Adaptive subtraction methods work to correct this by more accurately matching the prediction to the measured data. This hurdle is typically overcome by convolving a filter with the prediction, which allows for corrections in both phase and amplitude (Abma et al. (2005), Wang (2003), Verschuur et al. (1992). The result of this adaptive subtraction is,

$$a = d - Mf \tag{44}$$

Where (d, M, f) are the data, a convolution matrix of the multiple prediction, and the filter respectively, the goal being to solve for this filter. Verschuur et al. (1992) suggest accomplishing this by minimizing the L_2 norm, and thus minimizing the energy. Their idea is that if multiples and primaries do not overlap, then the filter which acts to completely

remove the multiples will have less energy than any other prediction where the multiples are not completely removed. The L_2 norm for equation (44) is as follows.

$$\|a\|_2 = \|d - Mf\|_2 = \sum_{n=1}^{N} a_n^2 \tag{45}$$

The least squares solution for f is,

$$f = (M^T M)^{-1} M^T d \tag{46}$$

If the assumption that multiples and primaries do not overlap is violated, as it often is with land data, then the minimization of the L_2 norm is no longer appropriate. When multiples overlap with primaries, the filter which minimizes the energy will also remove primary energy. In fact, because primaries contain more energy than multiples, the attenuation of primaries is actually given precedence.

To overcome this problem Verschuur et al. (2004) suggested instead minimizing the L_1 norm, which works to minimize amplitude instead of energy.

$$\|a\|_1 = \|d - Mf\|_1 = \sum_{n=1}^{N} |a_n| \tag{47}$$

Finding the filter from the L_1 norm requires solving the normal equations given by (Bube and Langan (1997)).

$$M^T W M f = M^T W d \tag{48}$$

Where W is a diagonal matrix with entries W_{ii} related to the residual at time "*i*" by:

$$W_{ii} = |r_i|^{-1} \tag{49}$$

The residuals are the given by,

$$r = Mf - d \tag{50}$$

While the L_1 norm works to correct for the short comings of the L_2 norm it is often hard to calculate in practice, whenever the residuals are zero, the elements of W become singular and the norm becomes difficult to calculate. (Keating et al., 2015). Keating et al. (2015) proposed the idea of a hybrid L_1/L_2 norm, a norm that acts like the L_1 norm when residuals are large but acts more like an L_2 norm when the residuals approach zero. To accomplish this (48) is solved for "*f*".

$$f = (M^T W M)^{-1} M^T W d \tag{51}$$

Where the weighting matrix takes the form.

$$W = \left(\frac{1}{1 + \left(\frac{r_i}{\sigma} \right)} \right)^{1/2} \tag{52}$$

Adaptive subtraction in MatLab

Equation 51 cannot be solved in a linear fashion, instead, it must be solved iteratively. To begin, the convolution matrix of the multiple prediction is calculated having the same number of rows as the length of the prediction, and having the same number of columns as

the filter length. Next, the first iteration of (51) is calculated with the weighting matrix set to an identify matrix with the MatLab function *eye()* having the same size as the length of the trace. After the first iteration is complete the residuals are calculated according to (50), and then the weight matrix is recalculated according to (52). The user of the function *adaptiveSubtraction.m* provides as an input the number of iterations to be performed. In addition, they also alter the filter length, smoother length, and norm type. Based on the selection of the norm type, sigma is chosen to more heavily weight either the L_1 or the L_2 norm.

Synthetic Example

In the previous section a 1D adaptive subtraction algorithm was reviewed, as such the input to the prediction algorithm will be assumed to be a normal incidence trace.

