1.5D internal multiple prediction in the wavenumber-time domain with implementation in Python

Matthew Eaid and Kris Innanen

ABSTRACT

Most seismic interpretation and processing algorithms treat primaries as signal and multiples as noise. Multiple reflections are separated into two major categories. Multiples which have at least one down going reflection at the earth's surface are called surface related internal multiples. Multiple reflections where all downward reflections are contained to the subsurface are referred to as internal multiples. While the prediction and subsequent removal of surface related internal multiples is a fairly well understood problem, the prediction of internal multiples is not. Historically internal multiples have been predicted by exploiting assumptions about the multiples that the primaries do not obey. However, when these assumptions are not met by the internal multiples the prediction algorithms fail to properly predict the internal multiples. Weglein et al., (1997) proposed a fully data driven, wave equation method of predicting internal multiples based on the inverse scattering series. The algorithm derived by Weglein et. al, performs the prediction in the frequency-wavenumber domain, and then converts the prediction back to the offset-time domain. In recent years The CREWES project has adapted the original algorithm into many different domains in order to investigate the optimal domain in which to predict internal multiples; one such domain is the wavenumber time domain. We will present the wavenumber-time algorithm, provide pseudocode examples of how to implement it in the Python programming language, and will show a synthetic prediction example.

INTRODUCTION

While primary reflections are typically treated as signal in seismic processing and interpretation, multiples are treated as noise, and their removal is usually desirable. Multiple reflections are seismic signals in which the wavefield has undergone more than one upward reflection before being recorded. Multiples come in two distinct categories, multiples in which one of the downward reflections has occurred at the surface of the earth are called free surface multiples, multiples in which all reflections are confined to the subsurface are termed internal multiples. Surface multiples are typically periodic and predictable in nature, and thus their removal is a fairly well understood problem. The same cannot be said for internal multiples, although promising strides have been made in recent years.

Many of the methods of predicting surface multiples rely on exploiting assumptions about the primary event. Primaries and multiples have differing moveout, and when the gathers are NMO corrected, then stacked, the under corrected multiples are attenuated (Yilmaz, 2001). Alam and Austin (1981) and Treitel et al. (1982) were the first to recognize that multiples are exactly periodic in the tau-p domain, and proposed using predictive deconvolution to attenuate them. While both of these methods performed well on surface multiples, their assumptions are generally violated by internal multiples. When the assumptions of these algorithms are violated the resulting prediction is a poor one.

Several wave equation-based internal multiple prediction schemes exist, (Weglein et al. 1997; Jakubowicz, 1998; Berkhout, 1999), nevertheless the inverse scattering series approach originally presented by Weglein et al. (1997) remains the pinnacle for predicting internal multiples. The original algorithm was originally designed to perform the prediction algorithm in the wavenumber-frequency domain, however an important research topic is adapting the algorithm to predict multiples in other domains. For instance, Sun and Innanen (2015) showed that performing the prediction in the tau-p domain reduced the number of artifacts in the resulting prediction. While the algorithm is limited to search for combinations of sub-events in either time or depth, the domains in which the prediction is calculated can vary quite widely.

One such domain is the wavenumber-time domain, this paper will review the work presented by Innanen (2015) on internal multiple prediction in the time domain. We will then discuss the implementation of the algorithm in the wavenumber time domain, and provide a synthetic prediction example.

WAVENUMBER-TIME DOMAIN PREDICTION

Let $d(x_g, t)$ represent a single split-spread shot record, over 1.5D laterally homogenous and isotropic medium, with t being time and x_g being the geophone locations. Since the inverse scattering series approach to internal multiple prediction searches through the dataset in an automatic way, and combines subevents obeying a lower-higher-lower relationship, it is important that the data is pre-processed prior to the prediction phase. In particular, it is important the data is noise free (although it is not required), the data is also assumed to have all ghosts, surface related multiples, and direct arrivals removed. Although it is helpful to deconvolve the data it is not always required, especially in the case of synthetics. The original formula presented by Weglein et al. (1997) after reduction to the 1.5D case is:

