Implementation of the marchenko method

Thorbecke, Jan Willem; Slob, Evert; Brackenhoff, Joeri; van der Neut, Joost; Wapenaar, Kees

DOI
10.1190/GEO2017-0108.1

Publication date
2017

Document Version
Publisher’s PDF, also known as Version of record

Published in
Geophysics

Citation (APA)

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.
Implementation of the Marchenko method

Jan Thorbecke¹, Evert Slob¹, Joeri Brackenhoff¹, Joost van der Neut¹, and Kees Wapenaar¹

ABSTRACT

The Marchenko method makes it possible to compute subsurface-to-surface Green’s functions from reflection measurements at the surface. Applications of the Marchenko method have already been discussed in many papers, but its implementation aspects have not yet been discussed in detail. Solving the Marchenko equation is an inverse problem. The Marchenko method computes a solution of the Marchenko equation by an (adaptive) iterative scheme or by a direct inversion. We have evaluated the iterative implementation based on a Neumann series, which is considered to be the conventional scheme. At each iteration of this scheme, a convolution in time and an integration in space are performed between a so-called focusing (update) function and the reflection response. In addition, by applying a time window, one obtains an update, which becomes the input for the next iteration. In each iteration, upgoing and downgoing focusing functions are updated with these terms. After convergence of the scheme, the resulting upgoing and downgoing focusing functions are used to compute the upgoing and downgoing Green’s functions with a virtual-source position in the subsurface and receivers at the surface. We have evaluated this algorithm in detail and developed an implementation that reproduces our examples. The software fits into the Seismic Unix software suite of the Colorado School of Mines.

INTRODUCTION

The Marchenko method relates Green’s function from a virtual source inside a medium to the reflection response at the surface of that medium. The method has been introduced to the geophysical world by Broggin and Snieder (2012). In close collaboration with these authors, their work is extended to 2D and 3D media by Broggin et al. (2012) and Wapenaar et al. (2013). Based on the outputs of the Marchenko method, upgoing and downgoing Green’s functions can be estimated, for any point in the subsurface to the surface array, using the reflection response recorded at the surface. The Green’s function is the earth impulse response and is fundamental in many processing schemes.

The application of the results of the Marchenko method is therefore manifold: It has been used for imaging the subsurface without the disturbing effect of internal multiples (Behura et al., 2014; Slob et al., 2014; Wapenaar et al., 2014b; da Costa Filho et al., 2015; van der Neut et al., 2015c; Vasconcelos et al., 2015; Meles et al., 2016; Ravasi et al., 2016), internal multiple elimination in the data domain (Meles et al., 2015; van der Neut and Wapenaar 2016; da Costa Filho et al., 2017), and retrieving the homogeneous Green’s function between any two points inside a medium from the reflection response (Wapenaar et al., 2017).

In the first geophysical Marchenko papers, the computation of the Green’s function is based on iteratively updating acoustic pressure fields (Wapenaar et al., 2013). This pressure-based algorithm requires two separate iterative updates to calculate the upgoing or downgoing Green’s functions at a virtual source position. Slob et al. (2014) combine these separate iterative schemes into one. In this combined scheme, upgoing and downgoing focusing functions are updated in each iteration. Wapenaar and Slob (2014) and da Costa Filho et al. (2014) extend the method to elastic media, Singh et al. (2015) include the free-surface multiples, and Slob (2016) adapts the method to dissipative acoustic media. Apart from iterative schemes, the Marchenko method can also be implemented as an adaptive subtraction (Staring et al., 2016) or a least-squares inversion (van der Neut et al., 2015b; Slob and Wapenaar, 2017).

In this paper, we describe in detail the implementational aspects of the 2D iterative acoustic-Marchenko method based on focusing functions. Although the algorithm is straightforward to implement, the treatment of amplitudes and the initialization steps of the first iterations require special attention. The input of the method is a...
reflection response without free-surface multiples, and it is deconvolved by its source wavelet. The output of a surface-related multiple elimination (SRME) scheme can (in principle) provide this reflection response. A (smooth) background model is needed to calculate an initial focusing function to start the algorithm. The “Numerical examples” section demonstrates the use of the algorithm and provides a user’s first steps with the Marchenko technique.

The software bundled with this paper contains all source code and scripts to reproduce the examples presented herein. The code can also be found at its GitHub repository (Thorbecke, 2017). The GitHub repository contains the most up-to-date stable version with bug fixes and the latest developments. In Appendix A, the input parameters of the programs are explained. To reproduce the figures in this manuscript and to carry out a few post- and preprocessing steps, Seismic Unix (SU) (Cohen and Stockwell, 2016) is required.

**MARCHENKO METHOD**

The Marchenko method is briefly introduced here, aiming at an explanation of the method that helps to understand the algorithm. The references mentioned in the “Introduction” section provide additional details on the derivation of this method. In an imagined observation point in the medium, and depth level downgoing parts of the above the surface boundary repository contains the most up-to-date stable version with bug fixes. The code can also scripts to reproduce the examples presents herein. The code can also be required.

\[ f_1(x, x_F, t) = f_1^+(x, x_F, t) + f_1^-(x, x_F, t), \] (1)

where \( x_F = (x_F, z_i) \) is a focal position on the boundary \( \partial \mathbb{D}_i \), \( x \) is an observation point in the medium, and \( t \) is the time (see Figure 1). In our notation, the first argument represents the receiver location and the second argument stands for the focal point. The superscript “+” in \( f_1^+ \) denotes a downgoing field at observation point \( x \), and the superscript “−” in \( f_1^- \) denotes an upgoing field, also at \( x \). Below boundary \( \partial \mathbb{D}_0 \) only \( f_1^+ \) continues as a diverging downgoing field into the reflection-free half-space.

The \( f_1^+ \) focusing functions are defined to relate the upgoing and downgoing Green’s functions in the actual medium with the reflection response at the surface (Wapenaar et al., 2014b):

\[ G^+(x_F, x_R, t) = \int_{\partial \mathbb{D}_0} \int_{t'=\infty}^t \frac{R(x_F, x, t-t') f_1^+(x, x_F, -t)}{\rho(x)} \, dt' \, dx + f_1^+(x_R, x_F, -t), \] (2)

\[ G^-(x_F, x_R, t) = \int_{\partial \mathbb{D}_i} \int_{t'=\infty}^t \frac{R(x_F, x, t-t') f_1^+(x, x_F, -t)}{\rho(x)} \, dt' \, dx + f_1^-(x_R, x_F, -t), \] (3)

where \( R(x_R, x, t) \) is the reflection response after surface multiple elimination, deghosting, and deconvolution of the wavelet. The first argument in \( R \) represents the receiver location, the second argument represents the source location, and the last argument is the time. The function \( R \) is a scaled (factor –2) pressure wavefield at \( x_R \) of a vertical force source \( F_z \) at \( x \), or, via reciprocity, the particle velocity field \( V_z \) at \( x \) of a point source of the volume injection rate at \( x_R \) (Wapenaar et al., 2012). This reflection response is related to a Green’s function by

\[ \frac{\partial R(x_R, x, t)}{\partial t} = \frac{2}{\rho(x)} \frac{\partial G^+(x_R, x, t)}{\partial z}, \] (4)

with \( G^+ \) the Green’s function of the scattered field only (it does not contain the direct field). The upper integration boundary \( t' = t \) of the time integral in equations 2 and 3 accounts for the causality of the reflection response \( R(x_F, x, t-t') \). Summing equations 2 and 3 and using source-receiver reciprocity for Green’s function gives (Wapenaar et al., 2017)

\[ G(x_R, x_F, t) = \int_{\partial \mathbb{D}_0} \int_{t'=\infty}^t \frac{R(x_R, x, t-t') f_2(x_F, x, t')}{\rho(x)} \, dt' \, dx + f_2(x_F, x_R, -t), \] (5)

where the Green’s function \( G(x_R, x_F) \) represents the response to a virtual point source of volume injection rate at \( x_F \) and pressure receivers at the surface \( x_R \). The focusing function \( f_2 \) is defined as

\[ f_2(x_F, x, t) = f_1^+(x, x_F, t) - f_1^-(x, x_F, t), \] (6)