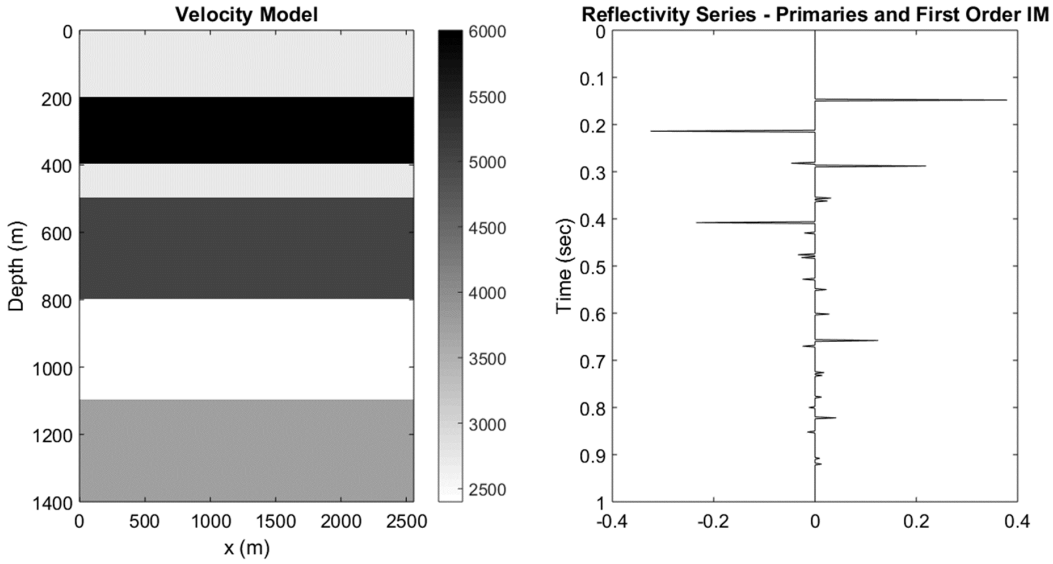

FIG. 11. Velocity model (left) and the reflectivity series used to create the input trace for the adaptive subtraction algorithm (right).

Figure 11 shows the velocity model and the resulting reflectivity series with first order internal multiples. The reflectivity series is convolved with a 45 Hz ricker wavelet to create the trace that will be used as an input to the prediction algorithm. The reflectivity series was once again created using *makeTraceWithIM.m*, the prediction, in this case, was carried out in the frequency domain.

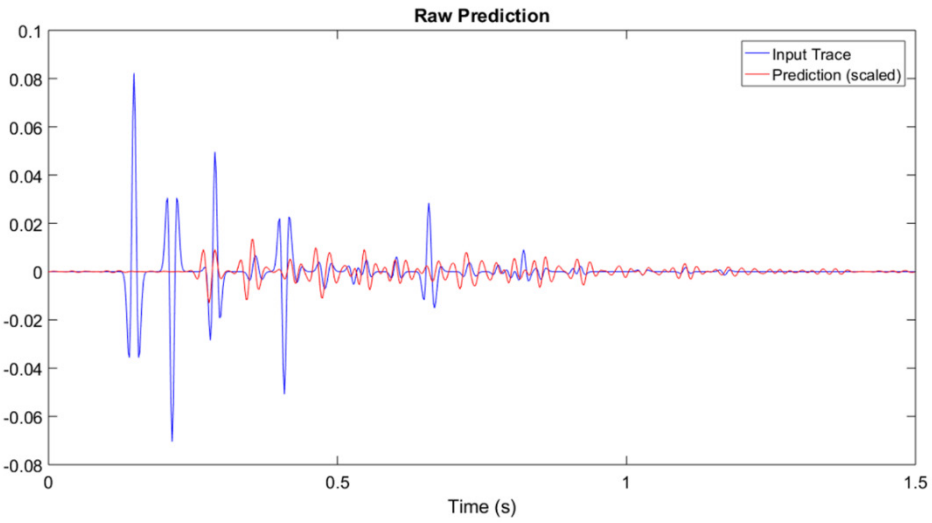

FIG. 12. The input trace (blue) and scaled prediction with no adaptive subtraction filter applied (red).