$$IM_{k\omega}(k_g,\omega) = \int_{-\infty}^{\infty} dz e^{ik_z z} b_1(k_g,z) \int_{-\infty}^{z-\epsilon} dz' e^{-ik_z z'} b_1(k_g,z') \times \int_{z'+\epsilon}^{\infty} dz'' e^{ik_z z''} b_1(k_g,z'') \quad (1)$$

where k_g is the Fourier conjugate of the geophone location x_g, z is the pseudo-depth, and ϵ is a search limiting parameter that enforces a minimum separation distance between events that may be combined. Replacing z with t and $b_1(k_g, \omega)$ with $d(x_g, t)$ has no effect on equation (1). Then letting $D(k_g, t)$ be the Fourier transform of $d(x_g, t)$, we arrive at the formula presented by Innanen (2015). Beginning with

$$IM_{k\omega}(k_g,\omega) = \int_{-\infty}^{\infty} dt e^{i\omega t} \mathrm{S}(k_g,t) \int_{-\infty}^{t-\epsilon} dt' e^{-i\omega t'} S(k_g,t') \times \int_{t'+\epsilon}^{\infty} dt'' e^{i\omega t''} \mathrm{S}(k_g,t'') \quad (2)$$

and recognizing the products of the integral as partial convolutions and cross-correlations over time, it can be shown that equation (2) can be written in the time domain as:

$$IM_{kt}(k_g,t) = \int_{-\infty}^{\infty} dt' S(k_g,t'-t) \int_{\alpha(t,t')}^{\beta(t)} dt'' S(k_g,t'-t'') S(k_g,t'') \quad (3)$$

where,

$$\alpha(t,t') = t' - (t-\epsilon)$$
$$\beta(t) = t - \epsilon \quad (4)$$

Equation (3) represents the wavenumber-time domain formula as presented by Innanen (2015), equation (4) evokes the limits on the convolution integral to restrict the combination of subevents to a lower-higher-lower relationship. Innanen (2015) produces a mathematical proof of how to transform equation (2) into equation (3).

The Masking Operator

Equation (3) represents a partial convolution, followed by a correlation of the input data. The limits on the second integral of equation (3) can be replaced through the use of a masking operator.

$$IM_{kt}(k_g,t) = \int_{-\infty}^{\infty} dt' S(k_g,t'-t) \int_{-\infty}^{\infty} dt'' [O(t,t',t'') S(k_g,t'-t'') S(k_g,t'') \quad (5)$$

The masking operator "O" consists of two Heaviside step functions that act as the integration limits.

$$O(t,t',t'') = H[t'' - \alpha(t,t')] H[\beta(t) - t''] \quad (6)$$

This masking operator works to remove contributions from the convolution for t" values less than $\alpha(t,t')$ and t'' values above $\beta(t)$. Both of these regions represent violations of the lower-higher-lower relationship, and any values within them must be suppressed.

In matrix notation equation (5) becomes:

$$IM_{kt}(k_g,t) = CCR \cdot (O \circ CNV) \cdot Trace \quad (7)$$

where $CCR, CNV,$ and O are the cross correlation matrix of the trace, the convolution matrix of the trace, and the masking operator matrix respectively, and "$\circ$" represents the Hadamard product. Letting the x-coordinate and the y-coordinate of the masking matrix be t'' and t' respectively, then the limit $t'' = t - \epsilon$ becomes a vertical line in the masking matrix, and the limit $t'' = t' - (t - \epsilon)$ becomes a sloped line. Figure 1 shows a schematic diagram of what the masking matrix looks like, the black regions in figure 1 represent regions of ones, or in other words pass regions, the white regions are regions of zeros, or rejection areas. It can be seen from figure one that the matrix acts as the integration limits rejecting values for which $t'' > t - \epsilon$ and $t" < t' - (t - \epsilon)$. Figure 1 also shows that as the algorithm searches through more of the data, the pass region grows, and the reject region shrinks. When the masking matrix is multiplied by the convolution matrix using the Hadamard product, the convolution becomes a partial convolution over the pass region. Each time sample of the prediction can be represented as:

$$IM_{kt}(k_g,t(j)) = CCR(j,:) \cdot (O(t(j),\epsilon) \circ CNV) \cdot Trace \quad (8)$$

Equation (8) is then repeated for every value of k_g.