Wapenaar et al. (2014b) introduce \( f_2(x_F, x, t) \) as a focusing function that has its focal plane at \( \partial \mathbb{D}_0 \). Here, we merely use \( f_2 \) as a compact notation for the combination of the one-way focusing functions \( f_1^+ \) and \( f_1^- \), as defined in equation 6. Note that the time-reversed upgoing function \( f_1^- \) can be interpreted as a downgoing function, similar to \( f_1^+ \). Hence, from here onward, we interpret \( f_2(x_F, x, t) \) as a downgoing focusing function that is emitted into the medium from \( x \) and that focuses at receiver location \( x_F \). The argument change in equation 6 between \( x \) and \( x_F \) in the left side \( f_2 \) and the right side \( f_1^+ \) follows from the same logic in the order of the arguments as defined by Wapenaar et al. (2014b). In the “Numerical examples” section, we demonstrate that \( f_2 \) can be back-propagated into the medium and focuses at \( x_F \). In Wapenaar
et al. (2013), a (reciprocal) relation between $f_2(x_F, x, t)$ and a downgoing wavefield $p^-(x, x_F, t)$ is given. Together with $p^+$ there is also a $p^-$ defined, the upgoing reflection response at $x$ from the focal point at $x_F$. The sum of $p^+$ and $p^-$ gives also the Green’s function of equation 5. The $p^+$ functions are just a different notation of the Marchenko method and can be used to compute the Green’s functions in a convenient way. These $p^+$ functions are therefore used in the implementation to compute the Green’s function. From an educational point of view, the Marchenko method is more easily understood using the focusing functions only, and we will continue along that way.

The above equations, on which the following implementation is based on, use pressure-normalized fields. Other papers derive similar equations based on flux-normalized fields (Wapenaar et al., 2014a; Singh et al., 2015; van der Neut et al., 2015b). The relationship between pressure- and flux-normalized representations is explained by Wapenaar et al. (2014a).

The Marchenko algorithm estimates focusing functions $f_1^+(x, x_F, t)$ and $f_1^-(x, x_F, t)$. However, equations 2 and 3 are by themselves insufficient to determine $f_1^+$; there are four unknowns, but only two equations. We can eliminate two unknowns by exploiting the fact that the focusing functions and Green’s functions have different causality properties in the time-space domain. Based on the principle of causality, we know that no energy arrives before the first arrival $t_d(x_F, x_F)$; hence, the Green’s function $G(x_F, x_F, t < t_d(x_F, x_F))$ is zero. This also holds for the upgoing and downgoing Green’s functions and leads to

$$0 = - \int_{t_0}^{t} \int_{t'=-\infty}^{t} R(x_R, x, t-t')f_1^+(x, x_F, t')dt'dx + f_1^+(x_R, x_F, t),$$

$$0 = \int_{-\infty}^{t} \int_{t'=-\infty}^{t} R(x_R, x, t-t')f_1^+(x, x_F, t')dt'dx - f_1^+(x_R, x_F, t),$$

where $t < t_d(x_F, x_F)$ in both equations above.

The combination of equations 7 and 8 is known as the Marchenko equation. These equations form the basis of the iterative scheme, which estimates focusing functions $f_1^+$ and $f_1^-$. In Wapenaar et al. (2014a), the relation

$$f_1^+(x, x_F, t) = T_{inv}^+(x_F, x, t)$$

is used to derive an initial estimate for $f_1^+$ that can start the inversion scheme. In equation 9, $T_{inv}^+(x_F, x, t)$ is the inverse of the transmission response of the truncated medium, which is equal to the actual medium between $d_{in}$ and $d_{out}$, for a source at $x$ (at $d_{out}$) and a receiver at $d_{in}$. It is assumed that this inverse transmission response $T_{inv}^+$ can be composed as a direct wave followed by scattering coda (van der Neut et al., 2015b):

$$f_1^+(x, x_F, t) = T_{inv}^+(x_F, x, t) + M^+(x, x_F, t),$$

where $M^+$ is the unknown scattering coda and $T_{inv}^+$ is the direct arrival of the inverse transmission response. In equation 10, the inverse of the direct arrival of the transmission response is needed. For simplicity, we take the time reversal of the direct arrival of Green’s function: $G_d(x, x_F, -t)$:

$$f_1^+(x, x_F, t) \approx G_d(x, x_F, -t) + M^+(x, x_F, t).$$

We thereby introduce an overall scaling error and an offset-dependent amplitude error, proportional to transmission losses, in the final result. The function $G_d(x, x_F, -t)$ is the time-reversed direct arrival part of the transmission response to subsurface focal point $x_F$ and can, for example, be computed from a smooth macromodel. As mentioned before, the arrival time of $G_d(x, x_F, t)$ is $t_d$; hence, $G_d(x, x_F, t)$ is zero before $t < t_d$. The multiple scattering coda $M^+(x, x_F, t)$ follows after the first arrival of $f_1^+$, and it is zero for $t \leq -t_d$. It can also be shown that it is also zero for $t \geq +t_d$ (Slob et al., 2014).

Equations 7 and 8 are only valid for $t < t_d$. Therefore, we define a time-window function:

$$\theta(x_R, x_F, t) = \begin{cases} 
1 & t \leq t'_d \\
\frac{t}{t'_d} & t = t'_d \\
0 & t > t'_d
\end{cases}$$

where $t'_d$ is the time of the direct arrival from the focal point $x_F$ to $x_R$ ($t_d$), minus a small positive constant $\epsilon$ to exclude the wavelet in the direct arrival $G_d$. For example, a time window that sets all times $t < -t'_d$ to zero and applied to equation 11 mutates $G_d(-t)$, but it leaves all events of $M^+$ in. In the following, we will use the shorthand notation $\theta(x_F, x_F, t)$ for $\theta(x_R, x_F, t)$. In the included Marchenko program, there is an input parameter (called smooth; see Appendix A for all input parameters) that defines a temporal tapering in this mute window to suppress high-frequency artifacts.

It is further assumed that it is possible to get an estimate of this direct arrival $G_d$ of the transmission response. Given the reflection response $R(x_R, x, t)$ and this direct arrival $G_d(x, x_F, t)$ from the focal point, the Marchenko algorithm solves for the scattering coda $M^+(x, x_F, t)$ to estimate $f_1^+(x, x_F, t)$ and $f_1^-(x, x_F, t)$.