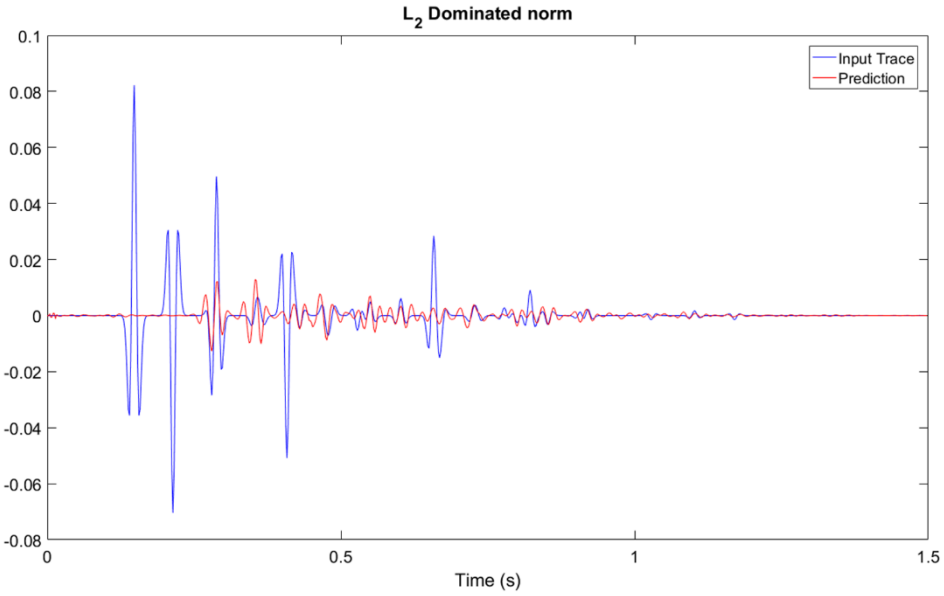

FIG. 13. Input trace (blue), and prediction with a filter dominated by the L_2 norm (red).

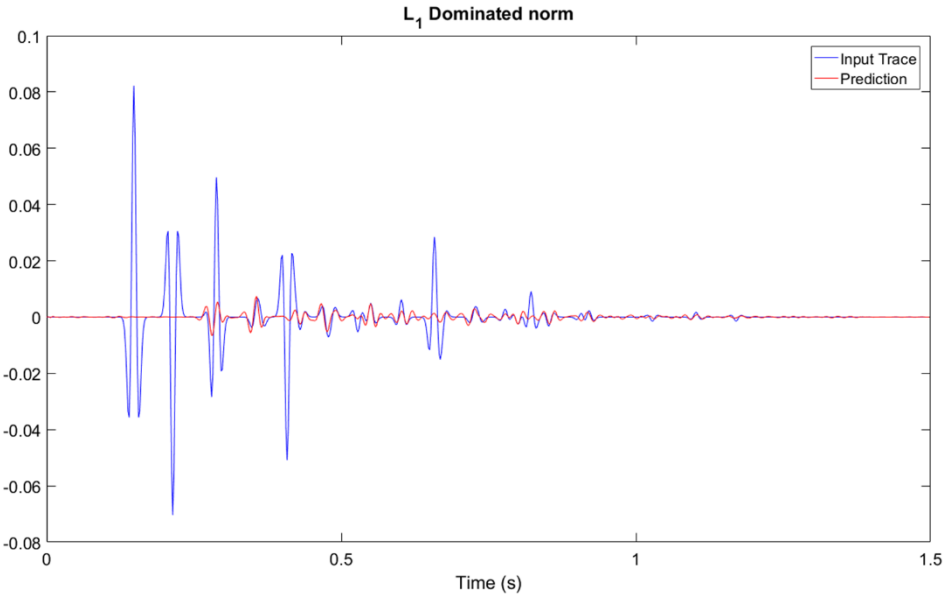

FIG. 14. Input trace (blue), and prediction with a filter dominated by the L_1 norm (red).

Figures 12-14 show the results of the raw prediction, the prediction convolved with a filter dominated by the L_2 norm, and the prediction convolved with a filter dominated by the L_1 norm. It is evident from figure 13 that the L_2 norm filter has improved the prediction but is struggling to match multiples in the more complicated middle portion of the trace. The L_1 dominated filter has matched the multiples better in all regions of the trace.