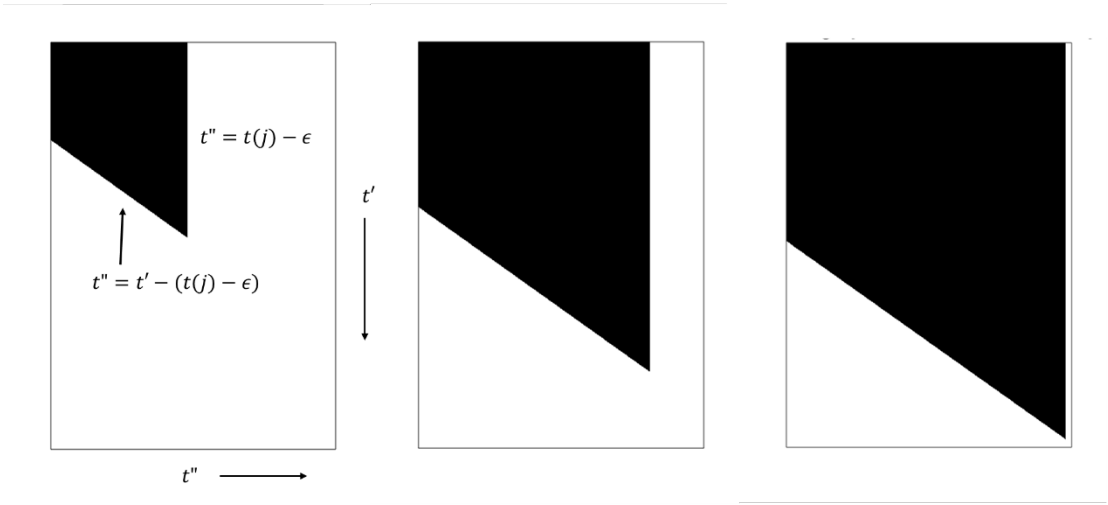

FIG. 1. Schematic diagram of the masking matrix and how it grows with time

PREDICTION IN PYTHON

Data Preparation

The first step for predicting internal multiples is to prepare the data so that they may be used in our desired domain. For some prediction domains this process can involve multiple steps, but in the wavenumber-time domain this is a fairly straightforward process. We assume the data takes the form of a single shot gather of the form $d(x_g, t)$ where x_g is a vector of geophone locations and t is the time vector. Reviewing equation (2) the input data to the algorithm must take the form $S(k_g, t)$, since k_g is the Fourier conjugate of x_g the data can simply be prepared through the use of a Fourier transform over x_g.

$$d(x_g, t) \rightarrow S(k_g, t) \tag{9}$$

Preparation of time vectors and matrices

Before internal multiple prediction can take place the matrices of equation (7) must first be created. In order to build the matrices, it is important to first initialize time vectors, corresponding to the matrix sizes. We will require a vector that is twice the size of our input trace, and is zero padded for negative times, we will call this vector t_{p1}. In addition, we also require a vector that has size, $(2t_{p1} - 1, 1)$ for use in the convolution we will call this (t_{p2}), and another that has size $(3t_{p1} - 2, 1)$ for use in the correlation matrix called (t_o). All of these vectors are created using a Python function called *imtimevectors*. The zero padded trace will be used as an input to the convolution matrix, giving it a size of $(2t_{p1} - 1, t_{p1})$. This zero padded trace is then time reversed and used as an input for the correlation matrix, giving it a size of $(3t_{pi} - 2, t_{p2})$. Both the convolution and correlation matrices are constructed using the *Toeplitz* function in the linear algebra Python library *linalg* contained in *scipy*.