The iterative solution of the Marchenko equations can now be developed. The iterative scheme is started with the following initialization of $M^+$:

$$M_{0}^+(x_R, x_F, t) = 0.$$ (13)

The subscript in $M_{0}^+$ defines the iteration number. By substituting equation 11, using 13 as an initialization, into equation 8 one arrives at the initialization of $f_1^+$:

$$f_{1,0}^+(x_R, x_F, t) = \theta(x_R, x_F, t) \int_{d_{in}}^{d_{out}} \int_{t'=-\infty}^{t} R(x_R, x, t-t')G_d(x, x_F, -t')dt'dx.$$ (14)

Equation 14 includes the previously defined time-window function $\theta$, Equations 7 and 11 are expressions for $f_1^+$. By combining these equations, the only part remaining in equation 11 is $M^+$. The iterative update of $M^+$ for step $k \geq 1$ is given by

$$M_{k}^+(x_R, x_F, t) = \theta(x_R, x_F, t) \int_{d_{in}}^{d_{out}} \int_{t'=-\infty}^{t} R(x_R, x, t-t')f_{1,k-1}^+(x, x_F, -t')dt'dx.$$ (15)
Following the assumption in equation 11, that it is possible to write \( f_{1,k}^{+} \) as a direct field plus scattering coda, the update at step \( k \) of \( f_{1,k}^{+} \) is given by

\[
f_{1,k}^{+}(x_R, x_F, t) = G_d(x_R, x_F, -t) + M_k^+(x_R, x_F, t). \tag{16}
\]

Using equation 8 and the expression of \( f_1^{+} \) in equation 16, the update of \( f_{1,k}^{+} \) at step \( k \) is given by

\[
f_{1,k}^{+}(x_R, x_F, t) = f_{1,0}^{+}(x_R, x_F, t) + \theta \int_{-\infty}^{t} \int_{t'} R(x_R, x, t-t') M_k^+(x, x_F, t-t') dt' dx. \tag{17}
\]

This completes the definition of the iterative Marchenko scheme. In the next section, the first few iterations are discussed in detail and illustrated with simple numerical examples.

**Marchenko Algorithm**

To compute \( f_1 \), focusing functions with the Marchenko method, two ingredients are needed:

- Reflection data without free-surface multiples, ghosts and deconvolved for the wavelet: \( R(x_R, x, t) \) with source \( x \) and receiver \( x_R \) on the same surface \( \partial \Omega_0 \), and small enough sampling for \( x_R \) and \( x \) to avoid spatial aliasing.
- An estimate of the direct arrival between the receiver positions at the surface \( (x_R) \), and the focal point at \( x_R \): \( G_d(x_R, x_F, t) \), and derived from it the direct arrival time \( t_d(x_R, x_F, t) \). Note that \( t_d \) can also be computed by another method, for example, an eikonal solver.

Given these two components, the iterative method can be initialized and the iterations of the Marchenko method can start.

**The First Few Iterations**

The initialization of the method is given in equations 13 and 14. The time-windowed expression for \( f_{1,0}^{+}(x_R, x_F, t) \) in equation 14 is renamed to

\[
-N_0(x_R, x_F, -t) = \theta \int_{-\infty}^{t} \int_{t'} R(x_R, x, t-t') G_d(x, x_F, -t') dt' dx. \tag{18}
\]

At each iteration, the spatial integration and temporal convolution with \( R \) plays an important role because it is used to define new focusing updates given by terms \( N_i \) (see also appendix B of Wapenaar et al., 2014b). These \( N_i \) terms are used to update the estimates of the focusing functions \( f_1^{+} \) and \( f_1^{-} \). Although the \( N_i \) terms are not strictly needed to describe the method, they are introduced here to remain as close as possible to the actual implementation.

For computational efficiency, the temporal convolution of \( R \) is implemented in the Fourier domain. The spatial integration is carried out by summing the resulting traces of the time convolution over a common-receiver gather. The introduced time-window sets events for \( t > t_d \) to zero, in accordance with equation 18. Applying the mute window is therefore a crucial and mandatory step in the Marchenko method; without it, the method would be incorrect.

Given these initializations, the first step in the algorithm, based on equations 15–17, can be computed. This first step, \( k = 1 \), involves two integration-convolutions with \( R \) to update \( f_1^{+} \) and \( f_1^{-} \):

\[
M_1^+(x_R, x_F, -t) = \theta \int_{-\infty}^{t} \int_{t'} R(x_R, x, t-t') f_{1,0}^{+}(x, x_F, -t') dt' dx
= -\theta \int_{-\infty}^{t} \int_{t'} R(x_R, x, t-t') N_0(x, x_F, t') dt' dx
= N_1(x_R, x_F, -t). \tag{19}
\]

\[
f_{1,1}^{+}(x_R, x_F, t) = G_d(x_R, x_F, -t) + M_1^+(x_R, x_F, t)
= G_d(x_R, x_F, -t) + N_1(x_R, x_F, t). \tag{20}
\]

\[
f_{1,1}^{-}(x_R, x_F, t) = f_{1,0}^{+}(x_R, x_F, t) + \theta \int_{-\infty}^{t} \int_{t'} R(x_R, x, t-t') M_1^+(x, x_F, t-t') dt' dx
= -N_0(x_R, x_F, -t) + \theta \int_{-\infty}^{t} \int_{t'} R(x_R, x, t-t') N_1(x, x_F, t-t') dt' dx
= -N_0(x_R, x_F, -t) - N_2(x_R, x_F, -t). \tag{21}
\]

\[
f_{2,1}^{+}(x_F, x_R, t) = G_d(x_F, x_R, -t) + N_0(x_R, x_F, t)
= G_d(x_F, x_R, -t) + N_1(x_R, x_F, t) + N_2(x_R, x_F, t). \tag{22}
\]

The first integration-convolution with \( R \) in equation 19 is used to update \( f_2^{+} \) as shown in equation 20. The second integration-convolution in equation 21 updates \( f_{1,1}^{-} \). The update of \( f_{2,1}^{+} \) introduced in equation 6, includes the results of all integration-convolutions with \( R \).

The next step for \( k = 2 \) results in the following updates:

\[
M_2^+(x_R, x_F, -t) = \theta \int_{-\infty}^{t} \int_{t'} R(x_R, x, t-t') f_{1,1}^{+}(x, x_F, -t') dt' dx
= -\theta \int_{-\infty}^{t} \int_{t'} R(x_R, x, t-t') \{N_0(x, x_F, t) + N_2(x, x_F, t)\} dt' dx
= N_1(x_R, x_F, -t) + N_3(x_R, x_F, -t), \tag{23}
\]

\[
f_{1,2}^{+}(x_R, x_F, t) = G_d(x_R, x_F, -t) + M_2^+(x_R, x_F, t)
= G_d(x_R, x_F, -t) + N_1(x_R, x_F, t) + N_3(x_R, x_F, t). \tag{24}
\]
\[ f_{1,2}(x_R, x_F, t) = f_{1,0}(x_R, x_F, t) + \theta \int_{d \in \partial \Omega} \int_{t'} R(x_R, x, t-t')M^2_{\alpha}(x, x_F, t') \, dt' \, dx \]
\[ = -N_0(x_R, x_F, -t) + \theta \int_{d \in \partial \Omega} \int_{t'} R(x_R, x, t-t') \times \{ N_1(x, x_F, t) + N_3(x, x_F, t') \} \, dt' \, dx \]
\[ = -N_0(x_R, x_F, -t) - N_2(x_R, x_F, -t) - N_4(x_R, x_F, -t), \]  
(25)

\[ f_{2,2}(x_R, x_F, t) = G_d(x_R, x_F, -t) + N_0(x_R, x_F, t) + N_1(x_R, x_F, t) + N_2(x_R, x_F, t) + N_3(x_R, x_F, t) + N_4(x_R, x_F, t). \]  
(26)

From these updates, it becomes clear that in updating \( f_1^+ \) in equation 24 \( G_d \) and the odd terms of \( N_i \) are used and in updating \( f_1^- \) in equation 25 the even terms of \( N_i \) are used. The \( f_2 \) function in equation 26 is built up from \( G_d \) and even and odd \( N_i \) terms.