Changing the filter length and smoother length causes the prediction to become more aggressive or less aggressive. A shorter filter matches less of the data in the least squares solution, where as a filter that is long enough will perfectly match and signal.

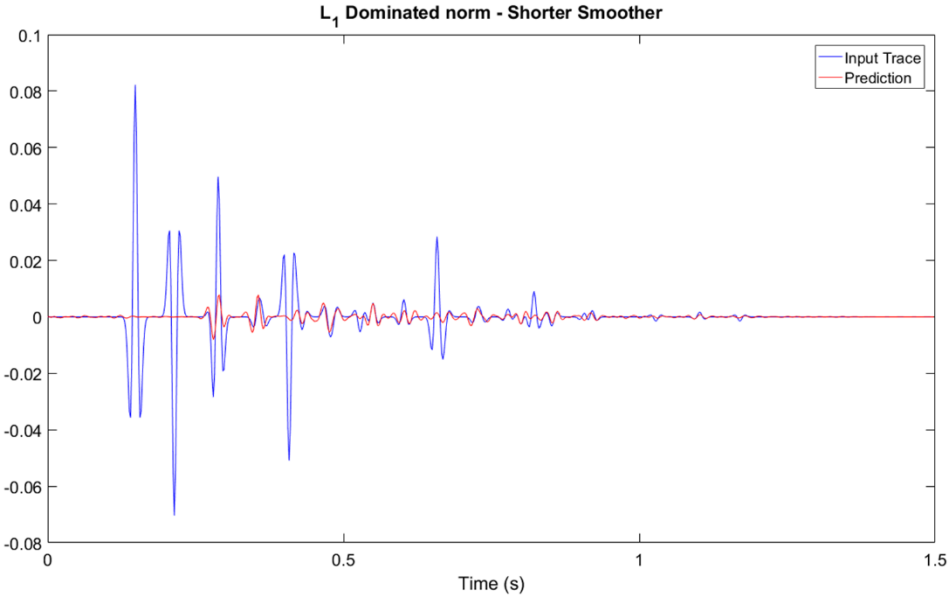

FIG. 15 The result of using a shorter smoother in the L1 norm solution

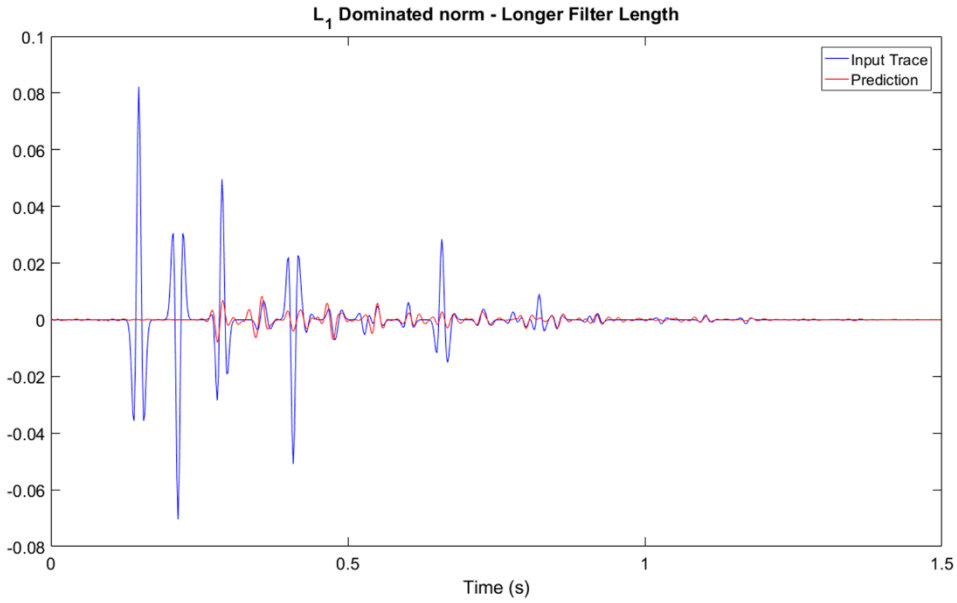

FIG. 16. The result of using a longer filter in the L1 norm solution

Figures 15 and 16 show the results of applying a shorter smoother, and longer filter respectively. Comparing both figures to figure 14, it is evident in this case that applying a shorter smoother, and the longer filter has provided a poorer match to the data. In both cases, the multiple that interacts with the third primary are poorly matched. In addition, poorer matches to other multiples are also evident.