The masking matrix

Creation of the masking matrix takes place inside a created function called *createMask*. This function takes two inputs, one called N_t, and another called t_{INDEX}. N_t is used to

initialize the size of the mask matrix, since the masking matrix is multiplied by the convolution matrix using the Hadamard product, both matrices must have the same size. Therefore, in our case N_t must be t_{p1} so that the mask matrix has a size $(2t_{p1}-1, t_{p1})$. The function input, t_{INDEX} takes the form of $(t-\epsilon)$, from figure 1 it can be seen that when t'' is zero, $t' = t-\epsilon$. Therefore, the variable t_{INDEX}, sets the limits for both the right vertical line of figure 1, and the sloped line. A for loop is used to construct the limits for the sloped line, for the first column of the masking matrix every value from the first row to the row equating to $(t-\epsilon)$ is given the value 1. The loop then updates the column number by 1, and fills in one more row, in this way the sloped limit is created with a slope of 1. The straight line limit can be realized by simply setting every column from the column equating to $(t-\epsilon)$ to the end of the matrix to zero. By setting up these two limits, the reject and pass regions are initialized, for every prediction output the mask matrix grows in size as t increases.

Table 1. Pseudocode for creating mask matrix in Python

For ii in range(1,t_{p1}):
u[1:t_{INDEX}+ii,ii]=np.ones(np.size(u[1:$t_{INDEX}+ii, ii$]))
END

u[:,t_{INDEX}: t_{p1}] = 0

return u

Prediction

With the masking, convolution, and cross-correlation matrices initialized and created the prediction phase may begin. Equation (8) is the mathematical formula that will be used to carry out the prediction in Python. The problem can be made more computationally efficient by limiting the k_g values over which the algorithm occurs, in this example the algorithm computed for every positive value of k_g, predictions for the negative values will be filled in through the use of conjugate symmetry. For every chosen value of k_g the algorithm will run through the trace for every time value, and combine subevents that obey the lower-higher-lower relationship. The start time will be initialized as $3N_t$ and the end time will be the length of t_{p2} due to the size of the cross-correlation matrix. At every j^{th} time sample, a new masking matrix is created, multiplied by the convolution matrix using the Hadamard product, and then multiplied by the row of the cross-correlation matrix relating to the j^{th} time sample; the product of which is then multiplied by the padded input trace. Let bK_g,eK_g be the first and last wavenumbers to perform the prediction on respectively. After the prediction we only need the last samples from 3Nt to the end of the vector, since our start time for the loop began at 3Nt.

Table 2. Pseudocode for the prediction of internal multiples in the wavenumber-time domain

```
For ii in range(bK_g,eK_g):
trace = S(:, N_g/2 + ii)
tracP1=np.concatenate([pad,trace])
CNV = convmtx(traceP1, len(tP1))
traceP1R = np.flipud(traceP1)
CRR = convmtx(traceP1R, len(tP2))
                                    For jj in range(3Nt:len(t_p2))
                                    iie = ii-3Nt
                                    tINDEX = Nt - epsilon[iie]+iie
                                    mask = alberta(len(tP1), tINDEX)
                                     prediction[ii] = (CRR[ii,:], (mask*CNV, traceP1))
```

Complex conjugation and inverse Fourier transform

The last step after the prediction phase is to fill in the negative wavenumber components, and then inverse Fourier transform the data to return it to the offset-time domain. The second half of the vector contains the positive wavenumbers, since the wavenumbers were shifted to center the wavenumber vector at zero. Therefore, the first half of the vector is to be filled in with the complex conjugates of the positive wavenumber predictions. The wavenumbers are then shifted back so that zero occurs at the start of the vector, this can be achieved with the Python numpy function *np.fft.ifftshift*. Once this has occurred, *np.fft.ifft* can be used to inverse Fourier transform the data back to the offset-time domain.