In the implementation, the \( N_i \) terms are computed by

\[ N_{l-1}(x_R, x_F, -t) = G_d(x_R, x_F, -t'), \]  
(27)

\[ N_{l}(x_R, x_F, t) = -\theta \int_{d \in \partial \Omega} \int_{t'} R(x_R, x, t-t') N_{l-1}(x, x_F, t') \, dt' \, dx, \]  
(28)

and are used to update the focusing functions \( f_1^+, f_1^- \), and \( f_2 \). This makes the algorithm simple and efficient. In summary, the relations for \( M^\pm_{\alpha} \), \( N_i \) and the updates for the focusing functions for \( m \) iterations with \( m \geq 1 \) are

\[ M^\pm_{\alpha}(x_R, x_F, t) = \sum_{l=0}^{m-1} N_{2l+1}(x_R, x_F, t), \]  
(29)

\[ f_{1,m}^+(x_R, x_F, t) = G_d(x_R, x_F, -t) + \sum_{l=0}^{m-1} N_{2l+1}(x_R, x_F, t), \]  
(30)

\[ f_{1,m}^-(x_R, x_F, t) = -\sum_{l=0}^{m} N_{2l}(x_R, x_F, -t), \]  
(31)

\[ f_{2,m}(x_F, x_R, t) = G_d(x_R, x_F, -t) + \sum_{l=0}^{2m} N_{l}(x_R, x_F, t). \]  
(32)

In the provided program, each computation of a focusing update term \( N_i \) is called one iteration. The implementation is shown in Algorithm 1, and a flowchart is shown in Figure 2.

The initializations of \( f_1^+, f_1^-, f_2 \), and \( N_i \) are done just before the iteration starts. The even and odd iterations for \( N_i \) update \( f_1^+ \) and \( f_1^- \), respectively. The Green’s function is computed by inserting the estimate of \( f_2 \) given by equation 32 into equation 5:

\[ G(x_F, x_R, t) = f_2(x_F, x_R, -t) + \int_{d \in \partial \Omega} \int_{t'} R(x_R, x, t-t') G_d(x, x_F, -t) \, dt' \, dx \]
\[ + \sum_{l=0}^{2m} \int_{d \in \partial \Omega} \int_{t'} R(x_R, x, t-t') N_{l}(x, x_F, t') \, dt' \, dx. \]  
(33)

**Algorithm 1. The Marchenko algorithm as implemented in the provided source code.**

<table>
<thead>
<tr>
<th><strong>Main begin</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Reading SU-style input Data and Allocate arrays</td>
</tr>
<tr>
<td>Initialization</td>
</tr>
<tr>
<td>( \text{Ni}(t) = f_2(p(t)) = f_1\text{plus}(t) = G_d(t) )</td>
</tr>
<tr>
<td>( \text{f1min}(t) = \text{pmin}(t) = 0.0 )</td>
</tr>
<tr>
<td><strong>for iter=0 to niter do</strong></td>
</tr>
<tr>
<td>synthesis(Refl, Ni, iRN)</td>
</tr>
<tr>
<td>( \text{Ni}(t) = -\text{iRN}(t) )</td>
</tr>
<tr>
<td>( \text{pmin}(t) += \text{iRN}(t) )</td>
</tr>
<tr>
<td>applyMute(Ni, muteW)</td>
</tr>
<tr>
<td>( f_2(p(t) += \text{Ni}(t) )</td>
</tr>
<tr>
<td><strong>else if (iter % 2 == 0) then</strong></td>
</tr>
<tr>
<td>( \text{f1min}(t) += \text{Ni}(t) )</td>
</tr>
<tr>
<td><strong>else</strong></td>
</tr>
<tr>
<td>( \text{f1plus}(t) += \text{Ni}(t) )</td>
</tr>
<tr>
<td><strong>end</strong></td>
</tr>
<tr>
<td><strong>end</strong></td>
</tr>
<tr>
<td>Green(t) = pmin(t) + f2p(t)</td>
</tr>
<tr>
<td><strong>end</strong></td>
</tr>
<tr>
<td>synthesis(Refl, Ni, iRN)</td>
</tr>
<tr>
<td><strong>begin</strong></td>
</tr>
<tr>
<td>( \text{iRN} = 0 )</td>
</tr>
<tr>
<td>( \forall l, i: \text{Fop}(l, o, i) = F { \text{Ni}(l, i, t) } )</td>
</tr>
<tr>
<td><strong>for k=0 to nshots do</strong></td>
</tr>
<tr>
<td>#pragma omp parallel for</td>
</tr>
<tr>
<td><strong>for l=0 to Nfoc do</strong></td>
</tr>
<tr>
<td>( \text{Fop}(l, o, i) = F { \text{Ni}(l, i, t) } )</td>
</tr>
<tr>
<td><strong>for o=o_min to o_max do</strong></td>
</tr>
<tr>
<td>( \text{sum}(o) = 0 )</td>
</tr>
<tr>
<td><strong>for i=0 to nrecv do</strong></td>
</tr>
<tr>
<td>( \text{sum}(o) = \text{Refl}(k, o, i) * \text{Fop}(l, o, i) )</td>
</tr>
<tr>
<td><strong>end</strong></td>
</tr>
<tr>
<td><strong>end</strong></td>
</tr>
<tr>
<td>( \text{iRN}(l, k, t) = F^{-1} { \text{sum}(o) } )</td>
</tr>
<tr>
<td><strong>end</strong></td>
</tr>
<tr>
<td><strong>end</strong></td>
</tr>
</tbody>
</table>
In equation 33, the integration-convolution terms can be recognized as the summation of the unmuted $N_i$ terms. By storing the sum of these unmuted terms of the integration-convolution (in $p^-$), the Green’s functions can be calculated as a summation of previously computed values.

The program can compute the results of multiple focal points at the same time ($N_{\text{foc}}$ in Algorithm 1). This is convenient for calculating the Marchenko results (e.g., the Green’s function) on a depth level or area of interest in one run. The computational advantage is that the reflection response has to be read in only once to compute the results of multiple focal points. The computations for different focal points are independent of each other. Hence, the code is OpenMP parallelized over the number of focal points ($N_{\text{foc}}$).

The function $\text{synthesis}$ in Algorithm 1 computes the integration-convolution, of the focusing update term $N_i$ with $R$, in the frequency domain (the Fourier operator is denoted with $\mathcal{F}$). For the computation of only one focal point, loading the required input data into memory usually takes more time than the computational work. The implementation has additional functionality (not shown in Algorithm 1) to compute the upgoing and downgoing Green’s functions in equations 2 and 3 and write intermediate computed fields ($N_i$) to disk.

![Flowchart of the Marchenko algorithm. In the notation, the lateral coordinates are omitted for a more compact notation. The symbol $\otimes$ represents the integration-convolution operator.](image)

**NUMERICAL EXAMPLES**

To use the Marchenko method with numerically modeled data, it is very important that the amplitudes of the reflection response are correct. This is certainly also true for field data. Therefore, the importance of amplitude scaling is explained first before discussing the numerical examples in more detail.