DISCUSSION

This paper has served to review four domains of internal multiple prediction, as well as 1D adaptive subtraction. Throughout this paper the prediction equation has been reviewed for the 1D frequency, 1D time, 1.5D tau-p, and the 1.5D wavenumber-frequency domains. In each section both the equation used to predict the internal multiples is reviewed, as well as the steps needed to prepare the dataset for the prediction step. Synthetic examples of each prediction type are shown to emphasize the robustness of inverse scattering series internal multiple prediction. Internal multiple prediction produces accurate traveltime estimates of internal multiples but contains phase and amplitude errors due to series truncation, noise, and incomplete deconvolution. Adaptive subtraction has been shown to be a promising method for correcting for phase and amplitude errors. An L_1/L_2 hybrid adaptive subtraction algorithm (Keating et al., 2015) was reviewed, and then a description of its MatLab implementation is given. The work presented here is intended to be a companion paper to the internal multiple toolbox contained in the 2016 CREWES software release. It will work to aid the user in a better understanding of the algorithms contained in the internal multiple toolbox and their applications.

ACKNOWLEDGEMENTS

I would like to thank the sponsors of the CREWES project as well as NSERC for funding this work through the grant CRDPJ 461179-13. I would like also like to acknowledge my supervisor Kris Innanen for his guidance, as well as Scott Keating and Jian Sun for their insights, discussion, and aid in programming some of the algorithms presented.

REFERENCES

Abma, R., Kabir, N., Matson, K., Shaw, S., and McLain, B., 2005, Comparison of adaptive subtraction methods for multiple attenuation: The Leading Edge, **24**, 277-280.

Bube, K., and Langan, R., 1997, Hybrid l1/l2 minimization with applications to tomography: Geophysics, **62**, 1183-1195.

Coates, R. T., A. B. Weglein, 1996, Internal multiple attenuation using inverse scattering: results from pre-stack 1D & 2D elastic synthetics, SEG Technical Program Expanded Abstracts pp 1522-1525.

Guitton, A., Verschuur, D. J., 2004, Adaptive subtraction of multiples using the L_1 – norm: Geophysical Prospecting, **52**, 27-38.

Innanen, K., 2015, Time domain internal multiple prediction: CREWES Research Report, **27**, 30.1-30.14.

Keating, S., Sun, J., Pan, P., Innanen, K., Nonstationary L_1 adaptive subtraction with application to inverse scattering multiple attenuation: CREWES Research Report, **27**, 37.1-37.17.

Pan, P., Innanen, K., 2013, Numerical analysis of 1.5D internal multiple prediction: CREWES Research Report, **25**, 65.1-65.12.

Sun, J., Innanen, K., 2014, 1.5D internal multiple prediction in the plane wave domain: CREWES Research Report, **26**, 74.1-74.11.

Verschuur, D., Berkhout, A., and Wapenaar, C., 1992, Adaptive surface-related multiple elimination: Geophysics, **57**, 1166-1177.

Wang, Y., 2003, Multiple subtraction using an expanded multichannel matching filter: Geophysics, **68**, 346-354.

Weglein, A. B., Araujo, F. V., Carvalho, P. M., Stolt, R. H., Matson, K. H., Coates, R. T., Corrigan, D., Foster, D. J., Shaw, S. A., Zhang, H., Inverse scattering series and seismic exploration: Institute of Physics Publishing, Inverse Problems, **19**, R27-R83.

Weglein, A. B., Gasparotto, F. A., Carvalho, P. M., Stolt, R. H., 1997, An inverse scattering series method for attenuating multiples in seismic reflection data: Geophysics, **62**, 1975-1989.