SYNTHETIC EXAMPLE

For 1.5D internal multiple prediction it is assumed that the input data is a single shot record acquired over a layered, horizontally homogenous geology. Figure 2 below shows the velocity model that was used to generate the synthetic data. This velocity model provides a simple example of the powerful nature of the inverse scattering series prediction algorithm.

Figure 3, shows the data that was created from the velocity model in figure 2. This data was created using the CREWES acoustic finite difference algorithm *afd_shotrec* contained in the CREWES toolbox. The data has had the direct arrival removed, and was given absorbing boundaries to limit the effects of ghosts and surface multiples. The first event seen is the primary from the first interface, while the second event is the primary from the second interface. The event with a zero-offset traveltime near 1.6s is a first order internal multiple, while the event with a traveltime of approximately 2s is a second order internal multiple.

Figure 4, shows the result of Fourier transform over x_g, the result of figure 4 is the image representation of $S(k_g, t)$. Figure 5, shows the result of carrying out the outlined prediction algorithm contained in table 2.

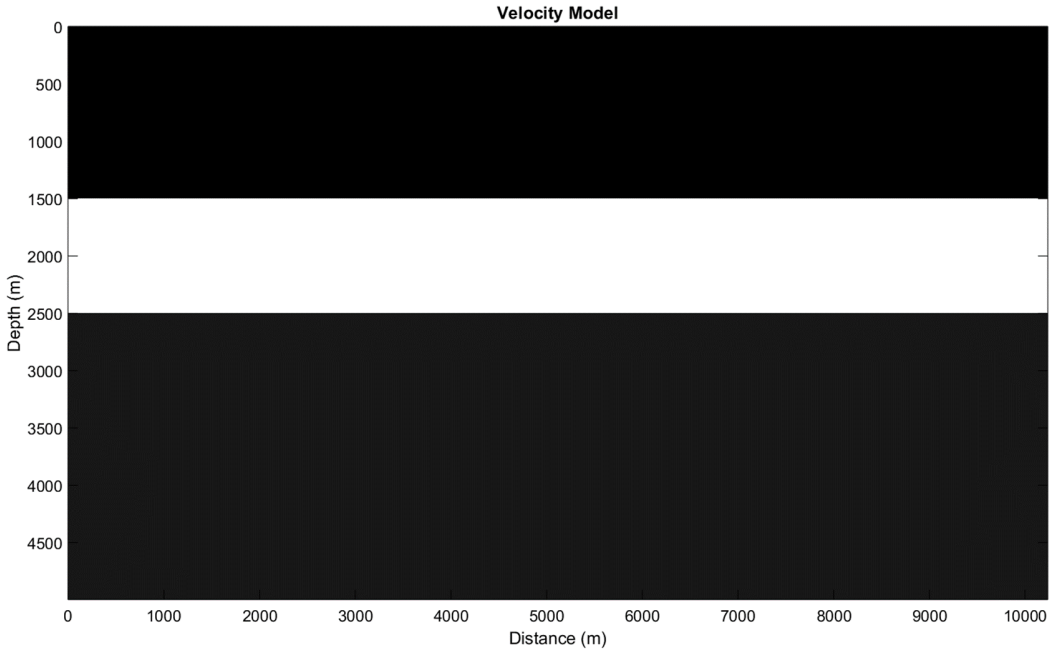

FIG. 2. Velocity model used in the 1.5D wavenumber-time domain prediction

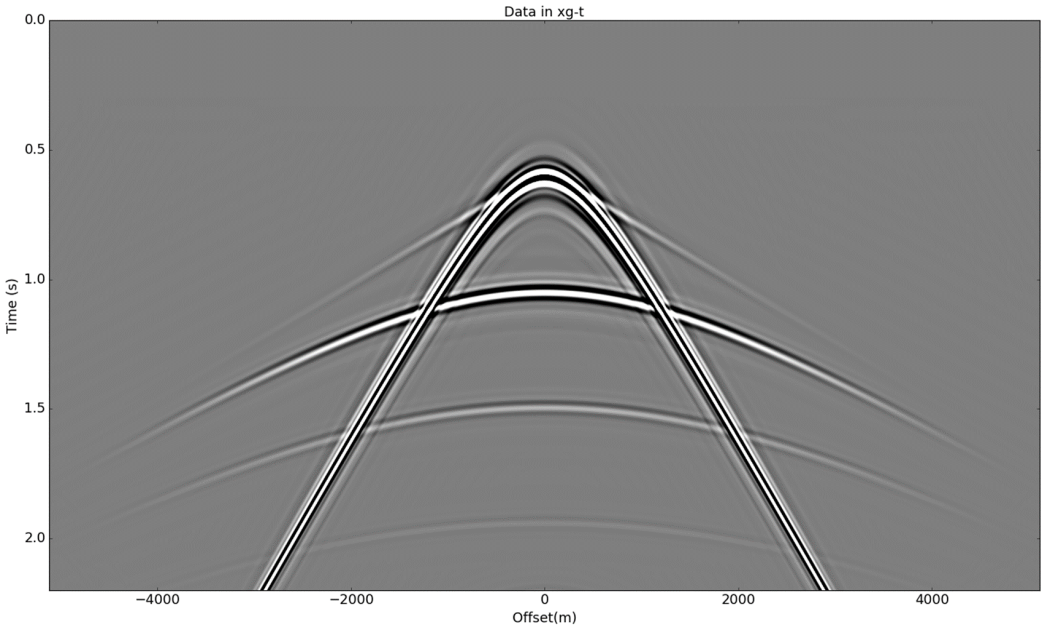

FIG. 3. Synthetic data set used in the 1.5D wavenumber-time prediction algorithm.