In the summation of $N_1$ and $G_1$ to compute $f_{1,1}^+$ in equation 20, it is important that the amplitude of the measured reflection data is accurate. A wrong amplitude of $R$ will result in a wrong amplitude of $f_{1,1}^+$ and the scheme will not converge. This is illustrated with the following equations. Let us assume that we introduce a wrong scaling factor $b$ in $R$ to update $f_{1,1}^+$. Then, the first iterations will compute

$$-bN_0(x_R, x_F, -t) = \theta_1 \int_{\partial\Omega_R} \int_{t'} b R(x_R, x, t-t') \times G_d(x, x_F, -t') \, dt \, dx,$$

$$-b^2 N_1(x_R, x_F, -t) = \theta_1 \int_{\partial\Omega_R} \int_{t'} b R(x_R, x, t-t') \times b N_0(x_R, x_F, t) \, dt \, dx,$$

$$f_{1,1}^+(x_R, x_F, t) = G_d(x_R, x_F, t) + b^2 N_1(x_R, x_F, t).$$

(34)

The update of $f_{1,1}^+$ involves an error of $b^2$ and in each next update of $f_{1,m}^+$ the error in the update $N_{2m+1}$ will grow with $b^{2m+2}$. An incorrect amplitude in $G_d$ is not a problem because the Marchenko equations are linear in the focusing function. An amplitude error can be factored out, and it does not change for higher iterations:

$$-aN_0(x_R, x_F, -t) = \theta_1 \int_{\partial\Omega_R} \int_{t'} R(x_R, x, t-t') a G_d(x, x_F, -t') \, dt \, dx,$$

$$-aN_1(x_R, x_F, -t) = \theta_1 \int_{\partial\Omega_R} \int_{t'} R(x_R, x, t-t') a N_0(x, x_F, t') \, dt \, dx,$$

$$af_{1,1}^+(x_R, x_F, t) = a G_d(x_R, x_F, t) + a N_1(x_R, x_F, t).$$

(35)

Van der Neut et al. (2015c) introduces an adaptive amplitude-correction factor to correct for possible amplitude errors in $R$. By solving the Marchenko equation in an explicit series, the sensitivity of amplitude errors can be adjusted by adaptive subtraction of the focusing update terms. This approach makes it better suited to apply to field data (van der Neut et al., 2015a; Staring et al., 2016).

Brackenhoff (2016) and Thomsen (2016) develop estimation methodologies for a constant scaling factor $b$ of $R$. These methods compensate for an overall amplitude mismatch in $R$, which is an important first step to apply the Marchenko method on measured data. Brackenhoff’s method, for example, makes use of the fact that $G^-(x, x_F, t)$ is identically zero for a point $x_F$ below the deepest reflector.

The following will provide step-by-step directions how to compute the reflection response with accurate amplitudes:

- The reflection data must be deconvolved for the wavelet (Mildner et al., 2017). The result of this deconvolution is the reflection response of a zero-phase wavelet with a flat spectrum between the frequencies $f_{\min}$ and $f_{\max}$. Because we are computing the reflection response, we can avoid deconvolution and directly models the reflection response with...
The discrete intervals according to the data is postponed with 0.3 s (parameter setting rec_delay = 0.3).

In the finite-difference modeling of the reflection response, the provided program makewave can generate these waveforms and the provided scripts give the makewave parameters used to calculate the source wavelet. Note: In the discrete implementation of the computation of the source wavelet in the frequency domain, one must not forget to multiply with the frequency interval $\Delta f$, when going from frequency to time with the Fourier transform. The source wavelet used in the examples is shown in Figure 3. A shift of 0.3 s (the parameter setting $t_0=0.3$ in makewave) is added to the source wavelet to make it causal and suitable to use in the finite-difference program.

In the finite-difference modeling of the reflection response, the recording of the data is postponed with 0.3 s (parameter setting rec_delay = 0.3 in fdelmodc) to set the peak of the wavelet back at the correct time.

1. In the finite-difference program for modeling $R(x, x, t)$, an $F_z$ source of vertical force is chosen (see the manual of the finite-difference modeling program fdelmodc for an explanation about the options). The receivers are placed at the same surface as the source and measure the pressure field.
2. The amplitude scaling factor, in the finite-difference scheme for an $F_z$ source with time signature $s(t)$, is defined in the update of particle velocity $V_z$ as

$$V_z(x, z, t + \Delta t) = V_z(x, z, t) - \frac{\Delta P(x, z, t)}{\rho \Delta z} + \frac{\Delta t}{\rho \Delta x^2} s(t).$$

The discrete intervals $\Delta t, \Delta x = \Delta z$ are the steps in the finite-difference program, and $\rho$ is the local density at the injection grid point of the source. The term $\Delta P/\Delta z$ is a fourth-order finite-difference implementation of the first derivative to $z$ of pressure field $P$.

1. To compute $R$, from Green’s functions calculated by the finite-difference program, only a factor of $-2$ is needed (equation 10 in Wapenaar et al., 2012). This factor $-2$ is included in the marchenko program when it reads in the reflection response $R$.
2. The time convolution of $R$ is implemented by a forward Fourier transformation from the time to the frequency domain, multiplication in the frequency domain, and back-transformation to the time domain. In the numerical implementation, the multiplication with $\Delta t$, for the convolution in time and with $\Delta x$ for the integration over $x$, must be included as well. Together with the standard scaling factor of $1/N$ for discrete Fourier transformations when going from the time to the frequency domain and back to the time domain, with $N$ the number of time samples, the scale factor to compute the time convolution and space integration in the frequency domain becomes

$$\Delta x \Delta t / N.$$

### Building up the Green’s function

The Marchenko algorithm is illustrated with a 2D horizontally layered model as shown in Figure 4. The numerical modeling is carried out with a finite-difference modeling program (Thorbecke and Draganov, 2011) that is also included in the software package. The input source signature used to model the reflection response $R(x, x, t)$ is approximately a sinc function with a flat spectrum of amplitude 1, as shown in Figure 3.

The full reflection matrix $R(x, x, t)$, for a fixed-spread geometry, can be constructed from one forward-modeled shot (Figure 4c) because the model contains no lateral variations. The constructed fixed-spread geometry ranges from $-2250$ to $2250$ m with a 5 m distance between the source and receiver positions. The 5 m distance is chosen to avoid spatial aliasing. We use a laterally invariant medium because the time to compute the reflection response $R$ in a laterally variant medium is too large to be practical for the desired reproducibility of the examples in this paper. However, the Marchenko method does not make any assumption about the medium and can handle lateral variations as well. Moreover, the demo directory

![Source wavelet for modeling R.](image)

Figure 3. Source wavelet with a flat frequency spectrum between $f_{\text{min}}(=5\text{ Hz})$ and $f_{\text{max}}(=80\text{ Hz})$ used to model the reflection response.
of the Marchenko program contains also an example with lateral variations in the model (marchenko/demo/twoD).

The transmission response, recorded at the surface for a source at a 900 m depth, is shown in Figure 4d. It has been modeled with a zero-phase Ricker source wavelet \( s(t) \) that has its peak at 25 Hz. It is important that the chosen source wavelet to model \( G_d \) be zero phase; otherwise, the time reversals applied in the algorithm would not work properly and the Marchenko scheme would not converge. It is also preferable to choose a source wavelet that decreases rapidly in time. This is to minimize the occurrence of overlap between the direct arrival and the first reflections as is assumed in equation 11. In case of an overlap, the defined window function \( (\theta, in \) equation 12) cuts through the overlapping events, and the first reflection is not retrieved correctly.

The initialization step used to compute \( f_{1,0}^- \) (equation 18) is illustrated in Figure 5. Each shot record in \( R(x_R, x, t) \) is convolved with \( G_d(x_R, x_F, -t) \), where \( G_d(x_R, x_F, -t) \) shown in Figure 5 only contains the time reversal of the full transmission response as shown in Figure 4d. By making use of shift invariance \( R(x_R, x, t) = R(x_R - x, 0, t) \), the time-convolution result is integrated (summed over all receiver positions \( x_F \)) and results in one trace at the \( x \) position in the \( N_0 \) panel.

In \( -N_0(x, x_F, -t) \), the dotted lines indicate the cutoff boundaries of the implemented time window \( \theta(x, x_F, t) \). To suppress wrap-around events (from positive times wrapping to negative times), the time window \( \theta(x, x_F, t) \) as introduced in equation 12, is symmetrized. Hence, from here onward \( \theta(x, x_F, t) \) is zero for \( t > t^*_d \) and \( t < -t^*_d \), and unity for times inside \( -t^*_d < t < t^*_d \). For deep focal points, one can also extend the time axis by padding zeros at the end of the array and in that way avoid the influence of wrap-around events in the time domain. In Appendix A, the treatment of time wrap-around is explained in more detail.

The events before the top dotted line and the events after the bottom dotted line are muted. The two remaining events originate from the two reflectors above the chosen focal point at a 900 m depth. A detailed explanation of the different events in the focusing functions is given by van der Neut et al. (2015b). Staring et al. (2016) give a similar explanation in case free-surface multiples are included in the Marchenko method. This initialization of \( f_{1,1}^+ \) is the input of the next step to compute a first estimate of \( f_{1,1}^- \), given in equations 19 and 20.

The computation of \( f_{1,1}^- \) involves the same time convolution and spatial integration operation, and it is illustrated in Figure 6. The result of the integration and convolution; \( -N_1(x, x_F, -t) \) is, according to equation 20, time reversed, multiplied by \(-1\) and added to \( G_d(x, x_F, -t) \) to get the first estimate \( f_{1,1}^- \). Note, that the lower (causal) part of the time window \( \theta(x, x_F, t) \) mutes also the event at direct arrival time. This event at the direct arrival time \( t_d \) will end up in the update of the Green’s function and will adjust the amplitude of the direct arrival in the Green’s function (van der Neut et al., 2015b). This update of the direct arrival in the Green’s function is explained in more detail below.

Figure 7 shows the results of the first four iterations of the Marchenko method. The first column represents the results of each convolution and integration of the focusing update term \( N_i \) with \( R \). From these figure parts (all with the same clipping factor) one can observe that the amplitude of \( N_i \) becomes smaller with each next iteration.

![Figure 4](https://example.com/figure4.png)

Figure 4. Four-layer model with (a) velocity and (b) density contrasts. (c) A shot record, with source position \( x(x = 0, z = 0) \) and receivers at \( x_F(x = x, z = 0) \), and (d) the transmission response from a source at \( x_F(x = 0, z = 900) \). Note that the source wavelet used to compute \( R \) (c) is given in Figure 3 and \( T \) (d) is modeled with a Ricker wavelet with a peak at 25 Hz.
The trace in the fifth column is a comparison between the reference Green’s function (solid gray) and the computed Green’s function (dotted black). In these traces, one can observe (indicated with arrows) that some events are weakened by subsequent iterations: The computed Green’s function converges to the reference Green’s function.

To get a better understanding of the computation of the Green’s function, the first few iterations are discussed in more detail. The initialization of the method starts with $G_d$ (equation 27), and $G$ is computed according to equation 33. This gives

$$f_{2,0}(x_F, x_R, t) = G_d(x_F, x_F, -t)$$

$$G_0(x_R, x_F, t) = G_d(x_R, x_F, -t) + \int_{\partial \Omega_0} \int_{t'=-\infty}^{-t} R(x_R, x, t-t') \times G_d(x, x_F, -t') \, dt' \, dx + N_0(x_R, x_F, -t).$$

(39)

Note that in equation 39, the result of the first integration-convolution with $R$ is not muted with $\theta$. The initial estimate of Green’s function is thus built up of three terms:

1) the direct arrival of the transmission response ($G_d(x_R, x_F, -t)$),
2) the integration-convolution of $R$ with $G_d$, this is the (unmuted) top left panel in Figure 7, and
3) a $\theta$ muted and multiplied by $-1$ version of the integration-convolution of $R$ with $G_d$: $N_0(x_R, x_F, -t) = -f_{1,0}(x_R, x_F, t)$, the second panel in the top row of Figure 7 multiplied by $-1$.

It is important to note that the result of the combination of the second and third terms just subtracts $f_{1,0}(t)$ (the events within black-dotted) from the unmuted integration-convolution of $R$ with $G_d$. This is the same as the inverse operation of the time window $\theta$. To get the first estimate of the Green’s function, $G_d(x_R, x_F, -t)$ is added to this result and gives the top-right panel in Figure 7. In the next iteration, we have

$$f_{2,1}(x_F, x_R, t) = G_d(x_F, x_R, -t) + N_0(x_F, x_R, t)$$

$$G_1(x_F, x_R, t) = G_0(x_R, x_F, t) + \int_{\partial \Omega_0} \int_{t'=-\infty}^{-t} R(x_R, x, t-t') \times N_0(x_F, x_F, t') \, dt' \, dx + N_1(x_R, x_F, -t).$$

(40)

Compared with the previous iteration, two new terms are added:

1) the integration-convolution of $R$ with $N_0$, this is the (unmuted) left panel for $i = 1$ in Figure 7 and
2) the $\theta$ muted, time reversed, multiplied by $-1$ version of the integration-convolution of $R$ with $N_0$.

Each next iteration follows this same pattern: The events within the time window $\theta$ (above $t'$) are used to update the $f_i$ focusing

Figure 5. Initialization step to compute $f_{1,0}(x, x_F, t) = -N_0(x, x_F, -t)$. After applying the time window $\theta(x, x_F, t) = \theta$, only events between the dotted lines remain in $N_0$. The mute window at $t < 0$ is applied to mute the wrap-around events of the temporal convolution. This extra window at $t < 0$ is only a practical solution and is not needed from the theory. Note the difference in the time axes of the three panels: positive for $R(t)$, negative for $G_d(-t)$, and negative and positive for $N_0(-t)$.
function, and the events outside the time window $\theta_t$ (below $t_\epsilon$) are used to update Green’s function. Application of the window function $\theta_t$ separates the focusing function and the Green’s function.