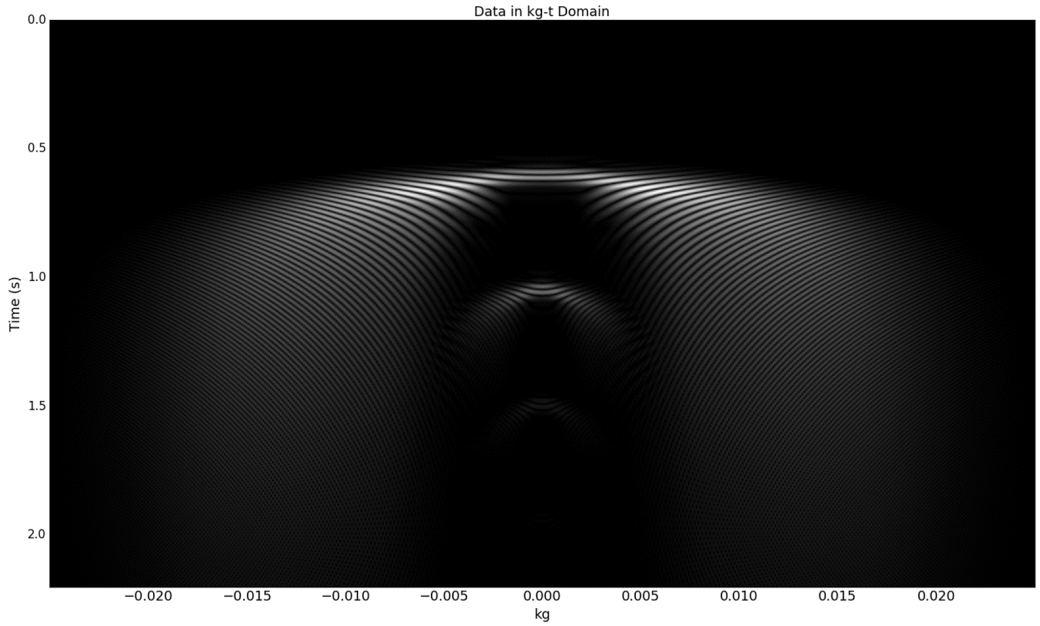

FIG. 4. Synthetic data after transformation to the wavenumber-time domain

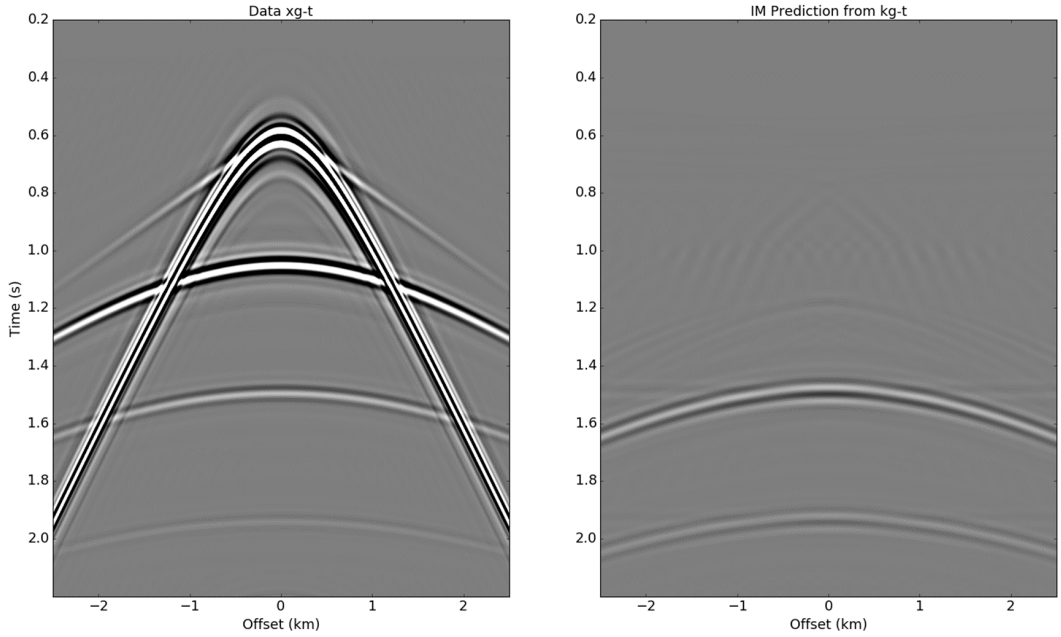

FIG. 5. Data from figure 3 (left), and the resulting internal multiple prediction (right).

CONCLUSION

Weglein et al. (1997) introduced a method of internal multiple prediction based on the inverse scattering series, that was fully data. The original algorithm derived by Weglein et al. performed the prediction the wavenumber-frequency domain. Since then the CREWES project has investigated inverse scattering internal multiple prediction in numerous domains in order to investigate of an optimal prediction domain exists. One such domain

is the wavenumber-time domain. Innanen (2015) presented the 1D time domain prediction algorithm as well as extensions to the wavenumber-time, and offset-time domains. This work expanded on the work presented by Innanen (2015), by explaining how to implement the algorithm within the Python programming language. A synthetic example was then shown to provide a simple demonstration of how the algorithm performs.

Figure 5 shows the input data created from the velocity model in figure 3, as well as the resulting prediction. Figure 5 shows that the prediction algorithm originally presented by Innanen, and explained in further detail in this paper, accurately predicts the traveltime, and approximately predicts the amplitude of the two internal multiples shown.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr. Kristopher Innanen for his support and guidance. The authors also thank the CREWES sponsors and NSERC through grant CRDPJ 461179-13 for supporting this work.

REFERENCES

Alam, A. and Austin, J., 1981, Multiple attenuation using slant stacks: Tech. Rep., Western Geophysical Company.

Berkhout, A. J., 1999, Multiple removal based on the feedback model: The Leading Edge, **18**, 127–131.

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

Jakubowicz, H., 1998, Wave equation prediction and removal of interbed multiples: Soc. Expl. Geophys. Internat. Exp. and 68th Annual Mtg., 1527-1530.

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

Treitel, S., Gutowski, P., Wagner, D. E., 1982, Plane wave decomposition of seismograms: Geophysics, **47**, 1372-1401.

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.

Yilmaz, O., 2001, Seismic data analysis, Soc. Expl. Geophys.