There is one important remark to make: The direct arrival $T_{inv}^d$ in the focusing function $f_1^+$ is not updated, whereas the direct arrival $G_d$ in the Green’s function $G$ is updated. In the first iteration (top row in Figure 7), the direct arrival in the Green’s function $G_0$ is equal to $G_d$. In the second iteration shown in Figure 7, the amplitude of the direct arrival is corrected by the event in the unmuted $-N_1(-t)$ just below the black-dotted line of the mute window. This event in the unmuted $-N_1(-t)$ has an opposite sign to the direct arrival and decreases the amplitude of the direct arrival (van der Neut et al., 2015b). In the plotted trace of $G_0$, the amplitude of the direct arrival (the dotted line) is much higher than the reference (the gray line). In $G_1$, the amplitudes of the direct arrival between

![Diagram](image_url)

Figure 6. First iteration to compute $f_{1,1}^+(x, x_F, t)$ from $f_{1,0}^-(x_R, x_F, t)$. In the summation of $G_d$ with $N_1$ it is important that the amplitudes of $R$ are correct.
Figure 7. Four successive iterations of the Marchenko method. The arrows point to an event that does not belong to Green’s function, and it is weakened at each iteration. The function $f_{1,i}^-$ (the second column) only changes from $i = 1$ to $i = 2$, whereas $f_{1,i}^+$ (the third column) changes from $i = 0$ to $i = 1$ and from $i = 2$ to $i = 3$. The clip level for $N_i$ and $G_i$ is the same for all panels. Labels of the horizontal and vertical axes are the same for all panels, and they are shown for the top and left panels.
reference and computed Green’s function are already much closer. We do not expect that the scheme started with the approximation $T_d^{\text{inv}} \approx G_d$ will arrive at the correct amplitudes; to achieve accurate amplitudes, the inverse of the transmission transpose had to be used and not $G_d$. There is an offset-dependent scaling factor between the reference and the computed Green’s function. Thorbecke et al. (2013) show that this estimate of the direct arrival does not have to be precise and can be based on a macromodel. The relative amplitudes between the events of the computed Green’s function are correct and are shown in the trace comparison with the reference output in Figure 7.

The iterative corrections of the amplitude of Green’s function are needed to take into account transmission losses. The result is that the upgoing field that arrives at $t = t_d$ has an amplitude that is equal to the local reflection coefficient of depth level $z_f$ (Slob et al., 2014). In the next section, we will see how good this correction is when the $f_2$ focusing function is emitted into the medium.

Figure 8. Logarithmic convergence rate of the marchenko/demo/oneD example for 16 iterations. The bumps at the end of the curve are caused by limited aperture artifacts. These artifacts are two orders of magnitude smaller than the main events.

Figure 9. Comparison of the Marchenko computed Green’s function after eight iterations with the reference Green’s function: The solid-gray trace in the background is the reference, and the dotted-black trace is the Green’s function computed with the Marchenko method.

Broggini et al. (2014) use the energy before the direct arrival in Green’s function to define the convergence of the scheme. In the provided Marchenko program, there is no stopping criteria built in, to give the user the freedom to choose the number of iterations carried out. The energy in the focusing update term $(\sqrt{\sum \text{sn}_i^2(x,t)})$ is computed and printed for each iteration and can be monitored for convergence. In each next iteration, this energy should become smaller. The convergence rate for eight iterations is shown in Figure 8.

A comparison with the reference Green’s function and the Marchenko-computed Green’s function after eight iterations is shown in Figure 9. The difference with the reference Green’s function is negligible in the middle part of the picture around $x = 0$. A small amplitude mismatch increases slightly with the increasing offset. Closer to the edge of the acquisition ($\pm 2250$ m), the difference with the reference becomes larger because the full Fresnel zone is not included in the acquisition. The higher wavenumbers, more present at earlier times, are also not captured by the limited acquisition. Note that after approximately 1.5 s, the presence of higher wavenumbers becomes smaller, and the amplitude error at the far offsets also decreases. To suppress artifacts from limited acquisition aperture, tapers can be applied to the edges of the initial focusing operator and/or the reflection response. In our experience, these tapers have limited effects on suppressing these artifacts. Depending on the specific events at the boundaries of the model the finite aperture effect could slightly be attenuated. In some cases, the taper shifts the problem to the nontapered part adjacent to the tapered region and finite aperture artifacts remain. Another, usually smaller, amplitude mismatch is caused by the use of the time reversal of the direct arrival in the transmission response $G_d$ (equation 11) instead of the inverse.

Propagating the focusing function

One of the properties of the defined $f_2(x_F, x, t)$ focusing function in equation 6 is that it will focus at $t = 0$ at the focal point $x_F$. This property can be demonstrated by emitting $f_2(x_F, x, t)$ from $\partial D_0$ into the medium and show that it has a focus at position $x_F$ at snapshot $t = 0$ (Singh et al., 2016; Wapenaar et al., 2017). If the transmission losses in the events in $f_2$ have correctly been taken into account, then all internal multiples will be canceled at (and only at) $t = 0$. The left column of Figure 10 shows five snapshots at times $-0.30, -0.15, -0.03, -0.02$, and 0.0. The snapshot at $t = 0$ indeed shows only a focus at the focal point. In the snapshots at times $t = -0.03$ and $t = -0.02$, it is observed that events related to internal multiples have opposite amplitude and travel toward each other to cancel out at $t = 0$.

The second column of Figure 10 shows snapshots at positive times, after the wavefield has focused at $t = 0$. After $t = 0$, the focused and dimmed events separate again and continue their path.

Adding the snapshots at negative times to the corresponding snapshots at positive times defines the snapshots of the homogeneous Green’s function (Wapenaar et al., 2016) with a virtual source at $x_F$. The third column in Figure 10 shows these combined snapshots, in which the snapshots at positive and negative times are summed, and they represent the causal part of the homogeneous Green’s function. These snapshots can be interpreted as the response of a virtual source located at the position of the focal point $x_F$. 

WB40 Thorbecke et al.

Downloaded 11/02/17 to 131.180.130.242. Redistribution subject to SEG license or copyright; see Terms of Use at http://library.seg.org/
CONCLUSION

The iterative Marchenko method computes focusing functions which in turn can be used to compute upgoing and downgoing Green’s functions from a virtual source position in the subsurface. For the method to converge, the amplitudes of the (modeled) reflection response must be deconvolved for the source signature and correctly scaled. In each iteration, a time convolution and spatial integration is carried out between a focusing-update term $N_i$ and the reflection data. The result of this integration is split by a time window that is defined by the first-arrival time from the virtual source position. The events before the first arrival define at each iteration a new $N_i$, which is used in the next iteration. The events after the time window update the Green’s function. The main com-

Figure 10. Snapshots of propagation of focusing function $f_2$ through the actual medium. The left column shows snapshots at acausal times, and the middle column shows snapshots at causal times. The rightmost column shows the addition of the acausal snapshots at negative times with the corresponding causal snapshots at positive times (time $T$). Labels of the horizontal and vertical axes are the same for all panels, and they are shown for the top and left panels.
computational work in each iteration is the computation of these focusing-update terms \( N_j \). The focusing functions are updated by adding the computed \( N_j \) terms.

ACKNOWLEDGMENTS

We are grateful for the many constructive comments and suggestions that we received from C. A. da Costa Filho, two anonymous reviewers, and the associate editor F. Broggini. The reviewers helped to improve the readability of the paper and also made it better suited for an introduction to the Marchenko method.

APPENDIX A

INPUT PARAMETERS AND IMPLEMENTATION DETAILS

The provided marchenko source-code package contains two main programs:

- \texttt{fmute}: picks the first-arrival time from a transmission response and mutes along this time
- \texttt{marchenko}: solves for the focusing functions in the Marchenko method and computes the Green’s functions.

The \texttt{fmute} program tracks the first arrival from a transmission response to a focal point in the subsurface. Its main use is to separate the direct arrival of the transmission response \( G_d \) from the multiple scattering coda \( M^\prime \), a similar separation to that presented in equation 11. In the examples provided, the transmission response is also computed by finite-difference modeling and the direct arrival needs to be separated from the coda. For example, \texttt{fmute} is used to compute \( G_d(t) \) in Figure 5 from \( T(t) \) in Figure 4d. The program \texttt{fmute} is not needed if a method is used (e.g., an eikonal solver) that computes the direct arrival in a direct way. The output \( G_d \) of the \texttt{fmute} program is the input \texttt{file\_inv} of the \texttt{marchenko} program. The different parameters of \texttt{fmute} are shown in the self-documentation of the program:

\texttt{fmute - mute in time domain file\_shot along curve of maximum amplitude in file\_mute

\texttt{fmute file\_shot= file\_mute= [optional parameters]}

Required parameters:
- \texttt{file\_mute} = input file with event that defines the mute line
- \texttt{file\_shot} = input data that are muted

Optional parameters:
- \texttt{file\_out=} = output file
- \texttt{above=} = mute after(0), before(1) or around(2) the maximum times of \texttt{file\_mute} ... options
- \texttt{check=} = plots muting window on top of \texttt{file\_mute}: output file \texttt{check\_su}
- \texttt{scale=} = scale data by dividing through the maximum
- \texttt{hw=} = number of time samples to look up and down in the next trace for the maximum
- \texttt{smooth=} = number of points to smooth the mute with a cosine window
- \texttt{verbose=} = silent option; >0 display info

If \texttt{file\_mute} is not provided, \texttt{file\_shot} will be used instead to pick the first-arrival times.

The \texttt{above} option is explained in Figure A-1 and separates in different ways the direct arrival time \( t_d \) from the coda. The \texttt{above=}0 and \texttt{above=}4 options have also a truncation point at the lower end of the time axis, with the time reversal of \( t_d \), to mute wrap-around events introduced by the periodicity of the discrete Fourier transform. Note that the lower end of the time axis can also represent negative times. The \texttt{above=}2 option defines a passing window around \( t_d \), and it is convenient to select the direct arrival of the transmission response in case the first arrival also contains head waves.

To find the first-arrival time in \texttt{file\_mute}, a simple tracking algorithm is implemented. At the trace position equal to the source

![Figure A-1](Image)

Figure A-1. The different options of the above parameter in the \texttt{fmute} and \texttt{marchenko} programs, illustrated with a shot panel consisting of noise.
position, the algorithm searches for the maximum value in that trace. It is assumed that this is the first-arrival time at the source position. For complex models, this might not be true, and it is therefore always good to enable check=1 and verify in the created output file check.su if the program has tracked the correct direct arrival time. Starting at the time-sample position of the maximum ($j_{\text{max}}$) in the source trace $i$, the algorithm looks in neighboring traces ($i \pm 1$) for the maximum. It only searches for this maximum in a restricted time window. For example, the maximum in the left trace is searched in the time window $j_{\text{max}} - \text{hw} < j < j_{\text{max}} + \text{hw}$, where hw is several samples given as the input parameter. If there are head waves present, the search algorithm can lose track of the direct arrival, so it is good practice to choose a small hw (four to eight samples).

The shift option represents the $\varepsilon$ in $t_j$, and it is needed to include the width of the wavelet in the mute window. Figure A-2 shows the effect of setting a negative or positive shift to exclude or include the width of the wavelet. With the above=1 option, a positive shift will mute the direct arrival, whereas a negative shift will preserve the direct arrival.

The parameter smooth defines a transition zone (in samples) going from one to zero in the mute window. Using a few time samples (3–5) for the smooth transition zone is enough to give satisfactory results. The direction of the taper, going from zero to one, is away from $t_j$.

The marchenko program has the following parameters and options:
- **MARCHENKO** – Iterative Green’s function and focusing functions retrieval
- marchenko file_tinv= file_shot= [optional parameters]

Required parameters:
- file_tinv= ...... direct arrival from focal point: G_d
- file_shot= ...... Reflection response: R

Optional parameters:
- **INTEGRATION**
  - tap=0 .............. lateral taper focusing(1), shot (2) or both(3)
  - ntap=0 ............ number of taper points at the boundaries
  - fmin=0 ............ minimum frequency in the Fourier transform
  - fmax=70 ............ maximum frequency in the Fourier transform
- **MARCHENKO ITERATIONS**
  - niter=10 ............ number of iterations
- **MUTE-WINDOW**
  - above=0 ............ mute above(1), around(0) or below(-1) the first travel-times of file_tinv
  - shift=12 ............ number of points above(positive) / below(negative) traveltime for mute
  - hw=8 ............ window in time samples to look for the maximum in the next trace
  - smooth=5 ............ number of points to smooth mute with the cosine window

**OUTPUT DEFINITION**
- file_green=...... output file with the full Green’s function(s)
- file_gplus= ...... output file with G+ (positive)
- file_gmin= ...... output file with G- (negative)
- file_f1plus= .... output file with f1+ (positive)
- file_f1min= ...... output file with f1- (negative)
- file_f2= ........ output file with f2 (=p+ + G+)
- file_pplus= ...... output file with p+ (positive)
- file_pmin= ...... output file with p- (negative)
- file_iter= ...... output file with Ni(=G) for each iteration
- verbose=0 ...... silent option; >0 displays info.

The number of iterations required for convergence depends on the reflection strengths and number of events in the model; a complex model will need more iterations. Typically, the number of iterations is between 8 and 20. An automatic stopping criterion could be based on the energy in the focusing update $N_i$. This stopping criterion is not implemented to give the user the freedom to choose the number of iterations.

To suppress artifacts from a limited acquisition aperture, tapers can be applied to the edges of the initial focusing operator (tap=1) and/or the reflection response (tap=2). In our experience, these tapers have limited effects on suppressing the finite-acquisition-
related artifacts and tapering is usually not enabled. The mute window parameters have the same meaning as in the mutef program.

The temporal convolution of events at positive times in the focusing update term causes events in \( R \) to be shifted forward in time. Events at negative times will shift events in \( R \) backward in time. In the Marchenko method, it is important that these backward-shifted events are properly handled. For deeper focal points, some events can be shifted to negative times; see, for example, \( N_0 \) in Figure 5. By implementing the temporal convolution in the frequency domain, we make use of the periodic property of the discrete Fourier transform: Negative times wrap around to the end of the discrete time axis.

The reason to symmetrize the time window \( \theta \) is to suppress unwanted time wrap-around effects. The time-wrap-around effects can also be avoided by padding zeros to the time traces in \( R \), making the time traces \( 2N \times n \) long, where the last \( n \) samples are zeros. The parameter \( N \) will pad zeros to the time traces of \( R \). Adding extra time samples will lead to longer computation times. Therefore, we prefer to use a symmetrized time window to suppress the unwanted effects of time wrap around.

The scale parameter can be useful when the modeled data do not have the correct amplitude, and it represents the previously mentioned \( b \) scaling factor of the reflection response. The program can optionally, when the file-names file_* are defined, output results of computed Green’s and focusing functions. Defining file_iter writes for each iteration the focusing update term \( \widetilde{N}_i(t) \) (Algorithm 1) before applying the mute window. By setting the verbose option to 2, the energy of the focusing update term is printed out for each iteration and can be used to monitor the convergence of the scheme.

The code to reproduce all figures in this paper can be found in the directory marchenko/demo/oneD. The README file in that directory explains in detail how to run the scripts. A more complete description of programs to calculate a gridded model (makemod), source wavelets (makewave), and programs for basic processing steps.

**REFERENCES**


